用户指南

Amazon CodeDeploy

API 版本 2014-10-06

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

Amazon CodeDeploy: 用户指南

Copyright © 2026 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 CodeDeploy？ 1
- 的好处 Amazon CodeDeploy 2
- CodeDeploy 计算平台概述 3
- CodeDeploy 部署类型概述 7
  - 就地部署概述 8
  - 蓝绿部署概述 9
- 我们希望听到您的意见和建议 12
- 主要组件 13
  - 应用程序 13
  - 计算平台 13
  - 部署配置 14
  - 部署组 15
  - 部署类型 15
  - Deployment type（部署类型） 16
  - 修订 16
  - 服务角色 16
  - 目标修订 17
  - 其他组件 17
- 部署 17
  - Amazon Lambda 计算平台上的部署 17
  - Amazon ECS 计算平台上的部署 20
  - 在 EC2 /本地计算平台上部署 31
- 应用程序规范文件 37
  - Amazon ECS 计算平台上的 AppSpec 文件 37
  - Amazon Lambda 计算平台上的 AppSpec 文件 37
  - EC2/本地计算平台上的 AppSpec 文件 38
  - CodeDeploy 代理如何使用 AppSpec 文件 38

开始使用 39
- 步骤 1：设置 39
  - 注册获取 Amazon Web Services 账户 39
  - 保护 IAM 用户 39
  - 授予编程式访问权限 40
- 步骤 2：创建服务角色 41
  - 创建服务角色（控制台） 43

创建服务角色（CLI） ... 44
获取服务角色 ARN（控制台） ... 45
获取服务角色 ARN（CLI） ... 45
步骤 3：限制 CodeDeploy 用户的权限 ... 46
步骤 4：创建 IAM 实例配置文件 ... 49
为您的 Amazon 实例 (CLI) 创建 IAM EC2 实例配置文件 ... 50
为您的 Amazon 实例创建 IAM EC2 实例配置文件（控制台） ... 54
产品和服务集成 ... 56
与其他 Amazon 服务集成 ... 56
Amazon A EC2 uto Scaling ... 61
ELB ... 69
与合作伙伴产品和服务集成 ... 72
GitHub ... 77
来自社区的集成示例 ... 80
博客帖子 ... 80
教程 ... 82
教程：将 WordPress 部署到非 Windows 实例 ... 82
步骤 1：启动 Amazon EC2 实例 ... 83
步骤 2：配置源内容 ... 85
步骤 3：将您的应用程序上传到 Amazon S3 ... 90
步骤 4：部署应用程序 ... 94
步骤 5：更新和重新部署您的应用程序 ... 100
步骤 6：清除 ... 104
教程：将 Hello World 应用程序部署到 Windows Server 实例 ... 107
步骤 1：启动 Amazon EC2 实例 ... 108
步骤 2：配置源内容 ... 110
步骤 3：将您的应用程序上传到 Amazon S3 ... 113
步骤 4：部署应用程序 ... 117
步骤 5：更新和重新部署您的应用程序 ... 122
步骤 6：清除 ... 125
教程：将应用程序部署到本地实例 ... 128
先决条件 ... 129
步骤 1：配置本地实例 ... 129
步骤 2：创建示例应用程序修订 ... 129
步骤 3：打包您的应用程序修订并将其上传到 Amazon S3 ... 134
步骤 4：部署应用程序修订 ... 134

步骤 5：验证您的部署 134
步骤 6：清理资源 135
教程：部署到 Auto Scaling 组 137
先决条件 137
步骤 1：创建和配置 Auto Scaling 组 138
步骤 2：将应用程序部署到 Auto Scaling 组 143
步骤 3：检查结果 149
步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数量 151
步骤 5：再次检查结果 152
步骤 6：清除 154
教程：从 GitHub 部署应用程序 156
先决条件 157
步骤 1：设置 GitHub 账户 157
步骤 2：创建 GitHub 存储库 157
步骤 3：将示例应用程序上传到 GitHub 存储库 160
步骤 4：预置实例 162
步骤 5：创建应用程序和部署组 163
步骤 6：将应用程序部署到实例 165
步骤 7：监控和验证部署 168
第 8 步：清除 170
教程：将应用程序部署到 Amazon ECS 171
先决条件 173
步骤 1：更新 Amazon ECS 应用程序 174
步骤 2：创建 AppSpec 文件 175
步骤 3：使用 CodeDeploy 控制台部署应用程序 176
步骤 4：清除 180
教程：部署具有验证测试的 Amazon ECS 服务 180
先决条件 182
步骤 1：创建测试侦听器 183
步骤 2：更新 Amazon ECS 应用程序 183
步骤 3：创建生命周期挂钩 Lambda 函数 183
第 4 步：更新您的 AppSpec文件 186
步骤 5：使用 CodeDeploy 控制台部署您的 Amazon ECS 服务 187
步骤 6：在日志中查看您的 Lambda 挂钩函数输出 CloudWatch 189
步骤 7：清除 190
教程：使用 SAM 部署 Lambda 函数 Amazon 191

先决条件 ... 192
步骤 1：设置基础设施 ... 192
步骤 2：更新 Lambda 函数 ... 206
步骤 3：部署更新的 Lambda 函数 ... 209
步骤 4：查看部署结果 ... 211
第 5 步：清理 ... 214
与 CodeDeploy 代理合作 ... 215
CodeDeploy 代理支持的操作系统 ... 215
支持的亚马逊 EC2 AMI 操作系统 ... 215
支持的本地操作系统 ... 216
CodeDeploy 代理的通信协议和端口 ... 216
CodeDeploy代理的版本历史记录 ... 216
管理 CodeDeploy 流程 ... 228
应用程序修订和日志文件清理 ... 229
CodeDeploy 代理安装的文件 ... 229
管理 CodeDeploy 代理操作 ... 232
验证 CodeDeploy 代理是否正在运行 ... 232
确定 CodeDeploy 代理的版本 ... 234
安装 CodeDeploy 代理 ... 236
更新 CodeDeploy 代理 ... 247
卸载 CodeDeploy 代理 ... 250
将 CodeDeploy 代理日志发送到 CloudWatch ... 251
使用实例 ... 256
将 Amazon EC2 实例与本地实例进行比较 ... 256
CodeDeploy 的实例任务 ... 257
为实例添加标签以进行 CodeDeploy 部署 ... 258
示例 1：唯一标签组，唯一标签 ... 259
示例 2：唯一标签组，多个标签 ... 260
示例 3：多个标签组，单个标签 ... 262
示例 4：多个标签组，多个标签 ... 264
使用亚马逊 EC2 实例 ... 268
为 CodeDeploy 创建 Amazon EC2 实例 ... 268
创建 Amazon EC2 实例（Amazon CloudFormation 模板） ... 274
配置 Amazon EC2 实例 ... 279
使用本地实例 ... 282
配置本地实例的先决条件 ... 283

注册本地实例 284
管理本地实例操作 314
查看实例详细信息 320
查看实例详细信息（控制台） 320
查看实例详细信息（CLI） 321
实例运行状况 322
运行状况 322
关于最小运行正常的实例数 323
关于每个可用区最小运行正常的实例数 326
使用部署配置 329
EC2/本地计算平台上的部署配置 329
预定义的部署配置 329
Amazon ECS 计算平台上的部署配置 333
Amazon ECS 的预定义部署配置 333
部署的 Amazon CloudFormation blue/green 部署配置 (Amazon ECS) 334
Amazon Lambda 计算平台上的部署配置 334
Lambda 的预定义部署配置 334
创建部署配置 335
创建部署配置（控制台） 336
创建部署配置（Amazon CLI） 338
查看部署配置详细信息 339
查看部署配置详细信息（控制台） 339
查看部署配置（CLI） 340
删除部署配置 340
使用 应用程序 341
创建 应用程序 342
为就地部署创建应用程序（控制台） 343
为蓝绿部署创建应用程序（控制台） 346
为 Amazon ECS 服务部署创建应用程序（控制台） 349
为 Amazon Lambda 函数部署创建应用程序（控制台） 351
创建应用程序（CLI） 353
查看应用程序详细信息 353
查看应用程序详细信息（控制台） 353
查看应用程序详细信息（CLI） 354
创建通知规则 354
重命名应用程序 357

删除应用程序 357
删除应用程序（控制台） 357
删除应用程序（Amazon CLI） 358
使用部署组 359
Amazon ECS 计算平台部署中的部署组 359
Amazon Lambda 计算平台部署中的部署组 359
EC2/本地计算平台部署中的部署组 359
创建部署组 360
为就地部署创建部署组（控制台） 361
为 EC2 /本地 blue/green 部署创建部署组（控制台） 364
为 Amazon ECS 部署创建部署组（控制台） 367
在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 369
为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器 370
创建部署组（CLI） 374
查看部署组详细信息 375
查看部署组详细信息（控制台） 375
查看部署组详细信息（CLI） 376
更改部署组设置 376
更改部署组设置（控制台） 377
更改部署组设置（CLI） 378
为部署组配置高级选项 378
删除部署组 381
删除部署组（控制台） 382
删除部署组（CLI） 382
使用应用程序修订 383
规划修订 383
添加 AppSpec 文件 384
为 Amazon ECS 部署添加 AppSpec 文件 384
为 Amazon Lambda 部署添加 AppSpec 文件 387
为 EC2/本地部署添加 AppSpec 文件 389
选择存储库类型 393
推送修订 395
使用推送修订版 Amazon CLI 397
查看应用程序修订详细信息 399
查看应用程序修订详细信息（控制台） 399
查看应用程序修订详细信息（CLI） 400

注册应用程序修订 401
使用 CodeDeploy 在 Amazon S3 中注册修订（CLI） 401
使用 CodeDeploy 在 GitHub 中注册修订（CLI） 402
使用部署 403
创建 部署。 404
部署先决条件 404
创建 Amazon ECS 计算平台部署（控制台） 407
创建 Amazon Lambda 计算平台部署（控制台） 409
创建 EC2 /本地计算平台部署（控制台） 410
创建 Amazon ECS 计算平台部署（CLI） 414
创建 Amazon Lambda 计算平台部署（CLI） 415
创建 EC2 /本地计算平台部署 (CLI) 417
通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation 420
查看部署详细信息 423
查看部署详细信息（控制台） 424
查看部署详细信息（CLI） 424
查看部署日志数据 425
在 Amazon CloudWatch 控制台中查看日志文件数据 425
查看实例上的日志文件 426
停止部署 428
停止部署（控制台） 429
停止部署（CLI） 430
重新部署和回滚部署 430
自动回滚 430
手动回滚 430
回滚和重新部署工作流程 431
现有内容的回滚行为 432
在其他 Amazon 账户中部署应用程序 434
步骤 1：在任一账户中创建 S3 存储桶 434
步骤 2：将 Amazon S3 存储桶权限授予生产账户的 IAM 实例配置文件 435
步骤 3：在生产账户中创建资源和跨账户角色 436
步骤 4：将应用程序修订上传到 Amazon S3 存储桶 437
步骤 5：代入跨账户角色和部署应用程序 437
验证本地机器上的部署程序包 438
先决条件 438
创建本地部署 440

示例 ... 443
监控部署 ... 445
自动化工具 ... 445
手动工具 ... 446
使用 Amazon CloudWatch 工具监控部署 ... 447
使用 CloudWatch 警报监控部署 ... 448
使用 Amazon CloudWatch 事件监控部署 ... 449
使用 Amazon CloudTrail 监控部署 ... 451
CloudTrail 中的 CodEDEPloy 信息 ... 452
了解 CodeDeploy 日志文件条目 ... 452
使用 Amazon SNS 事件通知监控部署 ... 454
向服务角色授予 Amazon SNS 权限 ... 455
为 CodeDeploy 事件创建触发器 ... 456
在部署组中编辑触发器 ... 462
从部署组中删除触发器 ... 464
触发器的 JSON 数据格式 ... 465
安全性 ... 467
数据保护 ... 467
互联网络流量隐私 ... 468
静态加密 ... 469
传输中加密 ... 469
加密密钥管理 ... 469
Identity and access management ... 469
受众 ... 470
使用身份进行身份验证 ... 470
使用策略管理访问 ... 471
如何 Amazon CodeDeploy 与 IAM 配合使用 ... 472
Amazon 的托管（预定义）策略 CodeDeploy ... 476
CodeDeploy Amazon 托管策略的更新 ... 484
基于身份的策略示例 ... 487
问题排查 ... 493
CodeDeploy 权限参考 ... 495
防止跨服务混淆座席 ... 503
事件响应 ... 505
审核与的所有互动 CodeDeploy ... 505
提醒和事件管理 ... 505

合规性验证 ... 506
恢复能力 ... 506
基础结构安全性 ... 506
参考 ... 508
AppSpec 文件参考 ... 508
AppSpec 亚马逊 ECS 计算平台上的文件 ... 509
AppSpec Amazon Lambda 计算平台上的文件 ... 509
AppSpec EC2/本地计算平台上的文件 ... 509
AppSpec 文件结构 ... 510
AppSpec 文件示例 ... 551
AppSpec 文件间距 ... 557
验证您的 AppSpec 文件和文件位置 ... 558
代理配置参考 ... 559
相关主题 ... 562
Amazon CloudFormation 模板参考 ... 563
将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用 ... 565
可用性 ... 566
为 CodeDeploy 创建 VPC 端点 ... 569
配置 CodeDeploy 代理和 IAM 权限 ... 569
资源工具包参考 ... 570
各区域的资源工具包存储桶名称 ... 570
资源工具包内容 ... 571
显示资源工具包文件列表 ... 572
下载资源工具包文件 ... 573
配额 ... 573
问题排查 ... 578
一般问题排查 ... 578
一般问题排查核对清单 ... 579
CodeDeploy 只有某些 Amazon 区域支持部署资源 ... 580
本指南中的步骤与 CodeDeploy 控制台不匹配 ... 580
所需的 IAM 角色不可用 ... 581
使用某些文本编辑器创建 AppSpec文件和 shell 脚本可能会导致部署失败 ... 581
使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 ... 581
排除 EC2 /本地部署问题 ... 582
CodeDeploy 插件 CommandPoller缺少凭据错误 ... 583
部署失败并显示消息 “ PKCS7 已签名邮件验证失败” ... 583

将相同的文件部署或重新部署到相同的实例位置失败，出现错误“The deployment failed because a specified file already exists at this location” 583
长文件路径会导致“没有这样的文件或目录”错误 585
长时间运行的进程可能会导致部署失败 586
在部署日志中未报告任何错误的情况下对失败的 AllowTraffic 生命周期事件进行故障排除 587
对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除 588
使用以下命令对失败的 DownloadBundle 部署生命周期事件进行故障排除 UnknownError：未打开供读取 589
排查所有生命周期事件跳过错误 589
默认情况下，Windows PowerShell 脚本无法使用 64 位版本 PowerShell 的 Windows 591
排查 Amazon ECS 部署问题 592
等待替换任务集时出现超时 592
等待通知继续时出现超时 593
IAM 角色没有足够的权限 593
等待状态回调时部署超时 594
由于一个或多个生命周期事件验证函数失败，部署失败 594
由于以下错误，ELB 无法更新：主任务集目标组必须位于监听器之后 595
使用 Auto Scaling 时，我的部署有时会失败 595
只有 ALB 支持渐进式流量路由，在 create/update 部署组时改用 AllAtOnce 流量路由 596
尽管我的部署成功了，但替换任务集未通过 ELB 运行状况检查，我的应用程序也已关闭 597
我能否将多个负载均衡器连接到一个部署组？ 597
我能否在没有负载均衡器的情况下执行 CodeDeploy 蓝/绿部署？ 598
在部署过程中，如何使用新信息更新我的 Amazon ECS 服务？ 598
对 Amazon Lambda 部署问题进行故障排除 598
Amazon Lambda 手动停止未配置回滚的 Lambda 部署后，部署失败 598
排查部署组问题 599
标记作为部署组的一部分的实例不会自动将您的应用程序部署到新实例 599
排查实例问题 599
必须正确设置标签 599
Amazon CodeDeploy 必须在实例上安装并运行代理 600
如果实例在部署期间终止，在最多 1 小时内部署不会失败。 600
分析日志文件以调查针对实例的部署失败 600
如果 CodeDeploy 日志文件被意外删除，请创建一个新的日志文件 600
疑难解答 “InvalidSignatureException — 签名已过期：[时间] 现在早于 [时间]” 部署错误 601
解决 GitHub 令牌问题 601

GitHub OAuth 令牌无效 601
已超过最大代 GitHub OAuth币数量 602
解决 Amazon A EC2 uto Scaling 问题 602
Amazon A EC2 uto Scaling 常规疑难解答 602
“CodeDeployRole 未授予您在以下 Amazon 服务中执行操作的权限： AmazonAutoScaling” 错误 603
在部署修订版之前，Amazon A EC2 uto Scaling 组中的实例会持续预配置和终止 604
终止或重启 Amazon A EC2 uto Scaling 实例可能会导致部署失败 604
避免将多个部署组与单个 Amazon A EC2 uto Scaling 组相关联 605
EC2 Amazon A EC2 uto Scaling 组中的实例无法启动并收到 “心跳超时” 错误 606
不匹配的 Amazon A EC2 uto Scaling 生命周期挂钩可能会导致对 Amazon A EC2 uto Scaling 群组的自动部署停止或失败 608
“由于未找到您的部署组的实例，部署失败”错误 609
错误代码 616
相关主题 619
资源 620
参考指南和支持资源 620
样本 620
博客 620
Amazon 软件开发工具包和工具 620
文档历史记录 622
早期更新 634
Amazon 术语表 649
..... dcl

什么是 CodeDeploy？

CodeDeploy 是一项部署服务，可自动将应用程序部署到亚马逊 EC2 实例、本地实例、无服务器 Lambda 函数或 Amazon ECS 服务。

您可以部署几乎无限种类的应用程序内容，包括：

- 代码
- 无服务器函数 Amazon Lambda
- Web 和配置文件
- 可执行文件
- 软件包
- 脚本
- 多媒体文件

CodeDeploy 可以部署在服务器上运行并存储在 Amazon S3 存储桶、存储库或 Bitbucket GitHub 存储库中的应用程序内容。 CodeDeploy 也可以部署无服务器 Lambda 函数。您无需更改现有代码即可开始使用 CodeDeploy。

CodeDeploy 让你更轻松：

- 快速发布新功能。
- 更新 Amazon Lambda 函数版本。
- 避免在应用程序配置过程中停机。
- 处理更新应用程序的复杂性，而没有许多与容易出错的手动部署关联的风险。

该服务会随您的基础设施进行扩展，因此您可以轻松地向一个实例或数千个实例部署。

CodeDeploy 可与各种系统配合使用，用于配置管理、源代码控制、持续集成、持续交付和持续部署。有关更多信息，请参阅产品集成。

CodeDeploy 控制台还提供了一种快速搜索资源的方法，例如存储库、生成项目、部署应用程序和管道。选择转到资源或按下 / 键，然后键入资源的名称。任何匹配结果都会显示在列表中。搜索不区分大小写。您只能看到您有权查看的资源。有关更多信息，请参阅 对 Amazon CodeDeploy进行身份和访问管理。

主题

- 的好处 Amazon CodeDeploy
- CodeDeploy 计算平台概述
- CodeDeploy 部署类型概述
- 我们希望听到您的意见和建议
- CodeDeploy 主要组件
- CodeDeploy 部署
- CodeDeploy 应用程序规范（AppSpec）文件

的好处 Amazon CodeDeploy

CodeDeploy 提供以下好处：

- 服务器、无服务器和容器应用程序。 CodeDeploy允许您在服务器上部署传统应用程序和部署无服务器 Amazon Lambda 功能版本或 Amazon ECS 应用程序的应用程序。
- 自动部署。 CodeDeploy 在开发、测试和生产环境中实现应用程序部署的完全自动化。 CodeDeploy 可根据您的基础架构进行扩展，以便您可以部署到一个或数千个实例。
- 最大程度减少停机时间。如果您的应用程序使用 EC2 /Unlide 计算平台，则 CodeDeploy 有助于最大限度地提高应用程序的可用性。在就地部署期间， CodeDeploy 对 Amazon EC2 实例执行滚动更新。您可以指定在进行更新时每次进入脱机状态的实例的数量。在 blue/green 部署期间，将在替换实例上安装最新的应用程序修订。在您选择时，流量会立即重新路由到这些实例，或者在完成新环境测试之后立即重新路由。对于两种部署类型， CodeDeploy 将根据您配置的规则跟踪应用程序运行状况。
- 停止并回滚。出现错误时，您可以自动或手动停止和回滚部署。
- 集中控制。您可以通过 CodeDeploy 控制台或启动部署并跟踪部署状态 Amazon CLI。您会收到一份报告，其中列出了每个应用程序修订的部署时间以及部署到哪些 Amazon EC2 实例。
- 易于采用。 CodeDeploy 不受平台限制，适用于任何应用程序。您可以轻松地重复使用您的设置代码。 CodeDeploy 还可以与您的软件发布流程或持续交付工具链集成。
- 并发部署。如果您有多个应用程序使用 EC2 /Unlide 计算平台，则 CodeDeploy 可以将它们同时部署到同一组实例。

CodeDeploy 计算平台概述

CodeDeploy 能够将应用程序部署到三个计算平台：

- EC2/本地：描述物理服务器的实例，这些实例可以是 Amazon EC2 云实例、本地服务器或两者兼而有之。使用 EC2 /Londest 计算平台创建的应用程序可以由可执行文件、配置文件、图像等组成。

  使用 EC2 /Onlide 计算平台的部署使用就地部署或蓝/绿部署类型来管理流量定向到实例的方式。有关更多信息，请参阅 CodeDeploy 部署类型概述。
- Amazon Lambda：用于部署由 Lambda 函数的更新版本组成的应用程序。 Amazon Lambda 在由高可用性计算结构组成的无服务器计算环境中管理 Lambda 函数。计算资源的所有管理均由执行 Amazon Lambda。有关更多信息，请参阅无服务器计算和应用程序。有关 Amazon Lambda 和 Lambda 函数的更多信息，请参阅。Amazon Lambda

  您可以通过选择金丝雀、线性或 all-at-once配置来管理部署期间流量转移到更新后的 Lambda 函数版本的方式。
- Amazon ECS：用于将 Amazon ECS 容器化应用程序部署为任务集。 CodeDeploy 通过将应用程序的更新版本安装为新的替换任务集来执行 blue/green 部署。 CodeDeploy 将生产流量从原始应用程序任务集重新路由到替换任务集。成功部署后，将会终止原始任务集。有关 Amazon ECS 的更多信息，请参阅 Amazon Elastic Container Service。

  通过选择金丝雀、线性或 all-at-once配置，您可以管理在部署期间将流量转移到更新的任务集的方式。

> **Note**
>
> 同时使用 CodeDeploy 和支持 Amazon ECS blue/green 部署 Amazon CloudFormation。这些部署的详细信息将在后续章节中介绍。

下表描述了如何在每个计算平台上使用 CodeDeploy 组件。有关更多信息，请参阅:

- 在中使用部署组 CodeDeploy
- 在中处理部署 CodeDeploy
- 在中使用部署配置 CodeDeploy
- 正在处理的应用程序修订版 CodeDeploy
- 在 CodeDeploy 中使用应用程序

CodeDeploy 组件	EC2/本地	Amazon Lambda	Amazon ECS
部署组	将修订部署到一组实例。	将无服务器 Lambda 函数的一个新版本部署到高可用性计算基础设施。	指定带有容器化应用程序的 Amazon ECS 服务以部署为任务集，同时指定用于为部署的应用程序提供流量的生产和可选测试侦听器，何时重新路由流量并终止已部署应用程序的原始任务集，以及可选的触发器、警报和回滚设置。
部署	部署由应用程序和 AppSpec 文件组成的新修订版。 AppSpec 指定如何将应用程序部署到部署组中的实例。	将生产流量从 Lambda 函数的一个版本转移到同一函数的新版本。该 AppSpec 文件指定要部署的 Lambda 函数版本。	将 Amazon ECS 容器化应用程序的更新版本部署为新的替代任务集。CodeDeploy 将生产流量从具有原始版本的任

CodeDeploy 组件	EC2/本地	Amazon Lambda	Amazon ECS
			务集重新路由到具有更新版本的新替换任务集。在部署完成后，会终止原始任务集。
部署配置	此设置确定部署速度以及在部署过程中任何时候都必须正常的最小实例数。	此设置确定流量如何转移到更新后的 Lambda 函数版本。	此设置确定流量如何转移到更新后的 Amazon ECS 任务集。

CodeDeploy 组件	EC2/本地	Amazon Lambda	Amazon ECS
修订	AppSpec 文件和应用程序文件的组合，例如可执行文件、配置文件等。	一个 AppSpec 文件，它指定要部署哪个 Lambda 函数以及可以在部署生命周期事件挂钩期间运行验证测试的 Lambda 函数。	指定以下 AppSpec 内容的文件： • Amazon ECS 服务的 Amazon ECS 任务定义，以及要部署的容器化应用程序。 • 部署更新的应用程序的容器。 • 重新路由生产流量的容器端口。 • 可选的网络配置设置以及在部署生命周期事件挂钩期间可运行验证测试的 Lambda 函数。

CodeDeploy 组件	EC2/本地	Amazon Lambda	Amazon ECS
应用程序	部署组和修订的集合。EC2/本地应用程序使用 EC2 /本地部署计算平台。	部署组和修订的集合。用于 Amazon Lambda 部署的应用程序使用无服务器 Lambd Amazon a 计算平台。	部署组和修订的集合。用于 Amazon ECS 部署的应用程序使用 Amazon ECS 计算平台。

CodeDeploy 部署类型概述

CodeDeploy 提供了两个部署类型选项：

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2 /Unlide 计算平台的部署才能使用就地部署。有关就地部署的更多信息，请参阅就地部署概述。

  > **Note**
  >
  > Amazon Lambda 和 Amazon ECS 部署不能使用就地部署类型。

- 蓝绿部署：部署的行为取决于使用的计算平台：
  - Blue/green on an EC2/On-本地计算平台：使用以下步骤将部署组（原始环境）中的实例替换为另一组实例（替换环境）：
    - 为替换环境配置实例。
    - 在替换实例上安装最新的应用程序修订。
    - 对于应用程序测试和系统验证等活动，可以选择等待时间。
    - 替换环境中的实例在一个或多个 ELB 负载均衡器中注册，从而导致流量被重新路由到这些负载均衡器。原始环境中的实例已注销，可以终止或继续运行以用于其他用途。

> **Note**
>
> 如果您使用 EC2 /本地计算平台，请注意 blue/green 部署仅适用于 Amazon EC2 实例。

- 或 Amazon Lambda Amazon ECS 计算平台上的蓝/绿：流量根据金丝雀、线性或all-at-once部署配置逐渐移动。
- 蓝/绿部署通过 Amazon CloudFormation：作为 Amazon CloudFormation 堆栈更新的一部分，流量将从您当前的资源转移到更新的资源。当前，仅支持 ECS blue/green 部署。

有关 blue/green 部署的更多信息，请参阅蓝绿部署概述。

> **Note**
>
> 使用该 CodeDeploy 代理，您无需应用程序、部署组甚至 Amazon 帐户，即可在已登录的实例上执行部署。有关信息，请参阅使用 CodeDeploy 代理验证本地机器上的部署程序包。

主题

- 就地部署概述
- 蓝绿部署概述

就地部署概述

> **Note**
>
> Amazon Lambda 和 Amazon ECS 部署不能使用就地部署类型。

以下是就地部署的工作原理：

1. 首先，在本地开发计算机或类似环境中创建可部署的内容，然后添加应用程序规范文件（AppSpec 文件）。该 AppSpec 文件是唯一的 CodeDeploy。它定义了您 CodeDeploy 要执行的部署操作。您可以将可部署内容和文件捆绑到存档 AppSpec 文件中，然后将其上传到 Amazon S3 存储桶或 GitHub存储库。此存档文件称为应用程序修订（简称修订）。
2. 接下来，您将 CodeDeploy 提供有关您的部署的信息，例如从哪个 Amazon S3 存储桶或 GitHub 存储库提取修订以及将其内容部署到哪组 Amazon EC2 实例。 CodeDeploy 将一组 Amazon EC2 实

例称为部署组。部署组包含单独标记的亚马逊 EC2 实例、Amazon A EC2 uto Scaling 组中的亚马逊 EC2 实例，或者两者兼而有之。

每次您成功上传要部署到部署组的新应用程序修订时，该捆绑包就会设置为部署组的目标修订。也就是说，当前设为部署目标的应用程序修订为目标修订。这也是为自动部署提取的修订。

3. 接下来，每个实例上的 CodeDeploy 代理都会轮询 CodeDeploy 以确定何时从指定的 Amazon S3 存储桶或存储库中提取内容和 GitHub 存储库。
4. 最后，每个实例上的 CodeDeploy 代理从 Amazon S3 存储桶或存储 GitHub 库中提取目标修订版，然后按照 AppSpec 文件中的说明将内容部署到该实例。

CodeDeploy 保留部署记录，以便您可以获取部署状态、部署配置参数、实例运行状况等。

蓝绿部署概述

blue/green 部署用于更新您的应用程序，同时最大限度地减少因更改新应用程序版本而造成的中断。CodeDeploy 在重新路由生产流量之前，请将新的应用程序版本与旧版本一起配置。

- Amazon Lambda：流量从 Lambda 函数的一个版本转移到同一 Lambda 函数的新版本。
- Amazon ECS：流量从 Amazon ECS 服务中的任务集转移到同一 Amazon ECS 服务中更新的替换任务集。
- EC2/On dless：流量从原始环境中的一组实例转移到一组替换的实例。

与就地部署相比，所有 Amazon Lambda 和 Amazon ECS blue/green. An EC2/On-Premises deployment can be in-place or blue/green. A blue/green 部署均为部署具有许多优势：

- 您可以在新的替换环境中安装和测试应用程序，只需通过重新路由流量即可将应用程序部署到生产环境中。
- 如果您使用的是 EC2 /Unlide 计算平台，则切换回应用程序的最新版本会更快、更可靠。这是因为只要原始实例没有被终止，流量就可以路由回原始实例。而在就地部署中，必须通过重新部署上一个版本的应用程序来回滚版本。
- 如果您使用的是 EC2 /Unlide 计算平台，则会为 blue/green 部署配置新实例，并反映大多数 up-to-date服务器配置。这将帮助您避免在长时间运行的实例上有时出现的问题类型。
- 如果您使用的是 Amazon Lambda 计算平台，则可以控制流量如何从原始 Lambda 函数版本转移到新 Amazon Lambda Amazon 函数版本。
- 如果您使用的是 Amazon ECS 计算平台，则可以控制流量从原始任务集转移到新任务集的方式。

使用进行 blue/green 部署 Amazon CloudFormation 可以使用以下方法之一：

- Amazon CloudFormation 部署模板：使用 Amazon CloudFormation 模板配置部署时，您的部署由 Amazon CloudFormation 更新触发。当您更改资源并上传模板更改时，中的堆栈更新 Amazon CloudFormation 会启动新的部署。有关可在 Amazon CloudFormation 模板中使用的资源列表，请参阅Amazon CloudFormation 模板供 CodeDeploy参考。
- 通过 Amazon CloudFormation以下方式进行蓝/绿部署：您可以使用堆栈更新 Amazon CloudFormation 来管理 blue/green 部署。除了指定流量路由和稳定设置外，您还可以在堆栈模板中定义蓝绿资源。然后，如果您在堆栈更新期间更新所选资源，则 Amazon CloudFormation 会生成所有必要的绿色资源，根据指定的流量路由参数转移流量，然后删除蓝色资源。有关更多信息，请参阅Amazon CloudFormation 用户指南 Amazon CloudFormation中的 CodeDeploy 使用自动 blue/green 部署 Amazon ECS。

> **Note**
>
> 仅支持 Amazon ECS blue/green 部署。

如何配置 blue/green 部署取决于您的部署使用的计算平台。

在或 Amazon Lambda Amazon ECS 计算平台上部署蓝/绿

如果您使用的是 Amazon Lambda 或 Amazon ECS 计算平台，则必须说明流量是如何从原始 Amazon Lambda 函数或 Amazon ECS 任务集转移到新函数或任务集的。要指示流量是如何转移的，您必须指定下列部署配置之一：

- 金丝雀
- 线性
- all-at-once

有关在金丝雀配置、线性配置或 all-at-once部署配置中流量如何转移的信息，请参阅部署配置。

有关 Lambda 部署配置的详细信息，请参阅Amazon Lambda 计算平台上的部署配置。

有关 Amazon ECS 部署配置的详细信息，请参阅Amazon ECS 计算平台上的部署配置。

Blue/Green deployment on an EC2/on-本地计算平台

> **Note**
> 您必须使用 Amazon EC2 实例在 EC2 /本地计算平台上进行 blue/green 部署。该 blue/green 部署类型不支持本地实例。

如果您使用的是 EC2 /Unlide 计算平台，则以下内容适用：

您必须有一个或多个带有识别亚马逊 EC2 标签的亚马逊 EC2 实例或一个 Amazon A EC2 uto Scaling 组。这些实例必须满足这些额外要求：

- 每个 Amazon EC2 实例都必须附上正确的 IAM 实例配置文件。
- 必须在每个实例上安装并运行 CodeDeploy 代理。

> **Note**
> 通常，您还会在原始环境中的实例上运行应用程序修订，但这不是 blue/green 部署的必要条件。

创建用于部署的部署组时，可以选择如何指定替换环境： blue/green

复制现有的 Amazon A EC2 uto Scaling 组：在 blue/green 部署期间，在部署期间为您的替代环境 CodeDeploy 创建实例。使用此选项， CodeDeploy 使用您指定的 Amazon A EC2 uto Scaling 组作为替换环境的模板，包括相同数量的运行实例和许多其他配置选项。

手动选择实例：您可以使用亚马逊实例标签、Amazon A EC2 uto Scaling 组名称或两者来指定要计作替换 EC2 实例的实例。如果您选择此选项，则在创建部署前无需指定替换环境的实例。

下面将介绍操作方式：

1. 您已经有实例或用作原始环境的 Amazon A EC2 uto Scaling 组。首次运行 blue/green 部署时，通常使用已在就地部署中使用的实例。
2. 在现有 CodeDeploy 应用程序中，您可以创建一个 blue/green 部署组，除了就地部署所需的选项外，您还可以在其中指定以下内容：
   - 一种或多个负载均衡器，用于在 blue/green 部署过程中将流量从原始环境路由到替代环境。

- 立即将流量重新路由到替换环境还是等待您手动路由。
- 流量路由到替换实例的速率。
- 被替换的实例是终止还是继续运行。

3. 您为此部署组创建一个部署，在此期间，将会发生如下情况：
   a. 如果您选择复制 Amazon A EC2 uto Scaling 组，则会为您的替代环境配置实例。
   b. 您为部署指定的应用程序修订将安装在替换实例上。
   c. 如果您在部署组设置中指定了等待时间，部署将暂停。这是您可以在替换环境中运行测试和验证的时间。如果您未在等待期结束之前手动路由流量，部署将停止。
   d. 替换环境中的实例已向 ELB 负载均衡器注册，流量开始路由到它们。
   e. 原始环境中的实例将取消注册，并根据部署组中的规范进行处理，要么终止，要么继续运行。

通过蓝/绿部署 Amazon CloudFormation

您可以使用模板对资源进行建模，从而管理 CodeDeploy 蓝/绿部署。 Amazon CloudFormation

使用 Amazon CloudFormation 模板对 blue/green 资源进行建模时，可以在中创建堆栈更新 Amazon CloudFormation 以更新您的任务集。生产流量将从服务的原始任务集转移到替换任务集，可以一次全部转移，也可以使用 Canary 部署进行转移。堆栈更新在 CodeDeploy 中启动部署。您可以在中查看部署状态和历史记录 CodeDeploy，但不能以其他方式创建或管理 Amazon CloudFormation 模板之外的 CodeDeploy 资源。

> **Note**
>
> 对于通过进行的 blue/green 部署 Amazon CloudFormation，您无需创建 CodeDeploy 应用程序或部署组。

此方法仅支持 Amazon ECS blue/green 部署。有关通过进行 blue/green 部署的更多信息 Amazon CloudFormation，请参阅通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation。

我们希望听到您的意见和建议

我们欢迎您提供反馈。要与我们联系，请访问 CodeDeploy 论坛。

主题

- Primary Components

- Deployments
- Application Specification Files

CodeDeploy 主要组件

在开始使用该服务之前，您应该熟悉 CodeDeploy 部署过程的主要组成部分。

主题

- 应用程序
- 计算平台
- 部署配置
- 部署组
- 部署类型
- IAM 实例配置文件
- 修订
- 服务角色
- 目标修订
- 其他组件

应用程序

应用程序是唯一标识要部署的应用程序的名称。 CodeDeploy 使用此名称（用作容器）来确保在部署期间引用修订版、部署配置和部署组的正确组合。

计算平台

计算平台是 CodeDeploy 部署应用程序的平台。有三个计算平台：

- EC2/本地：描述物理服务器的实例，这些实例可以是 Amazon EC2 云实例、本地服务器或两者兼而有之。使用 EC2 /Londest 计算平台创建的应用程序可以由可执行文件、配置文件、图像等组成。

  使用 EC2 /Onlide 计算平台的部署使用就地部署或蓝/绿部署类型来管理流量定向到实例的方式。有关更多信息，请参阅 CodeDeploy 部署类型概述。
- Amazon Lambda：用于部署由 Lambda 函数的更新版本组成的应用程序。 Amazon Lambda 在由高可用性计算结构组成的无服务器计算环境中管理 Lambda 函数。计算资源的所有管理均由执

行 Amazon Lambda。有关更多信息，请参阅无服务器计算和应用程序。有关 Amazon Lambda 和 Lambda 函数的更多信息，请参阅。Amazon Lambda

您可以通过选择金丝雀、线性或 all-at-once配置来管理部署期间流量转移到更新后的 Lambda 函数版本的方式。

- Amazon ECS：用于将 Amazon ECS 容器化应用程序部署为任务集。 CodeDeploy 通过将应用程序的更新版本安装为新的替换任务集来执行 blue/green 部署。 CodeDeploy 将生产流量从原始应用程序任务集重新路由到替换任务集。成功部署后，将会终止原始任务集。有关 Amazon ECS 的更多信息，请参阅 Amazon Elastic Container Service。

  通过选择金丝雀、线性或 all-at-once配置，您可以管理在部署期间将流量转移到更新的任务集的方式。

> **Note**
>
> CodeDeploy 和都支持 Amazon ECS blue/green 部署 Amazon CloudFormation。这些部署的详细信息将在后续章节中介绍。

部署配置

部署配置是部署 CodeDeploy 期间使用的一组部署规则以及部署成功和失败条件。如果您的部署使用 EC2 /Unlide 计算平台，则可以为部署指定运行正常的实例的最小数量。如果您的部署使用 Amazon Lambda 或 Amazon ECS 计算平台，则可以指定如何将流量路由到更新后的 Lambda 函数或 ECS 任务集。

有关为使用 EC2 /Londest 计算平台的部署指定最少运行正常主机数的更多信息，请参阅关于最小运行正常的实例数。

在使用 Lambda 或 ECS 计算平台的部署期间，有一些部署配置可指定流量的路由方式：

- Canary：流量将通过两次递增进行转移。您可以从预定义的金丝雀部署选项中选择，这些选项指定在第一次增量中转移到更新后的 Lambda 函数或 ECS 任务集的流量百分比以及以分钟为单位的间隔；然后指定在第二次增量中转移剩余的流量。
- 线性部署：流量使用相等的增量转移，在每次递增之间间隔的分钟数相同。您可以从预定义的线性选项中进行选择，这些选项指定在每次增量中转移的流量百分比以及每次增量之间的分钟数。
- 答 ll-at-once：所有流量将同时从原来的 Lambda 函数或 ECS 任务集转移到更新的函数或任务集。

部署组

部署组 是一组单独的实例。部署组包含单独标记的实例、Amazon A EC2 uto Scaling 组中的亚马逊 EC2 实例，或两者兼而有之。有关 Amazon EC2 实例标签的信息，请参阅使用控制台处理标签。有关本地实例的信息，请参阅Working with On-Premises Instances。有关 Amazon A EC2 uto Scaling 的信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

部署类型

部署类型 是一种用于在部署组中的实例上提供最新应用程序修订的方法。具有两种部署类型：

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2 /Unlide 计算平台的部署才能使用就地部署。有关就地部署的更多信息，请参阅就地部署概述。
- 蓝绿部署：部署的行为取决于使用的计算平台：
  - Blue/green on an EC2/On-本地计算平台：使用以下步骤将部署组（原始环境）中的实例替换为另一组实例（替换环境）：
    - 为替换环境配置实例。
    - 在替换实例上安装最新的应用程序修订。
    - 对于应用程序测试和系统验证等活动，可以选择等待时间。
    - 替换环境中的实例在一个或多个 ELB 负载均衡器中注册，从而导致流量被重新路由到这些负载均衡器。原始环境中的实例已注销，可以终止或继续运行以用于其他用途。

    > **Note**
    > 如果您使用 EC2 /本地计算平台，请注意 blue/green 部署仅适用于 Amazon EC2 实例。

  - 或 Amazon Lambda Amazon ECS 计算平台上的蓝/绿：流量根据金丝雀、线性或all-at-once部署配置逐渐移动。
  - 蓝/绿部署通过 Amazon CloudFormation：作为 Amazon CloudFormation 堆栈更新的一部分，流量将从您当前的资源转移到更新的资源。当前，仅支持 ECS blue/green 部署。

  有关 blue/green 部署的更多信息，请参阅蓝绿部署概述。

> **Note**
>
> 同时使用 CodeDeploy 和支持 Amazon ECS blue/green 部署 Amazon CloudFormation。这些部署的详细信息将在后续章节中介绍。

IAM 实例配置文件

IAM 实例配置文件是您附加到您的 Amazon EC2 实例的 IAM 角色。此配置文件包括访问存储应用程序的 Amazon S3 存储桶或存储 GitHub 库所需的权限。有关更多信息，请参阅 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

修订

修订 是您的应用程序的一个版本。 Amazon Lambda 部署修订版是一个 YAML 或 JSON 格式的文件，用于指定要部署的 Lambda 函数的相关信息。 EC2/Unlide 部署修订版是一个存档文件，其中包含源内容（源代码、网页、可执行文件和部署脚本）和应用程序规范文件（AppSpec 文件）。 Amazon Lambda 修订版可以存储在亚马逊 S3 存储桶中。 EC2/本地版本存储在 Amazon S3 存储桶或 GitHub 存储库中。对于 Amazon S3，修订由其 Amazon S3 对象密钥和 ETag /或版本进行唯一标识。对于 GitHub，修订版本由其提交 ID 进行唯一标识。

服务角色

服务角色是一个 IAM 角色，它向 Amazon 服务授予权限，使其可以访问 Amazon 资源。您附加到服务角色的策略决定了服务可以访问哪些 Amazon 资源以及它可以对这些资源执行的操作。对于 CodeDeploy，服务角色用于以下用途：

- 读取应用于实例的标签或与实例关联的 Amazon A EC2 uto Scaling 组名称。这样就可以 CodeDeploy 确定它可以将应用程序部署到的实例。
- 要对实例、Amazon A EC2 uto Scaling 组和 ELB 负载均衡器执行操作。
- 向 Amazon SNS 主题发布信息，以便在发生指定部署或实例事件时发送通知。
- 检索有关 CloudWatch 警报的信息，为部署设置警报监控。

有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

目标修订

目标修订 是您已上传到存储库并要部署到部署组中的实例的应用程序修订的最新版本。换言之，当前面向部署的应用程序版本。这也是为自动部署提取的修订。

其他组件

有关 CodeDeploy 工作流中其他组件的信息，请参阅以下主题：

- 选择 CodeDeploy 存储库类型
- Deployments
- Application Specification Files
- Instance Health
- 与 CodeDeploy 代理合作
- Working with On-Premises Instances

CodeDeploy 部署

本主题提供 CodeDeploy 中部署的组件和工作流的相关信息。部署过程会有所不同，具体取决于您用于部署的计算平台或部署方法（Lambda、Amazon ECS、 EC2 /Ondless 或通过 Amazon CloudFormation）。

主题

- Amazon Lambda 计算平台上的部署
- Amazon ECS 计算平台上的部署
- 在 EC2 /本地计算平台上部署

Amazon Lambda 计算平台上的部署

本主题提供有关使用 Amazon Lambda 计算平台的 CodeDeploy 部署的组件和工作流程的信息。

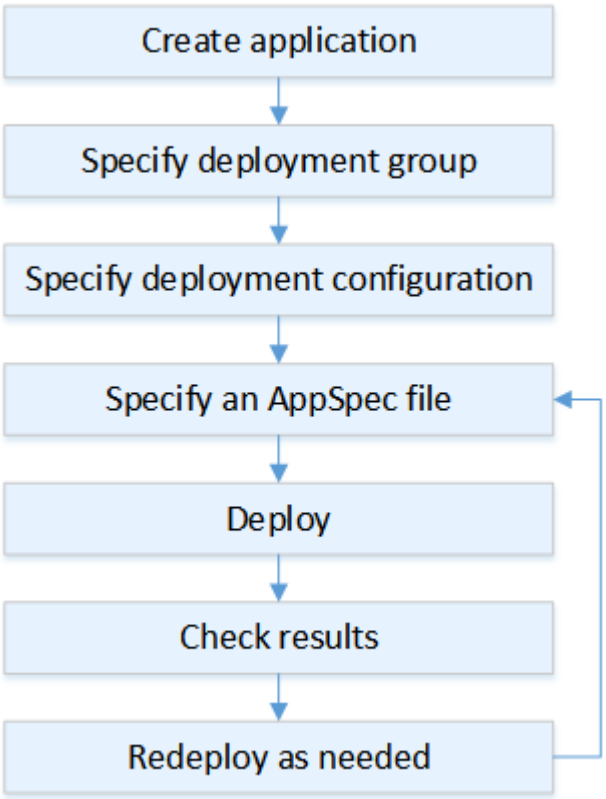
主题

- Amazon Lambda 计算平台上的部署工作流
- 上传应用程序修订

- 创建应用程序和部署组
- 部署应用程序修订
- 更新 应用程序
- 停止和失败的部署
- 重新部署和部署回滚

Amazon Lambda 计算平台上的部署工作流

下图显示了部署新增和更新 Amazon Lambda 函数的主要步骤。



这些步骤包括：

1. 创建应用程序并为其指定唯一标识要部署的应用程序修订的名称。要部署 Lambda 函数，请在创建应用程序时 Amazon 选择 Lambda 计算平台。 CodeDeploy 在部署期间使用此名称来确保它引用了正确的部署组件，例如部署组、部署配置和应用程序修订。有关更多信息，请参阅 使用创建应用程序 CodeDeploy。
2. 通过指定部署组的名称设置部署组。

3. 选择部署配置以指定流量如何从原始 Amazon Lambda 函数版本转移到新 Lambda 函数版本。有关更多信息，请参阅 View Deployment Configuration Details。
4. 将应用程序规范文件（AppSpec 文件）上传到 Amazon S3。该 AppSpec 文件指定了 Lambda 函数版本和用于验证部署的 Lambda 函数。如果您不想创建 AppSpec 文件，则可以使用 YAML 或 JSON 直接在控制台中指定 Lambda 函数版本和 Lambda 部署验证函数。有关更多信息，请参阅 正在处理的应用程序修订版 CodeDeploy。
5. 将您的应用程序修订部署到部署组。 Amazon CodeDeploy 将部署您指定的 Lambda 函数修订版。流量将使用您在创建应用程序时选择的部署 AppSpec 文件转移到您的 Lambda 函数修订版。有关更多信息，请参阅 使用创建部署 CodeDeploy。
6. 检查部署结果。有关更多信息，请参阅 监控中的部署 CodeDeploy。

上传应用程序修订

将 AppSpec 文件放入 Amazon S3 或将其直接输入控制台或 Amazon CLI。有关更多信息，请参阅 Application Specification Files。

创建应用程序和部署组

Amazon Lambda 计算平台上的 CodeDeploy 部署组可识别一个或多个 AppSpec 文件的集合。每个 AppSpec 文件可以部署一个 Lambda 函数版本。部署组还定义一些用于未来部署的配置选项，例如警报和回滚配置。

部署应用程序修订

现在，您可以将 AppSpec 文件中指定的函数修订版部署到部署组了。您可以使用 CodeDeploy 控制台或创建部署命令。可以指定一些参数（包括修订、部署组和部署配置）来控制部署。

更新 应用程序

您可以对应用程序进行更新，然后使用 CodeDeploy 控制台或调用 create- deployment 命令来推送修订。

停止和失败的部署

您可以使用 CodeDeploy 控制台或停止部署命令来停止部署。当您尝试停止部署时，将发生下面三种情况之一：

- 部署将停止，并且操作将返回成功状态。在这种情况下，没有更多的部署生命周期事件将在已停止部署的部署组上运行。

- 部署将不会立即停止，并且操作将返回挂起状态。在这种情况下，一些部署生命周期事件可能仍在部署组上运行。在挂起的操作完成后，停止部署的后续调用将返回成功状态。
- 部署无法停止，并且操作将返回错误。有关更多信息 ErrorInformation，请参阅 Amazon CodeDeploy API 参考中的常见错误。

与停止的部署一样，失败的部署可能导致某些部署生命周期事件已在运行。要查明部署失败的原因，可以使用 CodeDeploy 控制台或分析失败部署中的日志文件数据。有关更多信息，请参阅应用程序修订和日志文件清理和查看 CodeDeploy EC2/本地部署的日志数据。

重新部署和部署回滚

CodeDeploy 通过将先前部署的修订版作为新部署重新部署来实现回滚。

您可以对部署组进行配置，使之在满足特定条件（例如部署失败或达到警报监控阈值）时自动回滚部署。您还可以在单个部署中覆盖为部署组指定的回滚设置。

另外，也可以选择通过手动重新部署以前部署的版本回滚失败的部署。

在所有情况下，新的或回滚的部署都分配有自己的部署 ID。您可以在 CodeDeploy 控制台中查看的部署列表显示了哪些部署是自动部署的结果。

有关更多信息，请参阅 使用 CodeDeploy 重新部署和回滚部署。

Amazon ECS 计算平台上的部署

本主题提供有关使用 Amazon ECS 计算平台的 CodeDeploy 部署组件和工作流程的信息。

主题

- 在开始 Amazon ECS 部署之前
- Amazon ECS 计算平台上的部署工作流（高级）
- 在 Amazon ECS 部署过程中发生的事件
- 上传应用程序修订
- 创建应用程序和部署组
- 部署应用程序修订
- 更新应用程序
- 停止和失败的部署

- 重新部署和部署回滚
- 通过 Amazon ECS blue/green 部署 Amazon CloudFormation

在开始 Amazon ECS 部署之前

在开始 Amazon ECS 应用程序部署之前，必须准备好以下事项。有些要求是在创建部署组时指定的，有些则在 AppSpec 文件中指定。

要求	指定的位置
Amazon ECS 集群	部署组
Amazon ECS 服务	部署组
应用程序负载均衡器和网络负载均衡器	部署组
生产侦听器	部署组
测试侦听器（可选）	部署组
两个目标组	部署组
Amazon ECS 任务定义	AppSpec 文件
容器名称	AppSpec 文件
容器端口	AppSpec 文件

Amazon ECS 集群

Amazon ECS 集群是任务或服务的逻辑分组。在创建 CodeDeploy 应用程序的部署组时，您可以指定包含您的 Amazon ECS 服务的 Amazon ECS 集群。有关更多信息，请参阅《Amazon Elastic Container Service 用户指南》中的 Amazon ECS 集群。

Amazon ECS 服务

Amazon ECS 服务 维护并运行 Amazon ECS 集群中任务定义的指定实例。必须启用您的 Amazon ECS 服务 CodeDeploy。默认情况下，Amazon ECS 部署已启用 Amazon ECS 服务。当您创建部署组时，您需要选择部署 Amazon ECS 集群中的 Amazon ECS 服务。有关更多信息，请参阅《Amazon Elastic Container Service 用户指南》中的 Amazon ECS 服务。

应用程序负载均衡器和网络负载均衡器

您必须将 ELB 与要在 Amazon ECS 部署中更新的亚马逊 ECS 服务一起使用。您可以使用应用程序负载均衡器或网络负载均衡器。我们建议使用应用程序负载均衡器，以便您可以利用动态端口映射和基于路径的路由和优先级规则等功能。您可以在创建 CodeDeploy 应用程序的部署组时指定负载均衡器。有关更多信息，请参阅《Amazon Elastic Container Service 用户指南》中的为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器和创建负载均衡器。

一个或两个侦听器

侦听器 由负载均衡器用于将流量定向到目标组。必须提供一个生产侦听器。您可以指定可选的第二个测试侦听器，在您运行验证测试时该侦听器可以将流量定向到替换任务集。在创建部署组时，您需要指定一个或两个侦听器。如果您使用 Amazon ECS 控制台创建 Amazon ECS 服务，系统会为您创建侦听器。有关更多信息，请参阅 ELB 用户指南中的应用程序负载均衡器监听器和亚马逊弹性容器服务用户指南中的创建服务。

两个 Amazon ECS 目标群体

目标组 用于将流量路由到一个注册目标。一个 Amazon ECS 部署需要两个目标组：一个用于 Amazon ECS 应用程序的原始任务集，另一个用于其替换任务集。在部署期间， CodeDeploy 创建替换任务集并将流量从原始任务集重新路由到新任务集。在创建 CodeDeploy 应用程序的部署组时，您需要指定目标组。

在部署过程中， CodeDeploy 确定哪个目标组与您的 Amazon ECS 服务中具有状态的任务集相关联`PRIMARY`（这是原始任务集），并将一个目标组与该任务集相关联，然后将另一个目标组与替换任务集相关联。如果执行其他部署，则与当前部署的原始任务集关联的目标组将与下一个部署的替换任务集关联。有关更多信息，请参阅《ELB 用户指南》中的应用程序负载均衡器的目标组。

Amazon ECS 任务定义

需要任务定义 才能运行包含 Amazon ECS 应用程序的 Docker 容器。您可以在 CodeDeploy 应用程序文件中指定任务定义的 AppSpec ARN。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 任务定义和 AppSpec Amazon ECS 部署的 “资源” 部分。

您的 Amazon ECS 应用程序的容器

Docker 容器 是一套打包代码及其依赖项以便应用程序运行的软件。容器可以隔离您的应用程序，使其在不同的计算环境中运行。负载均衡器将流量定向到 Amazon ECS 应用程序任务集中的容器。您可以在 CodeDeploy 应用程序 AppSpec 的文件中指定容器的名称。 AppSpec 文件中指定的容器必须是 Amazon ECS 任务定义中指定的容器之一。有关更多信息，请参阅《Amazon

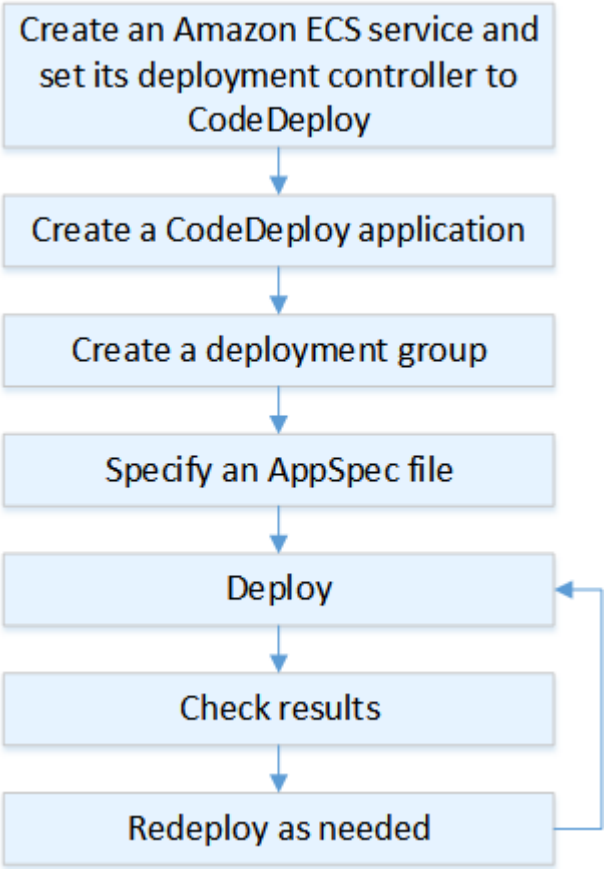
Elastic Container Service 用户指南》中的什么是 Amazon Elastic Container Service？和 AppSpec Amazon ECS 部署的 “资源” 部分。

您的替换任务集的端口

在 Amazon ECS 部署期间，您的负载均衡器会将流量引导到 CodeDeploy应用程序 AppSpec 文件中指定的容器上的此端口。你可以在 CodeDeploy 应用程序 AppSpec 的文件中指定端口。有关更多信息，请参阅 AppSpec Amazon ECS 部署的 “资源” 部分。

Amazon ECS 计算平台上的部署工作流（高级）

下图显示部署更新的 Amazon ECS 服务的主要步骤。



这些步骤包括：

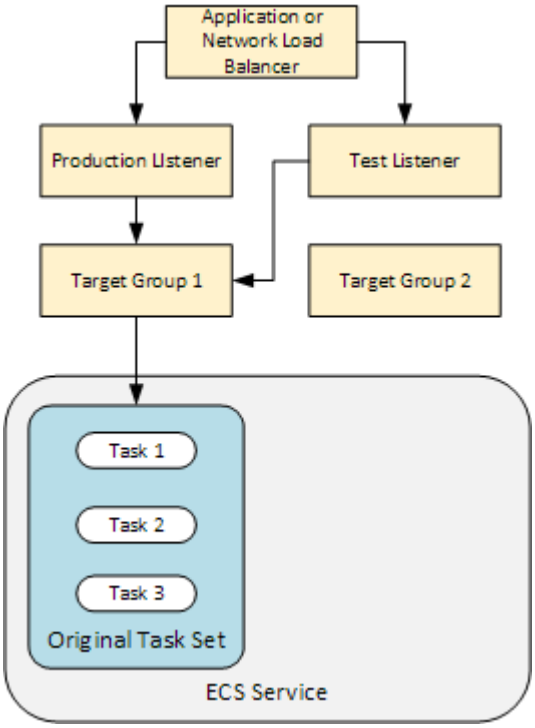
1. 通过指定一个唯一代表您要部署的内容的名称来创建 Amazon CodeDeploy 应用程序。要部署 Amazon ECS 应用程序，请在您的 Amazon CodeDeploy 应用程序中选择 Amazon ECS 计算平台。 CodeDeploy 在部署期间使用应用程序来引用正确的部署组件，例如部署组、目标组、侦听器、流量重新路由行为以及应用程序修订。有关更多信息，请参阅 使用创建应用程序 CodeDeploy。

2. 通过指定以下内容设置部署组：
   - 部署组名称。
   - Amazon ECS 集群和服务名称。Amazon ECS 服务的部署控制器必须设置为 CodeDeploy。
   - 生产侦听器、可选的测试侦听器以及在部署期间使用的目标组。
   - 部署设置，例如何时将生产流量重新路由到 Amazon ECS 服务中的替换 Amazon ECS 任务集以及何时终止 Amazon ECS 服务中的原始Amazon ECS任务集。
   - 可选设置，如触发器、警报和回滚行为。
3. 指定应用程序规范文件（AppSpec 文件）。您可以将其上传到 Amazon S3，以 YAML 或 JSON 格式将其输入控制台，或者使用 Amazon CLI 或软件开发工具包进行指定。该 AppSpec 文件指定了部署的 Amazon ECS 任务定义、用于路由流量的容器名称和端口映射，以及部署生命周期挂钩之后运行的 Lambda 函数。容器名称必须是您的 Amazon ECS 任务定义中的容器。有关更多信息，请参阅 正在处理的应用程序修订版 CodeDeploy。
4. 部署您的应用程序修订版。 Amazon CodeDeploy 将流量从 Amazon ECS 服务中任务集的原始版本重新路由到新的替代任务集。部署组中指定的目标组用于处理传输到原始和替换任务集的流量。在部署完成后，会终止原始任务集。您可以指定一个可选的测试侦听器，以便在将流量重新路由到替换版本之前为其提供测试流量。有关更多信息，请参阅 使用创建部署 CodeDeploy。
5. 检查部署结果。有关更多信息，请参阅 监控中的部署 CodeDeploy。

在 Amazon ECS 部署过程中发生的事件

在启动具有测试侦听器的 Amazon ECS 部署之前，您必须配置其组件。有关更多信息，请参阅 在开始 Amazon ECS 部署之前。

下图显示了在准备好启动 Amazon ECS 部署时，这些组件之间的关系。



当部署启动时，开始一次执行一个部署生命周期事件。某些生命周期事件是仅执行文件中指定的 Lambda 函数的 AppSpec 挂钩。下表中的部署生命周期事件按照执行的顺序列出。有关更多信息，请参阅 AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。

生命周期事件	生命周期事件操作
`BeforeInstall` （Lambda 函数的挂钩）	运行 Lambda 函数。
安装	设置替换任务集。
`AfterInstall` （Lambda 函数的挂钩）	运行 Lambda 函数。
AllowTestTraffic	将流量从测试侦听器路由至目标组 2。
`AfterAllowTestTraffic` （Lambda 函数的挂钩）	运行 Lambda 函数。
`BeforeAllowTraffic` （Lambda 函数的挂钩）	运行 Lambda 函数。
AllowTraffic	将流量从生产侦听器路由至目标组 2。

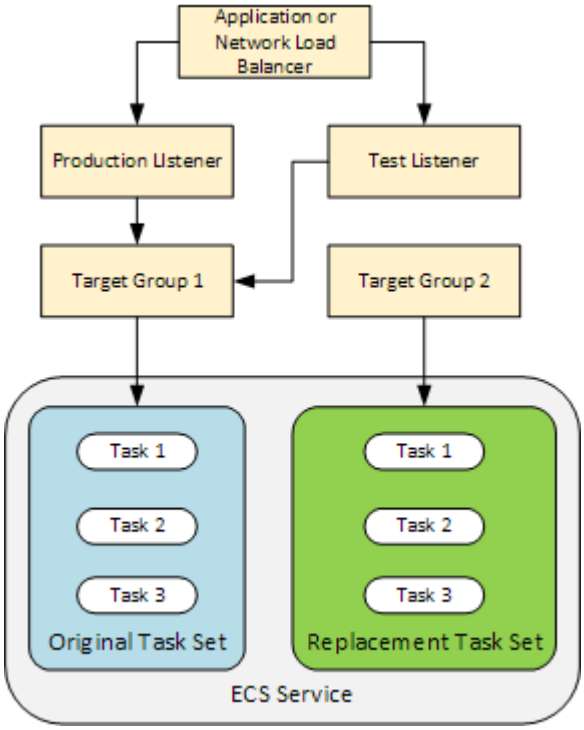
生命周期事件	生命周期事件操作
`AfterAllowTraffic`	运行 Lambda 函数。

> **Note**
>
> 挂钩中的 Lambda 函数是可选的。

1. 在文件中执行`BeforeInstall`挂钩中指定的任何 Lambda 函数。 AppSpec
2. 在 `Install` 生命周期事件期间：

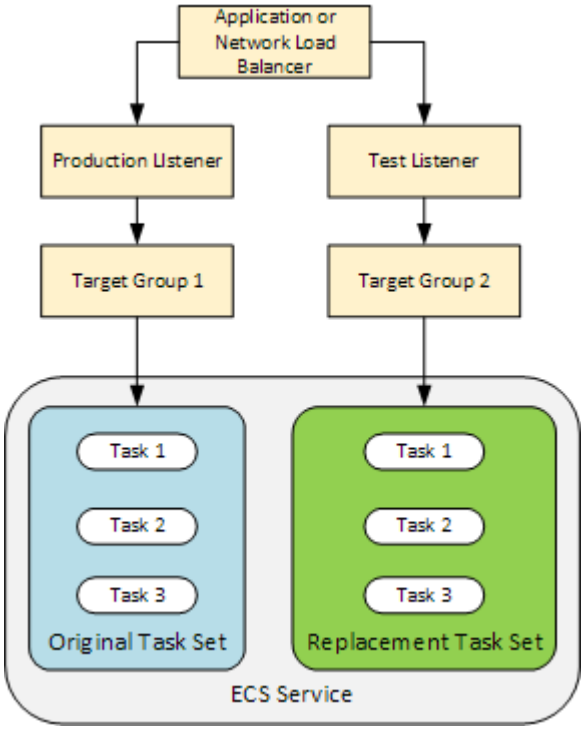
   a. 在您的 Amazon ECS 服务中创建替换任务集。
   b. 更新后的容器化应用程序安装到替换任务集中。
   c. 第二个目标组与替换任务集关联。

   此图显示具有新替换任务集的部署组件。容器化应用程序位于此任务集的内部。任务集包含三个任务。（应用程序可以具有任意数量的任务。）第二个目标组现与替换任务集关联。



3. 在文件中执行`AfterInstall`挂钩中指定的任何 Lambda 函数。 AppSpec

4. 调用了 `AllowTestTraffic` 事件。在此生命周期事件中，测试侦听器将流量路由到更新后的容器化应用程序中。



5. 在文件中执行`AfterAllowTestTraffic`挂钩中指定的任何 Lambda 函数。 AppSpec Lambda 函数可以使用测试流量来验证部署。例如，Lambda 函数可以向测试侦听器提供流量，并跟踪替换任务集的指标。如果配置了回滚，则可以配置 CloudWatch 警报，在 Lambda 函数中的验证测试失败时触发回滚。

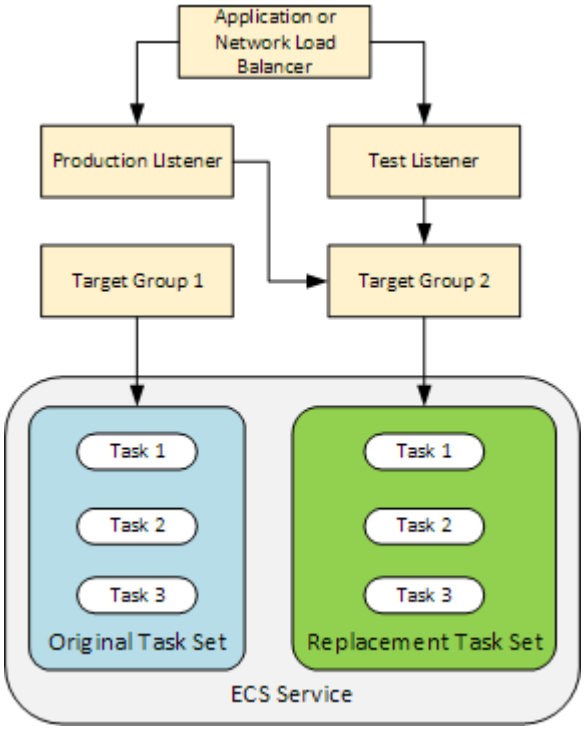
   验证测试完成后，将会发生以下情况之一：

   - 如果验证失败并配置了回滚，则部署状态标记为 `Failed`，组件返回其开始部署时的状态。
   - 如果验证失败但未配置回滚，则部署状态标记为 `Failed`，并且组件保持其当前状态。
   - 如果验证成功，则部署将继续到 `BeforeAllowTraffic` 挂钩。

   有关更多信息，请参阅 使用 CloudWatch 警报监控部署 CodeDeploy、自动回滚 和 为部署组配置高级选项。

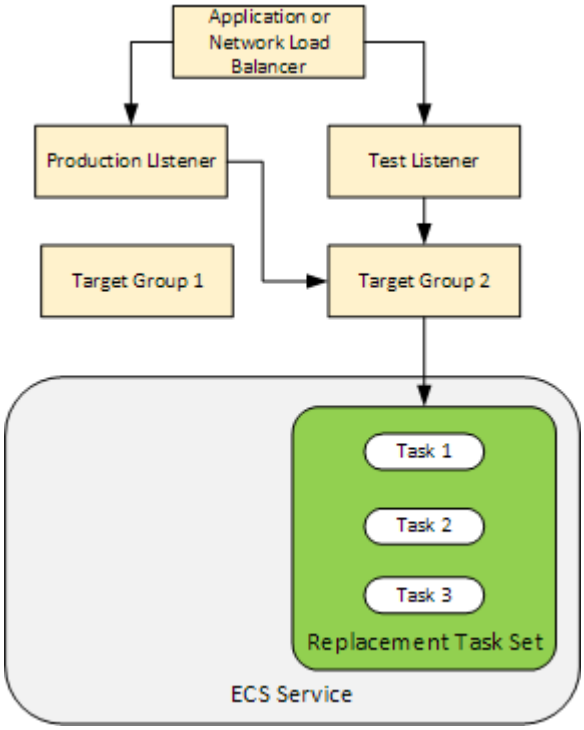
6. 在文件中执行`BeforeAllowTraffic`挂钩中指定的任何 Lambda 函数。 AppSpec

7. 调用了 `AllowTraffic` 事件。生产流量从原始任务集重新路由到替换任务集。下图显示了接收生产流量的替换任务集。



8. 在文件中执行`AfterAllowTraffic`挂钩中指定的任何 Lambda 函数。 AppSpec

9. 所有事件成功后，部署状态设置为 `Succeeded` 并删除原始任务集。



上传应用程序修订

将 AppSpec 文件放入 Amazon S3 或将其直接输入控制台或 Amazon CLI。有关更多信息，请参阅 Application Specification Files。

创建应用程序和部署组

Amazon ECS 计算平台上的 CodeDeploy 部署组可识别向更新后的 Amazon ECS 应用程序提供流量的侦听器和部署期间使用的两个目标组。部署组还定义一组配置选项，例如警报和回滚配置。

部署应用程序修订

现在，您已准备好部署在部署组中指定的更新的 Amazon ECS 服务。您可以使用 CodeDeploy 控制台或创建部署命令。可以指定一些参数（包括修订和部署组）来控制部署。

更新应用程序

您可以对应用程序进行更新，然后使用 CodeDeploy 控制台或调用 create- deployment 命令来推送修订。

停止和失败的部署

您可以使用 CodeDeploy 控制台或停止部署命令来停止部署。当您尝试停止部署时，将发生下面三种情况之一：

- 部署将停止，并且操作将返回成功状态。在这种情况下，没有更多的部署生命周期事件将在已停止部署的部署组上运行。
- 部署将不会立即停止，并且操作将返回挂起状态。在这种情况下，一些部署生命周期事件可能仍在部署组上运行。在挂起的操作完成后，停止部署的后续调用将返回成功状态。
- 部署无法停止，并且操作将返回错误。有关更多信息，请参阅 Amazon CodeDeploy API 参考中的错误信息和常见错误。

重新部署和部署回滚

CodeDeploy 通过将流量从替换任务集重新路由到原始任务集来实现回滚。

您可以对部署组进行配置，使之在满足特定条件（例如部署失败或达到警报监控阈值）时自动回滚部署。您还可以在单个部署中覆盖为部署组指定的回滚设置。

另外，也可以选择通过手动重新部署以前部署的版本回滚失败的部署。

在所有情况下，新的或回滚的部署都分配有自己的部署 ID。 CodeDeploy 控制台显示列出自动部署结果的部署列表。

如果进行重新部署，则与当前部署的原始任务集关联的目标组将与重新部署的替换任务集关联。

有关更多信息，请参阅 使用 CodeDeploy 重新部署和回滚部署。

通过 Amazon ECS blue/green 部署 Amazon CloudFormation

您可以使用 Amazon CloudFormation 通过管理 Amazon ECS blue/green 部署 CodeDeploy。有关更多信息，请参阅 通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation。

> **Note**
>
> 亚太地区（大阪）区域 Amazon CloudFormation 不支持使用管理 Amazon ECS blue/green 部署。

在 EC2 /本地计算平台上部署

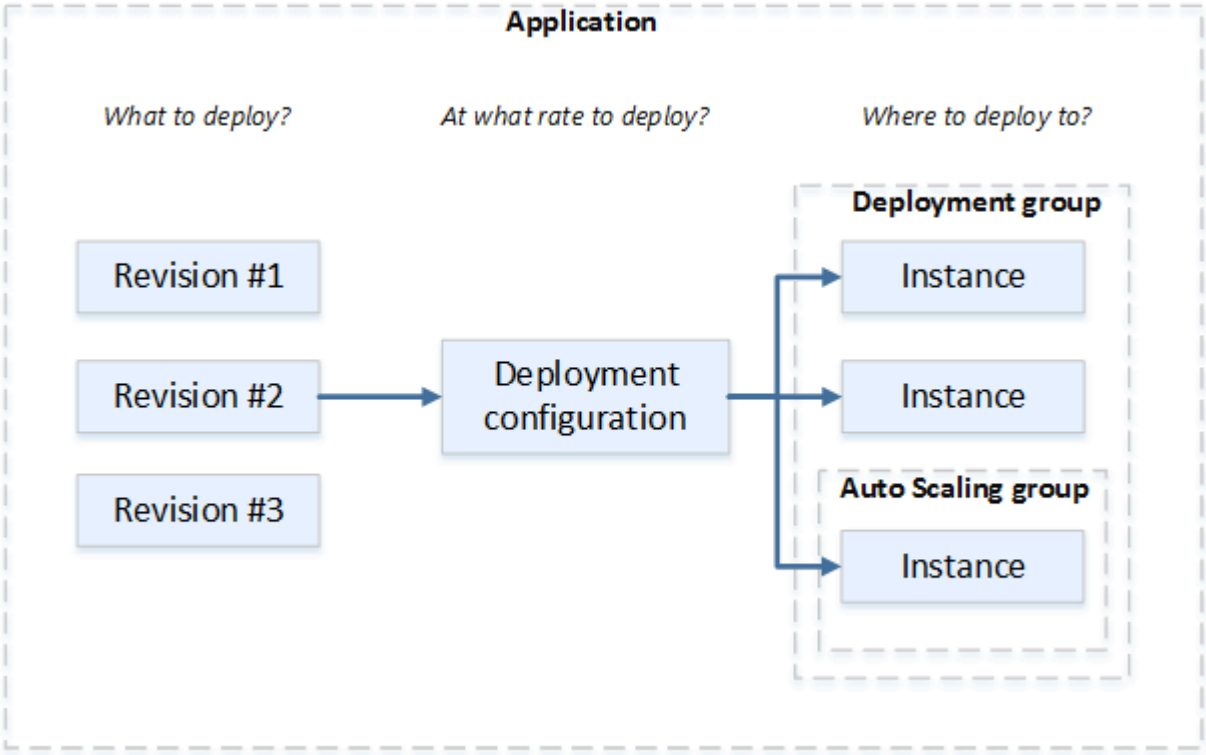
本主题提供有关使用 EC2 /Londest 计算平台的 CodeDeploy 部署的组件和工作流程的信息。有关 blue/green 部署的信息，请参阅蓝绿部署概述。

主题

- 在 EC2 /本地计算平台上部署组件
- EC2/本地计算平台上的部署工作流程
- 设置实例
- 上传应用程序修订
- 创建应用程序和部署组
- 部署应用程序修订
- 更新 应用程序
- 停止和失败的部署
- 重新部署和部署回滚

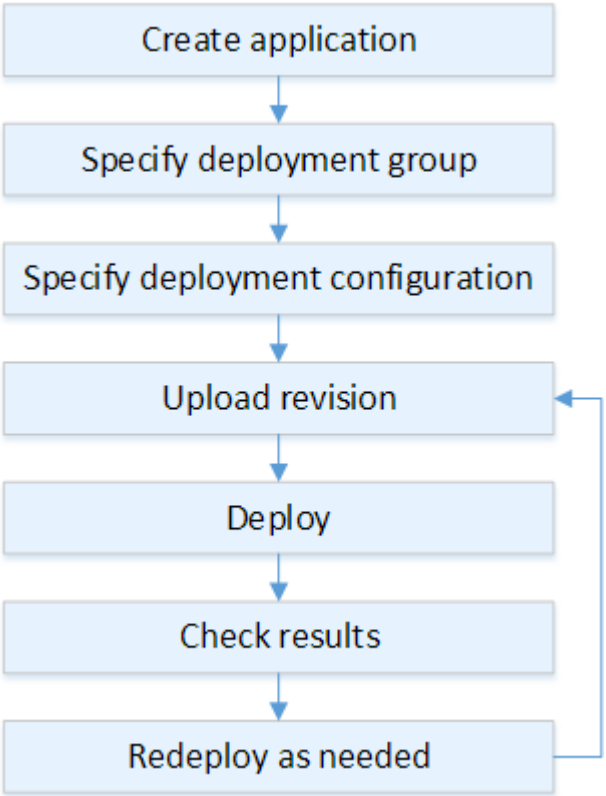
在 EC2 /本地计算平台上部署组件

下图显示了 EC2 /Onlide 计算平台上的 CodeDeploy 部署中的组件。



EC2/本地计算平台上的部署工作流程

下图显示部署应用程序修订的主要步骤：



这些步骤包括：

1. 创建应用程序并为其命名，以唯一标识要部署的应用程序修订版和应用程序的计算平台。CodeDeploy 在部署期间使用此名称来确保它引用了正确的部署组件，例如部署组、部署配置和应用程序修订。有关更多信息，请参阅 使用创建应用程序 CodeDeploy。
2. 指定部署类型以及要将应用程序修订部署到的实例，以此来设置部署组。就地部署将使用最新的应用程序修订更新实例。 blue/green 部署使用负载均衡器为部署组注册一组替换实例，并注销原始实例。

   您可以指定应用于实例的标签、Amazon A EC2 uto Scaling 组名称或两者兼而有之。

   如果您在部署组中指定了一组标签，则会 CodeDeploy 部署到至少应用了其中一个指定标签的实例。如果您指定两个或更多标签组，则仅 CodeDeploy 部署到符合每个标签组标准的实例。有关更多信息，请参阅 Tagging Instances for Deployments。

   在所有情况下，都必须将实例配置为在部署中使用（也就是说，它们必须被标记或属于 Amazon A EC2 uto Scaling 组），并且必须安装并运行 CodeDeploy 代理。

   我们为您提供一个 Amazon CloudFormation 模板，您可以使用该模板快速设置基于亚马逊 Linux 或 Windows 服务器的亚马逊 EC2实例。我们还为您提供独立 CodeDeploy 代理，以便您可以将其安装

在亚马逊 Linux、Ubuntu 服务器、红帽企业 Linux (RHEL) 或 Windows 服务器实例上。有关更多信息，请参阅 使用创建部署组 CodeDeploy。

您还可以指定以下选项：

- Amazon SNS 通知。创建触发器，以便在部署和实例中发生指定的事件（例如，成功或失败事件）时，向 Amazon SNS 主题的订阅者发送通知。有关更多信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications。
- 基于警报的部署管理。实施 Amazon CloudWatch 警报监控，以便在您的指标超过或低于中设定的阈值时停止部署 CloudWatch。
- 自动部署回滚。配置部署，使之在部署失败或达到警报阈值时自动回滚到已知良好的版本。

3. 指定部署配置，以指示您的应用程序修订需要同时部署多少个实例，以及说明部署到成功和失败条件。有关更多信息，请参阅 View Deployment Configuration Details。
4. 将应用程序修订版上传到 Amazon S3 或 GitHub。除了要部署的文件和要在部署期间运行的任何脚本外，还必须包括应用程序规范文件（AppSpec 文件）。该文件包含部署说明，例如，要将文件复制到的每个实例上的位置，以及运行部署脚本的时间。有关更多信息，请参阅 正在处理的应用程序修订版 CodeDeploy。
5. 将应用程序修订部署到部署组。部署组中每个实例上的 CodeDeploy 代理会将您的应用程序修订从 Amazon S3 复制 GitHub 到该实例。然后， CodeDeploy 代理解绑修订版，并使用该 AppSpec 文件将文件复制到指定位置并执行所有部署脚本。有关更多信息，请参阅 使用创建部署 CodeDeploy。
6. 检查部署结果。有关更多信息，请参阅 监控中的部署 CodeDeploy。
7. 部署修订。如果您需要修复源内容中的错误，或以不同顺序运行部署脚本，或处理失败的部署，则可能需要重新部署。为此，请将修改后的源内容、所有部署脚本和 AppSpec 文件重新捆绑到新的修订版中，然后将修订版上传到 Amazon S3 存储桶或存储 GitHub 库。然后，使用新修订执行到同一部署组的新部署。有关更多信息，请参阅 使用创建部署 CodeDeploy。

设置实例

您必须先设置实例，然后才能首次部署应用程序修订。如果一个应用程序修订需要三个生产服务器和两个备份服务器，您将启动或使用五个实例。

要手动预配置实例，请执行以下操作：

1. 在实例上安装 CodeDeploy 代理。该 CodeDeploy 代理可以安装在亚马逊 Linux、Ubuntu Server、RHEL 和 Windows 服务器实例上。

2. 如果您使用标签来识别部署组中的实例，请启用标记。 CodeDeploy 依靠标签来识别实例并将其分组到 CodeDeploy 部署组中。尽管入门教程同时使用了键和值，但是您可以只使用键或值为部署组定义标签。
3. 启动附有 IAM EC2 实例配置文件的 Amazon 实例。IAM 实例配置文件必须在 Amazon EC2 实例启动时附加到该实例， CodeDeploy 代理才能验证该实例的身份。
4. 创建服务角色。提供服务访问权限，以便 CodeDeploy 可以扩展您 Amazon 账户中的标签。

对于初始部署， Amazon CloudFormation 模板会为您完成所有这些操作。它基于已安装 CodeDeploy 代理的亚马逊 Linux 或 Windows 服务器创建和配置新的单个亚马逊 EC2 实例。有关更多信息，请参阅 使用适用于 CodeDeploy 的实例。

> **Note**
>
> 对于 blue/green 部署，您可以选择使用已有的替换环境实例，也可以让您在部署过程中 CodeDeploy 配置新实例。

上传应用程序修订

将 AppSpec 文件放在应用程序源内容文件夹结构中的根文件夹下。有关更多信息，请参阅 Application Specification Files。

将应用程序源内容文件夹结构捆绑成存档文件格式，例如 zip、tar 或压缩的 tar。将存档文件（修订版）上传到 Amazon S3 存储桶或 GitHub 存储库。

> **Note**
>
> Windows Server 实例不支持 tar 和压缩的 tar 存档文件格式（.tar 和.tar.gz）。

创建应用程序和部署组

CodeDeploy 部署组根据实例的标签、Amazon A EC2 uto Scaling 组名称或两者来识别一组实例。可以向同一实例部署多个应用程序修订。可以向多个实例部署一个应用程序修订。

例如，可以为三个生产服务器添加标签“Prod”，为两个备份服务器添加标签“Backup”。这两个标签可用于在 CodeDeploy 应用程序中创建两个不同的部署组，从而允许您选择哪组（或两者）应参与部署。

您可以在部署组中使用多个标签组，将部署限制到一组较少的实例。有关信息，请参阅Tagging Instances for Deployments。

部署应用程序修订

现在，您可以从 Amazon S3 或 GitHub 部署组部署您的应用程序修订了。您可以使用 CodeDeploy 控制台或创建部署命令。可以指定一些参数（包括修订、部署组和部署配置）来控制部署。

更新 应用程序

您可以对应用程序进行更新，然后使用 CodeDeploy 控制台或调用 create- deployment 命令来推送修订。

停止和失败的部署

您可以使用 CodeDeploy 控制台或停止部署命令来停止部署。当您尝试停止部署时，将发生下面三种情况之一：

- 部署将停止，并且操作将返回成功状态。在这种情况下，没有更多的部署生命周期事件将在已停止部署的部署组上运行。一些文件可能已复制到部署组中的一个或多个实例，并且一些脚本可能已在一个或多个实例上运行。
- 部署将不会立即停止，并且操作将返回挂起状态。在这种情况下，一些部署生命周期事件可能仍在部署组上运行。一些文件可能已复制到部署组中的一个或多个实例，并且一些脚本可能已在一个或多个实例上运行。在挂起的操作完成后，停止部署的后续调用将返回成功状态。
- 部署无法停止，并且操作将返回错误。有关更多信息 ErrorInformation，请参阅 Amazon CodeDeploy API 参考中的常见错误。

与停止的部署一样，失败的部署可能导致一些部署生命周期事件已在部署组中的一个或多个实例上运行。要找出部署失败的原因，您可以使用 CodeDeploy 控制台、调用get-deployment-instance命令或分析失败部署的日志文件数据。有关更多信息，请参阅应用程序修订和日志文件清理和查看 CodeDeploy EC2/本地部署的日志数据。

重新部署和部署回滚

CodeDeploy 通过将先前部署的修订版作为新部署重新部署来实现回滚。

您可以对部署组进行配置，使之在满足特定条件（例如部署失败或达到警报监控阈值）时自动回滚部署。您还可以在单个部署中覆盖为部署组指定的回滚设置。

另外，也可以选择通过手动重新部署以前部署的版本回滚失败的部署。

在所有情况下，新的或回滚的部署都分配有自己的部署 ID。您可以在 CodeDeploy 控制台中查看的部署列表显示了哪些部署是自动部署的结果。

有关更多信息，请参阅 使用 CodeDeploy 重新部署和回滚部署。

CodeDeploy 应用程序规范（AppSpec）文件

对 CodeDeploy 唯一的应用程序规范文件（AppSpec 文件）是 YAML 格式或 JSON 格式文件。AppSpec 文件用于将每个部署作为在文件中定义的一系列生命周期事件挂钩进行管理。

有关如何创建格式正确的 AppSpec 文件的信息，请参阅CodeDeploy AppSpec 文件参考。

主题

- Amazon ECS 计算平台上的 AppSpec 文件
- Amazon Lambda 计算平台上的 AppSpec 文件
- EC2/本地计算平台上的 AppSpec 文件
- CodeDeploy 代理如何使用 AppSpec 文件

Amazon ECS 计算平台上的 AppSpec 文件

如果您的应用程序使用 Amazon ECS 计算平台，则 AppSpec 文件可以是 YAML 或 JSON 格式的。它还可以直接键入到控制台中的编辑器内。AppSpec 文件用于指定：

- 用于将流量定向到新任务集的 Amazon ECS 服务名称以及容器名称和端口。
- 要用作验证测试的函数。

可以在部署生命周期事件后验证 Lambda 函数。有关更多信息，请参阅AppSpec 亚马逊 ECS 部署的 “挂钩” 部分、 AppSpec Amazon ECS 部署的文件结构 和 AppSpec Amazon ECS 部署的文件示例 。

Amazon Lambda 计算平台上的 AppSpec 文件

如果您的应用程序使用 Amazon Lambda 计算平台，则 AppSpec 文件也可以是 YAML 或 JSON 格式的。它还可以直接键入到控制台中的编辑器内。AppSpec 文件用于指定：

- 要部署的 Amazon Lambda 函数版本。

- 要用作验证测试的函数。

可以在部署生命周期事件后验证 Lambda 函数。有关更多信息，请参阅 AppSpec Amazon Lambda 部署的 “挂钩” 部分。

EC2/本地计算平台上的 AppSpec 文件

如果您的应用程序使用 EC2/本地计算平台，则 AppSpec 文件始终是 YAML 格式的。AppSpec 文件用于：

- 将应用程序修订中的源文件映射到其在实例上的目的地。
- 为部署的文件指定自定义权限。
- 指定要在部署过程的各个阶段在每个实例上运行的脚本。

您可以在多个单独的部署生命周期事件之后在一个实例上运行脚本。CodeDeploy 只运行此文件中指定的脚本，但这些脚本可以调用实例上的其他脚本。您可以运行任何类型的脚本，只要该脚本受实例上运行的操作系统支持即可。有关更多信息，请参阅 AppSpec EC2/本地部署的 “挂钩” 部分。

CodeDeploy 代理如何使用 AppSpec 文件

在部署期间，CodeDeploy 代理会在 AppSpec 文件的 hooks 部分查找当前事件的名称。如果找不到该事件，CodeDeploy 代理将移到下一步。如果找到该事件，CodeDeploy 代理将检索要执行的脚本列表。脚本按其在文件中的出现顺序运行。每个脚本的状态记录在实例上的 CodeDeploy 代理日志文件中。

如果脚本运行成功，则返回退出代码 0（零）。

> **Note**
>
> Amazon Lambda 或 Amazon ECS 部署中不使用 CodeDeploy 代理。

在安装事件期间，CodeDeploy 代理使用 AppSpec 文件的 files 部分中定义的映射，来确定要从修订复制到实例的文件夹或文件。

如果在操作系统上安装的 CodeDeploy 代理与 AppSpec 文件中列出的代理不匹配，则部署将失败。

有关 CodeDeploy 代理日志文件的信息，请参阅与 CodeDeploy 代理合作。

入门 CodeDeploy

主题

- 步骤 1：设置
- 步骤 2：为创建服务角色 CodeDeploy
- 步骤 3：限制 CodeDeploy 用户的权限
- 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件

步骤 1：设置

在 Amazon CodeDeploy 首次使用之前，必须完成设置步骤。这些步骤包括创建一个 Amazon 帐户（如果您还没有）和一个具有编程访问权限的管理用户。

在本指南中，管理用户被称为CodeDeploy管理用户。

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，将接到电话或收到短信，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

现在，您已创建并以CodeDeploy 管理员用户身份登录。

授予编程式访问权限

如果用户想在 Amazon 外部进行交互，则需要编程访问权限 Amazon Web Services 管理控制台。Amazon APIs 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期证书签署对 Amazon CLI 或的编程请求 Amazon APIs（直接或使用 Amazon SDKs）。	按照 IAM 用户指南中的将临时证书与 Amazon 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期证书签署对 Amazon CLI 或的编程请求 Amazon APIs（直接或使用 Amazon SDKs）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

> **Important**
>
> 我们强烈建议您将 CodeDeploy 管理员用户配置为员工身份（在 IAM Identity Center 中管理的用户），使用。 Amazon CLI本指南中的许多步骤都假设您使用 Amazon CLI 来执行配置。

> **Important**
>
> 如果您配置了 Amazon CLI，则系统可能会提示您指定 Amazon 区域。选择《Amazon Web Services 一般参考》的区域和终端节点中列出的受支持区域之一。

步骤 2：为创建服务角色 CodeDeploy

在中 Amazon，服务角色用于向 Amazon 服务授予权限，使其可以访问 Amazon 资源。附加到服务角色的策略将确定服务可访问的 资源以及可使用这些资源执行的操作。

您为其创建的服务角色 CodeDeploy 必须被授予计算平台所需的权限。如果您部署到多个计算平台，请为每个平台创建一个服务角色。要添加权限，请附加以下一个或多个 Amazon 提供的策略：

对于 EC2 /本地部署，请附加**`AWSCodeDeployRole`**策略。该策略为您的服务角色提供以下权限：

- 阅读您的实例上的标签或通过 Amazon A EC2 uto Scaling 组名识别您的亚马逊 EC2 实例。
- 读取、创建、更新和删除 Amazon A EC2 uto Scaling 群组、生命周期挂钩和扩展策略。
- 将信息发布到 Amazon SNS 主题。
- 检索有关 CloudWatch 警报的信息。
- 阅读并更新 Elastic Load Balancing。

> **Note**
>
> 如果您使用启动模板创建 Auto Scaling 组，则必须添加以下权限：
>
> - `ec2:RunInstances`
> - `ec2:CreateTags`
> - `iam:PassRole`
>
> 有关更多信息步骤 2：创建服务角色，请参阅《Amazon A EC2 uto Scaling 用户指南》中的“为 Auto Scalin g 组创建启动模板” 和 “启动模板支持”。

对于 Amazon ECS 部署，如果您想要针对支持服务的完全访问权限，请附加 **`AWSCodeDeployRoleForECS`** 策略。该策略为您的服务角色提供以下权限：

- 读取、更新和删除 Amazon ECS 任务集。

- 更新 ELB 目标组、监听器和规则。
- 调用 Amazon Lambda 函数。
- 访问 Amazon S3 存储桶中的修订文件。
- 检索有关 CloudWatch 警报的信息。
- 将信息发布到 Amazon SNS 主题。

对于 Amazon ECS 部署，如果您想要针对支持服务的有限访问权限，请附加 **AWSCodeDeployRoleForECSLimited** 策略。该策略为您的服务角色提供以下权限：

- 读取、更新和删除 Amazon ECS 任务集。
- 检索有关 CloudWatch 警报的信息。
- 将信息发布到 Amazon SNS 主题。

对于 Amazon Lambda 部署，如果您想允许发布到 Amazon SNS，请附上该政策。**AWSCodeDeployRoleForLambda**该策略为您的服务角色提供以下权限：

- 读取、更新和调用 Amazon Lambda 函数和别名。
- 访问 Amazon S3 存储桶中的修订文件。
- 检索有关 CloudWatch 警报的信息。
- 将信息发布到 Amazon SNS 主题。

对于 Amazon Lambda 部署，如果您想限制对 Amazon SNS 的访问，请附上该政策。**AWSCodeDeployRoleForLambdaLimited**该策略为您的服务角色提供以下权限：

- 读取、更新和调用 Amazon Lambda 函数和别名。
- 访问 Amazon S3 存储桶中的修订文件。
- 检索有关 CloudWatch 警报的信息。

在设置服务角色过程中，您还可以更新其信任关系，指定您希望向其授予访问权限的终端节点。

您可以使用 IAM 控制台 Amazon CLI、或 IAM 创建服务角色 APIs。

主题

- 创建服务角色（控制台）

- 创建服务角色（CLI）
- 获取服务角色 ARN（控制台）
- 获取服务角色 ARN（CLI）

创建服务角色（控制台）

1. 登录 Amazon Web Services 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选择创建角色。
3. 选择Amazon 服务，然后在用例下的下拉列表中选择CodeDeploy。
4. 选择您的使用案例：
   - 对于 EC2 /本地部署，请选择CodeDeploy。
   - 对于 Amazon Lambda 部署，请选择 CodeDeploy Lambda。
   - 对于 Amazon ECS 部署，请选择 CodeDeploy -ECS。
5. 选择下一步。
6. 在添加权限页面上，将显示该使用案例的正确权限策略。选择下一步。
7. 在命名、查看和创建页面上的角色名称中，输入服务角色的名称（例如，**`CodeDeployServiceRole`**），然后选择创建角色。

   您还可以在角色描述中输入此服务角色的描述。
8. 如果您希望此服务角色有权访问所有当前支持的终端节点，则您已完成了此过程。

   要限制此服务角色访问某些端点，请继续执行此过程中的其余步骤。
9. 在角色列表中搜索并选择您刚刚创建的角色（`CodeDeployServiceRole`）。
10. 选择 Trust relationships（信任关系）选项卡。
11. 选择编辑信任策略。

    您应该看到以下策略，该策略向服务角色提供访问所有支持的终端节点的权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

有关创建服务角色的更多信息，请参阅 IAM 用户指南中的创建角色以向 Amazon 服务委派权限。

创建服务角色（CLI）

1. 例如，在您的开发计算机上，创建一个名为 `CodeDeployDemo-Trust.json` 的文本文件。使用此文件可允许 CodeDeploy 代表您执行操作。

   在文件中保存以下内容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 从相同的目录调用 create-role 命令，根据刚刚创建的文本文件中的信息创建名为 **`CodeDeployServiceRole`** 的服务角色：

```
aws iam create-role --role-name CodeDeployServiceRole --assume-role-policy-document
 file://CodeDeployDemo-Trust.json
```

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

在命令输出中，记录 `Arn` 对象下方 `Role` 条目的值。稍后创建部署组时将需要它。如果您忘记了值，请按照获取服务角色 ARN（CLI） 中的说明操作。

3. 调用 attach-role-policy 命令以根据名为 **`AWSCodeDeployRole`** 的 IAM 托管策略向名为 **`CodeDeployServiceRole`** 的服务角色授予权限。例如：

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn arn:aws:-
cn:iam::aws:policy/service-role/AWSCodeDeployRole
```

有关创建服务角色的更多信息，请参阅 IAM 用户指南中的为 Amazon 服务创建角色。

获取服务角色 ARN（控制台）

要使用 IAM 控制台获取服务角色的 ARN，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 在筛选条件框中，键入 **`CodeDeployServiceRole`**，然后按 Enter。
4. 选择 CodeDeployServiceRole。
5. 记下 Role ARN（角色 ARN）字段的值。

获取服务角色 ARN（CLI）

要使用获取服务角色的 ARN，请对名为的服务角色调用get-role命令： Amazon CLI **`CodeDeployServiceRole`**

```
aws iam get-role --role-name CodeDeployServiceRole --query "Role.Arn" --output text
```

返回值是服务角色的 ARN。

步骤 3：限制 CodeDeploy 用户的权限

出于安全考虑，我们建议您将您在中创建的管理用户的权限限制步骤 1：设置为在中创建和管理部署所需的权限 CodeDeploy。

使用以下一系列过程来限制 CodeDeploy 管理用户的权限。

开始前的准备工作

- 确保按照中的说明在 IAM Identity Center 中创建了 CodeDeploy 管理用户步骤 1：设置。

创建权限集

稍后，您将向 CodeDeploy 管理用户分配此权限集。

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon IAM Identity Center 控制台，网址为https://console.aws.amazon.com/singlesignon/。
2. 在导航窗格中，选择权限集，然后选择创建权限集。
3. 选择自定义权限集。
4. 选择下一步。
5. 选择内联策略。
6. 删除示例代码。
7. 添加以下策略代码：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeDeployAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "autoscaling:*",
                "codedeploy:*",
                "ec2:*",
                "lambda:*",
                "ecs:*",
                "elasticloadbalancing:*",
```

```
                "iam:AddRoleToInstanceProfile",
                "iam:AttachRolePolicy",
                "iam:CreateInstanceProfile",
                "iam:CreateRole",
                "iam:DeleteInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListInstanceProfilesForRole",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:PutRolePolicy",
                "iam:RemoveRoleFromInstanceProfile",
                "s3:*",
                "ssm:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeDeployRolePolicy",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/
CodeDeployServiceRole"
        }
    ]
}
```

在此策略中，*arn:aws:iam::account-ID:role/CodeDeployServiceRole*用您在中创建的 CodeDeploy 服务角色的 ARN 值替换。步骤 2：为创建服务角色 CodeDeploy您可以在 IAM 控制台的服务角色的详细信息页面中找到 ARN 值。

上述策略允许您将应用程序部署到 Amazon Lambda 计算平台、 EC2 /本地计算平台和 Amazon ECS 计算平台。

您可以使用本文档中提供的 Amazon CloudFormation 模板启动与兼容的 Amazon EC2 实例 CodeDeploy。要使用 Amazon CloudFormation 模板创建应用程序、部署组或部署配置，必须通

过向 CodeDeploy 管理用户的权限策略添加权限来提供访问`cloudformation:*`权限 Amazon CloudFormation以及 Amazon CloudFormation 依赖的 Amazon 服务和操作，如下所示：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": "*"
        }
    ]
}
```

8. 选择下一步。
9. 在权限集名称中，输入：

```
CodeDeployUserPermissionSet
```

10. 选择下一步。
11. 在查看和创建页面上，检查相应信息，然后选择创建。

将权限集分配给 CodeDeploy 管理用户

1. 在导航窗格中 Amazon Web Services 账户，选择，然后选中您当前登录 Amazon Web Services 账户 的旁边的复选框。
2. 选择分配用户或组按钮。
3. 选择用户选项卡。
4. 选中 CodeDeploy 管理员用户旁边的复选框。
5. 选择下一步。
6. 选中 `CodeDeployUserPermissionSet` 旁边的复选框。
7. 选择下一步。
8. 检查相应信息，然后选择提交。

现在，您已将 CodeDeploy 管理员用户和分配`CodeDeployUserPermissionSet`给您的 Amazon Web Services 账户，并将它们绑定在一起。

注销并以 CodeDeploy 管理员用户身份重新登录

1. 在注销之前，请确保您拥有 CodeDeploy 管理员 Amazon 用户的访问门户 URL 以及用户名和一次性密码。

   > **Note**
   >
   > 如果您没有这些信息，请前往 IAM Identity Center 中的 CodeDeploy 管理员用户详细信息页面，选择重置密码，生成一次性密码 [...] ，然后再次重置密码以在屏幕上显示信息。

2. 退出 Amazon。
3. 将 Amazon 访问门户 URL 粘贴到浏览器的地址栏中。
4. 以 CodeDeploy 管理员用户身份登录。

   屏幕上会出现一个 Amazon Web Services 账户框。

5. 选择 Amazon Web Services 账户，然后选择您 Amazon Web Services 账户 为其分配了 CodeDeploy 管理员用户和权限集的名称。
6. 在 `CodeDeployUserPermissionSet` 旁，选择管理控制台。

   Amazon Web Services 管理控制台 出现了。现在，您已以 CodeDeploy 具有有限权限的管理员用户身份登录。现在，您可以以该用户身份执行 CodeDeploy CodeDeploy相关操作，并且只能执行相关的操作。

步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件

> **Note**
>
> 如果您使用的是 Amazon ECS 或 Amazon Lambda 计算平台，请跳过此步骤。

您的 Amazon EC2 实例需要获得访问存储应用程序的 Amazon S3 存储桶或存储 GitHub 库的权限。要启动与兼容的 Amazon EC2 实例 CodeDeploy，您必须创建其他 IAM 角色，即实例配置文件。这些说

明向您展示了如何创建 IAM 实例配置文件以附加到您的 Amazon EC2 实例。此角色授予 CodeDeploy 代理访问存储应用程序的 Amazon S3 存储桶或存储 GitHub 库的权限。

您可以使用 Amazon CLI、IAM 控制台或 IAM 创建 IAM 实例配置文件 APIs。

> **Note**
>
> 您可以在启动 Amazon 实例时将 IAM EC2 实例配置文件附加到该实例或之前启动的实例。有关更多信息，请参阅实例配置文件。

主题

- 为您的 Amazon 实例 (CLI) 创建 IAM EC2 实例配置文件
- 为您的 Amazon 实例创建 IAM EC2 实例配置文件（控制台）

为您的 Amazon 实例 (CLI) 创建 IAM EC2 实例配置文件

在这些步骤中，我们假定您已遵循入门 CodeDeploy中前三步的说明。

1. 在您的开发计算机上，创建一个名为 `CodeDeployDemo-EC2-Trust.json` 的文本文件。粘贴以下内容，这样 Amazon EC2 就可以代表您开展工作：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ec2.cn-north-1.amazonaws.com",
                    "ec2.cn-northwest-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 在相同的目录中，创建一个名为 `CodeDeployDemo-EC2-Permissions.json` 的文本文件。粘贴以下内容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 我们建议您将此政策限制为仅适用于您的亚马逊 EC2实例必须访问的 Amazon S3 存储桶。确保允许访问包含 CodeDeploy 代理的 Amazon S3 存储桶。否则，在实例上安装或更新 CodeDeploy 代理时可能会出现错误。例如：
>
> ```
> {
>  "Version": "2012-10-17",
>  "Statement": [
>  {
>  "Effect": "Allow",
>  "Action": [
>  "s3:Get*",
>  "s3:List*"
>  ],
>  "Resource": [
>  "arn:aws:s3:::amzn-s3-demo-bucket/*",
>  "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
>  "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
>  ]
>  }
>  ]
> }
> ```

> **Note**
>
> 如果您想将 IAM 授权或亚马逊虚拟私有云 (VPC) 终端节点 CodeDeploy与一起使用，则需要添加更多权限。有关更多信息，请参阅 CodeDeploy 与 Amazon Virtual Private Cloud 配合使用。

3. 从同一目录调用 create-role 命令，根据第一个文件中的信息创建名为 的 **`CodeDeployDemo-EC2-Instance-Profile`** 角色：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws iam create-role --role-name CodeDeployDemo-EC2-Instance-Profile --assume-role-policy-document file://CodeDeployDemo-EC2-Trust.json
```

4. 从同一目录调用 put-role-policy 命令，根据第二个文件中的信息为名为 **`CodeDeployDemo-EC2-Instance-Profile`** 的角色提供权限：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws iam put-role-policy --role-name CodeDeployDemo-EC2-Instance-Profile --policy-name CodeDeployDemo-EC2-Permissions --policy-document file://CodeDeployDemo-EC2-Permissions.json
```

5. 致电attach-role-policy为角色授予 Amazon S EC2 ystems Manager 权限，以便 SSM 可以安装 CodeDeploy 代理。如果您计划使用命令行从公用 Amazon S3 存储桶安装代理，则不需要此策略。了解有关安装 CodeDeploy 代理的详细信息。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore --role-name CodeDeployDemo-EC2-Instance-Profile
```

6. 依次调用 create-instance-profile 命令和 add-role-to-instance-profile 命令，创建名为 **`CodeDeployDemo-EC2-Instance-Profile`** 的 IAM 实例配置文件。实例配置文件 EC2 允许亚马逊在首次启动 EC2 实例时**`CodeDeployDemo-EC2-Instance-Profile`**将名为的 IAM 角色传递给该实例：

```
aws iam create-instance-profile --instance-profile-name CodeDeployDemo-EC2-Instance-Profile
aws iam add-role-to-instance-profile --instance-profile-name CodeDeployDemo-EC2-Instance-Profile --role-name CodeDeployDemo-EC2-Instance-Profile
```

   如果您需要获取 IAM 实例配置文件的名称，请参阅Amazon CLI 参考中 IAM 部分list-instance-profiles-for中的- role。

现在，您已经创建了一个 IAM 实例配置文件以附加到您的 Amazon EC2 实例。有关更多信息，请参阅《亚马逊 EC2 用户指南》 EC2中的 Amazon IAM 角色。

为您的 Amazon 实例创建 IAM EC2 实例配置文件（控制台）

1. 登录 Amazon Web Services 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择策略，然后选择创建策略。
3. 在指定权限页面上，选择 JSON。
4. 删除示例 JSON 代码。
5. 粘贴以下代码：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 我们建议您将此政策限制为仅适用于您的亚马逊 EC2实例必须访问的 Amazon S3 存储桶。确保允许访问包含 CodeDeploy 代理的 Amazon S3 存储桶。否则，在实例上安装或更新 CodeDeploy 代理时可能会出现错误。例如：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
```

```
                "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
                "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
            ]
        }
    ]
}
```

> **Note**
>
> 如果您想将 IAM 授权或亚马逊虚拟私有云 (VPC) 终端节点 CodeDeploy与一起使用，则需要添加更多权限。有关更多信息，请参阅 CodeDeploy 与 Amazon Virtual Private Cloud 配合使用。

6. 选择下一步。
7. 在审核和创建页面上的策略名称框中，键入 **`CodeDeployDemo-EC2-Permissions`**。
8. （可选）对于描述，键入策略的描述。
9. 选择创建策略。
10. 在导航窗格中，选择 Roles（角色），然后选择 Create role（创建角色）。
11. 在 “用例” 下，选择EC2用例。
12. 选择下一步。
13. 在策略列表中，选中您刚刚创建的策略旁边的复选框（CodeDeployDemo-EC2-权限）。如有必要，请使用搜索框查找该策略。
14. 要使用 Systems Manager 安装或配置 CodeDeploy 代理，请选中亚马逊旁边的复选框 SSMManagedInstanceCore。此 Amazon 托管策略允许实例使用 Systems Manager 服务核心功能。如有必要，请使用搜索框查找该策略。如果您计划使用命令行从公用 Amazon S3 存储桶安装代理，则不需要此策略。了解有关安装 CodeDeploy 代理的详细信息。
15. 选择下一步。
16. 在命名、查看和创建页面上的角色名称中，输入服务角色的名称（例如，**`CodeDeployDemo-EC2-Instance-Profile`**），然后选择创建角色。

    您还可以在角色描述中输入此服务角色的描述。

现在，您已经创建了一个 IAM 实例配置文件以附加到您的 Amazon EC2 实例。有关更多信息，请参阅《亚马逊 EC2 用户指南》 EC2中的 Amazon IAM 角色。

与 CodeDeploy 集成的产品和服务

默认情况下，CodeDeploy 与许多 Amazon 服务以及合作伙伴的产品和服务集成。以下信息可帮助您配置 CodeDeploy 以与您使用的产品和服务集成。

- 与其他 Amazon 服务集成
- 与合作伙伴产品和服务集成
- 来自社区的集成示例

与其他 Amazon 服务集成

CodeDeploy 与下列 Amazon 服务集成：

Amazon CloudWatch ()	Amazon CloudWatch 是一项针对 Amazon 云资源以及您在 Amazon 运行的应用程序的监控服务。Amazon CloudWatch 可用于收集和跟踪指标，收集和监控日志文件以及设置警报。CodeDeploy 支持以下 CloudWatch 工具： • CloudWatch 警报：用于监控您的部署，并在指定的监控指标超出或低于您在 CloudWatch 警报规则中指定的阈值时停止部署。要使用警报监控，请先在 CloudWatch 中设置警报，然后在 CodeDeploy 中将其添加到当警报激活时应停止部署的应用程序或部署组中。 了解更多： • 创建 CloudWatch Logs 警报 • Amazon CloudWatch Events：用于在您的 CodeDeploy 操作中检测实例或部署状态的更改并采取相应的操作。然后 CloudWatch Events 将在部署或实例进入您在规则中指定的状态时，根据您创建的规则调用一个或多个目标操作。 了解更多：

	• 使用 Amazon CloudWatch 事件监控部署 • Amazon CloudWatch Logs：用于监控由 CodeDeploy 代理创建的三种类型的日志，而不必逐一登录到实例。 了解更多： • 在 CloudWatch Logs 控制台中查看 CodeDeploy 日志
Amazon EC2 Auto Scaling	CodeDeploy 支持 Amazon EC2 Auto Scaling。此 Amazon 服务可以根据您指定的标准自动启动 Amazon EC2 实例，例如： • 超出指定的 CPU 利用率的限制。 • 磁盘读取或写入数。 • 在指定时间间隔内的入站或出站网络流量。 您可根据需要扩展一组 Amazon EC2 实例，然后使用 CodeDeploy 自动将应用程序修订部署到这些实例。当不再需要这些 Amazon EC2 实例时，Amazon EC2 Auto Scaling 将终止它们。 了解更多： • CodeDeploy 与 Amazon A EC2 uto Scaling 集成 • 教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组 • 深入剖析：CodeDeploy 和 Auto Scaling 集成

Amazon Elastic Container Service	您可以使用 CodeDeploy 将 Amazon ECS 容器化应用程序部署为任务集。CodeDeploy 通过安装应用程序的更新版本作为新的替换任务集来执行蓝绿部署。CodeDeploy 将生产流量从原始应用程序任务集重新路由到替换任务集。成功部署后，将会终止原始任务集。有关 Amazon ECS 的更多信息，请参阅 Amazon Elastic Container Service。 您可以通过选择 Canary、线性或一次性部署配置，管理部署过程中流量转移到更新后的任务集的方式。有关 Amazon ECS 部署的更多信息，请参阅 Amazon ECS 计算平台上的部署。
Amazon CloudTrail	CodeDeploy 与 Amazon CloudTrail 集成。该服务在您的 Amazon 账户中捕获由 CodeDeploy 或代表它发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 可捕获来自 CodeDeploy 控制台、通过 Amazon CLI 发送的 CodeDeploy 命令或直接来自 CodeDeploy API 的 API 调用。通过使用 CloudTrail 收集的信息，您可以确定以下内容： • 向 CodeDeploy 提出了什么请求。 • 已从中发出请求的源 IP 地址。 • 谁发出了请求。 • 发出请求的时间。 了解更多： • Monitoring Deployments

Amazon Cloud9	Amazon Cloud9 是一个在线的、基于云的集成开发环境（IDE），可用于编写、运行、调试和部署代码（只需使用已连接 Internet 的计算机中的浏览器）。Amazon Cloud9 包含代码编辑器、调试程序、终端和基本工具（如 Amazon CLI 和 Git）。 • 您可以使用 Amazon Cloud9 IDE 运行、调试和构建 GitHub 存储库中的代码。可以使用 IDE 环境窗口和编辑器选项卡查看、更改和保存代码。准备就绪后，可以在 Amazon Cloud9 终端会话中使用 Git 将代码更改推送到您的 GitHub 存储库，然后使用 Amazon CodeDeploy 来部署更新。有关将 Amazon Cloud9 与 GitHub 结合使用的更多信息，请参阅适用于 Amazon Cloud9 的 GitHub 示例。 • 可使用 Amazon Cloud9 IDE 更新 Amazon Lambda 函数。然后，可以使用 Amazon CodeDeploy 创建一个部署，该部署将流量转移到 Amazon Lambda 函数的新版本。有关更多信息，请参阅在 Amazon Cloud9 集成式开发环境（IDE）中使用 Amazon Lambda 函数。 有关 Amazon Cloud9 的更多信息，请参阅什么是 Amazon Cloud9 和 Amazon Cloud9 入门。

Amazon CodePipeline	Amazon CodePipeline 是一种持续交付服务，可用于建模、可视化和自动执行在持续交付过程中发布软件所需的步骤。可以使用 Amazon CodePipeline 定义您自己的发布过程，以便服务在每次发生代码更改时构建、测试和部署代码。例如，一个应用程序可以有三个部署组：Beta、Gamma 和 Prod。您可以设置管道，以便每次源代码发生更改时，将更新逐一部署到每个部署组。 您可以将 Amazon CodePipeline 配置为使用 CodeDeploy 来： • 将代码部署到 Amazon EC2 实例和/或本地实例。 • 无服务器 Amazon Lambda 函数版本。 您可以创建 CodeDeploy 应用程序、部署和部署组，以便在创建管道之前的某个阶段的部署操作中使用，也可以在创建管道向导中使用。 了解更多： • Amazon for DevOps 入门指南 - 了解如何结合使用 CodePipeline 和 CodeDeploy 以将 CodeCommit 存储库中的源代码持续交付和部署到 Amazon EC2 实例。 • 简单管道演练（Amazon S3 存储桶） • 简单的管道演练（CodeCommit 存储库） • 四阶段管道教程

Amazon 无服务器应用程序模型	Amazon 无服务器应用程序模型（Amazon SAM）是定义无服务器应用程序的模型。它能够扩展 Amazon CloudFormation，以提供简化方式来定义无服务器应用程序所需的 Amazon Lambda 函数、Amazon API Gateway API 和 Amazon DynamoDB 表。如果您已使用 Amazon SAM，则可以添加部署首选项，以便开始使用 CodeDeploy 管理 Amazon Lambda 应用程序部署期间的流量转移方式。 有关更多信息，请参阅 Amazon 无服务器应用程序模型。
Elastic Load Balancing	CodeDeploy 支持 Elastic Load Balancing，后者是一项可将传入的应用程序流量自动分配到多个 Amazon EC2 实例的服务。 对于 CodeDeploy 部署，负载均衡器还能阻止流量路由到未就绪的实例，目前正在部署的实例，或环境不再需要的实例。 了解更多： • Integrating CodeDeploy with Elastic Load Balancing

主题

- CodeDeploy 与 Amazon A EC2 uto Scaling 集成
- CodeDeploy 与 ELB 集成

CodeDeploy 与 Amazon A EC2 uto Scaling 集成

CodeDeploy 支持 Amazon A EC2 uto Scaling，这是一项根据您定义的条件自动启动亚马逊 EC2 实例的 Amazon 服务。这些条件可以包括：在指定时间间隔内超出 CPU 利用率、磁盘读写、入站或出站网络流量的限制。当不再需要这些实例时，Amazon A EC2 uto Scaling 会将其终止。有关更多信息，请参阅什么是 Amazon A EC2 uto Scaling？ 在 Amazon A EC2 uto Scaling 用户指南中。

当作为 Amazon Aut EC2 o Scaling 组的一部分启动新的亚马逊 EC2 实例时， CodeDeploy 可以自动将您的修订部署到新实例。您还可以 CodeDeploy使用在 ELB 负载均衡器中注册的 Amazon A EC2 uto Scaling 实例来协调部署。有关更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing和在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器。

> **Note**
>
> 如果您将多个部署组与单个 Amazon A EC2 uto Scaling 组关联起来，则可能会遇到问题。例如，如果一个部署失败，则实例将开始关闭，但已在运行的其他部署可能在一个小时后才超时。有关更多信息，请参阅幕后花絮：避免将多个部署组与单个 Amazon A EC2 uto Scaling 组相关联 CodeDeploy 以及 Amazon A EC2 uto Scaling 集成。

主题

- 将 CodeDeploy 应用程序部署到 Amazon A EC2 uto Scaling 群组
- 在 Auto Scaling 横向缩减事件期间启用终止部署
- Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy
- 将自定义 AMI 与 CodeDeploy Amazon A EC2 uto Scaling 配合使用

将 CodeDeploy 应用程序部署到 Amazon A EC2 uto Scaling 群组

要将 CodeDeploy 应用程序修订版部署到 Amazon A EC2 uto Scaling 群组，请执行以下操作：

1. 创建或找到允许 Amazon A EC2 uto Scaling 组使用 Amazon S3 的 IAM 实例配置文件。有关更多信息，请参阅 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

   > **Note**
   >
   > 您还可以使用 CodeDeploy 将版本从 GitHub 存储库部署到 Amazon A EC2 uto Scaling 群组。尽管 Amazon EC2 实例仍然需要 IAM 实例配置文件，但该配置文件不需要任何其他权限即可从 GitHub 存储库进行部署。

2. 创建或使用 Amazon A EC2 uto Scaling 群组，在启动配置或模板中指定 IAM 实例配置文件。有关更多信息，请参阅在 A mazon EC2 实例上运行的应用程序的 IAM 角色。
3. 创建或找到允许 CodeDeploy 创建包含 Amazon A EC2 uto Scaling 组的部署组的服务角色。

4. 使用 CodeDeploy指定 Amazon A EC2 uto Scaling 组名称、服务角色和其他一些选项创建部署组。有关更多信息，请参阅 为就地部署创建部署组（控制台）或 为就地部署创建部署组（控制台）。
5. 用于 CodeDeploy 将您的修订部署到包含 Amazon A EC2 uto Scaling 组的部署组。

有关更多信息，请参阅 教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组。

在 Auto Scaling 横向缩减事件期间启用终止部署

终止部署是一种在 Auto Scaling 缩减事件发生时自动激活的 CodeDeploy 部署。 CodeDeploy 在 Auto Scaling 服务终止实例之前立即执行终止部署。在终止部署期间， CodeDeploy不部署任何东西。相反，它会生成生命周期事件，您可以将其挂接到自己的脚本以启用自定义关闭功能。例如，您可以将 `ApplicationStop` 生命周期事件挂接到一个脚本，该脚本会在实例被终止之前妥善关闭您的应用程序。

有关终止部署期间 CodeDeploy 生成的生命周期事件的列表，请参阅生命周期事件挂钩可用性。

如果终止部署因任何原因失败， CodeDeploy 则允许继续终止实例。这意味着，即使在完成之前 CodeDeploy没有运行全套（或任何）生命周期事件，该实例也将被关闭。

如果您不启用终止部署，Auto Scaling 服务仍会在发生缩减事件时终止 Amazon EC2 实例，但 CodeDeploy 不会生成生命周期事件。

> **Note**
>
> 无论您是否启用终止部署，如果 Auto Scaling 服务在 CodeDeploy 部署进行期间终止了 Amazon EC2 实例，那么 Auto Scaling 生成的生命周期事件和 CodeDeploy 服务之间都可能出现争用条件。例如，`Terminating`生命周期事件（由 Auto Scaling 服务生成）可能会覆盖该`ApplicationStart`事件（由 CodeDeploy 部署生成）。在这种情况下，您可能会因终止 Amazon EC2 实例或 CodeDeploy 部署而遇到故障。

启用 CodeDeploy 以执行终止部署

- 创建或更新部署组时，选中向 Auto Scaling 组添加终止钩子复选框。有关说明，请参阅 为就地部署创建部署组（控制台） 或 为 EC2 /本地 blue/green 部署创建部署组（控制台）。

  启用此复选框会 CodeDeploy 将 Auto Scaling 生命周期挂钩安装到您在创建或更新 CodeDeploy 部署组时指定的 Auto Scaling 组中。此挂钩称为终止挂钩，用于启用终止部署。

安装完终止挂钩后，横向缩减（终止）事件将按如下方式展开：

1. Auto Scaling 服务（或者简称为 Auto Scaling）确定需要发生缩减事件，并与该 EC2 服务联系以终止实例 EC2。
2. 该 EC2 服务开始终止 EC2 实例。实例进入 `Terminating` 状态，然后进入 `Terminating:Wait` 状态。
3. 在此期间`Terminating:Wait`，Auto Scaling 会运行附加到 Auto Scaling 组的所有生命周期挂钩，包括安装的终止挂钩 CodeDeploy。
4. 终止挂钩会向由轮询的 Amazon SQS 队列发送通知。 CodeDeploy
5. 收到通知后， CodeDeploy 解析消息，执行一些验证，然后执行终止部署。
6. 在终止部署运行期间，每五分钟向 Auto Scaling CodeDeploy 发送一次心跳，让其知道该实例仍在运行中。
7. 到目前为止，该 EC2 实例仍处于`Terminating:Wait`状态（如果您启用了 Auto Scaling 组温池，则可能处于`Warmed:Pending:Wait`状态）。
8. 部署完成后，无论 EC2 终止部署成功还是失败，都会向 Auto Scaling CodeDeploy 指示终止进程。`CONTINUE`

Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy

当您创建或更新 CodeDeploy 部署组以包含 Auto Scaling 组时，请使用 CodeDeploy 服务角色 CodeDeploy访问 Auto Scaling 组，然后将 Auto Sc aling 生命周期挂钩安装到您的 Aut o Scaling 组中。

> **Note**
>
> Auto Scaling 生命周期挂钩不同于本指南生成 CodeDeploy 并在本指南中描述的生命周期事件（也称为生命周期事件挂钩）。AppSpec “挂钩” 部分

CodeDeploy 安装的 Auto Scaling 生命周期挂钩有：

- 启动挂钩 — 此挂钩通知 CodeDeploy Auto S caling 横向扩展事件正在进行中， CodeDeploy 需要开始启动部署。

  在启动部署期间， CodeDeploy：

  - 将您的应用程序修订部署到横向扩展实例。

- 生成生命周期事件以指明部署的进度。您可以将这些生命周期事件挂接到自己的脚本以启用自定义启动功能。有关更多信息，请参阅生命周期事件挂钩可用性中的表。

  启动挂钩和关联的启动部署将始终处于启用状态，无法关闭。

- 终止挂钩 — 此可选挂钩通知 CodeDeploy Auto Scaling 缩减事件正在进行中， CodeDeploy 需要启动终止部署。

  在终止部署期间， CodeDeploy 生成生命周期事件以指示实例关闭的进度。有关更多信息，请参阅在 Auto Scaling 横向缩减事件期间启用终止部署。

主题

- CodeDeploy安装生命周期挂钩后，它们是如何使用的？
- 如何 CodeDeploy 命名 Amazon A EC2 uto Scaling 群组
- 自定义生命周期挂钩事件的执行顺序
- 部署期间的横向扩展事件
- 部署期间的横向缩减事件
- Amazon CloudFormation cfn-init 脚本中的事件顺序

CodeDeploy安装生命周期挂钩后，它们是如何使用的？

安装启动和终止生命周期挂钩后，将分别 CodeDeploy 在 Auto Scaling 组横向扩展和缩小活动期间使用它们。

横向扩展（启动）事件的展开方式如下：

1. Auto Scaling 服务（或者简称为 Auto Scaling）确定需要发生扩展事件，并与该 EC2 服务联系以启动新 EC2实例。
2. 该 EC2 服务启动一个新 EC2 实例。实例进入 `Pending` 状态，然后进入 `Pending:Wait` 状态。
3. 在此期间`Pending:Wait`，Auto Scaling 会运行附加到 Auto Scaling 组的所有生命周期挂钩，包括安装的启动挂钩 CodeDeploy。
4. 启动挂钩会向通过轮询的 Amazon SQS 队列发送通知。 CodeDeploy
5. 收到通知后， CodeDeploy 解析消息，执行一些验证，然后开始启动部署。
6. 在启动部署运行期间，每五分钟向 Auto Scaling CodeDeploy 发送一次心跳，让其知道该实例仍在运行中。

7. 到目前为止，该 EC2 实例仍处于`Pending:Wait`状态。
8. 部署完成后，会向 Auto Scaling CodeDeploy 指示其中一个`CONTINUE`或`ABANDON` EC2 启动过程，具体取决于部署成功还是失败。
   - 如果 CodeDeploy 指示`CONTINUE`，Auto Scaling 将继续启动过程，要么等待其他挂钩完成，要么将实例置于`Pending:Proceed`然后进入`InService`状态。
   - 如果 CodeDeploy 指示`ABANDON`，则 Auto Scaling 会终止 EC2 实例，并在需要时重新启动启动过程，以满足所需的实例数量，如 Auto Scaling 所需容量设置中所定义。

横向缩减（终止）事件的展开方式如下：

请参阅在 Auto Scaling 横向缩减事件期间启用终止部署。

如何 CodeDeploy 命名 Amazon A EC2 uto Scaling 群组

在 EC2 /Onlide 计算平台上 blue/green 部署期间，您可以通过两种方式将实例添加到替换（绿色）环境中：

- 使用已存在或者您手动创建的实例。
- 使用您指定的 Amazon A EC2 uto Scaling 组中的设置在新的 Amazon A EC2 uto Scaling 组中定义和创建实例。

如果您选择第二个选项， CodeDeploy 请为您配置一个新的 Amazon A EC2 uto Scaling 组。它使用以下约定来命名组：

```
CodeDeploy_deployment_group_name_deployment_id
```

例如，如果带有 ID `10` 的部署部署了一个名为的部署组`alpha-deployments`，则会命名为已配置的 Amazon A EC2 uto Scaling 组。`CodeDeploy_alpha-deployments_10`有关更多信息，请参阅为 EC2 /本地 blue/green 部署创建部署组（控制台）和GreenFleetProvisioningOption。

自定义生命周期挂钩事件的执行顺序

您可以将自己的生命周期挂钩添加到要 CodeDeploy 部署的 Amazon A EC2 uto Scaling 群组中。但是，无法根据 CodeDeploy 默认部署生命周期事件预先确定这些自定义生命周期挂钩事件的执行顺序。例如，如果您向 Amazon A EC2 uto Scaling 组添加一个名为`ReadyForSoftwareInstall`的自定义生命周期挂钩，则无法事先知道该挂钩是在第一个默认部署生命周期事件之前还是最后一个 CodeDeploy默认部署生命周期事件之后执行。

要了解如何向 Amazon A EC2 uto Scaling 群组添加自定义生命周期挂钩，请参阅 Ama zon A EC2 uto Scaling 用户指南中的添加生命周期挂钩。

部署期间的横向扩展事件

如果在部署过程中发生 Auto Scaling 横向扩展事件，新实例将使用之前部署的应用程序修订进行更新，而不是最新的应用程序修订。如果部署成功，则旧实例和最近横向扩展的实例将会托管不同的应用程序修订。要更新版本较旧的实例，请 CodeDeploy自动启动后续部署（在第一次部署之后立即启动）以更新所有过时的实例。如果您想更改此默认行为，以便将过时的 EC2 实例保留在较旧的版本中，请参阅Automatic updates to outdated instances。

如果您想在部署过程中暂停 Amazon A EC2 uto Scaling 向外扩展流程，则可以通过`common_functions.sh`脚本中用于进行负载平衡的设置来完成此操作。 CodeDeploy如果`HANDLE_PROCS=true`，则以下 Auto Scaling 事件在部署过程中将自动暂停：

- AZRebalance
- AlarmNotification
- ScheduledActions
- ReplaceUnhealthy

> **Important**
>
> 只有 CodeDeployDefault. OneAtATime 部署配置支持此功能。

有关在使用 `HANDLE_PROCS=true` Amazon A EC2 uto Scaling 时使用避免部署问题的更多信息，请参阅 on 中aws-codedeploy-samples关于处理 AutoScaling 流程的重要通知 GitHub。

部署期间的横向缩减事件

如果 Auto Scaling 组在该 Auto Scaling 组上进行 CodeDeploy 部署时开始缩容，则终止进程（包括终止部署生命周期事件）和 CodeDeploy 终止实例上的其他 CodeDeploy 生命周期事件之间可能会出现争用条件。如果在所有 CodeDeploy 生命周期事件完成之前终止该实例，则在该特定实例上的部署可能会失败。此外，整体 CodeDeploy 部署可能会失败，也可能不会失败，具体取决于您在部署配置中如何设置最低运行状况主机数设置。

Amazon CloudFormation cfn-init 脚本中的事件顺序

如果您使用 `cfn-init`（或 `cloud-init`）在新预配的 Linux 实例上运行脚本，则除非您严格控制实例启动之后的事件发生顺序，否则部署可能会失败。

该顺序必须是：

1. 新预配的实例启动。
2. 所有 `cfn-init` 引导脚本运行直至完成。
3. CodeDeploy 代理启动。
4. 将最新的应用程序修订部署到实例。

如果未仔细控制事件的顺序，则 CodeDeploy 代理可能会在所有脚本完成运行之前开始部署。

要控制事件的顺序，请使用以下最佳实践之一：

- 通过`cfn-init`脚本安装 CodeDeploy 代理，将其放在所有其他脚本之后。
- 将 CodeDeploy 代理包含在自定义 AMI 中并使用`cfn-init`脚本启动它，将其放在所有其他脚本之后。

有关使用 `cfn-init` 的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 cfn-init。

将自定义 AMI 与 CodeDeploy Amazon A EC2 uto Scaling 配合使用

您可以通过两种方式指定在 Amazon A EC2 uto Scaling 组中启动新的亚马逊 EC2 实例时要使用的基本 AMI：

- 您可以指定已安装 CodeDeploy 代理的基本自定义 AMI。由于代理已经安装，因此此选项启动新 Amazon EC2 实例的速度比其他选项更快。但是，此选项使Amazon EC2 实例的初始部署失败的可能性更大，尤其是在 CodeDeploy 代理已过期的情况下。如果您选择此选项，我们建议您定期更新基本自定义 AMI 中的 CodeDeploy 代理。
- 您可以指定一个未安装 CodeDeploy 代理的基本 AMI，并在在 Amazon A EC2 uto Scaling 组中启动每个新实例时安装代理。尽管此选项启动新 Amazon EC2 实例的速度比其他选项慢，但它使实例的初始部署成功的可能性更大。此选项使用最新版本的 CodeDeploy 代理。

CodeDeploy 与 ELB 集成

在 CodeDeploy 部署期间，当互联网流量尚未准备就绪、正在部署到或不再需要作为环境的一部分时，负载均衡器会阻止将互联网流量路由到这些实例。但是，负载均衡器扮演的确切角色取决于它是用于 blue/green 部署还是就地部署。

> **Note**
>
> 在部署中必须使用 Elastic Load Balancing 负载均衡器，在就地 blue/green 部署中是可选的。

弹性负载均衡类型

ELB 提供了三种可用于 CodeDeploy 部署的负载均衡器：经典负载均衡器、应用程序负载均衡器和网络负载均衡器。

Classic 负载均衡器

在传输层进行路由和负载平衡 (TCP/SSL) or the application layer (HTTP/HTTPS)。它支持 VPC。

> **Note**
>
> Amazon ECS 部署不支持经典负载均衡器。

应用程序负载均衡器

路由和负载均衡在应用程序层（HTTP/HTTPS）进行，并支持基于路径的路由。它可以将请求路由到您的虚拟私有云 (VPC) 中每个 EC2实例或容器实例上的端口。

> **Note**
>
> `instance`对于 EC2 实例部署和 Fargate 部署，Application Load Balancer 目标组`IP`的目标类型必须为。有关更多信息，请参阅目标类型。

Network Load Balancer

路由和负载均衡在传输层（TCP/UDP 层，即第 4 层）进行，依据是从 TCP 数据包标头中而非从数据包内容中提取的地址信息。Network Load Balancer 可以处理突发流量，保留客户端的源 IP，并在负载均衡器的使用寿命内使用固定 IP。

要了解有关 ELB 负载均衡器的更多信息，请参阅以下主题：

- 什么是 ELB？
- 什么是经典负载均衡器？
- 什么是应用程序负载均衡器？
- 什么是网络负载均衡器？

蓝/绿部署

在 Elastic Load Balancing 负载均衡器后重新路由实例流量是 CodeDeploy blue/green 部署的基础。

在 blue/green 部署期间，负载均衡器允许根据您指定的规则将流量路由到已部署最新应用程序修订的部署组（替换环境）中的新实例，然后阻止来自运行先前应用程序修订的旧实例（原始环境）的流量。

替换环境中的实例注册一个或多个负载均衡器后，将取消注册原始环境中的实例，并根据您的需要终止。

对于 blue/green 部署，您可以在部署组中指定一个或多个传统负载均衡器、Application Load Balancer 目标组或 Network Load Balancer 目标组。您可以使用 CodeDeploy 控制台或 Amazon CLI 将负载均衡器添加到部署组。

有关 blue/green 部署中的负载均衡器的更多信息，请参阅以下主题：

- 在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器
- 为蓝绿部署创建应用程序（控制台）
- 为 EC2 /本地 blue/green 部署创建部署组（控制台）

就地部署

在就地部署的过程中，负载均衡器可以防止 Internet 流量路由到要部署的实例；实例部署完成后，可恢复对该实例的流量路由。

如果就地部署期间未使用负载均衡器，Internet 流量在部署过程中可能仍会引向该实例。因此，您的客户可能会遇到中断、不完整或过时的 Web 应用程序。当您将 Elastic Load Balancing 负载均衡器与就地部署一起使用时，部署组中的实例将从负载均衡器中注销，使用最新的应用程序版本进行更新，然后在部署成功后作为同一部署组的一部分在负载均衡器中重新注册。 CodeDeploy 将在负载均衡器后等待最多 1 小时让实例恢复正常。如果在等待期间负载均衡器未将该实例标记为运行正常，则根据部署配置， CodeDeploy 要么移动到下一个实例，要么部署失败。

对于就地部署，您可以指定一个或多个经典负载均衡器、应用程序负载均衡器目标组或网络负载均衡器目标组。您可以将负载均衡器指定为部署组配置的一部分，也可以使用提供的脚本 CodeDeploy 来实现负载均衡器。

使用部署组指定就地部署负载均衡器

要将负载均衡器添加到部署组，请使用 CodeDeploy 控制台或 Amazon CLI。有关就地部署期间在部署组中指定负载均衡器的信息，请参阅以下主题：

- 为就地部署创建应用程序（控制台）
- 为就地部署创建部署组（控制台）
- 在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器

使用脚本指定就地部署负载均衡器

通过执行以下过程中的步骤，使用部署生命周期脚本为就地部署设置负载均衡。

> **Note**
>
> 你应该使用 CodeDeployDefault. OneAtATime 只有当您使用脚本为就地部署设置负载均衡器时，才会进行部署配置。不支持并发运行，并且 CodeDeployDefault. OneAtATime 设置可确保脚本的串行执行。有关部署配置的更多信息，请参阅在中使用部署配置 CodeDeploy。

在的 CodeDeploy 示例存储库中 GitHub，我们提供了您可以调整以使用 CodeDeploy ELB 负载均衡器的说明和示例。这些存储库包含三个示例脚本（`register_with_elb.sh`、`deregister_from_elb.sh` 和 `common_functions.sh`），这些脚本提供了开始操作所需的全部代码。只需编辑这 3 个脚本中的占位符，然后从 `appspec.yml` 文件中引用这些脚本。

要使用在 ELB 负载均衡 CodeDeploy 器中注册的 Amazon EC2 实例设置就地部署，请执行以下操作：

1. 下载要用于就地部署的负载均衡器的类型的示例：
   - 经典负载均衡器
   - 应用程序负载均衡器或网络负载均衡器（两种类型可以使用同一脚本）
2. 确保您的每个目标 Amazon EC2 实例都 Amazon CLI 安装了。
3. 确保您的每个目标 Amazon EC2 实例都有一个 IAM 实例配置文件，其中至少附有 elasticloadbalancing: * 和 autoScaling: * 权限。
4. 将部署生命周期事件脚本（`register_with_elb.sh`、`deregister_from_elb.sh` 和 `common_functions.sh`）包含在应用程序的源代码目录中。
5. 在`appspec.yml`应用程序修订版中，提供有关 CodeDeploy 在活动期间运行`register_with_elb.sh`脚本和在ApplicationStart活动期间运行`deregister_from_elb.sh`脚本的ApplicationStop说明。
6. 如果该实例是 Amazon A EC2 uto Scaling 组的一部分，则可以跳过此步骤。

   在 `common_functions.sh` 脚本中：
   - 如果您使用的是 Cl assic Load Balancer，请在中指定 ELB 负载均衡器的名称`ELB_LIST=""`，并对文件中的其他部署设置进行所需的更改。
   - 如果您使用的是 App lication Loa d Balancer 或 Network Load Balancer`TARGET_GROUP_LIST=""`，请在中指定 ELB 目标组名称的名称，并对文件中的其他部署设置进行所需的更改。
7. 将应用程序的源代码、`appspec.yml` 和部署生命周期事件脚本绑定到一个应用程序修订中，然后上传该修订。将修订版部署到 Amazon EC2 实例。在部署期间，部署生命周期事件脚本将向负载均衡器注销 Amazon EC2 实例，等待连接耗尽，然后在部署完成后向负载均衡器重新注册 Amazon EC2 实例。

与合作伙伴产品和服务集成

CodeDeploy 内置集成了下列合作伙伴产品和服务：

Ansible	如果您已有一套 Ansible 操作手册，但只需在某个位置运行它们，那么借助适用于 Ansible 和 CodeDeploy 的模板，即可演示如何使用几个简单的部署挂钩来确保 Ansible 在本地部署实例上可用并运行这些操作手册。如果您已有关于构建

	和维护清单的过程，那么也可以使用 Ansible 模块来安装和运行 CodeDeploy 代理。 了解更多： • Ansible 和 CodeDeploy
Atlassian – Bamboo 和 Bitbucket	适用于 Bamboo 的 CodeDeploy 任务将包含 AppSpec 文件的目录压缩为 .zip 文件，将该文件上传到 Amazon S3，然后根据 CodeDeploy 应用程序中提供的配置开始部署。 CodeDeploy 的 Atlassian Bitbucket 支持使您能够直接从 Bitbucket UI 按需将代码推送到 Amazon EC2 实例（推送到任何部署组）。这意味着，在您更新 Bitbucket 存储库中的代码之后，您不必登录到持续集成（CI）平台或 Amazon EC2 实例即可运行手动部署过程。 了解更多： • 使用适用于 Bamboo 的 CodeDeploy 任务 • 宣布 Atlassian Bitbucket 支持 CodeDeploy
Chef	Amazon 提供两个有关将 Chef 与 CodeDeploy 集成的模板示例。第一个是将安装并启动 CodeDeploy 代理的 Chef 说明书。这使您能够在使用 CodeDeploy 的同时，使用 Chef 继续管理您的主机基础设施。第二个示例模板演示如何使用 CodeDeploy 通过 chef-solo 在每个节点上协调说明书和方法的运行。 了解更多： • Chef 和 CodeDeploy

CircleCI	CircleCI 提供了一个自动测试和持续集成以及部署工具集。在 Amazon 中创建 IAM 角色以用于 CircleCI 并在 circle.yml 文件中配置部署参数之后，可以结合使用 CircleCI 和 CodeDeploy 来创建应用程序修订，将其上传到 Amazon S3 存储桶，然后启动并监控您的部署。 了解更多： • 使用 CircleCI Orb 将应用程序部署到 Amazon CodeDeploy
CloudBees	您可以使用 CloudBees DEV@cloud 上提供的 CodeDeploy Jenkins 插件作为构建后操作。例如，在持续交付管道结束时，可以使用它向服务器队列部署应用程序修订。 了解更多： • DEV@cloud 上现提供 CodeDeploy Jenkins 插件
Codeship	可以使用 Codeship 通过 CodeDeploy 部署应用程序修订。您可以使用 Codeship UI 将 CodeDeploy 添加到分支的部署管道中。 了解更多： • 部署到 CodeDeploy • 在 Codeship 上进行 CodeDeploy 集成

GitHub	可以使用 CodeDeploy 从 GitHub 存储库部署应用程序修订。只要 GitHub 存储库中的源代码发生了更改，您就可以触发从该存储库的部署。 了解更多： • 将 CodeDeploy 与 GitHub 集成 • 教程：使用 CodeDeploy 从 GitHub 部署应用程序 • 使用 CodeDeploy 自动从 GitHub 部署
HashiCorp Consul	可以使用开源 HashiCorp Consul 工具来帮助确保应用程序环境在 CodeDeploy 中部署应用程序时的运行状况和稳定性。可以使用 Consul 注册要在部署期间发现的应用程序，将应用程序和节点置于维护模式中以将其从部署中排除，并在目标实例变得运行状况不佳时停止部署。 了解更多： • 使用 HashiCorp Consul 执行的 CodeDeploy 部署
Jenkins	CodeDeploy Jenkins 插件提供 Jenkins 项目的构建后步骤。成功构建后，它将压缩工作区，上传到 Amazon S3，并启动新的部署。 了解更多： • CodeDeploy Jenkins 插件 • 为 CodeDeploy 设置 Jenkins 插件

Puppet Labs	Amazon 提供 Puppet 和 CodeDeploy 的示例模板。第一个是将安装并启动 CodeDeploy 代理的 Puppet 模块。这使您能够在使用 CodeDeploy 的同时，使用 Puppet 继续管理您的主机基础设施。第二个示例模板演示如何使用 CodeDeploy 通过无主 Puppet 在每个节点上协调模块和清单的运行。 了解更多： • Puppet 和 CodeDeploy
SaltStack	您可以将 SaltStack 基础设施与 CodeDeploy 集成。您可以使用 CodeDeploy 模块在自己的设备上安装并运行 CodeDeploy 代理，也可以使用 CodeDeploy 通过几个简单的部署挂钩协调 Salt States 的运行。 了解更多： • SaltStack 和 CodeDeploy
TeamCity	可以使用 CodeDeploy Runner 插件直接从 TeamCity 部署应用程序。此插件添加一个 TeamCity 构建步骤，此步骤将准备应用程序修订并将它上传到 Amazon S3 存储桶，注册 CodeDeploy 应用程序中的修订，创建 CodeDeploy 部署以及（如果选择）等待部署完成。 了解更多： • CodeDeploy Runner（下载） • CodeDeploy Runner 插件（文档）

Travis CI	您可以将 Travis CI 配置为在成功构建后触发 CodeDeploy 中的部署。 了解更多： • Travis CI 和 CodeDeploy 部署

主题

- 将 CodeDeploy 与 GitHub 集成

将 CodeDeploy 与 GitHub 集成

CodeDeploy 支持 GitHub，这是一项基于 Web 的代码托管和共享服务。CodeDeploy 可以将存储在 GitHub 存储库或 Amazon S3 存储桶中的应用程序修订部署到实例。CodeDeploy 仅在 EC2/本地部署中支持 GitHub。

主题

- 从 GitHub 部署 CodeDeploy 修订
- 使用 CodeDeploy 实现 GitHub 行为

从 GitHub 部署 CodeDeploy 修订

要将应用程序修订从 GitHub 存储库部署到实例，请执行以下步骤：

1. 创建与 CodeDeploy 和将部署到的 Amazon EC2 实例类型兼容的修订。

   要创建兼容版本，请按照为 CodeDeploy 规划修订和将应用程序规范文件添加到 CodeDeploy 的修订中的说明执行操作。
2. 使用 GitHub 账户将您的修订添加到 GitHub 存储库。

   要创建 GitHub 账户，请参阅联接 GitHub。要创建 GitHub 存储库，请参阅创建存储库。
3. 使用 CodeDeploy 控制台中的创建部署页面或 Amazon CLI create-deployment 命令将您的修订从 GitHub 存储库部署到配置为在 CodeDeploy 部署中使用的目标实例。

   如果您需要调用 create-deployment 命令，则必须先使用此控制台的创建部署页，针对指定应用程序为 CodeDeploy 授予代表您的首选 GitHub 账户与 GitHub 交互的权限。每个应用程序只需进行一次这样的操作。

要了解如何使用 Create deployment 页从 GitHub 存储库进行部署，请参阅使用创建部署 CodeDeploy。

要了解如何调用 create-deployment 命令从 GitHub 存储库进行部署，请参阅创建 EC2 /本地计算平台部署 (CLI)。

要了解如何准备实例以在 CodeDeploy 部署中使用，请参阅使用适用于 CodeDeploy 的实例。

有关更多信息，请参阅 教程：使用 CodeDeploy 从 GitHub 部署应用程序。

使用 CodeDeploy 实现 GitHub 行为

主题

- GitHub 对 CodeDeploy 中的应用程序进行的身份验证
- CodeDeploy 与私有和公有 GitHub 存储库进行交互
- CodeDeploy 与组织托管的 GitHub 存储库进行交互
- 使用 CodeDeploy 从 CodePipeline 自动部署

GitHub 对 CodeDeploy 中的应用程序进行的身份验证

在向 CodeDeploy 提供与 GitHub 交互的权限后，GitHub 账户和应用程序之间的关联将存储在 CodeDeploy 中。您可以将应用程序链接到其他 GitHub 账户。也可以撤消 CodeDeploy 与 GitHub 交互的权限。

将 GitHub 账户链接到 CodeDeploy 中的应用程序

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 选择要链接到其他 GitHub 账户的应用程序。
4. 如果您的应用程序没有部署组，请选择创建部署组来创建一个部署组。有关更多信息，请参阅 使用创建部署组 CodeDeploy。需要部署组在下一步中选择创建部署。

5. 从部署中，选择创建部署。

> **Note**
>
> 您无需创建新的部署。这是目前将其他 GitHub 账户链接到应用程序的唯一方式。

6. 在部署设置中，对于修订类型，选择我的应用程序将存储在 GitHub 中。
7. 请执行以下操作之一：
   - 要为 Amazon CodeDeploy 应用程序创建与 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。在 GitHub 令牌名称中，键入一个名称来标识此连接，然后选择连接到 GitHub。该网页提示您授权 CodeDeploy 与您应用程序的 GitHub 进行交互。继续执行步骤 10。
   - 要使用已创建的连接，请在 GitHub 令牌名称中，选择其名称，然后选择连接到 GitHub。继续执行步骤 8。
   - 要创建与其他 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。在 GitHub 令牌名称中，键入一个名称来标识此连接，然后选择连接到 GitHub。该网页提示您授权 CodeDeploy 与您应用程序的 GitHub 进行交互。继续执行步骤 10。
8. 如果您尚未登录 GitHub，请按照 Sign in 页上的说明执行操作以使用要将应用程序链接到的 GitHub 账户进行登录。
9. 选择授权应用程序。GitHub 向 CodeDeploy 提供代表所选应用程序的已登录 GitHub 账户与 GitHub 交互的权限。
10. 如果您不需要创建部署，请选择 Cancel。

撤消 CodeDeploy 与 GitHub 交互的权限

1. 使用要撤销 Amazon CodeDeploy 权限的 GitHub 账户的凭据登录 GitHub。
2. 打开 GitHub 应用程序页，在已授权应用程序列表中找到 CodeDeploy，然后按照用于撤消应用程序授权的 GitHub 过程执行操作。

CodeDeploy 与私有和公有 GitHub 存储库进行交互

CodeDeploy 支持从私有和公有 GitHub 存储库部署应用程序。当您向 CodeDeploy 授予代表您访问 GitHub 的权限后，CodeDeploy 针对您的 GitHub 账户有权访问的所有私有 GitHub 存储库都将拥有读写访问权。不过，CodeDeploy 只能从 GitHub 存储库进行读取。它将不会对任何私有 GitHub 存储库进行写入。

CodeDeploy 与组织托管的 GitHub 存储库进行交互

默认情况下，组织托管的 GitHub 存储库（与账户自己的私有或公有存储库相对）未授予对第三方应用程序（包括 CodeDeploy）的访问权。如果在 GitHub 中启用组织的第三方应用程序限制，并且您尝试从其 GitHub 存储库中部署代码，则部署将失败。可通过两种方式解决此问题。

- 作为组织成员，您可以要求组织所有者批准对 CodeDeploy 的访问权。请求此访问权的步骤取决于您是否已授权单个账户访问 CodeDeploy：
  - 如果您已在账户中授予对 CodeDeploy 的访问权，请参阅请求组织批准您的授权应用程序。
  - 如果您尚未在账户中授予对 CodeDeploy 的访问权，请参阅请求组织批准第三方应用程序。
- 组织所有者可禁用组织的所有第三方应用程序限制。有关信息，请参阅禁用组织的第三方应用程序限制。

有关更多信息，请参阅关于第三方应用程序限制。

使用 CodeDeploy 从 CodePipeline 自动部署

当源代码发生更改时，您可以触发从 CodePipeline 进行的部署。有关更多信息，请参阅 CodePipeline。

来自社区的集成示例

以下各部分提供的链接指向博客文章、文章和社区提供的示例。

> **Note**
>
> 提供的这些链接仅供参考，不应视为全面列表或支持示例内容。Amazon 对这些内容或外部内容的准确性不承担责任。

博客帖子

- 在 Amazon CloudFormation 中自动配置 CodeDeploy

  了解如何使用 Amazon CloudFormation 在 CodeDeploy 中预置应用程序部署。

  发布时间：2016 年 1 月
- Amazon Toolkit for Eclipse 与 CodeDeploy 集成（第 1 部分）

Amazon Toolkit for Eclipse 与 CodeDeploy 集成（第 2 部分）

Amazon Toolkit for Eclipse 与 CodeDeploy 集成（第 3 部分）

了解 Java 开发人员如何使用适用于 Eclipse 的 CodeDeploy 插件，直接从 Eclipse 开发环境将 Web 应用程序部署到 Amazon。

发布时间：2015 年 2 月

- 使用 CodeDeploy 自动从 GitHub 部署

  了解如何利用从 GitHub 自动部署到 CodeDeploy 的过程来创建从源代码控制到测试环境或生产环境的端到端管道。

  发布时间：2014 年 12 月

CodeDeploy 教程

本部分包含的一些教程可帮助您了解如何使用 CodeDeploy。

这些教程中的过程提供了建议的文件存储位置（例如，c:\temp）和建议的存储桶、子文件夹或文件的名称（例如，分别为 amzn-s3-demo-bucket、HelloWorldApp 和 CodeDeployDemo-EC2-Trust.json），但您并不是必须使用它们。在执行这些过程时，请确保替换您的文件位置和名称。

主题

- 教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix）
- 教程：使用 CodeDeploy 部署“Hello, World!” 带有 CodeDeploy（Windows 服务器）的应用程序
- 教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）
- 教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组
- 教程：使用 CodeDeploy 从 GitHub 部署应用程序
- 教程：将应用程序部署到 Amazon ECS
- 教程：部署具有验证测试的 Amazon ECS 服务
- 教程：使用无 Amazon 服务器应用程序模型部署更新的 Lambda 函数 CodeDeploy

教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix）

在本教程中，您将 WordPress（一种基于 PHP 和 MySQL 的开源博客工具和内容管理系统）部署到运行 Amazon Linux 或 Red Hat Enterprise Linux（RHEL）的单个 Amazon EC2 实例。

不是您要找的内容？

- 要改为部署到运行 Windows Server 的 Amazon EC2 实例，请参阅教程：使用 CodeDeploy 部署“Hello, World!” 带有 CodeDeploy（Windows 服务器）的应用程序。
- 要部署到本地实例而不是 Amazon EC2 实例，请参阅教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）。

本教程的步骤是从运行 Linux、macOS 或 Unix 的本地开发计算机的角度提供的。虽然您可以在运行 Windows 的本地计算机上完成其中的大部分步骤，但您必须适应涵盖命令（如 chmod 和 wget）、应用程序（如 sed）和目录路径（例如 `/tmp`）的步骤。

在开始本教程前，您必须完成入门 CodeDeploy中的先决条件。这包括配置用户、安装或升级 Amazon CLI、创建 IAM 实例配置文件和服务角色。

主题

- 步骤 1：启动和配置 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例
- 步骤 2：将源内容配置为部署到 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例
- 步骤 3：将您的 WordPress 应用程序上传到 Amazon S3
- 步骤 4：部署 WordPress 应用程序
- 步骤 5：更新和重新部署您的 WordPress 应用程序
- 步骤 6：清理 WordPress 应用程序和相关资源

步骤 1：启动和配置 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例

要使用 CodeDeploy 部署 WordPress 应用程序，您需要一个运行 Amazon Linux 或 Red Hat Enterprise Linux（RHEL）的 Amazon EC2 实例。Amazon EC2 实例需要允许 HTTP 连接的新入站安全规则。此规则是为了在成功部署后，通过浏览器查看 WordPress 页面。

按照为 CodeDeploy 创建 Amazon EC2 实例中的说明进行操作。当您进入这些说明中关于向实例分配 Amazon EC2 实例标签的部分时，请确保指定标签键 **`Name`** 和标签值 **`CodeDeployDemo`**。（如果您指定不同的标签密钥或标签值，则步骤 4：部署 WordPress 应用程序中的说明可能会产生意外结果。）

在您按照说明启动 Amazon EC2 实例之后，请返回到此页，并继续下一部分。请勿继续使用创建应用程序 CodeDeploy作为下一步骤。

连接到 Amazon Linux 或 RHEL Amazon EC2 实例

启动新的 Amazon EC2 实例之后，请按照下面的说明执行连接操作。

1. 使用 ssh 命令（或支持 SSH 的终端仿真器，如 PuTTY）连接到 Amazon Linux 或 RHEL Amazon EC2 实例。您将需要实例的公有 DNS 地址以及您在启动 Amazon EC2 实例时使用的密钥对的私有密钥。有关更多信息，请参阅连接到实例。

例如，如果公有 DNS 地址为 **ec2-01-234-567-890.compute-1.amazonaws.com**，并且用于 SSH 访问的 Amazon EC2 实例密钥对名为 **codedeploydemo.pem**，您可键入：

```
ssh -i /path/to/codedeploydemo.pem ec2-
user@ec2-01-234-567-890.compute-1.amazonaws.com
```

将 `/path/to/codedeploydemo.pem` 替换为 `.pem` 文件的路径，并将示例 DNS 地址替换为 Amazon Linux 或 RHEL Amazon EC2 实例的地址。

> **Note**
>
> 如果您收到关于密钥文件的权限太开放的错误，您将需要限制其权限，仅向当前用户（您）授予访问权限。例如，在 Linux、macOS 或 Unix 上使用 chmod 命令时，键入：

```
chmod 400 /path/to/codedeploydemo.pem
```

2. 登录后，您将会看到 Amazon EC2 实例的 AMI 横幅。对于 Amazon Linux，应如下所示：

```
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
```

3. 您现在可以从运行的 Amazon EC2 实例注销。

> **Warning**
>
> 请不要停止或终止 Amazon EC2 实例。否则，CodeDeploy 将不能部署到该实例。

添加入站规则，允许 HTTP 流量指向您的 Amazon Linux 或 RHEL Amazon EC2 实例

下一步是确认您的 Amazon EC2 实例具有开放的 HTTP 端口，以便您可以在浏览器中查看已部署 WordPress 应用程序的主页。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 选择实例，然后选择您的实例。
3. 在描述选项卡上的安全组下，选择查看入站规则。

   您应在安全组中看到类似如下的规则列表：

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports     Protocol     Source     launch-wizard-N
    22        tcp          0.0.0.0/0          #
   ```

4. 在安全组下，为您的 Amazon EC2 实例选择安全组。其可能被命名为 **launch-wizard-*N***。名称中的 ***N*** 是创建实例时分配到您安全组的编号。

   选择入站选项卡。如果实例的安全组配置正确，则应看到一条具有以下值的规则：

   - 类型：HTTP
   - 协议：TCP
   - 端口范围：80
   - 来源：0.0.0.0/0
5. 如果您没有看到包含这些值的规则，请使用向安全组添加规则中的过程将其添加到新的安全规则中。

步骤 2：将源内容配置为部署到 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例

现在是时候配置应用程序的源内容以拥有可部署到实例的内容了。

主题

- 获取源代码
- 创建脚本以运行应用程序
- 添加应用程序规范文件

获取源代码

在此教程中，您将 WordPress 内容发布平台从开发计算机部署到目标 Amazon EC2 实例。要获取 WordPress 源代码，可使用内置命令行调用。或者，如果您的开发计算机上已安装 Git，可改用 Git。

对于这些步骤，我们假定您将 WordPress 源代码的副本下载到开发计算机上的 `/tmp` 目录。（您可以选择所需的任何目录，但请记住，将这些步骤中指定的任何 `/tmp` 替换为您的位置。）

选择以下两个选项之一，将 WordPress 源文件复制到您的部署计算机上。第一个选项使用内置命令行调用。第二个选项使用 Git。

主题

- 获取 WordPress 源代码的副本（内置命令行调用）
- 获取 WordPress 源代码的副本（Git）

获取 WordPress 源代码的副本（内置命令行调用）

1. 调用 wget 命令可将 WordPress 源代码的副本以 .zip 文件的形式下载到当前目录：

```
wget https://github.com/WordPress/WordPress/archive/master.zip
```

2. 调用 unzip、mkdir、cp 和 rm 命令可执行以下任务：

   - 将 `master` .zip 文件解压缩到 `/tmp/WordPress_Temp` 目录（文件夹）。
   - 将其解压缩的内容复制到 `/tmp/WordPress` 目标文件夹中。
   - 删除临时 `/tmp/WordPress_Temp` 文件夹和 `master` 文件。

   运行这些命令（一次运行一条命令）：

```
unzip master -d /tmp/WordPress_Temp
```

```
mkdir -p /tmp/WordPress
```

```
cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
```

```
rm -rf /tmp/WordPress_Temp
```

```
rm -f master
```

这将在 `/tmp/WordPress` 文件夹中保留一组干净的 WordPress 源代码文件。

获取 WordPress 源代码的副本（Git）

1. 在您的开发计算机上下载并安装 Git。
2. 在 `/tmp/WordPress` 文件夹中，调用 git init 命令。
3. 调用 git clone 命令可克隆公有 WordPress 存储库，并在 `/tmp/WordPress` 目标文件夹中创建此存储库的副本：

```
git clone https://github.com/WordPress/WordPress.git /tmp/WordPress
```

这将在 `/tmp/WordPress` 文件夹中保留一组干净的 WordPress 源代码文件。

创建脚本以运行应用程序

接下来，在目录中创建一个文件夹和一些脚本。CodeDeploy 使用这些脚本在目标 Amazon EC2 实例上设置和部署应用程序修订。您可使用任何文本编辑器来创建脚本。

1. 在 WordPress 源代码的副本中创建脚本目录：

```
mkdir -p /tmp/WordPress/scripts
```

2. 在 `install_dependencies.sh` 中创建一个 `/tmp/WordPress/scripts` 文件。将以下行添加到该文件中。此 `install_dependencies.sh` 脚本安装 Apache、MySQL 和 PHP。还会将 MySQL 支持添加到 PHP 中。

```
#!/bin/bash
sudo amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

3. 在 `start_server.sh` 中创建一个 `/tmp/WordPress/scripts` 文件。将以下行添加到该文件中。此 `start_server.sh` 脚本启动 Apache 和 MySQL。

```
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

4. 在 `stop_server.sh` 中创建一个 `/tmp/WordPress/scripts` 文件。将以下行添加到该文件中。此 `stop_server.sh` 脚本停止 Apache 和 MySQL。

```
#!/bin/bash
isExistApp="pgrep httpd"
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service

fi
```

5. 在 `create_test_db.sh` 中创建一个 `/tmp/WordPress/scripts` 文件。将以下行添加到该文件中。此 `create_test_db.sh` 脚本使用 MySQL 创建 **`test`** 数据库以供 WordPress 使用。

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

6. 最后，在 `change_permissions.sh` 中创建 `/tmp/WordPress/scripts` 脚本。这将用于更改 Apache 中的文件夹权限。

> **Important**
>
> 此脚本更新 `/tmp/WordPress` 文件夹上的权限，以便所有人可以写入其中。这是必需的，以便 WordPress 可在步骤 5：更新和重新部署您的 WordPress 应用程序期间写入其数据库。设置 WordPress 应用程序之后，运行以下命令，将权限更新为更安全的设置：

```
chmod -R 755 /var/www/html/WordPress
```

```
#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

7. 为所有脚本提供可执行文件权限。在命令行上，键入：

```
chmod +x /tmp/WordPress/scripts/*
```

添加应用程序规范文件

接下来，将应用程序规范文件（AppSpec 文件）（CodeDeploy 使用的 YAML 格式文件）添加到：

- 将应用程序修订中的源文件映射到其在目标 Amazon EC2 实例上的目标。
- 为部署的文件指定自定义权限。
- 指定部署期间要在目标 Amazon EC2 实例上运行的脚本。

AppSpec 文件必须命名为 `appspec.yml`。它必须放置在应用程序源代码的根目录中。在本教程中，根目录为 `/tmp/WordPress`

使用文本编辑器创建一个名为 `appspec.yml` 的文件。将以下行添加到该文件中：

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
```

```
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

CodeDeploy 使用此 AppSpec 文件将开发计算机上的 `/tmp/WordPress` 文件夹中的所有文件复制到目标 Amazon EC2 实例上的 `/var/www/html/WordPress` 文件夹中。在部署过程中，CodeDeploy 会在部署生命周期内的指定事件中作为 `root`（位于目标 Amazon EC2 实例上的 `/var/www/html/WordPress/scripts` 文件夹中）运行指定脚本，例如 **`BeforeInstall`** 和 **`AfterInstall`**。如果这些脚本中的任意脚本的运行时间超过 300 秒（5 分钟），CodeDeploy 会停止部署并将部署标记为失败。

有关这些设置的更多信息，请参阅 CodeDeploy AppSpec 文件参考。

> **Important**
>
> 此文件中各项之间的空格的位置和数量很重要。如果间距不正确，CodeDeploy 会引发可能难以调试的错误。有关更多信息，请参阅 AppSpec 文件间距。

步骤 3：将您的 WordPress 应用程序上传到 Amazon S3

现在，您将准备源内容并将其上传到一个位置（CodeDeploy 可从该位置部署源内容）。以下说明向您演示如何预置 Amazon S3 存储桶、为存储桶准备应用程序修订的文件、对修订的文件打包，然后将修订推送到存储桶。

> **Note**
>
> 虽然本教程未涵盖此内容，但您可使用 CodeDeploy 将应用程序从 GitHub 存储库部署到实例。有关更多信息，请参阅 将 CodeDeploy 与 GitHub 集成。

主题

- 预置 Amazon S3 存储桶
- 为存储桶准备应用程序的文件
- 将应用程序的文件打包到单个存档文件并推送此存档文件

预置 Amazon S3 存储桶

在 Amazon S3 中创建存储容器或存储桶，或者使用现有的存储桶。确保您可将修订上传到存储桶，并确保部署中使用的 Amazon EC2 实例可从存储桶下载修订。

您可以使用 Amazon CLI、Amazon S3 控制台或 Amazon S3 API 创建 Amazon S3 存储桶。创建存储桶后，请确保提供对存储桶和您的 Amazon 账户的访问权限。

> **Note**
>
> 对于所有 Amazon 账户，存储桶名称在 Amazon S3 中必须是唯一的。如果您无法使用 **amzn-s3-demo-bucket**，请尝试其他存储桶名称，例如 **amzn-s3-demo-bucket** 后跟短划线和您的姓名首字母或其他某个唯一标识符。之后，请确保将此教程中的任何 **amzn-s3-demo-bucket** 替换为您的存储桶名称。
> 必须在启动目标 Amazon EC2 实例的同一 Amazon 区域创建 Amazon S3 存储桶。例如，如果您在美国东部（弗吉尼亚州北部）区域创建存储桶，则您的目标 Amazon EC2 实例必须在美国东部（弗吉尼亚州北部）区域启动。

主题

- 创建 Amazon S3 存储桶（CLI）
- 创建 Amazon S3 存储桶（控制台）
- 对 Amazon S3 存储桶和 Amazon 账户授予权限

创建 Amazon S3 存储桶（CLI）

调用 mb 命令以创建一个名为 **amzn-s3-demo-bucket** 的 Amazon S3 存储桶：

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

创建 Amazon S3 存储桶（控制台）

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在 Amazon S3 控制台中，选择创建存储桶。
3. 在 Bucket name 框中，键入存储桶的名称。
4. 在 Region 列表中，选择目标区域，然后选择 Create。

对 Amazon S3 存储桶和 Amazon 账户授予权限

您必须拥有上传到 Amazon S3 存储桶的权限。您可以通过 Amazon S3 存储桶策略指定这些权限。例如，在以下 Amazon S3 存储桶策略中，使用通配符（*）允许 Amazon 账户 `111122223333` 将文件上传到名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任何目录：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

要查看您的 Amazon 账户 ID，请参阅查找您的 Amazon 账户 ID。

现在是验证 Amazon S3 存储桶是否允许来自每个参与的 Amazon EC2 实例的下载请求的好时机。您可以通过 Amazon S3 存储桶策略来指定这一点。例如，在下面的 Amazon S3 存储桶策略中，使用通配符（*）允许任何附有包含 ARN `arn:aws:iam::444455556666:role/CodeDeployDemo` 的 IAM 实例配置文件的 Amazon EC2 实例，从名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任何目录下载文件：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
```

```
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

有关如何生成和附加 Amazon S3 存储桶策略的信息，请参阅存储桶策略示例。

有关如何创建和附加 IAM policy 的信息，请参阅使用策略。

为存储桶准备应用程序的文件

请确保 WordPress 应用程序文件 AppSpec 文件和脚本在您的开发计算机上的组织结构如下：

```
/tmp/
  |--WordPress/
      |-- appspec.yml
      |-- scripts/
      |    |-- change_permissions.sh
      |    |-- create_test_db.sh
      |    |-- install_dependencies.sh
      |    |-- start_server.sh
      |    |-- stop_server.sh
      |-- wp-admin/
      |    |-- (various files...)
      |-- wp-content/
      |    |-- (various files...)
      |-- wp-includes/
      |    |-- (various files...)
      |-- index.php
      |-- license.txt
      |-- readme.html
      |-- (various files ending with .php...)
```

将应用程序的文件打包到单个存档文件并推送此存档文件

将 WordPress 应用程序文件和 AppSpec 文件打包到存档文件（称为应用程序修订）中。

> **Note**
>
> 将对象存储在存储桶中以及将应用程序修订传入和传出存储桶可能需支付费用。有关更多信息，请参阅 Amazon S3 定价。

1. 在开发计算机上，切换到这些文件存储到的文件夹：

```
cd /tmp/WordPress
```

> **Note**
>
> 如果您未切换到此文件夹，则将在您的当前文件夹中启动文件打包。例如，如果您当前的文件夹是 `/tmp` 而非 `/tmp/WordPress`，则打包操作将从 `tmp` 文件夹中的文件和子文件夹开始，这可能包括 `WordPress` 子文件夹之外的内容。

2. 调用 create-application 命令可注册名为 **`WordPress_App`** 的新应用程序：

```
aws deploy create-application --application-name WordPress_App
```

3. 调用 CodeDeploy push 命令可将文件捆绑在一起，将修订上传到 Amazon S3，并向 CodeDeploy 注册有关已上传修订的信息，所有这些只需一个操作即可完成。

```
aws deploy push \
  --application-name WordPress_App \
  --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
  --ignore-hidden-files
```

此命令将当前目录中的文件（不包括任何隐藏文件）捆绑到一个名为 **`WordPressApp.zip`** 的存档文件中，将修订上传到 **`amzn-s3-demo-bucket`** 存储桶，然后向 CodeDeploy 注册有关已上传修订的信息。

步骤 4：部署 WordPress 应用程序

现在，您部署已上传到 Amazon S3 的示例 WordPress 应用程序修订。您可以使用 Amazon CLI 或 CodeDeploy 控制台部署修订并监视部署进度。成功部署应用程序修订之后，可以检查结果。

主题

- 使用 CodeDeploy 部署您的应用程序修订
- 监控您的部署并排除故障
- 验证您的部署

使用 CodeDeploy 部署您的应用程序修订

使用 Amazon CLI 或控制台部署您的应用程序修订。

主题

- 部署您的应用程序修订（CLI）
- 部署应用程序修订（控制台）

部署您的应用程序修订（CLI）

1. 部署需要部署组。不过，在创建部署组之前，您需要服务角色 ARN。服务角色是 IAM 角色，该角色授予某个服务代表您执行操作的权限。在这种情况下，服务角色将为 CodeDeploy 提供访问您的 Amazon EC2 实例的权限，以扩展（读取）其 Amazon EC2 实例标签。

   您应该已经按照创建服务角色（CLI）中的说明创建了服务角色。要获取服务角色的 ARN，请参阅获取服务角色 ARN（CLI）。

2. 现在您已拥有此服务角色 ARN，请调用 create-deployment-group 命令，使用名为 **CodeDeployDemo** 的 Amazon EC2 标签和名为 **CodeDeployDefault.OneAtATime** 的部署配置，创建一个与名为 **WordPress_App** 的应用程序相关联的名为 **WordPress_DepGroup** 的部署组：

```
aws deploy create-deployment-group \
  --application-name WordPress_App \
  --deployment-group-name WordPress_DepGroup \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
  --service-role-arn serviceRoleARN
```

> **Note**
>
> create-deployment-group 命令支持创建触发器，触发器会向主题订阅者发送有关部署和实例中指定事件的 Amazon SNS 通知。该命令还支持用于自动回滚部署和设置警报的选

项，以便在达到 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中不包含用于这些操作的命令。

3. 在创建部署之前，部署组中的实例必须安装 CodeDeploy 代理。您可以使用以下命令通过 Amazon Systems Manager 从命令行安装代理：

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \
  --parameters action=Install,name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

   此命令在 Systems Manager 状态管理器中创建一个关联，这将安装 CodeDeploy 代理，然后尝试在每周日凌晨 2:00 更新它。有关 CodeDeploy 代理的详细信息，请参阅使用 CodeDeploy 代理。有关 Systems Manager 的详细信息，请参阅什么是 Amazon Systems Manager。

4. 现在调用 create-deployment 命令，使用名为 **WordPress_App** 的存储桶中名为 **CodeDeployDefault.OneAtATime** 的应用程序修订，创建一个与名为 **WordPress_DepGroup** 的应用程序、名为 **WordPressApp.zip** 的部署配置、名为 **amzn-s3-demo-bucket** 的部署组关联的部署。

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip
```

部署应用程序修订（控制台）

1. 在使用 CodeDeploy 控制台部署您的应用程序修订之前，您需要服务角色 ARN。服务角色是 IAM 角色，该角色授予某个服务代表您执行操作的权限。在这种情况下，服务角色将为 CodeDeploy 提供访问您的 Amazon EC2 实例的权限，以扩展（读取）其 Amazon EC2 实例标签。

   您应该已经按照创建服务角色（控制台）中的说明创建了服务角色。要获取服务角色的 ARN，请参阅获取服务角色 ARN（控制台）。

2. 现在您已拥有 ARN，可以开始使用 CodeDeploy 控制台部署您的应用程序修订：

   登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

> **Note**
> 使用您在入门 CodeDeploy中设置的同一用户登录。

3. 在导航窗格中，展开部署，然后选择应用程序。
4. 在应用程序列表中，选择 WordPress_App。
5. 在部署组选项卡中，选择创建部署组。
6. 在 Deployment group name（部署组名称）中，输入 **`WordPress_DepGroup`**。
7. 在 Deployment type 下，选择 In-place deployment。
8. 在环境配置中，选择 Amazon EC2 实例。
9. 在使用 Amazon Systems Manager 的代理配置中，保留默认值。
10. 在键中，输入 **`Name`**。
11. 在值中，输入 **`CodeDeployDemo`**。

> **Note**
> 在键入 **`CodeDeployDemo`** 后，匹配实例下方应显示 1 来确认 CodeDeploy 已找到一个匹配的 Amazon EC2 实例。

12. 在部署配置中，选择 CodeDeployDefault.OneAtATime。
13. 在服务角色 ARN 中，选择服务角色 ARN，然后选择创建部署组。
14. 选择 Create deployment（创建部署）。
15. 在部署组中，选择 **`WordPress_DepGroup`**。
16. 在存储库类型旁，选择我的应用程序存储在 Amazon S3 中。在修订位置中，输入您之前上传到 Amazon S3 的示例 WordPress 应用程序修订的位置。获取位置：

    a. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
    b. 在存储桶列表中，选择 amzn-s3-demo-bucket（或您将应用程序修订上传到的存储桶的名称）。
    c. 在对象列表中，选择 WordPressApp.zip。
    d. 在概述选项卡中，将链接字段的值复制到您的剪贴板。

    其内容如下所示：

https://s3.amazonaws.com/amzn-s3-demo-bucket/WordPressApp.zip

e. 返回 CodeDeploy 控制台，在修订位置中，粘贴链接字段值。

17. 如果文件类型列表中显示的消息告诉您无法检测文件类型，请选择 .zip。
18. （可选）在 Deployment description 框中键入注释。
19. 展开部署组覆盖，从部署配置中，选择 CodeDeployDefault.OneAtATime。
20. 选择开始部署。有关您新创建的部署的信息将显示在 Deployments 页上。

监控您的部署并排除故障

使用 Amazon CLI 或控制台监控部署和排除部署故障。

主题

- 监视您的部署并排除故障（CLI）
- 监视您的部署和故障排除（控制台）

监视您的部署并排除故障（CLI）

1. 针对名为 list-deployments 的应用程序和名为 **WordPress_App** 的部署组调用 **WordPress_DepGroup** 命令，以获取部署的 ID：

```
aws deploy list-deployments --application-name WordPress_App --deployment-group-
name WordPress_DepGroup --query 'deployments' --output text
```

2. 通过部署 ID 调用 get-deployment 命令：

```
aws deploy get-deployment --deployment-id deploymentID --query
 'deploymentInfo.status' --output text
```

3. 该命令将返回部署的整体状态。如果成功，该值将为 `Succeeded`。

   如果总体状态为 `Failed`，则可以调用诸如 list-deployment-instances 和 get-deployment-instance 之类的命令进行故障排除。有关更多故障排除选项，请参阅分析日志文件以调查针对实例的部署失败。

监视您的部署和故障排除（控制台）

在 CodeDeploy 控制台的部署页面上，您可以在状态列中监控部署的状态。

要获取有关部署的详细信息（特别是在 Status 列值为 Succeeded 之外的任何值的情况下），请执行以下操作：

1. 在部署表中，选择部署名称。部署失败后，将显示描述失败原因的消息。
2. 在实例活动中，将显示有关部署的更多信息。部署失败之后，您可以确定在哪些 Amazon EC2 实例上失败以及部署在哪个步骤失败。
3. 如果您要执行更多问题排查，则可使用与View Instance Details中描述方法类似的方法。您还可以分析有关 Amazon EC2 实例的部署日志文件。有关更多信息，请参阅 分析日志文件以调查针对实例的部署失败。

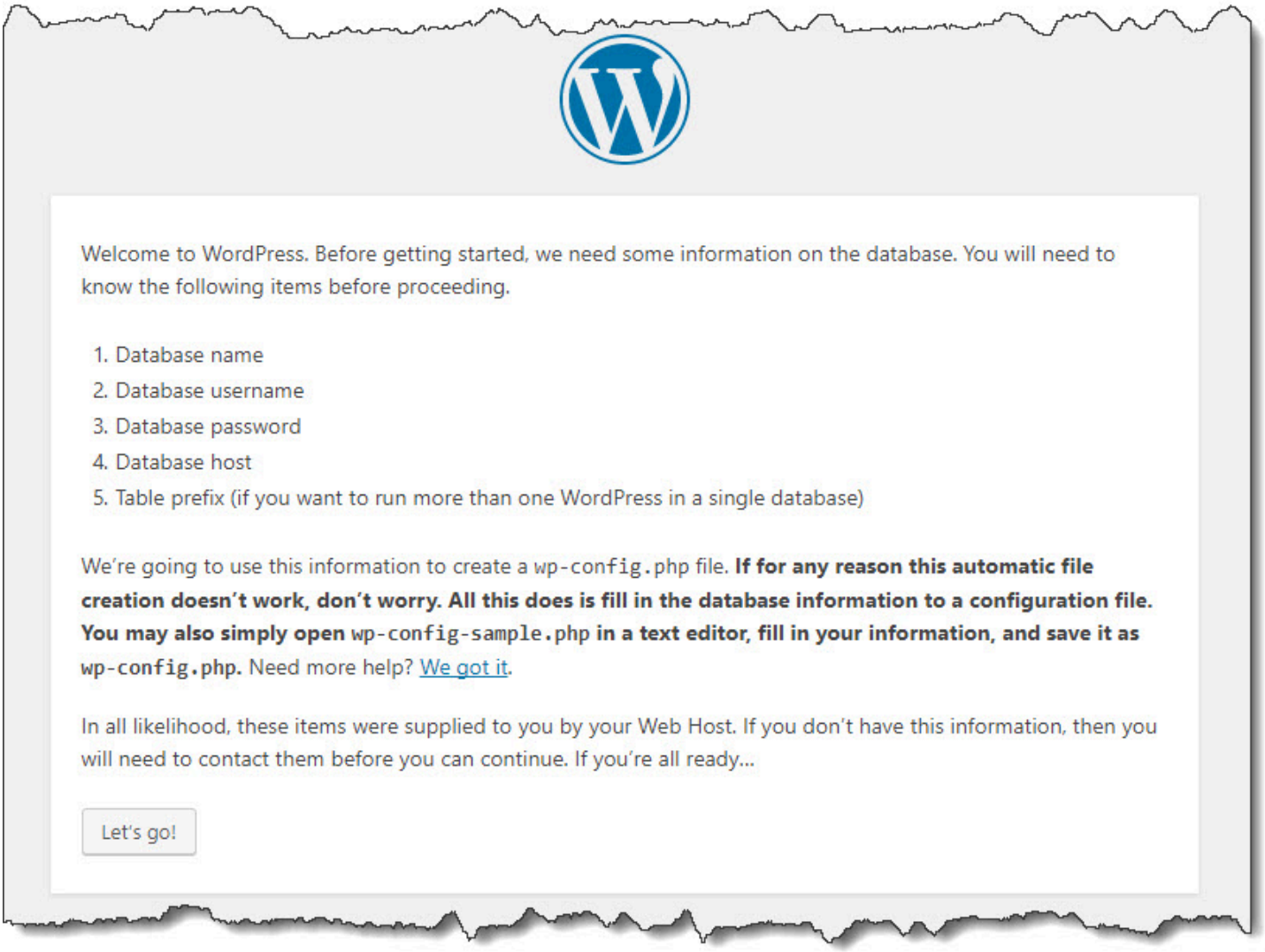
验证您的部署

部署成功后，请验证您的 WordPress 安装是否正常工作。使用 Amazon EC2 实例的公有 DNS 地址（后跟 `/WordPress`）以在 Web 浏览器中查看您的站点。（要获取公有 DNS 值，请在 Amazon EC2 控制台中，选择 Amazon EC2 实例，然后在描述选项卡上，查找公有 DNS 的值。）

例如，如果 Amazon EC2 实例的公有 DNS 地址为 **`ec2-01-234-567-890.compute-1.amazonaws.com`**，则您可以使用以下 URL：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress
```

当您在浏览器中查看站点时，您应看到类似于下文的 WordPress 欢迎页面：



如果您的 Amazon EC2 实例没有将 HTTP 入站规则添加到其安全组，则不显示 WordPress 欢迎页面。如果您看到说明远程服务器未响应的消息，请确保您 Amazon EC2 实例的安全组具有入站规则。有关更多信息，请参阅 添加入站规则，允许 HTTP 流量指向您的 Amazon Linux 或 RHEL Amazon EC2 实例。

步骤 5：更新和重新部署您的 WordPress 应用程序

现在您已成功部署应用程序修订，接下来可以在开发计算机上更新 WordPress 代码，然后使用 CodeDeploy 重新部署站点。之后，您应该可以看到 Amazon EC2 实例上的代码更改。

主题

- 设置 WordPress 站点
- 修改站点

- 重新部署站点

设置 WordPress 站点

要查看代码更改的影响，请完成 WordPress 站点的设置，这样您的安装就具备全部功能。

1. 将您站点的 URL 键入到 Web 浏览器中。该 URL 是 Amazon EC2 实例的公有 DNS 地址加上 `/WordPress` 扩展。对于此示例 WordPress 站点（以及示例 Amazon EC2 实例公有 DNS 地址），URL 为 **`http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress`**。
2. 如果您尚未设置站点，则将显示 WordPress 默认欢迎页面。选择开始！。
3. 要使用默认 MySQL 数据库，请在数据库配置页面上，键入以下值：

   - 数据库名称：**`test`**
   - 用户名：**`root`**
   - Password：留空。
   - 数据库主机：**`localhost`**
   - 表前缀：**`wp_`**

   选择 Submit 以设置数据库。
4. 继续站点设置。在 Welcome 页面上，填写所需的任意值，然后选择 Install WordPress。安装完成后，您可以登录到控制面板。

> **Important**
>
> 在 WordPress 应用程序的部署期间，**`change_permissions.sh`** 脚本将更新 `/tmp/WordPress` 文件夹的权限以便所有人可以写入。现在可以运行以下命令来限制权限，从而只有作为所有者的您才可以向其中写入：
>
> ```
> chmod -R 755 /var/www/html/WordPress
> ```

修改站点

要修改 WordPress 站点，请转到您的开发计算机上的应用程序文件夹：

```
cd /tmp/WordPress
```

要修改站点的某些颜色，请在 `wp-content/themes/twentyfifteen/style.css` 文件中，使用文本编辑器或 sed 将 `#fff` 更改为 `#768331`。

在 Linux 或其他具有 GNU sed 的系统上，使用：

```
sed -i 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

在 macOS、Unix 或其他具有 BSD sed 的系统上，使用：

```
sed -i '' 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

重新部署站点

您已经修改了站点的代码，现在可使用 Amazon S3 和 CodeDeploy 重新部署站点。

将更改打包并上传到 Amazon S3，如将应用程序的文件打包到单个存档文件并推送此存档文件中所述。（在按照这些说明操作时，请记住您不需要创建应用程序。）为新修订提供与之前一样的密钥（**`WordPressApp.zip`**）。将其上传到之前创建的同一个 Amazon S3 存储桶（例如，**`amzn-s3-demo-bucket`**）。

使用 Amazon CLI、CodeDeploy 控制台或 CodeDeploy API 重新部署站点。

主题

- 重新部署站点（CLI）
- 重新部署站点（控制台）

重新部署站点（CLI）

调用 create-deployment 命令，以便根据新上传的修订创建部署。使用名为 **`WordPress_App`** 的应用程序、名为 **`CodeDeployDefault.OneAtATime`** 的部署配置、名为 **`WordPress_DepGroup`** 的部署组以及名为 **`WordPressApp.zip`** 的存储桶中名为 **`amzn-s3-demo-bucket`** 的修订：

```
aws deploy create-deployment \
 --application-name WordPress_App \
 --deployment-config-name CodeDeployDefault.OneAtATime \
```

```
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip
```

您可以检查部署的状态，如 监控您的部署并排除故障 中所述。

在 CodeDeploy 重新部署了站点之后，在您的 Web 浏览器中重新访问站点，以验证颜色已发生更改。（您可能需要刷新浏览器。）如果颜色已发生更改，那么恭喜您！您已成功修改并重新部署了站点！

重新部署站点（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序列表中，选择 WordPress_App。
4. 在部署组选项卡上，选择 **WordPress_DepGroup**。
5. 选择 Create deployment（创建部署）。
6. 在 Create deployment 页面上：
   a. 在部署组中，选择 **WordPress_DepGroup**。
   b. 在存储库类型区域中，选择我的应用程序存储在 Amazon S3 中，然后将修订的 Amazon S3 链接复制到修订位置框中。要查找链接值，请执行以下操作：
      i. 在单独的浏览器选项卡中：

         登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

         浏览到并打开 amzn-s3-demo-bucket，然后选择您的修订 **WordPressApp.zip**。
      ii. 如果属性窗格在 Amazon S3 控制台中不可见，则选择属性按钮。
      iii. 在属性窗格中，将链接字段的值复制到 CodeDeploy 控制台的修订位置框中。
   c. 如果显示消息说明无法检测文件类型，请选择 .zip。
   d. 将 Deployment description 框留空。
   e. 展开部署组覆盖，从部署配置中，选择 CodeDeployDefault.OneAtATime。

f. 选择开始部署。有关您新创建的部署的信息将显示在 Deployments 页上。

g. 您可以检查部署的状态，如 监控您的部署并排除故障 中所述。

在 CodeDeploy 重新部署了站点之后，在您的 Web 浏览器中重新访问站点，以验证颜色已发生更改。（您可能需要刷新浏览器。）如果颜色已发生更改，那么恭喜您！您已成功修改并重新部署了站点！

步骤 6：清理 WordPress 应用程序和相关资源

您现在已成功更新 WordPress 代码并已重新部署站点。要避免为此教程创建的资源持续产生费用，您应删除：

- 任何 Amazon CloudFormation 堆栈（或终止任何 Amazon EC2 实例，条件是这些实例是在 Amazon CloudFormation 外部创建的）。
- 任何 Amazon S3 存储桶。
- CodeDeploy 中的 `WordPress_App` 应用程序。
- CodeDeploy 代理的 Amazon Systems Manager 状态管理器关联。

您可以使用 Amazon CLI、Amazon CloudFormation、Amazon S3、Amazon EC2 和 CodeDeploy 控制台或 Amazon API 来执行清理。

主题

- 清除资源（CLI）
- 清除资源（控制台）
- 接下来做什么？

清除资源（CLI）

1. 如果您为此教程使用了 Amazon CloudFormation 模板，请对名为 delete-stack 的堆栈调用 **`CodeDeployDemoStack`** 命令。这将终止所有附带的 Amazon EC2 实例，并删除堆栈创建的所有附带的 IAM 角色：

```
aws cloudformation delete-stack --stack-name CodeDeployDemoStack
```

2. 要删除 Amazon S3 存储桶，请使用 --recursive 开关针对名为 rm 的存储桶调用 **`amzn-s3-demo-bucket`** 命令。这将删除存储桶以及该存储桶中的所有对象：

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
```

3. 要删除 `WordPress_App` 应用程序，请调用 delete-application 命令。这也将删除应用程序的所有关联的部署组记录和部署记录：

```
aws deploy delete-application --application-name WordPress_App
```

4. 要删除 Systems Manager 状态管理器关联，请调用 delete-association 命令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通过调用 describe-association 命令来获取 *`association-id`*。

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

如果您在此教程中未使用 Amazon CloudFormation 堆栈，则调用 terminate-instances 命令来终止您手动创建的任何 Amazon EC2 实例。提供要终止的 Amazon EC2 实例的 ID：

```
aws ec2 terminate-instances --instance-ids instanceId
```

清除资源（控制台）

如果您已为此教程使用 Amazon CloudFormation 模板，请删除关联的 Amazon CloudFormation 堆栈。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在筛选条件框中，键入之前创建的 Amazon CloudFormation 堆栈名称（例如，**`CodeDeployDemoStack`**）。
3. 选中堆栈名称旁边的框。在 Actions 菜单中，选择 Delete Stack。

   Amazon CloudFormation 将删除堆栈，终止所有附带的 Amazon EC2 实例并删除所有附带的 IAM 角色。

要终止在 Amazon CloudFormation 堆栈外部创建的 Amazon EC2 实例，请执行以下步骤：

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在 INSTANCES 列表中，选择 Instances。
3. 在搜索框中，键入要终止的 Amazon EC2 实例的名称（例如，**CodeDeployDemo**），然后按 Enter。
4. 选择 Amazon EC2 实例名称。
5. 在 Actions 菜单中，指向 Instance State，然后选择 Terminate。在系统提示时，选择 Yes, Terminate。

对每个实例重复这些步骤。

要删除 Amazon S3 存储桶，请执行以下步骤：

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，浏览到并选择之前创建的 Amazon S3 存储桶的名称（例如，**amzn-s3-demo-bucket**）。
3. 您必须先删除存储桶的内容，然后才能删除存储桶。选择存储桶中的所有文件（如 **WordPressApp.zip**）。在 Actions 菜单中，选择 Delete。在提示确认删除时，选择 OK。
4. 在清空存储桶后，可以删除存储桶。在存储桶列表中，选择存储桶的行（而不是存储桶名称）。选择 Delete bucket，当系统提示进行确认时，选择 OK。

要从 CodeDeploy 中删除 `WordPress_App` 应用程序，请执行以下步骤：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序列表中，选择 WordPress_App。
4. 在 Application details 页上，选择 Delete application。
5. 在系统提示时，输入应用程序的名称以确认要删除应用程序，然后选择删除。

要删除 Systems Manager 状态管理器关联，请执行以下操作：

1. 访问 https://console.aws.amazon.com/systems-manager，打开 Amazon Systems Manager 控制台。
2. 在导航窗格中，选择状态管理器。
3. 选择您创建的关联，然后选择删除。

接下来做什么？

如果您已到达此处，那么恭喜您！您已成功完成 CodeDeploy 部署，然后已更新您站点的代码并且已重新部署站点。

教程：使用 CodeDeploy 部署“Hello, World!” 带有 CodeDeploy（Windows 服务器）的应用程序

在本教程中，您将单个网页部署到运行互联网信息服务 (IIS) 作为其网络服务器的单个 Windows Server Amazon EC2 实例。此网页将显示简单的“Hello，World!” 消息。

不是您要找的内容？

- 要练习改为部署到亚马逊 Linux 或红帽企业 Linux (RHEL) 亚马逊 EC2 实例，请参阅教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix）。
- 要改而部署到本地实例，请参阅教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）。

本教程中的步骤是从 Windows 角度提供的。虽然您可以在运行 Linux、macOS 或 Unix 的本地计算机上完成其中的大部分步骤，但您必须适应涵盖基于 Windows 的目录路径（例如 `c:\temp`）的步骤。此外，如果您想连接到亚马逊 EC2 实例，则需要一个能够通过远程桌面协议 (RDP) 连接到运行 Windows Server 的亚马逊 EC2 实例的客户端应用程序。（默认情况下，Windows 包含 RDP 连接客户端应用程序。）

在开始本教程之前，您必须完成中的先决条件入门 CodeDeploy，包括配置您的用户、安装或升级 Amazon CLI，以及创建 IAM 实例配置文件和服务角色。

主题

- 步骤 1：启动 Windows Server Amazon EC2 实例
- 步骤 2：将您的源内容配置为部署到 Windows Server Amazon EC2 实例
- 步骤 3：将“Hello, World!” 应用程序上传到 Amazon S3
- 步骤 4：部署 Hello World 应用程序
- 步骤 5：更新和重新部署“Hello, World!” 应用程序
- 步骤 6：清理“Hello，World!” 应用程序和相关资源

步骤 1：启动 Windows Server Amazon EC2 实例

要使用 CodeDeploy 部署 Hello World 应用程序，您需要使用运行 Windows Server 的 Amazon EC2 实例。

按照为 CodeDeploy 创建 Amazon EC2 实例中的说明进行操作。当您准备好向实例分配 Amazon EC2 实例标签时，请确保指定标签键 **Name** 和标签值 **CodeDeployDemo**。（如果您指定不同的标记键或标签值，则步骤 4：部署 Hello World 应用程序中的说明可能会产生意外结果。）

在您启动 Amazon EC2 实例之后，请返回到此页，并继续下一部分。请勿继续使用创建应用程序 CodeDeploy作为下一步骤。

连接到您的 Amazon EC2 实例

启动 Amazon EC2 实例之后，请按照说明来练习连接操作。

> **Note**
>
> 在这些说明中，我们假定您运行 Windows 和 Windows Desktop Connection 客户端应用程序。有关信息，请参阅使用 RDP 连接到您的 Windows 实例。对于其他操作系统或其他 RDP 连接客户端应用程序，您可能需要相应修改这些说明。

1. 登录 Amazon Web Services 管理控制台，打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的 Instances 下，选择 Instances。
3. 浏览并在列表中选择您的 Windows Server 实例。
4. 选择连接。

5. 选择获取密码，然后选择选择文件。
6. 浏览并选择与 Windows Server Amazon EC2 实例关联的 Amazon EC2 实例密钥对，然后选择打开。
7. 选择 Decrypt Password。记录显示的密码。您在步骤 10 中需要它。
8. 选择 Download Remote Desktop File，然后打开文件。
9. 如果系统提示您连接（即使无法确定远程连接的发布程序），请继续。
10. 键入您在步骤 7 中记录的密码，然后继续。（如果 RDP 连接客户端应用程序提示您输入用户名，请键入 **Administrator**。）
11. 如果系统提示您连接（即使无法验证远程计算机的身份），请继续。
12. 在连接之后，将显示运行 Windows Server 的 Amazon EC2 实例的桌面。
13. 现在您可以断开 Amazon EC2 实例的连接。

> **Warning**
>
> 请不要停止或终止实例。否则，CodeDeploy 将不能部署到该实例。

添加允许 HTTP 流量传输到您的 Windows Server Amazon EC2 实例的入站规则

下一步是确认您的 Amazon EC2 实例具有开放的 HTTP 端口，以便您可以在浏览器中查看在您的 Windows Server Amazon EC2 实例上部署的网页。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 选择实例，然后选择您的实例。
3. 在描述选项卡上的安全组下，选择查看入站规则。

   您应在安全组中看到类似如下的规则列表：

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports      Protocol      Source      launch-wizard-N
    22         tcp           0.0.0.0/0           #
   ```

4. 在安全组下，为您的 Amazon EC2 实例选择安全组。其可能被命名为 **launch-wizard-*N***。名称中的 ***N*** 是创建实例时分配到您安全组的编号。

   选择入站选项卡。如果实例的安全组配置正确，则应看到一条具有以下值的规则：

- 类型：HTTP
- 协议：TCP
- 端口范围：80
- 来源：0.0.0.0/0

5. 如果您没有看到包含这些值的规则，请使用向安全组添加规则中的过程将其添加到新的安全规则中。

步骤 2：将您的源内容配置为部署到 Windows Server Amazon EC2 实例

现在可以配置您的应用程序的源内容了，这样您就有可以部署到 Amazon EC2 实例的内容了。在此教程中，您会将一个网页部署到运行 Windows Server 的 Amazon EC2 实例，后者将 Internet Information Services（IIS）作为其 Web 服务器运行。此网页将简单地显示“Hello，World!” 消息。

主题

- 创建网页
- 创建运行应用程序的脚本
- 添加应用程序规范文件

创建网页

1. 在您的 `HelloWorldApp` 文件夹中创建一个名为 `c:\temp` 的子目录（子文件夹），然后切换到该文件夹。

```
mkdir c:\temp\HelloWorldApp
cd c:\temp\HelloWorldApp
```

> **Note**
>
> 您不必使用 `c:\temp` 作为位置或 `HelloWorldApp` 作为子文件夹名称。如果您使用不同的位置或子文件夹名称，请确保在本教程中通篇使用它。

2. 使用文本编辑器在文件夹内创建一个文件。将文件命名为 `index.html`。

```
notepad index.html
```

3. 将以下 HTML 代码添加到该文件中，然后保存文件。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
  <title>Hello, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #0188cc;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello, World!</h1></div>
  <div align="center"><h2>You have successfully deployed an application using
 CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

创建运行应用程序的脚本

接下来，您将创建一个脚本，CodeDeploy 将使用该脚本来设置目标 Amazon EC2 实例上的 Web 服务器。

1. 在保存 `index.html` 文件的相同子文件夹中，使用文本编辑器创建另一个文件。将文件命名为 `before-install.bat`。

```
notepad before-install.bat
```

2. 将以下批处理脚本代码添加到该文件中，然后保存文件。

```
REM Install Internet Information Server (IIS).
```

```
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Import-Module -
Name ServerManager
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Install-
WindowsFeature Web-Server
```

添加应用程序规范文件

接下来，除了网页和批处理脚本文件以外，您还将添加一个应用程序规范文件（AppSpec 文件）。AppSpec 文件是一个 YAML 格式的文件，由 CodeDeploy 用于：

- 将应用程序修订中的源文件映射到其在实例上的目的地。
- 指定在部署期间要在实例上运行的脚本。

AppSpec 文件必须命名为 `appspec.yml`。它必须放置在应用程序源代码的根文件夹中。

1. 在保存 `index.html` 和 `before-install.bat` 文件的相同子文件夹中，使用文本编辑器创建另一个文件。将文件命名为 `appspec.yml`。

   ```
   notepad appspec.yml
   ```

2. 将以下 YAML 代码添加到该文件中，然后保存该文件。

   ```
   version: 0.0
   os: windows
   files:
     - source: \index.html
       destination: c:\inetpub\wwwroot
   hooks:
     BeforeInstall:
       - location: \before-install.bat
         timeout: 900
   ```

CodeDeploy 将使用此 AppSpec 文件将应用程序源代码根文件夹中的 `index.html` 文件复制到目标 Amazon EC2 实例上的 `c:\inetpub\wwwroot` 文件夹。在部署过程中，在 **`BeforeInstall`** 部署生命周期事件期间，CodeDeploy 将在目标 Amazon EC2 实例上运行 `before-install.bat` 批处理脚本。如果此脚本运行时间超过 900 秒（15 分钟），则 CodeDeploy 将停止部署并将对 Amazon EC2 实例的部署标记为失败。

有关这些设置的更多信息，请参阅 CodeDeploy AppSpec 文件参考。

> **Important**
>
> 此文件中各项之间的空格的位置和数量很重要。如果间距不正确，CodeDeploy 将引发可能难以调试的错误。有关更多信息，请参阅 AppSpec 文件间距。

步骤 3：将“Hello, World!” 应用程序上传到 Amazon S3

现在，您将准备源内容并将其上传到 CodeDeploy 可以部署的地方。以下说明向您演示如何预置 Amazon S3 存储桶、为存储桶准备应用程序修订的文件、对修订的文件打包，然后将修订推送到存储桶。

> **Note**
>
> 尽管本教程中没有介绍，但您可以使用 CodeDeploy 将应用程序从 GitHub 存储库部署到实例。有关更多信息，请参阅 将 CodeDeploy 与 GitHub 集成。

主题

- 预置 Amazon S3 存储桶
- 为存储桶准备应用程序的文件
- 将应用程序的文件打包到单个存档文件并推送此存档文件

预置 Amazon S3 存储桶

在 Amazon S3 中创建存储容器或存储桶，或者使用现有的存储桶。确保您可以将修订版上传到存储桶，并且部署中使用的 Amazon EC2 实例可以从存储桶下载修订版。

您可以使用 Amazon CLI、亚马逊 S3 控制台或 Amazon S3 APIs 来创建 Amazon S3 存储桶。创建存储桶后，请确保提供对存储桶和您的 CodeDeploy 用户的访问权限。

> **Note**
>
> 在 Amazon S3 中，所有 Amazon 账户的存储桶名称必须是唯一的。如果您无法使用 **amzn-s3-demo-bucket**，请尝试其他存储桶名称，例如 **amzn-s3-demo-bucket** 后跟短划线和

您的姓名首字母或其他某个唯一标识符。之后，请确保将此教程中的任何 **amzn-s3-demo-bucket** 替换为您的存储桶名称。
Amazon S3 存储桶必须在启动您的目标 Amazon EC2 实例的同一 Amazon 区域创建。例如，如果您在美国东部（弗吉尼亚北部）地区创建存储桶，则您的目标 Amazon EC2 实例必须在美国东部（弗吉尼亚北部）地区启动。

主题

- 创建 Amazon S3 存储桶（CLI）
- 创建 Amazon S3 存储桶（控制台）
- 向 Amazon S3 存储桶和您的 Amazon 账户授予权限

创建 Amazon S3 存储桶（CLI）

调用 mb 命令以创建一个名为 **amzn-s3-demo-bucket** 的 Amazon S3 存储桶：

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

创建 Amazon S3 存储桶（控制台）

1. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 在 Amazon S3 控制台中，选择创建存储桶。
3. 在 Bucket name 框中，键入存储桶的名称。
4. 在 Region 列表中，选择目标区域，然后选择 Create。

向 Amazon S3 存储桶和您的 Amazon 账户授予权限

您必须拥有上传到 Amazon S3 存储桶的权限。您可以通过 Amazon S3 存储桶策略指定这些权限。例如，在以下 Amazon S3 存储桶策略中，使用通配符 (*) 允许 Amazon 账户将文件上传`111122223333`到 Amazon S3 存储桶中名为`amzn-s3-demo-bucket`的任何目录：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
```

```
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

要查看您的 Amazon 账户 ID，请参阅查找您的 Amazon 账户 ID。

现在是验证 Amazon S3 存储桶是否允许来自每个参与的亚马逊 EC2 实例的下载请求的好时机。您可以通过 Amazon S3 存储桶策略来指定这一点。例如，在以下 Amazon S3 存储桶策略中，使用通配符 (*) 允许任何附带包含 `arn:aws:iam::444455556666:role/CodeDeployDemo` ARN 的 IAM 实例配置文件的 Amazon EC2 实例从名为 Amazon S3 存储桶中的任何目录下载文件：`amzn-s3-demo-bucket`

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

有关如何生成和附加 Amazon S3 存储桶策略的信息，请参阅存储桶策略示例。

您在中创建的 CodeDeploy 管理员用户还步骤 1：设置必须有权将修订版上传到 Amazon S3 存储桶。指定这一点的一种方法是通过 IAM policy（添加到用户的权限集）或 IAM 角色（允许用户代入）。以

下 IAM policy 允许用户将修订上传到名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任意位置：

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

有关如何创建 IAM policy 的信息，请参阅《IAM 用户指南》中的创建 IAM policy。有关向权限集添加策略的信息，请参阅《Amazon IAM Identity Center 用户指南》中的创建权限集。

为存储桶准备应用程序的文件

确保网页、 AppSpec 文件和脚本在开发计算机上按如下方式组织：

```
c:\
  |-- temp\
        |--HelloWorldApp\
             |-- appspec.yml
             |-- before-install.bat
             |-- index.html
```

将应用程序的文件打包到单个存档文件并推送此存档文件

将这些文件打包到一个存档文件（称为应用程序修订）。

> **Note**
>
> 将对象存储在存储桶中以及将应用程序修订传入和传出存储桶可能需支付费用。有关更多信息，请参阅 Amazon S3 定价。

1. 在开发计算机上，切换到这些文件存储到的文件夹：

```
cd c:\temp\HelloWorldApp
```

> **Note**
>
> 如果您未切换到此文件夹，则将在您的当前文件夹中启动文件打包。例如，如果您当前的文件夹是 `c:\temp` 而非 `c:\temp\HelloWorldApp`，则打包操作将从 `c:\temp` 文件夹中的文件和子文件夹开始，这可能包括 `HelloWorldApp` 子文件夹之外的内容。

2. 调用create-application命令注册一个名为**`HelloWorld_App`**以下内容的新应用程序 CodeDeploy：

```
aws deploy create-application --application-name HelloWorld_App
```

3. 调用 p CodeDeploy ush 命令将文件捆绑在一起，将修订上传到 Amazon S3，并在其中注册 CodeDeploy 有关已上传修订的信息，所有这些操作只需一个操作即可。

```
aws deploy push --application-name HelloWorld_App --s3-location s3://amzn-s3-demo-
bucket/HelloWorld_App.zip --ignore-hidden-files
```

   此命令将当前目录中的文件（不包括任何隐藏文件）捆绑到名为的单个存档文件中`HelloWorld_App.zip`，将修订版上传到**`amzn-s3-demo-bucket`**存储桶，并在其中注册 CodeDeploy 有关已上传修订的信息。

步骤 4：部署 Hello World 应用程序

现在，您将部署已上传到 Amazon S3 的示例 Hello World 应用程序修订。您将使用 Amazon CLI 或 CodeDeploy 控制台部署修订并监控部署进度。成功部署应用程序修订之后，可以检查结果。

主题

- 使用 CodeDeploy 部署您的应用程序修订
- 监控您的部署并排除故障
- 验证您的部署

使用 CodeDeploy 部署您的应用程序修订

您可以使用 CLI 或控制台部署应用程序。

主题

- 部署您的应用程序修订（CLI）
- 部署应用程序修订（控制台）

部署您的应用程序修订（CLI）

1. 首先，部署需要部署组。不过，在创建部署组之前，您需要服务角色 ARN。服务角色是 IAM 角色，该角色授予某个服务代表您执行操作的权限。在这种情况下，服务角色将为 CodeDeploy 提供访问您的 Amazon EC2 实例的权限，以扩展（读取）其 Amazon EC2 实例标签。

   您应该已经按照创建服务角色（CLI）中的说明创建了服务角色。要获取服务角色的 ARN，请参阅获取服务角色 ARN（CLI）。

2. 现在您已拥有此 ARN，请调用 create-deployment-group 命令，使用名为 **CodeDeployDemo** 的 Amazon EC2 实例标签和名为 **CodeDeployDefault.OneAtATime** 的部署配置创建一个与名为 **HelloWorld_App** 的应用程序相关联的名为 **HelloWorld_DepGroup** 的部署组（具有指定的服务角色 ARN）：

```
aws deploy create-deployment-group --application-name HelloWorld_App
 --deployment-group-name HelloWorld_DepGroup --deployment-
config-name CodeDeployDefault.OneAtATime --ec2-tag-filters
 Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE --service-role-arn serviceRoleARN
```

   > **Note**
   >
   > create-deployment-group 命令支持创建触发器，触发器会向主题订阅者发送有关部署和实例中指定事件的 Amazon SNS 通知。该命令还支持用于自动回滚部署和设置警报的选项，以便在达到 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中不包含用于这些操作的命令。

3. 在创建部署之前，部署组中的实例必须安装 CodeDeploy 代理。您可以使用以下命令通过 Amazon Systems Manager 从命令行安装代理：

```
aws ssm create-association --name AWS-ConfigureAWSPackage
 --targets Key=tag:Name,Values=CodeDeployDemo --parameters
```

```
action=Install,name=AWSCodeDeployAgent --schedule-expression "cron(0 2 ? * SUN
*)"
```

此命令在 Systems Manager 状态管理器中创建一个关联，这将安装 CodeDeploy 代理，然后尝试在每周日凌晨 2:00 更新它。有关 CodeDeploy 代理的详细信息，请参阅使用 CodeDeploy 代理。有关 Systems Manager 的详细信息，请参阅什么是 Amazon Systems Manager。

4. 现在调用 create-deployment 命令，使用名为 **HelloWorld_App** 的存储桶中名为 **CodeDeployDefault.OneAtATime** 的应用程序修订，创建一个与名为 **HelloWorld_DepGroup** 的应用程序、名为 **HelloWorld_App.zip** 的部署配置、名为 **amzn-s3-demo-bucket** 的部署组关联的部署。

```
aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

部署应用程序修订（控制台）

1. 在使用 CodeDeploy 控制台部署您的应用程序修订之前，您需要服务角色 ARN。服务角色是 IAM 角色，该角色授予某个服务代表您执行操作的权限。在这种情况下，服务角色将为 CodeDeploy 提供访问您的 Amazon EC2 实例的权限，以扩展（读取）其 Amazon EC2 实例标签。

   您应该已经按照创建服务角色（控制台）中的说明创建了服务角色。要获取服务角色的 ARN，请参阅获取服务角色 ARN（控制台）。

2. 现在您已拥有 ARN，可以开始使用 CodeDeploy 控制台部署您的应用程序修订。

   登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

3. 在导航窗格中，展开部署，然后选择应用程序。
4. 选择 HelloWorld_App。
5. 在部署组选项卡中，选择创建部署组。
6. 在 Deployment group name（部署组名称）中，输入 **HelloWorld_DepGroup**。
7. 在服务角色中，选择服务角色的名称。

8. 在部署类型中，选择就地。
9. 在环境配置中，选择 Amazon EC2 实例。
10. 在使用 Amazon Systems Manager 的代理配置中，保留默认值。
11. 在键中，输入 **Name**。
12. 在值中，输入 **CodeDeployDemo**。
13. 在部署配置中，选择 CodeDeployDefault.OneAtATime。
14. 在负载均衡器中，清除启用负载均衡。
15. 选择 Create deployment group（创建部署组）。
16. 选择 Create deployment（创建部署）。
17. 在部署组中，选择 HelloWorld_DepGroup
18. 在修订类型中，选择我的应用程序将存储在 Amazon S3 中，然后在修订位置中输入已上传到 Amazon S3 的示例 Hello World 应用程序修订的位置。获取位置：
    a. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
    b. 在存储桶列表中，选择 amzn-s3-demo-bucket（或您将应用程序修订上传到的存储桶的名称）。
    c. 在对象列表中，选择 HelloWorld_App.zip。
    d. 在 Overview（概述）选项卡上，选择 Copy path（复制路径）。
    e. 返回 CodeDeploy 控制台，在修订位置中，粘贴链接字段值。
19. 对于 Revision file type（修订文件类型），选择 .zip。
20. （可选）在 Deployment description（部署描述）中输入注释。
21. 选择 Create deployment（创建部署）。有关您新创建的部署的信息将显示在 Deployments 页上。

监控您的部署并排除故障

使用 Amazon CLI 或控制台监控部署和排除部署故障。

主题

- 监视您的部署并排除故障（CLI）
- 监视您的部署和故障排除（控制台）

监视您的部署并排除故障（CLI）

1. 针对名为 list-deployments 的应用程序和名为 **HelloWorld_App** 的部署组调用 **HelloWorld_DepGroup** 命令，以获取部署的 ID：

```
aws deploy list-deployments --application-name HelloWorld_App --deployment-group-
name HelloWorld_DepGroup --query "deployments" --output text
```

2. 通过部署 ID 调用 get-deployment 命令：

```
aws deploy get-deployment --deployment-id deploymentID --query
 "deploymentInfo.status" --output text
```

3. 该命令将返回部署的整体状态。如果成功，该值将为 `Succeeded`。

   如果总体状态为 `Failed`，则可以调用诸如 list-deployment-instances 和 get-deployment-instance 之类的命令进行故障排除。有关更多故障排除选项，请参阅分析日志文件以调查针对实例的部署失败。

监视您的部署和故障排除（控制台）

在 CodeDeploy 控制台的部署页面上，您可以在状态列中监控部署的状态。

要获取有关部署的详细信息（特别是在 Status 列值为 Succeeded 之外的任何值的情况下），请执行以下操作：

1. 在部署表中，选择您的部署 ID。部署失败后，部署的详细信息页中将显示描述失败原因的消息。
2. 将显示有关部署实例的更多信息。部署失败之后，您可以确定在哪些 Amazon EC2 实例上失败以及部署在哪个步骤失败。
3. 如果您需要执行进一步的故障排除，可以使用类似于 View Instance Details 的技巧。您还可以分析有关 Amazon EC2 实例的部署日志文件。有关更多信息，请参阅 分析日志文件以调查针对实例的部署失败。

验证您的部署

部署成功后，请验证您的安装是否正常工作。使用 Amazon EC2 实例的公有 DNS 地址，在 Web 浏览器中查看网页。（要获取公有 DNS 值，请在 Amazon EC2 控制台中，选择 Amazon EC2 实例，然后在描述选项卡上，查找公有 DNS 的值。）

例如，如果 Amazon EC2 实例的公有 DNS 地址为 **`ec2-01-234-567-890.compute-1.amazonaws.com`**，则您可以使用以下 URL：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果成功，您应该看到 Hello, World! 网页。

步骤 5：更新和重新部署“Hello, World!” 应用程序

现在您已成功部署应用程序修订，接下来就可以在开发计算机上更新网页代码，然后使用 CodeDeploy 重新部署站点。重新部署后，您应该可以在 Amazon EC2 实例上看到更改。

主题

- 修改网页
- 重新部署站点

修改网页

1. 转到 `c:\temp\HelloWorldApp` 子文件夹并使用文本编辑器修改 `index.html` 文件：

   ```
   cd c:\temp\HelloWorldApp
   notepad index.html
   ```

2. 修订 `index.html` 文件的内容，以更改网页的背景颜色和一些文本，然后保存该文件：

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
   TR/html4/loose.dtd">
   <html>
   <head>
     <title>Hello Again, World!</title>
     <style>
       body {
         color: #ffffff;
         background-color: #66cc00;
         font-family: Arial, sans-serif;
         font-size:14px;
       }
     </style>
   </head>
   <body>
   ```

```
  <div align="center"><h1>Hello Again, World!</h1></div>
  <div align="center"><h2>You have successfully deployed a revision of an
 application using CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

重新部署站点

现在您已修改代码，接下来就可以使用 Amazon S3 和 CodeDeploy 重新部署网页。

将更改打包并上传到 Amazon S3，如将应用程序的文件打包到单个存档文件并推送此存档文件中所述。（在按照这些说明操作时，您不需要创建新的应用程序。）为修订提供与之前一样的密钥（**HelloWorld_App.zip**）。将其上传到之前创建的同一个 Amazon S3 存储桶（例如，**amzn-s3-demo-bucket**）。

使用 Amazon CLI 或 CodeDeploy 控制台重新部署站点。

主题

- 重新部署站点（CLI）
- 重新部署站点（控制台）

重新部署站点（CLI）

现在调用 create-deployment 命令，再次使用名为 **HelloWorld_App** 的应用程序、名为 **CodeDeployDefault.OneAtATime** 的部署配置、名为 **HelloWorld_DepGroup** 的部署组以及名为 **HelloWorld_App.zip** 的存储桶中名为 **amzn-s3-demo-bucket** 的修订，基于上传的修订创建部署。

```
 aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

您可以检查新部署的状态，如监控您的部署并排除故障中所述。

在 CodeDeploy 重新部署站点后，请在 Web 浏览器中重新访问站点，验证网页上的背景颜色和文本是否已更改。（您可能需要刷新浏览器。）如果背景颜色和文本已更改，那么恭喜！您已经修改并重新部署了站点！

重新部署站点（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格上，选择应用程序。
3. 在应用程序列表中，选择 HelloWorld_App。
4. 在部署选项卡中，选择创建部署。

   a. 在 Deployment group 列表中，选择 HelloWorld_DepGroup。
   b. 在修订位置中，输入您的修订的 Amazon S3 链接。

      要查找链接值，请执行以下操作：

      i. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

         浏览到并打开 amzn-s3-demo-bucket，然后在 Amazon S3 控制台中选择您的修订 **`HelloWorld_App.zip`**。

      ii. 如果属性窗格在 Amazon S3 控制台中不可见，则选择属性按钮。
      iii. 在属性窗格中，复制链接字段的值。
      iv. 返回 CodeDeploy 控制台，在修订位置中粘贴此链接。
      v.
   c. 在修订文件类型中，如果出现指示无法检测文件类型的消息，则选择 .zip。
   d. 保留部署描述为空。
   e. 展开部署组覆盖。在部署配置列表中，选择 CodeDeployDefault.OneAtATime，然后选择创建部署。

      您可以检查部署的状态，如监控您的部署并排除故障中所述。

在 CodeDeploy 重新部署站点后，请在 Web 浏览器中重新访问站点，验证网页上的背景颜色和文本是否已更改。（您可能需要刷新浏览器。）如果背景颜色和文本已更改，那么恭喜！您已经修改并重新部署了站点！

步骤 6：清理“Hello，World!” 应用程序和相关资源

您现在已成功更新“Hello, World!” 代码并已重新部署站点。要避免为完成此教程而创建的资源持续产生费用，您应删除：

- 任何 Amazon CloudFormation 堆栈（或终止任何 Amazon EC2 实例，条件是这些实例是在 Amazon CloudFormation 外部创建的）。
- 任何 Amazon S3 存储桶。
- CodeDeploy 中的 `HelloWorld_App` 应用程序。
- CodeDeploy 代理的 Amazon Systems Manager 状态管理器关联。

您可以使用 Amazon CLI、Amazon CloudFormation、Amazon S3、Amazon EC2 和 CodeDeploy 控制台或 Amazon API 来执行清理。

主题

- 使用清除资源（CLI）
- 清除资源（控制台）
- 接下来做什么？

使用清除资源（CLI）

1. 如果您为此教程使用了 Amazon CloudFormation 堆栈，请通过对名为 delete-stack 的堆栈调用 **`CodeDeployDemoStack`** 命令来删除堆栈。这将终止所有附带的 Amazon EC2 实例并删除所有最初由该堆栈创建的附带 IAM 角色。

```
aws cloudformation delete-stack --stack-name CodeDeployDemoStack
```

2. 要删除 Amazon S3 存储桶，请使用 --recursive 开关针对名为 rm 的存储桶调用 **`amzn-s3-demo-bucket`** 命令。这将删除存储桶以及该存储桶中的所有对象。

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
```

3. 要从 CodeDeploy 中删除 `HelloWorld_App` 应用程序，请调用 delete-application 命令。这将删除应用程序的所有关联部署组记录和部署记录。

```
aws deploy delete-application --application-name HelloWorld_App
```

4. 要删除 Systems Manager 状态管理器关联，请调用 delete-association 命令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通过调用 describe-association 命令来获取 *`association-id`*。

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

5. 如果您在此教程中未使用 Amazon CloudFormation 堆栈，则调用 terminate-instances 命令来终止您手动创建的 Amazon EC2 实例。提供要终止的 Amazon EC2 实例的 ID。

```
aws ec2 terminate-instances --instance-ids instanceId
```

清除资源（控制台）

如果您已为此教程使用 Amazon CloudFormation 模板，请删除关联的 Amazon CloudFormation 堆栈。

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在搜索框中，键入 Amazon CloudFormation 堆栈名称（例如，**`CodeDeployDemoStack`**）。
3. 选中堆栈名称旁边的框。
4. 在 Actions 菜单中，选择 Delete Stack。这将删除堆栈，终止所有附带的 Amazon EC2 实例，并删除所有附带的 IAM 角色。

要终止在 Amazon CloudFormation 堆栈外部创建的 Amazon EC2 实例，请执行以下步骤：

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在 Instances 区域中，选择 Instances。
3. 在搜索框中，键入要终止的 Amazon EC2 实例的名称，然后按 Enter。

4. 选择您的 Amazon EC2 实例。
5. 选择 Actions，指向 Instance State，然后选择 Terminate。在系统提示时，选择 Yes, Terminate。对任何其他 Amazon EC2 实例重复这些步骤。

要删除 Amazon S3 存储桶，请执行以下步骤：

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，浏览到并选择 Amazon S3 存储桶的名称（例如，**amzn-s3-demo-bucket**）。
3. 您必须先删除存储桶的内容，然后才能删除存储桶。选择存储桶中的所有文件（如 **HelloWorld_App.zip**）。在 Actions 菜单中，选择 Delete。在提示确认删除时，选择 OK。
4. 在清空存储桶后，可以删除存储桶。在存储桶列表中，选择存储桶的行（而不是存储桶名称）。选择 Delete bucket，当系统提示进行确认时，选择 OK。

要从 CodeDeploy 中删除 `HelloWorld_App` 应用程序，请执行以下步骤：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 选择 **HelloWorld_App**。
4. 选择删除应用程序。
5. 当系统提示时，输入 **Delete**，然后选择删除。

要删除 Systems Manager 状态管理器关联，请执行以下操作：

1. 访问 https://console.aws.amazon.com/systems-manager，打开 Amazon Systems Manager 控制台。
2. 在导航窗格中，选择状态管理器。

3. 选择您创建的关联，然后选择删除。

接下来做什么？

此时，您已成功完成了使用 CodeDeploy 执行的部署。恭喜您！

教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）

本教程通过引导您将示例应用程序修订部署到运行 Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux（RHEL）的单个本地实例（即，不是 Amazon EC2 实例的物理设备）上，帮助您获得 CodeDeploy 的使用经验。有关本地实例以及如何将它们与 CodeDeploy 一起使用的信息，请参阅Working with On-Premises Instances。

不是您要找的内容？

- 要练习部署到运行 Amazon Linux 或 RHEL 的 Amazon EC2 实例，请参阅教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix）。
- 要练习部署到运行 Windows Server 的 Amazon EC2 实例，请参阅教程：使用 CodeDeploy 部署“Hello, World!” 带有 CodeDeploy（Windows 服务器）的应用程序。

主题

- 先决条件
- 步骤 1：配置本地实例
- 步骤 2：创建示例应用程序修订
- 步骤 3：打包您的应用程序修订并将其上传到 Amazon S3
- 步骤 4：部署应用程序修订
- 步骤 5：验证您的部署
- 步骤 6：清理资源

先决条件

在开始此教程之前，您必须完成 入门 CodeDeploy中的先决条件，包括配置用户、安装或升级 Amazon CLI 以及创建服务角色。您无需按照先决条件中的所述创建 IAM 实例配置文件。本地实例不使用 IAM 实例配置文件。

您将配置作为本地实例的物理设备必须运行 CodeDeploy 代理支持的操作系统 中列出的操作系统之一。

步骤 1：配置本地实例

您必须先配置本地实例，然后才能在其中进行部署。按照Working with On-Premises Instances中的说明操作，然后返回此页。

安装 CodeDeploy 代理

配置本地实例后，请按照安装 CodeDeploy 代理中针对本地实例的步骤操作，然后返回到此页。

步骤 2：创建示例应用程序修订

在这一步中，您将创建要部署到本地实例的示例应用程序修订。

由于很难了解您的本地实例上已经安装了哪些软件和功能 - 或者您的组织策略允许安装哪些软件和功能 - 因此，我们在此处提供的示例应用程序修订仅使用批处理脚本（对于 Windows Server）或 shell 脚本（对于 Ubuntu Server 和 RHEL），将文本文件写入您本地实例上的某个位置。对于多个 CodeDeploy 部署生命周期事件（包括 Install、AfterInstall、ApplicationStart 和 ValidateService），每个事件写入一个文件。在 BeforeInstall 部署生命周期事件中，将运行一个脚本来删除此示例在先前部署中写入的旧文件，并在您的本地实例上创建一个位置用于写入新文件。

> **Note**
>
> 如果出现以下任何情况，此示例应用程序修订可能无法部署：
>
> - 在本地实例上启动 CodeDeploy 代理的用户无权执行脚本。
> - 用户无权在脚本中列出的位置创建或删除文件夹。
> - 用户无权在脚本中列出的位置创建文本文件。

> **Note**
>
> 如果您配置了 Windows Server 实例并希望部署其他示例，则可能需要使用教程：使用 CodeDeploy 部署“Hello, World!” 带有 CodeDeploy（Windows 服务器）的应用程序教程的步骤 2：将您的源内容配置为部署到 Windows Server Amazon EC2 实例中的示例。
> 如果您配置了 RHEL 实例并希望部署其他示例，则可能需要使用教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix）教程的步骤 2：将源内容配置为部署到 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例中的示例。
> 目前，Ubuntu Server 没有替代示例。

1. 在您的开发计算机上，创建名为 `CodeDeployDemo-OnPrem` 的子目录（子文件夹）来存储示例应用程序修订的文件，然后切换到该子文件夹。在此示例中，我们假定您将使用 `c:\temp` 文件夹作为 Windows Server 的根文件夹，或者使用 `/tmp` 文件夹作为 Ubuntu Server 和 RHEL 的根文件夹。如果您使用其他文件夹，请务必在整个教程中使用该文件夹取代我们提供的文件夹：

   对于 Windows：

   ```
   mkdir c:\temp\CodeDeployDemo-OnPrem
   cd c:\temp\CodeDeployDemo-OnPrem
   ```

   对于 Linux、macOS 或 Unix：

   ```
   mkdir /tmp/CodeDeployDemo-OnPrem
   cd /tmp/CodeDeployDemo-OnPrem
   ```

2. 在 `CodeDeployDemo-OnPrem` 子文件夹的根中，使用文本编辑器创建两个分别名为 `appspec.yml` 和 `install.txt` 的文件：

   对于 Windows Server 为 `appspec.yml`：

   ```
   version: 0.0
   os: windows
   files:
     - source: .\install.txt
       destination: c:\temp\CodeDeployExample
   hooks:
     BeforeInstall:
   ```

```
    - location: .\scripts\before-install.bat
      timeout: 900
  AfterInstall:
    - location: .\scripts\after-install.bat
      timeout: 900
  ApplicationStart:
    - location: .\scripts\application-start.bat
      timeout: 900
  ValidateService:
    - location: .\scripts\validate-service.bat
      timeout: 900
```

对于 Ubuntu Server 和 RHEL 为 `appspec.yml`：

```
version: 0.0
os: linux
files:
  - source: ./install.txt
    destination: /tmp/CodeDeployExample
hooks:
  BeforeInstall:
    - location: ./scripts/before-install.sh
      timeout: 900
  AfterInstall:
    - location: ./scripts/after-install.sh
      timeout: 900
  ApplicationStart:
    - location: ./scripts/application-start.sh
      timeout: 900
  ValidateService:
    - location: ./scripts/validate-service.sh
      timeout: 900
```

有关 AppSpec 文件的更多信息，请参阅将应用程序规范文件添加到 CodeDeploy 的修订和CodeDeploy AppSpec 文件参考。

`install.txt`:

```
The Install deployment lifecycle event successfully completed.
```

3. 在 `CodeDeployDemo-OnPrem` 子文件夹的根下，创建 `scripts` 子文件夹，然后切换到该子文件夹：

对于 Windows：

```
mkdir c:\temp\CodeDeployDemo-OnPrem\scripts
cd c:\temp\CodeDeployDemo-OnPrem\scripts
```

对于 Linux、macOS 或 Unix：

```
mkdir -p /tmp/CodeDeployDemo-OnPrem/scripts
cd /tmp/CodeDeployDemo-OnPrem/scripts
```

4. 在 `scripts` 子文件夹的根中，使用文本编辑器创建 4 个文件（对于 Windows Server，它们分别名为 `before-install.bat`、`after-install.bat`、`application-start.bat` 和 `validate-service.bat`；对于 Ubuntu Server 和 RHEL，它们分别名为 `before-install.sh`、`after-install.sh`、`application-start.sh` 和 `validate-service.sh`）：

   对于 Windows Server：

   `before-install.bat`:

```
set FOLDER=%HOMEDRIVE%\temp\CodeDeployExample

if exist %FOLDER% (
  rd /s /q "%FOLDER%"
)

mkdir %FOLDER%
```

   `after-install.bat`:

```
cd %HOMEDRIVE%\temp\CodeDeployExample

echo The AfterInstall deployment lifecycle event successfully completed. > after-
install.txt
```

   `application-start.bat`:

```
cd %HOMEDRIVE%\temp\CodeDeployExample
```

```
echo The ApplicationStart deployment lifecycle event successfully completed. >
 application-start.txt
```

`validate-service.bat`:

```
cd %HOMEDRIVE%\temp\CodeDeployExample

echo The ValidateService deployment lifecycle event successfully completed. >
 validate-service.txt
```

对于 Ubuntu Server 和 RHEL：

`before-install.sh`:

```
#!/bin/bash
export FOLDER=/tmp/CodeDeployExample

if [ -d $FOLDER ]
then
 rm -rf $FOLDER
fi

mkdir -p $FOLDER
```

`after-install.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The AfterInstall deployment lifecycle event successfully completed." > after-
install.txt
```

`application-start.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ApplicationStart deployment lifecycle event successfully completed." >
 application-start.txt
```

validate-service.sh:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ValidateService deployment lifecycle event successfully completed." >
 validate-service.txt

unset FOLDER
```

5. 仅对于 Ubuntu Server 和 RHEL，确保四个 shell 脚本都具有执行权限：

```
chmod +x ./scripts/*
```

步骤 3：打包您的应用程序修订并将其上传到 Amazon S3

在部署应用程序修订之前，您需要打包文件，然后将文件包上传到 Amazon S3 存储桶。按照使用创建应用程序 CodeDeploy和将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）中的说明操作。（虽然您可以为应用程序和部署组提供任意名称，不过我们建议您使用 `CodeDeploy-OnPrem-App` 作为应用程序名称，并使用 `CodeDeploy-OnPrem-DG` 作为部署组名称。）在您完成这些指示的操作之后，返回本页。

> **Note**
>
> 或者，您可以将文件包上传到 GitHub 存储库并从该位置部署。有关更多信息，请参阅 将 CodeDeploy 与 GitHub 集成。

步骤 4：部署应用程序修订

在您将应用程序修订上传到 Amazon S3 存储桶之后，尝试将其部署到本地实例。按照使用创建部署 CodeDeploy中的说明操作，然后返回此页。

步骤 5：验证您的部署

要验证部署已经成功，请按照查看 CodeDeploy 部署详细信息 中的说明操作，然后返回本页。

如果部署已成功，您将在 `c:\temp\CodeDeployExample` 文件夹（对于 Windows Server）或 `/tmp/CodeDeployExample` 文件夹（对于 Ubuntu Server 和 RHEL）中找到 4 个文本文件。

如果部署失败，请按照View Instance Details和 排查实例问题中的排除故障步骤操作。进行任何必要的修复，重新打包并上传应用程序修订，然后再次尝试部署。

步骤 6：清理资源

为避免对您为此教程创建的资源继续收费，请删除您不再使用的 Amazon S3 存储桶。您还可以清理关联的资源，例如 CodeDeploy 中的应用程序和部署组记录，以及本地实例。

您可以使用 Amazon CLI 或结合使用 CodeDeploy 和 Amazon S3 控制台及 Amazon CLI 来清理资源。

清理资源（CLI）

删除 Amazon S3 存储桶

- 使用 `--recursive` 开关对存储桶调用 rm 命令（例如 `amzn-s3-demo-bucket`）。该存储桶及存储桶中的所有对象将被删除。

```
aws s3 rm s3://your-bucket-name --recursive --region region
```

删除 CodeDeploy 中的应用程序和部署组记录

- 对应用程序（例如，`CodeDeploy-OnPrem-App`）调用 delete-application 命令。将删除部署和部署组的记录。

```
aws deploy delete-application --application-name your-application-name
```

注销本地实例并删除 IAM 用户

- 对本地实例和区域调用 deregister 命令：

```
aws deploy deregister --instance-name your-instance-name --delete-iam-user --
region your-region
```

> **Note**
>
> 如果您不希望删除与此本地实例关联的 IAM 用户，请改为使用 `--no-delete-iam-user` 选项。

卸载 CodeDeploy 代理并从本地实例中删除配置文件

- 从本地实例调用 uninstall 命令：

```
aws deploy uninstall
```

现在您已完成清除此教程所用资源的全部步骤。

清理资源（控制台）

删除 Amazon S3 存储桶

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择要删除的存储桶旁边的图标（例如，`amzn-s3-demo-bucket`），但不要选择存储桶本身。
3. 选择操作，然后选择删除。
4. 在提示删除存储桶时，选择 OK。

删除 CodeDeploy 中的应用程序和部署组记录

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，选择应用程序。
3. 选择要删除的应用程序的名称（例如，`CodeDeploy-OnPrem-App`），然后选择删除应用程序。
4. 在系统提示时，输入应用程序的名称以确认要删除应用程序，然后选择删除。

您不能使用 Amazon CodeDeploy 控制台取消注册本地实例或者卸载 CodeDeploy 代理。按照注销本地实例并删除 IAM 用户 中的说明进行操作。

教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组

在本教程中，您将使用 CodeDeploy 将应用程序修订部署到 Auto Scaling 组。Amazon EC2 Auto Scaling 使用预定义条件启动 Amazon EC2 实例，然后在不再需要这些实例时将其终止。Amazon EC2 Auto Scaling 通过确保 CodeDeploy 始终具有正确数量的 Amazon EC2 实例以处理部署负载，从而帮助 CodeDeploy 扩展。有关 Amazon EC2 Auto Scaling 与 CodeDeploy 集成的信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

主题

- 先决条件
- 步骤 1：创建和配置 Auto Scaling 组
- 步骤 2：将应用程序部署到 Auto Scaling 组
- 步骤 3：检查结果
- 步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数量
- 步骤 5：再次检查结果
- 步骤 6：清除

先决条件

在本教程中遵循以下步骤操作：

- 完成入门 CodeDeploy中的所有步骤，包括设置和配置 Amazon CLI 以及创建 IAM 实例配置文件（**`CodeDeployDemo-EC2-Instance-Profile`**）和服务角色（**`CodeDeployDemo`**）。服务角色是一种特殊类型的 IAM 角色，用于为服务提供代表您执行操作的权限。
- 如果您使用启动模板创建 Auto Scaling 组，则必须添加以下权限：
  - `ec2:RunInstances`
  - `ec2:CreateTags`
  - `iam:PassRole`

  有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 步骤 2：创建服务角色、为 Auto Scaling 组创建启动模板和启动模板支持。
- 创建和使用与 Ubuntu Server 实例和 CodeDeploy 兼容的修订。对于您的修订，可以执行以下操作之一：
  - 创建并使用步骤 2：创建示例应用程序修订教程的教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）中的示例修订。

- 自行创建修订，具体请参阅正在处理的应用程序修订版 CodeDeploy。
- 使用以下入站规则创建名为 **CodeDeployDemo-AS-SG** 的安全组：
  - 类型：HTTP
  - 源：任何位置

  这是查看您的应用程序和验证部署是否成功所必需的。有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的创建安全组。

步骤 1：创建和配置 Auto Scaling 组

在本步骤中，您将创建一个包含单个 Amazon Linux、RHEL 或 Windows Server Amazon EC2 实例的 Auto Scaling 组。在后面的步骤中，您将指示 Amazon EC2 Auto Scaling 再添加一个 Amazon EC2 实例，并且 CodeDeploy 将向其中部署您的修订。

主题

- 创建和配置 Auto Scaling 组（CLI）
- 创建和配置 Auto Scaling 组（控制台）

创建和配置 Auto Scaling 组（CLI）

1. 调用 create-launch-template 命令创建 Amazon EC2 启动模板。

   在调用此命令之前，您需要适用于本教程的 AMI 的 ID（由占位符 *image-id* 表示）。您还需要 Amazon EC2 实例密钥对的名称（由占位符 *key-name* 表示）才能访问 Amazon EC2 实例。

   要获取适用于本教程的 AMI 的 ID，请执行以下操作：

   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
   b. 在导航窗格中的 Instances 下，选择 Instances，然后选择 Launch Instance。
   c. 在选择一个 Amazon 系统映像页上的快速启动选项卡上，记下 Amazon Linux 2 AMI、Red Hat Enterprise Linux 7.1、Ubuntu Server 14.04 LTS 或 Microsoft Windows Server 2012 R2 旁边的 AMI 的 ID。

> **Note**
>
> 如果您拥有与 CodeDeploy 兼容的 AMI 自定义版本，则在此选择它，而不用浏览快速启动选项卡。有关将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 配合使用的信息，请参阅将自定义 AMI 与 CodeDeploy Amazon A EC2 uto Scaling 配合使用。

对于 Amazon EC2 实例密钥对，使用 Amazon EC2 实例密钥对的名称。

调用 create-launch-template 命令。

在本地 Linux、macOS 或 Unix 计算机上：

```
aws ec2 create-launch-template \
  --launch-template-name CodeDeployDemo-AS-Launch-Template \
  --launch-template-data file://config.json
```

文件 `config.json` 的内容：

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

在本地 Windows 计算机上：

```
aws ec2 create-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
Template --launch-template-data file://config.json
```

文件 `config.json` 的内容：

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
```

```
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

这些命令与 `config.json` 文件一起为您的 Auto Scaling 组创建名为 CodeDeployDemo-AS-Launch-Template 的 Amazon EC2 启动模板，该模板将在接下来的步骤中基于 t1.micro Amazon EC2 实例类型创建。根据您对 `ImageId`、`IamInstanceProfile` 和 `KeyName` 的输入，启动模板还指定 AMI ID、启动时要传递给实例的 IAM 角色关联的实例配置文件名称，以及连接到实例时要使用的 Amazon EC2 密钥对。

2. 调用 create-auto-scaling-group 命令以创建一个 Auto Scaling 组。您将需要《Amazon Web Services 一般参考》的区域和终端节点所列区域之一中的某个可用区的名称（由占位符 *`availability-zone`* 表示）。

> **Note**
>
> 要查看区域中的可用区列表，请调用：
>
> ```
> aws ec2 describe-availability-zones --region region-name
> ```
>
> 例如，要查看美国西部（俄勒冈州）区域中的可用区列表，请调用：
>
> ```
> aws ec2 describe-availability-zones --region us-west-2
> ```
>
> 有关区域名称标识符的列表，请参阅各区域的资源工具包存储桶名称。

在本地 Linux、macOS 或 Unix 计算机上：

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --launch-template CodeDeployDemo-AS-Launch-Template,Version='$Latest' \
  --min-size 1 \
  --max-size 1 \
  --desired-capacity 1 \
  --availability-zones availability-zone \
  --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

在本地 Windows 计算机上：

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name
 CodeDeployDemo-AS-Group --launch-template LaunchTemplateName=CodeDeployDemo-
AS-Launch-Template,Version="$Latest" --min-size 1 --max-size 1 --
desired-capacity 1 --availability-zones availability-zone --tags
 Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

这些命令基于名为 **CodeDeployDemo-AS-Launch-Template** 的 Amazon EC2 启动模板创建名为 **CodeDeployDemo-AS-Group** 的 Auto Scaling 组。此 Auto Scaling 组只有一个在指定可用区中创建的 Amazon EC2 实例。此 Auto Scaling 组中的每个实例都具有标签 `Name=CodeDeployDemo`。稍后安装 CodeDeploy 代理时将使用该标签。

3. 针对 describe-auto-scaling-groups 调用 **CodeDeployDemo-AS-Group** 命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
 LifecycleState]" --output text
```

请在返回的值显示 `Healthy` 和 `InService` 之后继续。

4. Auto Scaling 组中的实例必须安装 CodeDeploy 代理才能在 CodeDeploy 部署中使用。通过使用标签从 Amazon Systems Manager 调用 create-association 命令来安装 CodeDeploy 代理，该标签是在创建 Auto Scaling 组时添加的。

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \

 --parameters action=Install, name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

此命令在 Systems Manager 状态管理器中创建一个关联，这将在 Auto Scaling 组中的所有实例上安装 CodeDeploy 代理，然后尝试在每周日凌晨 2:00 更新它。有关 CodeDeploy 代理的详细信息，请参阅使用 CodeDeploy 代理。有关 Systems Manager 的详细信息，请参阅什么是 Amazon Systems Manager。

创建和配置 Auto Scaling 组（控制台）

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在全局导航栏中，确保选中《Amazon Web Services 一般参考》的区域和终端节点中列出的某个区域。Amazon EC2 Auto Scaling 资源与您所指定的区域相关联，并且 CodeDeploy 仅在选定区域中受支持。
3. 在导航栏中的实例下，选择启动模板。
4. 选择Create launch template（创建启动模板）。
5. 在启动模板名称和描述对话框中，为启动模板名称输入 **`CodeDeployDemo-AS-Launch-Template`**。对其他字段保留默认值。
6. 在亚马逊机器映像（AMI）对话框中，单击 AMI 下的下拉列表，选择适用于本教程的 AMI：
   - 在 AMI 下拉列表的快速启动选项卡上，选择以下选项之一：Amazon Linux 2 AMI、Red Hat Enterprise Linux 7.1、Ubuntu Server 14.04 LTS 或 Microsoft Windows Server 2012 R2。

     > **Note**
     >
     > 如果您拥有与 CodeDeploy 兼容的 AMI 自定义版本，则在此选择它，而不用浏览快速启动选项卡。有关将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 配合使用的信息，请参阅将自定义 AMI 与 CodeDeploy Amazon A EC2 uto Scaling 配合使用。
7. 在实例类型中，选择下拉列表，然后选择 t1.micro。您可以使用搜索栏来更快地找到它。
8. 在密钥对（登录）对话框中，选择选择现有密钥对。在选择密钥对下拉列表中，选择您在前面步骤中创建或使用的 Amazon EC2 实例密钥对。
9. 在网络设置对话框中，选择虚拟公有云（VPC）。

   在安全组下拉列表中，选择您在教程的先决条件部分中创建的安全组（**`CodeDeployDemo-AS-SG`**）。
10. 展开高级详细信息对话框。在 IAM 实例配置文件下拉列表中，在 IAM 实例配置文件下选择您之前创建的 IAM 角色（**`CodeDeployDemo-EC2-Instance-Profile`**）。

    保留其余的默认值。
11. 选择Create launch template（创建启动模板）。
12. 在后续步骤对话框中，选择创建 Auto Scaling 组。

13. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，键入 **CodeDeployDemo-AS-Group**。
14. 在启动模板对话框中，应该已经填充您的启动模板（**CodeDeployDemo-AS-Launch-Template**），如果没有，请从下拉菜单中将其选中。保留默认选择，然后选择下一步。
15. 在选择实例启动选项页面的网络部分，对于 VPC，选择默认的 VPC。然后为可用区和子网选择默认子网。如果无法选择默认 VPC，则必须创建 VPC。有关更多信息，请参阅 Amazon VPC 入门。
16. 在 Instance type requirements（实例类型要求）部分中，使用默认设置简化此步骤。（请勿覆盖启动模板。）在本教程中，您将仅使用启动模板中指定的实例类型启动按需实例。
17. 选择 Next（下一步）转至 Configure advanced options（配置高级选项）页面。
18. 保持默认值，然后选择下一步。
19. 在配置组大小和扩展策略页面上，保留默认组大小值 1。选择下一步。
20. 跳过配置通知的步骤，然后选择下一步。
21. 在添加标签页面上，添加一个标签，以便稍后安装 CodeDeploy 代理时使用。选择 Add tag（添加标签）。

    a. 在键中，输入 **Name**。

    b. 在值中，输入 **CodeDeployDemo**。

    选择下一步。
22. 在审核页面上，检查 Auto Scaling 组的详细信息，然后选择创建 Auto Scaling 组。
23. 在导航栏中，在 Auto Scaling 组处于选中状态的情况下，选择 **CodeDeployDemo-AS-Group**，然后选择实例管理选项卡。请在 InService 的值出现在 Lifecycle 列中并且 Healthy 的值出现在 Health Status 列中之后继续。
24. 按照安装 CodeDeploy 代理中的步骤并使用 `Name=CodeDeployDemo` 实例标签安装 CodeDeploy 代理。

步骤 2：将应用程序部署到 Auto Scaling 组

在此步骤中，您会将修订部署到 Auto Scaling 组中的单个 Amazon EC2 实例。

主题

- 创建部署（CLI）
- 创建部署（控制台）

创建部署（CLI）

1. 调用 create-application 命令以创建一个名为 **SimpleDemoApp** 的应用程序。

```
aws deploy create-application --application-name SimpleDemoApp
```

2. 您应该已经按照步骤 2：为创建服务角色 CodeDeploy中的说明创建了一个服务角色。该服务角色将向 CodeDeploy 授予访问 Amazon EC2 实例以展开（读取）其标签的权限。您将需要服务角色 ARN。要获取服务角色 ARN，请按照获取服务角色 ARN（CLI） 中的说明操作。

3. 现在您已拥有一个服务角色 ARN，请调用 create-deployment-group 命令，使用名为 **CodeDeployDemo-AS-Group** 的 Auto Scaling 组和名为 **CodeDeployDefault.OneAtATime** 的部署配置创建一个与名为 **SimpleDemoApp** 的应用程序相关联的名为 **SimpleDemoDG** 的部署组（具有指定的服务角色 ARN）。

> **Note**
>
> create-deployment-group 命令支持创建触发器，触发器会向主题订阅者发送有关部署和实例中指定事件的 Amazon SNS 通知。该命令还支持用于自动回滚部署和设置警报的选项，以便在达到 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中不包含用于这些操作的命令。

在本地 Linux、macOS 或 Unix 计算机上：

```
aws deploy create-deployment-group \
  --application-name SimpleDemoApp \
  --auto-scaling-groups CodeDeployDemo-AS-Group \
  --deployment-group-name SimpleDemoDG \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --service-role-arn service-role-arn
```

在本地 Windows 计算机上：

```
aws deploy create-deployment-group --application-name SimpleDemoApp --auto-scaling-
groups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG --deployment-
config-name CodeDeployDefault.OneAtATime --service-role-arn service-role-arn
```

4. 调用 create-deployment 命令，使用指定位置的修订创建一个与名为 **SimpleDemoApp** 的应用程序、名为 **CodeDeployDefault.OneAtATime** 的部署配置、名为 **SimpleDemoDG** 的部署组相关联的部署。

   对于 Amazon Linux 和 RHEL Amazon EC2 实例，从本地 Linux、macOS 或 Unix 计算机调用

   对于中国（北京）区域：

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=aws-codedeploy-cn-north-1,bundleType=zip,key=samples/latest/
SampleApp_Linux.zip
```

   对于中国（宁夏）区域：

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=aws-codedeploy-cn-northwest-1,bundleType=zip,key=samples/
latest/SampleApp_Linux.zip
```

   对于 Amazon Linux 和 RHEL Amazon EC2 实例，从本地 Windows 计算机调用

   对于中国（北京）区域：

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=aws-codedeploy-cn-north-1,bundleType=zip,key=samples/latest/
SampleApp_Linux.zip
```

   对于中国（宁夏）区域：

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
```

```
location bucket=aws-codedeploy-cn-northwest-1,bundleType=zip,key=samples/latest/
SampleApp_Linux.zip
```

对于 Windows Server Amazon EC2 实例，从本地 Linux、macOS 或 Unix 计算机调用

对于中国（北京）区域：

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=aws-codedeploy-cn-north-1,bundleType=zip,key=samples/latest/
SampleApp_Windows.zip
```

对于中国（宁夏）区域：

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=aws-codedeploy-cn-northwest-1,bundleType=zip,key=samples/
latest/SampleApp_Windows.zip
```

对于 Windows Server Amazon EC2 实例，从本地 Windows 计算机调用

对于中国（北京）区域：

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=aws-codedeploy-cn-north-1,bundleType=zip,key=samples/latest/
SampleApp_Windows.zip
```

对于中国（宁夏）区域：

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=aws-codedeploy-cn-northwest-1,bundleType=zip,key=samples/latest/
SampleApp_Windows.zip
```

> **Note**
>
> 当前，CodeDeploy 不提供要部署到 Ubuntu Server Amazon EC2 实例的示例修订。要自己创建修订，请参阅正在处理的应用程序修订版 CodeDeploy。

5. 调用 get-deployment 命令以确保部署已成功。

   在调用此命令之前，您需要应该已经通过调用 create-deployment 命令返回的部署 ID。如果需要再次获取部署 ID，请针对名为 list-deployments 的应用程序和名为 **SimpleDemoApp** 的部署组调用 **SimpleDemoDG** 命令：

```
aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-
name SimpleDemoDG --query "deployments" --output text
```

   现在，使用部署 ID 调用 get-deployment 命令：

```
aws deploy get-deployment --deployment-id deployment-id --query
 "deploymentInfo.status" --output text
```

   请在返回的值为 `Succeeded` 之后继续。

创建部署（控制台）

1. 您应该已经按照步骤 2：为创建服务角色 CodeDeploy中的说明创建了一个服务角色。该服务角色将向 CodeDeploy 授予访问实例以展开（读取）其标签的权限。在使用 CodeDeploy 控制台部署应用程序修订之前，您需要服务角色 ARN。要获取服务角色 ARN，请按照获取服务角色 ARN（控制台） 中的说明操作。
2. 现在您已拥有服务角色 ARN，可以开始使用 CodeDeploy 控制台部署您的应用程序修订。

   登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

3. 在导航窗格中，展开部署，然后选择应用程序。

4. 选择创建应用程序。
5. 选择自定义应用程序。
6. 在 Application name（应用程序名称）中，输入 **SimpleDemoApp**。
7. 在计算平台中，选择 EC2/本地。
8. 选择创建应用程序。
9. 在部署组选项卡中，选择创建部署组。
10. 在 Deployment group name（部署组名称）中，输入 **SimpleDemoDG**。
11. 在服务角色中，选择您的服务角色的名称。
12. 在部署类型中，选择就地。
13. 在环境配置中，选择 Auto Scaling 组，然后选择 **CodeDeployDemo-AS-Group**。
14. 在部署配置中，选择 CodeDeployDefault.OneAtATime。
15. 清除启用负载均衡。
16. 选择 Create deployment group（创建部署组）。
17. 在“部署组”页面中，选择创建部署。
18. 对于修订类型，选择我的应用程序存储在 Amazon S3 中。
19. 在修订位置中，输入您的操作系统的示例应用程序的位置和区域。

    对于 Amazon Linux 和 RHEL Amazon EC2 实例

    对于中国（北京）区域：

    ```
    https://s3.cn-north-1.amazonaws.com.cn/aws-codedeploy-cn-north-1/samples/latest/
    SampleApp_Linux.zip
    ```

    对于中国（宁夏）区域：

    ```
    https://s3.cn-northwest-1.amazonaws.com.cn/aws-codedeploy-cn-northwest-1/samples/
    latest/SampleApp_Linux.zip
    ```

    对于 Windows Server Amazon EC2 实例

    对于中国（北京）区域：

    ```
    https://s3.cn-north-1.amazonaws.com.cn/aws-codedeploy-cn-north-1/samples/latest/
    SampleApp_Windows.zip
    ```

对于中国（宁夏）区域：

```
https://s3.cn-northwest-1.amazonaws.com.cn/aws-codedeploy-cn-northwest-1/samples/
latest/SampleApp_Windows.zip
```

对于 Ubuntu Server Amazon EC2 实例

键入存储在 Amazon S3 中的自定义应用程序修订的位置。

20. 保留部署描述为空。
21. 展开高级。
22. 选择 Create deployment（创建部署）。

> **Note**
>
> 如果显示失败而不是成功，则您可能需要尝试监控您的部署并排除故障中的一些方法（使用应用程序名称 **SimpleDemoApp** 和部署组名称 **SimpleDemoDG**）。

步骤 3：检查结果

在此步骤中，您将检查 CodeDeploy 是否已在 Auto Scaling 组中的单个 Amazon EC2 实例上安装 **SimpleDemoApp** 修订。

主题

- 检查结果（CLI）
- 检查结果（控制台）

检查结果（CLI）

首先，您将需要 Amazon EC2 实例的公有 DNS。

使用 Amazon CLI 通过调用 describe-instances 命令来获取 Auto Scaling 组中的 Amazon EC2 实例的公有 DNS。

在调用此命令之前，您将需要 Amazon EC2 实例的 ID。要获取此 ID，请如前一样针对 describe-auto-scaling-groups 调用 **CodeDeployDemo-AS-Group**：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names CodeDeployDemo-
AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --output text
```

现在调用 describe-instances 命令：

```
aws ec2 describe-instances --instance-id instance-id --query
 "Reservations[0].Instances[0].PublicDnsName" --output text
```

返回的值是 Amazon EC2 实例的公有 DNS。

使用如下所示的 URL，在 Web 浏览器中显示部署到该 Amazon EC2 实例的 SimpleDemoApp 修订：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果您看到恭喜页面，则表示您已成功使用 CodeDeploy 将修订部署到 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，您将一个 Amazon EC2 实例添加到 Amazon EC2 组中。在 Amazon EC2 Auto Scaling 添加 Amazon EC2 实例后，CodeDeploy 会将您的修订部署到新实例。

检查结果（控制台）

首先，您将需要 Amazon EC2 实例的公有 DNS。

通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

在 Amazon EC2 导航窗格的 Auto Scaling 下，选择 Auto Scaling 组，然后选择 **CodeDeployDemo-AS-Group** 条目。

在实例选项卡上，选择列表中的 Amazon EC2 实例 ID。

在 Instances 页中的 Description 选项卡上，记下 Public DNS 值。它看上去应与下类似：**ec2-01-234-567-890.compute-1.amazonaws.com**。

使用如下所示的 URL，在 Web 浏览器中显示部署到该 Amazon EC2 实例的 SimpleDemoApp 修订：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果您看到恭喜页面，则表示您已成功使用 CodeDeploy 将修订部署到 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，将一个 Amazon EC2 实例添加到 Amazon EC2 组中。在 Amazon EC2 Auto Scaling 添加 Amazon EC2 实例后，CodeDeploy 会将您的修订部署到新 Amazon EC2 实例。

步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数量

在此步骤中，指示 Auto Scaling 组创建其他 Amazon EC2 实例。在 Amazon EC2 Auto Scaling 创建实例后，CodeDeploy 会将您的修订部署到该实例。

主题

- 扩展 Auto Scaling 组中的 Amazon EC2 实例数（CLI）
- 扩展部署组中的 Amazon EC2 实例数（控制台）

扩展 Auto Scaling 组中的 Amazon EC2 实例数（CLI）

1. 调用 update-auto-scaling-group 命令以将名为 **`CodeDeployDemo-AS-Group`** 的 Auto Scaling 组中的 Amazon EC2 实例数量从 1 增至 2。

   在本地 Linux、macOS 或 Unix 计算机上：

   ```
   aws autoscaling update-auto-scaling-group \
     --auto-scaling-group-name CodeDeployDemo-AS-Group \
     --min-size 2 \
     --max-size 2 \
     --desired-capacity 2
   ```

   在本地 Windows 计算机上：

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-
   AS-Group --min-size 2 --max-size 2 --desired-capacity 2
   ```

2. 确保 Auto Scaling 组现在有两个 Amazon EC2 实例。针对 describe-auto-scaling-groups 调用 **`CodeDeployDemo-AS-Group`** 命令：

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
    CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
    LifecycleState]" --output text
   ```

   请在返回的值显示 `Healthy` 和 `InService` 之后继续。

扩展部署组中的 Amazon EC2 实例数（控制台）

1. 在 Amazon EC2 导航栏的 Auto Scaling 下，选择 Auto Scaling 组，然后选择 **CodeDeployDemo-AS-Group**。
2. 选择操作，然后选择编辑。
3. 在 Details（详细信息）选项卡上的 Desired（所需数量）、Min（最小数量）和 Max（最大数量）框中，键入 **2**，然后选择 Save（保存）。
4. 选择 Instances 选项卡。新的 Amazon EC2 实例应该会出现在列表中。（如果该实例未出现，您可能需要选择几次 Refresh 按钮。）请在 InService 的值出现在 Lifecycle 列中并且 Healthy 的值出现在 Health Status 列中之后继续。

步骤 5：再次检查结果

在此步骤中，您将检查 CodeDeploy 是否已在 Auto Scaling 组中的新实例上安装 SimpleDemoApp 修订。

主题

- 检查自动部署结果（CLI）
- 检查自动部署结果（控制台）

检查自动部署结果（CLI）

1. 在调用 get-deployment 命令之前，您将需要自动部署的 ID。要获取 ID，请针对名为 list-deployments 的应用程序和名为 **SimpleDemoApp** 的部署组调用 **SimpleDemoDG** 命令：

   ```
   aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-name SimpleDemoDG --query "deployments" --output text
   ```

   应该有两个部署 ID。在对 get-deployment 命令的调用中使用您尚未使用的 ID：

   ```
   aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.[status, creator]" --output text
   ```

   除了部署状态外，在命令输出中，您应该还会看到 `autoScaling`。（`autoScaling` 意味着 Amazon EC2 Auto Scaling 已创建部署。）

请在部署状态显示 Succeeded 之后继续。

2. 在调用 describe-instances 命令之前，您将需要新的 Amazon EC2 实例的 ID。要获取此 ID，请针对 describe-auto-scaling-groups 再次调用 **CodeDeployDemo-AS-Group** 命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --
output text
```

现在调用 describe-instances 命令：

```
aws ec2 describe-instances --instance-id instance-id --query
 "Reservations[0].Instances[0].PublicDnsName" --output text
```

在 describe-instances 命令输出中，记下新的 Amazon EC2 实例的公有 DNS。

3. 使用如下所示的 URL，在 Web 浏览器中显示部署到该 Amazon EC2 实例的 SimpleDemoApp 修订：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果出现恭喜页面，则表示您已使用 CodeDeploy 将修订部署到 Auto Scaling 组中扩展的 Amazon EC2 实例！

检查自动部署结果（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

> **Note**
>
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择部署。

3. 选择 Amazon EC2 Auto Scaling 创建的部署的部署 ID。

.

4. 部署页显示有关部署的信息。通常，您会自己创建部署，但 Amazon EC2 Auto Scaling 会代表您创建一个部署以将您的修订部署到新的 Amazon EC2 实例。
5. 页面顶部显示成功之后，在实例上验证结果。您首先需要获取实例的公有 DNS：
6. 在 Amazon EC2 导航窗格的 Auto Scaling 下，选择 Auto Scaling 组，然后选择 **CodeDeployDemo-AS-Group** 条目。
7. 在实例选项卡上，选择新的 Amazon EC2 实例的 ID。
8. 在 Instances 页中的 Description 选项卡上，记下 Public DNS 值。它看上去应与下类似：**ec2-01-234-567-890.compute-1.amazonaws.com**。

使用如下所示的 URL，显示部署到该实例的 `SimpleDemoApp` 修订：

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

如果出现恭喜页面，则表示您已使用 CodeDeploy 将修订部署到 Auto Scaling 组中扩展的 Amazon EC2 实例！

步骤 6：清除

在此步骤中，您将删除 Auto Scaling 组，以避免对您在本教程中使用的资源持续收费。（可选）您可以删除 Auto Scaling 配置和 CodeDeploy 部署组件记录。

主题

- 清除资源（CLI）
- 清除资源（控制台）

清除资源（CLI）

1. 通过针对 **CodeDeployDemo-AS-Group** 调用 delete-auto-scaling-group 命令来删除 Auto Scaling 组。这同时将终止 Amazon EC2 实例。

   ```
   aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete
   ```

2. （可选）通过对名为 **CodeDeployDemo-AS-Launch-Template** 的启动配置调用 delete-launch-template 命令，删除 Auto Scaling 启动模板：

```
aws ec2 delete-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
Template
```

3. （可选）通过对名为 **SimpleDemoApp** 的应用程序调用 delete-application 命令，从 CodeDeploy 中删除应用程序。这同时将删除所有关联的部署、部署组和修订记录。

```
aws deploy delete-application --application-name SimpleDemoApp
```

4. 要删除 Systems Manager 状态管理器关联，请调用 delete-association 命令。

```
aws ssm delete-association --assocation-id association-id
```

您可以通过调用 describe-association 命令来获取 *association-id*。

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

清除资源（控制台）

要删除 Auto Scaling 组，同时终止 Amazon EC2 实例，请执行以下操作：

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在 Amazon EC2 导航窗格的 Auto Scaling 下，选择 Auto Scaling 组，然后选择 **CodeDeployDemo-AS-Group** 条目。
3. 依次选择 Actions、Delete 和 Yes, Delete。

（可选）要删除启动模板，请执行以下操作：

1. 在导航栏中的 Auto Scaling 下，选择启动配置，然后选择 **CodeDeployDemo-AS-Launch-Template**。
2. 依次选择 Actions、Delete launch configuration 和 Yes, Delete。

1. （可选）从 CodeDeploy 中删除应用程序。这同时将删除所有关联的部署、部署组和修订记录。从 https://console.aws.amazon.com/codedeploy 打开 CodeDeploy 控制台。

2. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

   在导航窗格中，展开部署，然后选择应用程序。

3. 在应用程序列表中，选择 SimpleDemoApp。
4. 在 Application details 页上，选择 Delete application。
5. 当系统提示时，输入 **`Delete`**，然后选择删除。

要删除 Systems Manager 状态管理器关联，请执行以下操作：

1. 访问 https://console.aws.amazon.com/systems-manager，打开 Amazon Systems Manager 控制台。
2. 在导航窗格中，选择状态管理器。
3. 选择您创建的关联，然后选择删除。

教程：使用 CodeDeploy 从 GitHub 部署应用程序

在本教程中，您将使用 CodeDeploy 将示例应用程序修订从 GitHub 部署到一个运行 Amazon Linux 的 Amazon EC2 实例、一个 Red Hat Enterprise Linux（RHEL）实例或一个 Windows Server 实例。有关 GitHub 与 CodeDeploy 的集成的信息，请参阅将 CodeDeploy 与 GitHub 集成。

> **Note**
>
> 您还可使用 CodeDeploy 将应用程序修订从 GitHub 部署到 Ubuntu Server 实例。您可使用教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）中的步骤 2：创建示例应用程序修订中描述的示例修订，也可创建与 Ubuntu Server 实例和 CodeDeploy 兼容的修订。要创建您自己的修订，请参阅为 CodeDeploy 规划修订和将应用程序规范文件添加到 CodeDeploy 的修订。

主题

- 先决条件
- 步骤 1：设置 GitHub 账户
- 步骤 2：创建 GitHub 存储库
- 步骤 3：将示例应用程序上传到 GitHub 存储库
- 步骤 4：预置实例
- 步骤 5：创建应用程序和部署组
- 步骤 6：将应用程序部署到实例
- 步骤 7：监控和验证部署
- 第 8 步：清除

先决条件

在开始本教程之前，请执行以下操作：

- 在本地计算机上安装 Git。要安装 Git，请参阅 Git 下载。
- 完成 入门 CodeDeploy中的步骤，包括安装和配置 Amazon CLI。这在您需要使用 Amazon CLI 将修订从 GitHub 部署到实例时特别重要。

步骤 1：设置 GitHub 账户

您需要拥有 GitHub 账户才能创建用于存储修订的 GitHub 存储库。如果您已拥有 GitHub 账户，请向前跳至步骤 2：创建 GitHub 存储库。

1. 转至 https://github.com/join。
2. 键入用户名、您的电子邮件地址和密码。
3. 选择 Sign up for GitHub，然后按照说明执行操作。

步骤 2：创建 GitHub 存储库

您将需要 GitHub 存储库来存储修订。

如果您已拥有 GitHub 存储库，请确保在本教程中用其名称替换 **`CodeDeployGitHubDemo`**，然后向前跳至步骤 3：将示例应用程序上传到 GitHub 存储库。

1. 在 GitHub 主页上，执行下列操作之一：
   - 在 Your repositories 中，选择 New repository。
   - 在导航栏上，选择 Create new（+），然后选择 New repository。
2. 在 Create a new repository 页上，执行下列操作：
   - 在存储库名称框中，输入 **`CodeDeployGitHubDemo`**。
   - 选择 Public。

     > **Note**
     > 选择默认的 Public（公有）选项意味着任何人均可查看此存储库。您可以选择 Private（私有）选项，限制可对存储库执行查看和提交的人员。

   - 清除 Initialize this repository with a README（使用自述文件初始化此存储库）复选框。您将改为在下一步中手动创建 `README.md` 文件。
   - 选择创建存储库。
3. 按照适用于您的本地计算机类型的说明操作以使用命令行创建存储库。

   > **Note**
   > 如果您已在 GitHub 上启用双重验证，请确保输入您的个人访问令牌而不是您的 GitHub 登录密码（如果系统提示输入密码）。有关信息，请参阅提供您的双重身份验证代码。

在本地 Linux、macOS 或 Unix 计算机上：

1. 从终端中运行以下命令（一次运行一条命令），其中 *`user-name`* 是您的 GitHub 用户名：

```
mkdir /tmp/CodeDeployGitHubDemo
```

```
cd /tmp/CodeDeployGitHubDemo
```

```
touch README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

2. 在 `/tmp/CodeDeployGitHubDemo` 位置保持终端打开。

在本地 Windows 计算机上：

1. 以管理员身份从命令提示符运行以下命令（一次运行一条命令）：

```
mkdir c:\temp\CodeDeployGitHubDemo
```

```
cd c:\temp\CodeDeployGitHubDemo
```

```
notepad README.md
```

2. 在记事本中，保存 `README.md` 文件。关闭记事本。运行以下命令（一次运行一条命令），其中 *user-name* 是您的 GitHub 用户名：

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

3. 在 `c:\temp\CodeDeployGitHubDemo` 位置保持终端打开。

步骤 3：将示例应用程序上传到 GitHub 存储库

在此步骤中，您将示例修订从公有 Amazon S3 存储桶中复制到 GitHub 存储库。（为简单起见，为本教程提供的示例修订为单一网页。）

> **Note**
>
> 如果您使用您的某个修订而不是我们的示例修订，则您的修订必须：
>
> - 遵循为 CodeDeploy 规划修订和将应用程序规范文件添加到 CodeDeploy 的修订中的准则。
> - 使用相应的实例类型。
> - 可从您的 GitHub 仪表板访问。
>
> 如果您的修订满足这些要求，请向前跳至步骤 5：创建应用程序和部署组。
> 如果您要部署到 Ubuntu Server 实例，则需将与 Ubuntu Server 实例和 CodeDeploy 兼容的修订上传到您的 GitHub 存储库。有关更多信息，请参阅为 CodeDeploy 规划修订和将应用程序规范文件添加到 CodeDeploy 的修订。

主题

- 从本地 Linux、macOS 或 Unix 计算机推送示例修订
- 从本地 Windows 计算机推送示例修订

从本地 Linux、macOS 或 Unix 计算机推送示例修订

例如，在终端在 `/tmp/CodeDeployGitHubDemo` 位置仍处于打开状态的情况下，运行以下命令（一次运行一条命令）：

> **Note**
>
> 如果您计划部署到 Windows Server 实例，请在命令中使用 `SampleApp_Windows.zip` 替换 `SampleApp_Linux.zip`。

```
(Amazon S3 copy command)
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中，*#Amazon S3 copy command#*为下列项之一：

- 中国（北京）区域为 `aws s3 cp s3://aws-codedeploy-cn-north-1/samples/latest/SampleApp_Linux.zip . --region cn-north-1`
- 中国（宁夏）区域为 `aws s3 cp s3://aws-codedeploy-cn-northwest-1/samples/latest/SampleApp_Linux.zip . --region cn-northwest-1`

从本地 Windows 计算机推送示例修订

例如，在命令提示符在 `c:\temp\CodeDeployGitHubDemo` 位置仍处于打开状态的情况下，运行以下命令（一次运行一条命令）：

> **Note**
>
> 如果您计划部署到 Amazon Linux 或 RHEL 实例，请在命令中使用 `SampleApp_Linux.zip` 替换 `SampleApp_Windows.zip`。

```
(Amazon S3 copy command)
```

将 ZIP 文件的内容直接解压缩到本地目录（the例如 `c:\temp\CodeDeployGitHubDemo`），而不要解压缩到一个新的子目录中。

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中，*#Amazon S3 copy command#*为下列项之一：

- 中国（北京）区域为 `aws s3 cp s3://aws-codedeploy-cn-north-1/samples/latest/SampleApp_Windows.zip . --region cn-north-1`
- 中国（宁夏）区域为 `aws s3 cp s3://aws-codedeploy-cn-northwest-1/samples/latest/SampleApp_Windows.zip . --region cn-northwest-1`

要将您自己的修订推送到 Ubuntu Server 实例，请将修订复制到您的本地存储库，然后调用：

```
git add .
git commit -m "Added Ubuntu app"
git push
```

步骤 4：预置实例

在此步骤中，您将创建或配置示例应用程序将部署到的实例。您可以部署到 Amazon EC2 实例或本地实例（运行 CodeDeploy 支持的操作系统之一）。有关信息，请参阅 CodeDeploy 代理支持的操作系统。（如果您已有配置为在 CodeDeploy 部署中使用的实例，请跳至下一步。）

预置实例

1. 按照启动 Amazon EC2 实例（控制台）中的说明预置实例。
2. 启动实例时，请记得在添加标签页面上指定一个标签。有关如何指定标签的详细信息，请参阅启动 Amazon EC2 实例（控制台）。

验证 CodeDeploy 代理是否正在实例上运行

- 按照验证 CodeDeploy 代理是否正在运行中的说明验证代理是否正在实例上运行。

在成功预置实例并确认 CodeDeploy 代理正在运行之后，转至下一步。

步骤 5：创建应用程序和部署组

在此步骤中，您将使用 CodeDeploy 控制台或 Amazon CLI 创建应用程序和部署组（用于部署来自 GitHub 存储库的示例修订）。

创建应用程序和部署组（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 选择创建应用程序，然后选择自定义应用程序。
4. 在 Application name（应用程序名称）中，输入 **CodeDeployGitHubDemo-App**。
5. 在 Compute Platform（计算平台）中，选择 EC2/On-premises（EC2/本地）。
6. 选择创建应用程序。
7. 在部署组选项卡中，选择创建部署组。
8. 在 Deployment group name（部署组名称）中，输入 **CodeDeployGitHubDemo-DepGrp**。
9. 在服务角色中，选择您在为 CodeDeploy 创建服务角色中创建的 CodeDeploy 服务角色的名称。
10. 在部署类型中，选择就地。
11. 在环境配置中，根据您使用的实例的类型，选择 Amazon EC2 实例或本地实例。对于键和值，根据步骤 4：预置实例的介绍输入应用于您的实例的实例标签键和值。
12. 在部署配置中，选择 CodeDeployDefault.AllatOnce。
13. 在负载均衡器中，清除启用负载均衡。
14. 展开高级。
15. 在警报中，选择忽略警报配置。
16. 选择创建部署组，然后继续下一步。

创建应用程序和部署组（CLI）

1. 调用 create-application 命令以在 CodeDeploy 中创建一个名为 `CodeDeployGitHubDemo-App` 的应用程序：

```
aws deploy create-application --application-name CodeDeployGitHubDemo-App
```

2. 调用 create-deployment-group 命令以创建一个名为 `CodeDeployGitHubDemo-DepGrp` 的部署组：

   - 如果您要部署到 Amazon EC2 实例，则 *ec2-tag-key* 是在步骤 4：预置实例中应用于您的 Amazon EC2 实例的 Amazon EC2 实例标签键。
   - 如果您要部署到 Amazon EC2 实例，则 *ec2-tag-value* 是在步骤 4：预置实例中应用于您的 Amazon EC2 实例的 Amazon EC2 实例标签值。
   - 如果您要部署到本地实例，则 *on-premises-tag-key* 是作为步骤 4：预置实例的一部分应用于本地实例的本地实例标签密钥。
   - 如果您要部署到本地实例，则 *on-premises-tag-value* 是作为步骤 4：预置实例的一部分应用于本地实例的本地实例标签值。
   - *service-role-arn* 是您在为 CodeDeploy 创建服务角色中创建的服务角色的 ARN。（按照获取服务角色 ARN（CLI）中的说明执行操作可查找服务角色 ARN。）

```
aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-App
 --ec2-tag-filters Key=ec2-tag-key,Type=KEY_AND_VALUE,Value=ec2-tag-value --on-
premises-tag-filters Key=on-premises-tag-key,Type=KEY_AND_VALUE,Value=on-premises-
tag-value --deployment-group-name CodeDeployGitHubDemo-DepGrp --service-role-
arn service-role-arn
```

> **Note**
>
> create-deployment-group 命令支持创建触发器，触发器会向主题订阅者发送有关部署和实例中指定事件的 Amazon SNS 通知。该命令还支持用于自动回滚部署和设置警报的选项，以便在达到 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中不包含用于这些操作的命令。

步骤 6：将应用程序部署到实例

在此步骤中，您使用 CodeDeploy 控制台或 Amazon CLI 将示例修订从 GitHub 存储库部署到实例。

部署修订（控制台）

1. 在部署组详细信息页上，选择创建部署。
2. 在部署组中，选择 **`CodeDeployGitHubDemo-DepGrp`**。
3. 在修订类型中，选择 GitHub。
4. 在 Connect to GitHub（连接到 GitHub）中，执行下列操作之一：
   - 要为 CodeDeploy 应用程序创建与 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。在 GitHub account（GitHub 账户）中，输入一个名称来标识此连接，然后选择 Connect to GitHub（连接到 GitHub）。该网页将提示您授权 CodeDeploy 与名为 `CodeDeployGitHubDemo-App` 的应用程序的 GitHub 进行交互。继续执行步骤 5。
   - 要使用已创建的连接，请在 GitHub account 中，选择其名称，然后选择 Connect to GitHub。继续执行步骤 7。
   - 要创建与其他 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account（连接到其他 GitHub 账户），然后选择 Connect to GitHub（连接到 GitHub）。继续执行步骤 5。
5. 按照 Sign in 页上的说明执行操作以使用您的 GitHub 账户进行登录。
6. 在 Authorize application 页上，选择 Authorize application。
7. 在 CodeDeploy 创建部署页面的存储库名称中，输入用于登录的 GitHub 用户名，依次后跟正斜杠（/）和已推送应用程序修订的存储库的名称（例如，***`my-github-user-name`***/**`CodeDeployGitHubDemo`**）。

   如果您不确定要输入的值，或者需要指定其他存储库，请执行以下步骤：

   a. 在单独的 Web 浏览器选项卡中，转到您的 GitHub 仪表板。

   b. 在 Your repositories（您的资料库）中，将鼠标指针悬停在目标存储库名称的上方。此时将显示工具提示，其中显示 GitHub 用户或组织名称，依次后跟正斜杠（/）和存储库的名称。将此值输入到存储库名称。

> **Note**
> 如果目标存储库名称未显示在您的存储库中，请使用搜索 GitHub 框查找目标存储库和 GitHub 用户或组织名。

8. 在提交 ID 框中，输入与将应用程序修订推送到 GitHub 这一操作关联的提交的 ID。

   如果您不确定要输入的值，请执行以下步骤：

   a. 在单独的 Web 浏览器选项卡中，转到您的 GitHub 仪表板。
   b. 在您的存储库中，选择 CodeDeployGitHubDemo
   c. 在提交列表中，查找并复制与将应用程序修订推送到 GitHub 这一操作关联的提交 ID。此 ID 的长度通常为 40 个字符并包含字母和数字。（请不要使用提交 ID 的较短版本，它通常是较长版本的前 10 个字符。）
   d. 将提交 ID 粘贴到 Commit ID 框中。
9. 选择 Deploy，然后继续执行下一步。

部署修订（CLI）

您必须先为 CodeDeploy 提供权限以使用您的 GitHub 用户账户与 `CodeDeployGitHubDemo-App` 应用程序的 GitHub 进行交互，然后才能调用任何与 GitHub 交互的 Amazon CLI 命令（例如您接下来将调用的 create-deployment 命令）。当前，您必须使用 CodeDeploy 控制台执行此操作。

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 选择 CodeDeployGitHubDemo-App。
4. 在部署选项卡上，选择创建部署。

> **Note**
>
> 您不会创建新的部署。这是当前授权 CodeDeploy 代表您的 GitHub 用户账户与 GitHub 交互的唯一方式。

5. 从部署组中选择 CodeDeployGitHubDemo-DepGrp。
6. 在修订类型中，选择 GitHub。
7. 在 Connect to GitHub（连接到 GitHub）中，执行下列操作之一：
   - 要为 CodeDeploy 应用程序创建与 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。在 GitHub account 中，键入一个名称来标识此连接，然后选择 Connect to GitHub。该网页将提示您授权 CodeDeploy 与名为 `CodeDeployGitHubDemo-App` 的应用程序的 GitHub 进行交互。继续执行步骤 8。
   - 要使用已创建的连接，请在 GitHub account 中，选择其名称，然后选择 Connect to GitHub。继续执行步骤 10。
   - 要创建与其他 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account（连接到其他 GitHub 账户），然后选择 Connect to GitHub（连接到 GitHub）。继续执行步骤 8。
8. 按照 Sign in 页上的说明执行操作以使用您的 GitHub 用户名或电子邮件和密码进行登录。
9. 在 Authorize application 页上，选择 Authorize application。
10. 在 CodeDeploy 创建部署页上，选择取消。
11. 调用 create-deployment 命令以将修订从您的 GitHub 存储库部署到实例，其中：
    - *`repository`* 是您的 GitHub 账户名，依次后跟正斜杠（`/`）和存储库名称（`CodeDeployGitHubDemo`），例如 `MyGitHubUserName/CodeDeployGitHubDemo`。

      如果您不确定要使用的值，或者需要指定其他存储库，请执行以下步骤：

      1. 在单独的 Web 浏览器选项卡中，转到您的 GitHub 仪表板。
      2. 在 Your repositories（您的资料库）中，将鼠标指针悬停在目标存储库名称的上方。此时将显示工具提示，其中显示 GitHub 用户或组织名称，依次后跟正斜杠（`/`）和存储库的名称。这是要使用的值。

> **Note**
>
> 如果目标存储库名称未显示在 Your repositories 中，请使用 Search GitHub 框查找目标存储库和对应的 GitHub 用户或组织名。

- *commit-id* 是与您已推送到存储库的应用程序修订的版本关联的提交（例如，`f835159a...528eb76f`）。

  如果您不确定要使用的值，请执行以下步骤：

  1. 在单独的 Web 浏览器选项卡中，转到您的 GitHub 仪表板。
  2. 在您的存储库中，选择 CodeDeployGitHubDemo
  3. 在提交列表中，查找与将应用程序修订推送到 GitHub 这一操作关联的提交 ID。此 ID 的长度通常为 40 个字符并包含字母和数字。（请不要使用提交 ID 的较短版本，它通常是较长版本的前 10 个字符。）请使用此值。

如果您使用的是本地 Linux、macOS 或 Unix 机器：

```
aws deploy create-deployment \
  --application-name CodeDeployGitHubDemo-App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name CodeDeployGitHubDemo-DepGrp \
  --description "My GitHub deployment demo" \
  --github-location repository=repository,commitId=commit-id
```

如果您正在本地 Windows 计算机上工作：

```
aws deploy create-deployment --application-name CodeDeployGitHubDemo-App --
deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name
 CodeDeployGitHubDemo-DepGrp --description "My GitHub deployment demo" --github-
location repository=repository,commitId=commit-id
```

步骤 7：监控和验证部署

在此步骤中，您将使用 CodeDeploy 控制台或 Amazon CLI 验证部署是否成功。您将使用 Web 浏览器来查看已部署到您已创建或配置的实例的网页。

> **Note**
>
> 如果您要部署到 Ubuntu Server 实例，请使用您自己的测试策略来确定已部署的修订是否在实例上按预期运行，然后转至下一步。

监控和验证部署（控制台）

1. 在导航窗格中，展开部署，然后选择部署。
2. 在部署列表中，查找应用程序值为 CodeDeployGitHubDemo-App 且部署组值为 CodeDeployGitHubDemo-DepGrp 的行。如果状态列中未显示成功或失败，请定期选择刷新按钮。
3. 如果状态列中出现失败，请按照查看实例详细信息（控制台）中的说明对部署进行故障排除。
4. 如果状态列中出现成功，则您现在可以通过 Web 浏览器验证部署。我们的示例修订将单个网页部署到实例。如果您要部署到 Amazon EC2 实例，请在您的 Web 浏览器中，转至该实例的 `http://public-dns`（例如，`http://ec2-01-234-567-890.compute-1.amazonaws.com`）。
5. 如果您能看到此网页，那么恭喜您！现在您已成功使用 Amazon CodeDeploy 从 GitHub 部署修订，可向前跳至第 8 步：清除。

监控和验证部署（CLI）

1. 调用 list-deployments 命令以获取名为 `CodeDeployGitHubDemo-App` 的应用程序和名为 `CodeDeployGitHubDemo-DepGrp` 的部署组的部署 ID：

```
aws deploy list-deployments --application-name CodeDeployGitHubDemo-App --
deployment-group-name CodeDeployGitHubDemo-DepGrp --query "deployments" --output
 text
```

2. 调用 get-deployment 命令，并在 list-deployments 命令的输出中提供部署 ID：

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text
```

3. 如果返回 Failed，请按照查看实例详细信息（控制台）中的说明执行操作以排查部署的问题。
4. 如果返回 Succeeded，则可立即尝试通过 Web 浏览器验证部署。我们的示例修订是已部署到实例的单个网页。如果您要部署到 Amazon EC2 实例，可通

过转至面向 Amazon EC2 实例的 `http://public-dns`（例如，`http://ec2-01-234-567-890.compute-1.amazonaws.com`）在 Web 浏览器中查看此页。

5. 如果您能看到此网页，那么恭喜您！您已成功使用 Amazon CodeDeploy 从您的 GitHub 存储库部署。

第 8 步：清除

为了避免您在本教程中使用的资源产生其他费用，您必须终止 Amazon EC2 实例及其关联资源。（可选）您可以删除与本教程关联的 CodeDeploy 部署组件记录。如果您在本教程中仅使用 GitHub 存储库，也可立即删除它。

删除 Amazon CloudFormation 堆栈（如果您使用 Amazon CloudFormation 模板创建了 Amazon EC2 实例）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。
2. 在 Stacks（堆栈）列中，选择以 `CodeDeploySampleStack` 开头的堆栈。
3. 选择 Delete（删除）。
4. 当系统提示时，选择 Delete stack（删除堆栈）。这将删除 Amazon EC2 实例以及关联的 IAM 实例配置文件和服务角色。

手动取消注册并清除本地实例（如果您已预置本地实例）

1. 使用 Amazon CLI 针对此处由 *your-instance-name* 表示的本地实例和由 *your-region* 表示的关联区域，调用 deregister 命令：

```
aws deploy deregister --instance-name your-instance-name --no-delete-iam-user --
region your-region
```

2. 从本地实例调用 uninstall 命令：

```
aws deploy uninstall
```

手动终止 Amazon EC2 实例（如果您手动启动了 Amazon EC2 实例）

1. 登录 Amazon Web Services 管理控制台，打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的 Instances 下，选择 Instances。
3. 选中要终止的 Amazon EC2 实例旁边的框。在 Actions 菜单中，指向 Instance State，然后选择 Terminate。
4. 在系统提示时，选择 Yes, Terminate。

删除 CodeDeploy 部署组件记录

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 选择 CodeDeployGitHubDemo-App。
4. 选择删除应用程序。
5. 当系统提示时，输入 **`Delete`**，然后选择删除。

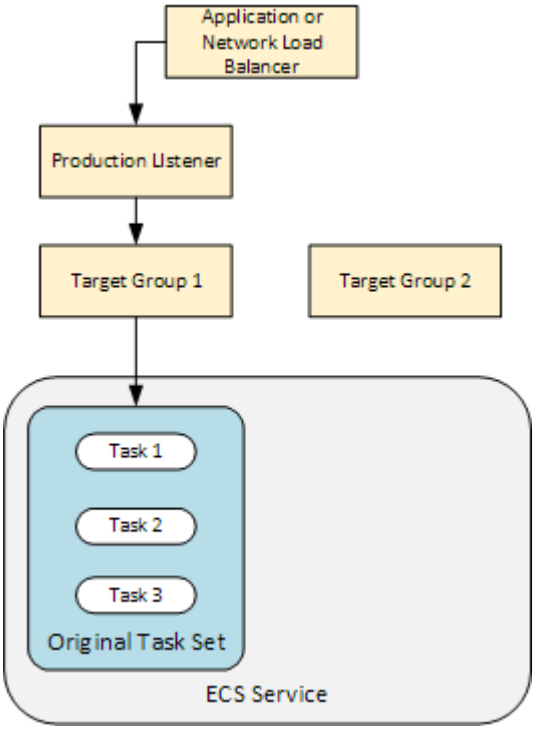
删除 GitHub 存储库

请参阅 GitHub 帮助中的删除存储库。

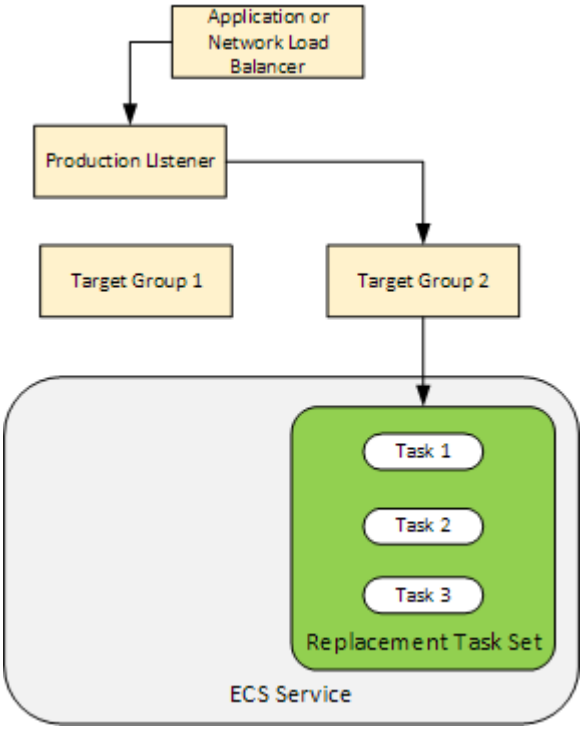
教程：将应用程序部署到 Amazon ECS

在本教程中，您将学习如何使用将应用程序部署到 Amazon ECS 中 CodeDeploy。您从已经创建并部署到 Amazon ECS 中的应用程序开始。第一步是采用新的标签修改应用程序的任务定义文件，以更新应用程序。接下来，您将使用 CodeDeploy 来部署更新。在部署期间，将更新 CodeDeploy 安装到新的替换任务集中。然后，将原始任务集中 Amazon ECS 应用程序原始版本的生产流量，转移到替换任务集中更新的版本。

在 Amazon ECS 部署期间， CodeDeploy 使用配置有两个目标组和一个生产流量侦听器的负载均衡器。下图显示了部署开始之前，负载均衡器、生产侦听器、目标组以及 Amazon ECS 应用程序之间的关联方式。本教程使用一个 Application Load Balancer。您也可以使用网络负载均衡器。



成功部署之后，生产流量侦听器将流量提供给新的替换任务集，原始任务集终止。下图显示了成功部署后资源之间的关联方式。有关更多信息，请参阅 在 Amazon ECS 部署过程中发生的事件。



有关如何使用将应用程序部署 Amazon CLI 到 Amazon ECS 的信息，请参阅教程：使用 blue/green 部署创建服务。有关 CodePipeline 如何使用检测和自动部署对 Amazon ECS 服务的更改的信息 CodeDeploy，请参阅教程：使用 Amazon ECR 源和 ECS-to-CodeDeploy部署创建管道。

完成本教程后，您可以使用您创建的 CodeDeploy 应用程序和部署组在中添加部署验证测试教程：部署具有验证测试的 Amazon ECS 服务。

主题

- 先决条件
- 步骤 1：更新 Amazon ECS 应用程序
- 步骤 2：创建 AppSpec 文件
- 步骤 3：使用 CodeDeploy 控制台部署应用程序
- 步骤 4：清除

先决条件

要完成本教程，您首先必须：

- 完成 入门 CodeDeploy 中的步骤 2 和步骤 3。

- 创建配置有两个目标组和一个侦听器的应用程序负载均衡器。有关使用控制台创建负载均衡器的信息，请参阅 为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器。有关使用创建负载均衡器的信息 Amazon CLI，请参阅《亚马逊弹性容器服务用户指南》中的步骤 1：创建应用程序负载均衡器。在创建负载均衡器时，记录以下内容以用于本教程：
  - 负载均衡器的名称。
  - 目标组的名称。
  - 负载均衡器侦听器所用的端口。
- 创建 Amazon ECS 集群和服务 有关更多信息，请参阅《亚马逊弹性容器服务用户指南》中的教程：使用 blue/green 部署创建服务中的步骤 2、3 和 4。记录以下内容以用于本教程：
  - Amazon ECS 集群的名称。
  - Amazon ECS 服务所用的任务定义的 ARN。
  - Amazon ECS 服务所用的容器的名称。
- 为您的 AppSpec 文件创建一个 Amazon S3 存储桶。

步骤 1：更新 Amazon ECS 应用程序

在本部分中，您将采用新的任务定义修订更新 Amazon ECS 应用程序。更新后的修订添加了新的标签键对。在 步骤 3：使用 CodeDeploy 控制台部署应用程序中，您将部署 Amazon ECS 应用程序的更新版本。

更新任务定义

1. 在 https://console.aws.amazon.com/ecs/v2 中打开控制台。
2. 在导航窗格中，选择 Task Definitions。
3. 选择 Amazon ECS 服务使用的任务定义。
4. 选择任务定义修订版，然后选择创建新的修订、创建新的修订。
5. 在本教程中，通过添加标签对任务定义进行细微更新。在页面底部的标签中，输入新的键值对以创建新的标签。
6. 选择创建。

   任务定义的修订号增加 1。
7. 选择 JSON 选项卡。记录以下内容，因为下一步需要这些信息。

- `taskDefinitionArn` 的值。其格式为 `arn:aws:ecs:aws-region:account-id:task-definition/task-definition-family:task-definition-revision`。这是已更新的任务定义的 ARN。
- `containerDefinitions` 元素中 `name` 的值。这是容器的名称。
- `portMappings` 元素中 `containerPort` 的值。这是容器的端口。

步骤 2：创建 AppSpec 文件

在本节中，您将创建 AppSpec 文件并将其上传到您在本先决条件节中创建的 Amazon S3 存储桶。Amazon ECS 部署 AppSpec 的文件指定了您的任务定义、容器名称和容器端口。有关更多信息，请参阅 AppSpec Amazon ECS 部署的文件示例 和 AppSpec Amazon ECS 部署的 “资源” 部分。

创建您的 AppSpec 文件

1. 如果要使用 YAML 创建 AppSpec 文件，请创建一个名为`appspec.yml`的文件。如果要使用 JSON 创建 AppSpec 文件，请创建一个名为的文件`appspec.json`。
2. 根据您的 AppSpec 文件使用的是 YAML 还是 JSON，选择相应的选项卡，然后将其内容复制到刚刚创建 AppSpec 的文件中。对于 `TaskDefinition` 属性，请使用您在 步骤 1：更新 Amazon ECS 应用程序 部分中记下的任务定义 ARN。

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "your-container-name",
            "ContainerPort": your-container-port
          }
        }
      }
    }
  ]
```

```
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "your-container-name"
          ContainerPort: your-container-port
```

> **Note**
>
> 替换任务集继承了原始任务集的子网、安全组、平台版本以及分配的公有 IP 值。您可以通过在 AppSpec 文件中设置替换任务集的可选属性来覆盖这些值。有关更多信息，请参阅 AppSpec Amazon ECS 部署的 “资源” 部分和 AppSpec Amazon ECS 部署的文件示例 。

3. 将您的 AppSpec 文件上传到您创建的 S3 存储桶，这是本教程的先决条件。

步骤 3：使用 CodeDeploy 控制台部署应用程序

在本节中，您将创建一个 CodeDeploy 应用程序和部署组，以便将更新的应用程序部署到 Amazon ECS 中。在部署期间，将应用程序的生产流量 CodeDeploy 转移到新的替换任务集中的新版本。要完成此步骤，您需要以下各项：

- Amazon ECS 集群名称。
- Amazon ECS 服务名称。
- 应用程序负载均衡器名称。
- 生产侦听器端口。
- 目标组名称。
- 您创建的 S3 存储桶的名称。

创建 CodeDeploy 应用程序

1. 登录 Amazon Web Services 管理控制台 并打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。
2. 选择创建应用程序。
3. 在 Application name（应用程序名称）中，输入 **ecs-demo-codedeploy-app**。
4. 在 Compute platform（计算平台）中，选择 Amazon ECS。
5. 选择创建应用程序。

创建 CodeDeploy 部署组

1. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
2. 在 Deployment group name（部署组名称）中，输入 **ecs-demo-dg**。
3. 在服务角色中，选择一个授予 CodeDeploy 对 Amazon ECS 访问权限的服务角色。有关更多信息，请参阅 对 Amazon CodeDeploy进行身份和访问管理。
4. 在环境配置中，选择 Amazon ECS 集群名称和服务名称。
5. 从负载均衡器中，选择将流量提供给 Amazon ECS 服务的负载均衡器的名称。
6. 在生产侦听器端口中，选择将生产流量提供给 Amazon ECS 服务的侦听器的端口和协议（例如，HTTP: 80）。本教程不包括可选的测试侦听器，因此请勿从 Test listener port（测试侦听器端口）中选择端口。
7. 从 Target group 1 name（目标组 1 名称）和 Target group 2 name（目标组 2 名称）中，选择两个不同的目标组以在部署期间路由流量。请确保它们是您为负载均衡器创建的目标组。哪个用于目标组 1 和哪个用于目标组 2 并不重要。
8. 选择 Reroute traffic immediately（立即重新路由流量）。
9. 对于 Original revision termination（原始修订终止），选择 0 天、0 小时和 5 分钟。与使用默认值（1 小时）相比，这可以让您更快地完成部署。

Environment configuration

Choose an ECS cluster name

ecs-tutorial-cluster

Choose an ECS service name

ecs-demo-service

Load balancers

Choose a load balancer

ecs-demo-alb

Production listener port

HTTP: 80

Test listener port - *optional*

A test listener is required if you want to test your replacement version before traffic reroutes to it

Target group 1 name

ecs-demo-tg-1

Target group 2 name

ecs-demo-tg-2

Deployment settings

Traffic rerouting

Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process

- (●) Reroute traffic immediately
- ( ) Specify when to reroute traffic

Deployment Configuration

CodeDeployDefault.ECSALLAtOnce

Original revision termination

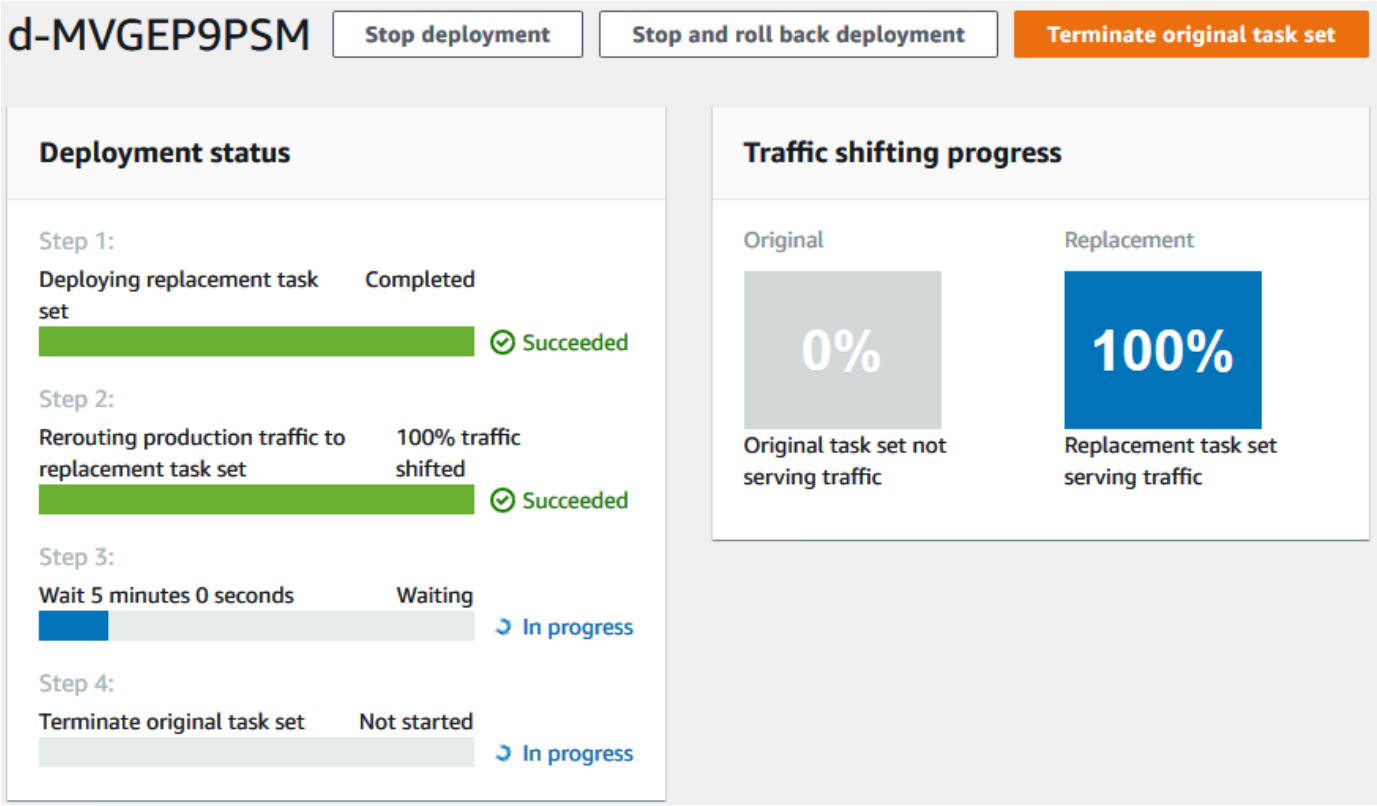
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically

Days	Hours	Minutes
0	0	5

10. 选择 Create deployment（创建部署组）。

部署您的 Amazon ECS 应用程序

1. 从部署组控制台页面中，选择 Create deployment（创建部署）。
2. 对于部署组，选择ecs-demo-dg。
3. 对于 Revision type（修订类型），选择 My application is stored in Amazon S3（我的应用程序存储在 Amazon S3 中）。在 Revision location（修订位置）中，输入 S3 存储桶的名称。
4. 对于 Revision file type（修订文件类型），根据情况选择 .json 或 .yaml。
5. （可选）在 Deployment description（部署描述）框中，为部署输入描述。
6. 选择 Create deployment（创建部署）。
7. 您可以在 Deployment status（部署状态）中监控部署。在生产流量已全部路由至替换任务集并等待五分钟之后，您可以选择终止原始任务集，以立即终止原始任务集。如果未选择 Terminate original task set（终止原始任务集），则原始任务集将在您指定的五分钟等待时间到期后终止。



步骤 4：清除

下一个教程 教程：部署具有验证测试的 Amazon ECS 服务 建立在本教程的基础上，使用的是您创建的 CodeDeploy 应用程序和部署组。如果您希望遵循该教程中的步骤，请跳过此步骤，不要删除您创建的资源。

> **Note**
>
> 您的 Amazon 账户不会因您创建的 CodeDeploy 资源而产生费用。

这些步骤中的资源名称是本教程中建议的名称（**ecs-demo-codedeploy-app**例如， CodeDeploy 应用程序的名称）。如果您使用的是不同的名称，请确保在清除过程中使用这些名称。

1. 使用delete-deployment-group命令删除 CodeDeploy 部署组。

```
aws deploy delete-deployment-group --application-name ecs-demo-codedeploy-app --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. 使用 delete-application 命令删除应用程序。 CodeDeploy

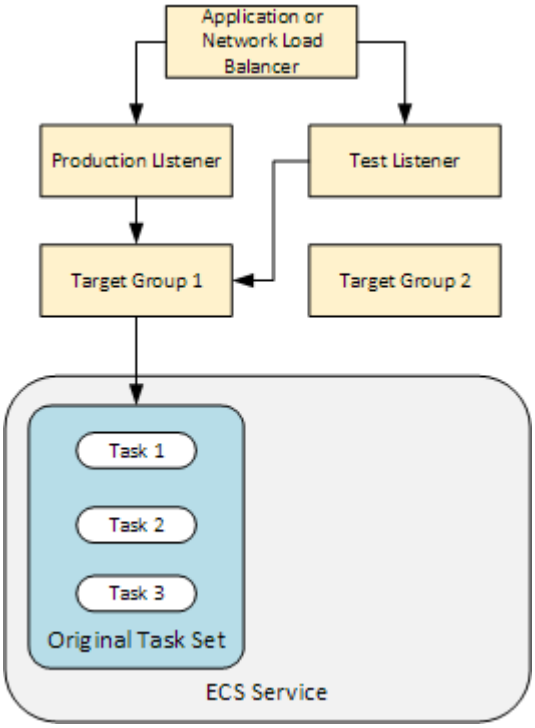
```
aws deploy delete-application --application-name ecs-demo-codedeploy-app --
region aws-region-id
```

教程：部署具有验证测试的 Amazon ECS 服务

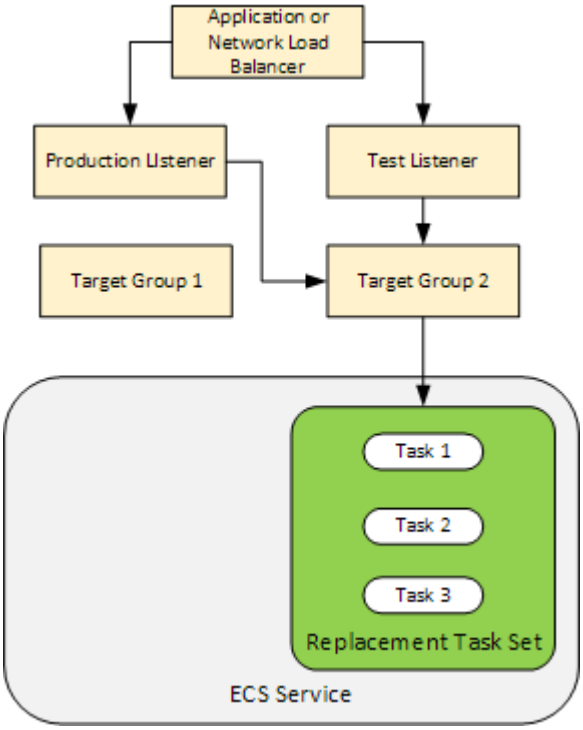
在本教程中，您将学习如何使用 Lambda 函数验证已更新的 Amazon ECS 应用程序的部分部署。本教程使用您在中使用的 CodeDeploy 应用程序、 CodeDeploy 部署组和 Amazon ECS 应用程序教程：将应用程序部署到 Amazon ECS。请先完成上述教程，然后再开始本教程。

要添加验证测试，首先应在 Lambda 函数中实施测试。接下来，在部署 AppSpec 文件中，为要测试的生命周期挂钩指定 Lambda 函数。如果验证测试失败，部署将停止，然后回滚，并标记为失败。如果测试成功，部署将继续下一个部署生命周期事件或挂钩。

在部署带有验证测试的 Amazon ECS 期间， CodeDeploy 使用配置有两个目标组的负载均衡器：一个生产流量侦听器和一个测试流量侦听器。下图显示了部署开始之前，负载均衡器、生产和测试侦听器、目标组以及 Amazon ECS 应用程序之间的关联方式。本教程使用一个 Application Load Balancer。您也可以使用网络负载均衡器。



在 Amazon ECS 部署过程中，有五个用于测试的生命周期挂钩。本教程在第三个生命周期部署挂钩（`AfterAllowTestTraffic`）期间实施了一次测试。有关更多信息，请参阅 用于 Amazon ECS 部署的生命周期事件挂钩的列表。成功部署之后，生产流量侦听器将流量提供给新的替换任务集，原始任务集终止。下图显示了成功部署后资源之间的关联方式。有关更多信息，请参阅 在 Amazon ECS 部署过程中发生的事件。



> **Note**
>
> 完成本教程可能会导致您的 Amazon 账户被扣款。这些费用包括 CodeDeploy Amazon Lambda、和可能收取的费用 CloudWatch。有关更多信息，请参阅Amazon CodeDeploy 定价、Amazon Lambda 定价和 Amazon CloudWatch 定价。

主题

- 先决条件
- 步骤 1：创建测试侦听器
- 步骤 2：更新 Amazon ECS 应用程序
- 步骤 3：创建生命周期挂钩 Lambda 函数
- 第 4 步：更新您的 AppSpec文件
- 步骤 5：使用 CodeDeploy 控制台部署您的 Amazon ECS 服务
- 步骤 6：在日志中查看您的 Lambda 挂钩函数输出 CloudWatch
- 步骤 7：清除

先决条件

要成功完成本教程，您首先必须：

- 满足先决条件中针对教程：将应用程序部署到 Amazon ECS 的先决条件。
- 完成 教程：将应用程序部署到 Amazon ECS 中的步骤。记录以下内容：
  - 负载均衡器的名称。
  - 目标组的名称。
  - 负载均衡器侦听器所用的端口。
  - 负载均衡器的 ARN。您可以使用此项创建新的侦听器。
  - 其中一个目标组的 ARN。您可以使用此项创建新的侦听器。
  - 您创建的 CodeDeploy 应用程序和部署组。
  - 您创建的、供 CodeDeploy 部署使用的 AppSpec 文件。您可以在本教程中编辑此文件。

步骤 1：创建测试侦听器

具有验证测试的 Amazon ECS 部署需要第二个侦听器。此侦听器用于为替换任务集中更新的 Amazon ECS 应用程序提供测试流量。验证测试针对测试流量运行。

测试流量侦听器可以使用任一目标组。使用 create-listener Amazon CLI 命令创建第二个监听器，其默认规则将测试流量转发到端口 8080。使用负载均衡器的 ARN 和其中一个目标组的 ARN。

```
aws elbv2 create-listener --load-balancer-arn your-load-balancer-arn \
--protocol HTTP --port 8080 \
--default-actions Type=forward,TargetGroupArn=your-target-group-arn --region your-aws-region
```

步骤 2：更新 Amazon ECS 应用程序

在本部分中，您将更新 Amazon ECS 应用程序以使用其任务定义的新修订。您可以创建新的修订，并通过添加标签向其添加次要更新。

更新任务定义

1. 打开 Amazon ECS 经典控制台，网址为https://console.aws.amazon.com/ecs/。
2. 在导航窗格中，选择 Task Definitions。
3. 选中 Amazon ECS 服务所使用的任务定义对应的复选框。
4. 选择 Create new revision（创建新修订）。
5. 通过添加标签对任务定义进行细微更新。在页面底部的 Tags（标签）中，输入新的键值对以创建新的标签。
6. 选择创建。您应当看到任务定义的修订号增加了 1。
7. 选择 JSON 选项卡。记下 `taskDefinitionArn` 的值。其格式为 `arn:aws:ecs:`*`aws-region`*`:` *`account-id`*`:task-definition/`*`task-definition-family`*`:` *`task-definition-revision`*。这是已更新的任务定义的 ARN。

步骤 3：创建生命周期挂钩 Lambda 函数

在本部分中，您将为 Amazon ECS 部署的 `AfterAllowTestTraffic` 挂钩实施一个 Lambda 函数。在安装更新的 Amazon ECS 应用程序之前，Lambda 函数将运行验证测试。对于本教程，Lambda 函数返回 `Succeeded`。在实际部署过程中，验证测试可能返回 `Succeeded` 或

`Failed`，具体取决于验证测试的结果。此外，您可能会对其他 Amazon ECS 部署生命周期事件挂钩（`BeforeInstall`、`AfterInstall`、`BeforeAllowTraffic` 和 `AfterAllowTraffic`）中的一个或多个实施 Lambda 测试函数。有关更多信息，请参阅 用于 Amazon ECS 部署的生命周期事件挂钩的列表。

必须具有 IAM 角色才能创建 Lambda 函数。该角色向 Lambda 函数授予写入 CloudWatch 日志和设置 CodeDeploy 生命周期挂钩状态的权限。

创建 IAM 角色

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 从导航窗格中选择角色，然后选择创建角色。
3. 创建具有以下属性的角色：
   - Trusted entity（可信任的实体）：Amazon Lambda。
   - 权限：AWSLambdaBasicExecutionRole。这会授予您的 Lambda 函数写入日志的权限。CloudWatch
   - Role name（角色名称）：**`lambda-cli-hook-role`**。

   有关更多信息，请参阅创建 Amazon Lambda 执行角色。
4. 将权限 `codedeploy:PutLifecycleEventHookExecutionStatus` 附加到您创建的角色。这会授予您的 Lambda 函数在部署期间设置 CodeDeploy 生命周期挂钩状态的权限。有关更多信息，请参阅Amazon Identity and Access Management 用户指南和 PutLifecycleEventHookExecutionStatusCodeDeploy API 参考中的添加 IAM 身份权限。

创建 **`AfterAllowTestTraffic`** 挂钩 Lambda 函数

1. 使用以下内容创建名为 `AfterAllowTestTraffic.js` 的文件。

```
'use strict';

 const AWS = require('aws-sdk');
 const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});

 exports.handler = (event, context, callback) => {

  console.log("Entering AfterAllowTestTraffic hook.");
```

```
  // Read the DeploymentId and LifecycleEventHookExecutionId from the event payload
  var deploymentId = event.DeploymentId;
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
  var validationTestResult = "Failed";

  // Perform AfterAllowTestTraffic validation tests here. Set the test result
  // to "Succeeded" for this tutorial.
  console.log("This is where AfterAllowTestTraffic validation tests happen.")
  validationTestResult = "Succeeded";

  // Complete the AfterAllowTestTraffic hook by sending CodeDeploy the validation
 status
  var params = {
   deploymentId: deploymentId,
   lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: validationTestResult // status can be 'Succeeded' or 'Failed'
  };

  // Pass CodeDeploy the prepared validation test results.
  codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
   if (err) {
    // Validation failed.
    console.log('AfterAllowTestTraffic validation tests failed');
    console.log(err, err.stack);
    callback("CodeDeploy Status update failed");
   } else {
    // Validation succeeded.
    console.log("AfterAllowTestTraffic validation tests succeeded");
    callback(null, "AfterAllowTestTraffic validation tests succeeded");
   }
  });
 }
```

2. 创建 Lambda 部署包。

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

3. 使用 `create-function` 命令为 `AfterAllowTestTraffic` 挂钩创建 Lambda 函数。

```
aws lambda create-function --function-name AfterAllowTestTraffic \
       --zip-file fileb://AfterAllowTestTraffic.zip \
       --handler AfterAllowTestTraffic.handler \
```

```
    --runtime nodejs10.x \
    --role arn:aws:iam::aws-account-id:role/lambda-cli-hook-role
```

4. 记下 `create-function` 响应中的 Lambda 函数 ARN。在下一步中更新 CodeDeploy 部署 AppSpec 文件时，您将使用此 ARN。

第 4 步：更新您的 AppSpec文件

在本节中，您将使用一个`Hooks`部分来更新您的 AppSpec 文件。在 `Hooks` 部分中，您将为 `AfterAllowTestTraffic` 生命周期挂钩指定 Lambda 函数。

更新您的 AppSpec 文件

1. 打开您在中 步骤 2：创建 AppSpec 文件 创建 AppSpec 的文件文件教程：将应用程序部署到 Amazon ECS。
2. 采用您在 步骤 2：更新 Amazon ECS 应用程序 中记下的任务定义 ARN 更新 `TaskDefinition` 属性。
3. 将该`Hooks`部分复制并粘贴到您的 AppSpec 文件文件中。采用您在步骤 3：创建生命周期挂钩 Lambda 函数中记下的 Lambda 函数的 ARN，在 `AfterAllowTestTraffic` 之后更新 ARN。

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "sample-website",
            "ContainerPort": 80
          }
        }
      }
    }
  ],
  "Hooks": [
```

```
    {
      "AfterAllowTestTraffic": "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
    }
  ]
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
Hooks:
  - AfterAllowTestTraffic: "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
```

4. 保存您的 AppSpec 文件并上传到其 S3 存储桶。

步骤 5：使用 CodeDeploy 控制台部署您的 Amazon ECS 服务

在本部分中，您将通过为测试侦听器指定端口更新部署组。这是您在 步骤 1：创建测试侦听器 中创建的侦听器。在部署过程中， CodeDeploy 在 `AfterAllowTestTraffic` 部署生命周期挂钩期间，使用通过测试侦听器提供给替换任务集的测试流量运行验证测试。您的验证测试返回 `Succeeded` 结果，因此，部署将继续下一个部署生命周期事件。在实际场景中，测试函数可能返回 `Succeeded` 或 `Failed`。

向部署组添加测试侦听器

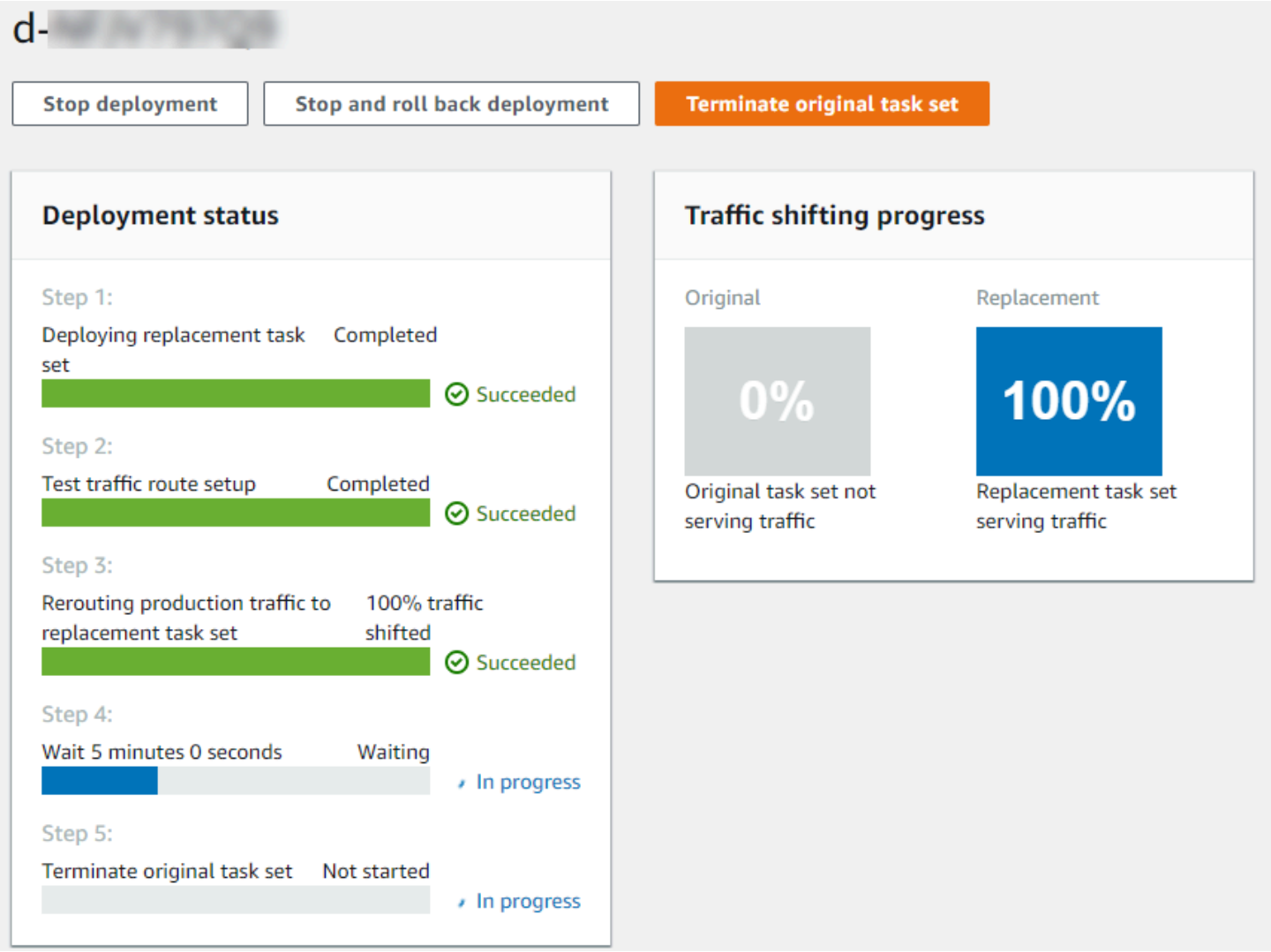
1. 登录 Amazon Web Services 管理控制台 并打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。
2. 从导航窗格中，选择 Applications（应用程序）。
3. 选择您在 教程：将应用程序部署到 Amazon ECS 中创建的应用程序。如果您使用建议的名称，则该名称是 ecs-demo-codedeploy-app。

4. 在 Deployment group（部署组）中，选择您在 教程：将应用程序部署到 Amazon ECS 中创建的部署组。如果您使用建议的名称，则该名称是 ecs-demo-dg。
5. 选择编辑。
6. 从 Test listener port（测试侦听器端口）中，为您之前在本教程中创建的测试侦听器选择端口和协议。应当为 HTTP: 8080。
7. 选择保存更改。

部署您的 Amazon ECS 应用程序

1. 从部署组控制台页面中，选择 Create deployment（创建部署）。
2. 对于部署组，选择ecs-demo-dg。
3. 对于 Revision type（修订类型），选择 My application is stored in Amazon S3（我的应用程序存储在 Amazon S3 中）。在修订位置中，输入您的 S3 存储桶和 AppSpec 文件的名称（例如，**`s3://my-s3-bucket/appspec.json`**）。
4. 对于 Revision file type（修订文件类型），根据情况选择 .json 或 .yaml。
5. （可选）在 Deployment description（部署描述）框中，为部署输入描述。
6. 选择 Create deployment（创建部署）。

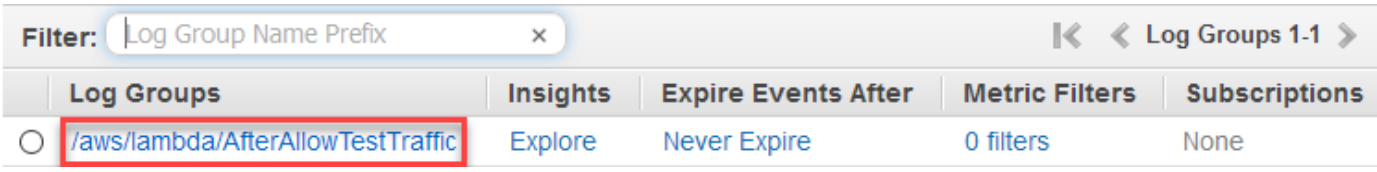
您可以在 Deployment status（部署状态）中监控部署。在生产流量已全部路由至替换任务集之后，您可以选择终止原始任务集，以立即终止原始任务集。如果未选择 Terminate original task set（终止原始任务集），则原始任务集将在您创建部署组时指定的持续时间之后终止。



步骤 6：在日志中查看您的 Lambda 挂钩函数输出 CloudWatch

如果您的 CodeDeploy 部署成功，那么您的 Lambda 挂钩函数中的验证测试也会成功。您可以通过在 Log CloudWatch s 中查看挂钩函数的日志来确认这一点。

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 从导航窗格中，选择 Logs（日志）。您应该看到一个与您在文件中指定的 Lambda 挂钩函数对应的新日志组。 AppSpec



3. 选择新的日志组。这应该是/aws/lambda/AfterAllowTestTrafficHook。

4. 选择日志流。如果您看到多个日志流，请在 Last Event Time（上次事件时间）下选择日期和时间最近的一个日志流。
5. 展开日志流事件，确认 Lambda 挂钩函数已成功将消息写入日志。下面显示了 `AfterAllowTraffic` Lambda 挂钩函数成功。

步骤 7：清除

完成本教程后，请清除与本教程关联的资源，以避免对您未使用的资源产生费用。此步骤中的资源名称是本教程中建议的名称（**`ecs-demo-codedeploy-app`**例如， CodeDeploy 应用程序的名称）。如果您使用的是不同的名称，请确保在清除过程中使用这些名称。

清除教程资源

1. 使用delete-deployment-group命令删除 CodeDeploy 部署组。

```
aws deploy delete-deployment-group --application-name ecs-demo-deployment-group --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. 使用 delete-application 命令删除应用程序。 CodeDeploy

```
aws deploy delete-application --application-name ecs-demo-deployment-group --
region aws-region-id
```

3. 使用 delete-function 命令删除 Lambda 挂钩函数。

```
aws lambda delete-function --function-name AfterAllowTestTraffic
```

4. 使用delete-log-group命令删除您的 CloudWatch 日志组。

```
aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic
```

教程：使用无 Amazon 服务器应用程序模型部署更新的 Lambda 函数 CodeDeploy

Amazon SAM 是一个用于构建无服务器应用程序的开源框架。它将 Amazon SAM 模板中的 YAML 语法转换并扩展为用于构建无服务器应用程序（例如 Lambda 函数）的 Amazon CloudFormation 语法。有关更多信息，请参阅什么是 Amazon Serverless Application Model？

在本教程中，您将使用 Amazon SAM 创建可执行以下操作的解决方案：

- 创建 Lambda 函数。
- 创建您的 CodeDeploy 应用程序和部署组。
- 创建两个 Lambda 函数，用于在 CodeDeploy生命周期挂钩期间执行部署验证测试。
- 检测 Lambda 函数的更新时间。Lambda 函数的更新会触发部署 CodeDeploy ，从而逐步将生产流量从您的 Lambda 函数的原始版本转移到更新的版本。

> **Note**
>
> 本教程要求您创建的资源可能会导致您的 Amazon 账户产生相关费用。这些费用包括 CodeDeploy、Amazon 和（亚马逊 CloudWatch）可能收取的费用 Amazon Lambda。有关更多信息，请参阅CodeDeploy 定价、Amazon CloudWatch 定价和Amazon Lambda 定价。

主题

- 先决条件
- 步骤 1：设置基础设施
- 步骤 2：更新 Lambda 函数
- 步骤 3：部署更新的 Lambda 函数
- 步骤 4：查看部署结果
- 第 5 步：清理

先决条件

要完成本教程，您首先必须：

- 完成 入门 CodeDeploy 中的步骤。
- 安装 C Amazon Serverless Application Model LI。有关信息，请参阅安装 Amazon SAM CLI。
- 创建 S3 存储桶。 Amazon SAM 会将 SA Amazon M 模板中引用的项目上传到此存储桶中。

步骤 1：设置基础设施

本主题向您展示 Amazon SAM 如何使用为您的 Amazon SAM 模板和 Lambda 函数创建文件。然后，使用 Amazon SAM `package`和`deploy`命令在基础架构中生成组件。基础设施准备就绪后，您将拥有一个 CodeDeploy 应用程序和部署组、一个要更新和部署的 Lambda 函数，以及两个 Lambda 函数，其中包含在您部署 Lambda 函数时运行的验证测试。完成后，您可以使用 Amazon CloudFormation 在 Lambda 控制台中查看您的组件，或者使用 Amazon CLI 来测试您的 Lambda 函数。

主题

- 创建文件
- Package 打包 Amazon SAM 应用程序
- 部署 S Amazon AM 应用程序
- （可选）检查并测试基础设施

创建文件

要创建基础设施，必须创建以下文件：

- `template.yml`
- `myDateTimeFunction.js`
- `beforeAllowTraffic.js`
- `afterAllowTraffic.js`

主题

- 创建你的 Amazon SAM 模板
- 为 Lambda 函数创建文件
- 为您的 BeforeAllowTraffic Lambda 函数创建一个文件

- 为您的 AfterAllowTraffic Lambda 函数创建一个文件

创建你的 Amazon SAM 模板

创建一个 Amazon SAM 模板文件来指定基础架构中的组件。

创建 Amazon SAM 模板

1. 创建一个名为 `SAM-Tutorial` 的目录。
2. 在 `SAM-Tutorial` 目录中创建名为 `template.yml` 的文件。
3. 将以下 YAML 代码复制到 `template.yml` 中。这是 Amazon SAM 模板。

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: A sample SAM template for deploying Lambda functions.

Resources:
# Details about the myDateTimeFunction Lambda function
  myDateTimeFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: myDateTimeFunction.handler
      Runtime: nodejs18.x
# Instructs your myDateTimeFunction is published to an alias named "live".
      AutoPublishAlias: live
# Grants this function permission to call lambda:InvokeFunction
      Policies:
        - Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: '*'
      DeploymentPreference:
# Specifies the deployment configuration
          Type: Linear10PercentEvery1Minute
# Specifies Lambda functions for deployment lifecycle hooks
          Hooks:
            PreTraffic: !Ref beforeAllowTraffic
            PostTraffic: !Ref afterAllowTraffic

# Specifies the BeforeAllowTraffic lifecycle hook Lambda function
```

```
  beforeAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: beforeAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          Statement:
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
# Grants this function permission to call lambda:InvokeFunction
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_beforeAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version

# Specifies the AfterAllowTraffic lifecycle hook Lambda function
  afterAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: afterAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          - Effect: "Allow"
            Action:
```

```
                - "codedeploy:PutLifecycleEventHookExecutionStatus"
              Resource:
                !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
          - Version: "2012-10-17"
            Statement:
# Grants this function permission to call lambda:InvokeFunction
            - Effect: "Allow"
              Action:
                - "lambda:InvokeFunction"
              Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_afterAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

此模板指定以下内容。有关更多信息，请参阅 Amazon SAM 模板概念。

一个名为 **myDateTimeFunction** 的 Lambda 函数

发布此 Lambda 函数时，模板中的 `AutoPublishAlias` 行将其链接到名为 `live` 的别名。在本教程的后面部分，此函数的更新会触发部署 Amazon CodeDeploy ，从而逐步将生产流量从原始版本转移到更新的版本。

两个 Lambda 部署验证函数

以下 Lambda 函数是在 CodeDeploy 生命周期挂钩期间执行的。该函数包含代码，用于验证更新的 `myDateTimeFunction` 的部署。验证测试的结果通过 `PutLifecycleEventHookExecutionStatus` API 方法传递给 CodeDeploy 。如果验证测试失败，则部署失败并回滚。

- `CodeDeployHook_beforeAllowTraffic` 在 `BeforeAllowTraffic` 挂钩期间运行。
- `CodeDeployHook_afterAllowTraffic` 在 `AfterAllowTraffic` 挂钩期间运行。

这两个函数的名称以 `CodeDeployHook_` 开头。`CodeDeployRoleForLambda` 角色仅允许在 Lambda 函数中，采用以此前缀开头的名称调用 Lambda `invoke` 方法。有关更多

信息，请参阅《CodeDeploy API 参考》中的AppSpec Amazon Lambda 部署的 “挂钩” 部分和PutLifecycleEventHookExecutionStatus。

自动检测更新的 Lambda 函数

`AutoPublishAlias` 术语指示框架检测 `myDateTimeFunction` 函数何时发生了变化，然后使用 `live` 别名进行部署。

部署配置

部署配置决定了您的 CodeDeploy应用程序将流量从原始版本的 Lambda 函数转移到新版本的速率。此模板指定预定义的部署配置 `Linear10PercentEvery1Minute`。

> **Note**
>
> 您无法在 SA Amazon M 模板中指定自定义部署配置。有关更多信息，请参阅 Create a Deployment Configuration。

部署生命周期挂钩函数

`Hooks` 部分指定在生命周期事件挂钩期间运行的函数。`PreTraffic` 指定在 `BeforeAllowTraffic` 挂钩期间运行的函数。`PostTraffic` 指定在 `AfterAllowTraffic` 挂钩期间运行的函数。

Lambda 调用另一个 Lambda 函数的权限

指定的`lambda:InvokeFunction`权限授予 Amazon SAM 应用程序使用的角色调用 Lambda 函数的权限。当 `CodeDeployHook_beforeAllowTraffic` 和 `CodeDeployHook_afterAllowTraffic` 函数在验证测试期间调用部署的 Lambda 函数时，必须具备该权限。

为 Lambda 函数创建文件

本教程稍后将为更新和部署的函数创建文件。

> **Note**
>
> Lambda 函数可以使用 Amazon Lambda支持的任何运行时。有关更多信息，请参阅 Amazon Lambda 运行时。

创建 Lambda 函数

1. 创建文本文件，并以 `myDateTimeFunction.js` 文件形式保存到 `SAM-Tutorial` 目录中。
2. 将以下 Node.js 代码复制到 `myDateTimeFunction.js` 中。

```
'use strict';

    exports.handler = function(event, context, callback) {

      if (event.body) {
        event = JSON.parse(event.body);
      }

      var sc; // Status code
      var result = ""; // Response payload

      switch(event.option) {
        case "date":
          switch(event.period) {
            case "yesterday":
              result = setDateResult("yesterday");
              sc = 200;
              break;
            case "today":
              result = setDateResult();
              sc = 200;
              break;
            case "tomorrow":
              result = setDateResult("tomorrow");
              sc = 200;
              break;
            default:
              result = {
                "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
              };
              sc = 400;
              break;
          }
          break;

    /*      Later in this tutorial, you update this function by uncommenting
            this section. The framework created by Amazon SAM detects the update
```

```
            and triggers a deployment by CodeDeploy. The deployment shifts
            production traffic to the updated version of this function.

            case "time":
            var d = new Date();
            var h = d.getHours();
            var mi = d.getMinutes();
            var s = d.getSeconds();

            result = {
              "hour": h,
              "minute": mi,
              "second": s
            };
            sc = 200;
            break;
    */
          default:
            result = {
              "error": "Must specify 'date' or 'time'."
            };
            sc = 400;
          break;
      }

      const response = {
        statusCode: sc,
        headers: { "Content-type": "application/json" },
        body: JSON.stringify( result )
      };

      callback(null, response);

      function setDateResult(option) {

        var d = new Date(); // Today
        var mo; // Month
        var da; // Day
        var y; // Year

        switch(option) {
          case "yesterday":
            d.setDate(d.getDate() - 1);
            break;
```

```
          case "tomorrow":
            d.setDate(d.getDate() + 1);
          default:
           break;
        }

        mo = d.getMonth() + 1; // Months are zero offset (0-11)
        da = d.getDate();
        y = d.getFullYear();

        result = {
          "month": mo,
          "day": da,
          "year": y
        };

        return result;
      }
    };
```

Lambda 函数返回昨天、今天或明天的日期、月份和年份。在本教程后面的部分中，您将取消注释更新函数的代码，以返回有关您指定的日期或时间的信息（例如，日期、月份和年份，或当前小时、分钟和秒）。创建的框架 Amazon SAM 会检测并部署该函数的更新版本。

> **Note**
>
> 教程中也使用了此 Lambda 函数。 Amazon Cloud9 Amazon Cloud9 是一个基于云的集成开发环境。有关如何在中创建、执行、更新和调试此函数的信息 Amazon Cloud9，请参阅的 Amazon Lambda 教程 Amazon Cloud9。

为您的 BeforeAllowTraffic Lambda 函数创建一个文件

为 `beforeAllowTraffic` 挂钩 Lambda 函数创建文件。

1. 创建文本文件，并以 `beforeAllowTraffic.js` 文件形式保存到 `SAM-Tutorial` 目录中。
2. 将以下 Node.js 代码复制到 `beforeAllowTraffic.js` 中。该函数在部署的 `BeforeAllowTraffic` 挂钩期间执行。

```
'use strict';
```

```
   const AWS = require('aws-sdk');
   const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
   var lambda = new AWS.Lambda();

   exports.handler = (event, context, callback) => {

    console.log("Entering PreTraffic Hook!");

    // Read the DeploymentId and LifecycleEventHookExecutionId from the event
payload
     var deploymentId = event.DeploymentId;
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    var functionToTest = process.env.NewVersion;
    console.log("BeforeAllowTraffic hook tests started");
    console.log("Testing new function version: " + functionToTest);

    // Create parameters to pass to the updated Lambda function that
    // include the newly added "time" option. If the function did not
    // update, then the "time" option is invalid and function returns
    // a statusCode of 400 indicating it failed.
    var lambdaParams = {
     FunctionName: functionToTest,
     Payload: "{\"option\": \"time\"}",
     InvocationType: "RequestResponse"
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
```

```
            console.log("Validation succeeded");
        lambdaResult = "Succeeded";
          }
          else {
            console.log("Validation failed");
          }

      // Complete the PreTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
 {
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
       }
      });
     }
    });
   }
```

为您的 AfterAllowTraffic Lambda 函数创建一个文件

为 `afterAllowTraffic` 挂钩 Lambda 函数创建文件。

1. 创建文本文件，并以 `afterAllowTraffic.js` 文件形式保存到 `SAM-Tutorial` 目录中。
2. 将以下 Node.js 代码复制到 `afterAllowTraffic.js` 中。该函数在部署的 `AfterAllowTraffic` 挂钩期间执行。

```
'use strict';
```

```
    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PostTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("AfterAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the original "date" parameter. If the function did not
     // update as expected, then the "date" option might be invalid. If
     // the parameter is invalid, the function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"date\", \"period\": \"today\"}",
      InvocationType: "RequestResponse"
     };

     var lambdaResult = "Failed";
     // Invoke the updated Lambda function.
     lambda.invoke(lambdaParams, function(err, data) {
      if (err){ // an error occurred
       console.log(err, err.stack);
       lambdaResult = "Failed";
      }
      else{ // successful response
       var result = JSON.parse(data.Payload);
       console.log("Result: " +  JSON.stringify(result));
          console.log("statusCode: " + result.statusCode);

          // Check if the status code returned by the updated
          // function is 400. If it is, then it failed. If
          // is not, then it succeeded.
       if (result.statusCode != "400"){
```

```
            console.log("Validation of time parameter succeeded");
        lambdaResult = "Succeeded";
          }
          else {
            console.log("Validation failed");
          }

      // Complete the PostTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
       }
      });
     }
    });
   }
```

Package 打包 Amazon SAM 应用程序

现在，SAM-Tutorial 目录下应当具备四个文件：

- beforeAllowTraffic.js
- afterAllowTraffic.js
- myDateTimeFunction.js
- template.yml

现在，您可以使用 Amazon SAM sam package 命令为您的 Lambda 函数和应用程序创建和 CodeDeploy 打包工件。构件将被上传到 S3 存储桶。命令的输出是名为 `package.yml` 的新文件。Amazon SAM sam deploy 命令将在下一步中使用此文件。

> **Note**
>
> 有关 sam package 命令的更多信息，请参阅《Amazon Serverless Application Model 开发人员指南》中的 Amazon SAM CLI 命令参考。

在 `SAM-Tutorial` 目录中，运行以下命令。

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

对于 `s3-bucket` 参数，指定作为本教程先决条件而创建的 Amazon S3 存储桶。指`output-template-file`定 Amazon SAM sam deploy 命令使用的新文件的名称。

部署 S Amazon AM 应用程序

使用带有`package.yml`文件的 Amazon SAM sam deploy 命令来创建您的 Lambda 函数以及 CodeDeploy 应用程序和部署组。 Amazon CloudFormation

> **Note**
>
> 有关 sam deploy 命令的更多信息，请参阅《Amazon Serverless Application Model 开发人员指南》中的 Amazon SAM CLI 命令参考。

在 `SAM-Tutorial` 目录中，运行以下命令。

```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

`--capabilities CAPABILITY_IAM` 参数是授权 Amazon CloudFormation 创建 IAM 角色的必需项。

（可选）检查并测试基础设施

本主题介绍了如何查看基础设施组件以及测试 Lambda 函数。

在运行 **sam deploy** 后查看堆栈结果

1. 在 https://console.aws.amazon.com/cloudformation 上打开 Amazon CloudFormation 控制台。
2. 在导航窗格中，选择 Stacks（堆栈）。`my-date-time-app` 堆栈显示在顶部。
3. 选择 Events（事件）选项卡，以查看哪些事件已完成。您可以在堆栈创建过程中查看事件。堆栈创建完成后，您可以查看所有的堆栈创建事件。
4. 在已选择堆栈的情况下，选择 Resources（资源）。在类型列中，您可以看到 Lambda 函数、`myDateTimeFunction`、`CodeDeployHook_beforeAllowTraffic` 和 `CodeDeployHook_afterAllowTraffic`。您的每个 Lambda 函数的物理 ID 列都包含一个用于在 Lambda 控制台中查看这些函数的链接。

   > **Note**
   >
   > `myDateTimeFunctionLambda` 函数的名称前面有 Amazon CloudFormation 堆栈的名称，并添加了一个标识符，所以看起来像。`my-date-time-app-myDateTimeFunction-123456ABCDEF`

5. 打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。
6. 在导航窗格中，展开 Deploy（部署），然后选择 Applications（应用程序）。
7. 您应该会看到一个由 Amazon CloudFormation 创建的新 CodeDeploy 应用程序，其名称以开头`my-date-time-app-ServerlessDeploymentApplication`。选择此应用程序。
8. 您应当看到一个名称以 `my-date-time-app-myDateTimeFunctionDeploymentGroup` 开头的部署组。选择此部署组。

   在 “部署配置” 下，您应该看到CodeDeployDefault。 LambdaLinear10 PercentEvery 1分钟。

（可选）测试函数（控制台）

1. 打开 Amazon Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 从导航窗格中，选择 `my-date-time-app-myDateTimeFunction` 函数。在控制台中，其名称包含一个标识符，因此看起来类似 `my-date-time-app-myDateTimeFunction-123456ABCDEF`。

3. 选择测试。
4. 在 Event name（事件名称）中，为测试事件输入名称。
5. 为测试事件输入以下内容，然后选择 Create（创建）。

```
{
  "option": "date",
  "period": "today"
}
```

6. 选择测试。在测试事件列表中，您应当只看到自己的测试事件。

   对于 Execution result（执行结果），您应当看到 succeeded（已成功）。
7. 在 Execution result（执行结果）下，展开 Details（详细信息）以查看结果。您应当看到当前的月份、日期和年份。

（可选）测试函数（Amazon CLI）

1. 找到 Lambda 函数的 ARN。当您查看函数时，它显示在 Lambda 控制台的顶部。
2. 运行如下命令。*your-function-arn*替换为函数 ARN。

```
aws lambda invoke \
--function your-function-arn \
--cli-binary-format raw-in-base64-out \
--payload "{\"option\": \"date\", \"period\": \"today\"}" out.txt
```

3. 打开 `out.txt` 以确认结果中是否包含当前的月份、日期和年份。

步骤 2：更新 Lambda 函数

在本主题中，您将更新 `myDateTimeFunction.js` 文件。在下一个步骤中，您将使用该文件部署更新的函数。这会触发通过 CodeDeploy 将生产流量从当前版本的 Lambda 函数转移到更新的版本来部署它。

更新 Lambda 函数

1. 打开 `myDateTimeFunction.js`。
2. 删除两个注释标记（“`/*`”和“`*/`”），以及 `switch` 块中名为 `time` 的 `case` 开头和结尾处的说明文本。

您可以通过取消注释代码，将新参数 time 传递给该函数。如果将 time 传递给更新的函数，则函数将返回当前的 hour、minute 和 second。

3. 保存 myDateTimeFunction.js。它应该类似以下内容：

```
'use strict';

exports.handler = function(event, context, callback) {

  if (event.body) {
    event = JSON.parse(event.body);
  }

  var sc; // Status code
  var result = ""; // Response payload

  switch(event.option) {
    case "date":
      switch(event.period) {
        case "yesterday":
          result = setDateResult("yesterday");
          sc = 200;
          break;
        case "today":
          result = setDateResult();
          sc = 200;
          break;
        case "tomorrow":
          result = setDateResult("tomorrow");
          sc = 200;
          break;
        default:
          result = {
            "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
          };
          sc = 400;
          break;
      }
      break;
      case "time":
        var d = new Date();
        var h = d.getHours();
        var mi = d.getMinutes();
```

```
        var s = d.getSeconds();

        result = {
          "hour": h,
          "minute": mi,
          "second": s
        };
        sc = 200;
        break;

      default:
        result = {
          "error": "Must specify 'date' or 'time'."
        };
        sc = 400;
      break;
  }

  const response = {
    statusCode: sc,
    headers: { "Content-type": "application/json" },
    body: JSON.stringify( result )
  };

  callback(null, response);

  function setDateResult(option) {

    var d = new Date(); // Today
    var mo; // Month
    var da; // Day
    var y; // Year

    switch(option) {
      case "yesterday":
        d.setDate(d.getDate() - 1);
        break;
      case "tomorrow":
        d.setDate(d.getDate() + 1);
      default:
       break;
    }

    mo = d.getMonth() + 1; // Months are zero offset (0-11)
```

```
    da = d.getDate();
    y = d.getFullYear();

    result = {
      "month": mo,
      "day": da,
      "year": y
    };

    return result;
  }
};
```

步骤 3：部署更新的 Lambda 函数

在本步骤中，您将使用更新的 `myDateTimeFunction.js`，以更新并启动 Lambda 函数的部署。您可以在 CodeDeploy 或 Amazon Lambda 控制台中监控部署进度。

Amazon SAM 模板中的这一`AutoPublishAlias: live`行会使您的基础架构检测到使用`live`别名的函数的更新。函数的更新会触发部署 CodeDeploy ，从而将生产流量从函数的原始版本转移到更新的版本。

sam package 和 sam deploy 命令用于更新和触发 Lambda 函数的部署。您已在 Package 打包 Amazon SAM 应用程序 和 部署 S Amazon AM 应用程序 中执行这些命令。

部署更新的 Lambda 函数

1. 在 `SAM-Tutorial` 目录中，运行以下命令。

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml  \
  --s3-bucket amzn-s3-demo-bucket
```

   这会创建一组新的构件，这些构件引用您的 S3 存储桶中更新的 Lambda 函数。

2. 在 `SAM-Tutorial` 目录中，运行以下命令。

```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
```

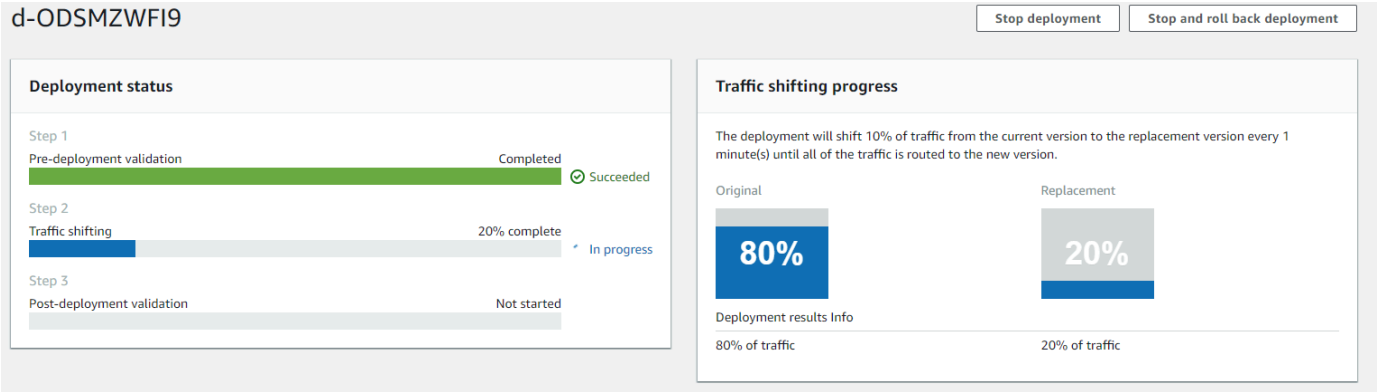
```
  --capabilities CAPABILITY_IAM
```

由于堆栈名称仍为`my-date-time-app`，因此可以 Amazon CloudFormation 识别这是堆栈更新。要查看更新的堆栈，请返回 Amazon CloudFormation 控制台，然后从导航窗格中选择 Stacks。

（可选）用于在部署期间查看流量（CodeDeploy 控制台）

1. 打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。
2. 在导航窗格中，展开 “应用程序”，然后选择您的 my-date-time-app-ServerlessDeploymentApplication 应用程序。
3. 在 Deployment groups（部署组）中，选择应用程序的部署组。其状态应为 In progress（正在进行）。
4. 在 Deployment group history（部署组历史记录）中，选择正在进行中的部署。

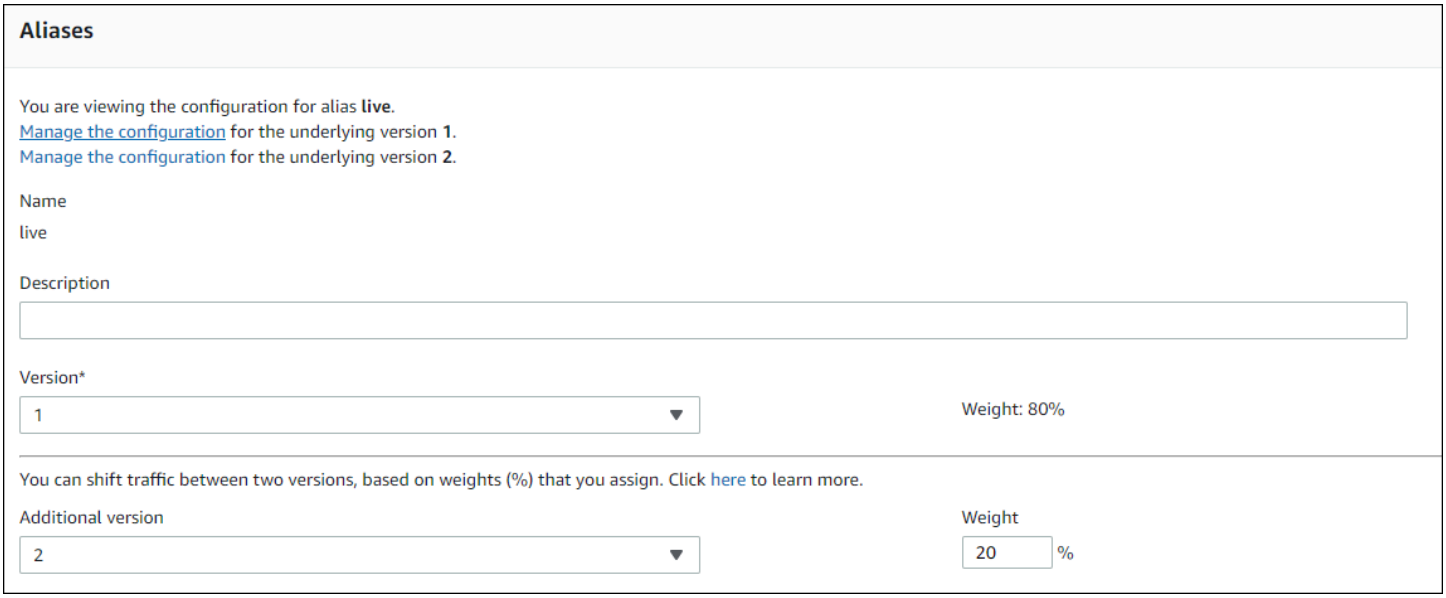
   此页面上的 Traffic shifting（流量转移）进度条以及 Original（原始）和 Replacement（替换）框中的百分比显示了其进度。



（可选）在部署期间查看流量（Lambda 控制台）

1. 打开 Amazon Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 从导航窗格中，选择 `my-date-time-app-myDateTimeFunction` 函数。
   在控制台中，其名称包含一个标识符，因此看起来类似 `my-date-time-app-myDateTimeFunction-123456ABCDEF`。
3. 依次选择别名和 live。

原始函数版本（版本 1）和更新的函数版本（版本 2）旁边的权重，显示了在加载此 Amazon Lambda 控制台页面时提供给每个版本的流量。该页面不会随时间更新权重。如果每隔一分钟刷新一次页面，则版本 1 的权重降低 10%，版本 2 的权重增加 10%，直到版本 2 的权重达到 100%。



步骤 4：查看部署结果

在本步骤中，您将查看部署的结果。如果部署成功，即可确认更新的 Lambda 函数收到了生产流量。如果部署失败，您可以使用 CloudWatch 日志在 Lambda 函数中查看在部署生命周期挂钩期间运行的验证测试的输出。

主题

- 测试部署的函数
- 在 CloudWatch 日志中查看挂钩事件

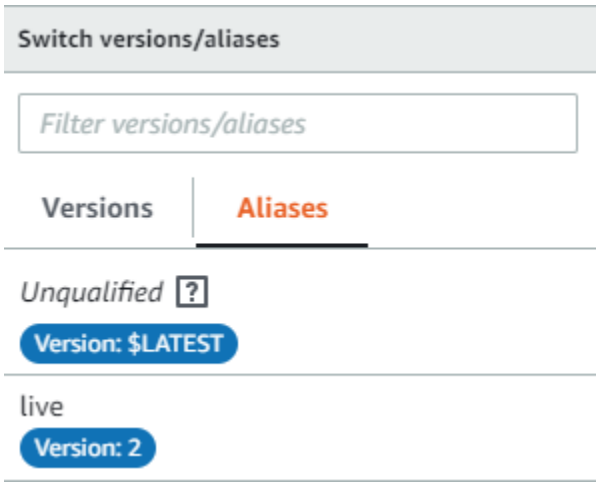
测试部署的函数

sam deploy 命令更新 `my-date-time-app-myDateTimeFunction` Lambda 函数。函数版本更新为 2 并添加到 `live` 别名。

在 Lambda 控制台中查看更新

1. 打开 Amazon Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 从导航窗格中，选择 `my-date-time-app-myDateTimeFunction` 函数。
   在控制台中，其名称包含一个标识符，因此看起来类似 `my-date-time-app-myDateTimeFunction-123456ABCDEF`。

3. 依次选择 Qualifiers（限定词）和 Aliases（别名）。部署完成后（大约 10 分钟），对于 `live` 别名，您应当看到 Version: 2（版本：2）。



4. 在 Function code（函数代码）中，查看函数的源代码。此时应显示您的更改。

5. （可选）您可以使用 步骤 2：更新 Lambda 函数 中的测试说明测试更新的函数。采用以下有效负载创建新的测试事件，然后确认结果中是否包含当前的小时、分钟和秒。

```
{
    "option": "time"
  }
```

要使用 Amazon CLI 来测试更新后的函数，请运行以下命令，然后打开`out.txt`以确认结果包含当前小时、分钟和秒。

```
aws lambda invoke --function your-function-arn --payload "{\"option\": \"time\"}"
 out.txt
```

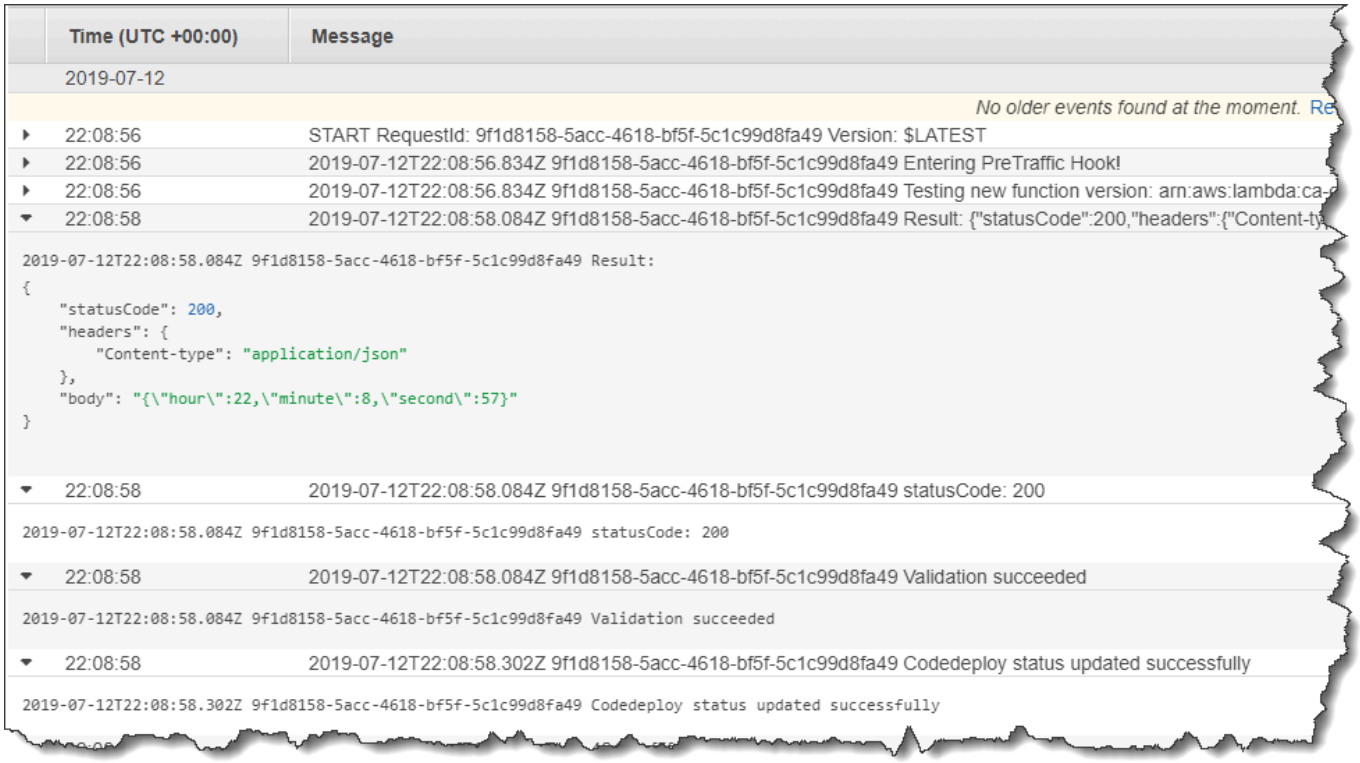
> **Note**
>
> 如果您在部署完成之前使用 Amazon CLI 来测试您的函数，则可能会收到意想不到的结果。这是因为每分钟将10%的流量 CodeDeploy 逐渐转移到更新的版本。在部署过程中，部分流量仍然指向原始版本，因此 `aws lambda invoke` 可能使用原始版本。10 分钟后，部署完成，所有流量均指向函数的新版本。

在 CloudWatch 日志中查看挂钩事件

在`BeforeAllowTraffic`挂钩期间， CodeDeploy 执行您的 Lambd `CodeDeployHook_beforeAllowTraffic` a 函数。在`AfterAllowTraffic`挂钩期间， CodeDeploy 执行您的 Lambd `CodeDeployHook_afterAllowTraffic` a 函数。每个函数运行验证测试，使用新的 `time` 参数调用更新的函数版本。如果 Lambda 函数更新成功，则 `time` 选项不会导致错误，并且验证成功。如果函数未更新，则无法识别的参数会导致错误，并且验证失败。这些验证测试仅供演示之用。您自行编写测试来验证部署。您可以使用 CloudWatch 日志控制台查看您的验证测试。

查看您的 CodeDeploy 挂钩事件

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 从导航窗格中，选择 Logs（日志）。
3. 从日志组列表中选择/aws/lambda/CodeDeployHook_ beforeAllowTraffic 或/aws/lambda/CodeDeployHook_ afterAllowTraffic。
4. 选择日志流。您应当只看到一个。
5. 展开事件以查看其详细信息。



第 5 步：清理

为避免对您在本教程中使用的资源收取更多费用，请删除您的 Amazon SAM 模板创建的资源和您的 Lambda 验证函数创建的 CloudWatch 日志。

删除您的 Amazon CloudFormation 堆栈

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/cloudformat ion 上打开 Amazon CloudFormation 控制台。
2. 在 Stacks（堆栈）列中，选择 `my-date-time-app` 堆栈，然后选择 Delete（删除）。
3. 当系统提示时，选择 Delete stack（删除堆栈）。由创建的 Lambda 函数、 CodeDeploy应用程序和部署组以及 IAM 角色 Amazon SAM 已删除。

在 “日志” 中删除您的 CloudWatch 日志

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 从导航窗格中，选择 Logs（日志）。
3. 从日志组列表中，选择/aws/lambda/CodeDeployHook_ 旁边的按钮beforeAllowTraffic。
4. 从 Actions（操作）中，选择 Delete log group（删除日志组），然后选择 Yes, Delete（是，删除）。
5. 从日志组列表中，选择/aws/lambda/CodeDeployHook_ 旁边的按钮afterAllowTraffic。
6. 从 Actions（操作）中，选择 Delete log group（删除日志组），然后选择 Yes, Delete（是，删除）。

与 CodeDeploy 代理合作

Amazon CodeDeploy 代理是一个软件包，在实例上安装和配置后，该实例就可以在 CodeDeploy 部署中使用。

Amazon 支持 CodeDeploy 代理的最新次要版本。目前，最新的次要版本是 1.7.x。

> **Note**
>
> 只有在部署到 EC2 /Onlide 计算平台时，才需要使用 CodeDeploy 代理。使用 Amazon ECS 或 Amazon Lambda 计算平台的部署不需要该代理。

安装该代理时，将在实例上放置一个配置文件。此文件用于指定代理的工作方式。此配置文件指定了与实例交互 Amazon CodeDeploy 时要使用的目录路径和其他设置。可以更改此文件中的某些配置选项。有关使用 CodeDeploy 代理配置文件的信息，请参见CodeDeploy 代理配置参考。

有关使用 CodeDeploy 代理的更多信息，例如安装、更新和验证版本的步骤，请参阅管理 CodeDeploy 代理操作。

主题

- CodeDeploy 代理支持的操作系统
- CodeDeploy 代理的通信协议和端口
- CodeDeploy代理的版本历史记录
- 管理 CodeDeploy 流程
- 应用程序修订和日志文件清理
- CodeDeploy 代理安装的文件
- 管理 CodeDeploy 代理操作

CodeDeploy 代理支持的操作系统

支持的亚马逊 EC2 AMI 操作系统

该 CodeDeploy 代理已在以下 Amazon EC2 AMI 操作系统上进行了测试：

- Amazon Linux 2023（ARM、x86）
- Amazon Linux 2（ARM、x86）
- Microsoft Windows Server 2022、2019
- Red Hat Enterprise Linux（RHEL）9.x、8.x、7.x
- Ubuntu Server 22.04 LTS、20.04 LTS、18.04 LTS、16.04 LTS

该 CodeDeploy 代理以开源形式提供，以供您适应您的需求。它可以与其他 Amazon EC2 AMI 操作系统一起使用。有关更多信息，请访问中的CodeDeploy 代理存储库 GitHub。

支持的本地操作系统

该 CodeDeploy 代理已在以下本地操作系统上进行了测试：

- Microsoft Windows Server 2022、2019
- Red Hat Enterprise Linux（RHEL）9.x、8.x、7.x
- Ubuntu Server 22.04 LTS、20.04 LTS

该 CodeDeploy 代理以开源形式提供，以供您适应您的需求。它可与其他本地实例操作系统配合使用。有关更多信息，请访问中的CodeDeploy 代理存储库 GitHub。

CodeDeploy 代理的通信协议和端口

CodeDeploy 代理使用 HTTPS 通过端口 443 进行出站通信。

当 CodeDeploy 代理在 EC2 实例上运行时，它将使用EC2 元数据终端节点来检索与实例相关的信息。了解有关限制和授予实例元数据服务访问权限的更多信息。

CodeDeploy代理的版本历史记录

您的实例必须运行支持的 CodeDeploy 代理版本。当前支持的最低版本为 1.7.x。

> **Note**
>
> 我们建议使用最新版本的 CodeDeploy 代理。如果您遇到问题，请在联系 Su Amazon pport 之前更新到最新版本。有关升级信息，请参阅更新 CodeDeploy 代理。

下表列出了该 CodeDeploy 代理的所有版本以及每个版本中包含的功能和增强功能。

版本	发行日期	详细信息
1.8.0	2025 年 7 月 31 日	改动：在 Windows CodeDeploy 代理中将捆绑的 Ruby 升级到 3.2。
1.7.1	2024 年 11 月 14 日	已更改：更新了安全补丁的依赖项。
1.7.0	2024 年 3 月 6 日	新增： CodeDeploy 代理`:disable_imds_v1:` 配置文件的配置设置。使用此设置可禁用 IMDSv2 错误发生 IMDSv1 时的回退功能。默认为 `false`（启用回退）。有关更多信息，请参阅CodeDeploy 代理配置参考。 新增：支持 Red Hat Enterprise Linux 9（RHEL 9）操作系统。 新增：在 Ubuntu Server 上支持 Ruby 版本 3.1 和 3.2。 已修复：如果 CodeDeploy 代理配置文件加载失败，CodeDeploy 代理现在会生成用户友好的错误。 改动：在 Windows CodeDeploy 代理中将 Ruby 升级到 2.7.8-1。
1.6.0	2023 年 3 月 30 日	新增：对 Ruby 3.1、3.2 的支持。 新增：对 Amazon Linux 2023 的支持。 新增：对 Windows Server 2022 的支持。 已更改：Windows Server 实例的 `verbose` 默认设置现在为 `false`。要继续在 Windows 的日志文件中打印调试消息，必须将 `verbose` 设置为 `true`。 已删除：对 Windows Server 2016 和 Windows Server 2012 R2 的支持。 已删除：对 Amazon Linux 2018.03.x 的支持。

版本	发行日期	详细信息
1.5.0	2023 年 3 月 3 日	新增：对 Ruby 3 的支持。 新增：对 Ubuntu 22.04 的支持。 已修复：启动后不久重新启动 CodeDeploy 代理会导致代理挂起的问题。 已@@ 更改：现在，如果 CodeDeploy 代理服务在运行挂钩脚本时意外重启，代理启动时将无法部署主机。此修复可让您避免在重试部署前等待 70 分钟的超时时间。 弃用通知： CodeDeploy 代理 1.5.0 是最后一个支持 Windows Server 2016 和 Windows Server 2012 R2 的版本。 已删除：在 Ubuntu 14.04 LTS、Windows Server 2008 R2 和 Windows Server 2008 R2 32 位上支持该 CodeDeploy代理。
1.4.1	2022 年 12 月 6 日	已修复：与日志记录相关的安全漏洞。 增强：改进了轮询主机命令时的日志记录。

版本	发行日期	详细信息
1.4.0	2022 年 8 月 31 日	新增：对 Red Hat Enterprise Linux 8 的支持。 新增：支持 Windows 版 CodeDeploy 代理上的长文件路径。要启用长文件路径，您需要设置相应的 Windows 注册表项，然后重新启动代理。有关更多信息，请参阅 长文件路径会导致“没有这样的文件或目录”错误。 已修复：磁盘已满时解压缩操作出现问题。现在， CodeDeploy 代理会检测到解压缩的退出代码 50，表示磁盘已满，删除部分提取的文件，并引发异常以将故障发布到 CodeDeploy服务器。该错误消息显示为生命周期事件错误消息，主机级部署将停止，而不会卡住或超时。 已修复：会导致代理失败的问题。 已修复：钩子在边缘争用条件下超时的问题。没有脚本的钩子现在将继续运行，并且不会再导致失败或超时。 已更改： CodeDeploy 代理`bin`目录中的`update`脚本已被删除，因为该脚本已不再使用。 改动：Windows 服务器 CodeDeploy 代理现在捆绑了 Ruby 2.7。 已更改：添加了新的环境变量，供挂钩脚本使用，具体取决于部署包的来源（Amazon S3 或 GitHub）。 有关更多信息，请参阅 挂钩的环境变量可用性。 **Important** 弃用通知： CodeDeploy代理 1.4.0 是包含 32 位 Windows 服务器安装程序的最后一个版本。 弃用通知： CodeDeploy代理 1.4.0 是最后一个支持 Windows Server 2008 R2 的版本。

版本	发行日期	详细信息
		已删除：在以下亚马逊上支持该 CodeDeploy 代理 EC2 AMIs：亚马逊 Linux 2014.09、2016.03、2016.09 和 2017.03。

版本	发行日期	详细信息
1.3.2	2021 年 5 月 6 日	**Important** CodeDeploy 代理 1.3.2 解决了影响运行该代理的 Windows 主机的 CVE-2018-1000201。CVE 引用了 ruby-ffi，这是代理的依赖关系。 CodeDeploy 如果您的代理安装了 Amazon S EC2 ystems Manager (SSM)，并且设置为自动更新，则无需执行任何操作。否则，需要采取措施来手动更新代理。要升级代理，请按照在 Windows 服务器上更新 CodeDeploy 代理中的说明进行操作。 已修复：在 Ubuntu 20.04 及更高版本上安装 CodeDeploy 代理时出现问题。 已修复：由于未正确处理相对路径，在提取压缩文件时出现间歇性问题。 新增：对 Windows 实例的 Amazon PrivateLink 和 VPC 终端节点的支持。 新增： AppSpec 文件改进，如下所述。 • 现在，您可以在创建本地部署时为 AppSpec 文件指定自定义文件名。有关更多信息，请参阅 创建本地部署。 • 该 AppSpec 文件现在可以有`.yaml`文件扩展名。 • 现在，您可以使用文件中的新可选`file_exists_behavior` 设置覆盖已部署的 AppSpec文件。有关更多信息，请参阅 AppSpec “文件” 部分（仅EC2限本地部署）。 已@@ 升级： CodeDeploy 现在使用适用于 Ruby 3.0 的 Amazon SDK。

版本	发行日期	详细信息
1.3.1	2020 年 12 月 22 日	已修复：导致本地实例无法启动的 1.3.0 问题。
1.3.0	2020 年 11 月 10 日	Important 此版本已被弃用。 已修复：删除了不再使用的过期证书。 已@@ 修复：从使用的代理卸载脚本中删除了提示消息 Amazon Systems Manager，从而可以更轻松地将主机或队列降级到代理的先前版本。
1.2.1	2020 年 9 月 23 日	已更改：将 适用于 Ruby 的 Amazon SDK 依赖项从 v2 升级到 v3。 新增：Support fo IMDSv2 r. 包括对 h IMDSv2 ttp 请求失败 IMDSv1 时的静默回退。 已更改：更新了安全补丁的 Rake 和 Rubyzip 依赖项。 已修复：确保代理启动时，空的 PID 文件将返回状态 No `CodeDeploy Agent Running` 并清理 PID 文件。

版本	发行日期	详细信息
1.1.2	2020 年 8 月 4 日	新增：对 Ubuntu Server 19.10 和 20.04 的支持。 注意：19.10版本已 end-of-life过期，Ubuntu或不再支持该版本。 CodeDeploy 新增：改进了 Linux 和 Ubuntu 的内存效率，可以更及时地释放预留内存。 新增：与 Windows Server“静默清理”的兼容性，这会导致代理在某些情况下无响应。 新增：在清理过程中忽略非空目录，以避免部署失败。 新增：对洛杉矶（洛杉矶） Amazon 本地区域的 Support 支持。 新增：从实例元数据中提取可用区以提供与 L Amazon ocal Zones 的兼容性。 新增：用户现在可以在子目录中提供存档，无需将其存储在根目录中。 新增：检测到 Rubyzip 存在的、可能导致内存泄漏的问题。更新了 unzip 命令，使其在使用 Rubyzip 之前首先尝试使用系统安装的 unzip 实用程序。 新增：`:enable_auth_policy:` 作为代理配置设置。 已更改：现在将忽略解压缩警告，因此部署将继续进行。

版本	发行日期	详细信息
1.1.0	2020 年 6 月 30 日	已更改： CodeDeploy代理的版本控制现在遵循 Ruby 标准版本控制惯例。 新增：安装和更新命令的新参数，允许从命令行安装特定代理版本。 已删除：移除了适用于 Linux 和 Ubuntu 的 CodeDeploy 代理自动更新程序。要配置 CodeDeploy 代理的自动更新，请参阅使用安装 CodeDeploy 代理 Amazon Systems Manager。
1.0.1.1597	2018 年 11 月 15 日	增强： CodeDeploy 支持 Ubuntu 18.04。 增强： CodeDeploy 支持 Ruby 2.5。 增强： CodeDeploy 支持 FIPS 端点。有关联邦信息处理标准端点的更多信息，请参阅 FIPS 140-2 概述。有关可与之配合使用的终端节点 CodeBuild，请参阅CodeDeploy区域和终端节点。
1.0.1.1518	2018 年 6 月 12 日	增强功能：修复了 CodeDeploy 代理在接受轮询请求时关闭时导致错误的问题。 增强：添加了部署跟踪功能，该功能可防止 CodeDeploy 代理在部署进行时关闭。 增强功能：改进了删除文件时的性能。
1.0.1.1458	2018 年 3 月 6 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强功能：改进了证书验证，以支持更多受信任的机构。 增强：修复了在包含 BeforeInstall 生命周期事件的部署期间导致本地 CLI 失败的问题。 增强：修复了更新 CodeDeploy 代理时可能导致主动部署失败的问题。

版本	发行日期	详细信息
1.0.1.1352	2017 年 11 月 16 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：引入了一项新功能，用于在安装 CodeDeploy 代理的 EC2本地计算机或实例上测试和调试 /Londest 部署。
1.0.1.1106	2017 年 5 月 16 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：引入了对处理目标位置上的不作为来自最新成功部署的应用程序修订的一部分的内容的新支持。针对现有内容的部署选项现在包括保留内容、覆盖内容或使部署失败。 增强：使 CodeDeploy 代理与 2.9.2 版本的 适用于 Ruby 的 Amazon SDK (aws-sdk-core2.9.2) 兼容。
1.0.1.1095	2017 年 3 月 29 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强：在中国（北京）地区引入了对 CodeDeploy 代理的支持。 增强功能：生命周期事件挂钩进行调用时，允许 Puppet 在 Windows Server 实例上运行。 增强功能：改进对 `untar` 操作的处理。
1.0.1.1067	2017 年 1 月 6 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强功能：修改了很多错误消息，以包含导致部署故障的更具体的原因。 增强功能：修复了在某些部署期间 CodeDeploy 代理无法识别要部署的正确应用程序修订的问题。 增强功能：恢复在 `untar` 操作之前和之后使用 `pushd` 和 `popd`。

版本	发行日期	详细信息
1.0.1.1045	2016 年 11 月 21 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强：使 CodeDeploy 代理与 2.6.11 版本的 适用于 Ruby 的 Amazon SDK (aws-sdk-core2.6.11) 兼容。
1.0.1.1037	2016 年 10 月 19 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 亚马逊 Linux、RHEL 和 Ubuntu 服务器实例的 CodeDeploy 代理已更新，其中包含以下更改。对于 Windows Server 实例，最新版本仍为 1.0.1.998。 增强功能：该代理现在可以确定哪个 Ruby 版本安装在实例上，以使它可以使用该版本调用 `codedeploy-agent` 脚本。
1.0.1.1011.1	2016 年 8 月 17 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强功能：删除了由于外壳支持问题而在 1.0.1.1011 版中引入的更改。此版本代理在功能上与 2016 年 7 月 11 日发布的 1.0.1.998 版相同。
1.0.1.1011	2016 年 8 月 15 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 亚马逊 Linux、RHEL 和 Ubuntu 服务器实例的 CodeDeploy 代理已更新，其中包含以下更改。对于 Windows Server 实例，最新版本仍为 1.0.1.998。 功能：增加了对在使用 systemd init 系统的操作系统上使用 bash shell 调用 CodeDeploy 代理的支持。 增强：在 CodeDeploy 代理和 CodeDeploy 代理更新程序中启用了对所有版本的 Ruby 2.x 的支持。更新的 CodeDeploy 代理不再仅依赖于 Ruby 2.0。（ CodeDeploy 代理安装程序的 deb 和 rpm 版本仍然需要 Ruby 2.0。）

版本	发行日期	详细信息
1.0.1.998	2016 年 7 月 11 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 增强：修复了对使用 roo t 用户以外的用户配置文件运行 CodeDeploy 代理的支持。名为 `USER` 的变量将被替换为 `CODEDEPLOY_USER` ，以避免与环境变量发生冲突。
1.0.1.966	2016 年 6 月 16 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：引入了对使用 roo t 用户以外的用户配置文件运行 CodeDeploy 代理的支持。 增强功能：修复了对指定您希望 CodeDeploy 代理为部署组存档的应用程序修订数量的支持。 增强：使 CodeDeploy 代理与 2.3 版本的 适用于 Ruby 的 Amazon SDK (aws-sdk-core 2.3) 兼容。 增强功能：修复了与部署期间的 UTF-8 编码有关的问题。 增强功能：提高了标识进程名称时的准确性。
1.0.1.950	2016 年 3 月 24 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：添加了安装代理支持。 增强：更新了安装脚本，使其在已安装最新版本时不下载 CodeDeploy 代理。
1.0.1.934	2016 年 2 月 11 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：引入了对指定您要 CodeDeploy 代理为部署组存档的应用程序修订数量的支持。

版本	发行日期	详细信息
1.0.1.880	2016 年 1 月 11 日	注意：此版本不再受支持，可能会导致部署失败。 增强：使 CodeDeploy 代理与 2.2 版本的 适用于 Ruby 的 Amazon SDK (aws-sdk-core 2.2) 兼容。版本 2.1.2 仍受支持。
1.0.1.854	2015 年 11 月 17 日	注意：此版本不再受支持。如果您使用此版本，您的部署可能会失败。 功能：引入了对 SHA-256 哈希算法的支持。 功能：在 `.version` 文件中引入了版本跟踪支持。 功能：通过使用环境变量使部署组 ID 可用。 增强：增加了对使用 Amazon 日志监控 CodeDeploy 代理 CloudWatch 日志的支持。

有关相关信息，请参阅下列内容：

- 确定 CodeDeploy 代理的版本
- 安装 CodeDeploy 代理

有关 CodeDeploy 代理版本的历史记录，请参阅上的 Release 存储库 GitHub。

管理 CodeDeploy 流程

默认情况下， CodeDeploy 代理的所有 Linux 发行版（rpm 和 deb）都使用 systemd 来管理代理进程。

但是，rpm 和 deb 发行版都附带了位于 `/etc/init.d/codedeploy-agent` 的启动脚本。根据您使用的发行版，在使用 `sudo service codedeploy-agent restart` 这样的命令时，可能会运行 `/etc/init.d` 中的脚本来启动代理进程，而不是允许 `systemd` 管理该进程。在 `/etc/init.d` 上运行脚本是不可取的。

为防止出现此问题，对于支持 `systemd` 的系统，我们建议使用 `systemctl` 实用程序进行任何代理操作，而不是使用 `service` 命令。

例如，要重新启动 CodeDeploy 代理，请使用`sudo systemctl restart codedeploy-agent`该`service`实用程序中的等效命令。

应用程序修订和日志文件清理

CodeDeploy 代理会存档实例上的修订和日志文件。 CodeDeploy 代理会清理这些工件以节省磁盘空间。

应用程序修订部署日志：您可以使用代理配置文件中的:max_revisions: 选项，通过输入任何正整数来指定要存档的应用程序修订数量。 CodeDeploy 还会存档这些修订的日志文件。所有其他文件将被删除，但上次成功部署的日志文件除外。该日志文件将始终保留，即使失败的部署数量超过保留的修订数量也是如此。如果未指定任何值，则除了当前部署的修订版外，还 CodeDeploy 保留五个最新的修订版。

CodeDeploy 日志：对于 Amazon Linux、Ubuntu Server 和 RHEL 实例， CodeDeploy 代理会轮换文件夹下的日志文件。`/var/log/aws/codedeploy-agent`日志文件将在每天 00:00:00（实例时间）轮换。日志文件会在七天后删除。已轮换日志文件的命名模式是 `codedeploy-agent.`*`YYYYMMDD`*`.log`。

CodeDeploy 代理安装的文件

CodeDeploy 代理将修订、部署历史记录和部署脚本存储在实例的根目录中。该目录的默认名称和位置是：

对于 Amazon Linux、Ubuntu Server 和 RHEL 实例为 `'/opt/codedeploy-agent/deployment-root'`。

对于 Windows Server 实例为 `'C:\ProgramData\Amazon\CodeDeploy'`。

您可以使用 CodeDeploy 代理配置文件中的 root_dir 设置来配置目录的名称和位置。有关更多信息，请参阅 CodeDeploy 代理配置参考。

下面是根目录下文件和目录结构的示例。该结构假定有 N 个部署组，每个部署组包含 N 个部署。

```
|--deployment-root/
|-- deployment group 1 ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
```

```
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment group 2 ID
|    |-- deployment 1 ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment 2 ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment N ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|-- deployment group N ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment-instructions
|    |-- [deployment group 1 ID]_cleanup
|    |-- [deployment group 2 ID]_cleanup
|    |-- [deployment group N ID]_cleanup
|    |-- [deployment group 1 ID]_install.json
|    |-- [deployment group 2 ID]_install.json
|    |-- [deployment group N ID]_install.json
|    |-- [deployment group 1 ID]_last_successful_install
|    |-- [deployment group 2 ID]_last_successful_install
|    |-- [deployment group N ID]_last_successful_install
|    |-- [deployment group 1 ID]_most_recent_install
|    |-- [deployment group 2 ID]_most_recent_install
|    |-- [deployment group N ID]_most_recent_install
|-- deployment-logs
|    |-- codedeploy-agent-deployments.log
```

- Deployment Group ID 文件夹代表您的每个部署组。部署组目录的名称是其 ID（例如，`acde1916-9099-7caf-fd21-012345abcdef`）。每个部署组目录都包含该部署组中每次尝试的部署的一个子目录。

  您可以使用 batch-get-deployments 命令查找部署组 ID。

- Deployment ID 文件夹代表部署组中的每个部署。每个部署目录的名称都是其 ID。每个文件夹都包含：
  - bundle.tar，一个压缩文件，其中包含部署修订的内容。如果要查看修订，请使用 zip 解压实用程序。
  - deployment-archive，一个目录，其中包含部署修订的内容。
  - logs，一个包含 `scripts.log` 文件的目录。此文件列出了部署 AppSpec 文件中指定的所有脚本的输出。

  如果您想查找部署文件夹，但不知道其部署 ID 或部署组 ID，则可以使用Amazon CodeDeploy 控制台或 Amazon CLI 来查找它们。有关更多信息，请参阅 查看 CodeDeploy 部署详细信息 。

  部署组中可以存档的默认最大部署数为五个。达到该数量后，将来的部署将被存档，最旧的存档将被删除。您可以使用 CodeDeploy 代理配置文件中的 max_re visions 设置来更改默认值。有关更多信息，请参阅 CodeDeploy 代理配置参考。

  > **Note**
  >
  > 如果您想恢复已存档部署使用的硬盘空间，请将 max_revisions 设置更新为较低的数量，例如 1 或 2。下一次部署将删除已存档的部署，以便数量等于您指定的数量。

- deployment-instructions 包含每个部署组的四个文本文件：
  - [Deployment Group ID]-cleanup，一个文本文件，其中包含部署期间运行的每个命令的撤消版本。示例文件名为 `acde1916-9099-7caf-fd21-012345abcdef-cleanup`。
  - [Deployment Group ID]-install.json，在最近的部署过程中创建的 JSON 文件。它包含部署期间运行的命令。示例文件名为 `acde1916-9099-7caf-fd21-012345abcdef-install.json`。
  - [Deployment Group ID]_last_successfull_install，一个文本文件，其中列出了上次成功部署的存档目录。此文件是在 CodeDeploy 代理将部署应用程序中的所有文件复制到实例时创建的。 CodeDeploy 代理在下次部署时使用它来确定要运行的`BeforeInstall`脚本`ApplicationStop`和脚本。示例文件名为 `acde1916-9099-7caf-fd21-012345abcdef_last_successfull_install`。

- [Deployment Group ID]_most_recent_install，一个文本文件，其中列出了最近一次部署的存档目录的名称。该文件在部署中的文件成功下载时创建。[deployment group ID]_last_successfull_install 文件在该文件之后创建，即下载的文件被复制到其最终目的地时。示例文件名为 `acde1916-9099-7caf-fd21-012345abcdef_most_recent_install`。
- deployment-logs 包含以下日志文件：
  - codedeploy-agent.yyyymmdd.log 文件按天创建，只要当天发生部署活动，就会创建该文件。每个日志文件都包含有关当天部署的信息。这些日志文件可能对调试权限问题等很有用。日志文件最初名为 `codedeploy-agent.log`。第二天，其部署日期将被插入到文件名中。例如，如果今天是 2018 年 1 月 3 日，您可以在 `codedeploy-agent.log` 中看到有关今天的所有部署的信息。明天，也就是 2018 年 1 月 4 日，日志文件将重命名为 `codedeploy-agent.20180103.log`。
  - codedeploy-agent-deployments.log 为每个部署编译`scripts.log`文件内容。`scripts.log` 文件位于每个 `logs` 文件夹下的 `Deployment ID` 子文件夹中。此文件中的条目前面带有部署 ID。例如，“`[d-ABCDEF123]LifecycleEvent - BeforeInstall`”可能是在 ID 为 `d-ABCDEF123` 的部署期间写入的。当`codedeploy-agent-deployments.log`达到其最大大小时， CodeDeploy 代理会继续向其写入内容，同时删除旧内容。

管理 CodeDeploy 代理操作

本部分中的说明向您介绍如何安装、卸载、重新安装或更新 CodeDeploy 代理，以及如何验证 CodeDeploy 代理是否正在运行。

主题

- 验证 CodeDeploy 代理是否正在运行
- 确定 CodeDeploy 代理的版本
- 安装 CodeDeploy 代理
- 更新 CodeDeploy 代理
- 卸载 CodeDeploy 代理
- 将 CodeDeploy 代理日志发送到 CloudWatch

验证 CodeDeploy 代理是否正在运行

本部分介绍在您怀疑 CodeDeploy 代理已停止在某个实例上运行时要运行的命令。

主题

- 验证适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理是否正在运行
- 验证适用于 Ubuntu Server 的 CodeDeploy 代理是否正在运行
- 验证适用于 Windows Server 的 CodeDeploy 代理是否正在运行

验证适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理是否正在运行

要查看 CodeDeploy 代理是否已安装且正在运行，请登录到相应的实例，并运行以下命令：

```
systemctl status codedeploy-agent
```

如果命令返回错误，则没有安装 CodeDeploy 代理。请按照安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理中所述进行安装。

如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于 `The AWS CodeDeploy agent is running` 的消息。

如果您看到类似于 `error: No AWS CodeDeploy agent running` 的消息，请启动该服务并依次运行以下两个命令：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

验证适用于 Ubuntu Server 的 CodeDeploy 代理是否正在运行

要查看 CodeDeploy 代理是否已安装且正在运行，请登录到相应的实例，并运行以下命令：

```
systemctl status codedeploy-agent
```

如果命令返回错误，则没有安装 CodeDeploy 代理。请按照安装适用于 Ubuntu Server 的 CodeDeploy 代理中所述进行安装。

如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于 `The AWS CodeDeploy agent is running` 的消息。

如果您看到类似于 `error: No AWS CodeDeploy agent running` 的消息，请启动该服务并依次运行以下两个命令：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

验证适用于 Windows Server 的 CodeDeploy 代理是否正在运行

要查看 CodeDeploy 代理是否已安装且正在运行，请登录到相应的实例，并运行以下命令：

```
powershell.exe -Command Get-Service -Name codedeployagent
```

您应该可以看到类似于如下所示的输出内容：

```
Status   Name               DisplayName
------   ----               -----------
Running codedeployagent    CodeDeploy Host Agent Service
```

如果命令返回错误，则没有安装 CodeDeploy 代理。请按照安装适用于 Windows Server 的 CodeDeploy 代理中所述进行安装。

如果 `Status` 显示除 `Running` 外的任何内容，请使用以下命令启动该服务：

```
powershell.exe -Command Start-Service -Name codedeployagent
```

您可以使用以下命令重新启动该服务：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

您可以使用以下命令停止该服务：

```
powershell.exe -Command Stop-Service -Name codedeployagent
```

确定 CodeDeploy 代理的版本

您可以通过两种方式确定在您的实例上运行的 CodeDeploy 代理的版本。

首先，从 CodeDeploy 代理版本 1.0.1.854 开始，您可以在实例上的 `.version` 文件中查看版本号。下表显示了每个受支持的操作系统的位置和示例版本字符串。

操作系统	文件位置	示例 agent_version 字符串
Amazon Linux 和 Red Hat Enterprise Linux（RHEL）	`/opt/codedeploy-agent/.version`	OFFICIAL_1.0.1.854_rpm
Ubuntu Server	`/opt/codedeploy-agent/.version`	OFFICIAL_1.0.1.854_deb
Windows Server	`C:\ProgramData\Amazon\CodeDeploy\.version`	OFFICIAL_1.0.1.854_msi

其次，您可以在实例上运行命令来确定 CodeDeploy 代理的版本。

主题

- 确定 Amazon Linux 或 RHEL 上的版本
- 确定 Ubuntu Server 上的版本
- 确定 Windows Server 上的版本

确定 Amazon Linux 或 RHEL 上的版本

登录到实例并运行以下命令：

```
sudo yum info codedeploy-agent
```

确定 Ubuntu Server 上的版本

登录到实例并运行以下命令：

```
sudo dpkg -s codedeploy-agent
```

确定 Windows Server 上的版本

登录到实例并运行以下命令：

```
sc qdescription codedeployagent
```

安装 CodeDeploy 代理

要在 EC2 实例或本地服务器上使用 CodeDeploy，必须先安装 CodeDeploy 代理。我们建议使用 Amazon Systems Manager 安装和更新 CodeDeploy 代理。有关 Systems Manager 的详细信息，请参阅什么是 Amazon Systems Manager。创建部署组时，您可以在控制台中使用 Systems Manager 设置 CodeDeploy 代理的安装和计划更新。

您还可以使用命令行直接从 S3 存储桶安装 CodeDeploy 代理。

有关要安装的推荐版本，请参阅 CodeDeploy代理的版本历史记录。

主题

- 使用 Amazon Systems Manager 安装 CodeDeploy 代理
- 使用命令行安装 CodeDeploy 代理

使用 Amazon Systems Manager 安装 CodeDeploy 代理

您可以使用 Amazon Systems Manager 通过 Amazon Web Services 管理控制台 控制台或 Amazon CLI 将 CodeDeploy 代理安装到您的 Amazon EC2 或本地实例中。您可以选择安装特定版本或选择始终安装最新版本的代理。有关 Amazon Systems Manager 的更多信息，请参阅什么是 Amazon Systems Manager。

安装和更新 CodeDeploy 代理的推荐方法是使用 Amazon Systems Manager。您还可以从 Amazon S3 存储桶安装 CodeDeploy 代理。有关使用 Amazon S3 下载链接的信息，请参阅 使用命令行安装 CodeDeploy 代理。

主题

- 先决条件
- 安装 CodeDeploy 代理

先决条件

按照 入门 CodeDeploy 中的步骤设置 IAM 权限和 Amazon CLI。

如果使用 Systems Manager 在本地服务器上安装 CodeDeploy 代理，则必须向 Amazon EC2 Systems Manager 注册本地服务器。有关更多信息，请参阅《Amazon Systems Manager 用户指南》中的在混合环境中设置 Systems Manager。

安装 CodeDeploy 代理

您必须确保为 Systems Manager 正确配置实例，然后才能使用 Systems Manager 安装 CodeDeploy 代理。

安装或更新 SSM Agent

在 Amazon EC2 实例上，CodeDeploy 代理要求实例运行 2.3.274.0 或更高版本。在安装 CodeDeploy 代理之前，如果还没有在实例上更新或安装 SSM 代理，请进行更新或安装。

SSM 代理已预先安装在由 Amazon 提供的某些 Amazon EC2 AMI 上。有关更多信息，请参阅预装了 SSM 代理的亚马逊机器映像（AMI）。

> **Note**
>
> 确保 CodeDeploy 代理也支持实例的操作系统。有关更多信息，请参阅 CodeDeploy 代理支持的操作系统。

有关在运行 Linux 的实例上安装或更新 SSM 代理的信息，请参阅《Amazon Systems Manager 用户指南》中的在 Linux 实例上安装和配置 SSM 代理。

有关在运行 Windows 服务器的实例上安装或更新 SSM 代理的信息，请参阅《Amazon Systems Manager 用户指南》中的在 Windows 实例上安装和配置 SSM 代理。

（可选）验证 Systems Manager 的先决条件

在使用 Systems Manager Run Command 安装 CodeDeploy 代理之前，请确保您的实例满足 Systems Manager 的最低要求。有关更多信息，请参阅《Amazon Systems Manager 用户指南》中的设置 Amazon Systems Manager。

安装 CodeDeploy 代理

使用 SSM，您可以安装一次 CodeDeploy 或设置计划来安装新版本。

要安装 CodeDeploy 代理，请选择 `AWSCodeDeployAgent` 软件包，同时按照使用 Amazon Systems Manager 分发服务器安装或更新软件包中的步骤操作。

使用命令行安装 CodeDeploy 代理

> **Note**
> 我们建议使用 Amazon Systems Manager 安装 CodeDeploy 代理，以便能够配置代理的计划更新。有关更多信息，请参阅 使用 Amazon Systems Manager 安装 CodeDeploy 代理。

使用以下主题，通过命令行安装和运行 CodeDeploy 代理。

主题

- 安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理
- 安装适用于 Ubuntu Server 的 CodeDeploy 代理
- 安装适用于 Windows Server 的 CodeDeploy 代理

安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理

登录到实例，并依次运行以下命令。首先运行命令 `sudo yum update` 被认为是使用 `yum` 安装软件包的最佳做法，但如果您不想更新所有软件包，也可以跳过它。

```
sudo yum update
```

```
sudo yum install ruby
```

```
sudo yum install wget
```

（可选）要清理 AMI 中以前的代理缓存信息，请运行以下脚本：

```
#!/bin/bash
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
$CODEDEPLOY_BIN stop
yum erase codedeploy-agent -y
```

转到您的主目录：

```
cd /home/ec2-user
```

> **Note**
>
> 在前面的命令中，`/home/ec2-user` 表示 Amazon Linux 或 RHEL Amazon EC2 实例的默认用户名。如果您的实例是使用某个自定义 AMI 创建的，该 AMI 所有者可能已指定不同的默认用户名。

下载 CodeDeploy 代理安装程序：

对于中国（北京）区域：

- ```
  wget https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com/latest/install
  ```

对于中国（宁夏）区域：

- ```
  wget https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com/latest/
  install
  ```

例如：

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install`

有关存储桶名称和区域标识符的列表，请参阅各区域的资源工具包存储桶名称。

为 `install` 文件设置执行权限：

```
chmod +x ./install
```

要安装最新版本的 CodeDeploy 代理，请执行以下操作：

- ```
  sudo ./install auto
  ```

要安装特定版本的 CodeDeploy 代理，请执行以下操作：

- 列出您所在区域的可用版本：
  ```

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier
 | grep '\.rpm$'
```

- 安装以下版本之一：

```
sudo ./install auto -v releases/codedeploy-agent-version.noarch.rpm
```

> **Note**
>
> Amazon 支持 CodeDeploy 代理的最新次要版本。目前，最新的次要版本是 1.7.x。

要检查服务是否正在运行，请运行以下命令：

```
systemctl status codedeploy-agent
```

如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于 `The AWS CodeDeploy agent is running` 的消息。

如果您看到类似于 `error: No AWS CodeDeploy agent running` 的消息，请启动该服务并依次运行以下两个命令：

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

安装适用于 Ubuntu Server 的 CodeDeploy 代理

> **Note**
>
> 我们建议使用 Amazon Systems Manager 安装 CodeDeploy 代理，以便能够配置代理的计划更新。有关更多信息，请参阅 使用 Amazon Systems Manager 安装 CodeDeploy 代理。

在 Ubuntu Server 上安装 CodeDeploy 代理

1. 登录到实例。

2. 依次输入以下命令：

```
sudo apt update
```

```
sudo apt install ruby-full
```

```
sudo apt install wget
```

3. 输入以下命令：

```
cd /home/ubuntu
```

/home/ubuntu 表示 Ubuntu Server 实例的默认用户名。如果您的实例是使用某个自定义 AMI 创建的，该 AMI 所有者可能已指定不同的默认用户名。

4. 输入以下命令：

对于中国（北京）区域：

- ```
  wget https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com/latest/install
  ```

对于中国（宁夏）区域：

- ```
  wget https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com/
  latest/install
  ```

例如：

```
https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/
install
```

有关存储桶名称和区域标识符的列表，请参阅各区域的资源工具包存储桶名称。

5. 输入以下命令：

```
chmod +x ./install
```

6. 请执行以下操作之一：

- 要在除 20.04 之外 的任何受支持版本的 Ubuntu Server 上安装最新版本的 CodeDeploy 代理，请执行以下操作：

```
sudo ./install auto
```

- 要在 Ubuntu Server 20.04 上安装最新版本的 CodeDeploy 代理，请执行以下操作：

> **Note**
>
> 将输出写入临时日志文件是一种变通方法，当我们在 Ubuntu Server 20.04 上解决 `install` 脚本的一个已知错误时，应该使用这种方法。

```
sudo ./install auto > /tmp/logfile
```

- 要在除 20.04 之外 的任何受支持版本的 Ubuntu Server 上安装特定版本的 CodeDeploy 代理，请执行以下操作：
  - 列出您所在区域的可用版本：

    ```
    aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier | grep '\.deb$'
    ```

  - 安装以下版本之一：

    ```
    sudo ./install auto -v releases/codedeploy-agent-###.deb
    ```

    > **Note**
    >
    > Amazon 支持 CodeDeploy 代理的最新次要版本。目前，最新的次要版本是 1.7.x。

- 要在 Ubuntu Server 20.04 上安装特定版本的 CodeDeploy 代理，请执行以下操作：
  - 列出您所在区域的可用版本：

    ```
    aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier | grep '\.deb$'
    ```

  - 安装以下版本之一：

```
sudo ./install auto -v releases/codedeploy-agent-###.deb > /tmp/logfile
```

> **Note**
>
> 将输出写入临时日志文件是一种变通方法，当我们在 Ubuntu Server 20.04 上解决 `install` 脚本的一个已知错误时，应该使用这种方法。

> **Note**
>
> Amazon 支持 CodeDeploy 代理的最新次要版本。目前，最新的次要版本是 1.7.x。

检查服务是否正在运行

1. 输入以下命令：

   ```
   systemctl status codedeploy-agent
   ```

   如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于 `The AWS CodeDeploy agent is running` 的消息。

2. 如果您看到类似于 `error: No AWS CodeDeploy agent running` 的消息，请启动该服务并依次运行以下两个命令：

   ```
   systemctl start codedeploy-agent
   ```

   ```
   systemctl status codedeploy-agent
   ```

安装适用于 Windows Server 的 CodeDeploy 代理

在 Windows Server 实例上，您可以使用下列方法之一来下载并安装 CodeDeploy 代理：

- 使用 Amazon Systems Manager（推荐）
- 运行一系列 Windows PowerShell 命令。
- 选择直接下载链接。

- 运行 Amazon S3 复制命令。

> **Note**
>
> 安装 CodeDeploy 代理的文件夹是 `C:\Program Data\Amazon\CodeDeploy`。确保此路径上没有目录连接或符号链接。

主题

- 使用 Systems Manager
- 使用 Windows PowerShell
- 使用直接链接
- 使用 Amazon S3 复制命令

使用 Systems Manager

按照使用 Amazon Systems Manager 安装 CodeDeploy 代理中的说明安装 CodeDeploy 代理。

使用 Windows PowerShell

登录到实例，然后在 Windows PowerShell 中运行以下命令：

1. 要求从 Internet 下载的所有脚本和配置文件由可信发布者签名。如果系统提示您更改执行策略，请键入“**Y**”。

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

2. 加载 Amazon Tools for Windows PowerShell。

   ```
   Import-Module AWSPowerShell
   ```

3. 创建目录，将 CodeDeploy 代理安装文件下载到其中。

   ```
   New-Item -Path "c:\temp" -ItemType "directory" -Force
   ```

4. 使用 `Set-AWSCredential` 和 `Initialize-AWSDefaultConfiguration` 命令配置 Amazon 凭证。有关更多信息，请参阅《Amazon tools for PowerShell 用户指南》中的使用 Amazon 凭证。

5. 下载 CodeDeploy 代理安装文件。

> **Note**
>
> Amazon 支持 CodeDeploy 代理的最新次要版本。目前，最新的次要版本是 1.7.x。

对于中国（北京）区域：

要安装最新版本的 CodeDeploy 代理，请执行以下操作：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-north-1 -Key
   latest/codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
  ```

要安装特定版本的 CodeDeploy 代理，请执行以下操作：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-north-1 -Key
   releases/codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi
  ```

对于中国（宁夏）区域：

要安装最新版本的 CodeDeploy 代理，请执行以下操作：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-northwest-1 -
  Key latest/codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
  ```

要安装特定版本的 CodeDeploy 代理，请执行以下操作：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-northwest-1 -
  Key releases/codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi
  ```

6. 运行 CodeDeploy 代理安装文件。

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

要检查服务是否正在运行，请运行以下命令：

```
powershell.exe -Command Get-Service -Name codedeployagent
```

如果刚刚安装了 CodeDeploy 代理并且未启动，则在运行 Get-Service 命令之后，您应在状态下看到 **Start...**：

```
Status     Name                DisplayName
------     ----                -----------
Start...   codedeployagent    CodeDeploy Host Agent Service
```

如果 CodeDeploy 代理已经运行，则在运行 Get-Service 命令之后，您应在状态下看到 **Running**：

```
Status     Name                DisplayName
------     ----                -----------
Running    codedeployagent    CodeDeploy Host Agent Service
```

使用直接链接

如果 Windows Server 实例上的浏览器安全设置提供权限（例如，针对 `http://*.amazonaws.com` 的权限），则您可以使用直接链接来下载 CodeDeploy 代理，然后手动运行安装程序。

对于中国（北京）区域：

```
https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com.cn/latest/codedeploy-agent.msi
```

对于中国（宁夏）区域：

```
https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com.cn/latest/codedeploy-agent.msi
```

使用 Amazon S3 复制命令

如果实例上已安装 Amazon CLI，则可使用 Amazon S3 cp 命令下载 CodeDeploy 代理，然后手动运行安装程序。有关信息，请参阅在 Microsoft Windows 上安装 Amazon Command Line Interface。

对于中国（北京）区域，使用：

```
aws s3 cp s3://aws-codedeploy-cn-north-1/latest/codedeploy-agent.msi codedeploy-
agent.msi --region cn-north-1
```

对于中国（宁夏）区域，使用：

```
aws s3 cp s3://aws-codedeploy-cn-northwest-1/latest/codedeploy-agent.msi codedeploy-
agent.msi --region cn-northwest-1
```

更新 CodeDeploy 代理

您可以使用 Amazon Systems Manager 在所有受支持的操作系统上配置 CodeDeploy 代理的自动计划更新。您也可以通过在实例上运行命令来在所有受支持的操作系统上进行强制更新。

主题

- 更新 Amazon Linux 或 RHEL 上的 CodeDeploy 代理
- 更新 Ubuntu Server 上的 CodeDeploy 代理
- 更新 Windows Server 上的 CodeDeploy 代理

更新 Amazon Linux 或 RHEL 上的 CodeDeploy 代理

要使用 Amazon Systems Manager 配置 CodeDeploy 代理的自动计划更新，请按照使用 Amazon Systems Manager 安装 CodeDeploy 代理中的步骤操作。

如果您要强制更新 CodeDeploy 代理，请登录到实例，并运行以下命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Ubuntu Server 上的 CodeDeploy 代理

要使用 Amazon Systems Manager 配置 CodeDeploy 代理的自动计划更新，请按照使用 Amazon Systems Manager 安装 CodeDeploy 代理中的步骤操作。

如果您要强制更新 CodeDeploy 代理，请登录到实例，并运行以下命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Windows Server 上的 CodeDeploy 代理

您可以使用 Amazon Systems Manager 启用 CodeDeploy 代理的自动更新。通过使用 Systems Manager，您可以通过创建与 Systems Manager 状态管理器的关联来为您的 Amazon EC2 或本地实例配置更新计划。您还可以通过卸载当前版本并安装较新版本来手动更新 CodeDeploy 代理。

主题

- 使用 Amazon Systems Manager 设置自动 CodeDeploy 代理更新
- 手动更新 CodeDeploy 代理
- （已弃用）使用 Windows Server 更新程序更新 CodeDeploy 代理

使用 Amazon Systems Manager 设置自动 CodeDeploy 代理更新

要配置 Systems Manager 并启用 CodeDeploy 代理的自动更新，请按照使用 Amazon Systems Manager 安装 CodeDeploy 代理中的说明进行操作。

手动更新 CodeDeploy 代理

要手动更新 CodeDeploy 代理，您可以从 CLI 或使用 Systems Manager 安装最新版本。按照安装 CodeDeploy 代理中的说明进行操作。建议您按照卸载 CodeDeploy 代理中的说明卸载旧版本的 CodeDeploy 代理。

（已弃用）使用 Windows Server 更新程序更新 CodeDeploy 代理

> **Note**
>
> 适用于 Windows Server 的 CodeDeploy 代理更新程序已弃用，并且不会更新到 1.0.1.1597 之后的任何版本。

要启用自动更新 CodeDeploy 代理的功能，请在新实例或现有实例上安装适用于 Windows Server 的 CodeDeploy 代理更新程序。更新程序定期检查新版本。当检测到新版本时，更新程序将在安装最新版本之前，卸载当前版本的代理（如果已安装）。

如果在更新程序检测到新版本时部署操作已在进行中，则部署操作将会继续完成。如果尝试在更新过程中启动部署操作，则部署操作将失败。

如果您希望强制更新 CodeDeploy 代理，请按照安装适用于 Windows Server 的 CodeDeploy 代理中的说明操作。

在 Windows Server 实例上，您可以通过以下方式下载并安装 CodeDeploy 代理更新程序：运行 Windows PowerShell 命令、使用直接下载链接或运行 Amazon S3 复制命令。

主题

- 使用 Windows PowerShell
- 使用直接链接
- 使用 Amazon S3 复制命令

使用 Windows PowerShell

登录到实例，然后在 Windows PowerShell 中依次运行以下命令：

```
Set-ExecutionPolicy RemoteSigned
```

如果系统提示您更改执行策略，请选择 **Y**，这样 Windows PowerShell 要求从 Internet 下载的所有脚本和配置文件由可信发布者签名。

```
Import-Module AWSPowerShell
```

```
New-Item -Path "c:\temp" -ItemType "directory" -Force
```

对于中国（北京）区域：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-north-1 -Key
   latest/codedeploy-agent-updater.msi -File c:\temp\codedeploy-agent-updater.msi
  ```

对于中国（宁夏）区域：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-northwest-1 -Key
   latest/codedeploy-agent-updater.msi -File c:\temp\codedeploy-agent-updater.msi
  ```

```
c:\temp\codedeploy-agent-updater.msi /quiet /l c:\temp\host-agent-updater-log.txt
```

```
powershell.exe -Command Get-Service -Name codedeployagent
```

如果需要排除更新过程中出现的错误，请键入以下命令来打开 CodeDeploy 代理更新程序日志文件：

```
notepad C:\ProgramData\Amazon\CodeDeployUpdater\log\codedeploy-agent.updater.log
```

使用直接链接

如果 Windows Server 实例上的浏览器安全设置提供必需的权限（例如，针对 `http://*.amazonaws.com` 的权限），则您可以使用直接链接来下载 CodeDeploy 代理更新程序。然后，手动运行安装程序。

对于中国（北京）区域，使用：

```
https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com.cn/latest/codedeploy-agent-updater.msi
```

对于中国（宁夏）区域，使用：

```
https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com.cn/latest/codedeploy-agent-updater.msi
```

使用 Amazon S3 复制命令

如果实例上已安装 Amazon CLI，则可使用 Amazon S3 cp 命令下载 CodeDeploy 代理更新程序，然后手动运行安装程序。有关信息，请参阅在 Microsoft Windows 上安装 Amazon Command Line Interface。

对于中国（北京）区域，使用：

```
aws s3 cp s3://s3-cn-north-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi --region cn-north-1
```

对于中国（宁夏）区域，使用：

```
aws s3 cp s3://s3-cn-northwest-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi --region cn-northwest-1
```

卸载 CodeDeploy 代理

当您不再需要 CodeDeploy 代理或者希望执行全新安装时，可以从实例中移除该代理。

从 Amazon Linux 或 RHEL 卸载 CodeDeploy 代理

要卸载 CodeDeploy 代理，请登录到实例并运行以下命令：

```
sudo yum erase codedeploy-agent
```

从 Ubuntu Server 卸载 CodeDeploy 代理

要卸载 CodeDeploy 代理，请登录到实例并运行以下命令：

```
sudo dpkg --purge codedeploy-agent
```

从 Windows Server 卸载 CodeDeploy 代理

要卸载 CodeDeploy 代理，请登录到实例并依次运行以下三个命令：

```
wmic
```

```
product where name="CodeDeploy Host Agent" call uninstall /nointeractive
```

```
exit
```

您还可以登录到实例，在控制面板中，打开程序和功能，选择 CodeDeploy 主机代理，然后选择卸载。

将 CodeDeploy 代理日志发送到 CloudWatch

您可以使用统一的 CloudWatch 代理（更简单地说 CloudWatch 代理）将 CodeDeploy 代理的指标和日志数据发送到 CloudWatch。

按照以下说明安装 CloudWatch 代理并将其配置为与 CodeDeploy 代理一起使用。

先决条件

开始之前，完成以下任务：

- 安装 CodeDeploy 代理并确保其正在运行。有关更多信息，请参阅安装 CodeDeploy 代理和验证 CodeDeploy 代理是否正在运行。
- 安装 CloudWatch 代理。有关更多信息，请参阅安装 CloudWatch 代理。
- 向 CodeDeploy IAM 实例配置文件添加以下权限：

- CloudWatchLogsFullAccess
- CloudWatchAgentServerPolicy

有关 CodeDeploy 实例配置文件的更多信息，请参阅入门 CodeDeploy的步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

配置 CloudWatch 代理以收集 CodeDeploy 代理

您可以通过逐步执行向导或手动创建或编辑配置文件来配置 CloudWatch 代理。

使用向导配置 CloudWatch 代理（Linux）

1. 按照运行 CloudWatch 代理配置向导中所述运行向导。
2. 在向导中，当系统询问 `Do you want to monitor any log files?` 时，输入 **1**。
3. 按如下方式指定 CodeDeploy 代理日志文件：
   1. 对于 `Log file path`，输入 CodeDeploy 日志文件的路径，例如：**/var/log/aws/codedeploy-agent/codedeploy-agent.log**。
   2. 对于 `Log group name`，输入日志组名称，例如：**codedeploy-agent-log**。
   3. 对于 `Log stream name`，输入日志流名称，例如：**{instance_id}-codedeploy-agent-log**。
4. 当系统询问 `Do you want to specify any additional log files?` 时，请输入 **1**。
5. 按如下方式指定 CodeDeploy 代理部署日志：
   1. 对于 `Log file path`，输入 CodeDeploy 部署日志文件的路径，例如：**/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log**。
   2. 对于 `Log group name`，输入日志组名称，例如：**codedeploy-agent-deployment-log**。
   3. 对于 `Log stream name`，输入日志流名称，例如：**{instance_id}-codedeploy-agent-deployment-log**。
6. 当系统询问 `Do you want to specify any additional log files?` 时，请输入 **1**。
7. 按如下方式指定 CodeDeploy 代理更新程序日志：
   1. 对于 `Log file path`，输入 CodeDeploy 更新程序日志文件的路径，例如：**/tmp/codedeploy-agent.update.log**。

2. 对于 `Log group name`，输入日志组名称，例如：**`codedeploy-agent-updater-log`**。
3. 对于 `Log stream name`，输入日志流名称，例如：**`{instance_id}-codedeploy-agent-updater-log`**。

使用向导配置 CloudWatch 代理（Windows）

1. 按照运行 CloudWatch 代理配置向导中所述运行向导。
2. 在向导中，当系统询问 `Do you want to monitor any customized log files?` 时，输入 **`1`**。
3. 按如下方式指定 CodeDeploy 日志文件：
   1. 对于 `Log file path`，输入 CodeDeploy 代理日志文件的路径，例如：**`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt`**。
   2. 对于 `Log group name`，输入日志组名称，例如：**`codedeploy-agent-log`**。
   3. 对于 `Log stream name`，输入日志流名称，例如：**`{instance_id}-codedeploy-agent-log`**。
4. 当系统询问 `Do you want to specify any additional log files?` 时，请输入 **`1`**。
5. 按如下方式指定 CodeDeploy 代理部署日志：
   1. 对于 `Log file path`，输入 CodeDeploy 部署日志文件的路径，例如：**`C:\ProgramData\Amazon\CodeDeploy\deployment-logs\codedeploy-agent-deployments.log`**。
   2. 对于 `Log group name`，输入日志组名称，例如：**`codedeploy-agent-deployment-log`**。
   3. 对于 `Log stream name`，输入日志流名称，例如：**`{instance_id}-codedeploy-agent-deployment-log`**。

通过手动创建或编辑配置文件来配置 CloudWatch 代理（Linux）

1. 按照手动创建或编辑 CloudWatch 代理配置文件中所述，创建或编辑 CloudWatch 代理配置文件。
2. 确保该文件名为 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`，并且它包含以下代码：

```
...
"logs": {
    "logs_collected": {
```

```
        "files": {
            "collect_list": [
                {
                    "file_path": "/var/log/aws/codedeploy-agent/codedeploy-
agent.log",
                    "log_group_name": "codedeploy-agent-log",
                    "log_stream_name": "{instance_id}-agent-log"
                },
                {
                    "file_path": "/opt/codedeploy-agent/deployment-root/deployment-
logs/codedeploy-agent-deployments.log",
                    "log_group_name": "codedeploy-agent-deployment-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-deployment-
log"
                },
                {
                    "file_path": "/tmp/codedeploy-agent.update.log",
                    "log_group_name": "codedeploy-agent-updater-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-updater-log"
                }
            ]
        }
    }
}
...
```

通过手动创建或编辑配置文件来配置 CloudWatch 代理（Windows）

1. 按照手动创建或编辑 CloudWatch 代理配置文件中所述，创建或编辑 CloudWatch 代理配置文件。
2. 确保该文件名为 `C:\ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`，并且它包含以下代码：

```
...
"logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\\log\
\codedeploy-agent-log.txt",
                        "log_group_name": "codedeploy-agent-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-log"
```

```
                    },
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\
\deployment-logs\\codedeploy-agent-deployments.log",
                        "log_group_name": "codedeploy-agent-deployment-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-
deployment-log"
                    }
                ]
            },
            ...
        }
    },
...
```

重新启动 CloudWatch 代理

进行更改后，按照启动 CloudWatch 代理中所述重新启动 CloudWatch 代理。

使用适用于 CodeDeploy 的实例

CodeDeploy 支持部署到运行 Amazon Linux、Ubuntu Server、Red Hat Enterprise（RHEL）和 Windows Server 的实例。

您可以使用 CodeDeploy 部署到 Amazon EC2 实例和本地实例。本地实例是可运行 CodeDeploy 代理并连接到公有 Amazon 服务终端节点的 Amazon EC2 实例以外的任何物理设备。您可以使用 CodeDeploy 将应用程序同时部署到云中的 Amazon EC2 实例、办公室的台式电脑或您自己的数据中心的服务器上。

将 Amazon EC2 实例与本地实例进行比较

下表将 Amazon EC2 实例和本地实例进行比较：

主题	Amazon EC2 实例	本地实例
需要您安装并运行可与实例上运行的操作系统兼容的 CodeDeploy 代理的版本。	支持	是
需要实例能够连接到 CodeDeploy 服务。	支持	是
需要将 IAM 实例配置文件附加到实例。IAM 实例配置文件必须有权参与 CodeDeploy 部署。有关信息，请参阅步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。	是	否
要对实例进行身份验证和注册，您需要执行以下操作之一： • 创建一个可让 IAM 用户在每个实例上都能担任的 IAM 角色，以获取通过 Amazon	否	是

主题	Amazon EC2 实例	本地实例
Security Token Service 生成的定期刷新的临时凭证。 • 为每个实例创建一个 IAM 用户，并以纯文本形式将该 IAM 用户的账户凭证存储在实例上。		
需要您先向 CodeDeploy 注册每个实例，然后才能对其进行部署。	否	是
需要您先标记每个实例，然后 CodeDeploy 才能对其进行部署。	支持	是
可以在 CodeDeploy 部署中参与 Amazon EC2 Auto Scaling 和 Elastic Load Balancing 场景。	是	否
可从 Amazon S3 存储桶和 GitHub 存储库进行部署。	支持	是
可支持在部署或实例中发生指定事件时提示发送 SMS 或电子邮件通知的触发器。	支持	是
可能要收取相关部署费用。	否	是

CodeDeploy 的实例任务

要启动或配置在部署中使用的实例，请从以下说明中进行选择：

我想启动一个新的 Amazon Linux 或 Windows Server Amazon EC2 实例。	要以最少的工作量启动 Amazon EC2 实例，请参阅为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）。

	要主要靠自己手动启动 Amazon EC2 实例，请参阅为 CodeDeploy 创建 Amazon EC2 实例（Amazon CLI 或 Amazon EC2 控制台）。
我想启动一个新的 Ubuntu Server 或 RHEL Amazon EC2 实例。	请参阅为 CodeDeploy 创建 Amazon EC2 实例（Amazon CLI 或 Amazon EC2 控制台）。
我想配置 Amazon Linux、Windows Server、Ubuntu Server 或 RHEL Amazon EC2 实例。	请参阅配置要 EC2 使用的亚马逊实例 CodeDeploy。
我要配置 Windows Server、Ubuntu Server 或 RHEL 本地实例（不是 Amazon EC2 实例的物理设备）。	请参阅Working with On-Premises Instances。
我希望 CodeDeploy 在蓝绿部署期间预置实例集。	请参阅在中处理部署 CodeDeploy。

要在 Amazon EC2 Auto Scaling 组中准备 Amazon EC2 实例，您必须执行一些额外步骤。有关更多信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

主题

- Tagging Instances for Deployments
- Working with Amazon EC2 Instances
- Working with On-Premises Instances
- View Instance Details
- Instance Health

在中为部署组标记实例 CodeDeploy

为了帮助管理您的 Amazon EC2 实例和本地实例，您可以使用标签为每个资源分配自己的元数据。您可使用标签按各种标准（例如用途、所有者或环境）对实例进行分类。如果您拥有许多实例，这种方法就很有用。您可以根据为实例分配的标签快速确定某个实例或一组实例。每个标签都包含定义的一个键和一个可选值。有关更多信息，请参阅标记您的 Amazon EC2 资源。

要指定 CodeDeploy 部署组中包含哪些实例，可以在一个或多个标签组中指定标签。针对部署组创建部署时，将在满足标签条件的实例上安装最新的应用程序修订。

> **Note**
>
> 您也可以将 Amazon A EC2 uto Scaling 组包含在部署组中，但这些组是通过名称而不是应用于实例的标签来标识的。有关信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

部署组中最简单的实例条件可以是唯一标签组中的唯一标签。复杂的条件可以在最多三个标签组中每组添加 10 个标签。

如果使用唯一标签组，该组中至少一个标签标记的实例即会包括在部署组中。如果使用多个标签组，只有由每个 标签组中至少一个标签标记的实例才会包括在内。

以下示例说明如何使用标签和标签组为部署组选择实例。

主题

- 示例 1：唯一标签组，唯一标签
- 示例 2：唯一标签组，多个标签
- 示例 3：多个标签组，单个标签
- 示例 4：多个标签组，多个标签

示例 1：唯一标签组，唯一标签

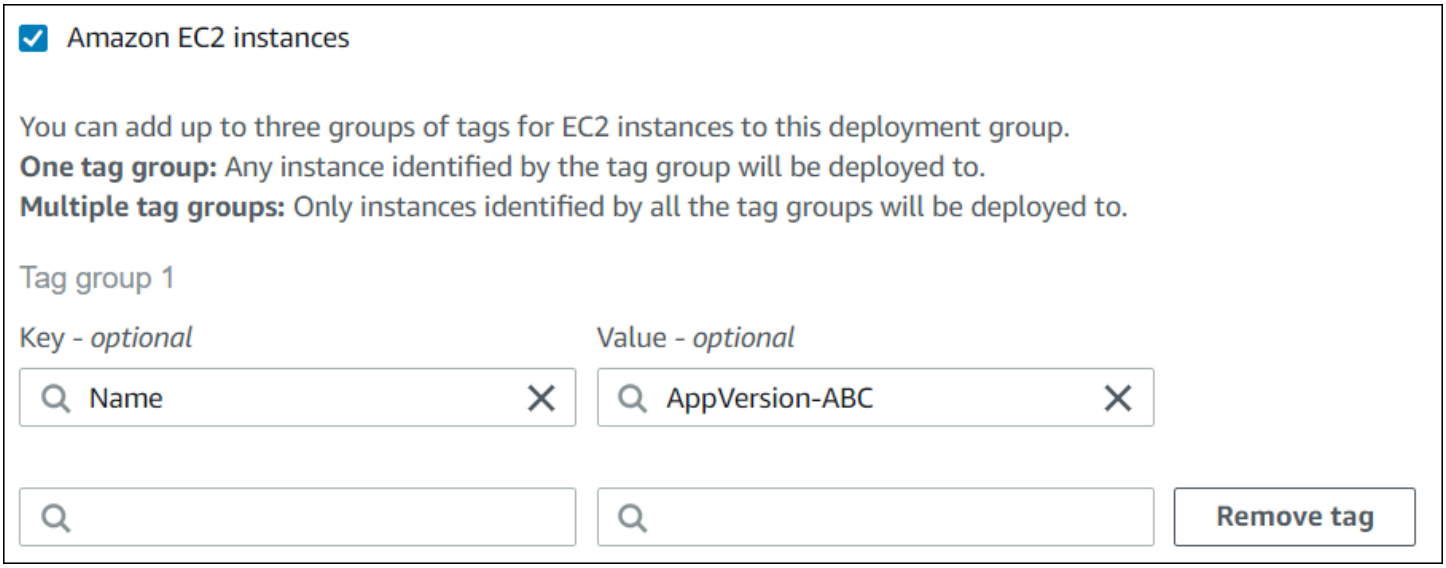
您可以在唯一标签组中指定唯一标签：

标签组 1

键	值
Name	AppVersion-美国广播公司

具有 `Name=AppVersion-ABC` 标签的每个实例都是部署组中的一部分，即使它也应用了其他标签。

CodeDeploy 控制台设置视图：



JSON 结构：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ]
         ]
      },
```

示例 2：唯一标签组，多个标签

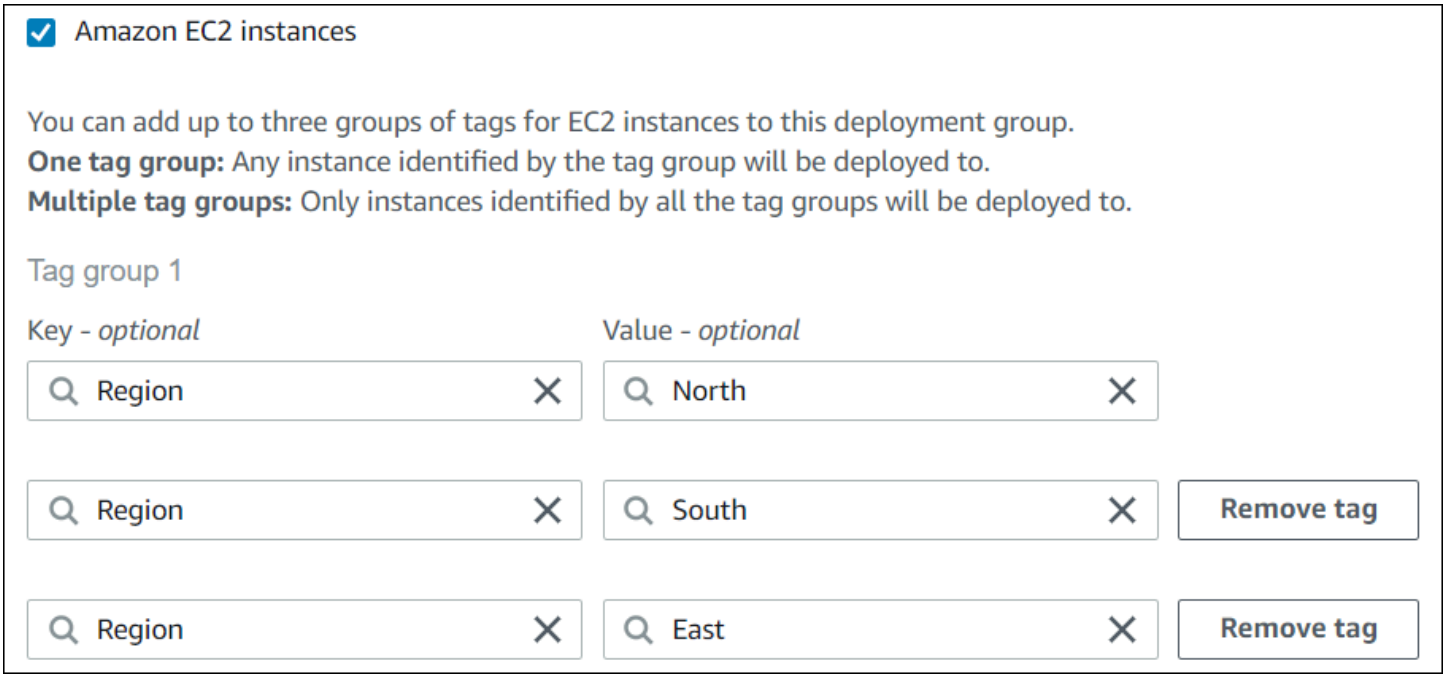
您也可以在唯一标签组中指定多个标签：

标签组 1

键	值
Region	北部
Region	南部
Region	东部

具有这三个标签之一的实例即属于部署组，即使它也应用了其他标签。例如，如果有其他实例具有 `Region=West` 标签，这些实例将不会包含在部署组中。

CodeDeploy 控制台设置视图：



JSON 结构：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
                              {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
                              {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ]
```

```
        ]
    },
```

示例 3：多个标签组，单个标签

您还可以使用多组标签组，每个标签组中只有唯一的键值对，指定部署组中实例的条件。如果在部署组中使用多个标签组，只有所有标签组均标记出的实例才会包含在部署组中。

标签组 1

键	值
Name	AppVersion-美国广播公司

标签组 2

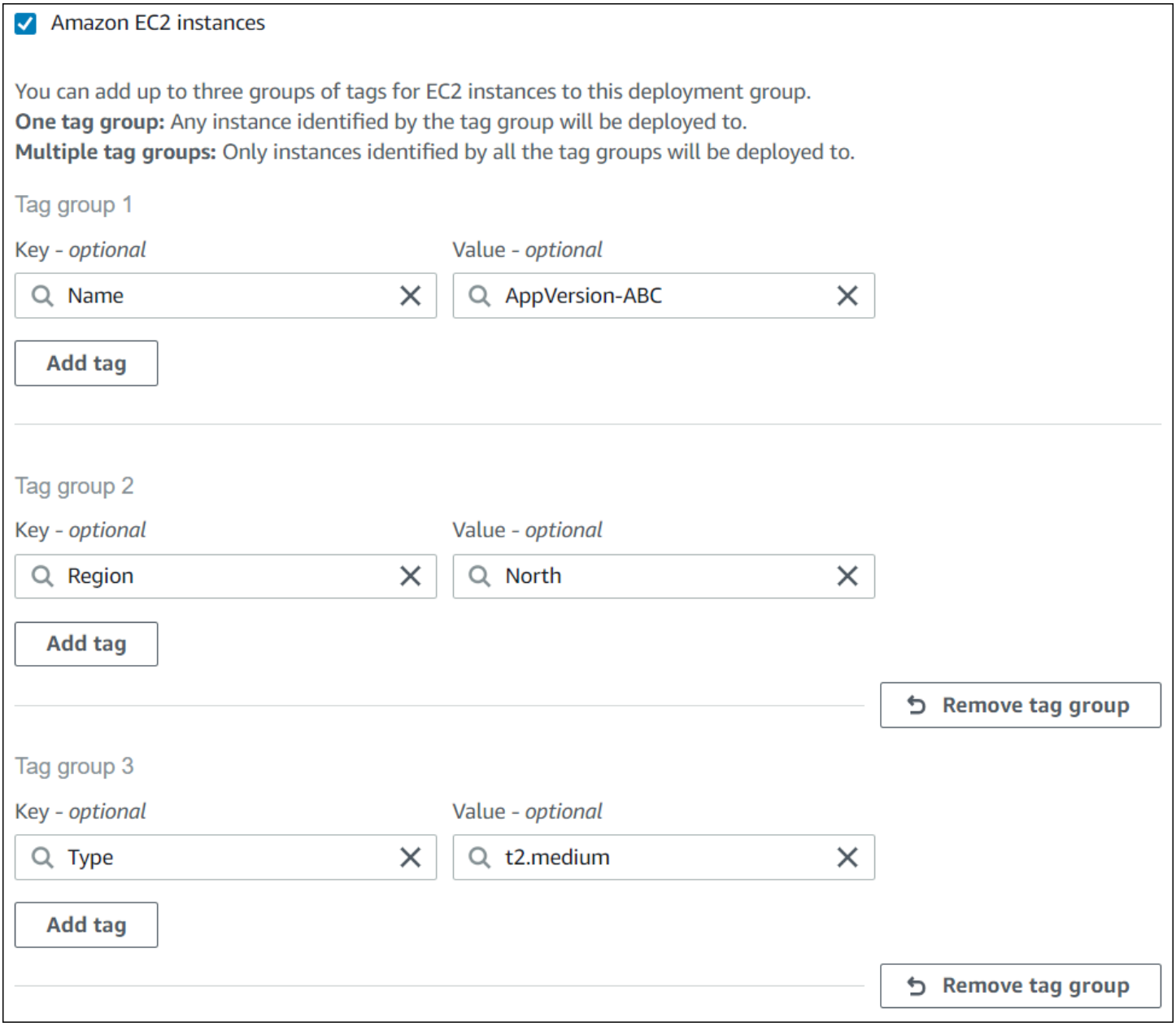
键	值
Region	北部

标签组 3

键	值
Type	t2.medium

可能在许多区域中都有具有 `Name=AppVersion-ABC` 标签的各种实例类型的实例。在此示例中，部署组中只包含也具有 `Region=North` 和 `Type=t2.medium` 标签的实例。

CodeDeploy 控制台设置视图：



JSON 结构：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ],
```

```
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               }
            ],
         ]
      },
```

示例 4：多个标签组，多个标签

如果使用多个标签组，一个或多个组中具有多个标签，实例必须与每个组中的至少一个标签匹配。

标签组 1

键	值
环境	测试版
环境	生产前调试

标签组 2

键	值
Region	北部
Region	南部
Region	东部

标签组 3

键	值
Type	t2.medium
Type	t2.large

在此示例中，实例要包含在部署组中，必须具有（1）`Environment=Beta` 或 `Environment=Staging` 标签；（2）`Region=North`、`Region=South` 或 `Region=East` 标签；以及（3）`Type=t2.medium` 或 `Type=t2.large` 标签。

例如，具有以下标签组的实例将会 包含在部署组中：

- `Environment=Beta`, `Region=North`,`Type=t2.medium`
- `Environment=Staging`,`Region=East`,`Type=t2.large`
- `Environment=Staging`,`Region=South`,`Type=t2.large`

具有以下标签组的实例不会 包含在部署组中。突出显示的键值使这些实例被排除在外：

- `Environment=Beta`，区域 =西部，`Type=t2.medium`
- `Environment=Staging`，`Region=East`，类型 =t2.micro
- 环境 =生产，`Region=South`，`Type=t2.large`

CodeDeploy 控制台设置视图：

Amazon EC2 instances

You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key - *optional*	Value - *optional*	
Environment	Staging	
Environment	Beta	Remove tag

Add tag

Tag group 2

Key - *optional*	Value - *optional*	
Region	South	
Region	North	Remove tag
Region	East	Remove tag

Add tag

Remove tag group

Tag group 3

Key - *optional*	Value - *optional*	
Type	t2.medium	
Type	t2.large	Remove tag

Add tag

Remove tag group

JSON 结构：

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Beta"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Staging"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.large"
               }
```

```
            ],
        ]
    },
```

使用以下项的 Amazon EC2 实例 CodeDeploy

Amazon EC2 实例是您使用亚马逊弹性计算云创建和配置的虚拟计算环境。Amazon 在 Amazon 云端 EC2 提供可扩展的计算容量。您可以根据 CodeDeploy 部署需要 EC2 ，使用 Amazon 启动任意数量或数量的虚拟服务器。

有关亚马逊的更多信息 EC2，请参阅《亚马逊 EC2 入门指南》。

本节中的说明向您展示如何创建和配置用于 CodeDeploy 部署的 Amazon EC2 实例。

主题

- 为 CodeDeploy 创建 Amazon EC2 实例（Amazon CLI 或 Amazon EC2 控制台）
- 为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）
- 配置要 EC2 使用的亚马逊实例 CodeDeploy

为 CodeDeploy 创建 Amazon EC2 实例（Amazon CLI 或 Amazon EC2 控制台）

这些说明向您介绍如何启动配置为在 CodeDeploy 部署中使用的新 Amazon EC2 实例。

您可使用 Amazon CloudFormation 模板启动已配置为在 CodeDeploy 部署中使用的运行 Amazon Linux 或 Windows Server 的 Amazon EC2 实例。我们不为运行 Ubuntu Server 或 Red Hat Enterprise Linux（RHEL）的 Amazon EC2 实例提供 Amazon CloudFormation 模板。有关模板使用的替代方法，请参阅使用适用于 CodeDeploy 的实例。

您可以使用 Amazon EC2 控制台、Amazon CLI 或 Amazon EC2 API 启动 Amazon EC2 实例。

启动 Amazon EC2 实例（控制台）

先决条件

如果您尚未这样做，请按照入门 CodeDeploy中的说明进行操作来设置和配置 Amazon CLI 并创建 IAM 实例配置文件。

启动 Amazon EC2 实例

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在导航窗格中，选择 Instances（实例），然后选择 Launch Instance（启动实例）。
3. 在 Step 1: Choose an Amazon Machine Image（AMI）（步骤 1: 选择 Amazon 系统映像（AMI））页上，从 Quick Start（快速启动）选项卡中，找到要使用的操作系统和版本，然后选择 Select（选择）。您必须选择一个由 CodeDeploy 支持的 Amazon EC2 AMI 操作系统。有关更多信息，请参阅 CodeDeploy 代理支持的操作系统。
4. 在步骤 2: 选择一个实例类型页上，选择任何可用的 Amazon EC2 实例类型，然后选择下一步: 配置实例详细信息。
5. 在步骤 3：配置实例详细信息页上的 IAM 角色列表中，选择您在步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件中创建的 IAM 实例角色。如果您使用建议的角色名称，则选择 **`CodeDeployDemo-EC2-Instance-Profile`**。如果您创建了自己的角色名称，请选择该名称。

   > **Note**
   > 如果网络列表中未显示默认虚拟私有云（VPC），则必须选择或创建 Amazon VPC 和子网。选择新建 VPC 和/或新建子网。有关更多信息，请参阅您的 VPC 和子网。

6. 选择下一步：添加存储。
7. 将 Step 4: Add Storage（步骤 4: 添加存储）页保持不变，然后选择 Next: Add Tags（下一步: 添加标签）。
8. 在 Step 5: Add Tags（步骤 5: 添加标记）页面上，选择 Add Tag（添加标记）。
9. 在 Key（键）框中，键入 **`Name`**。在 Value（值）框中，键入 **`CodeDeployDemo`**。

   > **Important**
   > Key（键）和 Value（值）框的内容是区分大小写的。

10. 选择 Next: Configure Security Group。
11. 在 Step 6: Configure Security Group（步骤 6: 配置安全组）页上，将 Create a new security group（创建新安全组）选项保持选中状态。

    将为运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例配置默认 SSH 角色。将为运行 Windows Server 的 Amazon EC2 实例配置默认 RDP 角色。

12. 如果您要打开 HTTP 端口，请选择 Add Rule（添加规则）按钮，然后从 Type（类型）下拉列表中，选择 **HTTP**。接受 Source（源）默认值 Custom 0.0.0.0/0（自定义 0.0.0.0/0），然后选择 Review and Launch（审核和启动）。

> **Note**
>
> 在生产环境中，建议您将访问限制为 SSH、RDP 和 HTTP 端口，而不是指定任何位置 0.0.0.0/0。CodeDeploy 不需要无限制的端口访问，也不需要 HTTP 访问。有关更多信息，请参阅有关保护您的 Amazon EC2 实例的提示。

如果 Boot from General Purpose（SSD）（由通用（SSD）启动）对话框出现，请遵循说明，然后选择 Next（下一步）。

13. 将 Step 7: Review Instance Launch（步骤 7: 查看实例启动）页保持不变，然后选择 Launch（启动）。
14. 在 Select an existing key pair or create a new key pair（选择现有密钥对或创建新密钥对）对话框中，选择 Choose an existing key pair（选择现有密钥对）或 Create a new key pair（创建新密钥对）。如果您已配置 Amazon EC2 实例密钥对，则可在此处选择它。

    如果您尚未拥有 Amazon EC2 实例密钥对，请选择创建新密钥对并为密钥对指定一个可识别的名称。选择下载密钥对以将 Amazon EC2 实例密钥对下载到您的计算机。

> **Important**
>
> 如果您需要使用 SSH 或 RDP 访问您的 Amazon EC2 实例，则必须具有密钥对。

15. 选择启动新实例。
16. 选择 Amazon EC2 实例的 ID。在实例启动并通过所有检查之前，请不要继续。

安装 CodeDeploy 代理

在 CodeDeploy 部署中使用 CodeDeploy 代理之前，必须先在您的 Amazon EC2 实例上安装该代理。有关更多信息，请参阅 安装 CodeDeploy 代理。

> **Note**
>
> 在控制台中创建部署组时，可以配置 CodeDeploy 代理的自动安装和更新。

启动 Amazon EC2 实例（CLI）

先决条件

如果您尚未这样做，请按照入门 CodeDeploy中的说明进行操作来设置和配置 Amazon CLI 并创建 IAM 实例配置文件。

启动 Amazon EC2 实例

1. 仅针对 Windows Server 如果您要创建运行 Windows Server 的 Amazon EC2 实例，请调用 create-security-group 和 authorize-security-group-ingress 命令来创建允许 RDP 访问（默认情况下不允许）或者 HTTP 访问的安全组。例如，要创建名为 CodeDeployDemo-Windows-Security-Group 的安全组，请逐条运行以下命令：

```
aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-Group --
description "For launching Windows Server images for use with CodeDeploy"
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port
 3389
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 80
```

> **Note**
>
> 在演示中，这些命令将创建一个安全组来允许通过端口 3389 的 RDP 无限制访问或通过端口 80 的 HTTP 无限制访问。作为最佳实践，建议您限制对 RDP 和 HTTP 端口的访问。CodeDeploy 不需要无限制的端口访问，也不需要 HTTP 访问。有关更多信息，请参阅有关保护您的 Amazon EC2 实例的提示。

2. 调用 run-instances 命令以创建并启动 Amazon EC2 实例。

   在您调用此命令之前，您需要收集以下内容：

   - 将用于此实例的 Amazon 系统映像（AMI）的 ID（*ami-id*）。要获取此 ID，请参阅查找合适的 AMI。
   - 您将创建的 Amazon EC2 实例的类型（*instance-type*）的名称，如 `t1.micro`。有关列表，请参阅 Amazon EC2 实例类型。

- IAM 实例配置文件的名称，该文件有权访问 Amazon S3 存储桶，其中存储着您的区域的 CodeDeploy 代理安装文件。

  有关创建 IAM 实例配置文件的信息，请参阅步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

- Amazon EC2 实例密钥对（*key-name*）的名称，用于允许通过 SSH 访问运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例，或者通过 RDP 访问运行 Windows Server 的 Amazon EC2 实例。

  > **Important**
  >
  > 仅键入密钥对名称而不是密钥对文件扩展名。例如，my-keypair，而不是 my-keypair.pem。

  要查找密钥对名称，请打开 Amazon EC2 控制台，网址为 https://console.aws.amazon.com/ec2。在导航窗格中，在 Network & Security（网络和安全）下，选择 Key Pairs（密钥对），然后记下列表中的密钥对名称。

  要生成密钥对，请参阅使用 Amazon EC2 创建您的密钥对。请务必在《Amazon Web Services 一般参考》的区域和终端节点中列出的其中一个区域中创建密钥对。否则，您无法将 Amazon EC2 实例密钥对与 CodeDeploy 结合使用。

对于 Amazon Linux、RHEL 和 Ubuntu Server

旨在调用 run-instances 命令以启动运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例并附加您在步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件中创建的 IAM 实例配置文件。例如：

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --count 1 \
  --instance-type instance-type \
  --iam-instance-profile Name=iam-instance-profile
```

> **Note**
>
> 此命令将为 Amazon EC2 实例创建一个允许访问多个端口的默认安全组，包括通过端口 22 的 SSH 无限制访问或通过端口 80 的 HTTP 无限制访问。作为最佳实践，建议您只限制对 SSH 和 HTTP 端口的访问。CodeDeploy 不需要不受限制的端口访问，也不需要 HTTP 端口访问权限。有关更多信息，请参阅有关保护您的 Amazon EC2 实例的提示。

对于 Windows Server

旨在调用 run-instances 命令以启动运行 Windows Server 的 Amazon EC2 实例、附加您在步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件中创建的 IAM 实例配置文件，并指定您在步骤 1 中创建的安全组的名称。例如：

```
aws ec2 run-instances --image-id ami-id --key-name key-name --count 1 --instance-
type instance-type --iam-instance-profile Name=iam-instance-profile --security-
groups CodeDeploy-Windows-Security-Group
```

这些命令将使用指定的 AMI 实例配置文件启动一个具有指定的 AMI、密钥对和实例类型的 Amazon EC2 实例，并在启动过程中运行指定脚本。

3. 记下输出中的 `InstanceID` 的值。如果您忘记此值，可稍后通过对 Amazon EC2 实例密钥对调用 describe-instances 命令来获取它。

```
aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
 "Reservations[*].Instances[*].[InstanceId]" --output text
```

使用实例 ID 调用 create-tags 命令，这将标记 Amazon EC2 实例，以便 CodeDeploy 在部署过程中找到它。在以下示例中，标签名为 **`CodeDeployDemo`**，但您可指定所需的任何 Amazon EC2 实例标签。

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=CodeDeployDemo
```

您可以为一个实例同时应用多个标签。例如：

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance
 Key=Region,Value=West Key=Environment,Value=Beta
```

要验证 Amazon EC2 实例是否已启动并通过所有检查，请使用实例 ID 调用 describe-instance-status 命令。

```
aws ec2 describe-instance-status --instance-ids instance-id --query
 "InstanceStatuses[*].InstanceStatus.[Status]" --output text
```

如果实例已启动并通过所有检查，输出中将显示 ok。

安装 CodeDeploy 代理

在 CodeDeploy 部署中使用 CodeDeploy 代理之前，必须先在您的 Amazon EC2 实例上安装该代理。有关更多信息，请参阅 安装 CodeDeploy 代理。

> **Note**
> 在控制台中创建部署组时，可以配置 CodeDeploy 代理的自动安装和更新。

为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）

您可使用 Amazon CloudFormation 模板快速启动运行 Amazon Linux 或 Windows Server 的 Amazon EC2 实例。您可使用 Amazon CLI、CodeDeploy 控制台或 Amazon API 通过模板启动实例。除了启动实例之外，模板还可用于：

- 指示 Amazon CloudFormation 为实例提供参与 CodeDeploy 部署的权限。
- 标记实例，以便 CodeDeploy 可在部署过程中找到它。
- 安装 CodeDeploy 代理并在实例上运行它。

您无需使用 Amazon CloudFormation 即可设置 Amazon EC2 实例。有关替代方法，请参阅使用适用于 CodeDeploy 的实例。

我们不为运行 Ubuntu Server 或 Red Hat Enterprise Linux（RHEL）的 Amazon EC2 实例提供 Amazon CloudFormation 模板。

主题

- 开始前的准备工作

- 使用 Amazon CloudFormation 模板启动 Amazon EC2 实例（控制台）
- 使用 Amazon CloudFormation 模板启动 Amazon EC2 实例（Amazon CLI）

开始前的准备工作

在使用 Amazon CloudFormation 模板启动 Amazon EC2 实例之前，请务必完成以下步骤。

1. 请确保您已创建管理员用户，如步骤 1：设置中所述。仔细检查用户是否具有以下最低权限，然后添加任何不存在的最低权限：

   - cloudformation:*
   - codedeploy:*
   - ec2:*
   - iam:AddRoleToInstanceProfile
   - iam:CreateInstanceProfile
   - iam:CreateRole
   - iam:DeleteInstanceProfile
   - iam:DeleteRole
   - iam:DeleteRolePolicy
   - iam:GetRole
   - iam:DeleteRolePolicy
   - iam:PutRolePolicy
   - iam:RemoveRoleFromInstanceProfile
2. 确保您有实例密钥对，以允许通过 SSH 访问运行 Amazon Linux 的 Amazon EC2 实例，或者允许通过 RDP 访问运行 Windows Server 的实例。

   要查找密钥对名称，请打开 Amazon EC2 控制台，网址为 https://console.aws.amazon.com/ec2。在导航窗格中，在 Network & Security（网络和安全）下，选择 Key Pairs（密钥对），然后记下列表中的密钥对名称。

   要生成新的密钥对，请参阅使用 Amazon EC2 创建密钥对。请确保密钥对是《Amazon Web Services 一般参考》的区域和终端节点中列出的其中一个区域中创建的。否则，您无法将实例密钥对与 CodeDeploy 结合使用。

使用 Amazon CloudFormation 模板启动 Amazon EC2 实例（控制台）

1. 登录到 Amazon Web Services 管理控制台 并打开 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

   > **Important**
   >
   > 使用您在Amazon Web Services 管理控制台中使用的同一账户登录 入门 CodeDeploy。在全局导航栏上的区域选择器中，选择《Amazon Web Services 一般参考》的区域和终端节点中列出的某个区域。CodeDeploy 仅支持这些区域。

2. 选择创建堆栈。
3. 在选择模板中，选择指定 Amazon S3 模板 URL。在框中键入您所在区域的 Amazon CloudFormation模板的位置，然后选择 Next（下一步）。

   对于中国（北京）区域：

   - `https://s3.cn-north-1.amazonaws.com.cn/aws-codedeploy-cn-north-1/templates/latest/CodeDeploy_SampleCF_Template.json`（适用于中国（北京）区域）
4. 在 Stack name（堆栈名称）框中，键入堆栈的名称（例如，**`CodeDeployDemoStack`**）。
5. 在 Parameters 中，键入以下内容，然后选择 Next。

   - 对于 InstanceCount，键入要启动的实例的数量。（建议您保留默认值 1。）
   - 对于 InstanceType，键入要启动的实例类型（或保留默认值 t1.micro）。
   - 对于 KeyPairName，键入实例密钥对名称。仅键入密钥对名称而不是密钥对文件扩展名。
   - 对于 OperatingSystem 框，键入 **`Windows`** 以启动运行 Windows Server 的实例（或保留默认值 Linux）。
   - 对于 SSHLocation，键入要用于使用 SSH 或 RDP 连接到实例的 IP 地址范围（或保留默认值 0.0.0.0/0）。

   > **Important**
   >
   > **`0.0.0.0/0`** 的默认值仅用于演示目的。CodeDeploy 不要求 Amazon EC2 实例可以不受限制地访问端口。作为最佳实践，建议您限制对 SSH（和 HTTP）端口的访问。

- 对于 TagKey，键入 CodeDeploy 将用于在部署过程中标识实例的实例标签键（或保留默认值 Name）。
- 对于 TagValue，键入 CodeDeploy 将用于在部署过程中标识实例的实例标签值（或保留默认值 CodeDeployDemo）。

6. 在 Options 页上，将选项框留空，然后选择 Next。

> **Important**
>
> Amazon CloudFormation 标签不同于 CodeDeploy 标签。Amazon CloudFormation 使用标签来简化对基础设施的管理。CodeDeploy 使用标签来识别 Amazon EC2 实例。您在指定参数页面上指定了 CodeDeploy 标签。

7. 在审核页上的功能中，选中我确认，Amazon CloudFormation 可能创建 IAM 资源框，然后选择创建。

   在 Amazon CloudFormation 创建堆栈并启动 Amazon EC2 实例之后，在 Amazon CloudFormation 控制台中，CREATE_COMPLETE 将显示在状态列中。此过程可能耗时数分钟。

要验证 CodeDeploy 代理是否正在 Amazon EC2 实例上运行，请参阅管理 CodeDeploy 代理操作，然后继续使用创建应用程序 CodeDeploy。

使用 Amazon CloudFormation 模板启动 Amazon EC2 实例（Amazon CLI）

1. 在调用 Amazon CloudFormation 命令时使用 create-stack 模板。此堆栈将使用安装的 CodeDeploy 代理启动新的 Amazon EC2 实例。

   要启动运行 Amazon Linux 的 Amazon EC2 实例，请执行以下操作：

```
aws cloudformation create-stack \
  --stack-name CodeDeployDemoStack \
  --template-url templateURL \
  --parameters ParameterKey=InstanceCount,ParameterValue=1
 ParameterKey=InstanceType,ParameterValue=t1.micro \
    ParameterKey=KeyPairName,ParameterValue=keyName
 ParameterKey=OperatingSystem,ParameterValue=Linux \
    ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
 ParameterKey=TagKey,ParameterValue=Name \
    ParameterKey=TagValue,ParameterValue=CodeDeployDemo \
  --capabilities CAPABILITY_IAM
```

要启动运行 Windows Server 的 Amazon EC2 实例，请执行以下操作：

```
aws cloudformation create-stack --stack-name CodeDeployDemoStack --template-
url template-url --parameters ParameterKey=InstanceCount,ParameterValue=1
 ParameterKey=InstanceType,ParameterValue=t1.micro
 ParameterKey=KeyPairName,ParameterValue=keyName
 ParameterKey=OperatingSystem,ParameterValue=Windows
 ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
 ParameterKey=TagKey,ParameterValue=Name
 ParameterKey=TagValue,ParameterValue=CodeDeployDemo --capabilities CAPABILITY_IAM
```

keyName 是实例密钥对的名称。仅键入密钥对名称而不是密钥对文件扩展名。

template-url 是您所在区域的 Amazon CloudFormation 模板的位置：

对于中国（北京）区域：

- `https://s3.cn-north-1.amazonaws.com.cn/aws-codedeploy-cn-north-1/templates/latest/CodeDeploy_SampleCF_Template.json`（适用于中国（北京）区域）

此命令使用 Amazon CloudFormation 模板在指定的 Amazon S3 存储桶中创建一个名为 **CodeDeployDemoStack** 的 Amazon CloudFormation 堆栈。虽然 Amazon EC2 实例基于 t1.micro 实例类型，但您可使用任何类型。虽然它是使用值 **CodeDeployDemo** 标记的，但您可使用任何值标记它。它已应用指定的实例密钥对。

2. 调用 describe-stacks 命令以验证名为 Amazon CloudFormation 的 **CodeDeployDemoStack** 堆栈是否已成功创建：

```
aws cloudformation describe-stacks --stack-name CodeDeployDemoStack --query
 "Stacks[0].StackStatus" --output text
```

在返回 `CREATE_COMPLETE` 值之前，不要继续。

要验证 CodeDeploy 代理是否正在 Amazon EC2 实例上运行，请参阅管理 CodeDeploy 代理操作，然后继续使用创建应用程序 CodeDeploy。

配置要 EC2 使用的亚马逊实例 CodeDeploy

这些说明向您展示了如何配置运行亚马逊 Linux、Ubuntu 服务器、红帽企业 Linux (RHEL) 或 Windows 服务器的亚马逊 EC2 实例以用于部署。 CodeDeploy

> **Note**
>
> 如果您没有亚马逊 EC2 实例，则可以使用该 Amazon CloudFormation 模板启动一个运行亚马逊 Linux 或 Windows Server 的实例。我们不提供适用于 Ubuntu Server 或 RHEL 的模板。

步骤 1：验证 IAM 实例配置文件已附加到您的 Amazon EC2 实例

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的 Instances 下，选择 Instances。
3. 在列表中浏览并选择您的 Amazon EC2 实例。
4. 在详细信息窗格中的描述选项卡上，记下 IAM 角色字段中的值，然后继续下一部分。

   如果该字段为空，您可以向实例附加 IAM 实例配置文件。有关信息，请参阅将 IAM 角色附加到实例。

步骤 2：验证附加的 IAM 实例配置文件是否具有正确的访问权限

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
3. 浏览并选择您在前一部分的步骤 4 中记下的 IAM 角色名称。

   > **Note**
   >
   > 如果要使用 Amazon CloudFormation 模板生成的服务角色，而不是按照中的说明创建的服务角色步骤 2：为创建服务角色 CodeDeploy，请注意以下几点：
   > 在我们 Amazon CloudFormation 模板的某些版本中，生成并附加到 Amazon 实例的 IAM EC2 实例配置文件的显示名称与 IAM 控制台中的显示名称不同。例如，IAM 实例配置文件的显示名称可能为 `CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX`，而 IAM 控制台中的 IAM 实例

配置文件的显示名称可能为 `CodeDeploySampleStack-expnyi6-InstanceRole-C5P33V1L64EX`。
为帮助您标识 IAM 控制台中的实例配置文件，您会看到二者的前缀 `CodeDeploySampleStack-expnyi6-InstanceRole` 是相同的。有关这些显示名称可能不同的原因的信息，请参阅实例配置文件。

4. 选择 Trust Relationships 选项卡。如果可信实体中没有显示身份提供商 ec2.amazonaws.com 的条目，则无法使用此亚马逊实例。 EC2 使用中的信息停止并创建 Amazon EC2 实例使用适用于 CodeDeploy 的实例。

   如果有一个显示身份提供商 ec2.amazonaws.com 的条目，并且您仅将应用程序存储在存储 GitHub库中，请直接跳至。步骤 3：为 Amazon EC2 实例添加标签

   如果有显示身份提供商 ec2.amazonaws.com 的条目，并且您将应用程序存储在 Amazon S3 存储桶中，则选择权限选项卡。

5. 如果权限策略区域中有一个策略，则展开该策略，然后选择编辑策略。

6. 选择 JSON 选项卡。如果您将应用程序存储在 Amazon S3 存储桶中，请确保 `"s3:Get*"` 和 `"s3:List*"` 位于指定操作列表中。

   它可能如下所示：

```
{"Statement":[{"Resource":"*","Action":[
  ... Some actions may already be listed here ...
  "s3:Get*","s3:List*"
  ... Some more actions may already be listed here ...
  ],"Effect":"Allow"}]}
```

   也可能如下所示：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
```

```
        ],
        "Resource": "*"
      }
    ]
  }
```

如果 "s3:Get*" 和 "s3:List*" 不在指定操作列表中，请选择 Edit 添加它们，然后选择 Save。（如果 "s3:Get*" 和 "s3:List*" 都不是列表中的最后一个操作，请确保在操作后添加逗号，以便策略文档进行验证。）

> **Note**
>
> 我们建议您将此政策限制为仅适用于您的亚马逊 EC2实例必须访问的 Amazon S3 存储桶。确保允许访问包含 CodeDeploy 代理的 Amazon S3 存储桶。否则，在实例上安装或更新 CodeDeploy 代理时可能会出现错误。例如：
>
> ```
> {
>  "Version": "2012-10-17",
>  "Statement": [
>  {
>  "Effect": "Allow",
>  "Action": [
>  "s3:Get*",
>  "s3:List*"
>  ],
>  "Resource": [
>  "arn:aws:s3:::amzn-s3-demo-bucket/*",
>  "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
>  "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
>  ]
>  }
>  ]
> }
> ```

步骤 3：为 Amazon EC2 实例添加标签

有关如何为 Amazon EC2 实例添加标签 CodeDeploy 以便在部署期间找到它的说明，请参阅在控制台中使用标签，然后返回此页面。

> **Note**
>
> 您可以使用任何您喜欢的密钥和值来标记 Amazon EC2 实例。只需确保在部署到此实例时指定此密钥和值即可。

步骤 4：在 Amazon EC2 实例上安装 Amazon CodeDeploy 代理

有关如何在 Amazon EC2 实例上安装 CodeDeploy 代理并验证其是否正在运行的说明，请参阅管理 CodeDeploy 代理操作，然后继续使用创建应用程序 CodeDeploy。

使用本地实例 CodeDeploy

本地实例是指任何不是 Amazon EC2 实例的物理设备，可以运行 CodeDeploy 代理并连接到公共 Amazon 服务终端节点。

将 CodeDeploy 应用程序修订部署到本地实例涉及两个主要步骤：

- 步骤 1-配置每个本地实例，向其注册 CodeDeploy，然后对其进行标记。
- 步骤 2 - 将应用程序修订部署到本地实例。

> **Note**
>
> 要尝试创建示例应用程序修订并将它部署到已正确配置并注册的本地实例，请参阅教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）。有关本地实例及其使用方式的信息 CodeDeploy，请参阅Working with On-Premises Instances。

如果您不想再在部署中使用本地实例，可以从部署组中删除本地实例的标签。更可靠的方式是从实例中删除本地实例标签。您也可以显式取消注册本地实例，使其无法再在任何部署中使用。有关更多信息，请参阅 在 CodeDeploy 中管理本地实例操作。

本节中的说明向您展示如何配置本地实例，然后对其进行注册和标记， CodeDeploy 以便在部署中使用该实例。本节还介绍 CodeDeploy 如何使用获取本地实例的相关信息，并在您不再计划部署本地实例后取消注册该实例。

主题

- 配置本地实例的先决条件
- 向注册本地实例 CodeDeploy
- 在 CodeDeploy 中管理本地实例操作

配置本地实例的先决条件

必须满足以下条件才能注册本地实例。

> **Important**
>
> 如果您使用register-on-premises-instance命令并定期刷新通过 Amazon Security Token Service (Amazon STS) 生成的临时证书，则还有其他先决条件。有关信息，请参阅IAM 会话 ARN 注册前提条件。

设备要求

您要准备、注册和标记为本地实例的设备 CodeDeploy 必须运行支持的操作系统。有关列表，请参阅 CodeDeploy 代理支持的操作系统。

如果您的操作系统不受支持，则该 CodeDeploy 代理可以作为开源版本供您使用，以适应您的需求。有关更多信息，请参阅中的CodeDeploy 代理存储库 GitHub。

出站通信

本地实例必须能够连接到公共 Amazon 服务终端节点才能与之通信 CodeDeploy。

CodeDeploy 代理使用 HTTPS 通过端口 443 进行出站通信。

管理控制

在本地实例上用于配置本地实例的本地账户或网络账户必须能够以 `sudo` 或 `root` 身份（对于 Ubuntu Server）或者以管理员身份（对于 Windows Server）运行。

IAM 权限

您在注册本地实例时使用的 IAM 身份必须具有完成注册的权限（以及在需要时取消注册本地实例的权限）。

请确保发出调用的 IAM 身份除了拥有步骤 3：限制 CodeDeploy 用户的权限中所述的策略之外，还另外附加了以下策略：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource": "*"
    }
  ]
}
```

有关如何附加 IAM policy 的信息，请参阅管理 IAM policy。

向注册本地实例 CodeDeploy

要注册本地实例，必须使用 IAM 身份来对您的请求进行身份验证。您可以针对您使用的 IAM 身份和注册方法从以下选项中进行选择：

- 使用 IAM 角色 ARN 对请求进行身份验证。
  - 使用register-on-premises-instance命令并定期刷新通过 Amazon Security Token Service (Amazon STS) 生成的临时凭证来手动配置大多数注册选项。此选项提供了最高级别的安全性，因为身份验证是使用临时令牌进行的，该令牌会超时，必须定期刷新。建议将此选项用于任何规模的生产部署。有关信息，请参阅使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例。
- （不推荐）使用 IAM 用户 ARN 对请求进行身份验证。
  - 使用 register 命令执行高度自动化的注册过程。此选项仅适用于安全性要求较低的非生产部署。此选项不太安全，因为它使用静态（永久）凭证进行身份验证。此选项适用于注册单个本地实例。有关信息，请参阅使用 register 命令（IAM 用户 ARN）注册本地实例。

- 使用 register-on-premises-instance 命令手动设置大多数注册选项。适合注册少量内部实例的情况。有关信息，请参阅使用 register-on-premises-instance命令（IAM 用户 ARN）注册本地实例。

主题

- 使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例
- 使用 register 命令（IAM 用户 ARN）注册本地实例
- 使用 register-on-premises-instance命令（IAM 用户 ARN）注册本地实例

使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例

为了最大限度地控制本地实例的身份验证和注册，您可以使用register-on-premises-instance命令并定期刷新通过 Amazon Security Token Service (Amazon STS) 生成的临时证书。实例的静态 IAM 角色扮演这些刷新的 Amazon STS 证书的角色来执行 CodeDeploy 部署操作。

当您需要注册大量实例时，此方法非常有用。它允许您使用自动完成注册过程 CodeDeploy。您可以使用自己的身份和身份验证系统对本地实例进行身份验证，并将服务中的 IAM 会话证书分发给实例以供使用 CodeDeploy。

> **Note**
>
> 或者，您可以使用分布到所有本地实例的共享 IAM 用户调用 Amazon STS AssumeRoleAPI 来检索本地实例的会话证书。此方法安全性较低，建议不要用于生产或关键任务型环境。

使用以下主题中的信息，使用生成的临时安全证书配置本地实例 Amazon STS。

主题

- IAM 会话 ARN 注册前提条件
- 步骤 1：创建本地实例将担任的 IAM 角色
- 步骤 2：使用为单个实例生成临时证书 Amazon STS
- 步骤 3：将配置文件添加到本地实例
- 步骤 4：为部署准备本地实例 CodeDeploy
- 步骤 5：向注册本地实例 CodeDeploy
- 步骤 6：标记本地实例

- 步骤 7：将应用程序修订部署到本地实例
- 步骤 8：跟踪对本地实例的部署

IAM 会话 ARN 注册前提条件

除了在 配置本地实例的先决条件 中列出的前提条件之外，还必须满足以下要求：

IAM 权限

必须向用于注册本地实例的 IAM 身份授予执行 CodeDeploy 操作的权限。确保 AWSCodeDeployFullAccess 托管策略已附加到该 IAM 身份。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略。

用于刷新临时凭证的系统

如果您使用 IAM 会话 ARN 注册本地实例，则您必须有一个系统来定期刷新该临时凭证。如果在生成凭证时指定的期限较短，临时凭证会在一小时（或更短时间）后过期。有两种刷新凭证的方法：

- 方法 1：使用您的企业网络中的身份和身份验证系统以及一个 CRON 脚本，该脚本定期轮询该身份和身份验证系统，并将最新的会话凭证复制到该实例。这使您能够将身份验证和身份结构与集成，而 Amazon 无需更改 CodeDeploy 代理或服务以支持您在组织中使用的身份验证类型。
- 方法 2：定期在实例上运行 CRON 作业以调用 Amazon STS AssumeRole操作并将会话凭证写入 CodeDeploy 代理可以访问的文件。此方法仍需要使用 IAM 用户并将凭证复制到本地实例，但您可以对您的各个本地实例重复使用相同的 IAM 用户和凭证。

> **Note**
>
> 无论您使用的是方法 1 还是方法 2，都必须设置一个进程，以便在临时会话证书更新后重新启动 CodeDeploy 代理，这样新的凭证才能生效。

有关创建和使用 Amazon STS 证书的信息，请参阅 Amazon Security Token Service API 参考和使用临时安全证书请求访问 Amazon 资源。

步骤 1：创建本地实例将担任的 IAM 角色

您可以使用 Amazon CLI 或 IAM 控制台创建 IAM 角色，您的本地实例将使用该角色进行身份验证和交互 CodeDeploy。

您只需要创建一个 IAM 角色。您的每个本地实例都可以担任此角色，以获取为此角色提供权限的临时安全凭证。

您创建的角色需要以下权限才能访问安装 CodeDeploy 代理所需的文件：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

我们建议您将此策略限制为您的本地实例需要访问的那些 Amazon S3 存储桶。如果您限制此策略，请确保允许访问包含 CodeDeploy 代理的 Amazon S3 存储桶。否则，每当在本地实例上安装或更新 CodeDeploy 代理时，都可能出现错误。有关如何控制对 Amazon S3 存储桶的访问权限的更多信息，请参阅管理您的 Amazon S3 资源的访问权限。

创建 IAM 角色

1. 调用 create-role 命令，同时使用 `--role-name` 选项指定 IAM 角色的名称（例如 `CodeDeployInstanceRole`），并使用 `--assume-role-policy-document` 选项提供权限。

   为此实例创建 IAM 角色时，您可以为其指定角色名称 `CodeDeployInstanceRole`，并在名为 `CodeDeployRolePolicy.json` 的文件中提供所需的权限：

   ```
   aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policy-
   document file://CodeDeployRolePolicy.json
   ```

2. 在调用 create-role 命令的输出中，记录 ARN 字段的值。例如：

```
arn:aws:iam::123456789012:role/CodeDeployInstanceRole
```

当您使用 Amazon STS AssumeRoleAPI 为每个实例生成短期证书时，您将需要角色 ARN。

有关创建 IAM 角色的更多信息，请参阅 I A M 用户指南中的创建角色以向 Amazon 服务委派权限。

有关为现有角色分配权限的信息，请参阅Amazon CLI 命令参考put-role-policy中的。

步骤 2：使用为单个实例生成临时证书 Amazon STS

在生成将用于注册本地实例的临时凭证之前，必须创建或选择您将为其生成临时凭证的 IAM 身份（用户或角色）。在此 IAM 身份的策略设置中，必须包含 `sts:AssumeRole` 权限。

有关向 IAM 身份授予`sts:AssumeRole`权限的信息，请参阅创建向 Amazon 服务委派权限的角色和AssumeRole。

有两种生成临时凭证的方法：

- 将 assume-role 命令与 Amazon CLI结合使用。例如：

  ```
  aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --role-
  session-name session-name
  ```

  其中：

  - *12345ACCOUNT*是贵组织的 12 位账号。
  - *role-arn*是你在中生成的要担任的角色的 ARN。步骤 1：创建本地实例将担任的 IAM 角色
  - *session-name*是您要为现在正在创建的角色会话起的名称。

  > **Note**
  > 如果您使用定期轮询身份和身份验证系统并将最新的会话凭证复制到实例的 CRON 脚本（用于刷新临时证书的方法 1，如所述IAM 会话 ARN 注册前提条件），则可以改用任何支持的 Amazon 软件开发工具包进行调AssumeRole用。

- 使用提供的工具 Amazon。

该 aws-codedeploy-session-helper工具会生成 Amazon STS 凭证并将其写入您放置在实例上的文件中。此工具最适用于 IAM 会话 ARN 注册前提条件 中所述的用于刷新临时凭证的方法 2。在此方法中，该 aws-codedeploy-session-helper工具放在每个实例上，并使用 IAM 用户的权限执行命令。每个实例都将相同的 IAM 用户凭证与此工具结合使用。

有关更多信息，请参阅aws-codedeploy-session-helper GitHub 存储库。

> **Note**
>
> 在您创建 IAM 会话凭证之后，请将它们置于本地实例上的任何位置。在下一步中，您将配置 CodeDeploy 代理以访问此位置的凭证。

在继续操作之前，请确保您将用来定期刷新临时凭证的系统已经准备就绪。如果未刷新临时凭证，则部署到本地实例的操作将失败。有关更多信息，请参阅 IAM 会话 ARN 注册前提条件 中的“用于刷新临时凭证的系统”。

步骤 3：将配置文件添加到本地实例

使用 root 或管理员权限将配置文件添加到本地实例。此配置文件用于声明 IAM 证书和要使用的目标 Amazon 区域 CodeDeploy。必须将该文件添加到本地实例上的特定位置。该文件必须包含 IAM 临时会话 ARN、其私有密钥 ID 和私有访问密钥以及目标 Amazon 区域。

添加配置文件

1. 在本地实例上的以下位置，创建名为 `codedeploy.onpremises.yml`（针对 Ubuntu Server 或 RHEL 本地实例）或名为 `conf.onpremises.yml`（针对 Windows Server 本地实例）的文件：
   - 对于 Ubuntu Server：`/etc/codedeploy-agent/conf`
   - 对于 Windows Server：`C:\ProgramData\Amazon\CodeDeploy`
2. 使用文本编辑器将以下信息添加到新创建的 `codedeploy.onpremises.yml` 文件（Linux）或 `conf.onpremises.yml` 文件（Windows）：

```
---
iam_session_arn: iam-session-arn
aws_credentials_file: credentials-file
region: supported-region
```

其中：

- *iam-session-arn*是您在中记下的 IAM 会话 ARN。步骤 2：使用为单个实例生成临时证书 Amazon STS
- *credentials-file*是临时会话 ARN 的证书文件的位置，如中所述。步骤 2：使用为单个实例生成临时证书 Amazon STS
- *supported-region*是 CodeDeploy支持的区域之一，如中的区域和终端节点中所列Amazon Web Services 一般参考。

步骤 4：为部署准备本地实例 CodeDeploy

安装和配置 Amazon CLI

在本地实例 Amazon CLI 上安装和配置。（ Amazon CLI 将用于在本地实例上下载和安装 CodeDeploy 代理。）

1. 要在本地实例 Amazon CLI 上安装，请按照Amazon Command Line Interface 用户指南中的开始设置 Amazon CLI中的说明进行操作。

> **Note**
>
> CodeDeploy 用于处理本地实例的命令已在 1.7.19 版本中提供。 Amazon CLI如果您 Amazon CLI 已经安装了的版本，则可以通过调用来检查其版本aws --version。

2. 要在本地实例 Amazon CLI 上配置，请按照Amazon Command Line Interface 用户指南中的配置 Amazon CLI中的说明进行操作。

> **Important**
>
> 在配置时 Amazon CLI（例如，通过调用aws configure命令），请务必指定至少具有中所述权限的 IAM 用户的私有密钥 ID 和私有访问密钥IAM 会话 ARN 注册前提条件。

设置 AWS_REGION环境变量（仅限 Ubuntu 服务器和 RHEL）

如果您没有在本地实例上运行 Ubuntu Server 或 RHEL，请跳过此步骤，直接转到 “安装代理”。CodeDeploy

在 Ubuntu 服务器或 RHEL 本地实例上安装代 CodeDeploy 理，并允许实例在新版本 CodeDeploy 可用时更新代理。为此，您可以将实例上的 `AWS_REGION` 环境变量设置为 CodeDeploy 支持的某个区域的标识符。我们建议您将该值设置为 CodeDeploy 应用程序、部署组和应用程序修订所在的区域（例如`us-west-2`）。有关区域的列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。

要设置环境变量，请从终端调用以下命令：

```
export AWS_REGION=supported-region
```

哪里*`supported-region`*是区域标识符（例如，`us-west-2`）。

安装代 CodeDeploy 理

- 对于 Ubuntu Server 本地实例，请按照安装适用于 Ubuntu Server 的 CodeDeploy 代理中的说明操作，然后返回本页。
- 对于 RHEL 本地实例，请按照安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理中的说明操作，然后返回本页。
- 对于 Windows Server 本地实例，请按照安装适用于 Windows Server 的 CodeDeploy 代理中的说明操作，然后返回本页。

步骤 5：向注册本地实例 CodeDeploy

这一步中的说明假设您从本地实例本身上注册该本地实例。您可以从单独的设备或已 Amazon CLI 安装和配置的实例注册本地实例。

使用 Amazon CLI 向注册本地实例， CodeDeploy 以便可以在部署中使用该实例。

在使用之前 Amazon CLI，您需要在中创建的临时会话证书的 ARN。步骤 3：将配置文件添加到本地实例例如，对于您指定为 `AssetTag12010298EX` 的实例：

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

调用 register-on-premises-instance 命令，在命令中指定：

- 唯一标识本地实例的名称（使用 `--instance-name` 选项）。

> **Important**
>
> 为了帮助标识本地实例，特别是用于调试用途，我们强烈建议您指定能够反映本地实例的某种唯一特性的名称（例如，在适用时可以指定 STS 凭证的会话名称以及序列号或内部资产标识符）。如果您指定 MAC 地址作为名称，请注意 MAC 地址包含 CodeDeploy 不允许使用的字符，例如冒号 (:)。有关允许字符的列表，请参阅CodeDeploy 配额。

- 您在步骤 1：创建本地实例将担任的 IAM 角色中设置以对多个本地实例进行身份验证的 IAM 会话 ARN。

例如：

```
aws deploy register-on-premises-instance --instance-name name-of-instance --iam-
session-arn arn:aws:sts::account-id:assumed-role/role-to-assume/session-name
```

其中：

- *name-of-instance*是您用来标识本地实例的名称，例如`AssetTag12010298EX`。
- *account-id*是贵组织的 12 位数账户 ID，例如`111222333444`。
- *role-to-assume*是您为实例创建的 IAM 角色的名称，例如`CodeDeployInstanceRole`。
- *session-name*是您在中指定的会话角色的名称步骤 2：使用为单个实例生成临时证书 Amazon STS。

步骤 6：标记本地实例

您可以使用 Amazon CLI 或 CodeDeploy 控制台来标记本地实例。（在部署期间CodeDeploy使用本地实例标签来识别部署目标。）

标记本地实例（CLI）

- 调用-premis add-tags-to-ones-instances 命令，指定：
  - 唯一标识本地实例的名称（使用 `--instance-names` 选项）。
  - 您要使用的本地实例标签密钥的名称和标签值（使用 `--tags` 选项）。必须同时指定名称和值。 CodeDeploy 不允许仅包含值的本地实例标签。

    例如：

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

标记本地实例（控制台）

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择本地实例。
3. 在本地实例列表中，选择要标记的本地实例名称。
4. 在标签列表中，选择或输入所需的标签键和标签值。当您输入标签键和标签值之后，将显示另一行。您可以重复此步骤，最多添加 10 个标签。要删除标签，请选择移除。
5. 在您添加标签之后，选择 Update Tags。

步骤 7：将应用程序修订部署到本地实例

现在，您已准备好将应用程序修订部署到已注册和标记的本地实例。

您可以将应用程序修订部署到本地实例，其方式类似于将应用程序修订部署到 Amazon EC2 实例。有关说明，请参阅使用创建部署 CodeDeploy。这些说明包含指向先决条件的链接，其中包括创建应用程序、创建部署组和准备应用程序修订。如果您希望部署简单的示例应用程序修订，可以创建教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）的步骤 2：创建示例应用程序修订中所述的修订。

> **Important**
>
> 如果您在创建以本地实例为目标的部署组时重复使用 CodeDeploy 服务角色，则必须在该服务角色的策略声明`Action`部分中包含`Tag:get*`该服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

步骤 8：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

您可以跟踪本地实例的部署，其方式与跟踪 Amazon EC2 实例部署的方式类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 。

使用 register 命令（IAM 用户 ARN）注册本地实例

> **Important**
>
> 不建议使用 IAM 用户注册实例，因为它使用静态（永久）凭证进行身份验证。为了提高安全性，我们建议使用临时凭证注册实例以进行身份验证。有关更多信息，请参阅 使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例。

> **Important**
>
> 确保您制定了轮换 IAM 用户的访问密钥（永久凭证）的计划。有关更多信息，请参阅轮换访问密钥。

本部分介绍如何以最少的工作量使用 CodeDeploy 配置本地实例并注册和标记该实例。当您处理一个或少量的本地实例时，register 命令非常有用。仅当您使用 IAM 用户 ARN 对实例进行身份验证时，才能使用 register 命令。使用 IAM 会话 ARN 进行身份验证时，不能使用 register 命令。

使用该register命令时，可以执行以下 CodeDeploy 操作：

- 如果您未使用命令指定 IAM 用户，请在中 Amazon Identity and Access Management 为本地实例创建 IAM 用户。
- 将该 IAM 用户的凭证保存到本地实例配置文件中。
- 向注册本地实例 CodeDeploy。
- 如果您在命令中指定了标签，则向本地实例添加标签。

> **Note**
>
> 该register-on-premises-instance命令是寄存器命令的替代命令。如果您希望主要靠自己使用 CodeDeploy 配置本地实例并注册和标记该实例，请使用 register-on-premises-instance 命

令。您也可以通过 register-on-premises-instance 命令来使用 IAM 会话 ARN 注册实例，而不是使用 IAM 用户 ARN 进行注册。如果您有大量本地实例，则此方法有很大的优势。具体来说，您可以使用一个 IAM 会话 ARN 对多个实例进行身份验证，而不必为每个本地实例都单独创建一个 IAM 用户。有关更多信息，请参阅使用 register-on-premises-instance命令（IAM 用户 ARN）注册本地实例和使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例。

主题

- 步骤 1：在本地实例 Amazon CLI 上安装和配置
- 步骤 2：调用 register 命令
- 步骤 3：调用 install 命令
- 步骤 4：将应用程序修订部署到本地实例
- 步骤 5：跟踪对本地实例的部署

步骤 1：在本地实例 Amazon CLI 上安装和配置

1. 在本地实例 Amazon CLI 上安装。请按照《Amazon Command Line Interface 用户指南》的使用 Amazon CLI进行设置中的说明操作。

   > **Note**
   >
   > CodeDeploy 1.7.19 及更高 Amazon CLI 版本中提供了用于使用本地实例的命令。如果您 Amazon CLI 已经安装了，请aws --version致电查看其版本。

2. 在本地实例 Amazon CLI 上配置。按照《Amazon Command Line Interface 用户指南》的配置 Amazon CLI 中的说明进行操作。

   > **Important**
   >
   > 在配置时 Amazon CLI（例如，通过调用aws configure命令），请务必指定除中指定的权限之外还至少具有以下访问权限的 IAM 用户的密钥 ID 和私有 Amazon 访问密钥配置本地实例的先决条件。这使得在本地实例上下载和安装 CodeDeploy 代理成为可能。访问权限应与以下示例类似：

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser",
        "tag:getTagKeys",
        "tag:getTagValues",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*"
      ]
    }
  ]
}
```

步骤 2：调用 register 命令

在这一步中，我们假设您从本地实例本身注册该本地实例。您也可以从单独的设备或实例注册本地实例，该设备或实例已按上一步所述 Amazon CLI 安装和配置了本地实例。

使用调用 Amazon CLI regist er 命令，指定：

- 用于唯一标识本地实例的名称 CodeDeploy（带`--instance-name`选项）。

> **Important**
>
> 为了帮助在以后标识本地实例，特别是用于调试用途，我们强烈建议您使用的名称能够映射到本地实例的某种唯一特性（例如，在适用时可以使用序列号或唯一内部资产标识符）。如果您为名称指定 MAC 地址，请注意 MAC 地址包含 CodeDeploy 不允许使用的字符，例如冒号 (`:`)。有关允许字符的列表，请参阅CodeDeploy 配额。

- （可选）您希望与此本地实例关联的现有 IAM 用户的 ARN（使用 `--iam-user-arn` 选项）。要获取 IAM 用户的 ARN，请调用 get-user 命令，或者在 IAM 控制台的用户部分中选择 IAM 用户名，然后在摘要部分找到用户 ARN 值。如果未指定此选项，则 CodeDeploy 将在您的 Amazon 账户中代表您创建一个 IAM 用户，并将其与本地实例关联。

> **Important**
>
> 如果您指定了 `--iam-user-arn` 选项，则还必须手动创建本地实例配置文件，如步骤 4：将配置文件添加到本地实例中所述。
>
> 您只能将一个 IAM 用户与一个本地实例关联。尝试将一个 IAM 用户与多个本地实例关联会导致出错，对这些本地实例的部署将失败，或者对这些本地实例的部署会停滞在永久等待的状态。

- （可选）一组本地实例标签（带`--tags`选项），CodeDeploy 用于标识要部署到的 Amazon EC2 实例集。使用 `Key=`*`tag-key`*`,Value=`*`tag-value` 指定各个标签（例如，`Key=Name,Value=Beta Key=Name,Value=WestRegion`）。如果未指定此选项，将不注册标签。要稍后注册标签，请调用-premises add-tags-to-on-instances 命令。
- （可选）将在其中注册本地实例的 Amazon 区域 CodeDeploy（带`--region`选项）。这必须是 Amazon Web Services 一般参考的区域和终端节点中列出的受支持区域之一（例如 `us-west-2`）。如果未指定此选项，则将使用与调用 IAM 用户关联的默认 Amazon 区域。

例如：

```
aws deploy register --instance-name AssetTag12010298EX --iam-user-
arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem --tags
 Key=Name,Value=CodeDeployDemo-OnPrem --region us-west-2
```

register 命令执行以下操作：

1. 如果未指定现有 IAM 用户，则创建一个 IAM 用户，向该用户附加必需的权限，然后生成对应的私有密钥和私有访问密钥。本地实例将使用此 IAM 用户及其权限和凭证进行身份验证和交互 CodeDeploy。
2. 向注册本地实例 CodeDeploy。
3. 如果指定，则 CodeDeploy 将使用`--tags`选项指定的标签与注册的本地实例名称相关联。
4. 如果创建了 IAM 用户，则还会在调用 register 命令的同一个目录中创建必需的配置文件。

如果此命令遇到错误，则将显示错误消息，说明您可以如何手动完成剩余步骤。否则，将显示成功消息，说明如何调用 install 命令，如接下来的步骤中所列。

步骤 3：调用 install 命令

在本地实例中，使用调 Amazon CLI 用安装命令，指定：

- 配置文件的路径（使用 `--config-file` 选项）。
- （可选）是否替换本地实例上已存在的配置文件（使用 `--override-config` 选项）。如果未指定，则不替换现有配置文件。
- （可选）将在其中注册本地实例的 Amazon 区域 CodeDeploy （带`--region`选项）。这必须是 Amazon Web Services 一般参考的区域和终端节点中列出的受支持区域之一（例如 `us-west-2`）。如果未指定此选项，则将使用与调用 IAM 用户关联的默认 Amazon 区域。
- （可选）安装 CodeDeploy 代理的自定义位置（带`--agent-installer`选项）。此选项对于安装 CodeDeploy 不正式支持的 CodeDeploy 代理的自定义版本（例如基于中CodeDeploy代理存储库的自定义版本 GitHub）非常有用。该值必须是指向包含以下二者之一的 Amazon S3 存储桶的路径：
  - CodeDeploy 代理安装脚本（适用于基于 Linux 或 UNIX 的操作系统，类似于中CodeDeploy代理存储库中的 GitHub安装文件）。
  - CodeDeploy 代理安装程序包 (.msi) 文件（适用于基于 Windows 的操作系统）。

  如果未指定此选项，则 CodeDeploy 将尽最大努力从自己的位置安装与本地实例上的操作系统兼容的官方支持的 CodeDeploy 代理版本。

例如：

```
aws deploy install --override-config --config-file /tmp/codedeploy.onpremises.yml --
region us-west-2 --agent-installer s3://aws-codedeploy-us-west-2/latest/codedeploy-
agent.msi
```

install 命令执行以下操作：

1. 检查本地实例是否是 Amazon EC2 实例。如果是，则将显示错误消息。
2. 将本地实例配置文件从实例上的指定位置复制到 CodeDeploy 代理期望找到的位置，前提是该文件尚未位于该位置。

   对于 Ubuntu Server 和 Red Hat Enterprise Linux（RHEL），这是 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。

   对于 Windows Server，这是 `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`。

   如果指定了 `--override-config` 选项，则创建或覆盖文件。
3. 在本地实例上安装 CodeDeploy 代理，然后启动它。

步骤 4：将应用程序修订部署到本地实例

现在，您已准备好将应用程序修订部署到已注册和标记的本地实例。

您可以将应用程序修订部署到本地实例，其方式类似于将应用程序修订部署到 Amazon EC2 实例。有关说明，请参阅使用创建部署 CodeDeploy。这些说明链接到各种先决条件，包括创建应用程序、创建部署组和准备应用程序修订。如果您希望部署简单的示例应用程序修订，可以创建教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）的步骤 2：创建示例应用程序修订中所述的修订。

> **Important**
>
> 如果您在创建以本地实例为目标的部署组时重复使用现有 CodeDeploy 服务角色，则必须在该服务角色的策略声明`Action`部分中包含`Tag:get*`该服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

步骤 5：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

您可以跟踪本地实例的部署，其方式与跟踪 Amazon EC2 实例部署的方式类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 。

有关更多选项，请参阅在 CodeDeploy 中管理本地实例操作。

使用 register-on-premises-instance命令（IAM 用户 ARN）注册本地实例

> **Important**
>
> 不建议使用 IAM 用户注册实例，因为它使用静态（永久）凭证进行身份验证。为了提高安全性，我们建议使用临时凭证注册实例以进行身份验证。有关更多信息，请参阅 使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例。

> **Important**
>
> 确保您制定了轮换 IAM 用户的访问密钥（永久凭证）的计划。有关更多信息，请参阅轮换访问密钥。

按以下说明，配置本地实例并主要靠自己通过 CodeDeploy 注册该实例并为其添加标签，同时使用静态 IAM 用户凭证进行身份验证。

主题

- 步骤 1：为本地实例创建 IAM 用户
- 步骤 2：向 IAM 用户分配权限
- 步骤 3：获取 IAM 用户凭证
- 步骤 4：将配置文件添加到本地实例
- 步骤 5：安装和配置 Amazon CLI
- 步骤 6：设置 AWS_REGION环境变量（仅限 Ubuntu 服务器和 RHEL）
- 步骤 7：安装代 CodeDeploy 理
- 步骤 8：向注册本地实例 CodeDeploy

- 步骤 9：标记本地实例
- 步骤 10：将应用程序修订部署到本地实例
- 步骤 11：跟踪对本地实例的部署

步骤 1：为本地实例创建 IAM 用户

创建一个 IAM 用户，本地实例将使用该用户进行身份验证和交互 CodeDeploy。

> **Important**
> 您必须为每个参与的本地实例创建单独的 IAM 用户。如果您尝试将单个 IAM 用户重复用于多个本地实例，则可能无法成功注册或标记这些本地实例 CodeDeploy。对这些本地实例的部署可能会停滞在永久等待的状态或完全失败。

我们建议您为 IAM 用户分配一个标识其用途的名称，例如 CodeDeployUser-OnPrem。

您可以使用 Amazon CLI 或 IAM 控制台创建 IAM 用户。有关更多信息，请参阅在您的 Amazon 账户中创建 IAM 用户。

> **Important**
> 无论您是使用还是 IAM 控制台创建新的 IAM 用户，都要记下为该用户提供的用户 ARN。Amazon CLI 稍后在步骤 4：将配置文件添加到本地实例 和步骤 8：向注册本地实例 CodeDeploy中您将需要此信息。

步骤 2：向 IAM 用户分配权限

如果您的本地实例将从 Amazon S3 存储桶部署应用程序修订，则您必须向 IAM 用户分配权限来与这些存储桶交互。您可以使用 Amazon CLI 或 IAM 控制台分配权限。

> **Note**
> 如果您要仅从 GitHub 存储库部署应用程序修订版，请跳过此步骤直接转至步骤 3：获取 IAM 用户凭证。（您仍需要有关您在步骤 1：为本地实例创建 IAM 用户中创建的 IAM 用户的信息。后面的步骤中将会用到这些信息。）

分配权限（CLI）

1. 在您用来调用的 Amazon EC2 实例或设备上创建包含以下策略内容的文件 Amazon CLI。采用类似于 **`CodeDeploy-OnPrem-Permissions.json`** 的格式命名文件，然后保存文件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 我们建议您将此策略限制为您的本地实例需要访问的那些 Amazon S3 存储桶。如果您限制此策略，请确保同时允许访问包含 Amazon CodeDeploy 代理的 Amazon S3 存储桶。否则，每当在关联的本地实例上安装或更新 CodeDeploy 代理时，都可能出现错误。
> 例如：
>
> ```
> {
>  "Version": "2012-10-17",
>  "Statement": [
>  {
>  "Effect": "Allow",
>  "Action": [
>  "s3:Get*",
>  "s3:List*"
>  ],
>  "Resource": [
>  "arn:aws:s3:::amzn-s3-demo-bucket/*",
>  "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
>  "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
>  ]
>  }
>  ]
> ```

```
}
```

2. 调用put-user-policy命令，指定 IAM 用户的名称（使用`--user-name`选项）、策略的名称（使用`--policy-name`选项）和新创建的策略文档的路径（使用`--policy-document`选项）。例如，假设 **`CodeDeploy-OnPrem-Permissions.json`** 文件位于您调用此命令时所在的同一个目录（文件夹）中：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name CodeDeploy-
OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-Permissions.json
```

分配权限（控制台）

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中选择 Policies，然后选择 Create Policy。（如果 Get Started 按钮出现，选择此按钮，然后选择 Create Policy。）
3. 在 Create Your Own Policy 旁，选择 Select。
4. 在 Policy Name（策略名称）框中，键入此策略的名称（例如，**`CodeDeploy-OnPrem-Permissions`**）。
5. 在策略文档框中，键入或粘贴以下权限表达式，该表达式 Amazon CodeDeploy 允许代表 IAM 用户将策略中指定的任何 Amazon S3 存储桶中的应用程序修订部署到本地实例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
```

```
    ]
}
```

6. 选择创建策略。
7. 在导航窗格中，选择 Users（用户）。
8. 在用户列表中，浏览并选择在步骤 1：为本地实例创建 IAM 用户中创建的 IAM 用户的名称。
9. 在 Permissions 选项卡上的 Managed Policies 中，选择 Attach Policy。
10. 选择名为 **CodeDeploy-OnPrem-Permissions** 的托管策略，然后选择 Attach Policy（附加策略）。

步骤 3：获取 IAM 用户凭证

获取 IAM 用户的私有密钥 ID 和秘密访问密钥。您在步骤 4：将配置文件添加到本地实例 中需要使用它们。您可以使用 Amazon CLI 或 IAM 控制台获取密钥 ID 和私有访问密钥。

> **Note**
>
> 如果您已具有私有密钥 ID 和秘密访问密钥，则跳过这一步，直接转到步骤 4：将配置文件添加到本地实例。
>
> 如果用户想在 Amazon 外部进行交互，则需要编程访问权限 Amazon Web Services 管理控制台。 Amazon APIs 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。
>
> 要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期证书签署对 Amazon CLI 或的编程请求 Amazon APIs（直接或使用 Amazon SDKs）。	按照 IAM 用户指南中的将临时证书与 Amazon 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期证书签署对 Amazon CLI 或的编程请求	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

哪个用户需要编程式访问权限？	目的	方式
	Amazon APIs（直接或使用 Amazon SDKs）。	

获取凭证（CLI）

1. 调用list-access-keys命令，指定 IAM 用户的姓名（使用`--user-name`选项），然后仅查询访问密钥 IDs（使用`--query`和`--output`选项）。例如：

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query
 "AccessKeyMetadata[*].AccessKeyId" --output text
```

2. 如果输出中没有显示任何密钥或输出中仅显示有关一个密钥的信息，请调用create-access-key命令，指定 IAM 用户的名称（带`--user-name`选项）：

```
aws iam create-access-key --user-name CodeDeployUser-OnPrem
```

在调用 create-access-key 命令的输出中，记录 `AccessKeyId` 和 `SecretAccessKey` 字段的值。在步骤 4：将配置文件添加到本地实例 中您将需要此信息。

> **Important**
>
> 这是您仅有的一次查看此秘密访问密钥的机会。如果您忘记或丢失了此秘密访问密钥，则需要按照步骤 3：获取 IAM 用户凭证 中的步骤生成新的密钥。

3. 如果已经列出了两个访问密钥，则必须通过调用delete-access-key命令删除其中一个，指定 IAM 用户的名称（带`--user-name`选项）和要删除的访问密钥的 ID（使用`--access-key-id`选项）。然后调用 create-access-key 命令，如此步骤中前面所述。下面是一个调用 delete-access-key 命令的示例：

```
aws iam delete-access-key --user-name CodeDeployUser-OnPrem --access-key-id access-
key-ID
```

> **Important**
>
> 如果您调用 delete-access-key 命令删除这些访问密钥之一，并且本地实例已经按步骤 4：将配置文件添加到本地实例中所述使用此访问密钥，则您需要再次按照步骤 4：将配置文件添加到本地实例中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对该本地实例的任何部署可能会停滞在永久等待的状态或完全失败。

获取凭证（控制台）

1. a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 如果用户的列表未显示，则在导航窗格中选择 Users。
   c. 在用户列表中，浏览并选择在步骤 1：为本地实例创建 IAM 用户中创建的 IAM 用户的名称。
2. 在 Security credentials 选项卡上，如果没有列出密钥或仅列出了一个密钥，请选择 Create access key。

   如果列出了两个访问密钥，则必须删除其中之一。选择其中一个访问密钥旁边的 Delete，然后选择 Create access key。

   > **Important**
   >
   > 如果您选择这些访问密钥之一旁边的删除，并且本地实例已经按步骤 4：将配置文件添加到本地实例中所述使用此访问密钥，则您需要再次按照步骤 4：将配置文件添加到本地实例中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对该本地实例的部署可能会停滞在永久等待的状态或完全失败。

3. 选择 Show 并记录访问密钥 ID 和秘密访问密钥。您在下一步中需要此信息。或者，您可以选择 Download .csv file 来保存访问密钥 ID 和秘密访问密钥的副本。

   > **Important**
   >
   > 除非您记录或下载了凭证，否则这是您仅有的一次查看此秘密访问密钥的机会。如果您忘记或丢失了此秘密访问密钥，则需要按照步骤 3：获取 IAM 用户凭证 中的步骤生成新的密钥。

4. 选择 “关闭” 返回到 “用户*IAM User Name*” > 页面。

步骤 4：将配置文件添加到本地实例

使用 root 或管理员权限将配置文件添加到本地实例。此配置文件将用于声明 IAM 用户证书和要使用的目标 Amazon 区域 CodeDeploy。必须将该文件添加到本地实例上的特定位置。该文件必须包含 IAM 用户的 ARN、私有密钥 ID、私有访问密钥和目标 Amazon 区域。该文件必须遵循特定格式。

1. 在本地实例上的以下位置，创建名为 `codedeploy.onpremises.yml`（针对 Ubuntu Server 或 RHEL 本地实例）或名为 `conf.onpremises.yml`（针对 Windows Server 本地实例）的文件：

   - 对于 Ubuntu Server：`/etc/codedeploy-agent/conf`
   - 对于 Windows Server：`C:\ProgramData\Amazon\CodeDeploy`

2. 使用文本编辑器将以下信息添加到新创建的 `codedeploy.onpremises.yml` 或 `conf.onpremises.yml` 文件：

   ```
   ---
   aws_access_key_id: secret-key-id
   aws_secret_access_key: secret-access-key
   iam_user_arn: iam-user-arn
   region: supported-region
   ```

   其中：

   - *`secret-key-id`*是您在步骤 1：为本地实例创建 IAM 用户或中记下的相应 IAM 用户的密钥 ID 步骤 3：获取 IAM 用户凭证。
   - *`secret-access-key`*是您在步骤 1：为本地实例创建 IAM 用户或中记下的相应 IAM 用户的私有访问密钥步骤 3：获取 IAM 用户凭证。
   - *`iam-user-arn`*是您在前面提到的 IAM 用户的 ARN。步骤 1：为本地实例创建 IAM 用户
   - *`supported-region`*是您的 CodeDeploy 应用程序、部署组和应用程序修订 CodeDeploy 所在区域所支持的标识符（例如，`us-west-2`）。有关区域的列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。

   > **Important**
   >
   > 如果您在步骤 3：获取 IAM 用户凭证中选择这些访问密钥之一旁边的删除，并且本地实例已经在使用关联的访问密钥 ID 和秘密访问密钥，则您需要按照步骤 4：将配置文件添加到

本地实例中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对您的本地实例的任何部署可能会停滞在永久等待的状态或完全失败。

步骤 5：安装和配置 Amazon CLI

在本地实例 Amazon CLI 上安装和配置。（ Amazon CLI 将在中步骤 7：安装代 CodeDeploy 理 用于在本地实例上下载和安装 CodeDeploy 代理。）

1. 要在本地实例 Amazon CLI 上安装，请按照Amazon Command Line Interface 用户指南中的开始设置 Amazon CLI中的说明进行操作。

   > **Note**
   >
   > CodeDeploy 用于处理本地实例的命令已在 1.7.19 版本中提供。 Amazon CLI如果您 Amazon CLI 已经安装了的版本，则可以通过调用来检查其版本aws --version。

2. 要在本地实例 Amazon CLI 上配置，请按照Amazon Command Line Interface 用户指南中的配置 Amazon CLI中的说明进行操作。

   > **Important**
   >
   > 在配置时 Amazon CLI（例如，通过调用aws configure命令），请务必指定除中指定的访问权限之外还至少具有以下 Amazon 访问权限的 IAM 用户的私有密钥 ID 和私有访问密钥配置本地实例的先决条件。这样，您就可以在本地实例上下载并安装 CodeDeploy 代理：
   >
   > JSON
   >
   > ```
   > {
   > "Version":"2012-10-17",
   > "Statement" : [
   > {
   > "Effect" : "Allow",
   > "Action" : [
   > "codedeploy:*"
   > ],
   > "Resource" : "*"
   > },
   > {
   > ```

```
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "*"
      ]
    }
  ]
}
```

这些访问权限可以分配给您在步骤 1：为本地实例创建 IAM 用户中创建的 IAM 用户或者其他 IAM 用户。要将这些权限分配给 IAM 用户，请按照步骤 1：为本地实例创建 IAM 用户中的说明操作，并使用这些访问权限而不是该步骤中的权限。

步骤 6：设置 AWS_REGION环境变量（仅限 Ubuntu 服务器和 RHEL）

如果您的本地实例上没有运行 Ubuntu Server 或 RHEL，则跳过此步骤，直接转到步骤 7：安装代 CodeDeploy 理 。

在 Ubuntu 服务器或 RHEL 本地实例上安装代 CodeDeploy 理，并允许该实例在新版本 CodeDeploy 可用时更新代理。为此，您可以将实例上的 `AWS_REGION` 环境变量设置为 CodeDeploy 支持的某个区域的标识符。我们建议您将该值设置为 CodeDeploy 应用程序、部署组和应用程序修订所在的区域（例如`us-west-2`）。有关区域的列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。

要设置环境变量，请从终端调用以下命令：

```
export AWS_REGION=supported-region
```

哪里*`supported-region`*是区域标识符（例如，`us-west-2`）。

步骤 7：安装代 CodeDeploy 理

在本地实例上安装 CodeDeploy 代理：

- 对于 Ubuntu Server 本地实例，请按照安装适用于 Ubuntu Server 的 CodeDeploy 代理中的说明操作，然后返回本页。
- 对于 RHEL 本地实例，请按照安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理中的说明操作，然后返回本页。
- 对于 Windows Server 本地实例，请按照安装适用于 Windows Server 的 CodeDeploy 代理中的说明操作，然后返回本页。

步骤 8：向注册本地实例 CodeDeploy

这一步中的说明假设您从本地实例本身上注册该本地实例。您可以从单独的设备或已 Amazon CLI 安装和配置的实例注册本地实例，如中所述步骤 5：安装和配置 Amazon CLI。

使用 Amazon CLI 向注册本地实例， CodeDeploy 以便可以在部署中使用该实例。

1. 在使用之前 Amazon CLI，您需要在中创建的 IAM 用户的用户 ARN。步骤 1：为本地实例创建 IAM 用户如果您还没有用户 ARN，请调用 get-user 命令，在命令中指定 IAM 用户的名称（使用 `--user-name` 选项）并仅查询用户 ARN（使用 `--query` 和 `--output` 选项）：

```
aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --output text
```

2. 调用 register-on-premises-instance 命令，在命令中指定：

   - 唯一标识本地实例的名称（使用 `--instance-name` 选项）。

     > **Important**
     > 为了帮助标识本地实例，特别是用于调试用途，我们强烈建议您指定能够反映本地实例的某种唯一特性的名称（例如，在适用时可以指定序列号或内部资产标识符）。如果指定 MAC 地址作为名称，请注意 MAC 地址包含 CodeDeploy 不允许使用的字符，例如冒号 (`:`)。有关允许字符的列表，请参阅CodeDeploy 配额。

   - 您在步骤 1：为本地实例创建 IAM 用户中创建的 IAM 用户的用户 ARN（使用 `--iam-user-arn` 选项）。

     例如：

```
aws deploy register-on-premises-instance --instance-name AssetTag12010298EX --
iam-user-arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem
```

步骤 9：标记本地实例

您可以使用 Amazon CLI 或 CodeDeploy 控制台来标记本地实例。（在部署期间CodeDeploy使用本地实例标签来识别部署目标。）

标记本地实例（CLI）

- 调用-premis add-tags-to-ones-instances 命令，指定：

  - 唯一标识本地实例的名称（使用 `--instance-names` 选项）。

- 您要使用的本地实例标签密钥的名称和标签值（使用 `--tags` 选项）。必须同时指定名称和值。 CodeDeploy 不允许仅包含值的本地实例标签。

  例如：

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

标记本地实例（控制台）

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 从 CodeDeploy 菜单中选择本地实例。
3. 在本地实例列表中，选择您要标记的本地实例旁边的箭头。
4. 在标签列表中，选择或输入所需的标签键和标签值。当您输入标签键和标签值之后，将显示另一行。您可以重复此步骤，最多添加 10 个标签。要删除标签，请选择删除图标（ ）。
5. 在您添加标签之后，选择 Update Tags。

步骤 10：将应用程序修订部署到本地实例

现在，您已准备好将应用程序修订部署到已注册和标记的本地实例。

您可以将应用程序修订部署到本地实例，其方式类似于将应用程序修订部署到 Amazon EC2 实例。有关说明，请参阅使用创建部署 CodeDeploy。这些说明包含指向先决条件的链接，其中包括创建应用程序、创建部署组和准备应用程序修订。如果您希望部署简单的示例应用程序修订，可以创建教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）的步骤 2：创建示例应用程序修订中所述的修订。

> **Important**
>
> 如果您在创建以本地实例为目标的部署组时重复使用 CodeDeploy 服务角色，则必须在该服务角色的策略声明`Action`部分中包含`Tag:get*`该服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

步骤 11：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

您可以跟踪本地实例的部署，其方式与跟踪 Amazon EC2 实例部署的方式类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 。

在 CodeDeploy 中管理本地实例操作

将本地实例注册到 CodeDeploy 之后，可按照本部分的说明管理本地实例的操作，例如针对本地实例获取更多相关信息、删除标签以及卸载和取消注册这些本地实例。

主题

- 获取有关单个本地实例的信息
- 获取有关多个本地实例的信息
- 从本地实例中手动删除本地实例标签
- 自动从本地实例卸载 CodeDeploy 代理并删除配置文件
- 自动注销本地实例
- 手动取消注册本地实例

获取有关单个本地实例的信息

您可以按照查看 CodeDeploy 部署详细信息 中的说明获取有关单个本地实例的信息。您可以使用 Amazon CLI 或 CodeDeploy 控制台来获取有关单个本地实例的更多信息。

获取有关单个本地实例的信息（CLI）

- 调用 get-on-premises-instance 命令，在命令中指定唯一标识本地实例的名称（使用 `--instance-name` 选项）：

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

获取有关单个本地实例的信息（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择本地实例。
3. 在本地实例列表中，选择本地实例的名称以查看其详细信息。

获取有关多个本地实例的信息

您可以按照查看 CodeDeploy 部署详细信息 中的说明获取有关本地实例的信息。您可以使用 Amazon CLI 或 CodeDeploy 控制台来获取有关本地实例的更多信息。

获取有关多个本地实例的信息（CLI）

1. 如需本地实例名称的列表，请调用 list-on-premises-instances 命令，在命令中指定：

   - 获取所有已注册本地实例还是所有已注销本地实例的相关信息（分别使用 `--registration-status` 选项和`Registered` 或 `Deregistered`）。如果忽略此项，则将同时返回已注册和已注销本地实例的名称。
   - 是否仅获取使用特定本地实例标签所标记的本地实例的相关信息（使用 `--tag-filters` 选项）。对于每个本地实例标签，请指定 `Key`、`Value` 和 `Type`（始终应为 `KEY_AND_VALUE`）。在各 `Key`、`Value` 和 `Type` 三元组之间使用空格分隔多个本地实例标签。

   例如：

   ```
   aws deploy list-on-premises-instances --registration-status Registered
    --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
    Key=Name,Value=CodeDeployDemo-OnPrem-Beta,Type=KEY_AND_VALUE
   ```

2. 有关详细信息，请调用 batch-get-on-premises-instances 命令，并提供本地实例的名称（使用 `--instance-names` 选项）：

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
 AssetTag09920444EX
```

获取有关多个本地实例的信息（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

> **Note**
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择本地实例。

   此时将显示有关本地实例的信息。

从本地实例中手动删除本地实例标签

通常，当一个标签不再使用时，或者您希望从依赖该标签的任何部署组中删除本地实例时，您可以从本地实例中删除本地实例标签。您可以使用 Amazon CLI 或 Amazon CodeDeploy 控制台从本地实例中删除本地实例标签。

您无需在注销本地实例之前从本地实例中删除本地实例标签。

手动删除本地实例的本地实例标签不会取消注册实例。此操作不会从实例中卸载 CodeDeploy 代理。此操作不会删除实例中的配置文件。此操作不会删除与实例关联的 IAM 用户。

要自动注销本地实例，请参阅自动注销本地实例。

要手动注销本地实例，请参阅手动取消注册本地实例。

要自动卸载 CodeDeploy 代理并删除本地实例中的配置文件，请参阅自动从本地实例卸载 CodeDeploy 代理并删除配置文件。

要仅手动卸载本地实例中的 CodeDeploy 代理，请参阅管理 CodeDeploy 代理操作。

要手动删除关联的 IAM 用户，请参阅删除您的 Amazon 账户中的 IAM 用户。

从本地实例删除本地实例标签（CLI）

- 调用 remove-tags-from-on-premises-instances，并指定：
  - 唯一标识本地实例的名称（使用 `--instance-names` 选项）。
  - 您要删除的标签的名称和值（使用 `--tags` 选项）。

  例如：

```
aws deploy remove-tags-from-on-premises-instances --instance-names
 AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

从本地实例删除本地实例标签（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择本地实例。
3. 在本地实例列表中，选择要从中删除标签的本地实例的名称。
4. 在 Tags（标签）中，选择要删除的每个标签旁的 Remove（删除）。
5. 删除标签之后，选择 Update tags。

自动从本地实例卸载 CodeDeploy 代理并删除配置文件

通常，当您不再计划部署到某个本地实例之后，您可以从该实例卸载 CodeDeploy 代理并删除配置文件。

> **Note**
> 从本地实例自动卸载 CodeDeploy 代理和删除配置文件不会取消注册本地实例。此操作不会取消关联与本地实例关联的任何本地实例标签。此操作不会删除与本地实例关联的 IAM 用户。
> 要自动注销本地实例，请参阅自动注销本地实例。
> 要手动注销本地实例，请参阅手动取消注册本地实例。

要手动取消任何已关联本地实例标签的关联，请参阅从本地实例中手动删除本地实例标签。
要从本地实例手动卸载 CodeDeploy 代理，请参阅管理 CodeDeploy 代理操作。
要手动删除关联的 IAM 用户，请参阅删除您的 Amazon 账户中的 IAM 用户。

从本地实例，使用 Amazon CLI 调用 uninstall 命令。

例如：

```
aws deploy uninstall
```

uninstall 命令执行以下操作：

1. 停止在本地实例上运行的 CodeDeploy 代理。
2. 从本地实例卸载 CodeDeploy 代理。
3. 从本地实例删除配置文件。（对于 Ubuntu 服务器和 RHEL，这是 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。对于 Windows 服务器，这是 `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`。）

自动注销本地实例

通常，当您不再计划部署到某个本地实例之后，您可以注销该实例。在您注销本地实例时，即使本地实例可能属于某个部署组的本地实例标签，该本地实例也不会包括在任何部署中。您可以使用 Amazon CLI 注销本地实例。

> **Note**
>
> 您不能使用 CodeDeploy 控制台注销本地实例。此外，注销本地实例还会删除与本地实例关联的任何本地实例标签。此操作不会从本地实例中卸载 CodeDeploy 代理。此操作不会从本地实例中删除本地实例配置文件。
> 要使用 CodeDeploy 控制台执行此部分中的一些（而非全部）活动，请参阅手动取消注册本地实例的 CodeDeploy 控制台部分。
> 要手动取消任何已关联本地实例标签的关联，请参阅从本地实例中手动删除本地实例标签。
> 要自动卸载 CodeDeploy 代理并删除本地实例中的配置文件，请参阅自动从本地实例卸载 CodeDeploy 代理并删除配置文件。
> 要仅从本地实例手动卸载 CodeDeploy 代理，请参阅管理 CodeDeploy 代理操作。

使用 Amazon CLI 调用 deregister 命令，在命令中指定：

- 唯一标识 CodeDeploy 的本地实例的名称（使用 `--instance-name` 选项）。
- （可选）是否删除与本地实例关联的 IAM 用户。默认行为是删除 IAM 用户。如果您不希望删除与本地实例关联的 IAM 用户，请在命令中指定 `--no-delete-iam-user` 选项。
- （可选）在其中向 CodeDeploy 注册本地实例的 Amazon 区域（使用 `--region` 选项）。这必须是《Amazon Web Services 一般参考》的区域和终端节点中列出的受支持区域之一（例如 `us-west-2`）。如果未指定此选项，则将使用与发出调用的 IAM 用户关联的默认 Amazon 区域。

注销实例并删除用户的示例：

```
aws deploy deregister --instance-name AssetTag12010298EX --region us-west-2
```

注销实例但不删除用户的示例：

```
aws deploy deregister --instance-name AssetTag12010298EX --no-delete-iam-user --region
 us-west-2
```

deregister 命令执行以下操作：

1. 使用 CodeDeploy 取消注册本地实例。
2. 如果指定，则删除与本地实例关联的 IAM 用户。

如果此命令遇到错误，则将显示错误消息，说明您可以如何手动完成剩余步骤。否则，将显示成功消息，说明如何调用 uninstall 命令。

手动取消注册本地实例

通常，当您不再计划部署到某个本地实例之后，您可以注销该实例。您可以使用 Amazon CLI 手动注销本地实例。

手动取消注册本地实例不会卸载 CodeDeploy 代理。此操作不会删除实例中的配置文件。此操作不会删除与实例关联的 IAM 用户。此操作不会删除与实例关联的任何标签。

要自动卸载 CodeDeploy 代理并删除本地实例中的配置文件，请参阅自动从本地实例卸载 CodeDeploy 代理并删除配置文件。

要仅手动卸载 CodeDeploy 代理，请参阅管理 CodeDeploy 代理操作。

要手动删除关联的 IAM 用户，请参阅删除您的 Amazon 账户中的 IAM 用户。

要仅手动删除关联的本地实例标签，请参阅从本地实例中手动删除本地实例标签。

- 调用 deregister-on-premises-instance 命令，在命令中指定唯一标识本地实例的名称（使用 `--instance-name` 选项）：

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

使用 CodeDeploy 查看实例详细信息

您可使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关部署中使用的实例的详细信息。

有关使用 CodeDeploy API 操作查看实例的信息，请参阅 GetDeploymentInstance、ListDeploymentInstances 和 ListOnPremisesInstances。

主题

- 查看实例详细信息（控制台）
- 查看实例详细信息（CLI）

查看实例详细信息（控制台）

查看实例详细信息：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择部署。

> **Note**
>
> 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

3. 要显示部署详细信息，请选择实例的部署 ID。
4. 您可以在部署的页面的 Instance activity（实例活动）部分中查看所有实例。
5. 要查看有关实例的各个部署生命周期事件的信息，请在部署详细信息页上的 Events 列中，选择 View events。

> **Note**
>
> 如果为任何生命周期事件显示 Failed，请在实例详细信息页上，选择 View logs 和/或 View in EC2。您可在排查实例问题中找到问题排查提示。

6. 如果要查看有关某个 Amazon EC2 实例的更多信息，请在实例 ID 列中选择实例的 ID。

查看实例详细信息（CLI）

要使用 Amazon CLI 查看实例详细信息，请调用 `get-deployment-instance` 命令或 `list-deployment-instances` 命令。

要查看有关单个实例的详细信息，请调用 get-deployment-instance 命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用 list-deployments 命令。
- 唯一实例 ID。要获取实例 ID，请调用 list-deployment-instances 命令。

要查看部署中使用的实例的 ID 列表，请调用 list-deployment-instances 命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用 list-deployments 命令。
- （可选）是否仅按实例的部署状态包含特定的实例 ID。（如果未指定，则将列出所有匹配实例的 ID，不管其部署状态如何。）

CodeDeploy 实例运行状况

CodeDeploy 监控部署组中实例的运行状况。如果运行正常的实例数小于部署期间已为部署组指定的最小运行正常实例数，则部署将失败。例如，如果 85% 的实例必须在部署期间保持运行正常，且部署组包含 10 个实例，则只要一个实例的部署失败，整个部署都将失败。这是因为，当一个实例脱机以安装最新的应用程序版本时，可用的运行正常的实例计数已降至 90%。一个出现故障的实例加上另一个脱机实例意味着只有 80% 的实例运行正常且可用。CodeDeploy 将使整个部署失败。

需要牢记的是，为了使整个部署成功，必须满足以下条件：

- CodeDeploy 可以部署到部署中的每个实例。
- 到至少一个实例的部署必须成功。也就是说，即使最小正常运行的主机数值为 0，也是到至少一个实例的部署必须成功（即至少有一个实例必须是正常运行的），才能使整个部署成功。

主题

- 运行状况
- 关于最小运行正常的实例数
- 关于每个可用区最小运行正常的实例数

运行状况

CodeDeploy 为每个实例分配两个运行状况值：修订运行状况 和实例运行状况。

修订运行状况

修订运行状况基于实例上当前安装的应用程序修订。它具有以下状态值：

- 当前：实例上安装的修订与部署组的上次成功部署的修订匹配。
- 旧：实例上安装的修订与旧版应用程序匹配。
- 未知：应用程序修订尚未安装成功到实例上。

实例运行状况

实例运行状况基于针对实例的部署是否成功。它具有以下值：

- 正常：针对实例的上次部署成功。
- 不正常：尝试将修订部署到实例失败，或修订尚未部署到实例。

CodeDeploy 使用修订运行状况和实例运行状况按以下顺序计划针对部署组的实例的部署：

1. “不正常”实例运行状况。
2. “未知”修订运行状况。
3. “旧”修订运行状况。
4. “当前”修订运行状况。

如果整个部署成功，则将更新修订，并更新部署组的运行状况值来反映最新的部署。

- 具有成功部署的所有当前实例将保持“当前”状态。否则，它们将变为“未知”状态。
- 具有成功部署的所有“旧”或“未知”实例将变为“当前”状态。否则，它们将保持“旧”或“未知”状态。
- 所有具有成功部署的正常实例都将保持“正常”状态。否则，它们将变为“不正常”状态。
- 所有具有成功部署的不正常实例都将变为“正常”状态。否则，它们将保持“不正常”状态。

如果整个部署失败或停止：

- CodeDeploy 尝试将应用程序修订部署到的每个实例的实例运行状况将设置为“正常”或“不正常”，具体取决于针对该实例的部署尝试是成功还是失败。
- CodeDeploy 未尝试将应用程序修订部署到的每个实例将保持其当前实例运行状况值。
- 部署组的修订保持不变。

关于最小运行正常的实例数

所需的最小运行正常实例数在部署配置中进行定义。

> **Important**
>
> 在蓝/绿部署期间，部署配置和最小运行正常主机值将应用于替换环境中的实例，而不应用于原始环境中的实例。但是，当原始环境中的实例从负载均衡器取消注册时，只要一个原始实例未能成功取消注册，整个环境就将标记为失败。

CodeDeploy 提供了三种具有常用的最小正常运行主机值的默认部署配置：

默认部署配置名称	预定义的最小正常运行主机值
CodeDeployDefault.OneAtATime	1

默认部署配置名称	预定义的最小正常运行主机值
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

在 在中使用部署配置 CodeDeploy 中，您将发现有关默认部署配置的更多信息。

您可以在 CodeDeploy 中创建自定义部署配置，以定义您自己的最小正常运行主机值。在使用以下操作时，您可以将这些值定义为整数或百分比：

- 在 Amazon CLI 中使用 create-deployment-config 命令时，作为 `minimum-healthy-hosts`。
- 作为 CodeDeploy API 的 MinimumHealthyHosts 数据类型中的 `Value`。
- 在 Amazon CloudFormation 模板中使用 AWS::CodeDeploy::DeploymentConfig 时作为 `MinimumHealthyHosts`。

CodeDeploy 允许您出于以下两大目的为部署指定最小运行正常的实例数。

- 确定整个部署是成功还是失败。如果应用程序修订已成功部署到至少最小数量的运行正常的实例，则部署将成功。
- 确定部署期间为允许继续部署而必须运行正常的实例的数量。

您可以实例数或实例总数百分比的形式为部署组指定最小运行正常的实例数。如果您指定百分比，则在部署开始时，CodeDeploy 将转换等量实例的百分比，并对实例数的小数部分取整。

CodeDeploy 将在部署过程中跟踪部署组的实例的运行状况，并使用部署的指定的最小运行正常的实例数来确定是否继续部署。基本原则是，部署绝对不能导致运行正常的实例数低于您指定的最小数量。此规则的一个例外情况是，当部署组最初具有的运行正常的实例数少于指定的最小运行正常的实例数时。在此情况下，部署过程不会进一步减少运行正常的实例数。

> **Note**
>
> CodeDeploy 将尝试部署到部署组中的所有实例，甚至包括当前处于停止状态的实例。在计算正常运行的最小主机数过程中，处于停止状态的实例和出现故障的实例的影响相同。如果由于处于停止状态的实例过多而导致部署失败，要解决这种问题，请重启实例或更改实例的标签，以将它们从部署组排除出去。

CodeDeploy 通过尝试将应用程序修订部署到部署组的不正常实例来开始部署过程。对于每次成功的部署，CodeDeploy 都会将实例的运行状况更改为运行正常，并将其添加到部署组的运行正常的实例中。然后，CodeDeploy 会将当前运行正常的实例数与指定的最小运行正常的实例数进行比较。

- 如果运行正常的实例数小于或等于指定的最小运行正常的实例数，则 CodeDeploy 将取消部署以确保运行正常的实例数不会随着部署的增加而减少。
- 如果运行正常的实例数比指定的最小运行正常的实例数至少多 1，则 CodeDeploy 会将应用程序修订部署到原始的运行正常的实例组。

如果针对运行正常的实例的部署失败，则 CodeDeploy 会将实例的运行状况更改为“不正常”。当部署进行时，CodeDeploy 将更新当前的运行正常的实例数并将它与指定的最小运行正常的实例数进行比较。如果运行正常的实例数在部署过程中的任何时间点小于指定的最小数量，则 CodeDeploy 将停止部署。此做法可防止下一个部署失败的可能性，并减小运行正常的实例数以使其小于指定的最小数量。

> **Note**
>
> 确保您指定的最小运行正常的实例数小于部署组中的实例总数。如果您指定百分比值，请记住此值将取整。否则，当部署开始时，运行正常的实例数将小于或等于指定的最小运行正常的实例数，并且 CodeDeploy 将立即使整个部署失败。

CodeDeploy 还使用指定的最小运行正常的实例数和实际的运行正常的实例数来确定是否以及如何将应用程序修订部署到多个实例。默认情况下，CodeDeploy 会将应用程序修订部署到尽可能多的实例，而没有使运行正常的实例数小于指定的最小运行正常的实例数的风险。

为了确定一次应部署到的实例数量，CodeDeploy 将使用以下计算方法：

```
[total-hosts] - [minimum-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

例如：

- 如果您的部署组具有 10 个实例，并且您将最小运行正常的实例数设置为 9，则 CodeDeploy 一次将部署到 1 个实例。
- 如果您的部署组具有 10 个实例，并且您将最小运行正常的实例数设置为 3，则 CodeDeploy 将在第一批同时部署到 7 个实例，然后在第二批部署到其余 3 个实例。
- 如果您的部署组具有 10 个实例，并且您将最小运行正常的实例数设置为 0，则 CodeDeploy 将同时部署到 10 个实例。

示例

以下示例假定一个具有 10 个实例的部署组。

最小运行正常的实例数：95%

CodeDeploy 会将最小运行正常的实例数取整为 10 个实例，这等于运行正常的实例数。如果未将修订部署到任何实例，则整个部署将立即失败。

最小运行正常的实例数：9

CodeDeploy 一次将修订部署到一个实例。如果到任何实例的部署失败，运行正常的实例数将等于最小运行正常的实例数，因此 CodeDeploy 会立即让整个部署失败。此规则的例外情况是，如果最后一个实例失败，部署仍将成功。

CodeDeploy 将继续部署，一次部署到一个实例，直至任一部署失败或整个部署完成。如果 10 个部署全都成功，则部署组现在将具有 10 个运行正常的实例。

最小运行正常的实例数：8

CodeDeploy 一次将修订部署到 2 个实例。如果这些部署中的 2 个部署失败，则 CodeDeploy 将立即使整个部署失败。此规则的例外情况是，如果最后一个实例是第二个失败的实例，则部署仍将成功。

最小运行正常的实例数：0

CodeDeploy 一次性将修订部署到整个部署组。至少一个到实例的部署必须成功，整个部署才能成功。如果 0 个实例正常运行，则部署会失败。这是因为如下要求：为了将整个部署标记为成功，在完成整个部署时，至少有一个实例必须是正常运行的，即使最小正常运行的实例数值为 0。

关于每个可用区最小运行正常的实例数

> **Note**
>
> 本节交替使用实例和主机这两个术语来指代 Amazon EC2 实例。

如果您要部署到多个可用区中的实例，可以选择启用zonal configuration功能，这允许 CodeDeploy 一次部署到一个可用区。

启用此功能后，CodeDeploy 将确保正常运行的主机数量保持在“每个区域最小正常运行的主机数”和“最小正常运行的主机数”值之上。如果正常运行的主机数低于任一值，CodeDeploy 将无法在所有可用区内部署。

为了计算一次部署到的主机数量，CodeDeploy 将同时使用“每个区域最小正常运行的主机数”和“最小正常运行的主机数”值。CodeDeploy 将使用 [A] 和 [B] 的计算结果中较小的那一个，其中 [A] 和 [B] 分别为：

```
[A] = [total-hosts] - [min-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

```
[B] = [total-hosts-per-AZ] - [min-healthy-hosts-per-AZ] =
        [number-of-hosts-to-deploy-to-at-once-per-AZ]
```

确定一次要部署到的主机数量后，CodeDeploy 将按此数量分批部署到主机，一次部署一个可用区，区域之间可以选择暂停（或“烘焙时间”）。

示例：

如果您的部署配置如下：

- [total-hosts] 为：200
- [minimum-healthy-hosts] 为：160
- [total-hosts-per-AZ] 为：100
- [minimum-healthy-hosts-per-AZ] 为：50

则...

- [A] = 200 - 160 = 40
- [B] = 100 - 50 = 50
- 40 小于 50

因此，CodeDeploy 将一次部署到 40 台主机。

在这种情况下，部署展开如下：

1. CodeDeploy 部署到第一个可用区：

   a. CodeDeploy 部署到第一批 40 台主机。
   b. CodeDeploy 部署到下一批 40 台主机。
   c. CodeDeploy 部署到剩余的 20 台主机。

   到第一个可用区的部署现已完成。

2. （可选）CodeDeploy 等待到第一个区域的部署进行“烘焙”，具体由监控持续时间或为第一个区域添加监控持续时间设置决定。如果没有问题，CodeDeploy 将继续。
3. CodeDeploy 部署到第二个可用区：
   a. CodeDeploy 部署到第一批 40 台主机。
   b. CodeDeploy 部署到下一批 40 台主机。
   c. CodeDeploy 部署到剩余的 20 台主机。

   到第二个也是最后一个可用区的部署现已完成。

要了解区域配置功能以及如何指定每个可用区正常运行主机的最小数量，请参阅zonal configuration。

在中使用部署配置 CodeDeploy

部署配置是 CodeDeploy 在部署期间使用的一组规则以及成功条件和失败条件。这些规则和条件会有所不同，具体取决于您是部署到 EC2本地计算平台、Lamb Amazon da 计算平台还是 Amazon ECS 计算平台。

EC2/本地计算平台上的部署配置

当您部署到 EC2 /Unlide 计算平台时，部署配置通过使用 “最少运行正常的主机” 值和可选的 “每个区域最少运行正常的主机数” 值来指定在部署期间任何时候必须保持可用的实例数量或百分比。

您可以使用提供的三种预定义部署配置之一， Amazon 也可以创建自定义部署配置。有关创建自定义部署配置的更多信息，请参阅Create a Deployment Configuration。如果您未指定部署配置，请 CodeDeploy 使用 CodeDeployDefault。 OneAtATime 部署配置。

有关如何在部署期间 CodeDeploy 监控和评估实例运行状况的更多信息，请参阅Instance Health。要查看已注册到您的 Amazon 账户的部署配置列表，请参阅View Deployment Configuration Details。

EC2/locide 计算平台的预定义部署配置

下表列出了预定义的部署配置。

> **Note**
>
> 没有支持 zonal configuration 功能（该功能允许您指定每个可用区正常运行的主机数量）的预定义部署配置。如果要使用此功能，您必须创建自己的部署配置。

部署配置	说明
CodeDeployDefault.AllAtOnce	就地部署： 一次性尝试将应用程序修订部署到尽可能多的实例。如果将应用程序修订部署到一个或多个实例，则整个部署的状态将显示为成功。如果尚未向任何实例部署应用程序修订，则整个部署的状态将显示为失败。以九个实例为例，CodeDeployDefault。 AllAtOnce 尝试同时部署

部署配置	说明
	到所有九个实例。如果部署到单个实例成功，则整体部署成功。仅当所有 9 个实例的部署失败时，它才会失败。 蓝/绿部署： • 部署到替代环境：遵循与相同的部署规则 CodeDeployDefault。 AllAtOnce 用于就地部署。 • 流量重新路由：将流量一次路由到替换环境的所有实例中。如果流量成功地重新路由到至少一个实例，则部署成功。如果重新路由到所有实例失败，则部署失败。

部署配置	说明
CodeDeployDefault.HalfAtATime	就地部署： 一次最多可部署到一半实例（小数向下取整）。如果将应用程序修订部署到至少一半实例（小数向下取整），则整个部署成功。否则，部署失败。在包含 9 个实例的示例中，一次部署到最多 4 个实例。如果成功部署到 5 个或更多实例，则整个部署成功。否则，部署失败。 **Note** 如果您要部署到多个 Auto Scaling 组中的实例，则无论这些实例属于哪个 Auto Scaling 组，都 CodeDeploy 将一次部署到多达一半的实例。例如，假设您有两个 Auto Scaling 组 `ASG1` 和 `ASG2`，每个组有 10 个实例。在这种情况下，CodeDeploy 可能会在短时间内部署到 10 个实例，`ASG1`并认为这是成功的，因为它已部署到至少一半的实例。 蓝/绿部署： • 部署到替代环境：遵循与相同的部署规则 CodeDeployDefault。 HalfAtATime 用于就地部署。 • 流量重新路由：每次将流量路由到替换环境的最多半数实例中。如果成功地重新路由到至少半数实例，则部署成功。否则， 失败。

部署配置	说明
CodeDeployDefault.OneAtATime	就地部署： 一次仅将应用程序修订部署到一个实例。 对于包含多个实例的部署组： • 如果已将应用程序修订部署到所有实例，则整个部署成功。此规则的例外情况是，如果无法部署到最后一个实例，则整个部署仍将成功。这是因为一次只 CodeDeploy 允许一个实例使用离线 CodeDeployDefault。 OneAtATime 配置。 • 一旦应用程序修订无法部署到任何实例（但最后一个实例除外），整个部署将失败。 • 在使用 9 个实例的示例中，将一次部署到一个实例。如果部署到前 8 个实例成功，则整体部署成功。如果部署到前 8 个实例中的任何一个实例失败，则整体部署失败。 对于仅包含一个实例的部署组，整个部署仅在成功部署到单个实例时成功。 蓝/绿部署： • 部署到替代环境：遵循与相同的部署规则 CodeDeployDefault。 OneAtATime 用于就地部署。 • 流量重新路由：每次将流量路由到替换环境的一个实例中。如果流量成功地重新路由到所有替换实例，则部署成功。在第一次重新路由失败后，部署失败。此规则的例外情况是，如果最后一个实例无法注册，则整个部署仍将成功。

Amazon ECS 计算平台上的部署配置

部署到 Amazon ECS 计算平台时，部署配置指定如何将流量转移到更新后的 Amazon ECS 任务集。您可以使用金丝雀、线性或all-at-once部署配置来转移流量。有关更多信息，请参阅 部署配置。

您也可以创建自定义 Canary 部署或线性部署配置。有关更多信息，请参阅 Create a Deployment Configuration。

Amazon ECS 计算平台的预定义部署配置

下表列出了可用于 Amazon ECS 部署的预定义配置。

> **Note**
>
> 如果您使用的是网络负载均衡器，仅支持 `CodeDeployDefault.ECSAllAtOnce` 预定义的部署配置。

部署配置	说明
CodeDeployDefault。 ECSLinear10 PercentEvery 1分钟	每分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault。 ECSLinear10 PercentEvery 3 分钟	每隔 3 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault。 ECSCanary10% 5 分钟	在第一次增量中转移 10% 的流量。其余 90% 部署在五分钟后进行转移。
CodeDeployDefault。 ECSCanary10Percent15 分钟	在第一次增量中转移 10% 的流量。剩下的 90% 将在 15 分钟后部署。
CodeDeployDefault.ECSAllAtOnce	将所有流量一次性转移到更新后的 Amazon ECS 容器。

部署的 Amazon CloudFormation blue/green 部署配置 (Amazon ECS)

当您通过 Amazon CloudFormation 蓝/绿部署部署到 Amazon ECS 计算平台时，部署配置会指定如何将流量转移到更新后的 Amazon ECS 容器。您可以使用金丝雀、线性或all-at-once部署配置来转移流量。有关更多信息，请参阅 部署配置。

对于 Amazon CloudFormation 蓝/绿部署，您无法创建自己的自定义金丝雀或线性部署配置。有关使用 Amazon CloudFormation 管理 Amazon ECS blue/green 部署的 step-by-step说明，请参阅Amazon CloudFormation 用户指南 Amazon CloudFormation中的通过 CodeDeploy 使用自动 blue/green 部署 ECS。

> **Note**
>
> 中国（北京）和中国（宁夏）区域不支持使用 Amazon CloudFormation 管理 Amazon ECS blue/green 部署。

Amazon Lambda 计算平台上的部署配置

当您部署到 Amazon Lambda 计算平台时，部署配置会指定流量转移到应用程序中新 Lambda 函数版本的方式。您可以使用金丝雀、线性或all-at-once部署配置来转移流量。有关更多信息，请参阅 部署配置。

您也可以创建自定义 Canary 部署或线性部署配置。有关更多信息，请参阅 Create a Deployment Configuration。

Amazon Lambda 计算平台的预定义部署配置

下表列出了可用于 Amazon Lambda 部署的预定义配置。

部署配置	说明
CodeDeployDefault。 LambdaCanary10% 5 分钟	在第一次增量中转移 10% 的流量。其余 90% 部署在五分钟后进行转移。
CodeDeployDefault。 LambdaCanary10% 10 分钟	在第一次增量中转移 10% 的流量。其余 90% 部署在 10 分钟后进行转移。

部署配置	说明
CodeDeployDefault。 LambdaCanary10% 15 分钟	在第一次增量中转移 10% 的流量。剩下的 90% 将在 15 分钟后部署。
CodeDeployDefault。 LambdaCanary10% 30 分钟	在第一次增量中转移 10% 的流量。其余 90% 部署在 30 分钟后进行转移。
CodeDeployDefault。 LambdaLinear10 PercentEvery 1Minute	每分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault。 LambdaLinear10 PercentEvery 2 分钟	每隔 2 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault。 LambdaLinear10 PercentEvery 3分钟	每隔 3 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault。 LambdaLinear10 PercentEvery 10分钟	每隔 10 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.LambdaAllAtOnce	所有流量一次性转移到更新后的 Lambda 函数。

主题

- Create a Deployment Configuration
- View Deployment Configuration Details
- Delete a Deployment Configuration

使用 CodeDeploy 创建部署配置

如果您不希望使用 CodeDeploy 提供的任一默认部署配置，可按照以下说明创建自己的配置。

可以使用 CodeDeploy 控制台、Amazon CLI、CodeDeploy API 或 Amazon CloudFormation 模板创建自定义部署配置。

有关使用 Amazon CloudFormation 模板创建部署配置的信息，请参阅 Amazon CloudFormation 模板供 CodeDeploy参考。

主题

- 创建部署配置（控制台）
- 使用 CodeDeploy 创建部署配置（Amazon CLI）

创建部署配置（控制台）

按照以下说明，使用 Amazon 控制台创建部署配置。

使用控制台在 CodeDeploy 中创建部署配置

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，选择部署配置。

   将显示一个内置部署配置列表。

3. 选择创建部署配置。
4. 在部署配置名称中，输入部署配置的名称。例如 **my-deployment-config**。
5. 在计算平台下，选择以下选项之一：
   - EC2/本地
   - Amazon Lambda
   - Amazon ECS：
6. 请执行以下操作之一：
   - 如果您选择了 EC2/本地：
     1. 在正常运行的最少主机数下，指定在部署期间任何时候都必须保持可用的实例的数量或百分比。有关部署期间 CodeDeploy 如何监控和评估实例运行状况的更多信息，请参阅Instance Health。
     2. （可选）在区域配置下，选择启用区域配置，让 CodeDeploy 将您的应用程序一次部署到 Amazon 区域内的一个可用区。一次部署到一个可用区后，随着您对部署性能和可

行性的信心逐渐增强，就可以向越来越多的受众展示自己的部署。如果您未启用区域配置，CodeDeploy 会将您的应用程序部署到某个区域中随机选择的主机上。

如果您启用了区域配置功能，请注意以下事项：

- 只有对 Amazon EC2 实例的就地部署才支持区域配置功能。（不支持蓝绿部署和本地实例。）有关就地部署的更多信息，请参阅部署类型。
- 预定义的部署配置不支持区域配置功能。要使用区域配置，必须按此处所述创建自定义部署配置。
- 如果 CodeDeploy 需要回滚部署，CodeDeploy 将在随机主机上执行回滚操作。（CodeDeploy 不会像您期望的那样一次回滚一个区域。）之所以选择这种回滚行为，是出于性能方面的考虑。有关回滚的更多信息，请参阅使用 CodeDeploy 重新部署和回滚部署。

3. 如果选中了启用区域配置复选框，可以选择指定以下选项：
   - （可选）在监控持续时间中，指定 CodeDeploy 在完成对可用区的部署后必须等待的时间段（以秒为单位）。CodeDeploy 将在开始部署到下一个可用区之前等待这段时间。请考虑添加监控持续时间，以便让部署在下一个可用区内发布之前，有时间在一个可用区中证明自己（或“烘焙”）。如果您未指定监控持续时间，CodeDeploy 会立即开始部署到下一个可用区。有关监控持续时间设置如何发挥作用的更多信息，请参阅 关于每个可用区最小运行正常的实例数。
   - （可选）选择为第一个区域添加监控持续时间以设置仅适用于第一个可用区的监控持续时间。如果您想为第一个可用区留出额外的烘焙时间，可以设置此选项。如果您未在添加第一个区域监控持续时间中指定值，CodeDeploy 将为第一个可用区使用监控持续时间值。
   - （可选）在每个区域正常运行的最少主机数下，指定在部署期间每个可用区必须保持可用的实例的数量或百分比。选择 FLEET_PERCENT 以指定一个百分比，或选择 HOST_COUNT 以指定一个数字。此字段将与正常运行的最少主机数字段共同发挥作用。有关更多信息，请参阅 关于每个可用区最小运行正常的实例数。

     如果您未在每个区域正常运行的最少主机数下指定值，CodeDeploy 将使用默认值 0%。

- 如果您选择了 Amazon Lambda 或 Amazon ECS：
  1. 对于类型，请选择线性或金丝雀。
  2. 在分步和间隔字段中，执行以下任一操作：
     - 如果您选择了金丝雀，请为分步输入要转移的流量百分比，介于 1 和 99 之间。这是在第一次递增中转移的流量百分比。剩余的流量将在选定的时间间隔后在第二次递增中转移。

       对于间隔，请输入第一次和第二次流量转移之间的分钟数。

- 如果您选择了线性，请为分步输入要转移的流量百分比，介于 1 和 99 之间。这是在每个间隔开始时转移的流量百分比。

  对于间隔，请输入每两次增量转移之间的分钟数。

7. 选择创建部署配置。

   现在，您就有了可以与部署组关联的部署配置。

使用 CodeDeploy 创建部署配置（Amazon CLI）

要使用 Amazon CLI 创建部署配置，请调用 create-deployment-config 命令。

以下示例创建了一个名为 `ThreeQuartersHealthy` 的 EC2/本地部署配置，此配置要求 75% 的目标实例在部署期间保持正常运行状态：

```
aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy --
minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

以下示例创建了一个名为 `300Total50PerAZ` 的 EC2/本地部署配置，此配置要求在每个部署中，总共要有 300 个目标实例保持正常运行状态，每个可用区要有 50 个目标实例保持正常运行状态。它还会将监控持续时间设置为 1 小时。

```
aws deploy create-deployment-config --deployment-config-name 300Total50PerAZ
 --minimum-healthy-hosts type=HOST_COUNT,value=300 --zonal-config
 '{"monitorDurationInSeconds":3600,"minimumHealthyHostsPerZone":
{"type":"HOST_COUNT","value":50}}'
```

以下示例创建了一个名为 `Canary25Percent45Minutes` 的 Amazon Lambda 部署配置。它使用 Canary 流量转移在第一次递增中转移 25% 的流量。其余 75% 在 45 分钟后进行转移：

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform Lambda
```

以下示例创建了一个名为 `Canary25Percent45Minutes` 的 Amazon ECS 部署配置。它使用 Canary 流量转移在第一次递增中转移 25% 的流量。其余 75% 在 45 分钟后进行转移：

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform ECS
```

使用 CodeDeploy 查看部署配置详细信息

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关与您的 Amazon 账户关联的部署配置的详细信息。有关预定义的 CodeDeploy 部署配置的说明，请参阅 EC2/locide 计算平台的预定义部署配置。

主题

- 查看部署配置详细信息（控制台）
- 查看部署配置（CLI）

查看部署配置详细信息（控制台）

要使用 CodeDeploy 控制台查看部署配置名称的列表，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择部署配置。

   在这里，您可以查看每个部署配置的部署配置名称和条件。

> **Note**
>
> 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

查看部署配置（CLI）

要使用 Amazon CLI 查看部署配置详细信息，请调用 `get-deployment-config` 命令或 `list-deployment-configs` 命令。

要查看有关单个部署配置的详细信息，请调用 get-deployment-config 命令，并指定唯一的部署配置名称。

要查看有关多个部署配置的详细信息，请调用 list-deployments 命令。

使用 CodeDeploy 删除部署配置

您可以使用 Amazon CLI 或 CodeDeploy API 删除与您的 Amazon 账户关联的自定义部署配置。您无法删除内置部署配置，例如 `CodeDeployDefault.AllAtOnce`、`CodeDeployDefault.HalfAtATime` 和 `CodeDeployDefault.OneAtATime`。

> **Warning**
>
> 您无法删除仍在使用中的自定义部署配置。如果您删除未使用的自定义部署配置，则不再能够将它与新部署和新部署组关联。并且无法撤消。

要使用 Amazon CLI 删除部署配置，请调用 delete-deployment-config 命令，并指定部署配置名称。要查看部署配置名称的列表，请调用 list-deployment-configs 命令。

以下示例删除名为 ThreeQuartersHealthy 的部署配置。

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

在 CodeDeploy 中使用应用程序

配置实例后，必须先在 CodeDeploy 中创建应用程序，然后才能部署修订。应用程序只是一个名称或容器，CodeDeploy 使用此名称或容器来确保在部署期间引用正确的修订、部署配置和部署组。

使用下表中的信息完成接下来的步骤：

计算平台	场景	下一步的信息
EC2/本地	我尚未创建实例。	请参阅使用适用于 CodeDeploy 的实例，然后返回此页。
EC2/本地	我已创建实例，但尚未完成标记。	请参阅Tagging Instances for Deployments，然后返回此页。
EC2/本地、Amazon Lambda 和 Amazon ECS	我尚未创建应用程序。	请参阅 使用创建应用程序 CodeDeploy。
EC2/本地、Amazon Lambda 和 Amazon ECS	我已创建应用程序，但尚未创建部署组。	请参阅使用创建部署组 CodeDeploy。
EC2/本地、Amazon Lambda 和 Amazon ECS	我已创建应用程序和部署组，但尚未创建应用程序修订。	请参阅正在处理的应用程序修订版 CodeDeploy。
EC2/本地、Amazon Lambda 和 Amazon ECS	我已创建应用程序和部署组，并且已上传我的应用程序修订。我已做好部署准备。	请参阅使用创建部署 CodeDeploy。

主题

- 使用创建应用程序 CodeDeploy
- 使用 CodeDeploy 查看应用程序详细信息
- 创建通知规则
- 重命名 CodeDeploy 应用程序
- 在 CodeDeploy 中删除应用程序

使用创建应用程序 CodeDeploy

应用程序只是一个名称或容器，用于 CodeDeploy 确保在部署期间引用正确的修订版、部署配置和部署组。您可以使用 CodeDeploy 控制台、 Amazon CLI CodeDeploy APIs、或 Amazon CloudFormation 模板来创建应用程序。

您的代码或应用程序修订通过称为部署的过程安装到实例中。 CodeDeploy 支持两种类型的部署：

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2 /Unlide 计算平台的部署才能使用就地部署。有关就地部署的更多信息，请参阅就地部署概述。
- 蓝绿部署：部署的行为取决于使用的计算平台：
  - Blue/green on an EC2/On-本地计算平台：使用以下步骤将部署组（原始环境）中的实例替换为另一组实例（替换环境）：
    - 为替换环境配置实例。
    - 在替换实例上安装最新的应用程序修订。
    - 对于应用程序测试和系统验证等活动，可以选择等待时间。
    - 替换环境中的实例在一个或多个 ELB 负载均衡器中注册，从而导致流量被重新路由到这些负载均衡器。原始环境中的实例已注销，可以终止或继续运行以用于其他用途。

    > **Note**
    > 如果您使用 EC2 /本地计算平台，请注意 blue/green 部署仅适用于 Amazon EC2 实例。

  - 或 Amazon Lambda Amazon ECS 计算平台上的蓝/绿：流量根据金丝雀、线性或all-at-once部署配置逐渐移动。
  - 蓝/绿部署通过 Amazon CloudFormation：作为 Amazon CloudFormation 堆栈更新的一部分，流量将从您当前的资源转移到更新的资源。当前，仅支持 ECS blue/green 部署。

  有关 blue/green 部署的更多信息，请参阅蓝绿部署概述。

使用 CodeDeploy 控制台创建应用程序时，可以同时配置其第一个部署组。使用创建应用程序时，可以在单独的步骤中创建其第一个部署组。 Amazon CLI

要查看已注册到您的 Amazon 账户的应用程序列表，请参阅使用 CodeDeploy 查看应用程序详细信息。有关使用 Amazon CloudFormation 模板创建应用程序的信息，请参阅Amazon CloudFormation 模板供 CodeDeploy参考。

这两个部署类型不适用于所有目标。下表列出了哪些部署类型与到三种部署目标类型的部署一起使用。

部署目标	就地	蓝/绿
Amazon EC2	支持	是
本地	是	否
无服务器 Lamb Amazon da 函数	否	是
Amazon ECS 应用程序	否	是

主题

- 为就地部署创建应用程序（控制台）
- 为蓝绿部署创建应用程序（控制台）
- 为 Amazon ECS 服务部署创建应用程序（控制台）
- 为 Amazon Lambda 函数部署创建应用程序（控制台）
- 创建应用程序（CLI）

为就地部署创建应用程序（控制台）

要使用 CodeDeploy 控制台为就地部署创建应用程序，请执行以下操作：

> **Warning**
>
> 以下情况下请勿按照这些步骤操作：
>
> - 您尚未准备好用于 CodeDeploy 部署的实例。要设置您的实例，请按照使用适用于 CodeDeploy 的实例中的说明操作，然后执行本主题中的步骤。
> - 您需要创建使用自定义部署配置的应用程序，但您尚未创建部署配置。按照Create a Deployment Configuration中的说明操作，然后执行本主题中的步骤。

- 您没有具有最低要求信任 CodeDeploy 和权限的信任服务角色。要创建和配置具有所需权限的服务角色，请按照步骤 2：为创建服务角色 CodeDeploy中的说明操作，然后返回到本主题中的相应步骤。
- 您想在 ELB 中为就地部署选择 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer，但尚未创建。

要使用 CodeDeploy 控制台为就地部署创建应用程序，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开 Deploy（部署），然后选择 Getting started（开始使用）。
3. 选择创建应用程序。
4. 在 Application name（应用程序名称）中，输入您的应用程序的名称。
5. 从 “计算平台” 中选择 EC2/本地。
6. 选择创建应用程序。
7. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
8. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

   > **Note**
   > 如果您要使用其他部署组中使用的相同设置（包括部署组名称、标签、Amazon A EC2 uto Scaling 组名称或两者；以及部署配置），请在此页面上指定这些设置。尽管此新部署组和现有部署组具有相同的名称，但仍将其 CodeDeploy 视为单独的部署组，因为它们各自与不同的应用程序关联。

9. 在 Service role (服务角色) 中，选择向 CodeDeploy 授予访问您的目标实例的权限的服务角色。
10. 在部署类型中，选择就地。
11. 在 Environment configuration（环境配置）中，选择以下任一项：

a. Amazon A EC2 uto Scaling 群组：输入或选择要将您的应用程序修订部署到的 Amazon A EC2 uto Scaling 群组的名称。当作为 Amazon Aut EC2 o Scaling 组的一部分启动新的亚马逊 EC2实例时， CodeDeploy 可以自动将您的修订部署到新实例。您最多可以向一个部署组添加 10 个 Amazon A EC2 uto Scaling 群组。

b. Amazon EC2 实例或本地实例：在 “密钥” 和 “值” 字段中，输入您用来标记实例的键值对的值。一个标签组中最多可标记 10 对键值对。

   i. 您可以在 “值” 字段中使用通配符来识别以特定模式标记的所有实例，例如类似的 Amazon EC2 实例、成本中心和组名等。例如，如果您在 “键” 字段中选择 “名称”，然后在 “值” 字段**GRP-*a**中输入，则会 CodeDeploy 标识符合该模式的所有实例**GRP-1a**，例如**GRP-2a**、和**GRP-XYZ-a**。

   ii. Value（值）字段区分大小写。

   iii. 要从列表中删除键值对，请选择 Remove tag（删除图标）。

   当 CodeDeploy 找到与每个指定的键值对或 Amazon A EC2 uto Scaling 组名匹配的实例时，它会显示匹配的实例的数量。选择该数字可查看有关这些实例的更多信息。

   如果您希望更精细地确定部署实例的条件，请选择 Add tag group 创建标签组。您最多可以创建 3 个标签组，每组最多可包含 10 个键值对。如果在部署组中使用多个标签组，只有所有标签组均标记出的实例才会包含在部署组中。也就是说，只有与每组中至少一个标签匹配的实例才会包含在部署组中。

   有关使用标签组优化部署组的信息，请参阅Tagging Instances for Deployments。

12. 在 Deployment settings（部署设置）中，选择一个部署配置以控制将应用程序部署到实例的速率，如一次部署一个或一次全部部署。有关部署配置的更多信息，请参阅在中使用部署配置 CodeDeploy。

13. （可选）在负载均衡器中，选择启用负载平衡，然后从列表中选择经典负载均衡器、Application Load Balancer 目标组和 Network Load Balancer 目标组，以便在 CodeDeploy 部署期间管理实例的流量。您最多可以选择 10 个经典负载均衡器和 10 个目标组，总共可以选择 20 个项目。确保您要部署到的 Amazon EC2 实例已注册到选定的负载均衡器（传统负载均衡器）或目标组（应用程序负载均衡器和网络负载均衡器）。

    在部署期间，原始实例将从选定的负载均衡器和目标组中注销，以防止在部署期间将流量路由到这些实例。部署完成后，将向所有 选定的经典负载均衡器和目标组重新注册每个实例。

有关用于 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing。

14. （可选）展开 “高级”，然后配置要包含在部署中的任何选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

    有关更多信息，请参阅 为部署组配置高级选项。

15. 选择 Create deployment group（创建部署组）。

下一步是准备要部署到应用程序和部署组的修订。有关说明，请参阅正在处理的应用程序修订版 CodeDeploy。

为蓝绿部署创建应用程序（控制台）

要使用 CodeDeploy 控制台为 blue/green 部署创建应用程序，请执行以下操作：

> **Note**
>
> 部署到 Amazon Lambda 计算平台始终是一种 blue/green 部署。您不需要指定部署类型选项。

> **Warning**
>
> 以下情况下请勿按照这些步骤操作：
>
> - 在 blue/green 部署过程中，您没有要替换的安装了 CodeDeploy 代理的实例。要设置您的实例，请按照使用适用于 CodeDeploy 的实例中的说明操作，然后执行本主题中的步骤。
> - 您需要创建使用自定义部署配置的应用程序，但您尚未创建部署配置。按照Create a Deployment Configuration中的说明操作，然后执行本主题中的步骤。
> - 您没有至少信任中描述 CodeDeploy 的信任和权限的服务角色步骤 2：为创建服务角色 CodeDeploy。要创建和配置服务角色，请按照步骤 2：为创建服务角色 CodeDeploy中的说明操作，然后执行本主题中的步骤。
> - 您尚未在 Elastic Load Balancing 中创建用于在替代环境中注册实例的经典负载均衡器、应用程序负载均衡器或网络负载均衡器。有关更多信息，请参阅 在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

> **Note**
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开 Deploy（部署），然后选择 Getting started（开始使用）。
3. 在 Application name（应用程序名称）中，输入您的应用程序的名称。
4. 从 “计算平台” 中选择 EC2/本地。
5. 选择创建应用程序。
6. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
7. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

> **Note**
> 如果您想使用其他部署组中使用的相同设置（包括部署组名称标签、Amazon A EC2 uto Scaling 组名称和部署配置），请在此页面上选择这些设置。虽然这个新的部署组与现有部署组同名， CodeDeploy 仍认为它们是两个部署组，因为与它们关联的应用程序不同。

8. 在 Service role (服务角色) 中，选择向 CodeDeploy 授予访问您的目标实例的权限的服务角色。
9. 在 Deployment type（部署类型）中选择 Blue/green（蓝/绿）。
10. 在 Environment configuration 中，选择为替换环境提供实例的方法：
    a. 自动复制 Amazon A EC2 uto Scaling 群组：通过复制您指定的群组来 CodeDeploy 创建 Amazon A EC2 uto Scaling 群组。
    b. Manually provision instances：在创建部署前，您不会为替换环境指定实例。您必须在启动部署前创建实例。您应于此处指定要替换的实例。
11. 根据您在步骤 10 中的选择，请执行以下操作之一：
    - 如果您选择了 “自动复制 Amazon A EC2 uto Scaling 组”：在 Amazon A EC2 uto Scal ing 组中，选择或输入要用作替换环境中实例的 A EC2 mazon A EC2 uto Scaling 组模板的 Amazon Auto Scaling 组的名称。您选择的 Amazon A EC2 uto Scaling 组中当前运行良好的实例数量是在您的替代环境中创建的。

- 如果您选择手动配置实例：启用 Amazon A EC2 uto Scaling 组、Amazon EC2 实例或两者来指定要添加到此部署组的实例。输入 Amazon EC2 标签值或 Amazon A EC2 uto Scaling 组名称，以识别您的原始环境中的实例（即您要替换的实例或正在运行当前应用程序修订版的实例）。

12. 在负载均衡器中，选择启用负载平衡，然后从列表中选择您要注册替换的 Amazon EC2 实例的传统负载均衡器、Application Load Balancer 目标组和网络负载均衡器目标组。每个替换实例都将在所有 选定的经典负载均衡器和目标组中注册。您最多可以选择 10 个经典负载均衡器和 10 个目标组，总共可以选择 20 个项目。

    根据您选择的流量重新路由和部署配置设置，流量将从原始实例重新路由到替换实例。

    有关用于 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing。

13. 在 Deployment settings 中，查看用于将流量重新路由到替换环境的默认选项、要用于部署的部署配置以及在部署后处理原始环境中的实例的方式。

    如果您要更改设置，请继续执行下一步。否则，请跳至步骤 15。

14. 要更改部署的 blue/green 部署设置，请更改以下任一设置。

设置	选项
Traffic rerouting	• 立即重新路由流量：一旦预置替换环境中的实例并在这些实例上安装最新应用程序修订，这些实例将自动注册到指定的负载均衡器和目标组，从而使流量重新路由到它们。原始环境中的实例随后将取消注册。 • 我将选择是否重新路由流量：替换环境中的实例不会注册到指定的负载均衡器和目标组，除非您手动重新路由流量。如果在没有重新路由流量的情况下经过了指定的等待时间，部署状态将更改为“Stopped”。
部署配置	选择替换环境中的实例向负载均衡器和目标组注册的速度，如一次一个或一次全部。

设置	选项
	Note 将流量成功路由到替换环境后，无论选择了哪个部署配置，原始环境中的实例都将一次全部取消注册。 有关更多信息，请参阅 在中使用部署配置 CodeDeploy。
Original instances	• 终止部署组中的原始实例：将流量重新路由到替换环境后，已从负载均衡器和目标组取消注册的实例将在您指定的一段等待时间后终止。 • 继续运行部署组中的原始实例：将流量重新路由到替换环境后，已从负载均衡器和目标组取消注册的实例将继续运行。

15. （可选）在 “高级” 中，配置要包含在部署中的选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

    有关在部署组中指定高级选项的信息，请参阅为部署组配置高级选项。

16. 选择 Create deployment group（创建部署组）。

下一步是准备要部署到应用程序和部署组的修订。有关说明，请参阅正在处理的应用程序修订版 CodeDeploy。

为 Amazon ECS 服务部署创建应用程序（控制台）

您可以使用 CodeDeploy 控制台为 Amazon ECS 服务部署创建应用程序。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

> **Note**
>
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择开始。
3. 在创建应用程序页面上，选择使用 CodeDeploy。
4. 在 Application name（应用程序名称）中，输入您的应用程序的名称。
5. 在计算平台中，选择 Amazon ECS。
6. 选择创建应用程序。
7. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。有关为 Amazon ECS 部署创建部署组所需内容的更多信息，请参阅在开始 Amazon ECS 部署之前。
8. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

> **Note**
>
> 如果您需要使用其他部署组中使用的相同设置（包括部署组名称和部署配置），请在此页上选择这些设置。尽管此新组和现有组可能具有相同的名称，但仍将其 CodeDeploy 视为单独的部署组，因为每个部署组都与单独的应用程序相关联。

9. 在服务角色中，选择一个授予 CodeDeploy对 Amazon ECS 访问权限的服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。
10. 从负载均衡器名称中，选择将流量提供给 Amazon ECS 服务的负载均衡器的名称。
11. 在生产侦听器端口中，选择将生产流量路由至您的 Amazon ECS 服务的侦听器的端口和协议。
12. （可选）从测试侦听器端口中，选择测试侦听器的端口和协议，该侦听器在部署期间将流量路由至 Amazon ECS 服务中的替换任务集。您可以在挂钩期间运行 AppSpec 的文件中指定一个或多个 Lambda 函数。`AfterAllowTestTraffic`这些函数可以运行验证测试。如果验证测试失败，将触发部署回滚。如果验证测试成功，则会触发部署生命周期中的下一个挂钩 `BeforeAllowTraffic`。如果未指定测试侦听器端口，则 `AfterAllowTestTraffic` 挂接期间不会发生任何事情。有关更多信息，请参阅 AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。
13. 从目标组 1 名称和目标组 2 名称中，选择部署期间用于路由流量的目标组。 CodeDeploy 将一个目标组绑定到您的 Amazon ECS 服务的原始任务集，将另一个目标组绑定到其替换任务集。有关更多信息，请参阅应用程序负载均衡器的目标组。

14. 选择立即重新路由流量或指定重新路由流量的时间，以确定何时将流量重新路由到更新后的 Amazon ECS 服务。

    如果您选择立即重新路由流量，则部署会在预置替换任务集后自动重新路由流量。

    如果选择指定重新路由流量的时间，则选择在成功预置替换任务集后要等待的天数、小时数和分钟数。在这段等待时间内，将在 AppSpec 文件中指定的 Lambda 函数中执行验证测试。如果在重新路由流量之前等待时间已过，则部署状态将更改为 `Stopped`。

15. 对于原始修订终止，请选择成功部署后，在终止 Amazon ECS 服务中的原始任务集之前要等待的天数、小时数和分钟数。

16. （可选）在 “高级” 中，配置要包含在部署中的任何选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

    有关更多信息，请参阅 为部署组配置高级选项。

为 Amazon Lambda 函数部署创建应用程序（控制台）

您可以使用 CodeDeploy 控制台为 Amazon Lambda 函数部署创建应用程序。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择开始。
3. 在创建应用程序页面上，选择使用 CodeDeploy。
4. 在 Application name（应用程序名称）中输入您的应用程序的名称。
5. 从 Compute platform（计算平台）中，选择 Amazon Lambda。
6. 选择创建应用程序。
7. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
8. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

> **Note**
>
> 如果您需要使用其他部署组中使用的相同设置（包括部署组名称和部署配置），请在此页上选择这些设置。尽管此新部署组和现有部署组可能具有相同的名称，但仍将其 CodeDeploy 视为单独的部署组，因为每个部署组都与单独的应用程序关联。

9. 在服务角色中，选择授予 CodeDeploy访问权限的服务角色 Amazon Lambda。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。
10. 如果您要使用预定义的部署配置，请从 Deployment configuration（部署配置）中选择一个然后跳至步骤 12。要创建自定义配置，请继续执行下一步。

    有关部署配置的更多信息，请参阅Amazon Lambda 计算平台上的部署配置。
11. 要创建自定义配置，请选择 Create deployment configuration（创建部署配置）然后执行以下操作：

    a. 对于 Deployment configuration name（部署配置名称），输入配置的名称。

    b. 从 Type（类型）中，选择配置类型。如果您选择 Canary，则流量将通过两次递增进行转移。如果您选择 Linear，则流量使用相等的递增转移，在每次递增之间的分钟数相同。

    c. 对于 Step，输入将要转移的流量百分比，介于 1 和 99 之间。如果您的配置类型是 Canary，则这是在第一次递增中转移的流量百分比。剩余的流量将在选定的时间间隔后在第二次递增中转移。如果您的配置类型是 Linear，则这是在每个间隔开始时转移的流量百分比。

    d. 在 Interval（间隔）中，输入分钟数。如果您的配置类型是 Canary，则这是第一次和第二次流量转移之间间隔的分钟数。如果您的配置类型是 Linear（线性），则这是每次增量流量转移之间间隔的分钟数。

    > **Note**
    >
    > 一次 Amazon Lambda 部署的最大时长为两天，即 2,880 分钟。因此，为 Canary 配置的 Interval 指定的最大值为 2,800 分钟。线性配置的最大值取决于 Step 的值。例如，如果线性流量转移的步长百分比是 25%，则有四次流量转移。最大时间间隔值是 2,880 除以 4，即 720 分钟。

    e. 选择 Create deployment configuration（创建部署配置）。
12. （可选）在 “高级” 中，配置要包含在部署中的任何选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

有关更多信息，请参阅 为部署组配置高级选项。

13. 选择 Create deployment group（创建部署组）。

创建应用程序（CLI）

要使用创建应用程序，请调用 create-application 命令，指定一个唯一代表该应用程序的名称。 Amazon CLI （在一个 Amazon 账户中，每个区域只能使用一次 CodeDeploy 应用程序名称。 您可以在不同的区域重复使用应用程序名称。）

使用创建应用程序后，下一步是创建一个部署组，指定要向其部署修订的实例。 Amazon CLI 有关说明，请参阅使用创建部署组 CodeDeploy。

创建部署组后，下一步是准备要部署到应用程序和部署组的修订。有关说明，请参阅正在处理的应用程序修订版 CodeDeploy。

使用 CodeDeploy 查看应用程序详细信息

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关与您的 Amazon 账户关联的所有应用程序的详细信息。

主题

- 查看应用程序详细信息（控制台）
- 查看应用程序详细信息（CLI）

查看应用程序详细信息（控制台）

要使用 CodeDeploy 控制台查看应用程序详细信息，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择开始。
3. 要查看其他应用程序的详细信息，请在列表中选择该应用程序的名称。

查看应用程序详细信息（CLI）

要使用 Amazon CLI 查看应用程序详细信息，请调用 get-application 命令、batch-get-application 命令或 list-applications 命令。

要查看有关单个应用程序的详细信息，请调用 get-application 命令，并指定应用程序名称。

要查看有关多个应用程序的详细信息，请调用 batch-get-applications 命令，并指定多个应用程序名称。

要查看应用程序名称的列表，请调用 list-applications 命令。

创建通知规则

您可以使用通知规则在部署应用程序发生更改（如部署成功和失败）时通知用户。通知规则指定用于发送通知的事件和 Amazon SNS 主题。有关更多信息，请参阅什么是通知？

您可以使用控制台或 Amazon CLI 为 Amazon CodeDeploy 创建通知规则。

创建通知规则（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy/。
2. 选择 Application（应用程序），然后选择要在其中添加通知的应用程序。
3. 在应用程序页面上，选择 Notify（通知），然后选择 Create notification rule（创建通知规则）。您也可以转到应用程序的 Settings（设置）页面，然后选择 Create notification rule（创建通知规则）。
4. 在 Notification name（通知名称）中，输入规则的名称。
5. 如果您只想在通知中包含提供给 Amazon EventBridge 的信息，则在 Detail type（详细信息类型）中，选择 Basic（基本）。如果您希望包含提供给 Amazon EventBridge 的信息以及 CodeDeploy 或通知管理器可能提供的信息，请选择完整。

   有关更多信息，请参阅了解通知内容和安全性。
6. 在 Events that trigger notifications（触发通知的事件）中，选择要为其发送通知的事件。

类别	事件
部署	失败

类别	事件
	成功 Started

7. 在 Targets（目标）中，选择 Create SNS topic（创建 SNS 主题）。

> **Note**
> 创建主题时，将为您应用允许 CodeDeploy 向主题发布事件的策略。使用专为 CodeDeploy 通知创建的主题还有助于确保仅将用户添加到您要查看有关此部署应用程序的通知的该主题的订阅列表中。

在 codestar-notifications- 前缀后面，输入主题的名称，然后选择 Submit（提交）。

> **Note**
> 如果要使用现有 Amazon SNS 主题而不是创建新主题，请在 Targets（目标）中选择其 ARN。请确保主题具有适当的访问策略，并且订阅者列表仅包含允许查看有关部署应用程序的信息的用户。有关更多信息，请参阅为通知配置现有 Amazon SNS 主题以及了解通知内容和安全性。

8. 要完成规则创建，请选择 Submit（提交）。
9. 您必须为用户订阅规则的 Amazon SNS 主题，然后他们才能接收通知。有关更多信息，请参阅为用户订阅作为目标的 Amazon SNS 主题。您还可以在聊天应用程序中设置通知与 Amazon Q 开发者版之间的集成，以将通知发送到 Amazon Chime 聊天室或 Slack 频道。有关更多信息，请参阅在聊天应用程序中配置通知与 Amazon Q 开发者版之间的集成。

创建通知规则（Amazon CLI）

1. 在终端或命令提示符处，运行 create-notification rule 命令以生成 JSON 骨架：

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

您可以将此文件命名为所需的任意名称。在本示例中，文件命名为 *rule.json*。

2. 在纯文本编辑器中打开 JSON 文件，然后对其进行编辑，以包括该规则所需的资源、事件类型和 Amazon SNS 目标。以下示例显示了一个名为 **MyNotificationRule** 的通知规则，该规则适用于 ID 为 *123456789012* 的 Amazon 账户中名为 *MyDeploymentApplication* 的应用程序。部署成功后，系统会向名为 *codestar-notifications-MyNotificationTopic* 的 Amazon SNS 主题发送通知，并附上完整的详细信息类型：

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codedeploy-application-deployment-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDeploymentApplication",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

保存该文件。

3. 通过使用您刚编辑的文件，在终端或命令行上，再次运行 create-notification-rule 命令以创建通知规则：

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 如果成功，该命令将返回通知规则的 ARN，类似于以下内容：

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

重命名 CodeDeploy 应用程序

您可以使用 Amazon CLI 或 CodeDeploy API 更改应用程序的名称。

要查看应用程序名称列表，请使用 Amazon CLI 调用 list-applications 命令。

有关使用 Amazon CLI 更改应用程序名称的信息，请参阅 update-application。

有关使用 CodeDeploy API 更改应用程序名称的信息，请参阅 API_UpdateApplication。

在 CodeDeploy 中删除应用程序

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 操作删除应用程序。有关使用 CodeDeploy API 操作的信息，请参阅 DeleteApplication。

> **Warning**
>
> 删除应用程序将从 CodeDeploy 系统中删除有关该应用程序的信息，包括所有相关的部署组信息和部署详细信息。删除为 EC2/本地部署创建的应用程序时，既不会从实例中删除任何应用程序修订，也不会从 Amazon S3 存储桶中删除修订。删除为 EC2/本地部署创建的应用程序时，不会终止任何 Amazon EC2 实例或取消注册任何本地实例。并且无法撤消。

主题

- 删除应用程序（控制台）
- 删除应用程序（Amazon CLI）

删除应用程序（控制台）

要使用 CodeDeploy 控制台删除应用程序，请执行以下步骤：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序列表中，选择要删除的应用程序。

   将出现一个页面，其中包含有关该应用程序的详细信息。
4. 选择右上角的删除应用程序。
5. 在系统提示时，输入 **delete** 以确认要删除应用程序，然后选择删除。

删除应用程序（Amazon CLI）

要使用 Amazon CLI 删除应用程序，请调用 delete-application 命令，并指定应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。

在中使用部署组 CodeDeploy

您可以为 CodeDeploy 应用程序指定一个或多个部署组。每个应用程序部署使用其中一个部署组。部署组包含在部署期间使用的设置和配置。大多数部署组设置取决于您的应用程序使用的计算平台。可以为任何计算平台的部署组配置某些设置，例如回滚、触发器和警报。

Amazon ECS 计算平台部署中的部署组

在 Amazon ECS 部署中，部署组指定 Amazon ECS 服务、负载均衡器、可选测试侦听器和两个目标组。它还指定何时将流量重新路由到替换任务集以及在成功部署后何时终止原始任务集和 Amazon ECS 应用程序。

Amazon Lambda 计算平台部署中的部署组

在 L Amazon ambda 部署中，部署组为函数的未来部署定义了一组 CodeDeploy配置。 Amazon Lambda 例如，部署组指定如何将流量路由到新版本的 Lambda 函数。它还可以指定警报和回滚。Amazon Lambda 部署组中的单个部署可以覆盖一个或多个组配置。

EC2/本地计算平台部署中的部署组

在 EC2 /Unlide 部署中，部署组是一组针对部署的单个实例。部署组包含单独标记的实例、Amazon A EC2 uto Scaling 组中的亚马逊 EC2 实例，或两者兼而有之。

在就地部署中，部署组中的实例会使用最新的应用程序修订进行更新。

在 blue/green 部署中，通过从一个或多个负载均衡器取消注册原始实例，然后注册一组通常已安装最新应用程序修订版的替换实例，将流量从一组实例重新路由到另一组实例。

您可以将多个部署组与 CodeDeploy 中的一个应用程序关联。这使得能够在不同的时间将一个应用程序修订部署到不同的实例组。例如，您可以使用一个部署组将一个应用程序修订部署到一组标记为 `Test` 的实例，以便在其中确保代码质量。接下来，将相同应用程序修订部署到包含标记为 `Staging` 的实例的部署组，以便进行进一步验证。最后，当您准备好向客户发布最新应用程序时，部署到包括标记为 `Production` 的实例的部署组。

您也可以使用多个标签组，进一步优化部署组中所包含实例的条件。有关信息，请参阅Tagging Instances for Deployments。

使用 CodeDeploy 控制台创建应用程序时，可以同时配置其第一个部署组。使用创建应用程序时，可以在单独的步骤中创建其第一个部署组。 Amazon CLI

要查看已与您的 Amazon 账户关联的部署组列表，请参阅使用 CodeDeploy 查看部署组详细信息。

有关 Amazon EC2 实例标签的信息，请参阅使用控制台处理标签。有关本地实例的信息，请参阅Working with On-Premises Instances。有关 Amazon A EC2 uto Scaling 的信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

主题

- the section called “创建部署组”
- the section called “查看部署组详细信息”
- the section called “更改部署组设置”
- the section called “为部署组配置高级选项”
- the section called “删除部署组”

使用创建部署组 CodeDeploy

您可以使用 CodeDeploy 控制台、 Amazon CLI CodeDeploy APIs、或 Amazon CloudFormation 模板来创建部署组。有关使用 Amazon CloudFormation 模板创建部署组的信息，请参阅Amazon CloudFormation 模板供 CodeDeploy参考。

使用 CodeDeploy 控制台创建应用程序时，可以同时配置其第一个部署组。使用创建应用程序时，可以在单独的步骤中创建其第一个部署组。 Amazon CLI

作为创建部署组的一部分，您必须指定服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

主题

- 为就地部署创建部署组（控制台）
- 为 EC2 /本地 blue/green 部署创建部署组（控制台）
- 为 Amazon ECS 部署创建部署组（控制台）
- 在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器
- 为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器
- 创建部署组（CLI）

为就地部署创建部署组（控制台）

要使用 CodeDeploy 控制台为就地部署创建部署组，请执行以下操作：

> **Warning**
>
> 以下情况下请勿按照这些步骤操作：
>
> - 您尚未为实例做好在应用程序的首次 CodeDeploy 部署中使用的准备。要设置您的实例，请按照使用适用于 CodeDeploy 的实例中的说明操作，然后执行本主题中的步骤。
> - 您需要创建使用自定义部署配置的部署组，但您尚未创建部署配置。按照Create a Deployment Configuration中的说明操作，然后执行本主题中的步骤。
> - 您没有至少信任中描述 CodeDeploy 的信任和权限的服务角色步骤 2：为创建服务角色 CodeDeploy。要创建和配置服务角色，请按照步骤 2：为创建服务角色 CodeDeploy中的说明操作，然后执行本主题中的步骤。
> - 您想在 ELB 中为就地部署选择 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer，但尚未创建。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在 Applications 页上，选择要为其创建部署组的应用程序的名称。
4. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
5. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

   > **Note**
   >
   > 如果您要使用其他部署组中使用的相同设置（包括部署组名称、标签、Amazon A EC2 uto Scaling 组名称或两者兼而有之；以及部署配置），请在此页面上指定这些设置。尽管此

新部署组和现有部署组具有相同的名称，但仍将其 CodeDeploy 视为单独的部署组，因为它们各自与不同的应用程序关联。

6. 在 Service role (服务角色) 中，选择向 CodeDeploy 授予访问您的目标实例的权限的服务角色。
7. 在部署类型中，选择就地。
8. 在环境配置中，执行以下操作：

   a. 如果要将应用程序部署到 Amazon A EC2 uto Scaling 组，请选择 Amazon A EC2 uto Scaling 组，然后选择要将应用程序修订部署到的 Amazon A EC2 uto Scaling 组的名称。当作为 Amazon Aut EC2 o Scaling 组的一部分启动新的亚马逊 EC2实例时， CodeDeploy 可以自动将您的修订部署到新实例。您最多可以向一个部署组添加 10 个 Amazon A EC2 uto Scaling 群组。有关更多信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

   b. 如果您选择了 Amazon A EC2 uto Scaling 组，则可以选择向 Auto Scaling 组添加终止挂钩，以便在创建或更新部署组时将终止挂钩 CodeDeploy 安装到您的 Auto Scaling 组中。安装此挂钩后， CodeDeploy 将执行终止部署。有关更多信息，请参阅 在 Auto Scaling 横向缩减事件期间启用终止部署。

   c. 如果要为实例添加标签，请选择 Amazon EC2 实例或本地实例。在键和值字段中，输入用于标记实例的键值对的值。一个标签组中最多可标记 10 对键值对。

      i. 您可以在 “值” 字段中使用通配符来识别以特定模式标记的所有实例，例如类似的 Amazon EC2 实例、成本中心和组名等。例如，如果您在 “键” 字段中选择 “名称”，然后在 “值” 字段**GRP-*a**中输入，则会 CodeDeploy 标识符合该模式的所有实例**GRP-1a**，例如**GRP-2a**、和**GRP-XYZ-a**。

      ii. Value（值）字段区分大小写。

      iii. 要从列表中删除键值对，请选择删除图标。

      当 CodeDeploy 找到与每个指定的键值对或 Amazon A EC2 uto Scaling 组名匹配的实例时，它会显示匹配的实例的数量。要查看有关这些实例的更多信息，请单击该数字。

      如果您希望更精细地确定部署实例的条件，请选择 Add tag group 创建标签组。您最多可以创建三个标签组，每组中最多可包含 10 对键值对。如果在部署组中使用多个标签组，只有所有标签组均标记出的实例才会包含在部署组中。也就是说，只有与每组中至少一个标签匹配的实例才会包含在部署组中。

      有关使用标签组优化部署组的信息，请参阅Tagging Instances for Deployments。

9. 在 S ystems Manager 的 CodeDeploy 代理配置中，指定您希望如何在部署组中的实例上安装和更新代理。有关 CodeDeploy 代理的更多信息，请参阅使用代 CodeDeploy 理。有关 Systems Manager 的详细信息，请参阅什么是 Systems Manager？

   a. 从不：跳过使用 Systems Manager 配置 CodeDeploy 安装。实例必须安装代理才能在部署中使用，因此只有在以其他方式安装 CodeDeploy 代理时才应选择此选项。

   b. 仅限一次：Systems Manager 将在部署组中的每个实例上安装一次 CodeDeploy 代理。

   c. 立即安排更新：Systems Manager 将创建与状态管理器的关联，该关联将按照您配置的计划安装 CodeDeploy 代理。有关状态管理器和关联的详细信息，请参阅关于状态管理器。

10. 在 Deployment configuration（部署配置）中，选择一个部署配置以控制部署实例的速率，如一次部署一个或一次全部部署。有关部署配置的更多信息，请参阅在中使用部署配置 CodeDeploy。

11. （可选）在负载均衡器中，选择启用负载平衡，然后从列表中选择经典负载均衡器、Application Load Balancer 目标组和 Network Load Balancer 目标组，以便在 CodeDeploy 部署期间管理实例的流量。您最多可以选择 10 个经典负载均衡器和 10 个目标组，总共可以选择 20 个项目。确保您要部署到的 Amazon EC2 实例已注册到选定的负载均衡器（传统负载均衡器）或目标组（应用程序负载均衡器和网络负载均衡器）。

    在部署期间，原始实例将从选定的负载均衡器和目标组中注销，以防止在部署期间将流量路由到这些实例。部署完成后，将向所有 选定的经典负载均衡器和目标组重新注册每个实例。

    有关用于 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing。

    > **Warning**
    >
    > 如果您要在此部署组中同时配置 Auto Scaling 组和 ELB 负载均衡器，并且想要将负载均衡器连接到 Auto Scaling 组，我们建议您在通过此部署组创建 CodeDeploy 部署之前完成此附件。在创建部署后尝试完成连接可能会导致所有实例意外从负载均衡器取消注册。

12. （可选）展开 “高级” 并配置要包含在部署中的任何选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报、Auto Scaling 选项或自动回滚。

    有关更多信息，请参阅 为部署组配置高级选项。

13. 选择 Create deployment group（创建部署组）。

为 EC2 /本地 blue/green 部署创建部署组（控制台）

要使用 CodeDeploy 控制台为蓝/绿部署创建部署组，请执行以下操作：

> **Warning**
>
> 以下情况下请勿按照这些步骤操作：
>
> - 在 blue/green 部署过程中，您没有要替换的安装了 CodeDeploy 代理的实例。要设置您的实例，请按照使用适用于 CodeDeploy 的实例中的说明操作，然后执行本主题中的步骤。
> - 您需要创建使用自定义部署配置的应用程序，但您尚未创建部署配置。按照Create a Deployment Configuration中的说明操作，然后执行本主题中的步骤。
> - 您没有至少信任中描述 CodeDeploy 的信任和权限的服务角色步骤 2：为创建服务角色 CodeDeploy。要创建和配置服务角色，请按照步骤 2：为创建服务角色 CodeDeploy中的说明操作，然后执行本主题中的步骤。
> - 您尚未在 Elastic Load Balancing 中创建经典负载均衡器或应用程序负载均衡器来注册替换环境中的实例。有关更多信息，请参阅 在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在 Applications 页上，选择要为其创建部署组的应用程序的名称。
4. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。
5. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

   > **Note**
   >
   > 如果您想使用其他部署组中使用的相同设置（包括部署组名称、标签、Amazon A EC2 uto Scaling 组名称和部署配置），请在此页面上选择这些设置。虽然这个新的部署组与现有

部署组同名， CodeDeploy 仍认为它们是两个部署组，因为与它们关联的应用程序各不相同。

6. 在 Service role (服务角色) 中，选择向 CodeDeploy 授予访问您的目标实例的权限的服务角色。
7. 在 Deployment type（部署类型）中选择 Blue/green（蓝/绿）。
8. 在环境配置中，执行以下操作：
   - 选择用于为您的替换环境提供实例的方法。您有以下选项：
     - 自动复制 Amazon A EC2 uto Scaling 群组：通过复制您指定的群组来 CodeDeploy创建 Amazon A EC2 uto Scaling 群组。
     - Manually provision instances：在创建部署前，您不会为替换环境指定实例。您必须在启动部署前创建实例。您应于此处指定要替换的实例。
   - 如果您选择了“自动复制 Amazon A EC2 uto Scaling 组”，则可以选择向 Auto Scaling 组添加终止挂钩，以便在创建或更新部署组时将终止挂钩 CodeDeploy 安装到您的 Auto Scaling 组中。安装此挂钩后， CodeDeploy 将执行终止部署。有关更多信息，请参阅 在 Auto Scaling 横向缩减事件期间启用终止部署。
9. 在 S ystems Manager 的 CodeDeploy 代理配置中，指定您希望如何在部署组中的实例上安装和更新代理。有关 CodeDeploy 代理的更多信息，请参阅使用代 CodeDeploy 理。有关 Systems Manager 的详细信息，请参阅什么是 Systems Manager？
   a. 从不：跳过使用 Systems Manager 配置 CodeDeploy 安装。实例必须安装代理才能在部署中使用，因此只有在以其他方式安装 CodeDeploy 代理时才应选择此选项。
   b. 仅限一次：Systems Manager 将在部署组中的每个实例上安装一次 CodeDeploy 代理。
   c. 立即安排更新：Systems Manager 将创建与状态管理器的关联，该关联将按照您配置的计划安装 CodeDeploy 代理。有关状态管理器和关联的详细信息，请参阅关于状态管理器。
10. 根据您在步骤 8 中的选择，请执行以下操作之一：
    - 如果您选择了“自动复制 Amazon A EC2 uto Scal ing 组”：在 Amazon A EC2 uto Sc aling 组中，选择或输入要用作为替换环境中的实例创建的 Amazon A EC2 uto Scaling 组模板的 Amazon Auto Scaling 组的名称。 EC2 您选择的 Amazon A EC2 uto Scaling 组中当前运行良好的实例数量是在您的替代环境中创建的。
    - 如果您选择手动配置实例：选择 Amazon A EC2 uto Scaling 组、Amazon A EC2 uto Scaling 实例或两者兼而有之，以指定要添加到此部署组的实例。输入 Amazon A EC2 uto Scaling 标签值或 Amazon A EC2 uto Scaling 组名称，以识别您的原始环境中的实例（即您要替换的实例或正在运行当前应用程序修订版的实例）。

11. 在负载均衡器中，选择启用负载平衡，然后从列表中选择您要注册替换的 Amazon EC2 实例的传统负载均衡器、Application Load Balancer 目标组和网络负载均衡器目标组。每个替换实例都将在所有 选定的经典负载均衡器和目标组中注册。您最多可以选择 10 个经典负载均衡器和 10 个目标组，总共可以选择 20 个项目。

    根据您选择的流量重新路由和部署配置设置，流量将从原始实例重新路由到替换实例。

    有关用于 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing。

    > **Warning**
    > 如果您要在此部署组中同时配置 Auto Scaling 组和 ELB 负载均衡器，并且想要将负载均衡器连接到 Auto Scaling 组，我们建议您在通过此部署组创建 CodeDeploy 部署之前完成此附件。在创建部署后尝试完成连接可能会导致所有实例意外从负载均衡器取消注册。

12. 在 Deployment settings 中，查看用于将流量重新路由到替换环境的默认选项、要用于部署的部署配置以及在部署后处理原始环境中的实例的方式。

    如果您要更改设置，请继续执行下一步。否则，请跳至步骤 14。

13. 要更改部署的 blue/green 部署设置，请选择以下任一设置。

设置	选项
Traffic rerouting	• 立即重新路由流量：一旦预置替换环境中的实例并在这些实例上安装最新应用程序修订，这些实例将自动注册到指定的负载均衡器和目标组，从而使流量重新路由到它们。原始环境中的实例随后将取消注册。 • 我将选择是否重新路由流量：替换环境中的实例不会注册到指定的负载均衡器和目标组，除非您手动重新路由流量。如果在没有重新路由流量的情况下经过了指定的等待时间，部署状态将更改为“Stopped”。
部署配置	选择替换环境中的实例向负载均衡器和目标组注册的速度，如一次一个或一次全部。

设置	选项
	Note 将流量成功路由到替换环境后，无论选择了哪个部署配置，原始环境中的实例都将一次全部取消注册。 有关更多信息，请参阅 在中使用部署配置 CodeDeploy。
Original instances	• 终止部署组中的原始实例：将流量重新路由到替换环境后，已从负载均衡器和目标组取消注册的实例将在您指定的一段等待时间后终止。 • 继续运行部署组中的原始实例：将流量重新路由到替换环境后，已从负载均衡器和目标组取消注册的实例将继续运行。

14. （可选）在 “高级” 中，配置要包含在部署中的选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报、Auto Scaling 选项或自动回滚。

    有关在部署组中指定高级选项的信息，请参阅为部署组配置高级选项。

15. 选择 Create deployment group（创建部署组）。

为 Amazon ECS 部署创建部署组（控制台）

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

    > **Note**
    > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序表中，选择与要编辑的部署组关联的应用程序的名称。

4. 在应用程序页面的 Deployment groups（部署组）中，选择要编辑的部署组的名称。
5. 在应用程序页面的 Deployment groups（部署组）选项卡上，选择 Create deployment group（创建部署组）。有关为 Amazon ECS 部署创建部署组所需内容的更多信息，请参阅在开始 Amazon ECS 部署之前。
6. 在 Deployment group name（部署组名称）中，输入一个描述部署组的名称。

> **Note**
>
> 如果您需要使用其他部署组中使用的相同设置（包括部署组名称和部署配置），请在此页上选择这些设置。尽管此新组和现有组可能具有相同的名称，但仍将其 CodeDeploy 视为单独的部署组，因为每个部署组都与单独的应用程序相关联。

7. 在服务角色中，选择一个授予 CodeDeploy对 Amazon ECS 访问权限的服务角色。有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。
8. 从负载均衡器名称中，选择将流量提供给 Amazon ECS 服务的负载均衡器的名称。
9. 在生产侦听器端口中，选择将生产流量路由至您的 Amazon ECS 服务的侦听器的端口和协议。
10. （可选）从测试侦听器端口中，选择测试侦听器的端口和协议，该侦听器在部署期间将流量路由至 Amazon ECS 服务中的替换任务集。您可以在挂钩期间运行 AppSpec 的文件中指定一个或多个 Lambda 函数。`AfterAllowTestTraffic`这些函数可以运行验证测试。如果验证测试失败，将触发部署回滚。如果验证测试成功，则会触发部署生命周期中的下一个挂钩 `BeforeAllowTraffic`。如果未指定测试侦听器端口，则 `AfterAllowTestTraffic` 挂接期间不会发生任何事情。有关更多信息，请参阅 AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。
11. 从目标组 1 名称和目标组 2 名称中，选择部署期间用于路由流量的目标组。 CodeDeploy 将一个目标组绑定到您的 Amazon ECS 服务的原始任务集，将另一个目标组绑定到其替换任务集。有关更多信息，请参阅应用程序负载均衡器的目标组。
12. 选择立即重新路由流量或指定重新路由流量的时间，以确定何时将流量重新路由到更新后的 Amazon ECS 服务。

    如果您选择立即重新路由流量，则部署会在预置替换任务集后自动重新路由流量。

    如果选择指定重新路由流量的时间，则选择在成功预置替换任务集后要等待的天数、小时数和分钟数。在这段等待时间内，将在 AppSpec 文件中指定的 Lambda 函数中执行验证测试。如果在重新路由流量之前等待时间已过，则部署状态将更改为 `Stopped`。
13. 对于原始修订终止，请选择成功部署后，在终止 Amazon ECS 服务中的原始任务集之前要等待的天数、小时数和分钟数。

14. （可选）在 “高级” 中，配置要包含在部署中的任何选项，例如 Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

    有关更多信息，请参阅 为部署组配置高级选项。

在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器

在运行任何 blue/green 部署或要在部署组中为其指定可选负载均衡器的就地部署之前，必须在 ELB 中至少创建一个 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer。对于 blue/green 部署，您可以使用该负载均衡器注册构成替代环境的实例。您的原始环境中的实例可选择性地注册到此同一负载均衡器。对于就地部署，负载均衡器用于取消注册正在处理的实例 CodeDeploy，并在工作完成后重新注册它们。

CodeDeploy 支持 blue/green 并就地部署到多个负载均衡器后面的 Amazon EC2 实例。例如，假设您有 200 个 Amazon EC2 实例，其中 100 个在 2 个经典负载均衡器中注册，另外 100 个在 2 个应用程序负载均衡器的 4 个目标组中注册。在这种情况下，允许 CodeDeploy 您 blue/green 对所有 200 个实例进行就地部署，即使它们分布在 2 个经典负载均衡器、2 个应用程序负载均衡器和 4 个目标组中。

CodeDeploy 最多支持 10 个经典负载均衡器和 10 个目标组，总共支持 20 个项目。

要配置一个或多个经典负载均衡器，请按照《经典负载均衡器用户指南》中的教程：创建经典负载均衡器中的说明进行操作。注意以下几点：

- 在步骤 2：定义负载均衡器中的创建内部负载均衡器中，选择创建实例时所选的同一 VPC。
- 在步骤 5：向您的负载均衡器注册 EC2 实例中，选择您的部署组中当前的实例（就地部署）或您已指定在原始环境中的实例（蓝/绿部署）。
- 在步骤 7：创建并验证您的负载均衡器中，记录负载均衡器的 DNS 地址。

  例如，如果您已将负载均衡器命名为 `my-load-balancer`，则 DNS 地址将以类似于 `my-load-balancer-1234567890.us-east-2.elb.amazonaws.com` 的格式显示。

要配置一个或多个应用程序负载均衡器，请按照以下主题之一中的说明进行操作：

- 创建应用程序负载均衡器
- 教程：使用 Application Load Balancer Amazon CLI

要配置一个或多个网络负载均衡器，请按照以下主题之一中的说明进行操作：

- 创建网络负载均衡器
- 教程：使用创建 Network Load Balancer Amazon CLI

为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器

在使用 Amazon ECS 计算平台运行部署之前，必须创建应用程序负载均衡器或网络负载均衡器、两个目标组以及一个或两个侦听器。本主题介绍如何创建应用程序负载均衡器。有关更多信息，请参阅 在开始 Amazon ECS 部署之前。

其中一个目标组会将流量定向到 Amazon ECS 应用程序的原始任务集。另一个目标组会将流量定向到其替换任务集。在部署期间， CodeDeploy 创建替换任务集并将流量从原始任务集重新路由到新任务集。 CodeDeploy 决定每个任务集使用哪个目标组。

侦听器 由负载均衡器用于将流量定向到目标组。必须提供一个生产侦听器。您可以指定一个可选的测试侦听器，在您运行验证测试时将流量定向到替换任务集。

负载均衡器必须使用在不同的可用区中具有两个公有子网的 VPC。以下步骤向您展示如何确认默认 VPC、创建 Amazon A EC2 pplication Load Balancer，然后为您的负载均衡器创建两个目标组。有关更多信息，请参阅您的网络负载均衡器的目标组。

验证您的默认 VPC、公有子网和安全组

本主题介绍如何创建可在 Amazon ECS 部署期间使用的 Amazon App EC2 lication Load Balancer、两个目标组和两个端口。其中一个端口是可选的，仅当您在部署过程中需要将流量定向到验证测试的测试端口时为必需。

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon VPC 控制台，网址为https://console.aws.amazon.com/vpc/。
2. 验证要使用的默认 VPC。在导航窗格中，选择您的 VPCs。请注意哪个 VPC 在默认 VPC 列中显示为是。这是您的默认 VPC。其中包含您使用的默认子网。
3. 选择 Subnets（子网）。记下默认子网列中显示为 “是” IDs 的两个子网的子网。您可以在创建负载均衡器 IDs 时使用它们。
4. 选择每个子网，然后选择 Description（描述）选项卡。验证要使用的子网是否位于不同的可用区中。
5. 选择子网，然后选择 Route Table（路由表）选项卡。要确认您要使用的每个子网是否为公有子网，请确认路由表中是否包含指向 Internet 网关链接的行。

6. 登录 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
7. 从导航窗格中，选择 Security Groups（安全组）。
8. 验证要使用的安全组是否可用，并记下其组 ID（例如，sg-abcd1234）。在创建负载均衡器时，您可以使用此 ID。

创建一个 Amazon A EC2 pplication Load Balancer、两个目标组和侦听器（控制台）

要使用亚马逊 EC2 控制台创建 Amazon Application EC2 Load Balancer，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，选择负载均衡器。
3. 选择 Create Load Balancer（创建负载均衡器）。
4. 选择 Application Load Balancer（应用程序负载均衡器），然后选择 Create（创建）。
5. 在 Name（名称）中，输入负载均衡器的名称。
6. 在 Scheme（模式）中，选择 internet-facing（面向 internet）。
7. 在 IP address type（IP 地址类型）中，选择 ipv4。
8. （可选）为负载均衡器配置第二个侦听器端口。您可以使用提供给该端口的测试流量，运行部署验证测试。
   a. 在 Load Balancer Protocol（负载均衡器协议）下，选择 Add listener（添加侦听器）。
   b. 在第二个侦听器的 Load Balancer Protocol（负载均衡器协议）下，选择 HTTP。
   c. 在 Load Balancer Port（负载均衡器端口）下，输入 **8080**。
9. 在 Availability Zones（可用区）的 VPC 中，选择默认 VPC，然后选择要使用的两个默认子网。
10. 选择下一步：配置安全设置。
11. 选择下一步：配置安全组。
12. 选择 Select an existing security group（选择现有安全组），然后选择默认安全组并记下其 ID。
13. 选择下一步：配置路由。
14. 在 Target group（目标组）中，选择 New target group（新建目标组）并配置您的第一个目标组：
   a. 在 Name（名称）中，输入目标组名称（例如，**target-group-1**）。
   b. 在 Target type（目标类型）中，选择 IP。

c. 在 Protocol（协议）中，选择 HTTP。在 Port（端口）中，输入 **80**。

d. 选择下一步：注册目标。

15. 选择 Next: Review（下一步：审核），然后选择 Create（创建）。

为您的负载均衡器创建第二个目标组

1. 配置好负载均衡器后，打开 Amazon EC2 控制台。在导航窗格中，选择目标组。
2. 选择创建目标组。
3. 在 Name（名称）中，输入目标组名称（例如，**target-group-2**）。
4. 在 Target type（目标类型）中，选择 IP。
5. 在 Protocol（协议）中，选择 HTTP。在 Port（端口）中，输入 **80**。
6. 在 VPC 中，选择默认 VPC。
7. 选择创建。

> **Note**
>
> 要使 Amazon ECS 部署运行，您必须为您的负载均衡器创建了两个目标组。您在创建 Amazon ECS 服务时，可使用某个目标组的 ARN。有关更多信息，请参阅《Amazon ECS 用户指南》中的步骤 4：创建 Amazon ECS 服务。

创建一个 Amazon A EC2 pplication Load Balancer、两个目标组和侦听器 (CLI)

要使用 Amazon CLI创建应用程序负载均衡器，请执行以下操作：

1. 使用create-load-balancer命令创建 Application Load Balancer。指定不在相同可用区的两个子网以及安全组。

```
aws elbv2 create-load-balancer --name bluegreen-alb \
--subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --
region us-east-1
```

输出包含负载均衡器的 Amazon 资源名称（ARN），格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. 使用create-target-group命令创建您的第一个目标组。 CodeDeploy 将此目标组的流量路由到您的服务中设置的原始任务或替换任务。

```
aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

输出包含第一个目标组的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. 使用create-target-group命令创建第二个目标组。 CodeDeploy 将目标组的流量路由到您的第一个目标组未提供的任务集。例如，如果您的第一个目标组将流量路由到原始任务集，则该目标组会将流量路由到替换任务集。

```
aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

输出包含第二个目标组的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/209a844cd01825a4
```

4. 使用 create-listener 命令创建侦听器，该侦听器具有将生产流量转发到端口 80 的默认规则。

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

输出包含侦听器的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

5. （可选）使用 create-listener 命令创建另一个侦听器，该侦听器具有将测试流量转发到端口 8080 的默认规则。您可以使用提供给该端口的测试流量，运行部署验证测试。

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 8080 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget2/209a844cd01825a4 --region us-east-1
```

输出包含侦听器的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

创建部署组（CLI）

要使用创建部署组，请调用create-deployment-group命令，指定： Amazon CLI

- 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 部署组的名称。将为指定应用程序创建具有此名称的部署组。部署组只能与一个应用程序关联。
- 有关标识要包含在部署组中的实例的标签、标签组或 Amazon A EC2 uto Scaling 组名称的信息。
- 服务角色的 Amazon 资源名称 (ARN) 标识符， CodeDeploy 允许在与其他 Amazon 服务交互时代表您的 Amazon 账户执行操作。要获取服务角色 ARN，请参阅获取服务角色 ARN（CLI）。有关服务角色的更多信息，请参阅《IAM 用户指南》中的角色术语和概念。
- 与部署组关联的部署类型（就地部署或蓝/绿部署）的相关信息。
- （可选）现有部署配置的名称。要查看部署配置列表，请参阅View Deployment Configuration Details。如果未指定， CodeDeploy 将使用默认部署配置。
- （可选）用于创建向订阅了 Amazon Simple Notification Service 主题的用户推送有关部署和实例事件的通知的触发器的命令。有关更多信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications。
- （可选）用于将现有 CloudWatch 警报添加到部署组的命令，如果警报中指定的指标低于或超过定义的阈值，则会激活这些警报。
- （可选）当部署失败或 CloudWatch 警报激活时，部署将回滚到上次已知良好的修订版的命令。
- （可选）部署在自动扩缩横向缩减事件期间生成生命周期事件钩子时使用的命令。有关更多信息，请参阅 Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy。

- 对于就地部署：
  - （可选）ELB 中用于管理部署过程中实例流量的传统负载均衡器、应用程序负载均衡器或网络负载均衡器的名称。
- 对于 blue/green 部署：
  - blue/green 部署过程的配置：
    - 如何预置替换环境中的新实例。
    - 立即将流量重新路由到替换环境还是等待指定的一段时间后手动重新路由流量。
    - 是否应终止原始环境中的实例。
  - ELB 中的经典负载均衡器、应用程序负载均衡器或网络负载均衡器的名称，用于在替换环境中注册的实例。

> **Warning**
>
> 如果您在部署组中同时配置 Auto Scaling 组和 ELB 负载均衡器，并且想要将负载均衡器连接到 Auto Scaling 组，我们建议您在通过此部署组创建 CodeDeploy 部署之前完成此附件。在创建部署后尝试完成连接可能会导致所有实例意外从负载均衡器取消注册。

使用 CodeDeploy 查看部署组详细信息

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关与应用程序关联的所有部署组的详细信息。

主题

- 查看部署组详细信息（控制台）
- 查看部署组详细信息（CLI）

查看部署组详细信息（控制台）

要使用 CodeDeploy 控制台查看部署组详细信息，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

> **Note**
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在 Applications 页上，选择与部署组关联的应用程序名称。

> **Note**
> 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

4. 要查看有关单个部署组的详细信息，请在 Deployment groups（部署组）选项卡上，选择部署组的名称。

查看部署组详细信息（CLI）

要使用 Amazon CLI 查看部署组详细信息，请调用 `get-deployment-group` 命令或 `list-deployment-groups` 命令。

要查看有关单个部署组的详细信息，请调用 get-deployment-group 命令，并指定：

- 与部署组关联的应用程序名称。要获取应用程序名称，请调用 list-applications 命令。
- 部署组名称。要获取部署组名称，请调用 list-deployment-groups 命令。

要查看部署组名称的列表，请调用 list-deployment-groups 命令，并指定与部署组关联的应用程序名称。要获取应用程序名称，请调用 list-applications 命令。

使用 CodeDeploy 更改部署组设置

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 更改部署组的设置。

> **Warning**
> 如果您希望部署组使用尚未创建的自定义部署组，请不要执行这些步骤。而是按照Create a Deployment Configuration中的说明操作，然后返回到本主题。如果您希望部署组使用尚未创

建的不同服务角色，请不要执行这些步骤。服务角色必须信任 CodeDeploy，而且至少具有步骤 2：为创建服务角色 CodeDeploy中所述的权限。要创建和配置具有正确权限的服务角色，请按照步骤 2：为创建服务角色 CodeDeploy中的说明操作，然后返回到本主题。

主题

- 更改部署组设置（控制台）
- 更改部署组设置（CLI）

更改部署组设置（控制台）

要使用 CodeDeploy 控制台更改部署组设置，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序列表中，选择与要更改的部署组关联的应用程序的名称。

   > **Note**
   > 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

4. 选择 Deployment groups（部署组）选项卡，然后选择要更改的部署组的名称。
5. 在 Depoyment group（部署组）页面上，选择 Edit（编辑）。
6. 根据需要修改部署组选项。

   有关部署组组件的信息，请参阅使用创建部署组 CodeDeploy。

7. 选择保存更改。

更改部署组设置（CLI）

要使用 Amazon CLI 更改部署组设置，请调用update-deployment-group 命令，并指定：

- 对于 EC2/本地和 Amazon Lambda 部署：
  - 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
  - 当前部署组名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。
  - （可选）不同的部署组名称。
  - （可选）与某个服务角色对应的另一 Amazon 资源名称（ARN），当与其他 Amazon 服务交互时，该服务角色允许 CodeDeploy 代表您的 Amazon 账户执行操作。要获取服务角色 ARN，请参阅获取服务角色 ARN（CLI）。有关服务角色的更多信息，请参阅《IAM 用户指南》中的角色术语和概念。
  - （可选）部署配置的名称。要查看部署配置列表，请参阅View Deployment Configuration Details。（如果未指定，CodeDeploy 将使用默认部署配置。）
  - （可选）向部署组添加一个或多个现有 CloudWatch 警报的命令（当警报中指定的指标低于或超出定义的阈值时，将激活这些警报）。
  - （可选）当部署失败或 CloudWatch 警报激活时用于将部署回滚到上一个已知正常版本的命令。
  - （可选）部署在自动扩缩横向缩减事件期间生成生命周期事件钩子时使用的命令。有关更多信息，请参阅 Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy。
  - （可选）用于创建或更新触发器的命令。触发器用于向 Amazon Simple Notification Service 中的某个主题发布信息，以便该主题的订阅者可以接收有关此部署组中的部署和实例事件的通知。有关信息，请参阅Monitoring Deployments with Amazon SNS Event Notifications。
- 仅对于 EC2/本地部署：
  - （可选）唯一标识要包括在部署组中的实例的替换标签或标签组。
  - （可选）要添加到部署组的替换 Amazon EC2 Auto Scaling 组的名称。
- 仅适用于 Amazon ECS 部署：
  - 要部署的 Amazon ECS 服务。
  - 负载均衡器信息，包括应用程序负载均衡器或网络负载均衡器，Amazon ECS 部署所需的目标组，以及生产和可选的测试侦听器信息。

为部署组配置高级选项

在创建或更新部署组时，可以配置大量选项以更好地控制和监督部署组的部署。

使用此页面上的信息可帮助您在以下主题中使用部署组时配置高级选项：

- 使用创建应用程序 CodeDeploy
- 使用创建部署组 CodeDeploy
- 使用 CodeDeploy 更改部署组设置

Amazon SNS 通知触发器：您可以将触发器添加到一个 CodeDeploy 部署组，以接收与该部署组中的部署相关的事件的通知。对于您加入到该触发器操作中的 Amazon SNS 主题，通知将发送到已订阅该主题的接收人。

您必须已设置此触发器将指向的 Amazon SNS 主题，并且 CodeDeploy 必须有权从该部署组发布到此主题。如果您尚未完成这些设置步骤，可稍后向部署组添加触发器。

如果您需要立即创建触发器以接收有关此应用程序的部署组中的部署事件的通知，请选择 Create trigger。

如果您要部署到 Amazon EC2 实例，则可以为实例创建通知并接收有关实例的通知。

有关更多信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications。

Amazon CloudWatch 警报：您可以创建一个 CloudWatch 警报来按指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。对于 Amazon EC2 部署，您可以为在 CodeDeploy 操作中使用的实例或 Amazon EC2 Auto Scaling 组创建一个警报。对于 Amazon Lambda 和 Amazon ECS 部署，您可以在 Lambda 函数中为错误创建警报。

可将部署配置为在 Amazon CloudWatch 警报检测到某个指标低于或超出定义的阈值时停止。

您必须先在 CloudWatch 中创建警报，然后才能将警报添加到部署组。

1. 要向部署组添加警报监视，请在 Alarms（警报）中，选择 Add alarm（添加警报）。
2. 输入您已经设置好监控此部署的 CloudWatch 警报的名称。

   您必须严格按照 CloudWatch 中创建的名称输入 CloudWatch 警报。要查看警报列表，请打开位于 https://console.amazonaws.cn/cloudwatch/ 的 CloudWatch 控制台，然后选择警报。

其他选项:

- 如果您希望继续部署而不考虑已添加的警报，请选择 Ignore alarm configuration。

  当您希望暂时停用对部署组的警报监视而无需稍后重新添加相同警报时，此选项很有用。

- （可选）如果您希望部署在 CodeDeploy 无法从 Amazon CloudWatch 中检索警报状态的情况下继续，请选择即使警报状态不可用仍可继续部署。

> **Note**
>
> 此选项对应于 CodeDeploy API 中 AlarmConfiguration 对象的 ignorePollAlarmFailure。

有关更多信息，请参阅 使用 CloudWatch 警报监控部署 CodeDeploy。

自动回滚：您可以对部署组或部署进行配置，使之在部署失败或达到您指定的监控阈值时自动回滚。在这种情况下，将会部署上一个已知良好的应用程序版本。您可以在使用控制台创建应用程序、创建部署组或更新部署组时配置部署组的可选设置。创建新部署时，您还可以选择覆盖已为部署组指定的自动回滚配置。

- 您可通过选择下面一个或两个选项，允许部署在发生错误时回滚到已知正常的最近修订：
  - 部署失败时回滚。CodeDeploy 会将上一个已知良好的版本重新部署为新的部署。
  - Roll back when alarm thresholds are met。如果您在前面的步骤中为此应用程序添加了警报，CodeDeploy 将在激活一个或多个指定警报时重新部署上一个已知良好的版本。

> **Note**
>
> 要暂时忽略回滚配置，请选择 Disable rollbacks。当您希望暂时禁止自动回滚而无需稍后重新设置相同配置时，此选项很有用。

有关更多信息，请参阅 使用 CodeDeploy 重新部署和回滚部署。

自动更新过时的实例：在某些情况下，CodeDeploy 可能会将您的应用程序的过时版本部署到您的 Amazon EC2 实例。例如，如果您的 EC2 实例在 CodeDeploy 部署进行期间在 Auto Scaling 组（ASG）中启动，则这些实例将收到您的应用程序的旧版本，而不是最新版本。为了使这些实例保持最新状态，CodeDeploy 会自动启动后续部署（在第一次部署之后立即启动）以更新所有过时的实例。如果您想更改此默认行为，以便将过时的 EC2 实例保留在较旧的版本中，则可以通过 CodeDeploy API 或 Amazon Command Line Interface（CLI）进行更改。

要通过 API 配置过期实例的自动更新，请在 `UpdateDeploymentGroup` 或 `CreateDeploymentGroup` 操作中包含 `outdatedInstancesStrategy` 请求参数。有关详细信息，请参阅《Amazon CodeDeploy API 参考》。

要通过 Amazon CLI 配置自动更新，请使用以下命令之一：

```
aws deploy update-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

或者...

```
aws deploy create-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

... 其中 *arguments* 将替换为部署所需的参数， *UPDATE|IGNORE* 将替换为 `UPDATE` 以启用自动更新，或替换为 `IGNORE` 以禁用自动更新。

示例：

```
aws deploy update-deployment-group --application-name "MyApp" --current-
deployment-group-name "MyDG" --region us-east-1 --outdated-instances-
strategy IGNORE
```

有关这些 Amazon CLI 命令的详细信息，请参阅《Amazon CLI 命令参考》。

使用 CodeDeploy 删除部署组

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 删除与您的 Amazon 账户关联的部署组。

> **Warning**
>
> 如果您删除部署组，则也将从 CodeDeploy 中删除与部署组关联的所有详细信息。部署组中使用的实例将保持不变。并且无法撤消。

主题

- 删除部署组（控制台）
- 删除部署组（CLI）

删除部署组（控制台）

要使用 CodeDeploy 控制台删除部署组，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在应用程序列表中，选择与部署组关联的应用程序的名称。
4. 在 Application details（应用程序详细信息）页面的 Deployment groups（部署组）选项卡上，选择要删除的部署组的名称。
5. 在 Deployment details（部署详细信息）页面上，选择 Delete（删除）。
6. 在系统提示时，键入部署组的名称以确认要删除此部署组，然后选择 Delete。

删除部署组（CLI）

要使用 Amazon CLI 删除部署组，请调用 delete-deployment-group 命令，并指定：

- 与部署组关联的应用程序的名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 与应用程序关联的部署组的名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。

正在处理的应用程序修订版 CodeDeploy

在中 CodeDeploy，修订版包含 CodeDeploy 将部署到您的实例或将在您的实例上运行的脚本 CodeDeploy 的源文件版本。

您计划修订，向修订版添加 AppSpec 文件，然后将修订推送到 Amazon S3 或 GitHub。在推送修订之后，您可以部署它。

主题

- 为 CodeDeploy 规划修订
- 将应用程序规范文件添加到 CodeDeploy 的修订
- 选择 CodeDeploy 存储库类型
- 将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）
- 使用 CodeDeploy 查看应用程序修订详细信息
- 使用 CodeDeploy 在 Amazon S3 中注册应用程序修订

为 CodeDeploy 规划修订

良好的计划使部署修订变得轻松多了。

对于向 Amazon Lambda 或 Amazon ECS 计算平台进行的部署，修订与 AppSpec 文件相同。以下信息不适用。有关更多信息，请参阅 将应用程序规范文件添加到 CodeDeploy 的修订。

对于向 EC2/本地计算平台进行的部署，首先在开发计算机上创建空的根目录（文件夹）。这是您将存储要部署到实例的源文件（如文本和二进制文件、可执行文件、包等）或要在实例上运行的脚本的位置。

例如，在 Linux、macOS 或 Unix 上的 `/tmp/` 根文件夹，或 Windows 上的 `c:\temp` 根文件夹中：

```
/tmp/ or c:\temp (root folder)
  |--content (subfolder)
  |     |--myTextFile.txt
  |     |--mySourceFile.rb
  |     |--myExecutableFile.exe
  |     |--myInstallerFile.msi
  |     |--myPackage.rpm
  |     |--myImageFile.png
```

```
|--scripts (subfolder)
|    |--myShellScript.sh
|    |--myBatchScript.bat
|    |--myPowerShellScript.ps1
|--appspec.yml
```

根文件夹也应包含应用程序规范文件（AppSpec 文件），如此处所示。有关更多信息，请参阅 将应用程序规范文件添加到 CodeDeploy 的修订。

将应用程序规范文件添加到 CodeDeploy 的修订

本主题介绍如何向部署中添加 AppSpec 文件。它还包括用于为 Amazon Lambda 和 EC2/本地部署创建 AppSpec 文件的模板。

主题

- 为 Amazon ECS 部署添加 AppSpec 文件
- 为 Amazon Lambda 部署添加 AppSpec 文件
- 为 EC2/本地部署添加 AppSpec 文件

为 Amazon ECS 部署添加 AppSpec 文件

对于向 Amazon ECS 计算平台进行的部署：

- AppSpec 文件指定用于部署的 Amazon ECS 任务定义，用于路由流量的容器名称和端口映射，以及在部署生命周期事件之后运行的可选 Lambda 函数。
- 修订和 AppSpec 文件相同。
- AppSpec 文件可以使用 JSON 或 YAML 编写。
- 在创建部署时，可以将 AppSpec 文件另存为文本文件或者直接输入控制台中。有关更多信息，请参阅 创建 Amazon ECS 计算平台部署（控制台）。

创建 AppSpec 文件

1. 将 JSON 或 YAML 模板复制到文本编辑器或输入控制台中的 AppSpec 编辑器。
2. 根据需要修改模板。
3. 使用 JSON 或 YAML 验证程序验证 AppSpec 文件。如果您使用 AppSpec 编辑器，则会在您选择 Create deployment（创建部署）时验证该文件。

4. 如果您使用文本编辑器，请保存该文件。如果您使用 Amazon CLI 创建部署，请在 AppSpec 文件位于硬盘驱动器上或 Amazon S3 存储桶中时引用它。如果您使用控制台，则必须将 AppSpec 文件推送到 Amazon S3。

用于 Amazon ECS 部署的 YAML AppSpec 文件模板及其说明

下面是 Amazon ECS 部署的 AppSpec 文件的 YAML 模板以及所有可用选项。有关在 hooks 部分中使用的生命周期事件的信息，请参阅AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。

```
# This is an appspec.yml template file for use with an Amazon ECS deployment in
 CodeDeploy.
# The lines in this template that start with the hashtag are
#   comments that can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section, you must specify the following: the Amazon ECS service,
 task definition name,
# and the name and port of the load balancer to route traffic,
# target version, and (optional) the current version of your Amazon Lambda function.
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "" # Specify the ARN of your task definition
 (arn:aws:ecs:region:account-id:task-definition/task-definition-family-name:task-
definition-revision-number)
        LoadBalancerInfo:
          ContainerName: "" # Specify the name of your Amazon ECS application's
 container
          ContainerPort: "" # Specify the port for your container where traffic
 reroutes
# Optional properties
        PlatformVersion: "" # Specify the version of your Amazon ECS Service
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["",""] # Specify one or more comma-separated subnets in your
 Amazon ECS service
            SecurityGroups: ["",""] # Specify one or more comma-separated security
 groups in your Amazon ECS service
```

```
            AssignPublicIp: "" # Specify "ENABLED" or "DISABLED"
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event.
Hooks:
# Hooks for Amazon ECS deployments are:
    - BeforeInstall: "" # Specify a Lambda function name or ARN
    - AfterInstall: "" # Specify a Lambda function name or ARN
    - AfterAllowTestTraffic: "" # Specify a Lambda function name or ARN
    - BeforeAllowTraffic: "" # Specify a Lambda function name or ARN
    - AfterAllowTraffic: "" # Specify a Lambda function name or ARN
```

用于 Amazon ECS 部署模板的 JSON AppSpec 文件

下面是 Amazon ECS 部署的 AppSpec 文件的 JSON 模板以及所有可用选项。有关模板说明，请参阅上一部分中 YAML 版本中的注释。有关在 hooks 部分中使用的生命周期事件的信息，请参阅AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。

```
{
 "version": 0.0,
 "Resources": [
  {
   "TargetService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
       "TaskDefinition": "",
       "LoadBalancerInfo": {
        "ContainerName": "",
        "ContainerPort":
       },
       "PlatformVersion": "",
       "NetworkConfiguration": {
        "AwsvpcConfiguration": {
         "Subnets": [
          "",
          ""
         ],
         "SecurityGroups": [
          "",
          ""
         ],
         "AssignPublicIp": ""
        }
       }
```

```
      }
    }
   }
  ],
  "Hooks": [
   {
    "BeforeInstall": ""
   },
   {
    "AfterInstall": ""
   },
   {
    "AfterAllowTestTraffic": ""
   },
   {
    "BeforeAllowTraffic": ""
   },
   {
    "AfterAllowTraffic": ""
   }
  ]
 }
```

为 Amazon Lambda 部署添加 AppSpec 文件

对于到 Amazon Lambda 计算平台的部署：

- AppSpec 文件包含有关要部署并用于部署验证的 Lambda 函数的说明。
- 修订和 AppSpec 文件相同。
- AppSpec 文件可以使用 JSON 或 YAML 编写。
- 在创建部署时，可以将 AppSpec 文件另存为文本文件或直接输入控制台 AppSpec 编辑器中。有关更多信息，请参阅 创建 Amazon Lambda 计算平台部署（控制台）。

要创建 AppSpec 文件，请执行以下操作：

1. 将 JSON 或 YAML 模板复制到文本编辑器或输入控制台中的 AppSpec 编辑器。
2. 根据需要修改模板。
3. 使用 JSON 或 YAML 验证程序验证 AppSpec 文件。如果您使用 AppSpec 编辑器，则会在您选择 Create deployment（创建部署）时验证该文件。

4. 如果您使用文本编辑器，请保存该文件。如果您使用 Amazon CLI 创建部署，请在 AppSpec 文件位于硬盘驱动器上或 Amazon S3 存储桶中时引用它。如果您使用控制台，则必须将 AppSpec 文件推送到 Amazon S3。

用于 Amazon Lambda 部署的 YAML AppSpec 文件模板及其说明

有关在 hooks 部分中使用的生命周期事件的信息，请参阅AppSpec Amazon Lambda 部署的 “挂钩” 部分。

```
# This is an appspec.yml template file for use with an Amazon Lambda deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section specify the name, alias,
# target version, and (optional) the current version of your Amazon Lambda function.
Resources:
  - MyFunction: # Replace "MyFunction" with the name of your Lambda function
      Type: AWS::Lambda::Function
      Properties:
        Name: "" # Specify the name of your Lambda function
        Alias: "" # Specify the alias for your Lambda function
        CurrentVersion: "" # Specify the current version of your Lambda function
        TargetVersion: "" # Specify the version of your Lambda function to deploy
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event. Replace "LifeCycleEvent" with BeforeAllowTraffic
# or AfterAllowTraffic.
Hooks:
    - LifeCycleEvent: "" # Specify a Lambda validation function between double-quotes.
```

用于 Amazon Lambda 部署模板的 JSON AppSpec 文件

在以下模板中，您需要将“MyFunction”替换为 Amazon Lambda 函数的名称。在可选的 Hooks 部分，将生命周期事件替换为 BeforeAllowTraffic 或 AfterAllowTraffic。

有关在 Hooks 部分中使用的生命周期事件的信息，请参阅AppSpec Amazon Lambda 部署的 “挂钩” 部分。

```
{
  "version": 0.0,
  "Resources": [{
   "MyFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "",
     "Alias": "",
     "CurrentVersion": "",
     "TargetVersion": ""
    }
   }
  }],
  "Hooks": [{
    "LifeCycleEvent": ""
   }
  ]
 }
```

为 EC2/本地部署添加 AppSpec 文件

如果没有 AppSpec 文件，CodeDeploy 无法将应用程序修订中的源文件映射到其目标，也无法为您向 EC2/本地计算平台中进行的部署运行脚本。

每个修订只能包含一个 AppSpec 文件。

要将 AppSpec 文件添加到修订中，请执行以下操作：

1. 将模板复制到文本编辑器。
2. 根据需要修改模板。
3. 使用 YAML 验证程序检查您的 AppSpec 文件是否有效。
4. 在修订的根目录中将文件另存为 `appspec.yml`。
5. 运行以下命令之一，验证您是否已将 AppSpec 文件放置在根目录中：

   - 对于 Linux、macOS 或 Unix：

     ```
     find /path/to/root/directory -name appspec.yml
     ```

     如果 AppSpec 文件不在根目录中，则不输出任何内容。

- 对于 Windows：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec 文件未存储在那里，则会显示找不到文件错误。

6. 将修订推送到 Amazon S3 或 GitHub。

有关说明，请参阅将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）。

用于 EC2/本地部署的 AppSpec 文件模板及其说明

> **Note**
>
> 部署到 Windows Server 实例不支持 `runas` 元素。如果您要部署到 Windows Server 实例，请不要将其包含在您的 AppSpec 文件中。

```
# This is an appspec.yml template file for use with an EC2/On-Premises deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux,
#   Red Hat Enterprise Linux (RHEL), or Ubuntu Server
#   instances.
# Specify "os: windows" if this revision targets Windows Server instances.
# (You cannot specify both "os: linux" and "os: windows".)
os: linux
# os: windows
# During the Install deployment lifecycle event (which occurs between the
#   BeforeInstall and AfterInstall events), copy the specified files
#   in "source" starting from the root of the revision's file bundle
#   to "destination" on the Amazon EC2 instance.
# Specify multiple "source" and "destination" pairs if you want to copy
#   from multiple sources or to multiple destinations.
# If you are not copying any files to the Amazon EC2 instance, then remove the
```

```
#   "files" section altogether. A blank or incomplete "files" section
#   may cause associated deployments to fail.
files:
  - source:
    destination:
  - source:
    destination:
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify a "permissions"
#   section here that describes special permissions to apply to the files
#   in the "files" section as they are being copied over to
#   the Amazon EC2 instance.
#   For more information, see the documentation.
# If you are deploying to Windows Server instances,
#   then remove the
#   "permissions" section altogether. A blank or incomplete "permissions"
#   section may cause associated deployments to fail.
permissions:
  - object:
    pattern:
    except:
    owner:
    group:
    mode:
    acls:
      -
    context:
      user:
      type:
      range:
    type:
      -
# If you are not running any commands on the Amazon EC2 instance, then remove
#   the "hooks" section altogether. A blank or incomplete "hooks" section
#   may cause associated deployments to fail.
hooks:
# For each deployment lifecycle event, specify multiple "location" entries
#   if you want to run multiple scripts during that event.
# You can specify "timeout" as the number of seconds to wait until failing the
 deployment
#   if the specified scripts do not run within the specified time limit for the
#   specified event. For example, 900 seconds is 15 minutes. If not specified,
#   the default is 1800 seconds (30 minutes).
#   Note that the maximum amount of time that all scripts must finish executing
```

```
#   for each individual deployment lifecycle event is 3600 seconds (1 hour).
#   Otherwise, the deployment will stop and CodeDeploy will consider the deployment
#   to have failed to the Amazon EC2 instance. Make sure that the total number of
 seconds
#   that are specified in "timeout" for all scripts in each individual deployment
#   lifecycle event does not exceed a combined 3600 seconds (1 hour).
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify "runas" in an event to
#   run as the specified user. For more information, see the documentation.
#   If you are deploying to Windows Server instances,
#   remove "runas" altogether.
# If you do not want to run any commands during a particular deployment
#   lifecycle event, remove that event declaration altogether. Blank or
#   incomplete event declarations may cause associated deployments to fail.
# During the ApplicationStop deployment lifecycle event, run the commands
#   in the script specified in "location" starting from the root of the
#   revision's file bundle.
  ApplicationStop:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the BeforeInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  BeforeInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the AfterInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  AfterInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ApplicationStart deployment lifecycle event, run the commands
#   in the script specified in "location".
```

```
  ApplicationStart:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ValidateService deployment lifecycle event, run the commands
#   in the script specified in "location".
  ValidateService:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
```

选择 CodeDeploy 存储库类型

CodeDeploy 需要的文件的存储位置称为“存储库”。对存储库的使用取决于您的部署使用哪个计算平台。

- EC2/本地：要将您的应用程序代码部署到一个或多个实例，必须将您的代码打包到一个存档文件，并将代码放置到 CodeDeploy 可在部署过程中访问它的存储库。您应将可部署内容和一个 AppSpec 文件打包到一个存档文件，然后将它上传到 CodeDeploy 支持的某个存储库类型。
- Amazon Lambda 和 Amazon ECS：部署需要 AppSpec 文件，在部署过程中可以通过下列方式之一访问它：
  - 从 Amazon S3 存储桶。
  - 从直接键入控制台中的 AppSpec 编辑器内的文本。有关更多信息，请参阅创建 Amazon Lambda 计算平台部署（控制台）和创建 Amazon ECS 计算平台部署（控制台）。
  - 如果您使用 Amazon CLI，则可以引用位于您的硬盘驱动器上或网络驱动器上的 AppSpec 文件。有关更多信息，请参阅 创建 Amazon Lambda 计算平台部署（CLI） 和 创建 Amazon ECS 计算平台部署（CLI）。

CodeDeploy 当前支持以下存储库类型：

存储库类型	存储库详细信息	支持的计算平台

Amazon S3 ()	Amazon Simple Storage Service（Amazon S3）是 Amazon 解决方案，用于安全、可扩展的对象存储。Amazon S3 将数据作为对象存储在存储桶 中。对象由文件和描述该文件的任何可选元数据组成。 要将对象存储到 Amazon S3 中，请将文件上传到存储桶中。上传文件时，可以设置对象的权限和元数据。 了解更多： • 在 Amazon S3 中创建存储桶 • 将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署） • 使用 CodeDeploy 从 Amazon S3 自动部署	使用以下计算平台的部署可以将修订存储在 Amazon S3 存储桶中。 • EC2/本地 • Amazon Lambda • Amazon ECS
GitHub	您可以将应用程序修订存储在 GitHub 存储库中。只要 GitHub 存储库中的源代码发生更改，您就可以触发源自该存储库的部署。 了解更多： • 将 CodeDeploy 与 GitHub 集成 • 教程：使用 CodeDeploy 从 GitHub 部署应用程序	只有 EC2/本地部署可以将修订存储在 GitHub 存储库中。

Bitbucket	您可以使用 Bitbucket Pipelines 中的 CodeDeploy 管道将代码部署到 EC2 实例的部署组。Bitbucket 管道提供持续集成和持续部署（CI/CD）功能，包括 Bitbucket 部署。CodeDeploy 管道首先将构件推送到您指定的 S3 存储桶，然后从存储桶部署代码构件。 了解更多： • 参见 Bitbucket 的 CodeDeploy 管道	只有 EC2/本地部署可以将修订存储在 BitBucket 存储库中。

> **Note**
>
> Amazon Lambda 部署仅适用于 Amazon S3 存储库。

将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）

按照中所述计划修订版为 CodeDeploy 规划修订并按照中所述向修订版添加 AppSpec 文件后将应用程序规范文件添加到 CodeDeploy 的修订，就可以捆绑组件文件并将修订推送到 Amazon S3 了。对于部署到 Amazon EC2 实例，在推送修订之后，您可以使用 CodeDeploy 将修订从 Amazon S3 部署到实例。

> **Note**
>
> CodeDeploy 也可以用来部署已推送到的修订版 GitHub。有关更多信息，请参阅您的 GitHub 文档。

我们假定您已遵循 入门 CodeDeploy中的说明来设置 Amazon CLI。这对于调用稍后描述的 push 命令来说特别重要。

请确保您拥有 Amazon S3 存储桶。遵循创建存储桶中的说明。

如果您的部署是在 Amazon EC2 实例上，则必须创建目标 Amazon S3 存储桶，或者该存储桶与目标实例位于同一区域。例如，如果您要将修订部署到美国东部（弗吉尼亚州北部）区域的某些实例和美国西部（俄勒冈州）区域的其他实例，那么您必须在美国东部（弗吉尼亚州北部）区域的一个存储桶中部署一个修订副本，在美国西部（俄勒冈州）区域的另一个存储桶中部署同一修订的另一个副本。在此方案中，您随后需要分别在美国东部（弗吉尼亚州北部）区域和美国西部（俄勒冈州）区域中各创建一个单独的部署，即使两个区域和存储桶中的修订相同也是如此。

您必须拥有上传到 Amazon S3 存储桶的权限。您可以通过 Amazon S3 存储桶策略指定这些权限。例如，在以下 Amazon S3 存储桶策略中，使用通配符 (*) 允许 Amazon 账户将文件上传`111122223333`到 Amazon S3 存储桶中名为`amzn-s3-demo-bucket`的任何目录：

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

要查看您的 Amazon 账户 ID，请参阅查找您的 Amazon 账户 ID。

要了解如何生成和附加 Amazon S3 存储桶策略，请参阅存储桶策略示例。

调用 push 命令的用户必须至少具有将修订上传到每个目标 Amazon S3 存储桶的权限。例如，以下策略允许用户将修订上传到名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任意位置：

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

要了解如何创建和附加 IAM policy，请参阅使用策略。

使用推送修订版 Amazon CLI

> **Note**
>
> 该push命令将应用程序工件和 AppSpec 文件捆绑到修订版中。此修订的文件格式是压缩的 ZIP 文件。该命令不能用于 Amazon Lambda 或 Amazon ECS 部署，因为每个部署都需要一个 JSON 格式或 YAML 格式文件的修订版。 AppSpec

调用 push 命令以便为部署绑定和推送修订。它的参数包括：

- --application-name：（字符串）必需。要应用程序修订关联的 CodeDeploy 应用程序的名称。
- --s3-location：（字符串）必需。要上传到 Amazon S3 的应用程序修订的位置的相关信息。您必须指定一个 Amazon S3 存储桶和一个键。关键是修订的名称。 CodeDeploy 在上传内容之前对其进行压缩。采用格式 `s3://amzn-s3-demo-bucket/your-key.zip`。
- --ignore-hidden-files 或 --no-ignore-hidden-files：（布尔值）可选。使用 `--no-ignore-hidden-files` 标志（默认值）会将隐藏文件绑定和上传到 Amazon S3。使用 `--ignore-hidden-files` 标志不会将隐藏文件绑定和上传到 Amazon S3。
- --source（字符串）可选。要部署的内容以及开发计算机上要压缩并上传到 Amazon S3 AppSpec 的文件的位置。该位置被指定为相对于当前目录的路径。如果未指定相对路径或者对路径使用了单个句点（“.”），则使用当前目录。
- --description（字符串）可选。用于概述应用程序修订的注释。如果未指定，则使用默认字符串 “按 Amazon CLI '时间'UTC上传”，其中'时间'是协调世界时 (UTC) 中的当前系统时间。

您可以使用 Amazon CLI 来推送 Amazon EC2 部署的修订。示例推送命令如下所示：

在 Linux、macOS 或 Unix 中：

```
aws deploy push \
  --application-name WordPress_App \
  --description "This is a revision for the application WordPress_App" \
  --ignore-hidden-files \
  --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
  --source .
```

在 Windows 中：

```
aws deploy push --application-name WordPress_App --description "This is a revision for
 the application WordPress_App" --ignore-hidden-files --s3-location s3://amzn-s3-demo-
bucket/WordPressApp.zip --source .
```

此命令执行以下操作：

- 将已绑定的文件与名为 `WordPress_App` 的应用程序关联。
- 将描述附加到修订。
- 忽略隐藏文件。
- 为修订 `WordPressApp.zip` 命名并将其推送到名为 `amzn-s3-demo-bucket` 的存储桶。
- 将根目录中的所有文件绑定到修订。

推送成功后，您可以使用 Amazon CLI 或 CodeDeploy 控制台从 Amazon S3 部署修订。要部署此修订版，请执行以下 Amazon CLI操作：

在 Linux、macOS 或 Unix 中：

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name your-deployment-config-name \
  --deployment-group-name your-deployment-group-name \
  --s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

在 Windows 中：

```
aws deploy create-deployment --application-name WordPress_App --deployment-config-
name your-deployment-config-name --deployment-group-name your-deployment-group-name --
s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

有关更多信息，请参阅 使用创建部署 CodeDeploy。

使用 CodeDeploy 查看应用程序修订详细信息

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关注册到指定应用程序的 Amazon 账户的所有应用程序修订的详细信息。

有关注册修订的信息，请参阅使用 CodeDeploy 在 Amazon S3 中注册应用程序修订。

主题

- 查看应用程序修订详细信息（控制台）
- 查看应用程序修订详细信息（CLI）

查看应用程序修订详细信息（控制台）

要查看应用程序修订详细信息，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开 Deploy（部署），然后选择 Applications（应用程序）。

   > **Note**
   > 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

3. 选择具有您要查看的修订的应用程序的名称。

4. 在 Application details（应用程序详细信息）页面上，选择 Revisions（修订）选项卡，然后查看为应用程序注册的修订的列表。选择一个修订，然后选择 View details（查看详细信息）。

查看应用程序修订详细信息（CLI）

要使用 Amazon CLI 查看应用程序修订，请调用 get-application-revision 命令或 list-application-revisions 命令。

> **Note**
>
> 对 GitHub 的引用仅适用于到 EC2/本地部署的部署。Amazon Lambda 部署的修订不能用于 GitHub。

要查看有关单个应用程序修订的详细信息，请调用 get-application-revision 命令，并指定：

- 应用程序名称。要获取应用程序名称，请调用 list-applications 命令。
- 对于存储在 GitHub 中的修订，GitHub 存储库名称和引用已推送到存储库的应用程序修订的提交的 ID。
- 对于存储在 Amazon S3 中的修订，包含修订的 Amazon S3 存储桶名称；上传的存档文件的名称和文件类型；以及（可选）存档文件的 Amazon S3 版本标识符和 ETag。如果在调用 register-application-revision 的过程中指定了版本标识符和/或 ETag，则必须在此处指定它们。

要查看有关多个应用程序修订的详细信息，请调用 list-application-revisions 命令，并指定：

- 应用程序名称。要获取应用程序名称，请调用 list-applications 命令。
- （可选）要仅查看 Amazon S3 应用程序修订的详细信息，则指定包含修订的 Amazon S3 存储桶名称。
- （可选）要仅查看 Amazon S3 应用程序修订的详细信息，则指定限制对 Amazon S3 应用程序修订的搜索的前缀字符串。（如果未指定，CodeDeploy 将列出所有匹配的 Amazon S3 应用程序修订。）
- （可选）是否基于每个修订是否为部署组的目标修订来列出修订详细信息。（如果未指定，CodeDeploy 将列出所有匹配的修订。）
- （可选）列名称和对修订详细信息列表进行排序的顺序。（如果未指定，CodeDeploy 将按任意顺序列出结果。）

您可以列出所有修订或仅列出存储在 Amazon S3 中的修订。您无法仅列出存储在 GitHub 中的修订。

使用 CodeDeploy 在 Amazon S3 中注册应用程序修订

如果您已调用 push 命令将应用程序修订推送到 Amazon S3，则无需注册修订。但是，如果您通过其他方法将修订上传到 Amazon S3，并且希望在 CodeDeploy 控制台中或者通过 Amazon CLI 显示修订，请先按照以下步骤注册修订。

如果您已将应用程序修订推送到 GitHub 存储库，并且希望在 CodeDeploy 控制台中或者通过 Amazon CLI 显示修订，则还必须遵循以下步骤。

您只能使用 Amazon CLI 或 CodeDeploy API 在 Amazon S3 或 GitHub 中注册应用程序修订。

主题

- 使用 CodeDeploy 在 Amazon S3 中注册修订（CLI）
- 使用 CodeDeploy 在 GitHub 中注册修订（CLI）

使用 CodeDeploy 在 Amazon S3 中注册修订（CLI）

1. 将修订上传到 Amazon S3。
2. 调用 register-application-revision 命令，并指定：
   - 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
   - 有关要注册的修订的信息：
     - 包含修订的 Amazon S3 存储桶的名称。
     - 已上传修订的名称和文件类型。对于 Amazon Lambda 部署，修订是用 JSON 或 YAML 编写的 AppSpec 文件。对于 EC2/本地部署，修订包含 CodeDeploy 将部署到您的实例的源文件版本或 CodeDeploy 将在您的实例上运行的脚本版本。

       > **Note**
       > Windows Server 实例不支持 tar 和压缩的 tar 存档文件格式（.tar 和.tar.gz）。

     - （可选）修订的 Amazon S3 版本标识符。（如果未指定版本标识符，CodeDeploy 将使用最新的版本。）
     - （可选）修订的 ETag。（如果未指定 ETag，则 CodeDeploy 将跳过对象验证。）
   - （可选）您要与修订关联的任何描述。

可以在命令行中，在 register-application-revision 调用中使用以下语法来指定有关 Amazon S3 中修订的信息。（`version` 和 `eTag` 为可选项。）

对于 EC2/本地部署的修订文件：

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|
zip,version=string,eTag=string
```

对于 Amazon Lambda 部署的修订文件：

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

使用 CodeDeploy 在 GitHub 中注册修订（CLI）

> **Note**
>
> Amazon Lambda 部署不能用于 GitHub。

1. 将修订上传到 GitHub 存储库。
2. 调用 register-application-revision 命令，并指定：
   - 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
   - 有关要注册的修订的信息：
     - 分配到包含修订的存储库中的 GitHub 用户或组名，后跟正斜杠（/）和存储库名称。
     - 引用存储库中修订的提交的 ID。
   - （可选）您要与修订关联的任何描述。

可以在 register-application-revision 调用中使用以下语法，在命令行上指定有关 GitHub 中修订的信息：

```
--github-location repository=string,commitId=string
```

在中处理部署 CodeDeploy

在中 CodeDeploy，部署是在一个或多个实例上安装内容的过程以及该过程中涉及的组件。此内容可以包括代码、Web 和配置文件、可执行文件、软件包、脚本等。 CodeDeploy 根据您指定的配置规则部署存储在源存储库中的内容。

如果您使用 EC2 /Unlide 计算平台，则可以同时运行对同一组实例的两个部署。

CodeDeploy 提供了两种部署类型选项：就地部署和蓝/绿部署。

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2 /Unlide 计算平台的部署才能使用就地部署。有关就地部署的更多信息，请参阅就地部署概述。
- 蓝绿部署：部署的行为取决于使用的计算平台：
  - Blue/green on an EC2/On-本地计算平台：使用以下步骤将部署组（原始环境）中的实例替换为另一组实例（替换环境）：
    - 为替换环境配置实例。
    - 在替换实例上安装最新的应用程序修订。
    - 对于应用程序测试和系统验证等活动，可以选择等待时间。
    - 替换环境中的实例在一个或多个 ELB 负载均衡器中注册，从而导致流量被重新路由到这些负载均衡器。原始环境中的实例已注销，可以终止或继续运行以用于其他用途。

    > **Note**
    > 如果您使用 EC2 /本地计算平台，请注意 blue/green 部署仅适用于 Amazon EC2 实例。

  - 或 Amazon Lambda Amazon ECS 计算平台上的蓝/绿：流量根据金丝雀、线性或all-at-once部署配置逐渐移动。
  - 蓝/绿部署通过 Amazon CloudFormation：作为 Amazon CloudFormation 堆栈更新的一部分，流量将从您当前的资源转移到更新的资源。当前，仅支持 ECS blue/green 部署。

  有关 blue/green 部署的更多信息，请参阅蓝绿部署概述。

有关从 Amazon S3 自动部署的信息，请参阅使用自动从 Amazon S3 进行部署 CodeDeploy。

主题

- 使用创建部署 CodeDeploy
- 查看 CodeDeploy 部署详细信息
- 查看 CodeDeploy EC2/本地部署的日志数据
- 通过 CodeDeploy 停止部署
- 使用 CodeDeploy 重新部署和回滚部署
- 在其他 Amazon 账户中部署应用程序
- 使用 CodeDeploy 代理验证本地机器上的部署程序包

使用创建部署 CodeDeploy

您可以使用 CodeDeploy 控制台 Amazon CLI、或 CodeDeploy APIs 来创建部署，该部署将您已经推送到 Amazon S3 的应用程序修订版安装在部署组中的实例上，如果您的部署是在 EC2 /Onlide 计算平台上 GitHub，则安装在部署组中的实例上。

创建部署的过程取决于部署使用的计算平台。

主题

- 部署先决条件
- 创建 Amazon ECS 计算平台部署（控制台）
- 创建 Amazon Lambda 计算平台部署（控制台）
- 创建 EC2 /本地计算平台部署（控制台）
- 创建 Amazon ECS 计算平台部署（CLI）
- 创建 Amazon Lambda 计算平台部署（CLI）
- 创建 EC2 /本地计算平台部署 (CLI)
- 通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation

部署先决条件

在您开始部署之前，请确保完成以下步骤。

Amazon Lambda 计算平台上的部署先决条件

- 创建一个应用程序，其中至少包括一个部署组。有关更多信息，请参阅 使用创建应用程序 CodeDeploy 和 使用创建部署组 CodeDeploy。
- 准备应用程序修订（也称为 AppSpec 文件），用于指定要部署的 Lambda 函数版本。该 AppSpec 文件还可以指定 Lambda 函数来验证您的部署。有关更多信息，请参阅 正在处理的应用程序修订版 CodeDeploy。
- 如果要将自定义部署配置用于您的部署，请在开始部署过程之前创建配置。有关信息，请参阅Create a Deployment Configuration。

EC2/本地计算平台上的部署先决条件

- 对于就地配置，创建或配置要部署到的实例。有关信息，请参阅使用适用于 CodeDeploy 的实例。对于 blue/green 部署，您要么有一个现有 Amazon A EC2 uto Scaling 组用作替换环境的模板，要么有一个或多个实例或 Amazon A EC2 uto Scaling 组指定为原始环境。有关更多信息，请参阅教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组和 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。
- 创建一个应用程序，其中至少包括一个部署组。有关更多信息，请参阅 使用创建应用程序 CodeDeploy 和 使用创建部署组 CodeDeploy。
- 准备好要部署到部署组中的实例的应用程序修订。有关信息，请参阅正在处理的应用程序修订版 CodeDeploy。
- 如果要将自定义部署配置用于您的部署，请在开始部署过程之前创建配置。有关信息，请参阅Create a Deployment Configuration。
- 如果您从 Amazon S3 存储桶部署应用程序修订，则该存储桶与您的部署组中的实例位于同一 Amazon 区域。
- 如果您要从 Amazon S3 存储桶部署应用程序修订，可对该存储桶应用 Amazon S3 存储桶策略。此策略为您的实例授予下载应用程序修订所需的权限。

  例如，以下 Amazon S3 存储桶策略允许任何附带包含 ARN 的 IAM 实例配置文件的 Amazon EC2 实例从名`arn:aws:iam::444455556666:role/CodeDeployDemo`为的 Amazon S3 存储桶中的任意位置进行下载：`amzn-s3-demo-bucket`

```
{
    "Statement": [
        {
            "Action": [
```

```
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

以下 Amazon S3 存储桶策略允许从名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任意位置，下载具有关联 IAM 用户（其中包含 ARN `arn:aws:iam::444455556666:user/CodeDeployUser`）的任意本地实例：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

有关如何生成和附加 Amazon S3 存储桶策略的信息，请参阅存储桶策略示例。

- 如果您正在创建 blue/green 部署，或者在部署组中为就地部署指定了可选的 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer，则您已经使用 Amazon VPC 创建了一个包含至少两个子网的 VPC。（CodeDeploy 使用 ELB，它要求负载均衡器组中的所有实例都位于单个 VPC 中。）

如果您尚未创建 VPC，请参阅 Amazon VPC 入门指南。

- 如果您正在创建 blue/green 部署，则您已在 ELB 中配置了至少一个 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer，并使用它来注册构成原始环境的实例。

> **Note**
>
> 替换环境中的实例稍后将用这个负载均衡器进行注册。

有关配置负载均衡器的更多信息，请参阅在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器和为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器。

通过以下方式进行部署的 blue/green 部署先决条件 Amazon CloudFormation

- 您的模板不需要为 CodeDeploy 应用程序或部署组对资源进行建模。
- 对于至少包含两个子网且使用 Amazon VPC 的 VPC，您的模板必须包含相应资源。
- 您的模板必须包含一个或多个传统负载均衡器、应用程序负载均衡器或 ELB 中的网络负载均衡器的资源，这些资源用于将流量引导至目标群组。

创建 Amazon ECS 计算平台部署（控制台）

本主题介绍了如何使用控制台部署 Amazon ECS 服务。有关更多信息，请参阅教程：将应用程序部署到 Amazon ECS和教程：部署具有验证测试的 Amazon ECS 服务。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 请执行以下操作之一：

   - 如果要部署应用程序，请在导航窗格中，展开 Deploy（部署），然后选择 Applications（应用程序）。选择要部署的应用程序的名称。确保您的应用程序的计算平台列为 Amazon ECS。

- 如果要重新部署某个部署，请在导航窗格中展开 Deploy（部署），然后选择 Deployments（部署）。选择要重新部署的部署并在 Application（应用程序）列中选择其应用程序的名称。确保您的部署的计算平台列为 Amazon ECS。

3. 在部署选项卡上，选择创建部署。

> **Note**
>
> 您的应用程序必须具有部署组才能部署。如果您的应用程序没有部署组，请在部署组选项卡上选择创建部署组。有关更多信息，请参阅 使用创建部署组 CodeDeploy。

4. 在 Deployment group（部署组）中，选择要用于此部署的部署组。
5. 在 Revision location（修订位置）旁边，选择您的修订所在的位置：

   - 我的应用程序存储在 Amazon S3 中 - 有关信息，请参阅指定存储在 Amazon S3 存储桶中的修订的相关信息，然后返回步骤 6。
   - 使用 AppSpec 编辑器-选择 JSON 或 YAML，然后在编辑器中输入您的 AppSpec 文件。您可以通过选择 “另存为文本 AppSpec 文件” 来保存该文件。如果您在这些步骤结束时选择 Deploy（部署），并且您的 JSON 或 YAML 无效，则您将收到错误。有关创建 AppSpec 文件的更多信息，请参阅将应用程序规范文件添加到 CodeDeploy 的修订。

6. （可选）在 Deployment description（部署描述）框中，输入此部署的描述。
7. （可选）在 Rollback configuration overrides 中，您可以为此部署指定与已为部署组指定的选项（如果有）不同的自动回滚选项。

   有关中回滚的信息 CodeDeploy，请参阅重新部署和部署回滚和。使用 CodeDeploy 重新部署和回滚部署

   请从以下内容中选择：

   - 部署失败时回滚 — 将最后一个已知良好的修订版 CodeDeploy重新部署为新部署。
   - 达到警报阈值时回滚-如果警报已添加到部署组，则在激活一个或多个指定警报时 CodeDeploy 重新部署上次已知的良好版本。
   - 禁用回滚 — 不为此部署执行回滚。

8. 选择 Create deployment（创建部署）。

   要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

创建 Amazon Lambda 计算平台部署（控制台）

本主题介绍了如何使用控制台部署 Lambda 函数。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

> **Note**
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 请执行以下操作之一：
   - 如果要部署应用程序，请在导航窗格中，展开 Deploy（部署），然后选择 Applications（应用程序）。选择要部署的应用程序的名称。确保您的应用程序的计算平台列为 Amazon Lambda。
   - 如果要重新部署某个部署，请在导航窗格中展开 Deploy（部署），然后选择 Deployments（部署）。选择要重新部署的部署并在 Application（应用程序）列中选择其应用程序的名称。确保您的部署的计算平台列为 Amazon Lambda。
3. 在部署选项卡上，选择创建部署。

> **Note**
> 您的应用程序必须具有部署组才能部署。如果您的应用程序没有部署组，请在部署组选项卡上选择创建部署组。有关更多信息，请参阅 使用创建部署组 CodeDeploy。

4. 在 Deployment group（部署组）中，选择要用于此部署的部署组。
5. 在 Revision location（修订位置）旁边，选择您的修订所在的位置：
   - 我的应用程序存储在 Amazon S3 中 - 有关信息，请参阅指定存储在 Amazon S3 存储桶中的修订的相关信息，然后返回步骤 6。
   - 使用 AppSpec 编辑器-选择 JSON 或 YAML，然后在编辑器中输入您的 AppSpec 文件。您可以通过选择 “另存为文本 AppSpec 文件” 来保存该文件。如果您在这些步骤结束时选择 Deploy（部署），并且您的 JSON 或 YAML 无效，则您将收到错误。有关创建 AppSpec 文件的更多信息，请参阅将应用程序规范文件添加到 CodeDeploy 的修订。
6. （可选）在 Deployment description（部署描述）框中，输入此部署的描述。
7. （可选）展开部署组覆盖以选择一个不同于部署组中指定项的部署配置，来控制流量如何转移到 Lambda 函数版本。

有关更多信息，请参阅 Amazon Lambda 计算平台上的部署配置。

8. （可选）在 Rollback configuration overrides 中，您可以为此部署指定与已为部署组指定的选项（如果有）不同的自动回滚选项。

   有关中回滚的信息 CodeDeploy，请参阅重新部署和部署回滚和。使用 CodeDeploy 重新部署和回滚部署

   请从以下内容中选择：

   - 部署失败时回滚 — 将最后一个已知良好的修订版 CodeDeploy重新部署为新部署。
   - 达到警报阈值时回滚-如果警报已添加到部署组，则在激活一个或多个指定警报时 CodeDeploy 重新部署上次已知的良好版本。
   - 禁用回滚 — 不为此部署执行回滚。

9. 选择 Create deployment（创建部署）。

   要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

创建 EC2 /本地计算平台部署（控制台）

本主题向您展示如何使用控制台将应用程序部署到 Amazon EC2 或本地服务器。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 请执行以下操作之一：

   - 如果要部署应用程序，请在导航窗格中，展开 Deploy（部署），然后选择 Applications（应用程序）。选择要部署的应用程序的名称。确保您的应用程序的 “计算平台” 列为 EC2/本地。
   - 如果要重新部署某个部署，请在导航窗格中展开 Deploy（部署），然后选择 Deployments（部署）。找到要重新部署的部署，然后在 Application（应用程序）列中选择其应用程序的名称。确保部署的 “计算平台” 列为 “EC2/本地”。

3. 在部署选项卡上，选择创建部署。

> **Note**
>
> 您的应用程序必须具有部署组才能部署。如果您的应用程序没有部署组，请在部署组选项卡上选择创建部署组。有关更多信息，请参阅 使用创建部署组 CodeDeploy。

4. 在 Deployment group（部署组）中，选择要用于此部署的部署组。
5. 在 Repository type 旁边，选择保存您的修订的存储库类型：
   - 我的应用程序存储在 Amazon S3 中 - 有关信息，请参阅指定存储在 Amazon S3 存储桶中的修订的相关信息，然后返回步骤 6。
   - 我的应用程序存储在 GitHub — 有关信息，请参阅指定存储在存储 GitHub 库中的修订的相关信息，然后返回到步骤 6。
6. （可选）在 Deployment description（部署描述）框中，输入此部署的描述。
7. （可选）展开 “覆盖部署配置”，选择部署配置，以控制流量如何转移到 Amazon EC2 或本地服务器，该配置与部署组中指定的服务器不同。

   有关更多信息，请参阅 在中使用部署配置 CodeDeploy。
8. a. 如果您希望在 ApplicationStop生命周期事件失败时成功部署到实例，请选择**`ApplicationStop`**生命周期事件失败时不要使部署失败。

   b. 展开其他部署行为设置以指定如何 CodeDeploy 处理部署目标位置中不属于先前成功部署的文件。

      请从以下内容中选择：

      - 使部署失败 - 系统报告出错，并且部署状态更改为 `Failed`。
      - 覆盖内容 - 如果目标位置存在同名文件，则来自应用程序修订的版本将替换它。
      - 保留内容 - 如果目标位置存在同名文件，则该文件将保留，并且应用程序修订中的版本不会复制到实例。

      有关更多信息，请参阅 现有内容的回滚行为。
9. （可选）在 Rollback configuration overrides 中，您可以为此部署指定与已为部署组指定的选项（如果有）不同的自动回滚选项。

   有关中回滚的信息 CodeDeploy，请参阅重新部署和部署回滚和。使用 CodeDeploy 重新部署和回滚部署

请从以下内容中选择：

- 部署失败时回滚 — 将最后一个已知良好的修订版 CodeDeploy重新部署为新部署。
- 达到警报阈值时回滚-如果警报已添加到部署组， CodeDeploy 则在激活一个或多个指定警报时部署上次已知的良好版本。
- 禁用回滚 — 不为此部署执行回滚。

10. 选择开始部署。

    要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

主题

- 指定存储在 Amazon S3 存储桶中的修订的相关信息
- 指定存储在存储 GitHub 库中的修订的相关信息

指定存储在 Amazon S3 存储桶中的修订的相关信息

如果您正在执行创建 EC2 /本地计算平台部署（控制台）中的步骤，请遵循这些步骤以添加有关存储在 Amazon S3 存储桶中的应用程序修订的详细信息。

1. 将您的修订的 Amazon S3 链接复制到修订位置。要查找链接值，请执行以下操作：

   a. 在单独的浏览器选项卡中：

      登录 Amazon Web Services 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。

      浏览并选择您的修订。

   b. 如果 Properties 窗格不可见，请选择 Properties 按钮。

   c. 在 “属性” 窗格中，将 “链接” 字段的值复制到 CodeDeploy 控制台的 “修订位置” 框中。

   要指定 ETag （文件校验和）作为修订位置的一部分，请执行以下操作：

   - 如果链接字段的值 ETag 以结尾 **`?versionId=`*`versionId`*`&etag=`**，请在链接字段值的末尾添加和。
   - 如果 “链接” 字段值未指定版本 ID，请在 “ ETag 链接” 字段值的末尾添加**`?etag=`**和。

> **Note**
>
> 您并不一定需要复制 Link 字段的值，也可以使用下列格式之一键入修订位置：
>
> `s3://bucket-name/folders/objectName`
> `s3://bucket-name/folders/objectName?versionId=versionId`
> `s3://bucket-name/folders/objectName?etag=etag`
> `s3://bucket-name/folders/objectName?versionId=versionId&etag=etag`
>
> 对于中国（北京）区域：
>
> - `bucket-name.s3.cn-north-1.amazonaws.com/folders/objectName`
>
> 对于中国（宁夏）区域：
>
> - `bucket-name.s3.cn-northwest-1.amazonaws.com/folders/objectName`

2. 如果 File type 列表中显示的消息说明无法检测文件类型，则选择修订的文件类型。否则，请接受检测到的文件类型。

指定存储在存储 GitHub 库中的修订的相关信息

如果您正在按照中的步骤操作创建 EC2 /本地计算平台部署（控制台），请按照以下步骤添加有关存储在存储 GitHub 库中的应用程序修订的详细信息。

1. 在 Connect t o 中 GitHub，执行以下任一操作：
   - 要创建 CodeDeploy 应用程序与 GitHub 帐户的连接，请在不同的 Web 浏览器选项卡中注销 GitHub。在GitHub 帐户中，输入用于标识此连接的名称，然后选择 Connect to GitHub。该网页会提示您授权 CodeDeploy 与您的应用程序进行交互。 GitHub 继续执行步骤 2。
   - 要使用您已经创建的连接，请在GitHub帐户中选择其名称，然后选择 Connect to GitHub。继续执行步骤 4。
   - 要创建与其他 GitHub 帐户的连接，请在不同的 Web 浏览器选项卡中注销 GitHub。选择 “连接到其他 GitHub 帐户”，然后选择 “连接到” GitHub。继续执行步骤 2。
2. 如果系统提示您登录 GitHub，请按照 “登录” 页面上的说明进行操作。使用您的 GitHub 用户名或电子邮件和密码登录。

3. 如果显示 Authorize application（授权应用程序）页面，则选择 Authorize application（授权应用程序）。
4. 在 “创建部署” 页面的 “存储库名称” 框中，输入包含修订的 GitHub 用户或组织名称，然后输入正斜杠 (/)，然后输入包含该修订的存储库的名称。如果您不确定要键入的值，请执行以下步骤：

   a. 在另一个 Web 浏览器选项卡中，转到您的GitHub控制面板。
   b. 在 Your repositories（您的资料库）中，将鼠标指针悬停在目标存储库名称的上方。将出现一个工具提示，显示 GitHub 用户或组织名称，后跟正斜杠 (/)，后跟存储库的名称。将显示的此值输入 Repository name（存储库名称）框中。

   > **Note**
   >
   > 如果目标存储库名称在 “您的存储库” 中不可见，请使用 GitHub “搜索” 框查找目标存储库名称以及 GitHub 用户或组织名称。

5. 在 Commit ID（提交 ID）框中，输入引用存储库中修订的提交的 ID。如果您不确定要键入的值，请执行以下步骤：

   a. 在另一个 Web 浏览器选项卡中，转到您的GitHub控制面板。
   b. 在 Your repositories（您的存储库）中，选择包含目标提交的存储库名称。
   c. 在提交列表中，找到并复制引用存储库中修订的提交 ID。此 ID 的长度通常为 40 个字符并包含字母和数字。（请勿使用提交 ID 的简短版本，此版本通常包含更长版本提交 ID 的前 10 个字符。）
   d. 将提交 ID 粘贴到 Commit ID 框中。

创建 Amazon ECS 计算平台部署（CLI）

在您创建了应用程序和修订版之后（在 Amazon ECS 部署中，这是 AppSpec 文件）：

调用 create-deployment 命令，并指定：

- 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 部署组名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。
- 有关要部署的修订的信息：

  对于存储在 Amazon S3 中的修订：

  - 包含修订的 Amazon S3 存储桶名称。

- 已上传修订的名称。
- （可选）修订的 Amazon S3 版本标识符。（如果未指定版本标识符，则 CodeDeploy 使用最新版本。）
- （可选）ETag 适用于修订版。（如果未指定，则 CodeDeploy 跳过对象验证。）ETag

  对于不在 Amazon S3 中的文件内存储的修订，您需要文件名及其路径。您的修订文件是使用 JSON 或 YAML 编写的，因此扩展名很可能为 .json 或 .yaml。
- （可选）部署的说明。

修订文件可指定为上传到 Amazon S3 存储桶的文件，也可以指定为字符串。在用作 create-deployment 命令的一部分时，各自的语法为

- Amazon S3 存储桶：

  `version` 和 `eTag` 是可选的。

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

- 字符串：

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

> **Note**
>
> create-deployment 命令可以从文件加载修订。有关更多信息，请参阅从文件中加载参数。

有关 Amazon Lambda 部署修订版模板的信息，请参阅为 Amazon Lambda 部署添加 AppSpec 文件。有关示例修订，请参阅 AppSpec Amazon Lambda 部署的文件示例 。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

创建 Amazon Lambda 计算平台部署（CLI）

创建应用程序和修订后（在 Amazon Lambda 部署中，这是 AppSpec 文件）：

调用 create-deployment 命令，并指定：

- 应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 部署组名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。
- 有关要部署的修订的信息：

  对于存储在 Amazon S3 中的修订：

  - 包含修订的 Amazon S3 存储桶名称。
  - 已上传修订的名称。
  - （可选）修订的 Amazon S3 版本标识符。（如果未指定版本标识符，则 CodeDeploy 使用最新版本。）
  - （可选）ETag 适用于修订版。（如果未指定，则 CodeDeploy 跳过对象验证。）ETag

  对于不在 Amazon S3 中的文件内存储的修订，您需要文件名及其路径。您的修订文件是使用 JSON 或 YAML 编写的，因此扩展名很可能为 .json 或 .yaml。
- （可选）要使用的部署配置的名称。要查看部署配置的列表，请调用 list-deployment-configs 命令。（如果未指定，则 CodeDeploy 使用特定的默认部署配置。）
- （可选）部署的说明。

修订文件可指定为上传到 Amazon S3 存储桶的文件，也可以指定为字符串。在用作 create-deployment 命令的一部分时，各自的语法为

- Amazon S3 存储桶：

  `version` 和 `eTag` 是可选的。

  ```
  --s3-location bucket=string,key=string,bundleType=JSON|
  YAML,version=string,eTag=string
  ```

- 字符串：

  ```
  --revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
  ```

> **Note**
>
> create-deployment 命令可以从文件加载修订。有关更多信息，请参阅从文件中加载参数。

有关 Amazon Lambda 部署修订版模板的信息，请参阅为 Amazon Lambda 部署添加 AppSpec 文件。有关示例修订，请参阅 AppSpec Amazon Lambda 部署的文件示例 。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

创建 EC2 /本地计算平台部署 (CLI)

要使用将修订版部署 Amazon CLI 到 EC2 /Unlide 计算平台，请执行以下操作：

1. 在您准备好了实例之后，创建应用程序，然后推送修订，执行以下操作之一：
   - 如果您希望从 Amazon S3 存储桶部署修订，请立即继续执行步骤 2。
   - 如果要从 GitHub 存储库部署修订，请先完成中的步骤将 CodeDeploy 应用程序连接到存储 GitHub 库，然后继续执行步骤 2。
2. 调用 create-deployment 命令，并指定：
   - `--application-name`：应用程序名称。要查看应用程序名称的列表，请调用 list-applications 命令。
   - `--deployment-group-name`: Amazon EC2 部署组的名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。
   - `--revision`：有关要部署的修订的信息：

     对于存储在 Amazon S3 中的修订：
     - `s3Location`：包含修订的 Amazon S3 存储桶名称。
     - `s3Location --> key`：已上传修订的名称。
     - `s3Location --> bundleType`：已上传修订的文件类型。

       > **Note**
       > Windows Server 实例不支持 tar 和压缩的 tar 存档文件格式（.tar 和.tar.gz）。

     - `s3Location --> version`：（可选）修订的 Amazon S3 版本标识符。（如果未指定版本标识符，则 CodeDeploy 使用最新版本。）
     - `s3Location --> eTag`: （可选） ETag 适用于修订版。（如果未指定，则 CodeDeploy 跳过对象验证。）ETag

     对于存储在 GitHub以下位置的修订版本：
     - `gitHubLocation --> repository`：分配给包含修订的存储库的 GitHub 用户名或组名，后跟正斜杠 (/)，后跟存储库名称。

  - `gitHubLocation --> commitId`：修订的提交 ID。
- `--deployment-config-name`：（可选）要使用的部署配置的名称。要查看部署配置的列表，请调用 list-deployment-configs 命令。（如果未指定，则 CodeDeploy 使用特定的默认部署配置。）
- `--ignore-application-stop-failures | --no-ignore-application-stop-failures`：（可选）您是否希望当 `ApplicationStop` 部署生命周期事件失败时，实例的部署仍继续 `BeforeInstall` 部署生命周期事件。
- `--description`：（可选）部署的说明。
- `--file-exists-behavior`:（可选）作为部署过程的一部分， CodeDeploy 代理会从每个实例中删除最新部署安装的所有文件。选择当不属于先前部署的文件出现在目标部署位置时会发生什么。
- `--target-instances`：对于 blue/green 部署，有关 blue/green 部署中属于替换环境的实例的信息，包括一个或多个 Amazon A EC2 uto Scaling 组的名称，或者用于识别 Amazon EC2 实例的标签筛选密钥、类型和值。

> **Note**
>
> 在 create-deployment 调用中使用此语法，可直接在命令行上指定有关 Amazon S3 中修订的信息。（`version` 和 `eTag` 可选。）
>
> ```
> --s3-location bucket=string,key=string,bundleType=tar|tgz|
> zip,version=string,eTag=string
> ```
>
> 使用以下语法作为create-deployment调用的一部分， GitHub 直接在命令行中指定有关修订的信息：
>
> ```
> --github-location repository=string,commitId=string
> ```
>
> 要获取有关已推送修订的信息，请调用 list-application-revisions 命令。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 。

create-deployment 命令参考

以下是 `create-deployment` 命令的命令结构和选项。有关更多信息，请参阅《Amazon CLI 命令参考》中的 create-deployment 参考。

```
create-deployment
--application-name <value>
[--deployment-group-name <value>]
[--revision <value>]
[--deployment-config-name <value>]
[--description <value>]
[--ignore-application-stop-failures | --no-ignore-application-stop-failures]
[--target-instances <value>]
[--auto-rollback-configuration <value>]
[--update-outdated-instances-only | --no-update-outdated-instances-only]
[--file-exists-behavior <value>]
[--s3-location <value>]
[--github-location <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

将 CodeDeploy 应用程序连接到存储 GitHub 库

在首次使用 GitHub 存储库部署应用程序之前 Amazon CLI，必须先授予代表您的 GitHub 账户与 GitHub 之交互的 CodeDeploy 权限。必须使用 CodeDeploy 控制台为每个应用程序完成一次此步骤。

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 选择应用程序。
3. 从应用程序中，选择要关联到 GitHub 用户帐户的应用程序，然后选择部署应用程序。

> **Note**
>
> 您没有创建部署。目前，这是授予代表您的 GitHub用户帐户进行交互 GitHub 的 CodeDeploy 权限的唯一途径。

4. 在 “存储库类型” 旁边，选择 “我的应用程序修订存储在” 中 GitHub。
5. 选择 “连接到” GitHub。

> **Note**
>
> 如果你看到 “Connect to 其他 GitHub账号” 链接：
> 您可能已经授权 CodeDeploy GitHub 代表该应用程序的另一个 GitHub 账户与之交互。
> 您可能已经撤销了 GitHub代表登录 GitHub 账户与中关联的所有应用程序进行交互的授权。 CodeDeploy CodeDeploy
> 有关更多信息，请参阅 GitHub 对 CodeDeploy 中的应用程序进行的身份验证。

6. 如果您尚未登录 GitHub，请按照 “登录” 页面上的说明进行操作。
7. 在 Authorize application 页上，选择 Authorize application。
8. 现在 CodeDeploy 已获得权限，请选择 “取消”，然后继续执行中的步骤创建 EC2 /本地计算平台部署 (CLI)。

通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation

您可以使用 Amazon CloudFormation 通过管理 Amazon ECS blue/green 部署 CodeDeploy。通过定义蓝绿资源并指定要在 Amazon CloudFormation中使用的流量路由和稳定设置来生成部署。本主题介绍由管理的 Amazon ECS blue/green 部署 CodeDeploy 和由管理的部署之间的区别 Amazon CloudFormation。

有关使用 Amazon CloudFormation 来管理 Amazon ECS 蓝/绿部署的 step-by-step说明，请参阅用户指南 Amazon CloudFormation中的 CodeDeploy 使用自动执行 ECS 蓝/绿部署。Amazon CloudFormation

> **Note**
>
> 亚太地区（大阪）区域 Amazon CloudFormation 不支持使用管理 Amazon ECS blue/green 部署。

通过和部署的 Amazon ECS 蓝/绿部署之间的区别 CodeDeploy Amazon CloudFormation

Amazon CloudFormation 堆栈模板对与 Amazon ECS 任务相关的资源和基础设施以及部署的配置选项进行了建模。因此，标准 Amazon ECS blue/green 部署和通过创建的 blue/green 部署之间存在差异 Amazon CloudFormation。

与标准 Amazon ECS blue/green 部署不同，您无需对以下内容进行建模或手动创建：

- 您不能通过指定唯一代表要部署的内容的名称来创建 Amazon CodeDeploy 应用程序。
- 您不创建 Amazon CodeDeploy 部署组。
- 您无需指定应用程序规范文件（AppSpec 文件）。通常使用该 AppSpec 文件管理的信息，例如加权配置选项或生命周期事件，由`AWS::CodeDeploy::BlueGreen`挂钩管理。

此表汇总了部署类型之间的高级工作流程中的差异。

函数	标准 blue/green 部署	蓝/绿部署 Amazon CloudFormation
指定 Amazon ECS 集群、Amazon ECS 服务、应用程序负载均衡器或网络负载均衡器、生产侦听器、测试侦听器和两个目标组。	创建指定这些资源的 CodeDeploy 部署组。	创建 Amazon CloudFormation 模板来对这些资源进行建模。
指定要部署的更改。	创建 CodeDeploy 应用程序。	创建指定容器镜像的 Amazon CloudFormation 模板。
指定 Amazon ECS 任务定义、容器名称和容器端口。	创建指定这些资源的 AppSpec 文件。	创建 Amazon CloudFormation 模板来对这些资源进行建模。
指定部署流量转移选项和生命周期事件挂钩。	创建一个指定这些选项的 AppSpec 文件。	创建一个使用`AWS::CodeDeploy::BlueGreen` 挂钩参数来指定这些选项的 Amazon CloudFormation 模板。

函数	标准 blue/green 部署	蓝/绿部署 Amazon CloudFormation
CloudWatch 警报。	创建触发回滚的 CloudWatch 警报。	在 Amazon CloudFormation 堆栈级别配置触发回滚的 CloudWatch 警报。
回滚/重新部署。	指定回滚和重新部署选项。	取消中的堆栈更新 Amazon CloudFormation。

通过以下方式监控 Amazon ECS blue/green 部署 Amazon CloudFormation

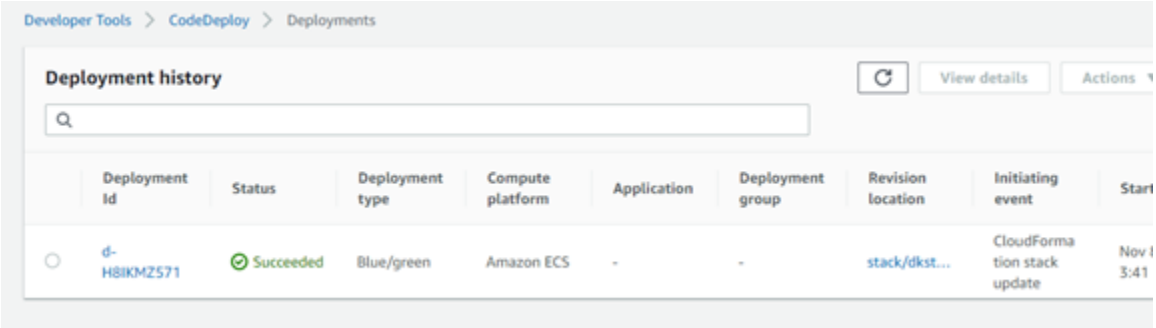
您可以通过 Amazon CloudFormation 和监控 blue/green 部署 CodeDeploy。有关通过监控的信息 Amazon CloudFormation，请参阅《Amazon CloudFormation 用户指南》 Amazon CloudFormation中的 “监控 blue/green 事件”。

要在中查看部署的 blue/green 部署状态 CodeDeploy

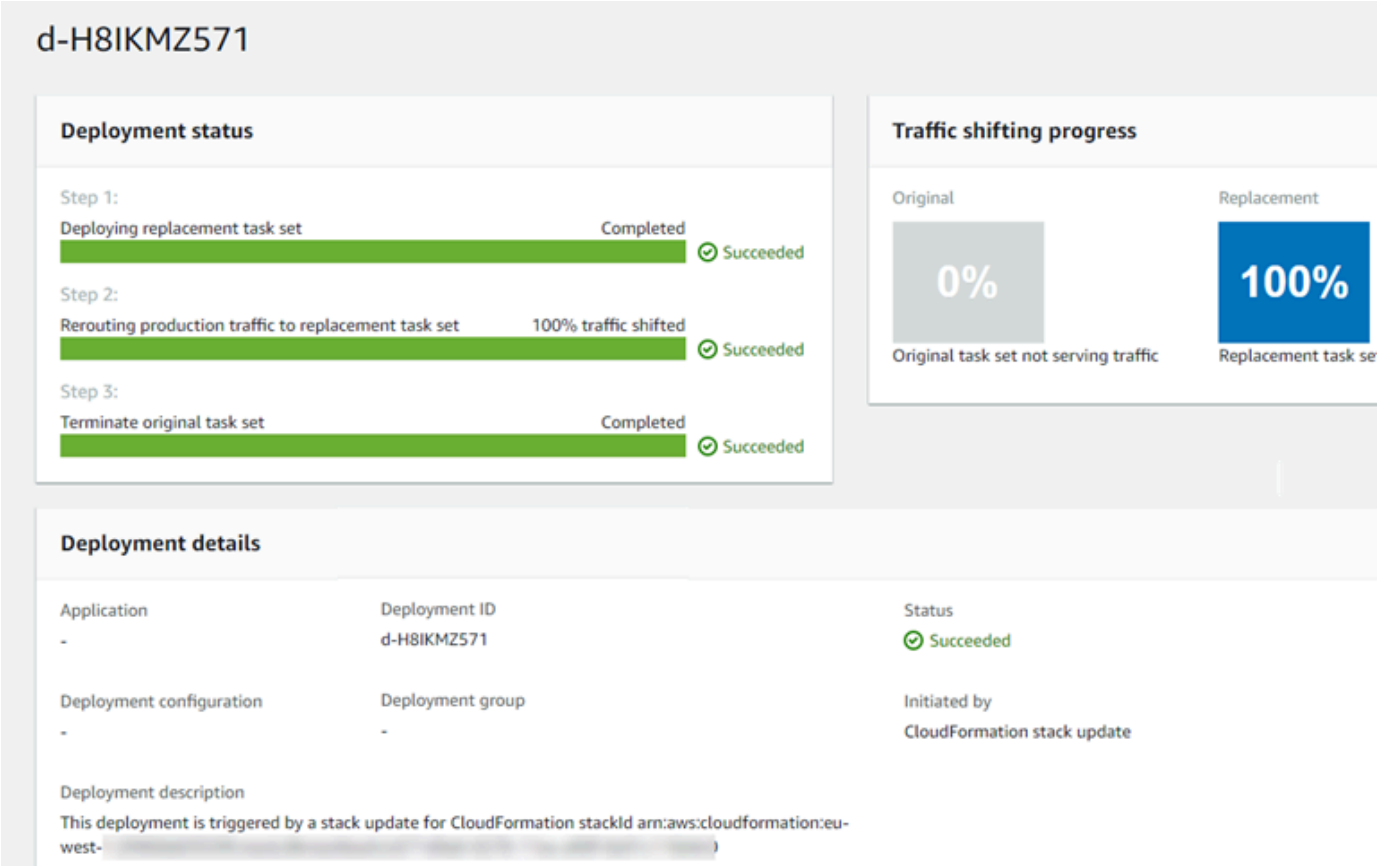
1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在 “部署” 中，将显示由 Amazon CloudFormation 堆栈更新触发的部署。选择部署以查看 Deployment history（部署历史记录）。



3. 选择部署以查看流量转移状态。请注意，不会创建应用程序和部署组。



4. 以下内容适用于回滚或停止部署：

   - 成功部署显示在中， CodeDeploy 并显示部署是由启动的 Amazon CloudFormation。
   - 如果要停止并回滚部署，则必须取消中的堆栈更新 Amazon CloudFormation。

查看 CodeDeploy 部署详细信息

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 查看有关与您的 Amazon 账户关联的部署的详细信息。

> **Note**
>
> 您可以在以下位置中查看您的实例的 EC2/本地部署日志：
>
> - Amazon Linux、RHEL 和 Ubuntu Server：`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`
> - Windows Server：C:\ProgramData\Amazon\CodeDeploy<DEPLOYMENT-GROUP-ID><DEPLOYMENT-ID>\logs\scripts.log

有关更多信息，请参阅 分析日志文件以调查针对实例的部署失败。

主题

- 查看部署详细信息（控制台）
- 查看部署详细信息（CLI）

查看部署详细信息（控制台）

要使用 CodeDeploy 控制台查看部署详细信息，请执行以下操作：

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择部署。

   > **Note**
   > 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

3. 要查看有关单个部署的更多详细信息，请在 Deployment history（部署历史记录）中，选择部署 ID 或选择部署 ID 旁边的按钮，然后选择 View（查看）。

查看部署详细信息（CLI）

要使用 Amazon CLI 查看部署详细信息，请调用 `get-deployment` 命令或 `batch-get-deployments` 命令。您可以调用 `list-deployments` 命令来获取要用作 `get-deployment` 命令和 `batch-get-deployments` 命令的输入的唯一部署 ID 的列表。

要查看有关单个部署的详细信息，请调用 get-deployment 命令，并指定唯一部署标识符。要获取部署 ID，请调用 list-deployments 命令。

要查看有关多个部署的详细信息，请调用 batch-get-deployments 命令，并指定多个唯一部署标识符。要获取部署 ID，请调用 list-deployments 命令。

要查看部署 ID 的列表，请调用 list-deployments 命令，并指定：

- 与部署关联的应用程序的名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 与部署关联的部署组的名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。
- （可选）是否按部署状态包含有关部署的详细信息。（如果未指定，则将列出所有匹配的部署，不管其部署状态如何。）
- （可选）是否按部署创建的开始时间和/或结束时间包含有关部署的详细信息。（如果未指定，则将列出所有匹配的部署，不管其创建时间如何。）

查看 CodeDeploy EC2/本地部署的日志数据

您可以通过以下方式查看 CodeDeploy 部署过程创建的日志数据：设置 Amazon CloudWatch 代理以便在 CloudWatch 控制台中查看聚合数据或者登录到单个实例以审核日志文件。

> **Note**
>
> 不支持生成 Amazon Lambda 或 Amazon ECS 部署的日志。只能为 EC2/本地部署创建日志。

主题

- 在 Amazon CloudWatch 控制台中查看日志文件数据
- 查看实例上的日志文件

在 Amazon CloudWatch 控制台中查看日志文件数据

在实例上安装 Amazon CloudWatch 代理后，可以在 CloudWatch 控制台中查看该实例上的所有部署的部署数据。为简单起见，我们建议您使用 CloudWatch 集中监控日志文件，而不是逐个实例查看这些文件。有关更多信息，请参阅 将 CodeDeploy 代理日志发送到 CloudWatch。

查看实例上的日志文件

要查看单个实例的部署日志数据，您可以登录实例并浏览有关错误或其他部署事件的信息。

主题

- 查看 Amazon Linux、RHEL 和 Ubuntu Server 实例上的部署日志文件
- 查看 Windows Server 实例上的部署日志文件

查看 Amazon Linux、RHEL 和 Ubuntu Server 实例上的部署日志文件

在 Amazon Linux、RHEL 和 Ubuntu Server 实例上，部署日志存储在以下位置：

`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`

要查看或分析 Amazon Linux、RHEL 和 Ubuntu Server 实例上的部署日志，请登录实例，然后键入以下命令打开 CodeDeploy 代理日志文件：

```
less /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

键入以下命令浏览日志文件以查看错误消息：

命令	结果
& ERROR	仅显示日志文件中的错误消息。在 **ERROR** 一词的前后使用一个空格。
/ ERROR	搜索下一条错误消息。[1]
? ERROR	搜索之前的错误消息。[2] 在单词 **ERROR** 前后使用一个空格。
G	转到日志文件的末尾。
g	转到日志文件的开头。
q	退出日志文件。
h	了解其他命令。

命令	结果
[1] 键入 **/ ERROR** 后，为下一条错误消息键入 **n**。为上一条错误消息键入 **N**。 [2] 键入 **n** 后，为下一条错误消息键入 **? ERROR** ，或为上一条错误消息键入 **N**。	

您也可以键入以下命令来打开 CodeDeploy 脚本日志文件：

```
less /opt/codedeploy-agent/deployment-root/deployment-group-ID/deployment-ID/logs/
scripts.log
```

键入以下命令浏览日志文件以查看错误消息：

命令	结果
&stderr	仅显示日志文件中的错误消息。
/stderr	搜索下一条错误消息。[1]
?stderr	搜索上一条错误消息。[2]
G	转到日志文件的末尾。
g	转到日志文件的开头。
q	退出日志文件。
h	了解其他命令。
[1] 键入 **/stderr** 后，为下一条错误消息键入 **n**。为上一条错误消息键入 **N**。 [2] 键入 **?stderr** 后，为上一条错误消息键入 **n**。为上一条错误消息键入 **N**。	

查看 Windows Server 实例上的部署日志文件

CodeDeploy 代理日志文件：在 Windows Server 实例上，CodeDeploy 代理日志文件存储在以下位置：

```
C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

要查看或分析 Windows Server 实例上的 CodeDeploy 代理日志文件，请登录实例，然后键入以下命令打开该文件：

```
notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

要浏览日志文件以查看错误消息，请按 CTRL+F，键入 **ERROR [**，然后按 Enter 以查找第一个错误。

CodeDeploy 脚本日志文件：在 Windows Server 实例上，部署日志存储在以下位置：

C:\ProgramData\Amazon\CodeDeploy*deployment-group-id**deployment-id*\logs\scripts.log

其中：

- *deployment-group-id* 是一个字符串，例如 examplebf3a9c7a-7c19-4657-8684-b0c68d0cd3c4
- *deployment-id* 是一个标识符，例如 d-12EXAMPLE

键入以下命令以打开 CodeDeploy 脚本日志文件：

```
notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\deployment-ID\logs
\scripts.log
```

要浏览日志文件以查看错误消息，请按 CTRL+F，键入 **stderr**，然后按 Enter 以查找第一个错误。

通过 CodeDeploy 停止部署

您可以使用 CodeDeploy 控制台、Amazon CLI 或 CodeDeploy API 停止与您的 Amazon 账户关联的部署。

> **Warning**
>
> 停止 EC2/本地部署可能会使您的部署组中的部分或全部实例处于不确定的部署状态。有关更多信息，请参阅 停止和失败的部署。

您可以停止部署，也可以停止和回滚部署。

- 停止部署（控制台）

- 停止部署（CLI）

> **Note**
> 如果您的部署是通过 Amazon CloudFormation 进行蓝绿部署，则无法在 CodeDeploy 控制台中执行此任务。转到 Amazon CloudFormation 控制台执行此任务。

停止部署（控制台）

1. 登录 Amazon Web Services 管理控制台并打开 CodeDeploy 控制台，网址为 https://console.aws.amazon.com/codedeploy。

   > **Note**
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择部署。

   > **Note**
   > 如果未显示任何条目，请确保选择了正确的区域。在全局导航栏上的区域选择器中，选择 Amazon Web Services 一般参考 的区域和终端节点中列出的某个区域。只有这些区域支持 CodeDeploy。

3. 选择要停止执行以下操作之一的部署：

   1. 选择 Stop deployment（停止部署）以停止部署而不进行回滚。
   2. 选择 Stop and roll back deployment（停止并回滚）以停止并回滚部署。

   有关更多信息，请参阅 使用 CodeDeploy 重新部署和回滚部署。

   > **Note**
   > 如果 Stop deployment（停止部署）和 Stop and roll back deployment（停止并回滚部署）不可用，则表示部署已进展到无法停止的时间点。

停止部署（CLI）

调用 stop-deployment 命令，并指定部署 ID。要查看部署 ID 的列表，请调用 list-deployments 命令。

使用 CodeDeploy 重新部署和回滚部署

CodeDeploy 回滚部署的方式是将以前的应用程序部署版本重新部署为新的部署。这些回滚部署在技术上属于新部署，具有新的部署 ID，不同于之前部署的存储版本。

可以自动或手动回滚部署。

主题

- 自动回滚
- 手动回滚
- 回滚和重新部署工作流程
- 现有内容的回滚行为

自动回滚

您可以对部署组或部署进行配置，使之在部署失败或达到您指定的监控阈值时自动回滚。在这种情况下，将会部署上一个已知良好的应用程序版本。您可以在创建应用程序或是创建或更新部署组时配置自动回滚。

创建新部署时，您还可以选择覆盖已为部署组指定的自动回滚配置。

> **Note**
>
> 可使用 Amazon Simple Notification Service 在部署自动回滚时接收通知。有关信息，请参阅Monitoring Deployments with Amazon SNS Event Notifications。

有关配置自动回滚的更多信息，请参阅为部署组配置高级选项。

手动回滚

如果您尚未设置自动回滚，可以通过以下方式手动回滚部署：创建一个使用以前部署的任何应用程序修订的新部署，然后根据步骤重新部署一个修订。如果应用程序进入了未知状态，您可能会这么做。您可

以将应用程序重新部署为已知工作状态，而不是花大量的时间排查故障。有关更多信息，请参阅 使用创建部署 CodeDeploy。

> **Note**
> 如果您从部署组中删除某个实例，CodeDeploy 不会卸载可能已安装在该实例上的任何内容。

回滚和重新部署工作流程

当启动自动回滚时，或在手动启动重新部署或手动回滚时，CodeDeploy 首先尝试从每个参与实例中删除上次成功安装的所有文件。为此，CodeDeploy 将检查下面的清理文件：

`/opt/codedeploy-agent/deployment-root/deployment-instructions/`*`deployment-group-ID`*`-cleanup` 文件（适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例）

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`*`deployment-group-ID`*`-cleanup` 文件（适用于 Windows Server 实例）

如果清理文件存在，则 CodeDeploy 将在开始新部署之前，使用它从实例中删除列出的所有文件。

例如，前面的两个文本文件和两个脚本文件已部署到运行 Windows Server 的 Amazon EC2 实例，并且这两个脚本在部署生命周期事件期间又创建了两个文本文件：

```
c:\temp\a.txt (previously deployed by CodeDeploy)
c:\temp\b.txt (previously deployed by CodeDeploy)
c:\temp\c.bat (previously deployed by CodeDeploy)
c:\temp\d.bat (previously deployed by CodeDeploy)
c:\temp\e.txt (previously created by c.bat)
c:\temp\f.txt (previously created by d.bat)
```

清理文件将仅列出前面的两个文本文件和两个脚本文件：

```
c:\temp\a.txt
c:\temp\b.txt
c:\temp\c.bat
c:\temp\d.bat
```

在新部署之前，CodeDeploy 将仅删除前面的两个文本文件和两个脚本文件，并保持最后两个文本文件不变：

```
c:\temp\a.txt will be removed
c:\temp\b.txt will be removed
c:\temp\c.bat will be removed
c:\temp\d.bat will be removed
c:\temp\e.txt will remain
c:\temp\f.txt will remain
```

作为此过程的一部分，CodeDeploy 将不会在后续重新部署（无论是手动还是自动回滚）期间尝试还原或以其他方式协调在以前的部署中由任何脚本所执行的任何操作。例如，如果 `c.bat` 和 `d.bat` 文件包含不重新创建 `e.txt` 和 `f.txt` 文件（如果它们已存在）的逻辑，则每当 CodeDeploy 在后续部署中运行 `c.bat` 和 `d.bat` 时，旧版本的 `e.txt` 和 `f.txt` 都将保持不变。您可以向 `c.bat` 和 `d.bat` 中添加逻辑，始终先检查并删除旧版本的 `e.txt` 和 `f.txt`，然后再创建新文件。

现有内容的回滚行为

作为部署过程的一部分，CodeDeploy 代理会从每个实例中删除由最新部署安装的所有文件。如果目标部署位置显示不是上一部署的一部分的文件，则可以选择 CodeDeploy 在下一个部署期间对这些文件执行哪些操作：

- 使部署失败 - 系统报告出错，并且部署状态更改为“失败”。
- 覆盖内容 - 来自应用程序修订的文件版本将替换实例上已有的版本。
- 保留内容 - 目标位置的文件将保留，并且应用程序修订中的版本不会复制到实例。

您可以在创建部署时选择此行为。如果在控制台中创建部署，请参阅创建 EC2 /本地计算平台部署（控制台）。如果使用 Amazon CLI 创建部署，请参阅创建 EC2 /本地计算平台部署 (CLI)。

您可以选择保留要作为下一个部署的一部分的文件，而无需将这些文件添加到应用程序修订包中。例如，您可以将部署所需的文件直接上传到实例，但这些文件不会添加到应用程序修订包。或者，如果应用程序已在生产环境中，但您想首次使用 CodeDeploy 来部署应用程序，则可将文件上传到实例。

对于回滚（其中由于部署失败，将重新部署最新的已成功部署的应用程序修订），上次成功部署的内容处理选项将应用于回滚部署。

但是，如果已将失败的部署配置为覆盖，而不是保留文件，则回滚期间可能出现意外结果。具体而言，部署失败可能会导致删除您预期保留的文件。当回滚部署运行时，这些文件未在实例上。

在以下示例中，有三种部署。在第三次部署期间再次部署应用程序修订 1 时，第二次失败部署期间覆盖（删除）的任何文件不再可用（无法保留）：

部署	应用程序修订	内容覆盖选项	部署状态	行为和结果
部署 1	应用程序修订 1	保留	成功	CodeDeploy 在目标位置检测上一部署中未部署的文件。这些文件可能特意放置在该位置以成为当前部署的一部分。它们将保留并记录为当前部署包的一部分。
部署 2	应用程序修订 2	覆盖	失败	在部署过程中，CodeDeploy 将删除作为上一成功部署的一部分的所有文件。这包括在部署 1 期间保留的文件。 但是，部署是因不相关的原因导致失败的。
部署 3	应用程序修订 1	保留		由于已为部署或部署组启用自动回滚，因此，CodeDeploy 将部署上一个已知正常的应用程序修订（应用程序修订 1）。 不过，您希望在部署 1 中保

部署	应用程序修订	内容覆盖选项	部署状态	行为和结果
				留的文件已在部署 2 失败之前被删除，且无法由 Amazon CodeDeploy 检索。您可以自行将这些文件添加到实例（如果应用程序修订 1 需要这些文件），也可以创建新的应用程序修订。

在其他 Amazon 账户中部署应用程序

Organizations 通常有多个用于不同目的的 Amazon 帐户（例如，一个用于系统管理任务，另一个用于开发、测试和生产任务，或者一个与开发和测试环境相关联，另一个与生产环境关联）。

尽管您可能在不同的账户中执行相关工作，但 CodeDeploy 部署组及其部署到的 Amazon EC2 实例与其创建时使用的账户严格关联。例如，您无法将在一个账户中启动的实例添加到另一个账户中的部署组。

假设你有两个 Amazon 账户：你的开发账户和你的生产账户。您主要使用开发账户进行工作，但是希望能够在生产账户中启动部署而无需完整的一组凭证，或者不希望注销开发账户并登录生产账户来进行工作。

完成以下跨账户配置步骤之后，您可以启动属于您组织的另一个账户下的部署，而无需另一个账户的一组完整权限。在此操作过程中，您需要使用 Amazon Security Token Service （Amazon STS）提供的功能，该功能可授予您对该账户的临时访问权限。

步骤 1：在任一账户中创建 S3 存储桶

在开发账户或生产账户中：

- 如果您还未创建，则创建将在其中存储生产账户的应用程序修订的 Amazon S3 存储桶。有关信息，请参阅在 Amazon S3 中创建存储桶。您甚至可以为两个账户使用相同的存储桶和应用程序修订，将您在开发账户中测试和验证的相同文件部署到您的生产环境。

步骤 2：将 Amazon S3 存储桶权限授予生产账户的 IAM 实例配置文件

如果您在步骤 1 中创建的 Amazon S3 存储桶位于您的生产账户中，则不需要此步骤。稍后您将承担的角色具有对此存储桶的访问权限，因为该存储桶也位于生产账户中。

如果您在开发账户中创建了 Amazon S3 存储桶，则执行以下操作：

- 在生产账户中，创建 IAM 实例配置文件。有关信息，请参阅步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

  > **Note**
  >
  > 记录此 IAM 实例配置文件的 ARN。您需要将它添加到接下来创建的跨存储桶策略中。

- 在开发账户中，将您在开发账户中创建的 Amazon S3 存储桶的访问权限授予刚刚在生产账户中创建的 IAM 实例配置文件。有关信息，请参阅示例 2：存储桶拥有者授予跨账户存储桶权限。

  完成授予跨账户存储桶权限的过程时，请注意以下内容：

  - 在示例演练中，账户 A 表示您的开发账户，账户 B 表示您的生产账户。
  - 当您执行账户 A（开发账户）任务时，修改以下存储桶策略以授予跨账户权限，而不是使用演练中提供的示例策略。

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Cross-account permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role-name"
            },
            "Action": [
                "s3:Get*",
                "s3:List*"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

*account-id*表示您刚刚在其中创建 IAM 实例配置文件的生产账户的账号。

*role-name*代表您刚刚创建的 IAM 实例配置文件的名称。

*bucket-name*表示您在步骤 1 中创建的存储桶的名称。请确保在存储桶名称之后包括 `/*`，以提供对存储桶中各个文件的访问权限。

步骤 3：在生产账户中创建资源和跨账户角色

在生产账户中：

- 按照本指南中的说明创建您的 CodeDeploy 资源（应用程序、部署组、部署配置、Amazon EC2 实例、Amazon 实例配置文件、服务角色等）。 EC2
- 创建一个额外的角色，即跨账户 IAM 角色，您的开发账户中的用户可以代入该角色来执行此生产账户中的 CodeDeploy 操作。

  使用演练：使用 IAM 角色委派跨 Amazon 账户访问权限作为指南，帮助您创建跨账户角色。与其将演练中的示例权限添加到策略文档中，不如将 Amazon 提供的以下两个策略附加到该角色：

  - `AmazonS3FullAccess`：只有当 S3 存储桶位于开发账户中时才需要。提供对开发账户（修订存储在其中）中的 Amazon S3 服务和资源具有完整访问权限的已代入生产账户角色。
  - `AWSCodeDeployDeployerAccess`：允许用户注册和部署修订。

  如果要创建和管理部署组而不是启动部署，请添加 `AWSCodeDeployFullAccess` 策略而不是 `AWSCodeDeployDeployerAccess` 策略。有关使用 IAM 托管策略授予 CodeDeploy 任务权限的更多信息，请参阅Amazon 的托管（预定义）策略 CodeDeploy。

  如果您希望在使用此跨账户角色时在其他 Amazon 服务中执行任务，可以附加其他策略。

> **Important**
>
> 在您创建跨账户 IAM 角色时，请记录详细信息，您需要这些详细信息来获取对生产账户的访问权限。
>
> 要使用 Amazon Web Services 管理控制台 来切换角色，您需要提供以下任一信息：
>
> - 用于通过所代入角色的凭证来访问生产账户的 URL。您可以在 Review 页面上找到该 URL，该页面在跨账户角色创建过程结束时显示。
> - 跨账户角色的名称以及账户 ID 编号或别名。
>
> 要使用 Amazon CLI 来切换角色，您需要提供以下信息：
>
> - 您将代入的跨账户角色的 ARN。

步骤 4：将应用程序修订上传到 Amazon S3 存储桶

在您创建了 Amazon S3 存储桶的账户中：

- 将您的应用程序修订上传到 Amazon S3 存储桶。有关信息，请参阅将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）。

步骤 5：代入跨账户角色和部署应用程序

在开发账户中，您可以使用 Amazon CLI 或代 Amazon Web Services 管理控制台 入跨账户角色并在生产账户中启动部署。

有关如何使用切换角色和启动部署的说明，请参阅切换到角色 (Amazon Web Services 管理控制台) 和创建 EC2 /本地计算平台部署（控制台）。 Amazon Web Services 管理控制台

有关如何使用担任跨账户角色和启动部署的说明，请参阅切换到 IAM 角色 (Amazon Command Line Interface) 和创建 EC2 /本地计算平台部署 (CLI)。 Amazon CLI

有关通过担任角色的更多信息 Amazon STS，请参阅《Amazon Security Token Service 用户指南》和《命令参考》AssumeRole中的 assume- role。Amazon CLI

相关主题：

- CodeDeploy: 从开发账户部署到生产账户

使用 CodeDeploy 代理验证本地机器上的部署程序包

您可以使用 CodeDeploy 代理在已登录的实例上部署内容。这样您就可以测试将要在部署中使用的应用程序规范文件（AppSpec 文件）的完整性，以及要部署的内容的完整性。

您不需要创建应用程序和部署组。如果您要部署的内容存储在本地实例中，您甚至不需要 Amazon 账户。对于最简单的测试，您可以在包含 AppSpec 文件和要部署内容的目录中运行 codedeploy-local 命令，而无需指定任何选项。工具中还有适用于其他测试用例的选项。

通过验证本地机器上的部署程序包，您可以：

- 测试应用程序修订的完整性。
- 测试 AppSpec 文件的内容。
- 利用您现有的应用程序代码首次试用 CodeDeploy。
- 登录实例后快速部署内容。

您要部署的内容可以存储在本地实例中，或存储在受支持的远程存储库类型（Amazon S3 存储桶或公有 GitHub 存储库）中。

先决条件

在开始本地部署之前，请先完成以下步骤：

- 创建或使用 CodeDeploy 代理支持的实例类型。有关信息，请参阅 CodeDeploy 代理支持的操作系统。
- 安装 CodeDeploy 代理的 1.0.1.1352 版本或更高版本。有关信息，请参阅安装 CodeDeploy 代理。
- 如果您要部署 Amazon S3 存储桶或 GitHub 存储库中的内容，请预置一个在 CodeDeploy 中使用的用户。有关信息，请参阅步骤 1：设置。
- 如果您要从 Amazon S3 存储桶部署应用程序修订，请在您工作的区域创建一个 Amazon S3 存储桶，并为该存储桶应用 Amazon S3 存储桶策略。此策略为您的实例授予下载应用程序修订所需的权限。

  例如，以下 Amazon S3 存储桶策略允许从名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任何位置，下载附加了 IAM 实例配置文件（其中包含 ARN `arn:aws:iam::444455556666:role/CodeDeployDemo`）的任意 Amazon EC2 实例：

```
{
    "Statement": [
```

```
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

以下 Amazon S3 存储桶策略允许从名为 `amzn-s3-demo-bucket` 的 Amazon S3 存储桶中的任意位置，下载具有关联 IAM 用户（其中包含 ARN `arn:aws:iam::444455556666:user/CodeDeployUser`）的任意本地实例：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

有关如何生成和附加 Amazon S3 存储桶策略的信息，请参阅存储桶策略示例。

- 如果您要从 Amazon S3 存储桶或 GitHub 存储库部署您的应用程序修订，请设置 IAM 实例配置文件并附加到实例。有关信息，请参阅步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件、为

CodeDeploy 创建 Amazon EC2 实例（Amazon CLI 或 Amazon EC2 控制台）和为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）。

- 如果您要从 GitHub 部署您的内容，请创建 GitHub 账户和公有存储库。要创建 GitHub 账户，请参阅联接 GitHub。要创建 GitHub 存储库，请参阅创建存储库。

> **Note**
>
> 目前不支持私有存储库。如果您的内容存储在私有 GitHub 存储库中，您可以将它下载到实例，并使用 `--bundle-location` 选项指定它的本地路径。

- 准备要部署到实例的内容（包括一个 AppSpec 文件），并将其放置在本地实例、Amazon S3 存储桶或 GitHub 存储库中。有关信息，请参阅正在处理的应用程序修订版 CodeDeploy。
- 对于其他配置选项，如果您希望使用默认值以外的其他值，请创建配置文件并将其放在实例中（对于 Amazon Linux、RHEL 或 Ubuntu Server 实例，该文件为 `/etc/codedeploy-agent/conf/codedeployagent.yml`；对于 Windows Server 实例，该文件为 `C:\ProgramData\Amazon\CodeDeploy\conf.yml`）。有关信息，请参阅CodeDeploy 代理配置参考。

> **Note**
>
> 如果您在 Amazon Linux、RHEL 或 Ubuntu Server 实例上使用了配置文件，则必须执行以下操作之一：
>
> - 对于部署根目录文件夹和日志目录文件夹，使用 `:root_dir:` 和 `:log_dir:` 变量指定默认位置以外的其他位置。
> - 使用 `sudo` 运行 CodeDeploy 代理命令。

创建本地部署

在要创建本地部署的实例上，打开终端会话（Amazon Linux、RHEL 或 Ubuntu Server 实例）或命令提示符（Windows Server）来运行工具命令。

> **Note**
>
> codedeploy-local 命令安装在以下位置：
>
> - 在 Amazon Linux、RHEL 或 Ubuntu Server 上：`/opt/codedeploy-agent/bin`。

- 在 Windows Server 上：`C:\ProgramData\Amazon\CodeDeploy\bin`。

基本命令语法

```
codedeploy-local [options]
```

摘要

```
codedeploy-local
[--bundle-location <value>]
[--type <value>]
[--file-exists-behavior <value>]
[--deployment-group <value>]
[--events <comma-separated values>]
[--agent-configuration-file <value>]
[--appspec-filename <value>]
```

选项：

-l，-bundle-location

应用程序修订数据包的位置。如果您没有指定位置，该工具将默认使用您当前的工作目录。如果为 `--bundle-location` 指定值，则必须为 `--type` 指定值。

数据包位置格式示例：

- 本地 Amazon Linux、RHEL 或 Ubuntu Server 实例：`/path/to/local/bundle.tgz`
- 本地 Windows Server 实例：`C:/path/to/local/bundle`
- Amazon S3 存储桶：`s3://amzn-s3-demo-bucket/bundle.tar`
- GitHub 存储库：`https://github.com/`*`account-name`*`/`*`repository-name`*`/`

-t, --type

应用程序修订数据包的格式。支持的类型包括 `tgz`、`tar`、`zip` 和 `directory`。如果您没有指定类型，该工具将默认使用 `directory`。如果为 `--type` 指定值，则必须为 `--bundle-location` 指定值。

-b, --file-exists-behavior

指明如何处理已存在于部署目标位置的文件（但并不是之前的成功部署放置的文件）。选项包括 DISALLOW、OVERWRITE、RETAIN。有关更多信息，请参阅 Amazon CodeDeploy API 参考中的 fileExistsBehavior。

-g, --deployment-group

要部署内容的目标位置的文件夹路径。如果您不指定文件夹，该工具将在部署根目录中创建一个名为 default-local-deployment-group 的文件夹。对于您创建的每个本地部署，该工具都会在此文件夹中创建一个子目录，名称示例为 d-98761234-local。

-e, --events

您要运行的一组覆盖生命周期事件挂钩（按顺序），而不是您在 AppSpec 文件中列出的事件。可指定多个挂钩，以逗号分隔。在以下情况下您可以使用此选项：

- 您希望在不必更新 AppSpec 文件的情况下运行另一组事件。
- 您希望将单一事件挂钩作为 AppSpec 文件中的例外来运行，例如 `ApplicationStop`。

如果您没有在覆盖列表中指定 DownloadBundle 和 Install 事件，它们会在您指定的所有事件挂钩之前运行。如果您在 `--events` 选项列表中包含DownloadBundle 和 Install ，只有 CodeDeploy 部署中正常情况下会在它们之前运行的事件才能位于它们之前。有关信息，请参阅AppSpec “挂钩” 部分。

-c, --agent-configuration-file

要用于部署的配置文件的位置（存储位置与默认位置不同时）。配置文件指定部署中默认值和默认行为的替代值和行为。

默认情况下，配置文件存储在 `/etc/codedeploy-agent/conf/codedeployagent.yml`（Amazon Linux、RHEL 或 Ubuntu Server 实例）或 `C:/ProgramData/Amazon/CodeDeploy/conf.yml`（Windows Server）中。有关更多信息，请参阅 CodeDeploy 代理配置参考。

-A, --appspec-filename

AppSpec 文件的名称。对于本地部署，可接受的值为 `appspec.yml` 和 `appspec.yaml`。默认情况下，AppSpec 文件称为 `appspec.yml`。

-h、--help

显示帮助内容的摘要。

-v, --version

显示工具的版本号。

示例

以下是有效命令格式的示例。

```
codedeploy-local
```

```
codedeploy-local --bundle-location /path/to/local/bundle/directory
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-group my-deployment-group
```

```
codedeploy-local --bundle-location /path/to/local/directory --type directory --deployment-group my-deployment-group
```

从 Amazon S3 部署捆绑包：

```
codedeploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.tgz --type tgz
```

```
codedeploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.zip?versionId=1234&etag=47e8 --type zip --deployment-group my-deployment-group
```

从公有 GitHub 存储库部署数据包：

```
codedeploy-local --bundle-location https://github.com/awslabs/aws-codedeploy-sample-tomcat --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/master --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/HEAD --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-sample-tomcat/zipball/1a2b3c4d --type zip
```

部署指定多个生命周期事件的数据包：

```
codedeploy-local --bundle-location /path/to/local/bundle.tar --type tar --application-
folder my-deployment --events DownloadBundle,Install,ApplicationStart,HealthCheck
```

使用 ApplicationStop 生命周期事件停止之前部署的应用程序：

```
codedeploy-local --bundle-location /path/to/local/bundle.tgz --type tgz --deployment-
group --events ApplicationStop
```

使用特定的部署组 ID 进行部署：

```
codedeploy-local --bundle-location C:/path/to/local/bundle/directory --deployment-group
 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-
group 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

监控中的部署 CodeDeploy

监控是维护 Amazon 解决方案的可靠性、可用性和性能的重要组成部分。 CodeDeploy 您应该从 Amazon 解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地对其进行调试。但是 CodeDeploy，在开始监控之前，您应该制定一份包含以下问题答案的监控计划：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步是通过测量不同时间和不同负载条件下的性能，为环境中的正常 CodeDeploy 性能建立基准。监控时 CodeDeploy，存储历史监控数据，以便您可以将其与当前性能数据进行比较，识别正常的性能模式和性能异常，并设计解决问题的方法。

例如，如果您正在使用 CodeDeploy，则可以监控部署和目标实例的状态。当部署或实例失败时，您可能需要重新配置应用程序规范文件、重新安装或更新 CodeDeploy 代理、更新应用程序或部署组中的设置，或者更改实例设置或文件。 AppSpec

要建立基准，您至少应监控以下各项：

- 部署事件和状态
- 实例事件和状态

自动监控工具

Amazon 提供了各种可用于监控的工具 CodeDeploy。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

您可以使用以下自动监控工具来监视 CodeDeploy 和报告何时出现问题：

- A CloudWatch mazon Alarms — 在您指定的时间段内观察单个指标，并根据该指标在多个时间段内相对于给定阈值的值执行一项或多项操作。该操作是发送到亚马逊简单通知服务 (Amazon SNS) Simple Notification Scaling 主题或亚马逊 A EC2 uto Scaling 政策的通知。 CloudWatch 警报不会仅

仅因为它们处于特定状态就调用操作；该状态必须已更改并保持了指定的时间段。有关更多信息，请参阅 Monitoring Deployments with Amazon CloudWatch Tools。

有关更新您的服务角色以配合使用 CloudWatch 警报监控的信息，请参阅向 CodeDeploy 服务角色授予 CloudWatch 权限。有关在 CodeDeploy 操作中添加 CloudWatch 警报监控的信息，请参阅使用创建应用程序 CodeDeploy使用创建部署组 CodeDeploy、或使用 CodeDeploy 更改部署组设置。

- Amazon CloudWatch Logs — 监控、存储和访问来自 Amazon CloudTrail 或其他来源的日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的监控日志文件。

  有关使用 CloudWatch 控制台查看 CodeDeploy日志的信息，请参阅在日志控制台中查看 CodeDeploy CloudWatch 日志。
- Amazon CloudWatch Events — 匹配事件并将其路由到一个或多个目标函数或流，以进行更改、捕获状态信息并采取纠正措施。有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的什么是亚马逊 CloudWatch 活动。

  有关在 CodeDeploy 操作中使用 CloudWatch 事件的信息，请参阅使用 Amazon CloudWatch 事件监控部署。
- Amazon CloudTrail 日志监控-在账户之间共享日志文件，通过将 CloudTrail 日志文件发送到 “日志” 来实时监控CloudWatch 日志文件，用 Java 编写日志处理应用程序，并验证您的日志文件在传送后是否未更改 CloudTrail。有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的 “使用 CloudTrail 日志文件”。

  有关 CloudTrail 与一起使用的信息 CodeDeploy，请参阅Monitoring Deployments。
- Amazon Simple Notification Service - 配置事件驱动的触发器，以接收有关部署和实例事件（如成功或失败）的短信或电子邮件通知。有关更多信息，请参阅创建主题和什么是 Amazon Simple Notification Service。

  有关为其设置 Amazon SNS 通知的信息 CodeDeploy，请参阅。Monitoring Deployments with Amazon SNS Event Notifications

手动监控工具

监控 CodeDeploy 的另一个重要部分是手动监控 CloudWatch 警报未涵盖的项目。 CodeDeploy CloudWatch、和其他 Amazon 控制台仪表板提供了 Amazon 环境状态的 at-a-glance视图。我们建议您同时检查 CodeDeploy 部署的日志文件。

- CodeDeploy 控制台显示：

  - 部署的状态。
  - 每个上次尝试和上次成功部署的版本的日期和事件
  - 部署中成功、失败、跳过或进行中的实例的数量
  - 本地实例的状态
  - 注册或注销本地实例的日期和时间
- CloudWatch 主页显示：
  - 当前告警和状态
  - 告警和资源图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务
  - 绘制指标数据图，以排除问题并弄清楚趋势
  - 搜索和浏览您的所有 Amazon 资源指标
  - 创建和编辑告警以接收问题通知

主题

- Monitoring Deployments with Amazon CloudWatch Tools
- Monitoring Deployments
- Monitoring Deployments with Amazon SNS Event Notifications

使用 Amazon CloudWatch 工具监控部署

您可以使用以下 CloudWatch 工具监控 CodeDeploy 部署：Amazon CloudWatch 事件、 CloudWatch 警报和亚马逊 CloudWatch 日志。

查看 CodeDeploy 代理和部署创建的日志可以帮助您排除部署失败的原因。除了一次查看一个实例上的 CodeDeploy 日志之外，您还可以使用 CloudWatch 日志在一个中心位置监控所有日志。

有关使用 CloudWatch 警报和 CloudWatch 事件监控 CodeDeploy 部署的信息，请参阅以下主题。

主题

- 使用 CloudWatch 警报监控部署 CodeDeploy

- 使用 Amazon CloudWatch 事件监控部署

使用 CloudWatch 警报监控部署 CodeDeploy

您可以为 CodeDeploy操作中使用的实例或 Amazon A EC2 uto Scaling 组创建 CloudWatch 警报。警报在您指定的时间段内监视单个指标，并根据该指标在多个时间段内相对于给定阈值的值执行一项或多项操作。 CloudWatch 警报的状态发生变化时会调用操作（例如，从变OK为ALARM）。

使用原生 CloudWatch 警报功能，您可以指定部署中使用的实例失败 CloudWatch 时支持的任何操作，例如发送 Amazon SNS 通知或停止、终止、重启或恢复实例。对于您的 CodeDeploy操作，您可以将部署组配置为在激活与部署组关联的任何 CloudWatch 警报时停止部署。

您最多可以将十个 CloudWatch 警报与一个 CodeDeploy 部署组相关联。如果任何指定警报激活，则部署将停止，状态将更新为 Stopped。要使用此选项，必须向您的 CodeDeploy 服务角色授予 CloudWatch 权限。

有关在 CloudWatch 控制台中设置 CloudWatch 警报的信息，请参阅亚马逊 CloudWatch 用户指南中的创建亚马逊 CloudWatch 警报。

有关在中将 CloudWatch 警报与部署组关联的信息 CodeDeploy，请参阅使用创建部署组 CodeDeploy和。使用 CodeDeploy 更改部署组设置

主题

- 向 CodeDeploy 服务角色授予 CloudWatch 权限

向 CodeDeploy 服务角色授予 CloudWatch 权限

在部署中使用 CloudWatch 警报监控之前，必须向您在 CodeDeploy 操作中使用的服务角色授予访问 CloudWatch 资源的权限。

向服务角色授予 CloudWatch 权限

1. 登录 Amazon Web Services 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 选择您在 Amazon CodeDeploy 操作中使用的服务角色的名称。
4. 在 Permissions 选项卡上的 Inline Policies 区域中，选择 Create Role Policy。

   –或者–

如果 Create Role Policy 按钮不可用，展开 Inline Policies 区域，然后选择 click here。

5. 在 Set Permissions 页面上，选择 Custom Policy，然后选择 Select。
6. 在 Review Policy 页面上的 Policy Name 字段中，键入一个名称以标识此策略，例如 `CWAlarms`。
7. 将以下内容粘贴到 Policy Document 字段中：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudwatch:DescribeAlarms",
            "Resource": "*"
        }
    ]
}
```

8. 选择应用策略。

使用 Amazon CloudWatch 事件监控部署

您可以使用 Amazon CloudWatch Events 来检测 CodeDeploy 操作中实例或部署（“事件”）状态的变化并做出反应。然后，根据您创建的规则，当部署或实例进入您在规则中指定的状态时， CloudWatch 事件将调用一个或多个目标操作。根据状态更改的类型，您可能想发送通知，捕获状态信息，采取纠正措施，启动事件或采取其他操作。在 CodeDeploy 操作中使用 CloudWatch 事件时，您可以选择以下类型的目标：

- Amazon Lambda 函数
- Kinesis Streams
- Amazon SQS 队列
- 内置目标（`EC2 CreateSnapshot API call`、`EC2 RebootInstances API call`、`EC2 StopInstances API call` 和 `EC2 TerminateInstances API call`）
- Amazon SNS 主题

下面是一些用例：

- 每当部署失败时使用 Lambda 函数向 Slack 通道传送通知。
- 将有关部署或实例的数据推送到 Kinesis 流，以支持全面、实时的状态监控。
- 当您指定的部署或 EC2实例事件发生时，使用 CloudWatch 警报操作自动停止、终止、重启或恢复 Amazon 实例。

本主题的其余部分描述了为创建 CloudWatch 事件规则的基本过程 CodeDeploy。但是，在创建用于 CodeDeploy 操作的事件规则之前，应执行以下操作：

- 完成 CloudWatch 活动先决条件。有关信息，请参阅 Amazon CloudWatch 活动先决条件。
- 熟悉事件中的事件、规则和目标。 CloudWatch 有关更多信息，请参阅什么是 Amazon CloudWatch 活动？ 和新 CloudWatch 事件 — 跟踪和响应 Amazon 资源的变化。
- 创建将在您的事件规则中使用的目标。

要为以下 CloudWatch 各项创建事件规则 CodeDeploy：

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 Events（事件）。
3. 选择创建规则，然后在事件选择器下选择 Amazon CodeDeploy。
4. 指定详细信息类型：
   - 要设置适用于所有实例和部署状态更改的规则，请选择 Any detail type，然后跳到步骤 6。
   - 要制定仅适用于实例的规则，请选择特定详细信息类型，然后选择CodeDeploy 实例状态更改通知。
   - 要制定仅适用于部署的规则，请选择特定详细信息类型，然后选择CodeDeploy 部署状态更改通知。
5. 指定规则适用的状态更改：
   - 要设置适用于所有状态更改的规则，请选择 Any state。
   - 要设置仅适用于部分状态更改的规则，请选择 Specific state（s），然后从列表中选择一个或多个状态值。下表列出了您可以选择的状态值：

部署状态值	实例状态值
FAILURE	FAILURE
START	START
STOP	就绪
QUEUED	成功
就绪	
成功	

6. 指定该规则适用于哪些 CodeDeploy 应用程序：
   - 要设置适用于所有应用程序的规则，请选择 Any application，然后跳到步骤 8。
   - 要设置仅适用于一个应用程序的规则，请选择 Specific application，然后从列表中选择该应用程序的名称。
7. 指定规则适用的部署组：
   - 要设置适用于与所选应用程序关联的所有部署组的规则，请选择 Any deployment group。
   - 要设置仅适用于与所选应用程序关联的一个部署组的规则，请选择 Specific deployment group（s），然后从列表中选择该部署组的名称。
8. 审查您的规则设置以确保其符合事件监控要求。
9. 在 Targets 区域，选择 Add target*。
10. 在 Select target type 列表中，选择您准备为此规则使用的目标类型，然后配置该类型所需的任何其他选项。
11. 选择 Configure details（配置详细信息）。
12. 在 Configure rule details 页面上，为规则键入名称和说明，然后选择 State 框以立即启用该规则。
13. 如果您对规则满意，请选择 Create rule。

使用 Amazon CloudTrail 监控部署

CodeDeploy 与 CloudTrail 集成在一起，后者是一种服务，它在 Amazon 账户中捕获由 CodeDeploy 或代表它发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 可捕获来自

CodeDeploy 控制台、通过 Amazon CLI 发送的 CodeDeploy 命令或直接来自 CodeDeploy API 的 API 调用。通过 CloudTrail 收集的信息，您可以确定向 CodeDeploy 发出了什么请求、发出请求的源 IP 地址、何人发出的请求以及发出请求的时间等。要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和启用），请参阅 Amazon CloudTrail 用户指南。

CloudTrail 中的 CodEDEPloy 信息

在您的 Amazon 账户中启用 CloudTrail 日志记录后，将在日志文件中跟踪对 CodeDeploy 操作执行的 API 调用。CodeDeploy 记录会与其他 Amazon 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建并写入新文件。

所有 CodeDeploy 操作都会被记录，并正式记载到 Amazon CodeDeploy 命令行参考和 Amazon CodeDeploy API 参考。例如，创建部署、删除应用程序和注册应用程序修订的调用会在 CloudTrail 日志文件中生成条目。

每个日志条目都包含有关生成请求的人员的信息。日志中的用户身份信息有助于确定发出的请求是否具有根或用户凭证，是否具有角色或联合用户的临时安全凭证，或者是否是由其他 Amazon 服务发出的。有关更多信息，请参阅 CloudTrail 日志事件参考中的 userIdentity 字段。

日志文件可以在存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。

可选择让 CloudTrail 在传输新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅为 CloudTrail 配置 Amazon Pinpoint 通知。

您也可以将多个 Amazon 区域和多个 Amazon 账户的 CodeDeploy 日志文件聚合到单个 Amazon S3 存储桶中。有关更多信息，请参阅接收来自多个区域的 CloudTrail 日志文件。

了解 CodeDeploy 日志文件条目

CloudTrail 日志文件可包含一个或多个日志条目，每个条目由多个 JSON 格式的事件组成。一个日志条目表示来自任何源的一个请求，包括有关所请求的操作、所有参数以及操作的日期和时间等信息。日志条目不一定具有任何特定顺序。也即，它们不是公用 API 调用的有序堆栈跟踪。

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了 CodeDeploy 创建部署组操作：

```
{
 "Records": [{
  "eventVersion": "1.02",
  "userIdentity": {
   "type": "AssumedRole",
   "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
```

```
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
     "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2014-11-27T03:57:36Z"
     },
     "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/example-role",
      "accountId": "123456789012",
      "userName": "example-role"
     }
    }
   },
   "eventTime": "2014-11-27T03:57:36Z",
   "eventSource": "codedeploy.amazonaws.com",
   "eventName": "CreateDeploymentGroup",
   "awsRegion": "us-west-2",
   "sourceIPAddress": "203.0.113.11",
   "userAgent": "example-user-agent-string",
   "requestParameters": {
    "applicationName": "ExampleApplication",
    "serviceRoleArn": "arn:aws:iam::123456789012:role/example-instance-group-role",
    "deploymentGroupName": "ExampleDeploymentGroup",
    "ec2TagFilters": [{
                 "value": "CodeDeployDemo",
     "type": "KEY_AND_VALUE",
     "key": "Name"
             }],
             "deploymentConfigName": "CodeDeployDefault.HalfAtATime"
   },
   "responseElements": {
    "deploymentGroupId": "7d64e680-e6f4-4c07-b10a-9e117EXAMPLE"
   },
   "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
   "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
   "eventType": "AwsApiCall",
   "recipientAccountId": "123456789012"
  },
     ... additional entries ...
     ]
```

```
}
```

使用 Amazon SNS 事件通知监控部署

您可以向 CodeDeploy 部署组添加触发器，以接收与该部署组中的部署或实例相关的事件的通知。对于您加入到该触发器操作中的 Amazon SNS 主题，通知将发送到已订阅该主题的接收人。

您可以通过 SMS 消息或电子邮件接收 CodeDeploy 事件通知。您也可以通过其他方式使用在指定事件发生时创建的 JSON 数据，如发送消息到 Amazon SQS 队列或调用 Amazon Lambda中的函数。若要查看为部署和实例触发器提供的 JSON 数据的结构，请参阅 CodeDeploy 触发器的 JSON 数据格式。

在以下情况下，您可以选择使用触发器来接收通知：

- 您是开发人员，需要知道部署失败或停止的时间，以便进行问题排查。
- 您是一名系统管理员，需要知道有多少实例出现故障，以便监控您的 Amazon EC2 队列的运行状况。
- 你是一名经理，需要 at-a-glance计算部署和实例事件，你可以通过筛选规则来获得这些信息，这些规则将不同类型的通知发送到桌面电子邮件客户端中的文件夹。

对于以下任一事件类型，您最多可以为每个 CodeDeploy 部署组创建 10 个触发器。

部署事件	实例事件
• 成功 • Failure • 已启动 • Stopped（已停止） • 回滚 • 就绪[1] • 所有部署事件	• 成功 • Failure • 已启动 • 就绪[1] • 所有实例事件

[1] 仅适用于 blue/green 部署。表示已在替换环境中的实例上安装最新应用程序修订并且现在可以在负载均衡器的后面重新路由来自原始环境的流量。有关更多信息，请参阅 在中处理部署 CodeDeploy。

主题

- 向服务角色授予 Amazon SNS 权限 CodeDeploy
- 为 CodeDeploy 事件创建触发器
- 在 CodeDeploy 部署组中编辑触发器
- 从 CodeDeploy 部署组中删除触发器
- CodeDeploy 触发器的 JSON 数据格式

向服务角色授予 Amazon SNS 权限 CodeDeploy

在触发器生成通知之前，必须向您在 CodeDeploy 操作中使用的服务角色授予访问 Amazon SNS 资源的权限。

向服务角色授予 Amazon SNS 权限

1. 登录 Amazon Web Services 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 选择您在 Amazon CodeDeploy 操作中使用的服务角色的名称。
4. 在 Permissions 选项卡上的 Inline Policies 区域中，选择 Create Role Policy。

   –或者–

   如果 Create Role Policy 按钮不可用，展开 Inline Policies 区域，然后选择 click here。
5. 在 Set Permissions 页面上，选择 Custom Policy，然后选择 Select。
6. 在审查策略页面上的策略名称字段中，输入一个名称以标识此策略（如 `SNSPublish`）。
7. 将以下内容粘贴到 Policy Document 字段中：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
```

```
            "Resource": "*"
        }
    ]
}
```

8. 选择应用策略。

为 CodeDeploy 事件创建触发器

您可以创建一个针对 Amazon CodeDeploy 部署或实例事件发布 Amazon Simple Notification Service（Amazon SNS）主题的触发器。然后，当该事件发生时，关联主题的所有订阅者都通过主题中指定的终端节点（如短信或电子邮件）接收通知。Amazon SNS 提供多种订阅主题的方式。

在创建触发器之前，必须设置触发器指向的 Amazon SNS 主题。有关信息，请参阅创建主题。在创建主题时，我们建议您为主题指定一个标识其用途的名称，并采用诸如 `Topic-group-us-west-3-deploy-fail` 或 `Topic-group-project-2-instance-stop` 这样的格式。

您还必须向 CodeDeploy 服务角色授予 Amazon SNS 权限，然后才能为您的触发器发送通知。有关信息，请参阅向服务角色授予 Amazon SNS 权限 CodeDeploy 。

创建主题后，您可以添加订阅者。有关创建、管理和订阅主题的信息，请参阅什么是 Amazon Simple Notification Service。

创建触发器以发送 CodeDeploy 事件通知（控制台）

您可以使用 CodeDeploy 控制台为 CodeDeploy 事件创建触发器。在设置过程结束时，将发送一条测试通知消息，以确保权限和触发器详细信息均已正确设置。

为 CodeDeploy 事件创建触发器

1. 在中 Amazon Web Services 管理控制台，打开 Amazon CodeDeploy 控制台。
2. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

   > **Note**
   >
   > 使用您在入门 CodeDeploy中设置的同一用户登录。

3. 在导航窗格中，展开部署，然后选择应用程序。

4. 在 Applications（应用程序）页面上，选择与您要在其中添加触发器的部署组关联的应用程序的名称。
5. 在 Application details（应用程序）页面上，选择您要在其中添加触发器的部署组。
6. 选择编辑。
7. 展开 Advanced - optional（高级 - 可选）。
8. 在 Triggers（触发器）区域中，选择 Create trigger（创建触发器）。
9. 在 Create deployment trigger（创建部署触发器）窗格中，执行以下操作：
   a. 在触发器名称中，输入触发器的名称以便于标识其用途。我们建议采用 `Trigger-group-us-west-3-deploy-fail` 或 `Trigger-group-eu-central-instance-stop` 这样的格式。
   b. 在事件中，选择将触发 Amazon SNS 主题发送通知的事件类型。
   c. 在 Amazon SNS 主题中，选择您为发送此触发器的通知而创建的主题的名称。
   d. 选择创建触发器。 CodeDeploy 发送测试通知，确认您已正确配置与 Amazon SNS 主题 CodeDeploy 之间的访问权限。如果您已订阅该主题，您会收到通过 SMS 消息或电子邮件发送的确认信息，具体取决于您为该主题选择的终端结点类型。
10. 选择保存更改。

创建触发器以发送 CodeDeploy 事件通知 (CLI)

您可以在创建部署组时使用 CLI 来加入触发器，也可以将触发器添加到现有部署组。

创建触发器以发送关于新部署组的通知

创建 JSON 文件来配置部署组，然后使用`--cli-input-json`选项运行该create-deployment-group命令。

创建 JSON 文件最简单的方式是使用 `--generate-cli-skeleton` 选项获取 JSON 格式的副本，然后在纯文本编辑器中提供所需的值。

1. 运行以下命令，然后将结果复制到纯文本编辑器中。

```
aws deploy create-deployment-group --generate-cli-skeleton
```

2. 将现有 CodeDeploy 应用程序的名称添加到输出中：

```
{
```

```
    "applicationName": "TestApp-us-east-2",
    "deploymentGroupName": "",
    "deploymentConfigName": "",
    "ec2TagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "onPremisesInstanceTagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "autoScalingGroups": [
        ""
    ],
    "serviceRoleArn": "",
    "triggerConfigurations": [
        {
            "triggerName": "",
            "triggerTargetArn": "",
            "triggerEvents": [
                ""
            ]
        }
    ]
}
```

3. 为您要配置的参数提供值。

   使用该create-deployment-group命令时，必须至少提供以下参数的值：

   - `applicationName`：已在您的账户中创建的应用程序的名称。
   - `deploymentGroupName`：您正在创建的部署组的名称。
   - `serviceRoleArn`：在您的账户中设置的现有服务角色的 CodeDeploy ARN。有关信息，请参阅步骤 2：为创建服务角色 CodeDeploy。

   在 `triggerConfigurations` 部分，为以下参数提供值：

- `triggerName`：您要为触发器给定的便于标识的名称。我们建议采用 `Trigger-group-us-west-3-deploy-fail` 或 `Trigger-group-eu-central-instance-stop` 这样的格式。
- `triggerTargetArn`：您创建的要与触发器关联的 Amazon SNS 主题的 ARN，采用以下格式：`arn:aws:sns:us-east-2:444455556666:NewTestTopic`。
- `triggerEvents`：您要为其触发通知的事件的类型。您可以指定一个或多个事件类型，多个事件类型名称用逗号分隔（例如，`"triggerEvents": ["DeploymentSuccess","DeploymentFailure","InstanceFailure"]`）。当您添加多个事件类型时，所有这些类型的通知都将发送到您指定的主题，而不是分别发送到不同的主题。您可以从以下事件类型中选择：
  - DeploymentStart
  - DeploymentSuccess
  - DeploymentFailure
  - DeploymentStop
  - DeploymentRollback
  - DeploymentReady（仅适用于 blue/green 部署中的替换实例）
  - InstanceStart
  - InstanceSuccess
  - InstanceFailure
  - InstanceReady（仅适用于 blue/green 部署中的替换实例）

以下配置示例将为一个名为 `TestApp-us-east-2` 的应用程序创建一个名为 `dep-group-ghi-789-2` 的部署组，并创建一个触发器，在部署启动、成功或失败时，该触发器都提示发送通知：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "deploymentGroupName": "dep-group-ghi-789-2",
    "ec2TagFilters": [
        {
            "Key": "Name",
            "Value": "Project-ABC",
            "Type": "KEY_AND_VALUE"
        }
    ],
```

```
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "triggerConfigurations": [
        {
            "triggerName": "Trigger-group-us-east-2",
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ]
        }
    ]
}
```

4. 以 JSON 文件格式保存更新，然后在运行 create-deployment-group 命令时使用 `--cli-input-json` 选项调用该文件：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

在创建过程结束时，您会收到一条测试通知消息，指示权限和触发器详细信息均已正确设置。

创建触发器以发送关于现有部署组的通知

要使用将 Amazon CLI CodeDeploy 事件触发器添加到现有部署组，请创建一个 JSON 文件来更新部署组，然后使用`--cli-input-json`选项运行update-deployment-group命令。

创建 JSON 文件最简单的方式是运行 get-deployment-group 命令以获取部署组配置的副本（采用 JSON 格式），然后在纯文本编辑器中更新参数值。

1. 运行以下命令，然后将结果复制到纯文本编辑器中。

```
aws deploy get-deployment-group --application-name application --deployment-group-
name deployment-group
```

2. 从输出中删除以下内容：

   - 在输出的开头处，删除 `{ "deploymentGroupInfo":`。
   - 在输出的结尾处，删除 `}`。
   - 删除包含 `deploymentGroupId` 的行。
   - 删除包含 `deploymentGroupName` 的行。

   现在，您的文本文件的内容看起来应类似于以下内容：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. 在 `triggerConfigurations` 部分，为 `triggerEvents`、`triggerTargetArn` 和 `triggerName` 参数添加数据。有关触发器配置参数的信息，请参见TriggerConfig。

   现在，您的文本文件的内容看起来应类似于以下内容。在部署启动、成功或失败时，此代码都会提示发送通知。

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
```

```
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

4. 将更新保存为 JSON 文件，然后使用`--cli-input-json`选项运行update-deployment-group命令。请务必包含该`--current-deployment-group-name`选项，并将您的 JSON 文件名替换*filename*为：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws deploy update-deployment-group --current-deployment-group-name deployment-
group-name --cli-input-json file://filename.json
```

在创建过程结束时，您会收到一条测试通知消息，指示权限和触发器详细信息均已正确设置。

在 CodeDeploy 部署组中编辑触发器

如果您的通知要求更改，您可以修改触发器而不必创建新的触发器。

修改 CodeDeploy触发器 (CLI)

Amazon CLI 要在更新部署组时使用更改 CodeDeploy 事件的触发器详细信息，请创建一个 JSON 文件来定义对部署组属性的更改，然后运行带有`--cli-input-json`选项的update-deployment-group命令。

创建 JSON 文件最简单的方式是运行 get-deployment-group 命令以获取当前部署组详细信息（采用 JSON 格式），然后在纯文本编辑器中编辑所需的值。

1. 运行以下命令，用应用程序和部署组的名称代替*application*和*deployment-group*：

```
aws deploy get-deployment-group --application-name application --deployment-group-name deployment-group
```

2. 将命令的结果复制到纯文本编辑器中，然后删除以下内容：

   - 在输出的开头处，删除 `{ "deploymentGroupInfo":`。
   - 在输出的结尾处，删除 `}`。
   - 删除包含 `deploymentGroupId` 的行。
   - 删除包含 `deploymentGroupName` 的行。

   现在，您的文本文件的内容看起来应类似于以下内容：

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "East-1-Instances",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure",
                "DeploymentStop"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:111222333444:Trigger-group-us-east-2",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
```

```
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. 根据需要更改任意参数。有关触发器配置参数的信息，请参见TriggerConfig。
4. 将更新保存为 JSON 文件，然后使用`--cli-input-json`选项运行update-deployment-group命令。请务必包含该`--current-deployment-group-name`选项，并将您的 JSON 文件名替换*filename*为：

> **Important**
>
> 务必在文件名前包含 `file://`。此命令中需要该项。

```
aws deploy update-deployment-group --current-deployment-group-name deployment-group-name --cli-input-json file://filename.json
```

在创建过程结束时，您会收到一条测试通知消息，指示权限和触发器详细信息均已正确设置。

从 CodeDeploy 部署组中删除触发器

由于每个部署组的触发器数量限制为 10 个，因此您可能需要删除不再使用的触发器。您无法撤消删除触发器的操作，但可以重新创建一个触发器。

从部署组中删除触发器（控制台）

1. 登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/codede ploy 上打开 CodeDeploy 控制台。

> **Note**
>
> 使用您在入门 CodeDeploy中设置的同一用户登录。

2. 在导航窗格中，展开部署，然后选择应用程序。
3. 在 Applications（应用程序）页面上，选择与您要在其中删除触发器的部署组关联的应用程序的名称。
4. 在 Application details（应用程序）页面上，选择您要在其中删除触发器的部署组。
5. 选择编辑。

6. 展开 Advanced - optional（高级 - 可选）。
7. 在 Triggers（触发器）区域中，选择要删除的触发器，然后选择 Delete trigger（删除触发器）。
8. 选择保存更改。

从部署组中删除触发器（CLI）

要使用 CLI 删除触发器，请使用空的触发器配置参数调用 update-deployment-group 命令，并指定：

- 与部署组关联的应用程序的名称。要查看应用程序名称的列表，请调用 list-applications 命令。
- 与应用程序关联的部署组的名称。要查看部署组名称的列表，请调用 list-deployment-groups 命令。

例如：

```
aws deploy update-deployment-group --application-name application-name --current-
deployment-group-name deployment-group-name --trigger-configurations
```

CodeDeploy 触发器的 JSON 数据格式

您可以使用在自定义通知工作流程中激活部署或实例的触发器时创建的 JSON 输出，如发送消息到 Amazon SQS 队列或调用 Amazon Lambda中的函数。

> **Note**
>
> 本指南不解决如何使用 JSON 配置通知的问题。有关使用 Amazon SNS 将消息发送到 Amazon SQS 队列的信息，请参阅将 Amazon SNS 消息发送至 Amazon SQS 队列。有关使用 Amazon SNS 调用 Lambda 函数的信息，请参阅使用 Amazon SNS 通知调用 Lambda 函数。

以下示例将显示适用于 CodeDeploy 触发器的 JSON 输出的结构。

适用于基于实例的触发器的示例 JSON 输出

```
{
    "region": "us-east-2",
    "accountId": "111222333444",
    "eventTriggerName": "trigger-group-us-east-instance-succeeded",
    "deploymentId": "d-75I7MBT7C",
    "instanceId": "arn:aws:ec2:us-east-2:444455556666:instance/i-496589f7",
```

```
    "lastUpdatedAt": "1446744207.564",
    "instanceStatus": "Succeeded",
    "lifecycleEvents": [
        {
            "LifecycleEvent": "ApplicationStop",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744188.595",
            "EndTime": "1446744188.711"
        },
        {
            "LifecycleEvent": "BeforeInstall",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744189.827",
            "EndTime": "1446744190.402"
        }
//More lifecycle events might be listed here
    ]
}
```

适用于基于部署的触发器的示例 JSON 输出

```
{
    "region": "us-west-1",
    "accountId": "111222333444",
    "eventTriggerName": "Trigger-group-us-west-3-deploy-failed",
    "applicationName": "ProductionApp-us-west-3",
    "deploymentId": "d-75I7MBT7C",
    "deploymentGroupName": "dep-group-def-456",
    "createTime": "1446744188.595",
    "completeTime": "1446744190.402",
    "deploymentOverview": {
        "Failed": "10",
        "InProgress": "0",
        "Pending": "0",
        "Skipped": "0",
        "Succeeded": "0"
    },
    "status": "Failed",
    "errorInformation": {
        "ErrorCode": "IAM_ROLE_MISSING",
        "ErrorMessage": "IAM Role is missing for deployment group: dep-group-def-456"
    }
}
```

安全性 Amazon CodeDeploy

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 Amazon 的共同责任。责任共担模式将其描述为云的 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 Amazon 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用的合规计划 Amazon CodeDeploy，请参阅按合规计划划分的范围内的 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。

本文档可帮助您了解在使用时如何应用分担责任模型 CodeDeploy。以下主题向您介绍如何进行配置 CodeDeploy 以满足您的安全和合规性目标。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 CodeDeploy 资源。

主题

- 中的数据保护 Amazon CodeDeploy
- 对 Amazon CodeDeploy进行身份和访问管理
- 登录和监控 CodeDeploy
- 合规性验证 Amazon CodeDeploy
- 韧性在 Amazon CodeDeploy
- 中的基础设施安全 Amazon CodeDeploy

中的数据保护 Amazon CodeDeploy

分 Amazon 分担责任模型适用于中的数据保护 Amazon CodeDeploy。如本模型所述 Amazon ，负责保护运行所有内容的全球基础架构 Amazon Web Services 云。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 Amazon Web Services 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 Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。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）。
- 用于 SSL/TLS 与 Amazon 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 Amazon CloudTrail。有关使用 CloudTrail 跟踪捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 Amazon 加密解决方案以及其中的所有默认安全控件 Amazon Web Services 服务。
- 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 中的敏感数据。
- 如果您在 Amazon 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://www.amazonaws.cn/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API CodeDeploy 或以其他 Amazon Web Services 服务 方式使用控制台 Amazon CLI、API 或时 Amazon SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

互联网络流量隐私

CodeDeploy 是一项完全托管的部署服务，支持 EC2 实例、Lambda 函数、Amazon ECS 和本地服务器。对于 EC2 实例和本地服务器，基于主机的代理使用 TLS 与 CodeDeploy 之通信。

当前，代理与服务的通信需要出站 Internet 连接，这样代理才能与公共终端节点 CodeDeploy 和 Amazon S3 服务终端节点通信。在 Virtual Private Cloud 中，这可以通过互联网网关、到企业网络的站点到站点 VPN 连接或直接连接来实现。

该 CodeDeploy 代理支持 HTTP 代理。

由提供支持的 Amazon PrivateLink Amazon VPC 终端节点可在某些 CodeDeploy 地区使用。有关更多信息，请参阅 将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用。

> **Note**
>
> 只有当您部署到 Amazon EC2 /本地计算平台时，才需要使用 CodeDeploy 代理。使用 Amazon ECS 或 Amazon Lambda 计算平台的部署不需要代理。

静态加密

客户代码未存储在 CodeDeploy。作为一项部署服务， CodeDeploy 正在向在 EC2 实例或本地服务器上运行的 CodeDeploy 代理发送命令。然后， CodeDeploy 代理使用 TLS 执行命令。部署、部署配置、部署组、应用程序和应用程序修订的服务模型数据存储在 Amazon DynamoDB 中，并使用 Amazon 拥有的密钥由拥有和管理的静态加密。 CodeDeploy有关更多信息，请参阅 Amazon 拥有的密钥。

传输中加密

CodeDeploy 代理通过端口 443 启动所有通信。 CodeDeploy 该代理会轮询 CodeDeploy 并侦听命令。该 CodeDeploy 代理是开源的。所有 service-to-service client-to-service通信在传输过程中均使用 TLS 进行加密。这样可以保护在与 Amazon S3 CodeDeploy 等其他服务之间传输的客户数据。

加密密钥管理

没有可供您管理的加密密钥。 CodeDeploy 服务模型数据使用由进行加密 Amazon 拥有的密钥，由其拥有和管理 CodeDeploy。有关更多信息，请参阅 Amazon 拥有的密钥。

对 Amazon CodeDeploy进行身份和访问管理

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 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 CodeDeploy 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
- 如何 Amazon CodeDeploy 与 IAM 配合使用
- Amazon 的托管（预定义）策略 CodeDeploy

- CodeDeploy Amazon 托管策略的更新
- Amazon CodeDeploy 基于身份的策略示例
- 排除 Amazon CodeDeploy 身份和访问问题
- CodeDeploy 权限参考
- 防止跨服务混淆座席

受众

您的使用方式 Amazon Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请向管理员申请权限（请参阅排除 Amazon CodeDeploy 身份和访问问题）
- 服务管理员 - 确定用户访问权限并提交权限请求（请参阅 如何 Amazon CodeDeploy 与 IAM 配合使用）
- IAM 管理员 - 编写用于管理访问权限的策略（请参阅 Amazon CodeDeploy 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 用户身份进行身份验证 Amazon Web Services 账户根用户，或者通过担任 IAM 角色进行身份验证。

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

Amazon Web Services 账户 root 用户

创建时 Amazon Web Services 账户，首先会有一个名为 Amazon Web Services 账户 root 用户的登录身份，该身份可以完全访问所有资源 Amazon Web Services 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

用户和组

IAM 用户是对单个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 Amazon 使用临时证书进行访问。

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

IAM 角色

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

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon 上运行的应用程序非常有用。 EC2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。

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。策略定义了与身份或资源关联时的权限。 Amazon 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 Amazon 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以对什么资源以及在什么条件下执行操作，来指定谁有权访问什么内容。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以代入这些角色。IAM 策略定义权限，而不考虑您使用哪种方法来执行操作。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可在何种条件下对哪些资源执行什么操作。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。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。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。

其他策略类型

Amazon 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，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)-在中指定组织或组织单位的最大权限 Amazon Organizations。有关更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在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。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。

如何 Amazon CodeDeploy 与 IAM 配合使用

在使用 IAM 管理访问权限之前 CodeDeploy，您应该了解哪些 IAM 功能可供使用 CodeDeploy。有关更多信息，请参阅《IAM 用户指南》中的与 IAM 结合使用的Amazon 服务。

主题

- CodeDeploy 基于身份的策略
- CodeDeploy 基于资源的政策
- 基于 CodeDeploy 标签的授权
- CodeDeploy IAM 角色

CodeDeploy 基于身份的策略

借助 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。CodeDeploy 支持操作、资源和条件键。如需在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

操作

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。在策略中包含操作以授予执行关联操作的权限。

正在执行的策略操作在操作之前 CodeDeploy 使用`codedeploy:`前缀。例如，`codedeploy:GetApplication` 权限授予用户执行 `GetApplication` 操作的权限。策略声明必须包含`Action`或`NotAction`元素。 CodeDeploy 定义了自己的一组操作，这些操作描述了您可以使用此服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "codedeploy:action1",
      "codedeploy:action2"
```

您也可以使用通配符（*）指定多个操作。例如，包括以下操作可以指定以单词 `Describe` 开头的所有操作：

```
"Action": "ec2:Describe*"
```

有关 CodeDeploy 操作列表，请参阅 IAM 用户指南 Amazon CodeDeploy中的由定义的操作。

有关列出所有 CodeDeploy API 操作及其适用的资源的表格，请参阅CodeDeploy 权限参考。

资源

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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

例如，您可以在语句中使用部署组 (*myDeploymentGroup*) 的 ARN 来指明部署组 ()，如下所示：

```
"Resource": "arn:aws:codedeploy:us-
west-2:123456789012:deploymentgroup:myApplication/myDeploymentGroup"
```

您还可以使用通配符（*）指定属于某个账户的所有部署组，如下所示：

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*"
```

要指定所有资源，或者如果 API 操作不支持 ARNs，请在`Resource`元素中使用通配符 (*)，如下所示：

```
"Resource": "*"
```

某些 CodeDeploy API 操作接受多个资源（例如，`BatchGetDeploymentGroups`）。要在单个语句中指定多个资源，请 ARNs 用逗号分隔它们，如下所示：

```
"Resource": ["arn1", "arn2"]
```

CodeDeploy 提供了一组使用 CodeDeploy资源的操作。有关可用操作的列表，请参阅 CodeDeploy 权限参考。

有关 CodeDeploy 资源类型及其类型的列表 ARNs，请参阅 IAM 用户指南 Amazon CodeDeploy中的由定义的资源。有关可指定每个资源的 ARN 的操作的信息，请参阅 Amazon CodeDeploy定义的操作。

CodeDeploy 资源和运营

在中 CodeDeploy，主要资源是部署组。在策略中，您可以使用 Amazon 资源名称 (ARN) 来标识该政策适用的资源。 CodeDeploy 支持可用于部署组的其他资源，包括应用程序、部署配置和实例。这些资源称作子资源。这些资源和子资源具有与之 ARNs 关联的独特性。有关更多信息，请参阅中的 Amazon 资源名称 (ARNs) Amazon Web Services 一般参考。

资源类型	ARN 格式
部署组	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :deploymentgroup:` *`application-name`*` /`*`deployment-group-name`*
应用程序	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :application:` *`application-name`*
部署配置	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :deploymentconfig:` *`deployment-configuration-name`*
实例	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :instance/` *`instance-ID`*
所有 CodeDeploy 资源	`arn:aws:codedeploy:*`

资源类型	ARN 格式
指定区域中指定账户拥有的所有 CodeDeploy 资源	`arn:aws:codedeploy:  region:account-id :*`

> **Note**
>
> 中的大多数服务都 Amazon 将冒号 (:) 或正斜杠 (/) 视为中的 ARNs相同字符。但是，在资源模式和规则中 CodeDeploy 使用精确匹配。请务必在创建事件模式时使用正确的 ARN 字符，以使其与资源中的 ARN 语法匹配。

条件键

CodeDeploy 不提供任何特定于服务的条件密钥，但它确实支持使用某些全局条件密钥。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

示例

要查看 CodeDeploy 基于身份的策略的示例，请参阅。Amazon CodeDeploy 基于身份的策略示例

CodeDeploy 基于资源的政策

CodeDeploy 不支持基于资源的策略。要查看详细的基于资源的策略页面的示例，请参阅使用基于资源的策略。 Amazon Lambda

基于 CodeDeploy 标签的授权

CodeDeploy 不支持标记资源或基于标签控制访问权限。

CodeDeploy IAM 角色

I AM 角色是您的 Amazon 账户中具有特定权限的实体。

将临时证书与 CodeDeploy

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用AssumeRole或之类的 Amazon STS API 操作来获取临时安全证书GetFederationToken。

CodeDeploy 支持使用临时证书。

服务关联角色

CodeDeploy 不支持服务相关角色。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 Amazon 账户中，并归该账户所有。这意味着用户可以更改此角色的权限。但是，这样做可能会中断服务的功能。

CodeDeploy 支持服务角色。

在中选择 IAM 角色 CodeDeploy

在中创建部署组资源时 CodeDeploy，必须选择一个角色 CodeDeploy 才能代表您访问 Amaz EC2 on。如果您之前已经创建了一个服务角色或服务相关角色，则 CodeDeploy 会为您提供一个角色列表供您选择。选择允许访问启动和停止 EC2 实例的角色非常重要。

Amazon 的托管（预定义）策略 CodeDeploy

Amazon 通过提供由创建和管理的独立 IAM 策略来解决许多常见用例 Amazon。这些 Amazon托管策略授予常见用例的权限，因此您可以不必调查需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略。

主题

- 的 Amazon 托管策略列表 CodeDeploy
- CodeDeploy 托管策略和通知

的 Amazon 托管策略列表 CodeDeploy

您可以将以下 Amazon 托管策略附加到账户中的用户，这些策略特定于 CodeDeploy：

- `AWSCodeDeployFullAccess`：授与对 CodeDeploy 的完全访问权限。

> **Note**
>
> AWSCodeDeployFullAccess 不为部署应用程序所需的其他服务（例如 Amazon EC2 和 Amazon S3）中的操作提供权限，仅向特定于的操作提供权限 CodeDeploy。

- `AWSCodeDeployDeployerAccess`：授予注册和部署修订版的权限。

- `AWSCodeDeployReadOnlyAccess`：授予对 CodeDeploy 的只读访问权限。

- `AWSCodeDeployRole`: 允 CodeDeploy 许：
  - 读取您的实例上的标签或通过 Amazon A EC2 uto Scaling 组名识别您的亚马逊 EC2 实例
  - 读取、创建、更新和删除 Amazon A EC2 uto Scaling 群组、生命周期挂钩、扩展策略和温池功能
  - 将信息发布到 Amazon SNS 主题
  - 检索有关亚马逊 CloudWatch 警报的信息
  - 读取和更新 ELB 服务中的资源

  策略包含以下代码：

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:EnableMetricsCollection",
        "autoscaling:DescribePolicies",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeNotificationConfigurations",
        "autoscaling:SuspendProcesses",
        "autoscaling:ResumeProcesses",
        "autoscaling:AttachLoadBalancers",
        "autoscaling:AttachLoadBalancerTargetGroups",
        "autoscaling:PutScalingPolicy",
```

```
        "autoscaling:PutScheduledUpdateGroupAction",
        "autoscaling:PutNotificationConfiguration",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:PutWarmPool",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "ec2:TerminateInstances",
        "tag:GetResources",
        "sns:Publish",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetGroupAttributes",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- `AWSCodeDeployRoleForLambda`：授予访问 CodeDeploy权限 Amazon Lambda 以及部署所需的任何其他资源。

- `AWSCodeDeployRoleForECS`：授 CodeDeploy 予访问部署所需的 Amazon ECS 和任何其他资源的权限。

- `AWSCodeDeployRoleForECSLimited`: 授予访问部署所需的 Amazon ECS 和任何其他资源的 CodeDeploy权限，但以下情况除外：
  - 在该 AppSpec 文件的`hooks`部分中，只能使用名称以开头的 Lambda 函数。`CodeDeployHook_`有关更多信息，请参阅 AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。

- S3 存储桶访问权限仅限于具有注册标记 `UseWithCodeDeploy` 且值为 `true` 的 S3 存储桶。有关更多信息，请参阅对象标记。

- `AmazonEC2RoleforAWSCodeDeployLimited`：授 CodeDeploy 予获取和列出 CodeDeploy Amazon S3 存储桶中对象的权限。策略包含以下代码：

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*/CodeDeploy/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            }
        }
    ]
}
```

部署过程某些方面的权限被授予代表以下两个角色的另外两个角色类型 CodeDeploy：

- IAM 实例配置文件是您附加到您的 Amazon EC2 实例的 IAM 角色。此配置文件包括访问存储应用程序的 Amazon S3 存储桶或存储 GitHub 库所需的权限。有关更多信息，请参阅 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。

- 服务角色是一个 IAM 角色，它向 Amazon 服务授予权限，使其可以访问 Amazon 资源。您附加到服务角色的策略决定了服务可以访问哪些 Amazon 资源以及它可以对这些资源执行的操作。对于 CodeDeploy，服务角色用于以下用途：
  - 读取应用于实例的标签或与实例关联的 Amazon A EC2 uto Scaling 组名称。这样就可以 CodeDeploy 确定它可以将应用程序部署到的实例。
  - 要对实例、Amazon A EC2 uto Scaling 组和 ELB 负载均衡器执行操作。
  - 向 Amazon SNS 主题发布信息，以便在发生指定部署或实例事件时发送通知。
  - 检索有关 CloudWatch 警报的信息，为部署设置警报监控。

  有关更多信息，请参阅 步骤 2：为创建服务角色 CodeDeploy。

您还可以创建自定义 IAM 策略来授予 CodeDeploy操作和资源的权限。您将这些自定义策略附加到 IAM 角色，然后将角色分配给需要权限的用户或组。

CodeDeploy 托管策略和通知

CodeDeploy 支持通知，让用户知道部署的重要更改。的托管策略 CodeDeploy 包括通知功能的策略声明。有关更多信息，请参阅什么是通知？。

完全访问托管式策略中通知的权限

`AWSCodeDeployFullAccess` 托管策略包含以下语句，以允许对通知进行完全访问。已应用此托管式策略的用户还可以创建和管理通知的 Amazon SNS 主题、为用户订阅和取消订阅主题，以及列出要选择作为通知规则的目标的主题。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codedeploy:*:*:application:*"}
```

```
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
      "Sid" : "CodeStarNotificationsChatbotAccess",
      "Effect" : "Allow",
      "Action" : [
        "chatbot:DescribeSlackChannelConfigurations"
      ],
      "Resource" : "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    }
```

只读托管式策略中通知的权限

`AWSCodeDeployReadOnlyAccess` 托管策略包含以下语句，以允许对通知进行只读访问。应用此托管策略的用户可以查看资源的通知，但无法创建、管理或订阅这些通知。

```
  {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codedeploy:*:*:application:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules"
        ],
        "Resource": "*"
    }
```

其它托管式策略中通知的权限

AWSCodeDeployDeployerAccess 托管式策略包括以下语句，可让用户创建、更新、订阅和查看资源的通知，但无法将其删除。应用了此托管式策略的用户还可以创建和管理用于通知的 Amazon SNS 主题。

此策略包括以下权限：

- codestar-notifications:CreateNotificationRule：支持主体创建通知。
- codestar-notifications:DescribeNotificationRule：支持主体检索有关通知的信息。
- codestar-notifications:UpdateNotificationRule：支持主体更新通知。
- codestar-notifications:Subscribe：支持主体订阅通知更新。
- codestar-notifications:Unsubscribe：支持主体取消订阅通知更新。
- codestar-notifications:ListNotificationRules：支持主体检索通知规则列表。
- codestar-notifications:ListTargets：支持主体检索目标列表。
- codestar-notifications:ListTagsforResource：支持主体检索标签列表。
- codestar-notifications:ListEventTypes：支持主体检索事件类型列表。

- `chatbot:DescribeSlackChannelConfiguration`：支持主体检索有关 Slack 通道配置的信息。
- `sns:ListTopics`：支持主体检索通知的 Amazon SNS 主题列表。

```
  {
     "Sid" : "CodeStarNotificationsReadWriteAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:CreateNotificationRule",
       "codestar-notifications:DescribeNotificationRule",
       "codestar-notifications:UpdateNotificationRule",
       "codestar-notifications:Subscribe",
       "codestar-notifications:Unsubscribe"
     ],
     "Resource" : "*",
     "Condition" : {
       "ArnLike" : {
         "codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"
       }
     }
   },
   {
     "Sid" : "CodeStarNotificationsListAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:ListNotificationRules",
       "codestar-notifications:ListTargets",
       "codestar-notifications:ListTagsforResource",
       "codestar-notifications:ListEventTypes"
     ],
     "Resource" : "*"
   },
   {
     "Sid" : "CodeStarNotificationsChatbotAccess",
     "Effect" : "Allow",
     "Action" : [
       "chatbot:DescribeSlackChannelConfigurations"
     ],
     "Resource" : "*"
   },
   {
```

```
      "Sid" : "SNSTopicListAccess",
      "Effect" : "Allow",
      "Action" : [
        "sns:ListTopics"
      ],
      "Resource" : "*"
    }
```

有关更多信息，请参阅 Amazon CodeStar 通知的身份和访问管理。

CodeDeploy Amazon 托管策略的更新

查看 CodeDeploy 自该服务开始跟踪这些更改以来 Amazon 托管策略更新的详细信息。要获得有关此页面更改的自动提示，请订阅 CodeDeploy 文档历史记录 上的 RSS 源。

更改	描述	日期
`AWSCodeDeployDeployerAccess` 托管策略 — 对现有策略的更新	更新了 `codestar-notifications:NotificationsForResource` 操作以支持 IAM 策略验证更改。原始资源 `arn:aws:codedeploy:*` 已更新为 `arn:aws:codedeploy:*:*:application:*` 。 有关此策略的更多信息，请参阅其它托管式策略中通知的权限。	2024 年 12 月 16 日
`AWSCodeDeployReadOnlyAccess` 托管策略 — 对现有策略的更新	更新了 `codestar-notifications:NotificationsForResource` 操作以支持 IAM 策略验证更改。原始资源 `arn:aws:codedeploy:*` 已更新为 `arn:aws:codedeploy:*:*:application:*` 。	2024 年 12 月 16 日

更改	描述	日期
	有关此策略的更多信息，请参阅只读托管式策略中通知的权限。	
AWSCodeDeployFullAccess 托管策略 — 对现有策略的更新	更新了 codestar-notifications:NotificationsForResource 操作以支持 IAM 策略验证更改。原始资源 arn:aws:codedeploy:* 已更新为 arn:aws:codedeploy:*:*:application:* 。 有关此策略的更多信息，请参阅完全访问托管式策略中通知的权限。	2024 年 12 月 16 日
AWSCodeDeployRole 托管策略 — 对现有策略的更新	在政策声明中添加了 elasticloadbalancing:DescribeLoadBalancerAttributes 和 elasticloadbalancing:DescribeTargetGroupAttributes 操作以支持 Elastic Load Balancing 更改。 有关此策略的更多信息，请参阅AWSCodeDeployRole。	2023 年 8 月 16 日

更改	描述	日期
AWSCodeDeployFullAccess 托管策略 — 对现有策略的更新	在政策声明中添加了 chatbot:ListMicrosoftTeamsChannelConfigurations 操作以支持通知更改。 有关此策略的更多信息，请参阅AWSCodeDeployRole。	2023 年 5 月 11 日
AWSCodeDeployRole 托管策略 — 对现有策略的更新	在政策声明中添加了支持 Amazon A EC2 uto Scaling 授权变更的 autoscaling:CreateOrUpdateTags 操作。 有关此策略的更多信息，请参阅AWSCodeDeployRole。	2023 年 2 月 3 日
AmazonEC2RoleforAWSCodeDeployLimited 托管策略 — 对现有策略的更新	从包含 s3:ExistingObjectTag/UseWithCodeDeploy 条件的政策声明中删除了 s3:ListBucket 操作。 有关此策略的更多信息，请参阅AmazonEC2RoleforAWSCodeDeployLimited。	2021 年 11 月 22 日
AWSCodeDeployRole 托管策略 — 对现有策略的更新	添加了支持向 Amazon A EC2 uto Scaling 群组添加暖池以进行蓝/绿部署的autoscaling:PutWarmPool 操作。 删除了不必要的重复操作。	2021 年 5 月 18 日
CodeDeploy 开始跟踪更改	CodeDeploy 开始跟踪其 Amazon 托管策略的更改。	2021 年 5 月 18 日

Amazon CodeDeploy 基于身份的策略示例

默认情况下，用户无权创建或修改 CodeDeploy资源。他们也无法使用 Amazon Web Services 管理控制台 Amazon CLI、或 Amazon API 执行任务。您必须创建 IAM policy，授予 IAM 角色在其所需的指定资源上执行 API 操作的权限。然后，您必须将这些 IAM 角色附加到需要这些权限的用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

在中 CodeDeploy，基于身份的策略用于管理与部署过程相关的各种资源的权限。您可以控制对所有以下类型资源的访问：

- 应用程序和应用程序修订。
- 部署。
- 部署配置。
- 实例和本地实例。

由资源策略所控制的能力因资源类型而异，如下表所述：

资源类型	功能
全部	查看和列出有关资源的详细信息
应用程序 部署配置 部署组	创建资源 删除资源
部署	创建部署 停止部署
应用程序修订	注册应用程序修订
应用程序 部署组	更新资源
本地实例	向实例中添加标签

资源类型	功能
	从实例中删除标签 注册实例 取消注册实例

下面的示例演示一个权限策略，该策略允许用户删除与 **us-west-2** 区域中名为 **WordPress_DepGroup** 的应用程序关联的名为 **WordPress_App** 的部署组。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    }
  ]
}
```

主题

- 客户管理型策略示例
- 策略最佳实践
- 使用 CodeDeploy 控制台
- 允许用户查看他们自己的权限

客户管理型策略示例

在本节中，您可以找到授予各种 CodeDeploy 操作权限的策略示例。这些政策在您使用 CodeDeploy API Amazon SDKs、或时起作用 Amazon CLI。您必须针对在控制台中执行的操作授予其他权限。要了解有关授予控制台权限的更多信息，请参阅使用 CodeDeploy 控制台。

> **Note**
>
> 所有示例都使用美国西部（俄勒冈）区域 (`us-west-2`)，并包含虚构账户。 IDs

示例

- 示例 1：允许在单个区域中执行 CodeDeploy 操作
- 示例 2：允许为单个应用程序注册修订
- 示例 3：允许为单个部署组创建部署

示例 1：允许在单个区域中执行 CodeDeploy 操作

以下示例仅授予**`us-west-2`**在该区域执行 CodeDeploy 操作的权限：

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:*"
      ]
    }
  ]
}
```

示例 2：允许为单个应用程序注册修订

以下示例授予为 **us-west-2** 区域中所有以 **Test** 开头的应用程序注册应用程序修订的权限：

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:Test*"
      ]
    }
  ]
}
```

示例 3：允许为单个部署组创建部署

以下示例允许为与名为 **WordPress_App** 的应用程序关联的名为 **WordPress_DepGroup** 的部署组、名为 **ThreeQuartersHealthy** 的自定义部署配置以及与名为 **WordPress_App** 的应用程序关联的任何应用程序修订创建部署。所有这些资源都位于 **us-west-2** 区域中。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
```

```
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-
west-2:111122223333:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:WordPress_App"
      ]
    }
  ]
}
```

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 CodeDeploy 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 中的安全最佳实操。

使用 CodeDeploy 控制台

如果您使用 CodeDeploy 控制台，则必须拥有一组允许您描述 Amazon 账户其他 Amazon 资源的最低权限。要 CodeDeploy 在 CodeDeploy 控制台中使用，您必须拥有以下服务的权限：

- Amazon A EC2 uto Scaling
- Amazon CodeDeploy
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- Amazon Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

如果创建的 IAM policy 比最低权限要求更严格，那么对于具有该 IAM policy 角色的用户，控制台将无法正常运行。为确保这些用户仍然可以使用 CodeDeploy控制台，还要将`AWSCodeDeployReadOnlyAccess`托管策略附加到分配给该用户的角色，如中所述Amazon 的托管（预定义）策略 CodeDeploy。

对于仅调用 Amazon CLI 或 CodeDeploy API 的用户，您无需为其设置最低控制台权限。

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": "*"
        }
    ]
}
```

排除 Amazon CodeDeploy 身份和访问问题

使用以下信息来帮助您诊断和修复在使用 CodeDeploy 和 IAM 时可能遇到的常见问题。

主题

- 我无权执行 iam：PassRole
- 我想允许 Amazon 账户之外的人访问我的 CodeDeploy资源

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。 CodeDeploy

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 CodeDeploy 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。

我想允许 Amazon 账户之外的人访问我的 CodeDeploy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：

- 要了解是否 CodeDeploy 支持这些功能，请参阅如何 Amazon CodeDeploy 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。

CodeDeploy 权限参考

在设置访问权限以及编写可附加到 IAM 身份的权限策略（基于身份的策略）时，请使用下表。该表列出了每个 CodeDeploy API 操作、您可以授予执行该操作的权限的操作，以及用于授予权限的资源 ARN 的格式。请在策略的 `Action` 字段中指定这些操作。您可以在策略的 `Resource` 字段中指定带或不带通配符（`*`）的 ARN 作为资源值。

您可以在 CodeDeploy 策略中使用 Amazon-wide 条件键来表达条件。有关 Amazon范围密钥的完整列表，请参阅 IAM 用户指南中的可用密钥。

要指定操作，请在 API 操作名称之前使用 `codedeploy:` 前缀（例如，`codedeploy:GetApplication` 和 `codedeploy:CreateApplication`）。要在单个语句中指定多项操作，请使用逗号将它们隔开（例如，`"Action": ["codedeploy:action1", "codedeploy:action2"]`）。

使用通配符

您可以在 ARN 使用通配符（`*`）以指定多个操作或资源。例如，`codedeploy:*`指定所有 CodeDeploy 动作并`codedeploy:Get*`指定以单词开头的所有 CodeDeploy 动作`Get`。以下示例授予对名称以 `West` 开头且与名称以 `Test` 开头的应用程序关联的所有部署组的访问权限。

```
arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:Test*/West*
```

您可以将通配符与表中列出的以下资源一起使用：

- *application-name*
- *deployment-group-name*
- *deployment-configuration-name*
- *instance-ID*

通配符不能与*region*或*account-id*一起使用。有关通配符的更多信息，请参阅 IAM 用户指南中的 IAM 标识符。

> **Note**
>
> 在每个操作的 ARN 中，资源后跟一个冒号（`:`）。您还可以让资源后跟正斜杠（`/`）。有关更多信息，请参阅CodeDeploy示例 ARNs。

CodeDeploy API 操作和操作所需的权限

AddTagsToOnPremisesInstances

操作：codedeploy:AddTagsToOnPremisesInstances

向一个或多个本地实例添加标签所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

BatchGetApplicationRevisions

操作：codedeploy:BatchGetApplicationRevisions

获取有关与 用户关联的多个应用程序版本的信息所必需的。

资源： arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

BatchGetApplications

操作：codedeploy:BatchGetApplications

获取有关与 用户关联的多个应用程序的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*

BatchGetDeploymentGroups

操作：codedeploy:BatchGetDeploymentGroups

获取有关与 用户关联的多个部署组的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeploymentInstances

操作：codedeploy:BatchGetDeploymentInstances

获取有关属于部署组的一个或多个实例的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeployments

操作：codedeploy:BatchGetDeployments

获取有关与 用户关联的多个部署的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetOnPremisesInstances

操作：codedeploy:BatchGetOnPremisesInstances

获取有关一个或多个本地实例的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:*

ContinueDeployment

操作：codedeploy:ContinueDeployment

在 blue/green 部署期间，需要在替代环境中使用 ELB 负载均衡器注册实例。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateApplication

操作：codedeploy:CreateApplication

创建与 用户关联的应用程序所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

CreateDeployment[1]

操作：codedeploy:CreateDeployment

为与 用户关联的应用程序创建部署所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateDeploymentConfig

操作：codedeploy:CreateDeploymentConfig

创建与 用户关联的自定义部署配置所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

CreateDeploymentGroup

操作：`codedeploy:CreateDeploymentGroup`

为与 用户关联的应用程序创建部署组所必需的。

资源：`arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

DeleteApplication

操作：`codedeploy:DeleteApplication`

删除与 用户关联的应用程序所必需的。

资源：`arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:application:`*`application-name`*

DeleteDeploymentConfig

操作：`codedeploy:DeleteDeploymentConfig`

删除与 用户关联的自定义部署配置所必需的。

资源：`arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentconfig/`*`deployment-configuration-name`*

DeleteDeploymentGroup

操作：`codedeploy:DeleteDeploymentGroup`

为与 用户关联的应用程序删除部署组所必需的。

资源：`arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

DeregisterOnPremisesInstance

操作：`codedeploy:DeregisterOnPremisesInstance`

取消注册本地实例所必需的。

资源： `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:instance/`*`instance-ID`*

GetApplication

操作：`codedeploy:GetApplication`

获取有关与 用户关联的单个应用程序的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetApplicationRevision

操作：codedeploy:GetApplicationRevision

获取有关与 用户关联的应用程序的单个应用程序修订的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetDeployment

操作：codedeploy:GetDeployment

获取针对与 用户关联的应用程序的部署组的单个部署的信息所必需的。

资源： arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentConfig

操作：codedeploy:GetDeploymentConfig

获取有关与 用户关联的单个部署配置的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

GetDeploymentGroup

操作：codedeploy:GetDeploymentGroup

获取有关与 用户关联的应用程序的单个部署组的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentInstance

操作：codedeploy:GetDeploymentInstance

获取有关部署中与 用户关联的单个实例的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetOnPremisesInstance

操作：codedeploy:GetOnPremisesInstance

获取有关单个本地实例的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

ListApplicationRevisions

操作：codedeploy:ListApplicationRevisions

获取有关与 用户关联的应用程序的所有应用程序修订的信息所必需的。

资源： arn:aws:codedeploy:*region*:*account-id*:application:*

ListApplications

操作：codedeploy:ListApplications

获取有关与 用户关联的所有应用程序的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*

ListDeploymentConfigs

操作：codedeploy:ListDeploymentConfigs

获取有关与 用户关联的所有部署配置的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*

ListDeploymentGroups

操作：codedeploy:ListDeploymentGroups

获取有关与 用户关联的应用程序的所有部署组的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*

ListDeploymentInstances

操作：codedeploy:ListDeploymentInstances

获取有关部署中与用户关联的所有实例的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListDeployments

操作：codedeploy:ListDeployments

获取有关针对与用户关联的部署组的所有部署的信息所必需的，或获取与用户的所有部署所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListGitHubAccountTokenNames

操作：codedeploy:ListGitHubAccountTokenNames

需要获取已存储的 GitHub账户连接的名称列表。

资源：arn:aws:codedeploy:*region*:*account-id*:*

ListOnPremisesInstances

操作：codedeploy:ListOnPremisesInstances

获取一个或更多本地实例名称的列表所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:*

RegisterApplicationRevision

操作：codedeploy:RegisterApplicationRevision

注册有关与 用户关联的应用程序的一个应用程序修订的信息所必需的。

资源：arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

RegisterOnPremisesInstance

操作：codedeploy:RegisterOnPremisesInstance

需要向注册本地实例 CodeDeploy。

资源：arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

RemoveTagsFromOnPremisesInstances

操作：`codedeploy:RemoveTagsFromOnPremisesInstances`

从一个或多个本地实例中删除标签所必需的。

资源：`arn:aws:codedeploy:region:account-id:instance/instance-ID`

SkipWaitTimeForInstanceTermination

操作：`codedeploy:SkipWaitTimeForInstanceTermination`

需要覆盖指定的等待时间，并在部署中成功路由流量后立即开始终止原始环境中的实例。 blue/green

资源：`arn:aws:codedeploy:region:account-id:instance/instance-ID`

StopDeployment

操作：`codedeploy:StopDeployment`

停止正在进行的部署到与 用户关联的应用程序的部署组所必需的。

资源：`arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

UpdateApplication[3]

操作：`codedeploy:UpdateApplication`

更改有关与 用户关联的应用程序的信息所必需的。

资源：`arn:aws:codedeploy:region:account-id:application:application-name`

UpdateDeploymentGroup[3]

操作：`codedeploy:UpdateDeploymentGroup`

更改有关与 用户关联的应用程序的单个部署组的信息所必需的。

资源：`arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

[1] 当您指定 `GetDeploymentConfig` 权限时，还必须为部署配置指定 `GetApplicationRevision` 权限，并且为应用程序修订指定 `CreateDeployment` 或 `RegisterApplicationRevision` 权限。

[2] 在提供某个部署组时对 `ListDeployments` 有效，但在列出所有与用户关联的部署时无效。

[3] 对于 `UpdateApplication`，您必须同时对旧应用程序名称和新应用程序名称具有 `UpdateApplication` 权限。对于涉及更改部署组名称的 `UpdateDeploymentGroup` 操作，您必须同时具有对旧的和新的部署组名称的 `UpdateDeploymentGroup` 权限。

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 Amazon，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，我们 Amazon 提供了一些工具，帮助您保护所有服务的数据，这些服务委托人已被授予对您账户中资源的访问权限。

我们建议在资源策略中使用 a ws: SourceArn 和 aws: SourceAccount 全局条件上下文密钥来限制为资源 CodeDeploy 提供其他服务的权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 `aws:SourceArn` 值，则 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想该账户中的任何资源与跨服务使用相关联，请使用 `aws:SourceAccount`。

对于 EC2 /本地、Lamb Amazon da 和常规 Amazon ECS 部署，`aws:SourceArn`的值应包括 CodeDeploy 允许其担任 IAM 角色 CodeDeploy 的部署组 ARN。

对于通过创建的 Amazon ECS 蓝/绿部署 Amazon CloudFormation，的值`aws:SourceArn`应包括允许担任 IAM 角色的 CodeDeploy 堆栈 CloudFormation ARN。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 键和资源的完整 ARN。如果不知道完整 ARN，或者正在指定多个资源，请针对未知部分使用通配符（*）。

例如，您可以在 EC2 /Limplesd、Lamb Amazon da 或常规 Amazon ECS 部署中使用以下信任策略：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codedeploy:us-
east-1:111122223333:deploymentgroup:myApplication/*"
                }
            }
        }
    ]
}
```

对于通过创建的 Amazon ECS 蓝/绿部署 Amazon CloudFormation，您可以使用：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyCloudFormationStackName/*"
                }
            }
        }
    ]
```

```
}
```

登录和监控 CodeDeploy

本部分概述了 CodeDeploy 中的监控、日志记录和事件响应。

审核与的所有互动 CodeDeploy

CodeDeploy 与一项服务集成 Amazon CloudTrail，该服务可捕获由您的账户或代表您的 Amazon 账户进行的 API 调用，并将日志文件传送到您指定的 S3 存储桶。 CodeDeploy CloudTrail 捕获来自 CodeDeploy 控制台、通过 CodeDeploy 命令或 CodeDeploy APIs 直接从控制台发出的 API 调用。Amazon CLI使用收集到的信息 CloudTrail，您可以确定向哪个请求发出 CodeDeploy、发出请求的源 IP 地址、谁发出了请求、何时发出请求等。要了解更多信息 CloudTrail，请参阅《Amazon CloudTrail 用户指南》中的使用 CloudTrail日志文件。

您可以通过将 Amazon CloudWatch 代理设置为在 CloudWatch 控制台中查看聚合数据或登录实例查看日志文件来查看 CodeDeploy 部署创建的日志数据。有关更多信息，请参阅 将 CodeDeploy 代理日志发送到 CloudWatch。

提醒和事件管理

您可以使用 Amazon CloudWatch Events 来检测 CodeDeploy 操作中实例或部署（事件）状态的变化并做出反应。然后，根据您创建的规则，当部署或实例进入您在规则中指定的状态时， CloudWatch 事件会调用一个或多个目标操作。根据状态更改的类型，您可能想发送通知，捕获状态信息，采取纠正措施，启动事件或采取其他操作。在 CodeDeploy 操作中使用 CloudWatch 事件时，可以选择以下类型的目标：

- Amazon Lambda 函数
- Kinesis Streams
- Amazon SQS 队列
- 内置目标（CloudWatch 警报动作）
- Amazon SNS 主题

下面是一些用例：

- 每当部署失败时使用 Lambda 函数向 Slack 通道传送通知。

- 将有关部署或实例的数据推送到 Kinesis 流，以支持全面、实时的状态监控。
- 当您指定的部署或 EC2 实例事件发生时，使用 CloudWatch 警报操作自动停止、终止、重启或恢复实例。

有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的什么是亚马逊 CloudWatch 活动。

合规性验证 Amazon CodeDeploy

要了解是否属于特定合规计划的范围，请参阅Amazon Web Services 服务 “” Amazon Web Services 服务 中的 “按合规计划划分的范围”，然后选择您感兴趣的合规计划。 Amazon Web Services 服务 有关一般信息，请参阅合规计划。

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。

您在使用 Amazon Web Services 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 Amazon Web Services 服务，请参阅Amazon 安全文档。

韧性在 Amazon CodeDeploy

Amazon 全球基础设施是围绕 Amazon 区域和可用区构建的。 Amazon 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。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失效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设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅Amazon 全球基础设施。

中的基础设施安全 Amazon CodeDeploy

作为一项托管服务， Amazon CodeDeploy 受到《Amazon Web Services：安全流程概述》白皮书中描述的 Amazon 全球网络安全程序的保护。

您可以使用 Amazon 已发布的 API 调用 CodeDeploy 通过网络进行访问。客户端必须支持传输层安全性（TLS）1.2 或更高版本。我们建议使用 TLS 1.3 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

必须使用访问密钥 ID 以及与 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。

参考

参考。

主题

- CodeDeploy AppSpec 文件参考
- CodeDeploy 代理配置参考
- Amazon CloudFormation 模板供 CodeDeploy参考
- 将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用
- CodeDeploy 资源包参考
- CodeDeploy 配额

CodeDeploy AppSpec 文件参考

本部分仅供参考。有关该 AppSpec 文件的概念性概述，请参见Application Specification Files。

应用程序规范文件（AppSpec 文件）是 YAML 格式或 JSON 格式的文件，用于管理部署 CodeDeploy 。

> **Note**
>
> 除非您正在执行 EC2本地部署`appspec.yml`，否则必须命名 /Unlide 部署的 AppSpec 文件。有关更多信息，请参阅 创建本地部署。

主题

- AppSpec 亚马逊 ECS 计算平台上的文件
- AppSpec Amazon Lambda 计算平台上的文件
- AppSpec EC2/本地计算平台上的文件
- AppSpec 文件结构
- AppSpec 文件示例
- AppSpec 文件间距
- 验证您的 AppSpec 文件和文件位置

AppSpec 亚马逊 ECS 计算平台上的文件

对于 Amazon ECS 计算平台应用程序，该 AppSpec 文件 CodeDeploy 用于确定：

- Amazon ECS 任务定义文件。这是在文件指`TaskDefinition`令中使用其 ARN 指定的。 AppSpec
- 替换任务集中的容器和端口，应用程序负载均衡器或网络负载均衡器在部署期间会在其中重新路由流量。这是通过 AppSpec 文件中的`LoadBalancerInfo`指令指定的。
- 有关 Amazon ECS 服务的可选信息，例如运行它的平台版本、它的子网及其安全组。
- 可选的 Lambda 函数，在与 Amazon ECS 部署期间的生命周期事件对应的挂钩期间运行。有关更多信息，请参阅 AppSpec 亚马逊 ECS 部署的 “挂钩” 部分。

AppSpec Amazon Lambda 计算平台上的文件

对于 Amazon Lambda 计算平台应用程序，该 AppSpec 文件 CodeDeploy 用于确定：

- 要部署的 Lambda 函数版本。
- 要用作验证测试的 Lambda 函数。

AppSpec 文件可以是 YAML 格式或 JSON 格式。创建部署时，您也可以将 AppSpec 文件内容直接输入 CodeDeploy 控制台。

AppSpec EC2/本地计算平台上的文件

如果您的应用程序使用 EC2 /Londest 计算平台，则该 AppSpec 文件必须是名为 YAML 格式的文件，`appspec.yml`并且必须位于应用程序源代码目录结构的根目录中。否则，部署会失败。它 CodeDeploy 用于确定：

- 它应该从 Amazon S3 中的应用程序修订版中安装到您的实例上的内容，或者 GitHub。
- 为响应部署生命周期事件而要运行的生命周期事件挂钩。

完成 AppSpec 文件后，将其与要部署的内容一起捆绑到存档文件（zip、tar 或压缩的 tar）中。有关更多信息，请参阅 正在处理的应用程序修订版 CodeDeploy。

> **Note**
>
> Windows Server 实例不支持 tar 和压缩的 tar 存档文件格式（.tar 和.tar.gz）。

获得捆绑的存档文件（称为修订版）后，将其上传到 Amazon S3 存储桶或 Git 存储库。 CodeDeploy 然后使用 CodeDeploy 来部署修订版。有关说明，请参阅使用创建部署 CodeDeploy。

EC2/本地计算平台部署的 appspec.yml 保存在修订版的根目录中。有关更多信息，请参阅为 EC2/本地部署添加 AppSpec 文件和为 CodeDeploy 规划修订。

AppSpec 文件结构

以下是用于部署到 Amazon Lambda 和 EC2 /本地计算平台的 AppSpec 文件的高级结构。

除非另有说明，否则 YAML 格式 AppSpec 文件中的字符串值不得用引号 (“”) 括起来。

AppSpec Amazon ECS 部署的文件结构

> **Note**
>
> 这个 AppSpec 文件是用 YAML 编写的，但你可以使用相同的结构用 JSON 写一个。JSON 格式 AppSpec 文件中的字符串总是用引号 (“”) 括起来。

```
version: 0.0
resources:
  ecs-service-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

版本

本节指定了 AppSpec 文件的版本。请勿更改此值。这是必需的。当前，允许的唯一值为 **0.0**。它由保留 CodeDeploy 以备将来使用。

使用字符串指定 version。

resources

此部分指定有关要部署的 Amazon ECS 应用程序的信息。

有关更多信息，请参阅 AppSpec Amazon ECS 部署的 “资源” 部分。

hooks

此部分指定用于运行特定部署生命周期事件挂钩以验证部署的 Lambda 函数。

有关更多信息，请参阅 用于 Amazon ECS 部署的生命周期事件挂钩的列表。

AppSpec Amazon Lambda 部署的文件结构

> **Note**
>
> 此 AppSpec 文件是用 YAML 编写的，但您可以使用相同的结构为 JSON 的 Lambda 部署写入 AppSpec 文件。JSON 格式 AppSpec 文件中的字符串总是用引号 (“”) 括起来。

```
version: 0.0
resources:
  lambda-function-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

版本

本节指定了 AppSpec 文件的版本。请勿更改此值。这是必需的。当前，允许的唯一值为 **0.0**。它由保留 CodeDeploy 以备将来使用。

使用字符串指定 version。

resources

此部分指定有关要部署的 Lambda 函数的信息。

有关更多信息，请参阅 AppSpec “资源” 部分（仅限 Amazon ECS 和 Amazon Lambda 部署）。

hooks

此部分指定用于运行特定部署生命周期事件以验证部署的 Lambda 函数。

有关更多信息，请参阅 AppSpec “挂钩” 部分。

AppSpec EC2/本地部署的文件结构

```
version: 0.0
os: operating-system-name
files:
```

```
  source-destination-files-mappings
permissions:
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

版本

本节指定了 AppSpec 文件的版本。请勿更改此值。这是必需的。当前，允许的唯一值为 **0.0**。它由保留 CodeDeploy 以备将来使用。

使用字符串指定 version。

os

本部分指定您部署到的实例的操作系统值。这是必需的。可以指定以下值：

- linux - 实例是 Amazon Linux, Ubuntu Server 或 RHEL 实例。
- windows - 实例是 Windows Server 实例。

使用字符串指定 os。

files

此部分指定应在部署的 Install 事件期间复制到实例的文件的名称。

有关更多信息，请参阅 AppSpec “文件” 部分（仅EC2限本地部署）。

permissions

此部分指定在将 `files` 部分中的文件复制到实例时，应如何向这些文件应用特殊权限（如果有）。本节仅适用于 Amazon Linux、Ubuntu Server 和 Red Hat Enterprise Linux（RHEL）实例。

有关更多信息，请参阅AppSpec “权限” 部分（仅EC2限本地部署）。

hooks

此部分指定在部署期间的特定部署生命周期事件处运行的脚本。

有关更多信息，请参阅 AppSpec “挂钩” 部分。

主题

- AppSpec “文件” 部分（仅EC2限本地部署）
- AppSpec “资源” 部分（仅限 Amazon ECS 和 Amazon Lambda 部署）
- AppSpec “权限” 部分（仅EC2限本地部署）
- AppSpec “挂钩” 部分

AppSpec “文件” 部分（仅EC2限本地部署）

提供 CodeDeploy 有关在部署的 In stall 事件期间应在实例上安装应用程序修订版中的哪些文件的信息。仅当您要在部署期间将修订中的文件复制到实例上的位置时，才需要此部分。

此部分具有以下结构：

```
files:
  - source: source-file-location-1
    destination: destination-file-location-1
file_exists_behavior: DISALLOW|OVERWRITE|RETAIN
```

可以设置多个 `destination` 和 `source` 对。

`source` 指令标识要从修订复制到实例的文件或目录：

- 如果 `source` 指的是文件，则仅将指定的文件复制到实例。
- 如果 `source` 指的是目录，则将该目录中的所有文件都复制到实例。
- 如果 `source` 是一个单斜杠（对于 Amazon Linux、RHEL 和 Ubuntu Server 实例，为“/”；对于 Windows Server 实例，为“\”），则将修订中的所有文件都复制到实例。

`source` 中使用的路径是相对于 `appspec.yml` 文件的路径，该文件应位于修订的根目录。有关修订文件结构的详细信息，请参阅为 CodeDeploy 规划修订。

`destination` 指令标识应将文件复制到的实例上的位置。这必须是完全限定的路径，例如 `/root/destination/directory`（在 Linux、RHEL 和 Ubuntu 上）或 `c:\destination\folder`（在 Windows 上）。

`source` 和 `destination` 分别使用字符串指定。

该`file_exists_behavior`指令是可选的，它指定如何 CodeDeploy处理已存在于部署目标位置但不属于先前成功部署的文件。此设置可以是以下值之一：

- DISALLOW：部署失败。这也是未指定选项时的默认行为。
- OVERWRITE：当前正在部署的应用程序修订的文件版本将替换实例上已有的版本。
- RETAIN：保留实例上已有的文件版本，并将其用作新部署的一部分。

使用 `file_exists_behavior` 设置时，请了解此设置：

- 只能指定一次，并且适用于 `files:` 下列出的所有文件和目录。
- 优先于`--file-exists-behavior` Amazon CLI 选项和 `fileExistsBehavior` API 选项（两者也是可选的）。

以下是 Amazon Linux、Ubuntu Server 或 RHEL 实例的示例 `files` 部分。

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
```

在此示例中，将在 Install 事件期间执行下面两个操作：

1. 将修订中的 `Config/config.txt` 文件复制到实例上的 `/webapps/Config/config.txt` 路径中。
2. 以递归方式将修订的 `source` 目录中的所有文件复制到实例上的 `/webapps/myApp` 目录中。

“files”部分示例

以下示例显示了如何指定 `files` 部分。尽管这些示例描述的是 Windows Server 文件和目录（文件夹）结构，但是它们可轻松适合于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

> **Note**
>
> 只有 EC2 /本地部署使用该`files`部分。它不适用于 Amazon Lambda 部署。

对于以下示例，我们假设这些文件出现在 `source` 的根的捆绑包中：

- `appspec.yml`

- my-file.txt
- my-file-2.txt
- my-file-3.txt

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
  - source: .\my-file.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#
# ---------------------
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
  - source: my-file-2.txt
    destination: c:\temp
  - source: my-file-3.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the appspec.yml
 file) to the destination folder c:\temp.
#
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
```

对于以下示例，我们假设 appspec.yml 与包含三个文件的名为 my-folder 的文件夹一起出现在 source 的根的捆绑包中：

- appspec.yml
- my-folder\my-file.txt
- my-folder\my-file-2.txt
- my-folder\my-file-3.txt

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the
 destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
# 5) Copy my-folder and its 3 files to my-folder within the destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder c:
\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\other-folder
```

```
#
# Result:
#   c:\temp\other-folder\my-file.txt
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\my-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\other-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the
 destination folder c:\temp. If any of the files already exist on the instance,
 overwrite them.
#
files:
  - source: \
```

```
    destination: c:\temp
file_exists_behavior: OVERWRITE
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
```

AppSpec “资源” 部分（仅限 Amazon ECS 和 Amazon Lambda 部署）

AppSpec 文件`'resources'`部分的内容因部署的计算平台而异。Amazon ECS 部署的`'resources'` 部分包含 Amazon ECS 任务定义、用于将流量路由到更新的 Amazon ECS 任务集的容器和端口以及其他可选信息。 Amazon Lambda 部署`'resources'`部分包含 Lambda 函数的名称、别名、当前版本和目标版本。

主题

- AppSpec Amazon Lambda 部署的 “资源” 部分
- AppSpec Amazon ECS 部署的 “资源” 部分

AppSpec Amazon Lambda 部署的 “资源” 部分

`'resources'` 部分指定要部署的 Lambda 函数，并具有以下结构：

YAML：

```
resources:
  - name-of-function-to-deploy:
      type: "AWS::Lambda::Function"
      properties:
        name: name-of-lambda-function-to-deploy
        alias: alias-of-lambda-function-to-deploy
        currentversion: version-of-the-lambda-function-traffic-currently-points-to
        targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

JSON:

```
"resources": [
    {
        "name-of-function-to-deploy" {
```

```
            "type": "AWS::Lambda::Function",
            "properties": {
                "name": "name-of-lambda-function-to-deploy",
                "alias": "alias-of-lambda-function-to-deploy",
                "currentversion": "version-of-the-lambda-function-traffic-currently-
points-to",
                "targetversion": "version-of-the-lambda-function-to-shift-traffic-to"
            }
        }
    }
]
```

每个属性均使用字符串指定。

- `name` – 必需。这是要部署的 Lambda 函数的名称。
- `alias` – 必需。这是 Lambda 函数的别名。
- `currentversion` – 必需。这是流量当前定向到的 Lambda 函数版本。此值必须为有效的正整数。
- `targetversion` – 必需。这是流量要转移到的 Lambda 函数版本。此值必须为有效的正整数。

AppSpec Amazon ECS 部署的 “资源” 部分

`'resources'` 部分指定要部署的 Amazon ECS 服务，并具有以下结构：

YAML：

```
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-arn"
        LoadBalancerInfo:
          ContainerName: "ecs-container-name"
          ContainerPort: "ecs-application-port"
# Optional properties
        PlatformVersion: "ecs-service-platform-version"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["ecs-subnet-1","ecs-subnet-n"]
            SecurityGroups: ["ecs-security-group-1","ecs-security-group-n"]
            AssignPublicIp: "ENABLED | DISABLED"
        CapacityProviderStrategy:
```

```
          - Base: integer
            CapacityProvider: "capacityProviderA"
            Weight: integer
          - Base: integer
            CapacityProvider: "capacityProviderB"
            Weight: integer
```

JSON:

```
"Resources": [
    {
        "TargetService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "TaskDefinition": "task-definition-arn",
                "LoadBalancerInfo": {
                    "ContainerName": "ecs-container-name",
                    "ContainerPort": "ecs-application-port"
                },
                "PlatformVersion": "ecs-service-platform-version",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "Subnets": [
                            "ecs-subnet-1",
                            "ecs-subnet-n"
                        ],
                        "SecurityGroups": [
                            "ecs-security-group-1",
                            "ecs-security-group-n"
                        ],
                        "AssignPublicIp": "ENABLED | DISABLED"
                    }
                },
                "CapacityProviderStrategy": [
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderA",
                        "Weight": integer
                    },
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderB",
                        "Weight": integer
```

```
                    }
                ]
            }
        }
    }
]
```

每个属性都使用字符串指定，但 `ContainerPort` 除外，它是一个数字。

- `TaskDefinition` – 必需。这是要部署的 Amazon ECS 服务的任务定义。它是使用任务定义的 ARN 指定的。ARN 格式为 `arn:aws:ecs:`*`aws-region`*`:`*`account-id`*`:task-definition/`*`task-definition-family`*`:`*`task-definition-revision`*。有关更多信息，请参阅 Amazon 资源名称 (ARNs) 和 Amazon 服务命名空间。

  > **Note**
  >
  > ARN 的 `:`*`task-definition-revision`* 部分是可选的。如果省略该部分，则 Amazon ECS 将使用任务定义的最新 ACTIVE 修订。

- `ContainerName` – 必需。这是包含 Amazon ECS 应用程序的 Amazon ECS 容器名称。它必须是在 Amazon ECS 任务定义中指定的容器。
- `ContainerPort` – 必需。这是流量将被路由到的容器上的端口。
- `PlatformVersion`: 可选。已部署的 Amazon ECS 服务中 Fargate 任务的平台版本。有关平台版本的更多信息，请参阅 Amazon Fargate 平台版本。如果没有指定任何版本，将默认使用 `LATEST`。
- `NetworkConfiguration`: 可选。在 `AwsvpcConfiguration` 下，您可以指定以下设置。有关更多信息，请参阅 Amazon ECS 容器服务 API 参考AwsVpcConfiguration中的。
  - `Subnets`: 可选。Amazon ECS 服务中一个或多个子网的逗号分隔列表。
  - `SecurityGroups`: 可选。Amazon Elastic Container Service 中一个或多个安全组的逗号分隔列表。
  - `AssignPublicIp`: 可选。一个字符串，它指定 Amazon ECS 服务的弹性网络接口是否接收公有 IP 地址。有效值为 `ENABLED` 和 `DISABLED`。

  > **Note**
  >
  > 必须指定 `NetworkConfiguration` 下的所有设置或不指定任何设置。例如，如果您要指定 `Subnets`，那么还必须指定 `SecurityGroups` 和 `AssignPublicIp`。如果未指定，则 CodeDeploy 使用当前网络 Amazon ECS 设置。

- `CapacityProviderStrategy`: 可选。您要用于部署的 Amazon ECS 容量提供程序的列表。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 容量提供程序。对于每个容量提供程序，您可以指定以下设置。有关这些设置的详细信息，请参阅《Amazon CloudFormation 用户指南》AWS::ECS::ServiceCapacityProviderStrategyItem中的
  - `Base`: 可选。基准值指明在指定的容量提供程序上至少运行多少个任务。在一个容量提供程序策略中，只能有一个容量提供程序策略定义了基准。如果未指定值，则使用默认值 0。
  - `CapacityProvider`: 可选。容量提供程序的简称。示例：capacityProviderA
  - `Weight`: 可选。

    权重 值指明应使用指定容量提供程序的已启动任务总数的相对百分比。在满足了 `base` 值（如果定义）之后，将考虑 `weight` 值。

    如果未指定 `weight` 值，则使用默认值 `0`。如果在容量提供程序策略中指定了多个容量提供程序，则至少有一个容量提供程序的权重值必须大于零，且任何权重为 `0` 的容量提供程序都不会被用来放置任务。如果您在策略中指定的多个容量提供程序的权重全部为 `0`，则使用该容量提供程序策略的任何 `RunTask` 或 `CreateService` 操作都将失败。

    下面是使用权重的示例情景：定义的策略包含两个容量提供程序，并且两个容量提供程序的权重均为 `1`，那么当满足 `base` 时，这些任务将在两个容量提供程序之间均匀分配。按照相同的逻辑，如果您为 capacityProviderA 指定权重 `1`，并为 capacityProviderB 指定权重 `4`，那么运行的每一个任务均使用 capacityProviderA，四个任务将使用 capacityProviderB。

AppSpec “权限” 部分（仅EC2限本地部署）

该`'permissions'`部分指定了应如何将特殊权限（如果有）应用于文件以及 directories/folders 将文件复制到实例后的`'files'`部分。您可以指定多个 `object` 指令。此部分是可选的。它仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

> **Note**
>
> 该`'permissions'`部分仅用于 EC2 /本地部署。它不用于 Amazon Lambda 或 Amazon ECS 部署。

此部分具有以下结构：

```
permissions:
  - object: object-specification
```

```
    pattern: pattern-specification
    except: exception-specification
    owner: owner-account-name
    group: group-name
    mode: mode-specification
    acls:
      - acls-specification
    context:
      user: user-specification
      type: type-specification
      range: range-specification
    type:
      - object-type
```

这些指令如下所示：

- object – 必需。这是一组文件系统对象（文件或目录/文件夹），这些文件系统对象复制到实例之后，将向其应用指定的权限。

  使用字符串指定 object。

- pattern：可选。指定权限应用模式。如果未指定或使用特殊字符 **"**"** 指定，则权限将应用于所有匹配的文件或目录，具体取决于 type。

  使用带引号（""）的字符串指定 pattern。

- except：可选。指定对于 pattern 而言例外的所有文件或目录。

  使用包含在方括号内的一组逗号分隔的字符串指定 except。

- owner：可选。object 所有者的名称。如果未指定，则在执行复制操作之后，应用于原始文件或目录/文件夹结构的所有现有所有者将保持不变。

  使用字符串指定 owner。

- group：可选。object 组的名称。如果未指定，则在执行复制操作之后，应用于原始文件或目录/文件夹结构的所有现有组将保持不变。

  使用字符串指定 group。

- mode：可选。一个数值，用于指定要应用于 object 的权限。模式设置遵循 Linux chmod 命令语法。

> **Important**
>
> 如果该值包含前导零，则必须用双引号将其括起来，或者删除前导零，以便只保留三位数字。

> **Note**
>
> `mode` 设置不支持诸如 **`u+x`** 之类的符号。

示例：

- `mode: "0644"` 向对象的所有者授予读写权限（6），为群组授予只读权限（4），向所有其他用户授予只读权限（4）。
- `mode: 644` 授予与 `mode: "0644"` 相同的权限。
- `mode: 4755` 设置 setuid 属性（4），向所有者授予完全控制权限（7），向群组授予读取和执行权限（5），并向所有其他用户授予读取和执行权限（5）。

  （有关更多示例，请参阅 Linux chmod 命令文档。）

  如果未指定 mode，则在执行复制操作之后，应用于原始文件或文件夹结构的所有现有模式将保持不变。

- `acls`：可选。一个字符串列表，表示应用于 `object` 的一个或多个访问控制列表（ACL）条目。例如，**`u:bob:rw`** 代表用户 **bob** 的读写权限。（有关更多示例，请参阅 Linux `setfacl` 命令文档中的 ACL 条目格式示例。）您可以指定多个 ACL 条目。如果未指定，`acls`则在执行复制操作后，ACLs 应用于原始文件或 directory/folder 结构的任何现有文件或结构将保持不变。它们取代了任何现有 ACLs的。

  使用短划线（`-`）后跟空格和字符串的形式指定 `acls`（例如，`- u:jane:rw`）。如果您有多个 ACL，每个 ACL 在单独的行中指定。

> **Note**
>
> 设置未命名的用户、未命名的群组或其他类似的 ACL 条目会导致 AppSpec 文件失败。可改用 `mode` 指定这些类型的权限。

- `context`：可选。对于启用了安全增强型 Linux (SELinux) 的实例，将列出要应用于复制对象的安全相关上下文标签。标签被指定为包含 `user`、`type` 和 `range` 的关键字。（有关更多信息，请参阅 SELinux 文档。）每个关键字使用一个字符串输入。如果未指定，则在执行复制操作后，应用于原始文件或 directory/folder 结构的所有现有标签将保持不变。
  - `user`：可选。 SELinux 用户。
  - `type`：可选。 SELinux 类型名称。
  - `range`：可选。 SELinux 范围说明符。这在计算机上启用 Multi-Level Security（MLS）和 Multi-Category Security（MCS）之后才生效。如果未启用，则 `range` 默认为 **`s0`**。

  使用字符串指定 `context`（例如， `user: unconfined_u`）。每个 `context` 在单独的行中指定。
- `type`：可选。将指定权限应用到的对象类型。`type` 是可设置为 **`file`** 或 **`directory`** 的字符串。如果指定了 **`file`**，则在执行复制操作之后，权限将仅应用于 `object` 中直接包含的文件（不应用于 `object` 自身）。如果**`directory`**指定，则权限将递归地应用于复制操作`object`后任何位置的所有 directories/folders 内容（但不适用于其`object`本身）。

  使用短划线（`-`）后跟空格和字符串的形式指定 `type`（例如，`- file`）。

“Permissions”部分示例

以下示例显示了如何使用 `object`、`pattern`、`except`、`owner`、`mode` 和 `type` 指令来指定 `'permissions'` 部分。此示例仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。在此示例中，假设以下文件和文件夹按此层次结构复制到实例：

```
/tmp
  `-- my-app
       |-- my-file-1.txt
       |-- my-file-2.txt
       |-- my-file-3.txt
       |-- my-folder-1
       |     |-- my-file-4.txt
       |     |-- my-file-5.txt
       |     `-- my-file-6.txt
       `-- my-folder-2
             |-- my-file-7.txt
             |-- my-file-8.txt
             |-- my-file-9.txt
            `-- my-folder-3
```

以下 AppSpec 文件显示了如何在复制这些文件和文件夹后对其设置权限：

```
version: 0.0
os: linux
# Copy over all of the folders and files with the permissions they
#  were originally assigned.
files:
  - source: ./my-file-1.txt
    destination: /tmp/my-app
  - source: ./my-file-2.txt
    destination: /tmp/my-app
  - source: ./my-file-3.txt
    destination: /tmp/my-app
  - source: ./my-folder-1
    destination: /tmp/my-app/my-folder-1
  - source: ./my-folder-2
    destination: /tmp/my-app/my-folder-2
# 1) For all of the files in the /tmp/my-app folder ending in -3.txt
#  (for example, just my-file-3.txt), owner = adm, group = wheel, and
#  mode = 464 (-r--rw-r--).
permissions:
  - object: /tmp/my-app
    pattern: "*-3.txt"
    owner: adm
    group: wheel
    mode: 464
    type:
      - file
# 2) For all of the files ending in .txt in the /tmp/my-app
#  folder, but not for the file my-file-3.txt (for example,
#  just my-file-1.txt and my-file-2.txt),
#  owner = ec2-user and mode = 444 (-r--r--r--).
  - object: /tmp/my-app
    pattern: "*.txt"
    except: [my-file-3.txt]
    owner: ec2-user
    mode: 444
    type:
      - file
# 3) For all the files in the /tmp/my-app/my-folder-1 folder except
#  for my-file-4.txt and my-file-5.txt, (for example,
#  just my-file-6.txt), owner = operator and mode = 646 (-rw-r--rw-).
  - object: /tmp/my-app/my-folder-1
    pattern: "**"
```

```
    except: [my-file-4.txt, my-file-5.txt]
    owner: operator
    mode: 646
    type:
      - file
# 4) For all of the files that are immediately under
#  the /tmp/my-app/my-folder-2 folder except for my-file-8.txt,
#  (for example, just my-file-7.txt and
#  my-file-9.txt), owner = ec2-user and mode = 777 (-rwxrwxrwx).
  - object: /tmp/my-app/my-folder-2
    pattern: "**"
    except: [my-file-8.txt]
    owner: ec2-user
    mode: 777
    type:
      - file
# 5) For all folders at any level under /tmp/my-app that contain
#  the name my-folder but not
#  /tmp/my-app/my-folder-2/my-folder-3 (for example, just
#  /tmp/my-app/my-folder-1 and /tmp/my-app/my-folder-2),
#  owner = ec2-user and mode = 555 (dr-xr-xr-x).
  - object: /tmp/my-app
    pattern: "*my-folder*"
    except: [tmp/my-app/my-folder-2/my-folder-3]
    owner: ec2-user
    mode: 555
    type:
      - directory
# 6) For the folder /tmp/my-app/my-folder-2/my-folder-3,
#  group = wheel and mode = 564 (dr-xrw-r--).
  - object: /tmp/my-app/my-folder-2/my-folder-3
    group: wheel
    mode: 564
    type:
      - directory
```

生成的权限如下所示：

```
-r--r--r-- ec2-user root  my-file-1.txt
-r--r--r-- ec2-user root  my-file-2.txt
-r--rw-r-- adm      wheel my-file-3.txt

dr-xr-xr-x ec2-user root  my-folder-1
```

```
-rw-r--r-- root     root  my-file-4.txt
-rw-r--r-- root     root  my-file-5.txt
-rw-r--rw- operator root  my-file-6.txt

dr-xr-xr-x ec2-user root  my-folder-2
-rwxrwxrwx ec2-user root  my-file-7.txt
-rw-r--r-- root     root  my-file-8.txt
-rwxrwxrwx ec2-user root  my-file-9.txt

dr-xrw-r-- root     wheel my-folder-3
```

以下示例显示了如何通过添加 `acls` 和 `context` 指令来指定 `'permissions'` 部分。此示例仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

```
permissions:
  - object: /var/www/html/WordPress
    pattern: "**"
    except: [/var/www/html/WordPress/ReadMe.txt]
    owner: bob
    group: writers
    mode: 644
    acls:
      - u:mary:rw
      - u:sam:rw
      - m::rw
    context:
      user: unconfined_u
      type: httpd_sys_content_t
      range: s0
    type:
      - file
```

AppSpec “挂钩” 部分

AppSpec 文件 `'hooks'` 部分的内容因部署的计算平台而异。 EC2/本地部署 `'hooks'` 部分包含将部署生命周期事件挂钩链接到一个或多个脚本的映射。Lambda 或 Amazon ECS 部署的 `'hooks'` 部分指定在部署生命周期事件期间运行的 Lambda 验证函数。如果某个事件的挂钩不存在，则不会对该事件执行任何操作。仅当您将在部署过程中运行脚本或 Lambda 验证函数时，才需要此部分。

主题

- AppSpec 亚马逊 ECS 部署的 “挂钩” 部分

- AppSpec Amazon Lambda 部署的 “挂钩” 部分
- AppSpec EC2/本地部署的 “挂钩” 部分

AppSpec 亚马逊 ECS 部署的 “挂钩” 部分

主题

- 用于 Amazon ECS 部署的生命周期事件挂钩的列表
- 钩子在 Amazon ECS 部署中的运行顺序。
- “hooks”部分的结构
- Lambda“hooks”函数示例

用于 Amazon ECS 部署的生命周期事件挂钩的列表

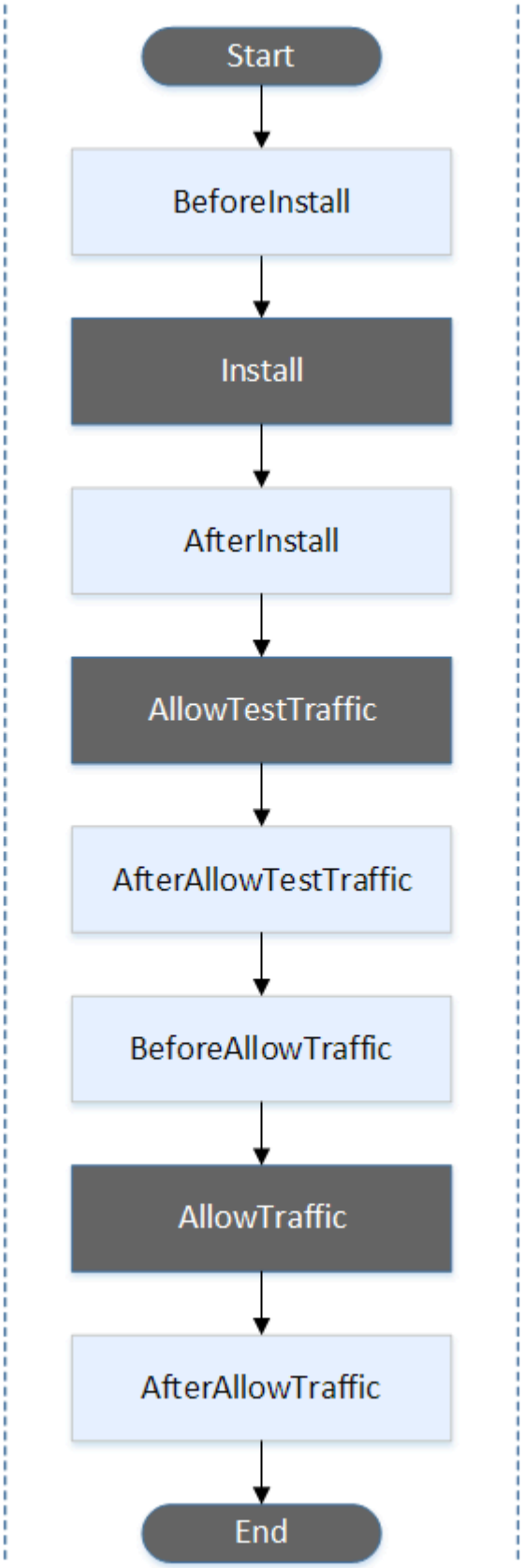
Amazon Lambda 挂钩是一个 Lambda 函数，在生命周期事件名称后的新行中使用字符串指定。对于每次部署，每个挂钩将执行一次。以下是在 Amazon ECS 部署期间可以运行挂钩的生命周期事件的描述。

- `BeforeInstall` – 用于在创建替换任务集之前运行任务。一个目标组与原始任务集相关联。如果指定了可选的测试侦听器，则它与原始任务集相关联。此时，无法执行回滚。
- `AfterInstall` – 用于在创建替换任务集并且其中一个目标组与之关联后运行任务。如果指定了可选的测试侦听器，则它与原始任务集相关联。在此生命周期事件时挂钩函数的运行结果可能会触发回滚。
- `AfterAllowTestTraffic` – 用于在测试侦听器为替换任务集提供流量后运行任务。此时挂钩函数的运行结果可能会触发回滚。
- `BeforeAllowTraffic` – 用于在第二个目标组与替换任务集关联之后但在流量转移到替换任务集之前运行任务。在此生命周期事件时挂钩函数的运行结果可能会触发回滚。
- `AfterAllowTraffic` – 用于在第二个目标组为替换任务集提供流量后运行任务。在此生命周期事件时挂钩函数的运行结果可能会触发回滚。

有关更多信息，请参阅在 Amazon ECS 部署过程中发生的事件和教程：部署具有验证测试的 Amazon ECS 服务。

钩子在 Amazon ECS 部署中的运行顺序。

在 Amazon ECS 部署中，事件挂钩按以下顺序运行：


Start
BeforeInstall
Install
AfterInstall
AllowTestTraffic
AfterAllowTestTraffic
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 部署中的 “开始” TestTrafficAllowTraffic、“安装” 和 “结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。

“hooks”部分的结构

以下示例说明了 `'hooks'` 部分的结构。

使用 YAML：

```
Hooks:
  - BeforeInstall: "BeforeInstallHookFunctionName"
  - AfterInstall: "AfterInstallHookFunctionName"
  - AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
  - BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
  - AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

使用 JSON：

```
"Hooks": [
  {
   "BeforeInstall": "BeforeInstallHookFunctionName"
  },
  {
   "AfterInstall": "AfterInstallHookFunctionName"
  },
  {
   "AfterAllowTestTraffic": "AfterAllowTestTrafficHookFunctionName"
  },
  {
   "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
  },
  {
   "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
  }
 ]
}
```

Lambda“hooks”函数示例

使用该`'hooks'`部分指定一个 Lambda 函数，该函数 CodeDeploy 可以调用该函数来验证 Amazon ECS 部署。对于`BeforeInstall`、`AfterInstall`、`AfterAllowTestTraffic`、`BeforeAllowTraffic` 和 `AfterAllowTraffic` 部署生命周期事件，您可以使用相同函数或不同的函数。验证测试完成后，Lambda `AfterAllowTraffic` 函数会回调 CodeDeploy并提供或的结果。`Succeeded` `Failed`

> **Important**
>
> 如果 Lambda 验证功能 CodeDeploy 未在一小时内发出通知，则认为部署已失败。

在调用 Lambda 挂钩函数之前，必须使用 `putLifecycleEventHookExecutionStatus` 命令向服务器通知部署 ID 和生命周期事件挂钩执行 ID。

下面是一个使用 Node.js 编写的 Lambda 挂钩函数示例。

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };
```

```
    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

AppSpec Amazon Lambda 部署的 “挂钩” 部分

主题

- Amazon Lambda 部署的生命周期事件挂钩列表
- 挂钩在 Lambda 函数版本部署中的运行顺序
- “hooks”部分的结构
- Lambda“hooks”函数示例

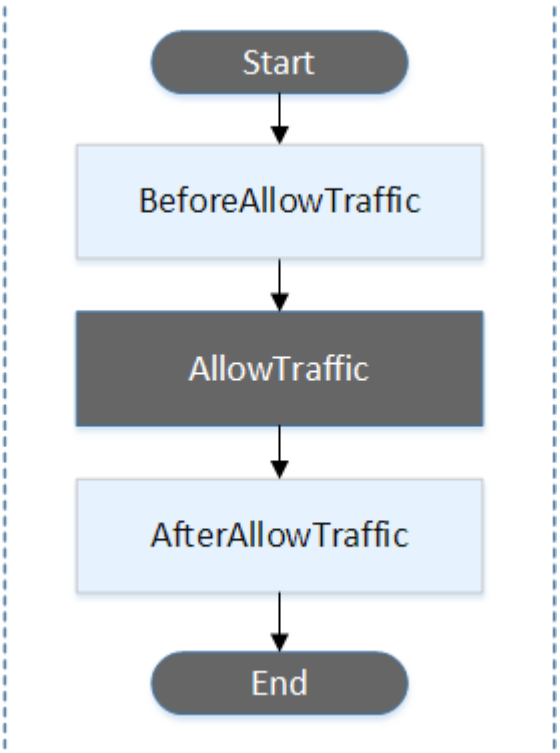
Amazon Lambda 部署的生命周期事件挂钩列表

Amazon Lambda 挂钩是一个 Lambda 函数，在生命周期事件名称后的新行中使用字符串指定。对于每次部署，每个挂钩将执行一次。以下是可在您的 AppSpec 文件中使用的挂钩的描述。

- BeforeAllowTraffic— 用于在流量转移到已部署的 Lambda 函数版本之前运行任务。
- AfterAllowTraffic— 用于在所有流量转移到已部署的 Lambda 函数版本后运行任务。

挂钩在 Lambda 函数版本部署中的运行顺序

在无服务器 Lambda 函数版本部署中，事件挂钩按以下顺序运行：



> **Note**
>
> 部署中的 “开始” AllowTraffic、“结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。

“hooks”部分的结构

以下示例说明了“hooks”部分的结构。

使用 YAML：

```
hooks:
   - BeforeAllowTraffic: BeforeAllowTrafficHookFunctionName
   - AfterAllowTraffic: AfterAllowTrafficHookFunctionName
```

使用 JSON：

```
"hooks": [{
    "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
    },
```

```
    {
    "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
}]
```

Lambda“hooks”函数示例

使用 “挂钩” 部分指定一个 Lambda 函数，该函数 CodeDeploy 可以调用该函数来验证 Lambda 部署。对于 `BeforeAllowTraffic` 和 `AfterAllowTraffic` 部署生命周期事件，您可以使用相同函数或不同的函数。验证测试完成后，Lambda 验证函数会回调 CodeDeploy 并提供或的结果。`Succeeded` `Failed`

> **Important**
>
> 如果 Lambda 验证功能 CodeDeploy 未在一小时内发出通知，则认为部署已失败。

在调用 Lambda 挂钩函数之前，必须使用 `putLifecycleEventHookExecutionStatus` 命令向服务器通知部署 ID 和生命周期事件挂钩执行 ID。

下面是一个使用 Node.js 编写的 Lambda 挂钩函数示例。

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
```

```
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

AppSpec EC2/本地部署的 “挂钩” 部分

主题

- 生命周期事件挂钩的列表
- 生命周期事件挂钩可用性
- 挂钩在部署中的运行顺序
- “hooks”部分的结构
- 在挂钩脚本中引用文件
- 挂钩的环境变量可用性
- 挂钩示例

生命周期事件挂钩的列表

每次部署到实例时，EC2/本地部署挂钩都会执行一次。在一个挂钩中，可以指定运行一个或多个脚本。生命周期事件的每个挂钩在单独的行中使用字符串指定。以下是可在您的 AppSpec 文件中使用的挂钩的描述。

有关哪些生命周期事件挂钩对哪些部署和回滚类型有效的信息，请参阅生命周期事件挂钩可用性。

- `ApplicationStop` – 此部署生命周期事件发生在下载应用程序修订之前。您可以为此事件指定脚本，以便从容地停止应用程序或在部署准备过程中删除当前已安装的软件包。用于此部署生命周期事件 AppSpec 的文件和脚本来自先前成功部署的应用程序修订版。

> **Note**
>
> 在部署到实例之前，AppSpec 文件不存在于该实例。因此，`ApplicationStop` 挂钩在您首次部署到实例时不会运行。您可以在第二次部署到实例时使用 `ApplicationStop` 挂钩。

要确定上次成功部署的应用程序修订版的位置，CodeDeploy 代理会查找*deployment-group-id*`_last_successful_install`文件中列出的位置。此文件位于：

`/opt/codedeploy-agent/deployment-root/deployment-instructions`亚马逊 Linux、Ubuntu 服务器和 RHEL 亚马逊 EC2 实例上的文件夹。

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions`Windows 服务器亚马逊 EC2 实例上的文件夹。

要对在 `ApplicationStop` 部署生命周期事件期间失败的部署进行故障排除，请参阅 对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除。

- `DownloadBundle`— 在此部署生命周期事件中，CodeDeploy 代理会将应用程序修订文件复制到临时位置：

  `/opt/codedeploy-agent/deployment-root/`*deployment-group-id*`/`*deployment-id*`/deployment-archive`亚马逊 Linux、Ubuntu 服务器和 RHEL 亚马逊 EC2 实例上的文件夹。

  `C:\ProgramData\Amazon\CodeDeploy\`*deployment-group-id*`\`*deployment-id*`\deployment-archive`Windows 服务器亚马逊 EC2 实例上的文件夹。

  此事件是为 CodeDeploy 代理保留的，不能用于运行脚本。

  要对在 `DownloadBundle` 部署生命周期事件期间失败的部署进行故障排除，请参阅 使用以下命令对失败的 DownloadBundle 部署生命周期事件进行故障排除 UnknownError：未打开供读取。

- `BeforeInstall` – 您可以使用此部署生命周期事件执行预安装任务，例如解密文件和创建当前版本的备份。
- `Install`— 在此部署生命周期事件中，CodeDeploy代理将修订文件从临时位置复制到最终目标文件夹。此事件是为 CodeDeploy 代理保留的，不能用于运行脚本。
- `AfterInstall` – 您可以使用此部署生命周期事件执行配置应用程序或更改文件权限等任务。

- `ApplicationStart` – 此部署生命周期事件通常用于重新启动在 `ApplicationStop` 期间停止的服务。
- `ValidateService` – 这是最后一个部署生命周期事件。它用于验证部署已成功完成。
- `BeforeBlockTraffic` – 在从负载均衡器取消注册实例之前，您可以使用此部署生命周期事件在这些实例上运行任务。

  要对在 `BeforeBlockTraffic` 部署生命周期事件期间失败的部署进行故障排除，请参阅 对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除。
- `BlockTraffic` – 在此部署生命周期事件期间，阻止互联网流量访问当前正在处理流量的实例。此事件是为 CodeDeploy 代理保留的，不能用于运行脚本。
- `AfterBlockTraffic` – 在从相应的负载均衡器取消注册实例之后，您可以使用此部署生命周期事件在这些实例上运行任务。

  要对在 `AfterBlockTraffic` 部署生命周期事件期间失败的部署进行故障排除，请参阅 对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除。
- `BeforeAllowTraffic` – 在将实例注册到负载均衡器之前，您可以使用此部署生命周期事件在这些实例上运行任务。
- `AllowTraffic` – 在此部署生命周期事件期间，允许互联网流量在部署后访问实例。此事件是为 CodeDeploy 代理保留的，不能用于运行脚本。
- `AfterAllowTraffic` – 在将实例注册到负载均衡器之后，您可以使用此部署生命周期事件在这些实例上运行任务。

生命周期事件挂钩可用性

下表列出了适用于每个部署和回滚方案的生命周期事件挂钩。

生命周期事件名称	Auto Scaling 启动部署[1]	Auto Scaling 终止部署[1]	就地部署[2]	蓝/绿部署：原始实例	蓝/绿部署：替换实例	蓝/绿部署回滚：原始实例	蓝/绿部署回滚：替换实例
ApplicationStop	✓	✓	✓		✓		
DownloadBundle[3]	✓		✓		✓		

生命周期事件名称	Auto Scaling 启动部署[1]	Auto Scaling 终止部署[1]	就地部署[2]	蓝/绿部署：原始实例	蓝/绿部署：替换实例	蓝/绿部署回滚：原始实例	蓝/绿部署回滚：替换实例
BeforeInstall	✓		✓		✓		
Install[3]	✓		✓		✓		
AfterInstall	✓		✓		✓		
ApplicationStart	✓		✓		✓		
ValidateService	✓		✓		✓		
BeforeBlockTraffic		✓	✓	✓			✓
BlockTraffic[3]		✓	✓	✓			✓
AfterBlockTraffic		✓	✓	✓			✓
BeforeAllowTraffic	✓		✓		✓	✓	
AllowTraffic[3]	✓		✓		✓	✓	
AfterAllowTraffic	✓		✓		✓	✓	

生命周期事件名称	Auto Scaling 启动部署[1]	Auto Scaling 终止部署[1]	就地部署[2]	蓝/绿部署：原始实例	蓝/绿部署：替换实例	蓝/绿部署回滚：原始实例	蓝/绿部署回滚：替换实例

[1] 有关 Amazon A EC2 uto Scaling 部署的信息，请参阅Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy。

[2] 也适用于就地部署的回滚。

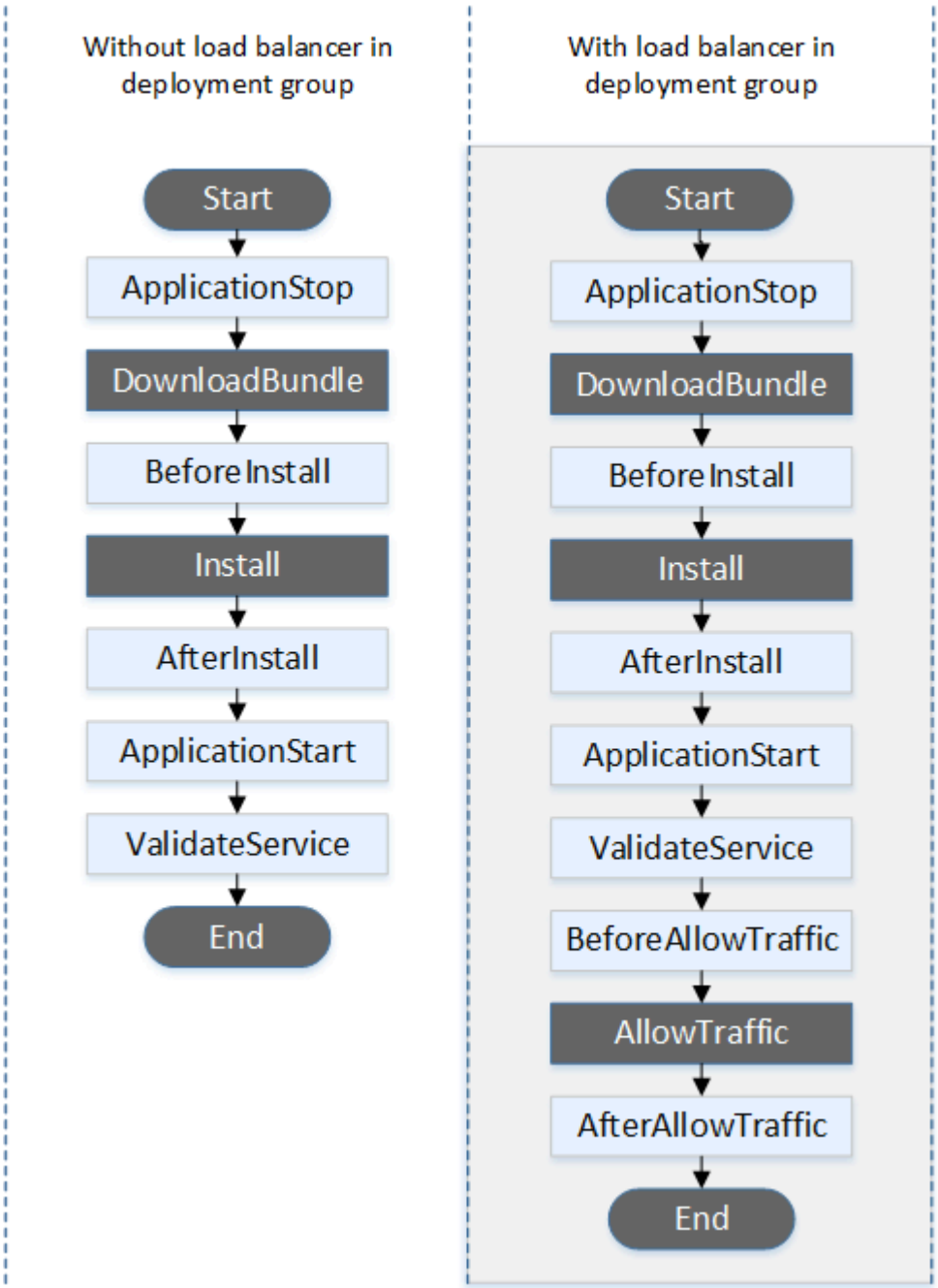
[3] 预留用于 CodeDeploy 操作。不能用于运行脚本。

挂钩在部署中的运行顺序

Auto Scaling 启动部署

在 Auto Scaling 启动部署期间，按以下顺序 CodeDeploy 运行事件挂钩。

有关 Auto Scaling 启动部署的更多信息，请参阅 Amazon A EC2 uto Scaling 是如何使用的 CodeDeploy。

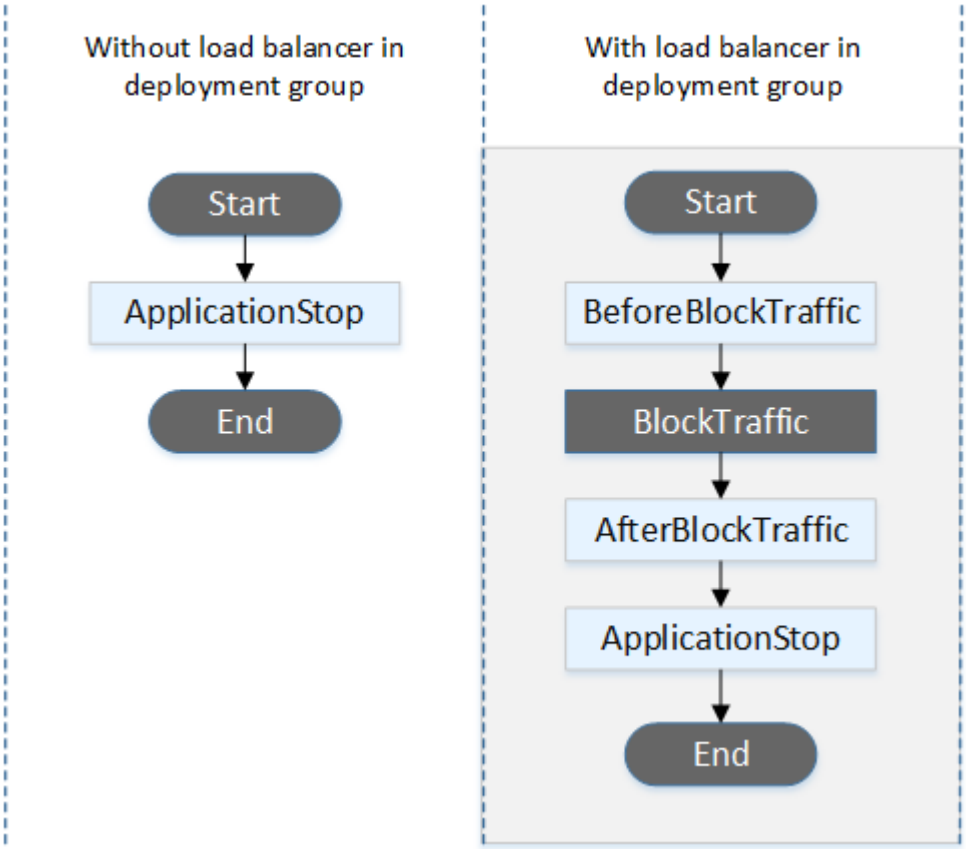


> **Note**
>
> 部署中的 “开始” AllowTraffic、“安装” 和 “结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。DownloadBundle但是，您可以编辑 AppSpec 文件`'files'`部分以指定在安装事件期间安装的内容。

Auto Scaling 终止部署

在 Auto Scaling 终止部署期间，按以下顺序 CodeDeploy 运行事件挂钩。

有关 Auto Scaling 终止部署的更多信息，请参阅 在 Auto Scaling 横向缩减事件期间启用终止部署。

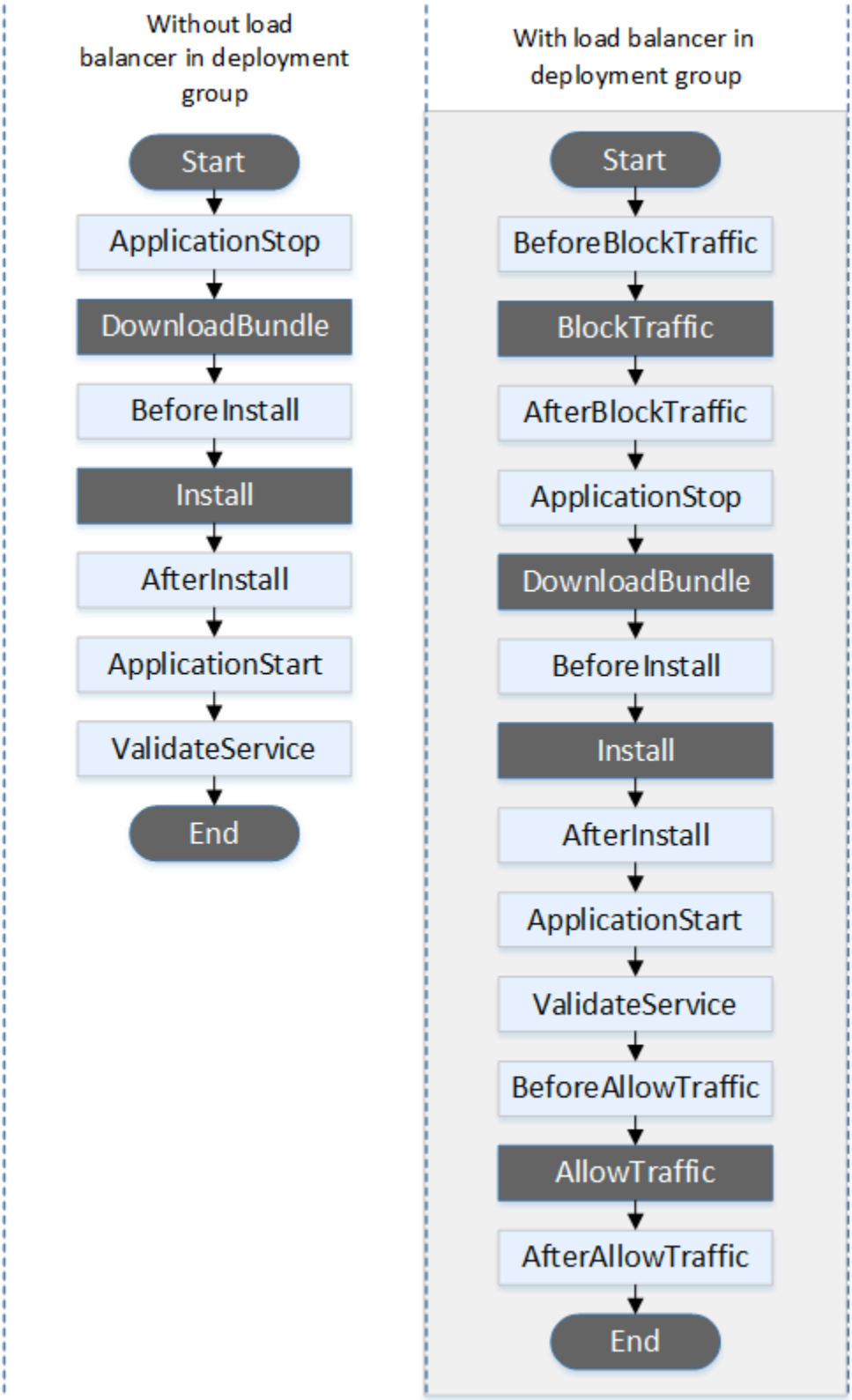


> **Note**
>
> 部署中的 “开始” BlockTraffic、“结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。

就地部署

在就地部署中（包括就地部署的回滚），事件挂钩按以下顺序运行：

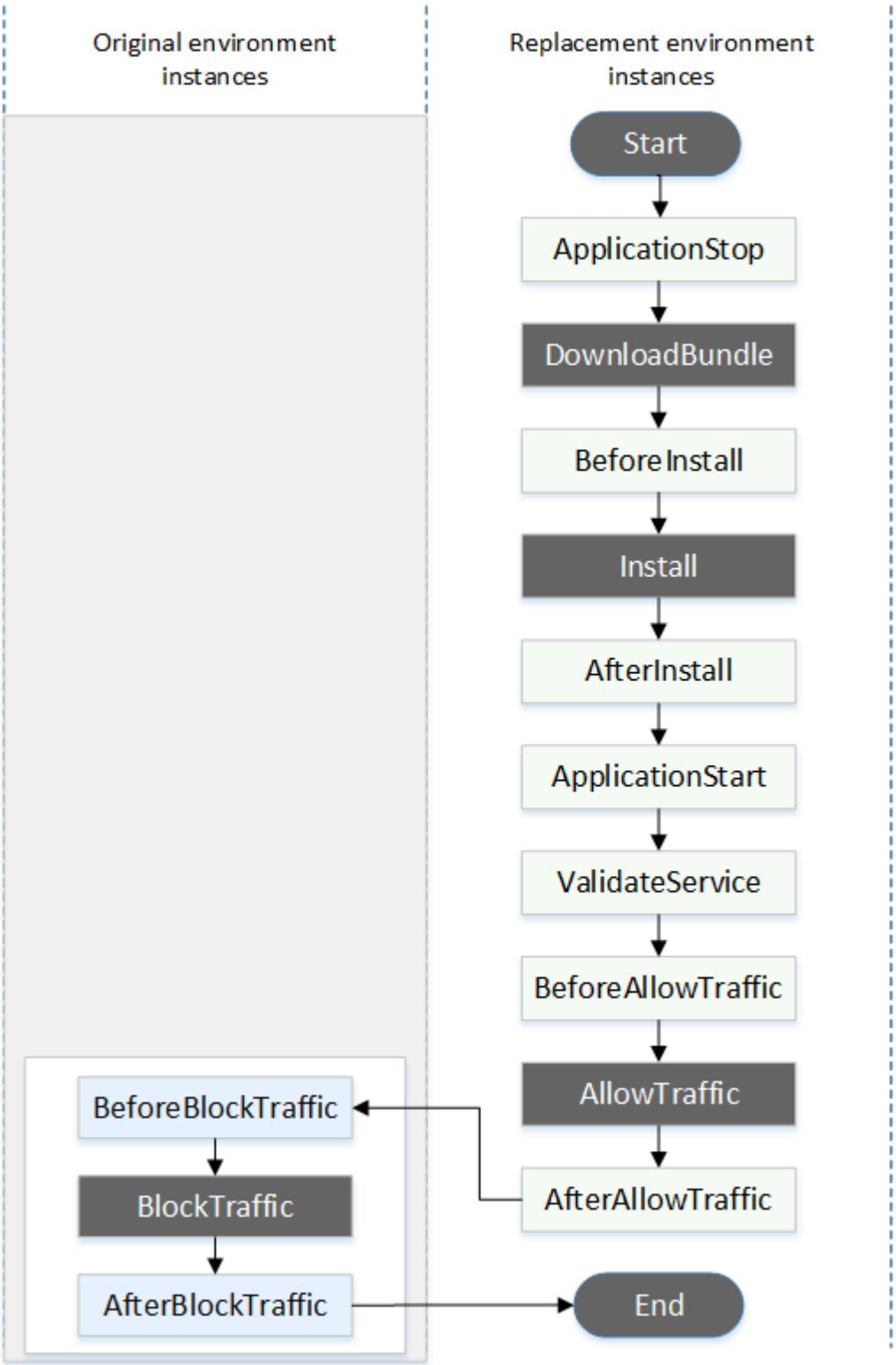
> **Note**
>
> 对于就地部署，只有在部署组中从 ELB 中指定 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer 时，与阻止和允许流量相关的六个挂钩才适用。


Without load balancer in deployment group
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
End
With load balancer in deployment group
Start
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 部署中的 “开始”、“安装” 和 “结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。DownloadBundle但是，您可以编辑 AppSpec 文件`'files'`部分以指定在安装事件期间安装的内容。

蓝/绿部署

在 blue/green 部署中，事件挂钩按以下顺序运行：


Original environment instances
Replacement environment instances
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic

> **Note**
>
> 部署中的 “开始”、“安装” BlockTrafficAllowTraffic、“、” 和 “结束” 事件无法编写脚本，这就是它们在此图中以灰色显示的原因。DownloadBundle但是，您可以编辑文件的 “文件” 部分，AppSpec 以指定安装事件期间安装的内容。

“hooks”部分的结构

`'hooks'` 部分具有以下结构：

```
hooks:
   deployment-lifecycle-event-name:
     - location: script-location
       timeout: timeout-in-seconds
       runas: user-name
```

可以在 hook 条目中的部署生命周期事件名称后包括以下元素：

地点

必需。修订的脚本文件包的位置。您在 `hooks` 部分中指定的脚本的位置是应用程序修订包根目录的相对路径。有关更多信息，请参阅 为 CodeDeploy 规划修订。

timeout

可选。在脚本被视为失败之前允许其执行的秒数。默认值为 3600 秒（1 小时）。

> **Note**
>
> 3600 秒（1 小时）是允许每个部署生命周期事件脚本执行的最长时间。如果脚本超过此限制，则部署将停止，并且部署到实例将失败。确保在 timeout 中为每个部署生命周期事件的所有脚本指定的总秒数不超过此限制。

runas

可选。运行脚本时要模拟的用户。默认情况下，这是在实例上运行的 CodeDeploy 代理。CodeDeploy 不存储密码，因此，如果 runas 用户需要密码，则无法模拟该用户。此元素仅适用于 Amazon Linux 和 Ubuntu Server 实例。

在挂钩脚本中引用文件

如果您要按中所述将脚本连接到 CodeDeploy 生命周期事件AppSpec “挂钩” 部分，并且想要在脚本中引用文件（例如`helper.sh`），则需要`helper.sh`使用以下命令进行指定：

- （推荐）绝对路径。请参阅使用绝对路径。
- 相对路径。请参阅使用相对路径。

使用绝对路径

要使用文件的绝对路径来引用文件，可以执行以下任一操作：

- 在 AppSpec 文件`files`部分的`destination`属性中指定绝对路径。然后，在挂钩脚本中指定相同的绝对路径。有关更多信息，请参阅 AppSpec “文件” 部分（仅EC2限本地部署）。
- 在挂钩脚本中指定一个动态绝对路径。有关更多信息，请参阅部署存档位置。

部署存档位置

在DownloadBundle生命周期事件期间， CodeDeploy 代理会将要部署的修订提取到以下格式的目录中：

root-directory/*deployment-group-id*/*deployment-id*/deployment-archive

路径*root-directory*部分始终设置为下表所示的默认路径，或者由`:root_dir`配置设置控制。有关配置设置的更多信息，请参阅CodeDeploy 代理配置参考。

代理平台	默认根目录
Linux – 所有 rpm 发行版	`/opt/codedeploy-agent/deployment-root`
Ubuntu Server – 所有 deb 发行版	`/opt/codedeploy-agent/deployment-root`
Windows Server	`%ProgramData%\Amazon\CodeDeploy`

从挂钩脚本中，您可以使用根目录路径以及 DEPLOYMENT_ID 和 DEPLOYMENT_GROUP_ID 环境变量访问当前的部署存档。有关您能使用的变量的更多信息，请参阅挂钩的环境变量可用性。

下面的示例介绍了如何在 Linux 上访问位于修订根目录下的 data.json 文件：

```
#!/bin/bash

rootDirectory="/opt/codedeploy-agent/deployment-root" # note: this will be different if
 you
                                                       # customize the :root_dir
 configuration
dataFile="$rootDirectory/$DEPLOYMENT_GROUP_ID/$DEPLOYMENT_ID/deployment-archive/
data.json"
data=$(cat dataFile)
```

再举一个例子，介绍了如何在 Windows 上使用 Powershell 访问位于修订根目录下的 data.json 文件：

```
$rootDirectory="$env:ProgramData\Amazon\CodeDeploy" # note: this will be different if
 you
                                                     # customize the :root_dir
 configuration
$dataFile="$rootDirectory\$env:DEPLOYMENT_GROUP_ID\$env:DEPLOYMENT_ID\deployment-
archive\data.json"
$data=(Get-Content $dataFile)
```

使用相对路径

要使用文件的相对路径来引用文件，你需要知道 CodeDeploy 代理的工作目录。文件路径是相对于该目录的。

下表显示了 CodeDeploy 代理的每个支持平台的工作目录。

代理平台	流程管理方法	生命周期事件脚本的工作目录
Linux – 所有 rpm 发行版	systemd（默认）	/
	init.d – 了解更多	/opt/codedeploy-agent

代理平台	流程管理方法	生命周期事件脚本的工作目录
Ubuntu Server – 所有 debian 发行版	全部	`/opt/codedeploy-agent`
Windows Server	不适用	`C:\Windows\System32`

挂钩的环境变量可用性

在每个部署生命周期事件期间，挂钩脚本可以访问以下环境变量：

APPLICATION_NAME

属于当前部署一部分 CodeDeploy 的应用程序的名称（例如，`WordPress_App`）。

DEPLOYMENT_ID

ID CodeDeploy 已分配给当前部署（例如，`d-AB1CDEF23`）。

DEPLOYMENT_GROUP_NAME

属于当前部署一部分 CodeDeploy 的部署组的名称（例如`WordPress_DepGroup`）。

DEPLOYMENT_GROUP_ID

属于当前部署一部分的部署组的 ID（例如，`b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE`）。CodeDeploy

LIFECYCLE_EVENT

当前部署生命周期事件的名称（例如 `AfterInstall`）。

这些是每个部署生命周期事件的本地环境变量。

根据部署包的来源，还有其他环境变量可用于挂接脚本：

来自 Amazon S3 的捆绑包

- BUNDLE_BUCKET

  从中下载部署包的 Amazon S3 存储桶的名称（例如 `my-s3-bucket`）。
- BUNDLE_KEY

  Amazon S3 存储桶中下载的捆绑包的对象键（例如 `WordPress_App.zip`）。

- BUNDLE_VERSION

  捆绑包的对象版本（例如 `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`）。只有在 Amazon S3 存储桶启用了对象版本控制时，才会设置此变量。
- BUNDLE_ETAG

  捆绑包的对象 etag（例如，`b10a8db164e0754105b7a99be72e3fe5-4`）。

捆绑包来自 GitHub

- BUNDLE_COMMIT

  Git 生成的捆绑包的 SHA256 提交哈希值（例如`d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26`）。

如果 DEPLOYMENT_GROUP_NAME 的值等于 `Staging`，则以下脚本会将 Apache HTTP 服务器上的侦听端口更改为 9090 而非 80。必须在 `BeforeInstall` 部署生命周期事件期间调用此脚本：

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

如果 DEPLOYMENT_GROUP_NAME 环境变量的值等于 `Staging`，则以下脚本示例会将其错误日志中记录的消息的详细级别从警告更改为调试。必须在 `BeforeInstall` 部署生命周期事件期间调用此脚本：

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

以下脚本示例将指定网页中的文本替换为显示这些环境变量值的文本。必须在 `AfterInstall` 部署生命周期事件期间调用此脚本：

```
#!/usr/bin/python

import os
```

```
strToSearch="<h2>This application was deployed using CodeDeploy.</h2>"
strToReplace="<h2>This page for "+os.environ['APPLICATION_NAME']+"
 application and "+os.environ['DEPLOYMENT_GROUP_NAME']+" deployment group with
 "+os.environ['DEPLOYMENT_GROUP_ID']+" deployment group ID was generated by a
 "+os.environ['LIFECYCLE_EVENT']+" script during "+os.environ['DEPLOYMENT_ID']+"
 deployment.</h2>"

fp=open("/var/www/html/index.html","r")
buffer=fp.read()
fp.close()

fp=open("/var/www/html/index.html","w")
fp.write(buffer.replace(strToSearch,strToReplace))
fp.close()
```

挂钩示例

以下是 hooks 条目的示例，该条目为 `AfterInstall` 生命周期事件指定两个挂钩：

```
hooks:
   AfterInstall:
     - location: Scripts/RunResourceTests.sh
       timeout: 180
     - location: Scripts/PostDeploy.sh
       timeout: 180
```

`Scripts/RunResourceTests.sh` 脚本在部署过程的 `AfterInstall` 阶段运行。如果该脚本的运行时间超过 180 秒（3 分钟），则部署将失败。

您在“hooks”部分中指定的脚本的位置是应用程序修订包根目录的相对路径。在上述示例中，名为 `RunResourceTests.sh` 的文件位于名为 `Scripts` 的目录中。该 `Scripts` 目录位于包的根级别。有关更多信息，请参阅 为 CodeDeploy 规划修订。

AppSpec 文件示例

本主题提供了 Amazon Lambda 和 EC2 /Londest 部署的示例 AppSpec 文件。

主题

- AppSpec Amazon ECS 部署的文件示例
- AppSpec Amazon Lambda 部署的文件示例

- AppSpec EC2/本地部署的文件示例

AppSpec Amazon ECS 部署的文件示例

以下是使用 YAML 编写的用于部署 Amazon ECS 服务的 AppSpec 文件的示例。

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:us-east-1:111222333444:task-definition/my-task-
definition-family-name:1"
        LoadBalancerInfo:
          ContainerName: "SampleApplicationName"
          ContainerPort: 80
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["subnet-1234abcd","subnet-5678abcd"]
            SecurityGroups: ["sg-12345678"]
            AssignPublicIp: "ENABLED"
        CapacityProviderStrategy:
          - Base: 1
            CapacityProvider: "FARGATE_SPOT"
            Weight: 2
          - Base: 0
            CapacityProvider: "FARGATE"
            Weight: 1
Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "LambdaFunctionToValidateAfterTestTrafficStarts"
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterAllowingProductionTraffic"
```

下面是用 JSON 编写的上述示例的版本。

```
{
    "version": 0.0,
    "Resources": [
```

```
        {
            "TargetService": {
                "Type": "AWS::ECS::Service",
                "Properties": {
                    "TaskDefinition": "arn:aws:ecs:us-east-1:111222333444:task-
definition/my-task-definition-family-name:1",
                    "LoadBalancerInfo": {
                        "ContainerName": "SampleApplicationName",
                        "ContainerPort": 80
                    },
                    "PlatformVersion": "LATEST",
                    "NetworkConfiguration": {
                        "AwsvpcConfiguration": {
                            "Subnets": [
                                "subnet-1234abcd",
                                "subnet-5678abcd"
                            ],
                            "SecurityGroups": [
                                "sg-12345678"
                            ],
                            "AssignPublicIp": "ENABLED"
                        }
                    },
                    "CapacityProviderStrategy": [
                        {
                            "Base" : 1,
                            "CapacityProvider" : "FARGATE_SPOT",
                            "Weight" : 2
                        },
                        {
                            "Base" : 0,
                            "CapacityProvider" : "FARGATE",
                            "Weight" : 1
                        }
                    ]
                }
            }
        }
    ],
    "Hooks": [
        {
            "BeforeInstall": "LambdaFunctionToValidateBeforeInstall"
        },
        {
```

```
            "AfterInstall": "LambdaFunctionToValidateAfterInstall"
        },
        {
            "AfterAllowTestTraffic": "LambdaFunctionToValidateAfterTestTrafficStarts"
        },
        {
            "BeforeAllowTraffic":
 "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
        },
        {
            "AfterAllowTraffic":
 "LambdaFunctionToValidateAfterAllowingProductionTraffic"
        }
    ]
}
```

下面是部署期间的事件序列：

1. 在替换任务集上安装更新的 Amazon ECS 应用程序之前，名为 `LambdaFunctionToValidateBeforeInstall` 的 Lambda 函数运行。
2. 在替换任务集上安装更新的 Amazon ECS 应用程序之后，但在该应用程序接收任何流量之前，名为 `LambdaFunctionToValidateAfterInstall` 的 Lambda 函数运行。
3. 在替换任务集上的 Amazon ECS 应用程序开始接收来自测试侦听器的流量之后，名为 `LambdaFunctionToValidateAfterTestTrafficStarts` 的 Lambda 函数运行。此函数可能会运行验证测试以确定部署是否继续。如果未在部署组中指定测试侦听器，则会忽略此挂钩。
4. 在完成 `AfterAllowTestTraffic` 挂钩中的任何验证测试之后，并且在将生产流量提供给更新的 Amazon ECS 应用程序之前，名为 `LambdaFunctionToValidateBeforeAllowingProductionTraffic` 的 Lambda 函数运行。
5. 生产流量在更换任务集上提供给更新的 Amazon ECS 应用程序后，名为 `LambdaFunctionToValidateAfterAllowingProductionTraffic` 的 Lambda 函数运行。

在任何挂钩期间运行的 Lambda 函数都可以执行验证测试或者收集流量指标。

AppSpec Amazon Lambda 部署的文件示例

以下是使用 YAML 编写的用于部署 Lambda 函数版本的 AppSpec 文件的示例。

```
version: 0.0
Resources:
```

```
  - myLambdaFunction:
      Type: AWS::Lambda::Function
      Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
Hooks:
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

下面是用 JSON 编写的上述示例的版本。

```
{
  "version": 0.0,
  "Resources": [{
   "myLambdaFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "myLambdaFunction",
     "Alias": "myLambdaFunctionAlias",
     "CurrentVersion": "1",
     "TargetVersion": "2"
    }
   }
  }],
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
      },
      {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
   }
  ]
 }
```

下面是部署期间的事件序列：

1. 在将流量从名为 `myLambdaFunction` 的 Lambda 函数的版本 1 转移到版本 2 前，运行名为 `LambdaFunctionToValidateBeforeTrafficShift` 的 Lambda 函数，该函数将验证部署是否已准备好开始流量转移。
2. 如果 `LambdaFunctionToValidateBeforeTrafficShift` 返回了退出代码 0（成功），则开始将流量转移到 `myLambdaFunction` 的版本 2。此部署的部署配置确定流量转移的速率。

3. 在完成将流量从名为 `myLambdaFunction` 的 Lambda 函数的版本 1 转移到版本 2 后，运行名为 `LambdaFunctionToValidateAfterTrafficShift` 的 Lambda 函数，该函数将验证部署是否已成功完成。

AppSpec EC2/本地部署的文件示例

以下是就地部署到亚马逊 Linux、Ubuntu 服务器或 RHEL 实例 AppSpec 的文件示例。

> **Note**
>
> 部署到 Windows Server 实例不支持 `runas` 元素。如果您要部署到 Windows 服务器实例，请不要将其包含在 AppSpec 文件中。

```
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

对于 Windows Server 实例，将 `os: linux` 更改为 `os: windows`。另外，您还必须完全限定 `destination` 路径（例如 `c:\temp\webapps\Config` 和 `c:\temp\webapps\myApp`）。不包括 `runas` 元素。

下面是部署期间的事件序列：

1. 运行位于 `Scripts/UnzipResourceBundle.sh` 的脚本。
2. 如果前面的脚本返回了退出代码 0（成功），则运行位于 `Scripts/UnzipDataBundle.sh` 中的脚本。
3. 将文件从 `Config/config.txt` 路径复制到 `/webapps/Config/config.txt` 路径中。
4. 以递归方式将 `source` 目录中的所有文件复制到 `/webapps/myApp` 目录中。
5. 运行位于 `Scripts/RunResourceTests.sh` 中的脚本，超时时间为 180 秒（3 分钟）。
6. 运行位于 `Scripts/RunFunctionalTests.sh` 中的脚本，超时时间为 3600 秒（1 小时）。
7. 以 `Scripts/MonitorService.sh` 用户身份运行位于 `codedeploy` 中的脚本，超时时间为 3600 秒（1 小时）。

AppSpec 文件间距

以下是 AppSpec 文件间距的正确格式。方括号中的数字表示各项之间必须存在的空格数。例如，`[4]`表示在项目之间插入四个空格。 CodeDeploy 如果 AppSpec 文件中的位置和空格数不正确，则会引发一个可能难以调试的错误。

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
[4]type:
[6]-[1]object-type
hooks:
[2]deployment-lifecycle-event-name:
[4]-[1]location:[1]script-location
```

```
[6]timeout:[1]timeout-in-seconds
[6]runas:[1]user-name
```

以下是间距正确的 AppSpec 文件示例：

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

有关间距的更多信息，请参阅 YAML 规范。

验证您的 AppSpec 文件和文件位置

文件语法

您可以使用 Amazon提供的 Assi AppSpec stant 脚本来验证 AppSpec 文件的内容。你可以在上找到脚本和 AppSpec 文件模板GitHub。

您还可以使用基于浏览器的工具帮助您检查 YAML 语法，如 YAML Lint 或 Online YAML Parser。

文件位置

要验证您是否已将 AppSpec 文件放在应用程序源内容目录结构的根目录中，请运行以下命令之一：

在本地 Linux、macOS 或 Unix 实例上：

```
ls path/to/root/directory/appspec.yml
```

如果 AppSpec 文件不在那里，则会显示 “没有这样的文件或目录” 错误。

在本地 Windows 实例上：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec 文件不在那里，则会显示 “找不到文件” 错误。

CodeDeploy 代理配置参考

安装 CodeDeploy 代理后，将在实例上放置一个配置文件。此配置文件指定 CodeDeploy 与实例交互时使用的目录路径和其他设置。可以更改此文件中的某些配置选项。

对于 Amazon Linux、Ubuntu Server 和 Red Hat Enterprise Linux（RHEL）实例，配置文件名为 `codedeployagent.yml`。它放置在 `/etc/codedeploy-agent/conf` 目录中。

对于 Windows Server 实例，配置文件名为 `conf.yml`。它放置在 `C:\ProgramData\Amazon\CodeDeploy` 目录中。

配置设置包括：

:log_aws_wire:	设为 `true` 表示 CodeDeploy 代理从 Amazon S3 捕获线路日志并将它们写入一个名为 `codedeploy-agent.wire.log` 的文件，该文件位于 :log_dir: 设置指向的位置。 **Warning** 您仅应在捕获线路日志需要的时间内将 :log_aws_wire: 设置为 `true`。`codedeploy-agent.wire.log` 文件可以快速增长到非常大的大小。此文件中的线路日志输出可能包含敏感信息，包括在此设置设为 `true` 时传入或传出 Amazon S3 的文件的纯

	文本内容。此设置设为 `true` 时，线路日志包含有关与 Amazon 账户关联的所有 Amazon S3 活动的信息，而不仅仅包含与 CodeDeploy 部署相关的活动的信息。 默认设置为 `false`。 此设置适用于所有实例类型。您必须将此配置设置添加到 Windows Server 实例才能使用它。
:log_dir:	实例上用于存储与 CodeDeploy 代理操作相关的日志文件的文件夹。 对于 Amazon Linux、Ubuntu Server 和 RHEL 实例，默认设置是 `'/var/log/aws/codedeploy-agent'` ，对于 Windows Server 实例，默认设置是 `C:\ProgramData\Amazon\CodeDeploy\log` 。
:pid_dir:	存储 `codedeploy-agent.pid` 的文件夹。 此文件包含 CodeDeploy 代理的进程 ID（PID）。默认设置为 `'/opt/codedeploy-agent/state/.pid'` 。 此设置仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。
:program_name:	CodeDeploy 代理程序名称。 默认设置为 `codedeploy-agent` 。 此设置仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

:root_dir:	实例上用于存储相关修订、部署历史记录和部署脚本的文件夹。 对于 Amazon Linux、Ubuntu Server 和 RHEL 实例，默认设置是 `/opt/codedeploy-agent/deployment-root`，对于 Windows Server 实例，默认设置是 `C:\ProgramData\Amazon\CodeDeploy`。
:verbose:	设为 `true` 表示 CodeDeploy 代理打印实例上的调试消息日志文件。 默认设置为 `false`。
:wait_between_runs:	CodeDeploy 代理为挂起的部署轮询 CodeDeploy 的时间间隔（以秒为单位）。 默认设置为 `1`。
:on_premises_config_file:	对于本地实例，这是名为 `codedeploy.onpremises.yml`（对于 Ubuntu Server 和 RHEL）或 `conf.onpremises.yml`（对于 Windows Server）的配置文件的备用位置的路径。 默认情况下，这些文件存储在 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`（对于 Ubuntu Server 和 RHEL）以及 `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`（对于 Windows Server）中。 在 CodeDeploy 代理版本 1.0.1.686 及更高版本中可用。

:proxy_uri:	（可选）您希望 CodeDeploy 代理通过其连接到 Amazon 以执行 CodeDeploy 操作的 HTTP 代理。使用类似于 `https://user:password@my.proxy:443/path?query` 的格式。 在 CodeDeploy 代理版本 1.0.1.824 及更高版本中可用。
:max_revisions:	（可选）您希望 CodeDeploy 代理存档的部署组的应用程序修订的数量。超过指定数量的任何修订都将被删除。 输入任意正整数。如果不指定任何值，则除了当前部署的修订之外，CodeDeploy 还将保留五个最新修订。 在 CodeDeploy 代理版本 1.0.1.966 及更高版本中受支持。
:enable_auth_policy：	（可选）如果您想使用 IAM 授权来配置访问控制和限制 CodeDeploy 代理正在使用的 IAM 角色或用户的权限，请将其设置为 `true`。要将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用，此值必须是 `true`。 默认设置为 `false`。
:disable_imds_v1:	此设置在 CodeDeploy 代理 1.7.0 及更高版本中可用。 设置为 `true` 可在出现 IMDSv2 错误时禁用回退到 IMDSv1。默认为 `false`（启用回退）。

相关主题

与 CodeDeploy 代理合作

管理 CodeDeploy 代理操作

Amazon CloudFormation 模板供 CodeDeploy参考

本节介绍专为处理 CodeDeploy 部署而设计的 Amazon CloudFormation 资源、转换和挂钩。有关创建由 Amazon CloudFormation 挂钩管理的堆栈更新的演练 CodeDeploy，请参阅 通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation

> **Note**
>
> Amazon CloudFormation 钩子是生命周期事件挂钩 Amazon CloudFormation 组件的一部分 Amazon ，不同于 CodeDeploy生命周期事件挂钩。

除了中提供的其他方法外 CodeDeploy，您还可以使用 Amazon CloudFormation 模板来执行以下任务：

- 创建应用程序。
- 创建部署组并指定目标修订。
- 创建部署配置。
- 创建 Amazon EC2 实例。

Amazon CloudFormation 是一项服务，可帮助您使用模板对 Amazon 资源进行建模和设置。 Amazon CloudFormation 模板是格式符合 JSON 标准的文本文件。您可以创建一个描述所有所需 Amazon 资源的模板，并 Amazon CloudFormation 负责为您配置和配置这些资源。

有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的什么是 Amazon CloudFormation？以及使用 Amazon CloudFormation 模板。

如果您计划在组织 CodeDeploy 中使用与兼容的 Amazon CloudFormation 模板，则作为管理员，您必须授予对 Amazon CloudFormation 所 Amazon CloudFormation 依赖的 Amazon 服务和操作的访问权限。要授予创建应用程序、部署组和部署配置的权限，请将以下策略添加到要使用的用户的权限集中 Amazon CloudFormation：

JSON

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

```
        "Effect": "Allow",
        "Action": [
          "cloudformation:*"
        ],
        "Resource": "*"
      }
    ]
}
```

有关策略的更多信息，请参阅以下主题：

- 要查看必须添加到将创建 Amazon EC2 实例的用户权限集中的策略，请参阅为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）。
- 有关向权限集添加策略的信息，请参阅《IAM 用户指南》中的创建权限集。
- 要了解如何限制用户只能使用一组有限的 CodeDeploy 操作和资源，请参阅Amazon 的托管（预定义）策略 CodeDeploy。

下表显示了 Amazon CloudFormation 模板可以代表您执行的操作，并包含指向您可以添加到 Amazon CloudFormation 模板中的 Amazon 资源类型及其属性类型的更多信息的链接。

Action	Amazon CloudFormation 参考	参考类型
创建 CodeDeploy 应用程序。	AWS::Code部署:: 应用程序	Amazon CloudFormation 资源
创建并指定要用于部署应用程序修订的部署组的详细信息。[1]	AWS::CodeDeploy::DeploymentGroup	Amazon CloudFormation 资源
创建一组 CodeDeploy 将在部署期间使用的部署规则、部署成功条件和部署失败条件。	AWS::CodeDeploy::DeploymentConfig	Amazon CloudFormation 资源
创建一个 Amazon EC2 实例。[2]	AWS:::EC2: 实例	Amazon CloudFormation 资源
使用 Amazon CloudFormation `AWS::CodeDeployBlueGreen` 转换和`AWS::CodeDeploy::BlueGreen` 挂钩	AWS::CodeDeployBlueGreen	`AWS::CodeDeployBlueGreen` 转换是一个由 Amazon CloudFormation 托管的宏

Action	Amazon CloudFormation 参考	参考类型
来管理堆栈更新、创建资源并转移 CodeDeploy 蓝/绿部署的流量。[3]	AWS::CodeDeploy::BlueGreen	`AWS::CodeDeploy::BlueGreen` 钩子结构为中的Hook资源 Amazon CloudFormation。该挂钩包含通过指向指定的 CodeDeploy生命周期事件挂钩来取代 CodeDeploy AppSpec 文件的参数。

[1] 如果您指定要作为部署组一部分部署的应用程序修订的版本，则在预配过程完成后将立即部署目标修订。有关模板配置的更多信息，请参阅《Amazon CloudFormation 用户指南》 GitHubLocation中的CodeDeploy DeploymentGroup 部署修订版 S3Location 和CodeDeploy DeploymentGroup 部署修订版。

[2] 我们提供模板，您可以使用这些 CodeDeploy 模板在支持的地区创建 Amazon EC2 实例。有关使用这些模板的更多信息，请参阅为 CodeDeploy 创建 Amazon EC2 实例（Amazon CloudFormation 模板）。

[3] 此 blue/green 部署配置仅支持 Amazon ECS 部署。有关通过 Amazon ECS 部署的 blue/green 部署配置的更多信息 Amazon CloudFormation，请参阅 部署的 Amazon CloudFormation blue/green 部署配置 (Amazon ECS)。有关通过 Amazon ECS blue/green 部署 Amazon CloudFormation 以及如何在中查看部署的更多信息 CodeDeploy，请参阅通过创建 Amazon ECS blue/green 部署 Amazon CloudFormation。

将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用

如果您使用 Amazon Virtual Private Cloud（Amazon VPC）托管 Amazon 资源，则可以在您的 VPC 和 CodeDeploy 之间建立私有连接。您可以使用此连接实现 CodeDeploy 与您的 VPC 上的资源的通信，而不用访问公共互联网。

Amazon VPC 是一项 Amazon 服务，可用来启动在虚拟网络中定义的 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。在使用 VPC 终端节点时，VPC 和 Amazon 服务之间的路由是由 Amazon 网络处理的，并且可以使用 IAM policy 控制对服务资源的访问。

要将 VPC 连接到 CodeDeploy，请为 CodeDeploy 定义一个接口 VPC 端点。接口端点是具有私有 IP 地址的弹性网络接口，用作发送到受支持的 Amazon 服务的通信的入口点。该端点提供了到 CodeDeploy 的可靠、可扩展的连接，无需互联网网关、网络地址转换（NAT）实例或 VPN 连接。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC。

接口 VPC 端点由 Amazon PrivateLink 提供支持，后者是一种 Amazon 技术，可将弹性网络接口与私有 IP 地址结合使用来支持 Amazon 服务之间的专有通信。有关更多信息，请参阅 Amazon PrivateLink。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的入门。

可用性

CodeDeploy 有两个 VPC 端点：一个用于 CodeDeploy 代理操作，另一个用于 CodeDeploy API 操作。下表显示了每个端点支持的 Amazon 区域。

有关 FIPS 端点的更多信息，请参阅 Amazon CodeDeploy 端点和配额。

区域名称	区域代码	代理端点	API 终端节点	符合 FIPS 标准的区域？
美国东部（弗吉尼亚州北部）	us-east-1	支持	是	是
美国东部（俄亥俄州）	us-east-2	支持	是	是
美国西部（加利福尼亚北部）	us-west-1	支持	是	是
美国西部（俄勒冈州）	us-west-2	支持	是	是
非洲（开普敦）	af-south-1	是	否	

区域名称	区域代码	代理端点	API 终端节点	符合 FIPS 标准的区域？
亚太地区（香港）	ap-east-1	支持	是	
亚太地区（海得拉巴）	ap-south-2	是	否	
亚太地区（雅加达）	ap-southeast-3	是	否	
亚太地区（墨尔本）	ap-southeast-4	是	否	
亚太地区（孟买）	ap-south-1	支持	是	
亚太地区（大阪）	ap-northeast-3	是	否	
亚太地区（首尔）	ap-northeast-2	支持	是	
亚太地区（新加坡）	ap-southeast-1	支持	是	
亚太地区（悉尼）	ap-southeast-2	支持	是	
亚太地区（东京）	ap-northeast-1	支持	是	
加拿大（中部）	ca-central-1	支持	是	
中国（北京）	cn-north-1	是	否	
中国（宁夏）	cn-northwest-1	否	否	

区域名称	区域代码	代理端点	API 终端节点	符合 FIPS 标准的区域？
欧洲地区（法兰克福）	eu-central-1	支持	是	
欧洲地区（爱尔兰）	eu-west-1	支持	是	
欧洲地区（伦敦）	eu-west-2	支持	是	
欧洲（米兰）	eu-south-1	是	否	
欧洲地区（巴黎）	eu-west-3	支持	是	
欧洲（西班牙）	eu-south-2	是	否	
欧洲地区（斯德哥尔摩）	eu-north-1	支持	是	
欧洲（苏黎世）	eu-central-2	是	否	
以色列（特拉维夫）	il-central-1	支持	是	
中东（巴林）	me-south-1	支持	是	
中东（阿联酋）	me-central-1	是	否	
South America（São Paulo）	sa-east-1	支持	是	
Amazon GovCloud（美国东部）	us-gov-east-1	否	否	是

区域名称	区域代码	代理端点	API 终端节点	符合 FIPS 标准的区域？
Amazon GovCloud（美国西部）	us-gov-west-1	否	否	是

为 CodeDeploy 创建 VPC 端点

要开始将您的 CodeDeploy 与 VPC 一起使用，请为 CodeDeploy 创建接口 VPC 端点。CodeDeploy 需要单独的端点来执行代理 Git 操作和 CodeDeploy API 操作。根据您的业务需求，您可能需要创建多个 VPC 终端节点。在为 CodeDeploy 创建 VPC 端点时，选择 Amazon 服务，然后在服务名称中，从以下选项中进行选择：

- com.amazonaws.***region***.codedeploy：如果要为 CodeDeploy API 操作创建 VPC 端点，请选择此选项。例如，如果您的用户使用 Amazon CLI、CodeDeploy API 或 Amazon 开发工具包来与 CodeDeploy 交互以进行 `CreateApplication`、`GetDeployment` 和 `ListDeploymentGroups` 等操作，请选择此选项。
- com.amazonaws.***region***.codedeploy-fips：如果要为 FIPS 端点的 CodeDeploy API 操作创建 VPC 端点，请选择此选项。
- com.amazonaws.***region***.codedeploy-commands-secure：如果要为 CodeDeploy 代理操作创建 VPC 端点，请选择此选项。您还需要在代理配置文件中将 `:enable_auth_policy:` 设置为 `true`，并附加所需的权限。有关更多信息，请参阅 配置 CodeDeploy 代理和 IAM 权限。

如果您使用的是 Lambda 或 ECS 部署，则只需为 com.amazonaws.***region***.codedeploy 创建 VPC 端点。使用 Amazon EC2 部署的客户需要 com.amazonaws.***region***.codedeploy 和 com.amazonaws.***region***.codedeploy-commands-secure 的 VPC 端点。

配置 CodeDeploy 代理和 IAM 权限

要将 Amazon VPC 终端节点与 CodeDeploy 配合使用，您必须在位于 EC2 或本地实例上的代理配置文件中将 `:enable_auth_policy:` 的值设置为 `true`。有关代理配置文件的更多信息，请参阅CodeDeploy 代理配置参考。

您还必须将以下 IAM 权限添加到您的 Amazon EC2 实例配置文件（如果您使用的是 Amazon EC2 实例）或 IAM 用户或角色（如果您使用的是本地实例）。

```
{
  "Statement": [
    {
      "Action": [
        "codedeploy-commands-secure:GetDeploymentSpecification",
        "codedeploy-commands-secure:PollHostCommand",
        "codedeploy-commands-secure:PutHostCommandAcknowledgement",
        "codedeploy-commands-secure:PutHostCommandComplete"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的创建接口终端节点。

CodeDeploy 资源包参考

CodeDeploy 依赖的许多文件都存储在公开可用的、 Amazon 特定于区域的 Amazon S3 存储桶中。这些文件包括 CodeDeploy 代理的安装文件、模板和示例应用程序文件。我们将此文件集合称为 CodeDeploy 资源工具包。

主题

- 各区域的资源工具包存储桶名称
- 资源工具包内容
- 显示资源工具包文件列表
- 下载资源工具包文件

各区域的资源工具包存储桶名称

下表列出了指南中某些步骤所需的*bucket-name* 替换件名称。这些是包含 CodeDeploy 资源包文件的 Amazon S3 存储桶的名称。

> **Note**
>
> 要访问亚太地区（香港）地区的 Amazon S3 存储桶，您必须在账户中 Amazon 启用该区域。有关更多信息，请参阅管理 Amazon 区域。

区域名称	*Bucket-name* 替换	区域标识符
中国（北京）	aws-codedeploy-cn-north-1	cn-north-1
中国（宁夏）	aws-codedeploy-cn-northwest-1	cn-northwest-1

资源工具包内容

下表列出了 CodeDeploy 资源包中的文件。

文件	说明
`LATEST_VERSION`	Amazon S EC2 ystems Manager 等更新机制用来确定 CodeDeploy 代理最新版本的文件。
`VERSION`	CodeDeploy 代理版本 1.1.0 中删除了自动更新机制，此文件已不再使用。 CodeDeploy 代理运行在实例上时用来更新自身的文件。
`codedeploy-agent.noarch.rpm`	亚马逊 Linux 和红帽企业 Linux (RHEL) 的 CodeDeploy 代理。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 `-1.0-0`）。
`codedeploy-agent_all.deb`	Ubuntu CodeDeploy 服务器的代理。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 `_1.0-0`）。
`codedeploy-agent.msi`	适用于 Windows 服务器的 CodeDeploy 代理。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 `-1.0-0`）。
`install`	一个可以用来更轻松地安装 CodeDeploy 代理的文件。
`CodeDeploy_SampleCF_Template.json`	该 Amazon CloudFormation 模板可用于启动一到三个运行亚马逊 Linux 或 Windows 服务器的

文件	说明
	亚马逊 EC2 实例。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 -1.0.0）。
CodeDeploy_SampleCF_ELB_Integration.json	可用于创建在 Apache Web 服务器上运行的负载平衡示例网站的 Amazon CloudFormation 模板。在创建应用程序的区域中，该应用程序配置为跨该区域的所有可用区。此模板创建了三个 Amazon EC2 实例和 IAM 实例配置文件，以授予这些实例访问 Amazon S3、Amazon A EC2 uto Scaling 和 ELB 中资源的权限。Amazon CloudFormation它还会创建负载均衡器和 CodeDeploy服务角色。
SampleApp_ELB_Integration.zip	您可以部署到注册到 ELB 负载均衡器的 Amazon EC2 实例的示例应用程序修订。
SampleApp_Linux.zip	您可以部署到运行 Amazon Linux 的亚马逊 EC2 实例、Ubuntu 服务器或 RHEL 实例的示例应用程序修订。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 -1.0）。
SampleApp_Windows.zip	您可以部署到 Windows Server 实例的示例应用程序修订。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本（例如 -1.0）。

显示资源工具包文件列表

要查看文件列表，请针对您的区域使用 aws s3 ls 命令。

> **Note**
> 各存储桶中的文件设计用于与对应区域中的资源配合使用。

- ```
  aws s3 ls --recursive s3://aws-codedeploy-cn-north-1  --region cn-north-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-cn-northwest-1  --region cn-northwest-1
  ```

下载资源工具包文件

要下载文件，请针对您的区域使用 aws s3 cp 命令。

> **Note**
>
> 请确保在靠近结尾的位置使用句点（`.`）。这会将文件下载到您的当前目录。

例如，以下命令从其中一个存储桶的 `/samples/latest/` 文件夹下载名为 `SampleApp_Linux.zip` 的单个文件：

- ```
  aws s3 cp s3://aws-codedeploy-cn-north-1/samples/latest/SampleApp_Linux.zip . --region cn-north-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-cn-northwest-1/samples/latest/SampleApp_Linux.zip . --region cn-northwest-1
  ```

要下载所有文件，请对您的区域使用以下命令之一：

- ```
  aws s3 cp --recursive s3://aws-codedeploy-cn-north-1 . --region cn-north-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-cn-northwest-1 . --region cn-northwest-1
  ```

CodeDeploy 配额

下表描述了中的配额 CodeDeploy。

> **Note**
>
> EC2/Ondlesd 就地部署以小时为单位运行的限制各不相同。对于在 2023 年 6 月之前创建的自定义部署配置，限制为 8 小时。对于在 2023 年 6 月或之后创建的自定义部署配置，限制为 12 小时。对于预定义的部署配置，限制为 12 小时。

Name	默认值	可调整	说明
Amazon Lambda 部署在数小时内即可运行	每个受支持的区域：50 个	否	Amazon Lambda 部署可以运行的最大小时数（第一次和最后一次流量转移之间的最长时间为 48 小时，再加上两个可能的生命周期挂钩各运行一小时）
每区域每账户关联的应用程序数	每个受支持的区域：1000 个	是	单个区域中与一个 Amazon 账户关联的最大应用程序数量
每个部署组的关联警报	每个受支持的区域：50 个	是	与部署组关联的最大警报数量
部署组中的 Auto Scaling 组数	每个受支持的区域：10 个	是	一个部署组中 Amazon A EC2 uto Scaling 组的最大数量
每个账户的并发部署数	每个受支持的区域：1,300 个	是	与一个 Amazon 账户关联的最大并发部署数。在 Amazon A EC2 uto Scaling 组中向扩展 Amazon EC2 实例的每次部署都算作与该 EC2 实例

Name	默认值	可调整	说明
			关联的每个应用程序的单个并发部署。
每个部署组的并发部署数	每个受支持的区域：1 个	否	针对部署组的并发部署的最大数量。此限制可以防止将同一应用程序并发部署到同一部署组。
每个账户的自定义部署配置	每个受支持的区域：50 个	否	与一个 Amazon 账户关联的自定义部署配置的最大数量
与单个应用程序关联的部署组	每个受支持的区域：1000 个	是	与单个应用程序关联的部署组的最大数量
EC2/本地 blue/green 部署可在数小时内运行	每个受支持的区域：109 个	否	EC2/Oldemy blue/green 部署可以运行的最大小时数（上述两个限制各为 48 小时，再加上 13 个可能的生命周期事件中每个事件各运行一小时）
EC2/本地就地部署可在数小时内运行	每个受支持的区域：12 个	否	EC2/Locide 就地部署可以运行的最大小时数
一个部署组中的事件通知触发器	每个受支持的区域：10 个	是	一个部署组中事件通知触发器的最大数量
GitHub 每个账户的连接令牌	每个受支持的区域：25 个	否	单个 Amazon 账户的最大 GitHub 连接令牌数量
在 EC2 /Un blue/green lide 部署期间，从部署完成到原始实例终止之间的小时数	每个受支持的区域：48 个	否	在 EC2/ blue/green Londest 部署期间，从完成部署到终止原始实例之间的最大小时数

Name	默认值	可调整	说明
从部署修订到在 EC2 /Un blue/green lide 部署期间流量转移到替换实例之间的小时数	每个受支持的区域：48 个	否	在 EC2 /Un blue/green lide 部署期间，从部署修订到流量转移到替换实例之间的最大小时数
每个部署的实例计数	us-east-1：2000 个 每个其他支持的区域：1000 个	是	单次部署中的最大实例数量
成功部署后， blue/green 部署可以等待几分钟，然后才能从原始部署中终止实例	每个受支持的区域：2800 个	否	成功部署后， blue/green 部署在从原始部署中终止实例之前可以等待的最大分钟数
在 Amazon Lambda 金丝雀部署或线性部署期间，第一次和最后一次流量转移之间的分钟数	每个受支持的区域：2880 个	否	在 Amazon Lambda 金丝雀部署或线性部署期间，第一次和最后一次流量转移之间的最大分钟数
如果生命周期事件未启动，部署失败前经过的分钟数：	每个受支持的区域：5 个	否	如果生命周期事件在 (1) 使用控制台或 Amazon CLI create-deployment 命令触发部署，或 (2) 上一个生命周期事件完成后仍未启动，则部署失败的最大分钟数。
可与 Amazon ECS 服务关联的部署组的数量	每个受支持的区域：1 个	否	可与 Amazon ECS 服务关联的部署组的最大数量

Name	默认值	可调整	说明
可以传递给 BatchGetOnPremises Instances API 操作的实例数量	每个受支持的区域：100 个	否	可以传递给 BatchGetOnPremisesInstances API 操作的最大实例数
每个账户正在进行的并发部署使用的实例数	us-east-1：3000 个 每个其他支持的区域：1000 个	是	正在进行并且与一个账户相关联的并发部署可以使用的实例数上限
Amazon ECS 部署期间流量路由上的侦听器数	每个受支持的区域：1 个	否	Amazon ECS 部署期间用于流量路由的侦听器的最大数量
未完成的部署生命周期事件失败之前经过的秒数	每个受支持的区域：3600 秒	否	未完成的部署生命周期事件失败之前经过的最大秒数
部署组名称的大小	每个受支持的区域：100 个	否	部署组名称中的最大字符数
标签键的大小	每个受支持的区域：128 个	否	标签键中的最大字符数
标签值的大小	每个受支持的区域：256 个	否	标签值中的最大字符数
部署组中的标签	每个受支持的区域：10 个	否	部署组中的最大标签数
在 Lamb Amazon da 部署期间可以以一个增量转移的流量	每个受支持的区域：99 个	否	在 Lambda 部署期间可以以一次增量转移的最 Amazon 大流量百分比

故障排除 CodeDeploy

使用本节中的主题来帮助解决您在使用时可能遇到的问题和错误 CodeDeploy。

> **Note**
>
> 可以通过查看部署过程中创建的日志文件来确定很多部署失败的原因。为简单起见，我们建议使用 Amazon Log CloudWatch s 来集中监控日志文件，而不是逐个实例查看它们。有关信息，请参阅Monitoring Deployments with Amazon CloudWatch Tools。

主题

- 一般问题排查
- 排除 EC2 /本地部署问题
- 排查 Amazon ECS 部署问题
- 对 Amazon Lambda 部署问题进行故障排除
- 排查部署组问题
- 排查实例问题
- 解决 GitHub 令牌问题
- 解决 Amazon A EC2 uto Scaling 问题
- Amazon CodeDeploy 的错误代码

一般问题排查

主题

- 一般问题排查核对清单
- CodeDeploy 只有某些 Amazon 区域支持部署资源
- 本指南中的步骤与 CodeDeploy 控制台不匹配
- 所需的 IAM 角色不可用
- 使用某些文本编辑器创建 AppSpec文件和 shell 脚本可能会导致部署失败
- 使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败

一般问题排查核对清单

您可以使用以下核对清单来排查失败部署问题。

1. 请参阅查看 CodeDeploy 部署详细信息 和View Instance Details以确定部署失败的原因。如果您无法确定原因，请查看此核对清单中的项。
2. 确保您已正确配置实例：
   - 实例启动时是否指定了 EC2 key pair？ 有关更多信息，请参阅 Amazon EC2 用户指南中的EC2密钥对。
   - 是否已将正确的 IAM 实例配置文件附加到实例？ 有关更多信息，请参阅配置要 EC2 使用的亚马逊实例 CodeDeploy和步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件。
   - 是否已标记实例？ 有关更多信息，请参阅 Amazon EC2 用户指南中的在控制台中使用标签。
   - 是否已在实例上安装、更新和运行 CodeDeploy 代理？ 有关更多信息，请参阅 管理 CodeDeploy 代理操作。要检查安装了哪个版本的代理，请参阅确定 CodeDeploy 代理的版本。
3. 检查应用程序和部署组设置：
   - 要检查应用程序设置，请参阅使用 CodeDeploy 查看应用程序详细信息。
   - 要检查部署组设置，请参阅使用 CodeDeploy 查看部署组详细信息。
4. 确认已正确配置应用程序修订：
   - 检查您的 AppSpec 文件格式。有关更多信息，请参阅将应用程序规范文件添加到 CodeDeploy 的修订和CodeDeploy AppSpec 文件参考。
   - 检查您的 Amazon S3 存储桶或 GitHub 存储库，确认您的应用程序修订位于预期位置。
   - 查看 CodeDeploy 应用程序修订版的详细信息，确保其注册正确。有关信息，请参阅使用 CodeDeploy 查看应用程序修订详细信息。
   - 如果您是从 Amazon S3 进行部署，请检查您的 Amazon S3 存储桶，以验证是否 CodeDeploy 已被授予下载应用程序修订版的权限。有关桶策略的信息，请参阅部署先决条件。
   - 如果您从中部署 GitHub，请检查您的 GitHub 存储库以确认是否 CodeDeploy已被授予下载应用程序修订版的权限。有关更多信息，请参阅使用创建部署 CodeDeploy和GitHub 对 CodeDeploy 中的应用程序进行的身份验证。
5. 确保已正确配置服务角色。有关信息，请参阅步骤 2：为创建服务角色 CodeDeploy。
6. 确认您已执行入门 CodeDeploy中的步骤，以便：
   - 已为用户预置适当的权限。
   - 安装或升级并配置 Amazon CLI。
   - 创建 IAM 实例配置文件和服务角色。

有关更多信息，请参阅 对 Amazon CodeDeploy进行身份和访问管理。

7. 确认您使用的是 Amazon CLI 版本 1.6.1 或更高版本。要检查已安装的版本，请调用 aws --version。

如果您仍无法排查失败部署的问题，请查看本主题中的其他问题。

CodeDeploy 只有某些 Amazon 区域支持部署资源

如果您在或 CodeDeploy 控制台中看不到或无法访问应用程序、部署组、实例或其他部署资源，请确保您引用的是区域和中终端节点中Amazon Web Services 一般参考列出的其中一个 Amazon 区域。Amazon CLI

EC2 CodeDeploy 部署中使用的实例和 Amazon A EC2 uto Scaling 组必须在其中一个 Amazon 区域启动和创建。

如果您使用的是 Amazon CLI，请从中运行aws configure命令 Amazon CLI。然后，您可以查看和设置您的默认 Amazon 区域。

如果您使用的是 CodeDeploy 控制台，请在导航栏的区域选择器中选择一个支持的 Amazon 区域。

> **Important**
>
> 要使用中国（北京）区域或中国（宁夏）区域中的服务，您必须拥有特定于这些区域的账户和凭证。其他 Amazon 区域的账户和凭证不适用于北京和宁夏区域，反之亦然。
> 有关中国区域某些资源的信息， CodeDeploy 例如资源套件存储桶名称 CodeDeploy 和代理安装过程，未包含在本版本的用户指南CodeDeploy 中。
> 有关更多信息：
>
> - CodeDeployin 中国（北京） Amazon 地区入门
> - CodeDeploy 中国地区用户指南（英文版 | 中文版）

本指南中的步骤与 CodeDeploy 控制台不匹配

本指南中的过程在撰写时体现了新的控制台设计。如果您使用的是旧版本的控制台，本指南中的许多概念和基本步骤仍然适用。要访问新控制台中的帮助，请选择信息图标。

所需的 IAM 角色不可用

如果您依赖作为 Amazon CloudFormation 堆栈一部分创建的 IAM 实例配置文件或服务角色，那么如果您删除堆栈，则所有 IAM 角色也会被删除。这可能就是 IAM 角色不再显示在 IAM 控制台中且 CodeDeploy 不再按预期工作的原因。要纠正此问题，您必须手动重新创建已删除的 IAM 角色。

使用某些文本编辑器创建 AppSpec文件和 shell 脚本可能会导致部署失败

某些文本编辑器会将非标准、非打印字符引入文件中。如果您使用文本编辑器创建或修改 AppSpec 文件或 shell 脚本文件以在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行，则依赖这些文件的任何部署都可能失败。在部署期间 CodeDeploy 使用这些文件时，这些字符的存在可能导致 hard-to-troubleshoot AppSpec 文件验证失败和脚本执行失败。

在 CodeDeploy 控制台中，在部署的事件详细信息页面上，选择查看日志。（或者你可以使用 Amazon CLI 来调用get-deployment-instance命令。）查找类似于 `invalid character`、`command not found` 或 `file not found` 的错误。

为了解决此问题，我们建议：

- 不要使用在 AppSpec 文件和 shell 脚本文件中引入非打印`^M`字符（如回车符（字符）的文本编辑器。
- 使用在 AppSpec 文件和 shell 脚本文件中显示非打印字符（例如回车符）的文本编辑器，这样您就可以查找和删除任何可能引入的字符。有关这些类型的文本编辑器的示例，请在 Internet 上搜索“显示回车的文本编辑器”。
- 使用在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的文本编辑器创建在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的 shell 脚本文件。有关这些类型的文本编辑器的示例，请在 Internet 上搜索“Linux Shell 脚本编辑器”。
- 如果您必须使用 Windows 或 macOS 中的文本编辑器来创建要在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的 Shell 脚本文件，请使用可将 Windows 或 macOS 格式的文本转换为 Unix 格式的程序或实用工具。有关这些程序和实用工具的示例，请在 Internet 上搜索“DOS 到 UNIX”或“Mac 到 UNIX”。请确保在目标操作系统上测试转换后的 Shell 脚本文件。

使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败

如果您在 Mac 上使用 Finder 图形用户界面 (GUI) 应用程序将 AppSpec 文件及相关文件和脚本捆绑 (zip) 到应用程序修订存档 (.zip) 文件中，则部署可能会失败。这是因为，Finder 将在 .zip 文件中创建一个中间 `__MACOSX` 文件夹并将组件文件放入其中。 CodeDeploy 找不到组件文件，因此部署失败。

要解决此问题，我们建议您使用调用 p ush 命令，该命令会将组件文件压缩到预期的结构中。 Amazon CLI 或者，您可以使用 Terminal（而不是 GUI）来压缩组件文件。Terminal 不创建中间 __MACOSX 文件夹。

排除 EC2 /本地部署问题

主题

- CodeDeploy 插件 CommandPoller缺少凭据错误
- 部署失败并显示消息 “ PKCS7 已签名邮件验证失败”
- 将相同的文件部署或重新部署到相同的实例位置失败，出现错误“The deployment failed because a specified file already exists at this location”
- 长文件路径会导致“没有这样的文件或目录”错误
- 长时间运行的进程可能会导致部署失败
- 在部署日志中未报告任何错误的情况下对失败的 AllowTraffic 生命周期事件进行故障排除
- 对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除
- 使用以下命令对失败的 DownloadBundle 部署生命周期事件进行故障排除 UnknownError：未打开供读取
- 排查所有生命周期事件跳过错误
- 默认情况下，Windows PowerShell 脚本无法使用 64 位版本 PowerShell 的 Windows

> **Note**
>
> 通过查看部署过程中创建的日志文件可以确定很多部署失败的原因。为简单起见，我们建议使用 Amazon Log CloudWatch s 来集中监控日志文件，而不是逐个实例查看它们。有关信息，请参阅在 CodeDeploy 日志控制台中查看 CloudWatch 日志。

> **Tip**
>
> 有关自动执行与 EC2 /Londest 部署相关的许多故障排除任务的运行手册，请参阅 S Amazon ystems Manager Automat ion 运行手册参考AWSSupport-TroubleshootCodeDeploy中。

CodeDeploy 插件 CommandPoller缺少凭据错误

如果您收到类似于 `InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile` 的错误，可能是由于下列原因之一导致：

- 您要部署的实例没有与之关联的 IAM 实例配置文件。
- 您的 IAM 实例配置文件没有配置正确的权限。

IAM 实例配置文件授予 CodeDeploy 代理与 Amazon S3 通信 CodeDeploy以及从 Amazon S3 下载您的修订版的权限。有关 EC2 实例，请参阅对 Amazon CodeDeploy进行身份和访问管理。对于本地实例，请参阅Working with On-Premises Instances。

部署失败并显示消息 “ PKCS7 已签名邮件验证失败”

此错误消息表示实例运行的 CodeDeploy 代理版本仅支持 SHA-1 哈希算法。2015 年 11 月发布的 CodeDeploy 代理版本 1.0.1.854 中引入了对 SHA-2 哈希算法的支持。自 2016 年 10 月 17 日起，如果安装的 CodeDeploy 代理版本低于 1.0.1.854，则部署将失败。有关更多信息，请参阅切换Amazon 到 SSL 证书的 SHA256 哈希算法、注意：停用早于 1.0.1.85 版本 CodeDeploy 的主机代理，以及。更新 CodeDeploy 代理

将相同的文件部署或重新部署到相同的实例位置失败，出现错误“The deployment failed because a specified file already exists at this location”

当 CodeDeploy 尝试将文件部署到实例，但指定的目标位置已存在同名文件时，部署到该实例可能会失败。您可能会收到错误消息 “部署失败，因为此位置已存在指定文件：*`location-name`*。” 这是因为，在每个部署期间， CodeDeploy 会先删除上一部署中的所有文件 (清除日志文件中列出了这些文件)。如果目标安装文件夹中存在未在此清理文件中列出的文件，则默认情况下， CodeDeploy 代理会将其解释为错误并导致部署失败。

> **Note**
>
> 在 Amazon Linux、RHEL 和 Ubuntu Server 实例上，清理文件位于 `/opt/codedeploy-agent/deployment-root/deployment-instructions/`。在 Windows Server 实例上，此位置为 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`。

避免此错误的最简单方式是，指定默认行为之外的选项以使部署失败。对于每个部署，您可以选择是使部署失败、覆盖清除文件中未列出的文件，还是保留实例上已有的文件。

覆盖选项在以下情况下很有用：您在上一个部署后手动将文件放置在实例上，然后将一个同名文件添加到下一个应用程序修订。

您可以为您在要成为下一部署的一部分的实例上放置的文件选择保留选项，而无需将这些文件添加到应用程序修订包。如果您的应用程序文件已存在于生产环境中，并且您想首次使用 CodeDeploy 进行部署，则保留选项也很有用。有关更多信息，请参阅创建 EC2 /本地计算平台部署（控制台）和现有内容的回滚行为。

排查 The deployment failed because a specified file already exists at this location 部署错误

如果您选择不指定选项来覆盖或保留 CodeDeploy 在目标部署位置检测到的内容（或者，如果您不指定任何部署选项来处理编程命令中的现有内容），则可以选择纠正错误。

以下信息仅在您选择不保留或覆盖内容的情况下适用。

如果您尝试重新部署具有相同名称和位置的文件，则如果指定应用程序名称和部署组的底层部署组 ID 与之前使用的相同，则重新部署更有可能成功。 CodeDeploy 使用底层部署组 ID 来识别要在重新部署之前删除的文件。

将新文件部署到实例上的相同位置或将相同的文件重新部署到实例上的相同位置可能会因以下原因而失败：

- 您为将相同修订重新部署到同一实例的操作指定不同的应用程序名称。重新部署失败，因为即使部署组名称相同，使用其他应用程序名称意味着将使用不同的基础部署组 ID。
- 您已删除并重新创建应用程序的部署组，然后尝试将同一修订重新部署到该部署组。重新部署失败，因为即使部署组名称相同，也会 CodeDeploy 引用不同的底层部署组 ID。
- 您在中删除了应用程序和部署组 CodeDeploy，然后创建了与您删除的应用程序和部署组同名的新应用程序和部署组。之后，您尝试重新将已部署到上一个部署组的修订部署到同名的新部署组。重新部署失败，因为尽管应用程序和部署组的名称相同，但 CodeDeploy 仍引用您删除的部署组的 ID。
- 您已将一个修订部署到一个部署组，然后将对另一个部署组的同一修订部署到相同的实例。第二次部署将失败，因为 CodeDeploy 引用不同的基础部署组 ID。
- 您已将一个修订部署到一个部署组，然后将对另一个部署组的其他修订部署到相同的实例。至少有一个文件具有相同名称且位于第二个部署组尝试部署的相同位置。第二次部署失败，因为在第二次部署开始之前 CodeDeploy 没有删除现有文件。两个部署都引用了不同的部署组 IDs。

- 您在中部署了修订版 CodeDeploy，但至少有一个名称相同且位于相同位置的文件。部署失败的原因是，默认情况下，在部署开始之前 CodeDeploy 不会删除现有文件。

要处理这些情况，请执行下列操作之一：

- 从文件之前部署到的位置和实例中删除文件，然后尝试重新部署。
- 在您的修订 AppSpec 文件中，在 ApplicationStop 或 BeforeInstall部署生命周期事件中，指定一个自定义脚本，以删除与您的修订将要安装的文件匹配的任何位置的文件。
- 将文件部署或重新部署到不属于之前的部署的位置或实例。
- 在删除应用程序或部署组之前，请部署一个修订版，该修订包含一个 AppSpec 文件，该文件没有指定要复制到实例的文件。对于部署，请指定应用程序名称和部署组名称，这些名称和部署组使用与要删除的应用程序和部署组 IDs 相同的底层应用程序和部署组。（您可以使用get-deployment-group命令来检索部署组 ID。）CodeDeploy使用底层部署组 ID 和 AppSpec 文件删除其在先前成功部署中安装的所有文件。

长文件路径会导致“没有这样的文件或目录”错误

对于到 Windows 实例的部署，如果您的 appspec.yml 文件的“files”部分的文件路径大于 260 个字符，则您可能会看到部署失败并显示类似以下内容的错误：

```
No such file or directory @ dir_s_mkdir - C:\your-long-file-path
```

之所以出现此错误，是因为默认情况下，Windows 不允许文件路径超过 260 个字符，详见 Microsoft 文档。

对于 CodeDeploy 代理版本 1.4.0 或更高版本，您可以通过两种方式启用长文件路径，具体取决于代理安装过程：

如果尚未安装 CodeDeploy 代理：

1. 在计划安装 CodeDeploy 代理的计算机上，使用以下命令启用 `LongPathsEnabled` Windows 注册表：

   ```
   New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
             -Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
   ```

2. 安装代 CodeDeploy 理。有关更多信息，请参阅 安装 CodeDeploy 代理。

如果已经安装了 CodeDeploy 代理：

1. 在 CodeDeploy 代理计算机上，使用以下命令启用 `LongPathsEnabled` Windows 注册表：

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
```

2. 重新启动 CodeDeploy 代理以使注册表项更改生效。要重新启动代理，请使用以下命令：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

长时间运行的进程可能会导致部署失败

对于部署到 Amazon Linux、Ubuntu Server 和 RHEL 实例，如果您的部署脚本启动了长时间运行的过程，则 CodeDeploy 可能会在部署生命周期事件中花费很长时间等待，然后部署失败。这是因为，在该进程运行的时间比该事件中的前台和后台进程预期的所需时间长的情况下， CodeDeploy 将停止部署并使其失败，即使该进程仍按预期运行也是如此。

例如，应用程序修订在其根目录下包含两个文件：`after-install.sh` 和 `sleep.sh`。其 AppSpec 文件包含以下指令：

```
version: 0.0
os: linux
files:
  - source: ./sleep.sh
    destination: /tmp
hooks:
  AfterInstall:
    - location: after-install.sh
      timeout: 60
```

该`after-install.sh`文件在 AfterInstall 应用程序生命周期事件期间运行。以下是其内容：

```
#!/bin/bash
/tmp/sleep.sh
```

`sleep.sh` 文件包含以下内容，它会使程序执行暂停 3 分钟（180 秒），并模拟某个长时间运行的进程：

```
#!/bin/bash
```

```
sleep 180
```

当`after-install.sh`呼叫时`sleep.sh`，`sleep.sh`启动并运行三分钟（180 秒），也就是说，比 CodeDeploy 预期停止运行的时间晚了两分钟`sleep.sh`（120 秒`after-install.sh`）。超时一分钟（60 秒）后，即使部署`sleep.sh`继续按预期运行，也会在 AfterInstall 应用程序生命周期事件中 CodeDeploy 停止部署并失败。将显示以下错误：

```
Script at specified location: after-install.sh failed to complete in 60
seconds.
```

仅在 & 中添加 `after-install.sh` 符号无法在后台运行 `sleep.sh`。

```
#!/bin/bash
# Do not do this.
/tmp/sleep.sh &
```

这样做可能会使部署在默认的一小时部署生命周期事件超时时间内处于待处理状态，之后部署会像以前一样在 AfterInstall 应用程序生命周期事件中 CodeDeploy 停止部署并失败。

在中`after-install.sh`，按`sleep.sh`如下方式调用 CodeDeploy ，这样可以在进程开始运行后继续：

```
#!/bin/bash
/tmp/sleep.sh > /dev/null 2> /dev/null < /dev/null &
```

在之前的调用中，`sleep.sh` 是要在后台开始运行的进程的名称，该进程将 stdout、stderr 和 stdin 重定向到 `/dev/null`。

在部署日志中未报告任何错误的情况下对失败的 AllowTraffic 生命周期事件进行故障排除

在某些情况下， blue/green 部署在 AllowTraffic 生命周期事件期间会失败，但部署日志并未指出失败的原因。

此故障通常是由于 ELB 中为用于管理部署组流量的 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer 的运行状况检查配置不正确。

要解决这一问题，请检查并更正负载均衡器的运行状况检查配置中的错误。

有关传统负载均衡器，请参阅经典负载均衡器用户指南和 Elastic Load Balancing API 参考版本 2012-06-01 ConfigureHealthCheck中的配置运行状况检查。

对于应用程序负载均衡器，请参阅《应用程序负载均衡器用户指南》中的目标组的运行状况检查。

对于网络负载均衡器，请参阅《网络负载均衡器用户指南》中的目标组的运行状况检查。

对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除

在部署期间， CodeDeploy 代理会运行上一次成功部署 AppSpec 的文件 AfterBlockTraffic 中为 ApplicationStop BeforeBlockTraffic、和指定的脚本。（所有其他脚本都从当前部署中的 AppSpec 文件运行。）如果这些脚本之一包含错误且未成功运行，则部署可能失败。

这些失败的可能原因包括：

- CodeDeploy 代理在正确位置找到了*deployment-group-id*_last_successful_install文件，但*deployment-group-id*_last_successful_install文件中列出的位置不存在。

  在 Amazon Linux、Ubuntu Server 和 RHEL 实例上，此文件必须存在于 `/opt/codedeploy-agent/deployment-root/deployment-instructions` 中。

  在 Windows Server 实例上，此文件必须存储在 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` 文件夹中。
- 在*deployment-group-id*_last_successful_install文件中列出的位置中，要么 AppSpec 文件无效，要么脚本无法成功运行。
- 这一脚本中包含无法更正的错误，所以永远无法成功运行。

使用 CodeDeploy 控制台调查在上述任何事件期间部署可能失败的原因。在部署的详细信息页上，选择查看事件。在实例的详细信息页面上，在ApplicationStopBeforeBlockTraffic、或AfterBlockTraffic行中，选择查看日志。或者使用 Amazon CLI 调用get-deployment-instance命令。

如果失败的原因是上次成功部署的脚本从未成功运行，请创建一个部署并指定应忽略 ApplicationStop BeforeBlockTraffic、和 AfterBlockTraffic 失败。有两种方式可执行此操作：

- 使用 CodeDeploy 控制台创建部署。在创建部署页面的ApplicationStop 生命周期事件失败下，选择实例上的此生命周期事件失败时不要使实例的部署失败。
- 使用调 Amazon CLI 用create-deployment命令并添加`--ignore-application-stop-failures`选项。

当您重新部署应用程序修订时，部署将继续，即使这三个生命周期事件中的任一事件失败也是如此。如果新的修订已包含针对这些生命周期事件的修复脚本，未来的部署无需应用此修复就能成功。

使用以下命令对失败的 DownloadBundle 部署生命周期事件进行故障排除 UnknownError：未打开供读取

如果您尝试从 Amazon S3 部署应用程序修订，但在部署生命周期事件期间 DownloadBundle 部署失败并`UnknownError: not opened for reading`显示错误：

- 发生内部 Amazon S3 服务错误。请重新部署应用程序修订。
- 您的实例上的 IAM EC2 实例配置文件无权访问 Amazon S3 中的应用程序修订。有关 Amazon S3 存储桶策略的信息，请参阅将修订推送 CodeDeploy 到 Amazon S3（EC2仅限本地部署）和部署先决条件。
- 您部署到的实例与一个 Amazon 区域（例如美国西部（俄勒冈））关联，但包含应用程序修订的 Amazon S3 存储桶与另一个 Amazon 区域（例如，美国东部（弗吉尼亚北部））关联。确保应用程序修订位于与实例相同 Amazon 区域关联的 Amazon S3 存储桶中。

在部署的事件详细信息页的下载服务包行中，选择查看日志。或者使用 Amazon CLI 调用get-deployment-instance命令。如果出现此错误，则输出中应有一个错误代码为 `UnknownError` 且错误消息为 `not opened for reading` 的错误。

要确定此错误的原因，请执行以下步骤：

1. 对至少一个实例启用线路日志记录，然后重新部署应用程序修订。
2. 查看线路日志记录文件以找到错误。此问题的常见错误消息包括短语“access denied”。
3. 在查看日志文件后，建议您禁用线路日志记录以减小日志文件的大小并减少将来可能会在实例上的输出中以纯文本格式出现的敏感信息量。

有关如何查找线路日志文件以及如何启用和禁用线路日志记录的信息，请参阅`:log_aws_wire:`CodeDeploy 代理配置参考中的。

排查所有生命周期事件跳过错误

如果跳过 EC2 或本地部署的所有生命周期事件，您可能会收到类似的`The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances`

`in your deployment group are experiencing problems. (Error code: HEALTH_CONSTRAINTS)`错误。这里介绍了一些可能的原因和解决方案：

- 该 CodeDeploy 代理可能未在实例上安装或运行。要确定 CodeDeploy 代理是否正在运行，请执行以下操作：
  - 对于 Amazon Linux RHEL 或 Ubuntu 服务器，请运行以下命令：

    ```
    systemctl status codedeploy-agent
    ```

  - 对于 Windows，请运行以下命令：

    ```
    powershell.exe -Command Get-Service -Name CodeDeployagent
    ```

  如果 CodeDeploy 代理未安装或未运行，请参见验证 CodeDeploy 代理是否正在运行。

  您的实例可能无法使用端口 443 访问 CodeDeploy 或 Amazon S3 公有终端节点。请尝试以下任一操作：

  - 将公有 IP 地址分配到实例并使用其路由表来允许 Internet 访问。确保与实例关联的安全组允许端口 443（HTTPS）上的出站访问。有关更多信息，请参阅 CodeDeploy 代理的通信协议和端口。
  - 如果是在私有子网中预配置了实例，请在路由表中使用 NAT 网关而不是 Internet 网关。有关更多信息，请参阅 NAT 网关。
- 的服务角色 CodeDeploy 可能没有所需的权限。要配置 CodeDeploy 服务角色，请参阅步骤 2：为创建服务角色 CodeDeploy。
- 如果您使用 HTTP 代理，请确保在 CodeDeploy 代理配置文件的`:proxy_uri:`设置中指定该代理。有关更多信息，请参阅 CodeDeploy 代理配置参考。
- 您部署实例的日期和时间签名可能与部署请求的日期和时间签名不匹配。
  在 CodeDeploy 代理日志文件`Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature expired`中查找类似的错误。如果您看到此错误，请按照疑难解答 “InvalidSignatureException — 签名已过期：[时间] 现在早于 [时间]” 部署错误中的步骤进行操作。有关更多信息，请参阅 查看 CodeDeploy EC2/本地部署的日志数据。
- CodeDeploy 代理可能会因为实例的内存或硬盘空间不足而停止运行。尝试通过更新 CodeDeploy 代理配置中的`max_revisions`设置来减少实例上的存档部署数量。如果您对某个 EC2 实例执行此操作，但问题仍然存在，请考虑使用更大的实例。例如，如果您的实例类型为 `t2.small`，请尝试使用 `t2.medium`。有关更多信息，请参阅 CodeDeploy 代理安装的文件、CodeDeploy 代理配置参考和实例类型。

- 您部署的实例可能没有附加 IAM 实例配置文件，或者可能附加了没有所需权限的 IAM 实例配置文件。
  - 如果 IAM 实例配置文件没有附加到您的实例，请创建具有所需权限的实例配置文件，然后附加该文件。
  - 如果 IAM 实例配置文件已经附加到您的实例，请确保它具有所需的权限。

  在您确认附加的实例配置文件配置有所需权限之后，重新启动实例。有关更多信息，请参阅步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件《亚马逊 EC2 用户指南》 EC2中的 Amazon 的 IAM 角色。

默认情况下，Windows PowerShell 脚本无法使用 64 位版本 PowerShell 的 Windows

如果在部署过程中运行的 Windows PowerShell 脚本依赖于 64 位功能（例如，因为它消耗的内存超过 32 位应用程序允许的内存或调用仅在 64 位版本中提供的库），则该脚本可能会崩溃或无法按预期运行。这是因为默认情况下， CodeDeploy 它使用 32 位版本的 Windows PowerShell 来运行作为应用程序修订版一部分的 Windows PowerShell 脚本。

在必须使用 64 位版本的 Windows PowerShell 运行的所有脚本的开头添加如下代码：

```
# Are you running in 32-bit mode?
#   (\SysWOW64\ = 32-bit mode)

if ($PSHOME -like "*SysWOW64*")
{
  Write-Warning "Restarting this script under 64-bit Windows PowerShell."

  # Restart this script under 64-bit Windows PowerShell.
  #   (\SysNative\ redirects to \System32\ for 64-bit mode)

  & (Join-Path ($PSHOME -replace "SysWOW64", "SysNative") powershell.exe) -File `
    (Join-Path $PSScriptRoot $MyInvocation.MyCommand) @args

  # Exit 32-bit script.

  Exit $LastExitCode
}

# Was restart successful?
Write-Warning "Hello from $PSHOME"
```

```
Write-Warning "  (\SysWOW64\ = 32-bit mode, \System32\ = 64-bit mode)"
Write-Warning "Original arguments (if any): $args"

# Your 64-bit script code follows here...
# ...
```

尽管此代码中的文件路径信息可能看起来违反直觉，但 32 位 Windows PowerShell 使用的路径如下：

```
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

64 位 Windows PowerShell 使用的路径如下所示：

```
c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

排查 Amazon ECS 部署问题

主题

- 等待替换任务集时出现超时
- 等待通知继续时出现超时
- IAM 角色没有足够的权限
- 等待状态回调时部署超时
- 由于一个或多个生命周期事件验证函数失败，部署失败
- 由于以下错误，ELB 无法更新：主任务集目标组必须位于监听器之后
- 使用 Auto Scaling 时，我的部署有时会失败
- 只有 ALB 支持渐进式流量路由，在 create/update 部署组时改用 AllAtOnce 流量路由
- 尽管我的部署成功了，但替换任务集未通过 ELB 运行状况检查，我的应用程序也已关闭
- 我能否将多个负载均衡器连接到一个部署组？
- 我能否在没有负载均衡器的情况下执行 CodeDeploy 蓝/绿部署？
- 在部署过程中，如何使用新信息更新我的 Amazon ECS 服务？

等待替换任务集时出现超时

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The deployment timed out while waiting for the replacement task set to
become healthy. This time out period is 60 minutes.
```

可能的原因：如果您的任务定义文件或其他与部署相关的文件中存在错误，则可能会出现此错误。例如，如果您的任务定义文件中的 `image` 字段中有拼写错误，Amazon ECS 将尝试提取错误的容器映像并持续失败，从而导致此错误。

可能的解决方法和后续步骤：

- 修复任务定义文件和其他文件中的排版错误和配置问题。
- 查看相关的 Amazon ECS 服务事件，找出替换任务无法正常运行的原因。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 事件。
- 请查看《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 疑难解答部分，了解与事件消息相关的错误。

等待通知继续时出现超时

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The deployment timed out while waiting for a notification to continue. This
time out period is n minutes.
```

可能的原因：如果您在创建部署组时在指定重新路由流量的时间字段中指定了等待时间，但是在等待时间结束之前部署未能完成，则可能会发生此错误。

可能的解决方法和后续步骤：

- 在您的部署组中，将指定重新路由流量的时间设置为更长的时间并重新部署。有关更多信息，请参阅为 Amazon ECS 部署创建部署组（控制台）。
- 在您的部署组中，将指定重新路由流量的时间设置更改为立即重新路由流量并重新部署。有关更多信息，请参阅 为 Amazon ECS 部署创建部署组（控制台）。
- 重新部署，然后在`--deployment-wait-type`选项设置为`READY_WAIT`的情况下运行`aws deploy continue-deployment` Amazon CLI 命令。确保在指定重新路由流量的时间中指定的时间结束之前 运行此命令。

IAM 角色没有足够的权限

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The IAM role role-arn does not give you permission to perform operations in
the following Amazon service: AWSLambda.
```

可能的原因：如果您在AppSpec 文件Hooks部分指定了 Lambda 函数，但未 CodeDeploy 授予 Lambda 服务的权限，则可能会发生此错误。

可能的解决方法：为 CodeDeploy 服务角色添加`lambda:InvokeFunction`权限。要添加此权限，请向该角色添加以下 Amazon托管策略之一：**`AWSCodeDeployRoleForECS`** 或 **`AWSCodeDeployRoleForECSLimited`**。有关这些策略以及如何将其添加到 CodeDeploy 服务角色的信息，请参阅步骤 2：为创建服务角色 CodeDeploy。

等待状态回调时部署超时

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The deployment timed out while waiting for a status callback. CodeDeploy
expects a status callback within one hour after a deployment hook is
invoked.
```

可能的原因：如果您在AppSpec 文件Hooks部分指定了 Lambda 函数，但是 Lambda Succeeded 函数无法调用必要的 `PutLifecycleEventHookExecutionStatus` API 来返回或状态，则可能会发生此错误。`Failed` CodeDeploy

可能的解决方法和后续步骤：

- 向您在文件中指定的 Lambda 函数使用的 Lambda 执行角色添加`codedeploy:putlifecycleEventHookExecutionStatus`权限。 AppSpec 此权限授予 Lambda 函数返回`Succeeded`或`Failed`状态的功能。 CodeDeploy有关 Lambda 执行角色的更多信息，请参阅《Amazon Lambda 用户指南》中的 Lambda 执行角色。
- 检查您的 Lambda 函数代码和执行日志，确保您的 Lambda 函数正在调用 CodeDeploy的 `PutLifecycleEventHookExecutionStatus` API，以告 CodeDeploy知生命周期验证测试是否还是。`Succeeded` `Failed`有关 `putlifecycleEventHookExecutionStatus` API 的信息，请参阅 Amazon CodeDeploy API 参考PutLifecycleEventHookExecutionStatus中的。有关 Lambda 执行日志的信息，请参阅访问亚马逊 CloudWatch 日志。 Amazon Lambda

由于一个或多个生命周期事件验证函数失败，部署失败

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The deployment failed because one or more of the lifecycle event validation
functions failed.
```

可能的原因：如果您在AppSpec 文件Hooks部分中指定了 Lambda 函数，但是 Lambda 函数在调用时返回Failed到 CodeDeploy ，则可能会发生此错误。PutLifecycleEventHookExecutionStatus此失败 CodeDeploy 表示生命周期验证测试失败。

下一步可能采取的措施：检查您的 Lambda 执行日志，了解验证测试代码失败的原因。有关 Lambda 执行日志的信息，请参阅访问亚马逊 CloudWatch 日志。 Amazon Lambda

由于以下错误，ELB 无法更新：主任务集目标组必须位于监听器之后

问题：在使用以下方式部署 Amazon ECS 应用程序时，您会看到以下错误消息 CodeDeploy：

```
The ELB could not be updated due to the following error: Primary taskset
target group must be behind listener
```

可能的原因：如果您配置了可选的测试侦听器，并且配置了错误的目标组，则可能会出现此错误。有关中测试侦听器的更多信息 CodeDeploy，请参见在开始 Amazon ECS 部署之前和在 Amazon ECS 部署过程中发生的事件。有关任务集的更多信息，请参阅TaskSet《亚马逊弹性容器服务 API 参考》和describe-task-set《Amazon CLI 命令参考》的 Amazon ECS 部分。

可能的解决方法：确保 ELB 的生产侦听器和测试侦听器都指向当前正在为您的工作负载提供服务的目标群组。有三个地方需要检查：

- 在 Amazon 中 EC2，在您的负载均衡器的监听器和规则设置中。有关更多信息，请参阅《应用程序负载均衡器用户指南》中的应用程序负载均衡器侦听器，或《网络负载均衡器用户指南》中的网络负载均衡器侦听器。
- 在 Amazon ECS 中，在您的集群中的服务的网络配置下。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的应用程序负载均衡器和网络负载均衡器注意事项。
- 在 CodeDeploy您的部署组设置中。有关更多信息，请参阅 为 Amazon ECS 部署创建部署组（控制台）。

使用 Auto Scaling 时，我的部署有时会失败

问题：您正在将 Auto Scaling CodeDeploy 与一起使用，但发现部署偶尔会失败。有关此问题症状的更多信息，请参阅《Amazon Elastic Container Service 开发者指南》中标题为 “对于配置为使用服务自动扩展的服务以及 blue/green 部署类型，在部署期间不会阻止自动扩展，但在某些情况下部署可能会失败。

可能的原因：如果 CodeDeploy 和 Auto Scaling 进程发生冲突，则可能会出现此问题。

可能的解决方法：在 CodeDeploy 部署期间使用 `RegisterScalableTarget` API（或相应的`register-scalable-target` Amazon CLI 命令）暂停和恢复 Auto Scaling 进程。有关更多信息，请参阅《Application Auto Scaling 用户指南》中的暂停和恢复 Application Auto Scaling 扩缩。

> **Note**
>
> CodeDeploy 无法`RegisterScaleableTarget`直接打电话。要使用此 API，您必须配置 CodeDeploy 为向亚马逊简单通知服务（或亚马逊 CloudWatch）发送通知或事件。然后，您必须将 Amazon SNS（或 CloudWatch）配置为调用 Lambda 函数，并配置 Lambda 函数以调用该 API。`RegisterScalableTarget`调用 `RegisterScalableTarget` API 时必须将 `SuspendedState` 参数设置为 `true` 以暂停 Auto Scaling 操作，然后设置为 `false` 以恢复这些操作。
>
> CodeDeploy 发送的通知或事件必须在部署开始（触发 Auto Scaling 暂停操作）或部署成功、失败或停止（触发 Auto Scaling 恢复操作）时发生。
>
> 有关如何配置 CodeDeploy 以生成 Amazon SNS 通知或 CloudWatch事件的信息，请参阅使用 Amazon CloudWatch 事件监控部署。和。Monitoring Deployments with Amazon SNS Event Notifications

只有 ALB 支持渐进式流量路由，在 create/update 部署组时改用 AllAtOnce 流量路由

问题：在中创建或更新部署组时，您会看到以下错误消息 CodeDeploy：

```
Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing
instead when you create/update Deployment group.
```

可能的原因：如果您使用的是网络负载均衡器并尝试使用除 `CodeDeployDefault.ECSAllAtOnce` 之外的预定义部署配置，则可能会发生此错误。

可能的修复方法：

- 将您的预定义部署配置更改为 `CodeDeployDefault.ECSAllAtOnce`。这是网络负载均衡器支持的唯一预定义部署配置。

  有关预定义部署配置的更多信息，请参阅Amazon ECS 计算平台的预定义部署配置。
- 将您的负载均衡器更改为应用程序负载均衡器。应用程序负载均衡器支持所有预定义的部署配置。有关创建应用程序负载均衡器的更多信息，请参阅为 A CodeDeploy mazon ECS 部署设置负载均衡器、目标组和侦听器。

尽管我的部署成功了，但替换任务集未通过 ELB 运行状况检查，我的应用程序也已关闭

问题：尽管 CodeDeploy 表明我的部署成功，但替换任务集未通过 ELB 的运行状况检查，并且我的应用程序已关闭。

可能的原因：如果您执行了 CodeDeploy all-at-once 部署，并且替换（绿色）任务集包含导致 ELB 运行状况检查失败的错误代码，则可能会出现此问题。使用 all-at-once部署配置，在流量转移到替换任务集之后（也就是说，`AllowTraffic`生命周期事件发生之后），负载均衡器的运行状况检查就会开始CodeDeploy对替换任务集运行。这就是为什么在流量转移之后，替换任务集的运行状况检查会失败，而在流量转移前却没有。有关 CodeDeploy 生成的生命周期事件的信息，请参阅在 Amazon ECS 部署过程中发生的事件。

可能的修复方法：

- 将部署配置从更改 all-at-once为灰色或线性。在灰度或线性配置中，在替换环境中 CodeDeploy 安装应用程序时，在流量转移之前（即*`Install`*生命周期事件期间和事件发生之前），负载均衡器的运行状况检查开始在替换任务集上*`AllowTraffic`*运行。通过允许在应用程序安装期间但在流量转移之前运行检查，可以在应用程序公开可用之前检测到错误的应用程序代码并导致部署失败。

  有关如何配置金丝雀部署或线性部署的信息，请参阅使用 CodeDeploy 更改部署组设置。

  有关在 Amazon ECS 部署期间运行的 CodeDeploy 生命周期事件的信息，请参阅在 Amazon ECS 部署过程中发生的事件。

  > **Note**
  >
  > 只有应用程序负载均衡器支持金丝雀和线性部署配置。

- 如果要保留 all-at-once部署配置，请设置测试侦听器并使用`BeforeAllowTraffic`生命周期挂钩检查替换任务集的运行状况。有关更多信息，请参阅 用于 Amazon ECS 部署的生命周期事件挂钩的列表。

我能否将多个负载均衡器连接到一个部署组？

不是。 如果您想使用多个应用程序负载均衡器或网络负载均衡器，请使用 Amazon ECS 滚动更新而不是 CodeDeploy 蓝/绿部署。有关滚动更新的更多信息，请参阅《Amazon Elastic Container

Service 开发人员指南》中的滚动更新。有关对 Amazon ECS 使用多个负载均衡器的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的向服务注册多个目标组。

我能否在没有负载均衡器的情况下执行 CodeDeploy 蓝/绿部署？

不，如果没有负载均衡器，则无法执行 CodeDeploy 蓝/绿部署。如果您无法使用负载均衡器，请改用 Amazon ECS 的滚动更新功能。有关 Amazon ECS 滚动更新的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的滚动更新。

在部署过程中，如何使用新信息更新我的 Amazon ECS 服务？

要在您的 Amazon ECS 服务进行部署时使用新参数对其进行 CodeDeploy 更新，请在 AppSpec 文件`resources`部分中指定该参数。仅支持少数 Amazon ECS 参数 CodeDeploy，例如任务定义文件和容器名称参数。有关 CodeDeploy 可以更新的 Amazon ECS 参数的完整列表，请参阅 AppSpec Amazon ECS 部署的 “资源” 部分。

> **Note**
>
> 如果您需要使用不支持的参数更新 Amazon ECS 服务 CodeDeploy，请完成以下任务：
>
> 1. 使用您要更新的参数调用 Amazon ECS 的 `UpdateService` API。有关可更新的参数的完整列表，请参阅UpdateService《亚马逊弹性容器服务 API 参考》。
> 2. 要将更改应用于任务，请创建新的 Amazon ECS blue/green 部署。有关更多信息，请参阅创建 Amazon ECS 计算平台部署（控制台）。

对 Amazon Lambda 部署问题进行故障排除

主题

- Amazon Lambda 手动停止未配置回滚的 Lambda 部署后，部署失败

Amazon Lambda 手动停止未配置回滚的 Lambda 部署后，部署失败

有时，在部署中指定的 Lambda 函数的别名可能引用函数的两个不同版本。结果是，后续部署 Lambda 函数的尝试会失败。当 Lambda 部署未配置回滚并被手动停止时，它可能会进入该状态。要继续，请使用 Amazon Lambda 控制台确保该功能未配置为在两个版本之间转移流量：

1. 登录 Amazon Web Services 管理控制台 并打开 Amazon Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 在左侧窗格中，选择函数。
3. 选择部署中的 Lambda 函数的 CodeDeploy 名称。
4. 从 “别名” 中，选择 CodeDeploy 部署中使用的别名，然后选择 “编辑”。
5. 从加权别名中选择 **none**。这可确保别名不配置为将流量百分比、权重转移到多个版本。记下在版本中选择的版本。
6. 选择保存。
7. 打开 CodeDeploy 控制台，尝试部署步骤 5 中下拉菜单中显示的版本。

排查部署组问题

标记作为部署组的一部分的实例不会自动将您的应用程序部署到新实例

CodeDeploy 不会自动将您的应用程序部署到新标记的实例。您必须在部署组中创建新的部署。

您可以使用自动部署 CodeDeploy 到 Amazon A EC2 uto Scaling 群组中的新 EC2 实例。有关更多信息，请参阅 CodeDeploy 与 Amazon A EC2 uto Scaling 集成。

排查实例问题

主题

- 必须正确设置标签
- Amazon CodeDeploy 必须在实例上安装并运行代理
- 如果实例在部署期间终止，在最多 1 小时内部署不会失败。
- 分析日志文件以调查针对实例的部署失败
- 如果 CodeDeploy 日志文件被意外删除，请创建一个新的日志文件
- 疑难解答 “InvalidSignatureException — 签名已过期：[时间] 现在早于 [时间]” 部署错误

必须正确设置标签

使用 list-deployment-instances 命令确认已正确标记用于部署的实例。如果输出中缺少 EC2 实例，请使用 EC2 控制台确认已在实例上设置标签。有关更多信息，请参阅 Amazon EC2 用户指南中的在控制台中使用标签。

> **Note**
>
> 如果您标记实例并立即使用 CodeDeploy 向其部署应用程序，则该实例可能不会包含在部署中。这是因为可能需要几分钟 CodeDeploy 才能读取标签。建议您在标记实例和尝试对实例进行部署之间等待至少 5 分钟。

Amazon CodeDeploy 必须在实例上安装并运行代理

要验证 CodeDeploy 代理是否已在实例上安装并正在运行，请参阅验证 CodeDeploy 代理是否正在运行。

要安装、卸载或重新安装 CodeDeploy 代理，请参阅安装 CodeDeploy 代理。

如果实例在部署期间终止，在最多 1 小时内部署不会失败。

CodeDeploy 为每个部署生命周期事件提供一小时的时间段，让其运行直至完成。这为长时间运行的脚本提供了足够的时间。

如果在生命周期事件进行期间（例如，如果实例终止或 CodeDeploy 代理关闭），则部署状态最多可能需要一个小时才能显示为 “失败”。即使脚本中指定的超时时段不到 1 小时，也会发生此情况。这是因为当实例终止时， CodeDeploy 代理会关闭，无法处理更多脚本。

不过，如果实例在生命周期事件之间或在第一个生命周期事件步骤开始之前终止，则超时将在 5 分钟后发生。

分析日志文件以调查针对实例的部署失败

如果部署中的实例具有 `Succeeded` 以外的任何状态，您可以查看部署日志文件数据来帮助确定问题。有关访问部署日志数据的信息，请参阅查看 CodeDeploy EC2/本地部署的日志数据。

如果 CodeDeploy 日志文件被意外删除，请创建一个新的日志文件

如果您不小心删除了实例上的部署日志文件，则 CodeDeploy 不会创建替换日志文件。要创建新的日志文件，请登录到实例，然后运行以下命令：

对于 Amazon Linux、Ubuntu Server 或 RHEL 实例，按此顺序运行这些命令（一次运行一个）：

```
systemctl stop codedeploy-agent
```

```
systemctl start codedeploy-agent
```

对于 Windows Server 实例：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

疑难解答 “InvalidSignatureException — 签名已过期：[时间] 现在早于 [时间]” 部署错误

CodeDeploy 需要精确的时间参考才能执行其操作。如果您的实例上的日期和时间设置不正确，则它们可能与您的部署请求的签名日期不匹配，您的部署请求会被 CodeDeploy 拒绝。

要避免与不正确的时间设置相关的部署失败，请参阅以下主题：

- 为 Linux 实例设置时间
- 为 Windows 实例设置时间

解决 GitHub 令牌问题

GitHub OAuth 令牌无效

CodeDeploy 2017 年 6 月之后创建的应用程序使用每个 Amazon 区域的 GitHub OAuth 令牌。使用绑定到特定 Amazon 区域的令牌可以让你更好地控制哪些 CodeDeploy 应用程序有权访问 GitHub 存储库。

如果您收到 GitHub 令牌错误，则可能是旧的令牌现在无效。

修复无效的 GitHub OAuth 令牌

1. 使用以下某种方法删除旧令牌：
   - 要使用 API 移除旧令牌，请使用 DeleteGitHubAccountToken。
   - 要使用 Amazon Command Line Interface移除旧令牌，请执行以下操作：
     a. 转到令牌所在的计算机。
     b. 确保在这台计算机上安装了。 Amazon CLI 有关说明，请参阅《Amazon Command Line Interface 用户指南》中的安装、更新和卸载 Amazon CLI。
     c. 在令牌所在的计算机上输入以下命令：

```
aws delete-git-hub-account-token
```

有关命令语法的详细信息，请参见 delete-git-hub-account-token。

2. 添加新 OAuth 令牌。有关更多信息，请参阅 将 CodeDeploy 与 GitHub 集成。

已超过最大代 GitHub OAuth币数量

创建 CodeDeploy 部署时，允许的最大 GitHub 令牌数为 10。如果您收到有关 GitHub OAuth 代币的错误消息，请确保您的代币数量不超过 10 个。如果您有 10 个以上的令牌，则最先创建的令牌无效。例如，如果您有 11 个令牌，则创建的第一个令牌无效。如果您有 12 个令牌，则最先创建的两个令牌无效。有关使用 CodeDeploy API 移除旧令牌的信息，请参阅 DeleteGitHubAccountToken。

解决 Amazon A EC2 uto Scaling 问题

主题

- Amazon A EC2 uto Scaling 常规疑难解答
- “CodeDeployRole 未授予您在以下 Amazon 服务中执行操作的权限： AmazonAutoScaling” 错误
- 在部署修订版之前，Amazon A EC2 uto Scaling 组中的实例会持续预配置和终止
- 终止或重启 Amazon A EC2 uto Scaling 实例可能会导致部署失败
- 避免将多个部署组与单个 Amazon A EC2 uto Scaling 组相关联
- EC2 Amazon A EC2 uto Scaling 组中的实例无法启动并收到 “心跳超时” 错误
- 不匹配的 Amazon A EC2 uto Scaling 生命周期挂钩可能会导致对 Amazon A EC2 uto Scaling 群组的自动部署停止或失败
- “由于未找到您的部署组的实例，部署失败”错误

Amazon A EC2 uto Scaling 常规疑难解答

由于以下原因， EC2 对 Amazon A EC2 uto Scaling 组中的实例的部署可能会失败：

- Amazon A EC2 uto Scaling 会持续启动和终止 EC2实例。如果 CodeDeploy 无法自动部署您的应用程序修订，Amazon A EC2 uto Scaling 会持续启动和终止 EC2 实例。

  解除 Amazon A EC2 uto Scaling 组与 CodeDeploy 部署组的关联或更改 Amazon A EC2 uto Scaling 组的配置，使所需的实例数量与当前的实例数量相匹配（从而防止 Amazon A EC2 uto Scaling 启动

更多 EC2 实例）。有关更多信息，请参阅使用 CodeDeploy 更改部署组设置或 Amazon A EC2 uto Scaling 的手动扩展。

- CodeDeploy 代理没有响应。如果在 EC2 实例启动或启动后立即运行的初始化脚本（例如，cloud-init 脚本）运行时间超过一小时，则可能无法安装 CodeDeploy 代理。 CodeDeploy CodeDeploy 代理响应待处理部署的超时时间为一小时。要解决此问题，请将您的初始化脚本移至 CodeDeploy 应用程序修订中。
- Amazon A EC2 uto Scaling 组中的 EC2 实例在部署期间会重新启动。如果在部署期间重启 EC2 实例，或者 CodeDeploy 代理在处理部署命令时关闭，则部署可能会失败。有关更多信息，请参阅 终止或重启 Amazon A EC2 uto Scaling 实例可能会导致部署失败。
- 将多个应用程序修订同时部署到 Amazon A EC2 uto Scaling 组中的同一个 EC2 实例。如果其中一个部署的脚本运行时间超过几分钟，则同时将多个应用程序修订部署到 Amazon A EC2 uto Scaling 组中的同一个 EC2 实例可能会失败。不要将多个应用程序修订部署到 Amazon A EC2 uto Scaling 组中的相同 EC2 实例。
- 对于作为 Amazon A EC2 uto Scaling 组的一部分启动的新 EC2 实例，部署失败。在这种情况下，在部署中运行脚本可能会阻止 Amazon A EC2 uto Scaling 组中的 EC2 实例启动。（Amazon A EC2 uto Scaling 组中的其他 EC2 实例可能看起来运行正常。）要解决此问题，请确保先完成所有其他脚本：
  - CodeDeploy 代理不包含在您的 AMI 中：如果您在启动新实例时使用`cfn-init`命令安装代理，请将代理安装脚本放在 Amazon CloudFormation 模板`cfn-init`部分的末尾。 CodeDeploy
  - CodeDeploy 代理包含在您的 AMI 中：配置 AMI，使代理在创建实例时`Stopped`处于状态，然后在脚本库中加入用于启动代理的`cfn-init`脚本作为最后一步。

“CodeDeployRole 未授予您在以下 Amazon 服务中执行操作的权限：AmazonAutoScaling” 错误

使用启动模板创建的 Auto Scaling 组的部署需要以下权限。这些是`AWSCodeDeployRole` Amazon 托管策略授予的权限之外的权限。

- `EC2:RunInstances`
- `EC2:CreateTags`
- `iam:PassRole`

如果您缺少这些权限，则可能就会收到此错误。有关更多信息教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组，请参阅《Amazon A EC2 uto Scaling 用户指南》中的 “为 Auto Scalin g 群组创建启动模板” 和 “权限”。

在部署修订版之前，Amazon A EC2 uto Scaling 组中的实例会持续预配置和终止

在某些情况下，错误可能会阻止成功部署到 Amazon A EC2 uto Scaling 组中新配置的实例。结果是没有运行正常的实例，部署失败。由于无法运行部署或无法成功完成部署，实例在创建之后很快即被终止。然后，Amazon A EC2 uto Scaling 组配置会导致配置另一批实例，以尝试满足最低运行状况主机要求。这批实例也会被终止，并不断进行这一循环。

可能的原因包括：

- Amazon A EC2 uto Scaling 群组运行状况检查失败。
- 应用程序修订中有错误

要解决这一问题，请遵循以下步骤：

1. 手动创建不属于 Amazon A EC2 uto Scaling 组的 EC2 实例。使用唯一的实例标签标记 EC2 实例。
2. 将这个新实例添加到受影响的部署组。
3. 将没有错误的新应用程序修订部署到部署组。

这会提示 Amazon A EC2 uto Scaling 组将应用程序修订版部署到 Amazon A EC2 uto Scaling 组中未来的实例。

> **Note**
>
> 确认部署成功后，请删除您创建的实例，以避免持续向您的 Amazon 账户收费。

终止或重启 Amazon A EC2 uto Scaling 实例可能会导致部署失败

如果通过 Amazon A EC2 uto Scaling 启动实例，然后该实例被终止或重启，则由于以下原因，对该实例的部署可能会失败： EC2

- 在进行中的部署期间，缩减事件或任何其他终止事件会导致实例与 Amazon A EC2 uto Scaling 组分离，然后终止。由于无法完成部署，因此它将失败。

- 实例已重启，但需要五分钟以上的时间才能启动实例。 CodeDeploy 将此视为超时。该服务将针对该实例的所有当前和将来部署失败。

解决此问题：

- 一般来说，请确保实例终止或重启之前完成所有部署。确保所有部署在实例启动或重启后开始。
- 如果您为 Amazon A EC2 uto Scaling 配置指定基于 Windows 服务器的亚马逊系统映像 (AMI)，并使用 EC2配置服务设置实例的计算机名称，则部署可能会失败。要修复此问题，请在 Windows Server 基础 AMI 中，在 “EC2 服务属性” 的 “常规” 选项卡上，清除 “设置计算机名称”。清除此复选框后，使用该 Windows Server 基础 AMI 启动的所有新 Windows Server Amazon A EC2 uto Scaling 实例都将禁用此行为。对于启用了此行为的 Windows Server Amazon A EC2 uto Scaling 实例，您无需清除此复选框。仅在重启实例后对其重新部署失败的部署。

避免将多个部署组与单个 Amazon A EC2 uto Scaling 组相关联

作为最佳实践，您应该只将一个部署组与每个 Amazon A EC2 uto Scaling 组关联起来。

这是因为，如果 Amazon A EC2 uto Scaling 扩展具有与多个部署组关联的挂钩的实例，它会同时发送所有挂钩的通知。这会导致针对每个实例的多个部署同时开始。当多个部署同时向 CodeDeploy 代理发送命令时，可能会达到生命周期事件与部署开始或上一个生命周期事件结束之间的五分钟超时时间。如果发生这种情况，即使部署过程按预期运行，部署也会失败。

> **Note**
>
> 生命周期事件中脚本的默认超时时间为 30 分钟。您可以在 AppSpec 文件中将超时时间更改为其他值。有关更多信息，请参阅 为 EC2/本地部署添加 AppSpec 文件。

如果尝试同时运行多个部署，则无法控制部署发生的顺序。

最后，如果部署到任何实例失败，Amazon A EC2 uto Scaling 会立即终止该实例。当第一个实例关闭时，正在运行的其他部署将开始失败。 CodeDeploy 由于 CodeDeploy 代理响应待处理部署的超时时间为一小时，因此每个实例最多可能需要 60 分钟才能超时。

有关 Amazon A EC2 uto Scaling 的更多信息，请参阅幕后花絮： CodeDeploy 和 Auto Scaling 集成。

EC2 Amazon A EC2 uto Scaling 组中的实例无法启动并收到 “心跳超时” 错误

Amazon A EC2 uto Scaling 组可能无法启动新 EC2 实例，从而生成类似于以下内容的消息：

```
Launching a new EC2 instance <instance-Id>. Status Reason: Instance failed
to complete user's Lifecycle Action: Lifecycle Action with token<token-Id>
was abandoned: Heartbeat Timeout.
```

此消息通常提示出现以下问题：

- 已达到与 Amazon 账户关联的最大并发部署数量。有关部署限制的更多信息，请参阅CodeDeploy 配额。
- Auto Scaling 组试图过快地启动太多 EC2 实例。对每个新实例RecordLifecycleActionHeartbeat或CompleteLifecycleAction针对每个新实例的 API 调用都已受到限制。
- 中的应用程序 CodeDeploy 在更新或删除其关联的部署组之前已被删除。

  当您删除应用程序或部署组时，会 CodeDeploy 尝试清理与之关联的任何 Amazon A EC2 uto Scaling 挂钩，但可能仍有一些挂钩。如果您运行命令来删除部署组，则剩余的挂钩将在输出中返回。但是，如果您运行命令来删除应用程序，则剩余的挂钩将不会出现在输出中。

  因此，作为最佳实践，在删除某个应用程序之前，您应删除与该应用程序关联的所有部署组。您可以使用命令输出来标识必须手动删除的生命周期挂钩。

如果您收到了“Heartbeat Timeout（检测信号超时）”错误消息，则可通过执行以下操作来确定剩余的生命周期挂钩是否为导致出现错误的原因并解决问题：

1. 请执行以下操作之一：

   - 调用delete-deployment-group命令删除与导致心跳超时的 Auto Scaling 组关联的部署组。
   - 使用非空的 Auto Scaling 组名称列表调用该update-deployment-group命令，以分离所有由托管的 Auto CodeDeploy Scaling 生命周期挂钩。

     例如，输入以下 Amazon CLI 命令：

     ```
     aws deploy update-deployment-group --application-name my-example-app
     --current-deployment-group-name my-deployment-group --auto-scaling-
     groups
     ```

再举一个例子，如果您将 CodeDeploy API 与 Java 一起使用，请调用`UpdateDeploymentGroup`并将其设置`autoScalingGroups`为`new ArrayList<String>()`。这会将 `autoScalingGroups` 设置为空列表并删除现有列表。不要使用 `null`，这是默认设置，因为这会将 `autoScalingGroups` 保持原样，这不是您想要的。

检查调用的输出。如果输出包含一个包含 Amazon A EC2 uto Scaling 生命周期挂钩列表的`hooksNotCleanedUp`结构，则还有剩余的生命周期挂钩。

2. 调用describe-lifecycle-hooks命令，指定与启动失败的 EC2 实例关联的 Amazon A EC2 uto Scaling 组的名称。在输出中，查找以下任何内容：
   - Amazon A EC2 uto Scaling 生命周期钩子名称与您在步骤 1 中确定的`hooksNotCleanedUp`结构相对应。
   - Amazon A EC2 uto Scaling 生命周期挂钩名称，其中包含与失败的 Auto Scaling 组关联的部署组的名称。
   - 可能导致 CodeDeploy 部署心跳超时的 Amazon A EC2 uto Scaling 生命周期挂钩名称。
3. 如果挂钩属于步骤 2 中列出的类别之一，请调用delete-lifecycle-hook命令将其删除。在调用中指定 Amazon A EC2 uto Scaling 组和生命周期挂钩。

   > **Important**
   > 仅删除导致问题的挂钩，如步骤 2 中所述。如果您删除可行的挂钩，您的部署可能会失败，或者 CodeDeploy 可能无法将应用程序修订部署到扩展实 EC2 例。

4. 使用所需的 Auto Scaling 组名调用update-deployment-group或create-deployment-group命令。CodeDeploy使用新 UUIDs功能重新安装 Auto Scaling 挂钩。

> **Note**
> 如果您将 Auto Scaling 组与 CodeDeploy 部署组分离，则任何正在进行的对 Auto Scaling 组的部署都可能失败，并且由 Auto Scaling 组扩展的新 EC2 实例将不会收到您的应用程序修订。CodeDeploy要让 Auto Scaling 再次使用 CodeDeploy，你需要将 Auto Scaling 组重新连接到部署组，然后调用一个新的组`CreateDeployment`来启动队列范围的部署。

不匹配的 Amazon A EC2 uto Scaling 生命周期挂钩可能会导致对 Amazon A EC2 uto Scaling 群组的自动部署停止或失败

Amazon A EC2 uto Scaling 并 CodeDeploy 使用生命周期挂钩来确定哪些应用程序修订在 Amazon A EC2 uto Scaling 群组中启动后应部署到哪些 EC2 实例。如果生命周期挂钩和有关这些挂钩的信息在 Amazon A EC2 uto Scaling 中不完全匹配，则自动部署可能会停止或失败 CodeDeploy。

如果对 Amazon A EC2 uto Scaling 组的部署失败，请查看 Amazon A EC2 uto Scaling 中的生命周期挂钩名称是否 CodeDeploy 匹配。如果不是，请使用这些 Amazon CLI 命令调用。

首先，获取 Amazon A EC2 uto Scaling 组和部署组的生命周期挂钩名称列表：

1. 调用describe-lifecycle-hooks命令，指定与中的部署组关联的 Amazon A EC2 uto Scaling 组的名称 CodeDeploy。在输出中，在 `LifecycleHooks` 列表中，记下每个 `LifecycleHookName` 值。
2. 调用get-deployment-group命令，指定与 Amazon A EC2 uto Scaling 组关联的部署组的名称。在输出中，在`autoScalingGroups`列表中，找到名称值与 Amazon A EC2 uto Scaling 组名称匹配的每个项目，然后记下相应的`hook`值。

现在比较两组生命周期挂钩的名称。如果它们完全匹配（字符对字符），则它不是问题。您可能需要尝试本节其他地方介绍的其他 Amazon A EC2 uto Scaling 疑难解答步骤。

但是，如果两组生命周期挂钩的名称未完全匹配（字符对字符），请执行以下操作：

1. 如果 describe-lifecycle-hooks 命令输出中包含 get-deployment-group 命令输出中未包含的生命周期挂钩名称，则执行以下操作：
   a. 对于describe-lifecycle-hooks命令输出中的每个生命周期挂钩名称，请调用该delete-lifecycle-hook命令。
   b. 调用update-deployment-group命令，指定原始 Amazon A EC2 uto Scaling 组的名称。CodeDeploy 在 Amazon A EC2 uto Scaling 组中创建新的替代生命周期挂钩，并将生命周期挂钩与部署组关联起来。现在，随着新实例添加到 Amazon A EC2 uto Scaling 组，自动部署应该会恢复。
2. 如果 get-deployment-group 命令输出中包含 describe-lifecycle-hooks 命令输出中未包含的生命周期挂钩名称，则执行以下操作：
   a. 调用该update-deployment-group命令，但不要指定原始 Amazon A EC2 uto Scaling 组的名称。

b. 再次调用该update-deployment-group命令，但这次要指定原始 Amazon A EC2 uto Scaling 组的名称。 CodeDeploy 在 Amazon A EC2 uto Scaling 组中重新创建缺失的生命周期挂钩。现在，随着新实例添加到 Amazon A EC2 uto Scaling 组，自动部署应该会恢复。

在您使两组生命周期挂钩名称完全匹配后，应重新部署应用程序修订，但只能在新实例添加到 Amazon A EC2 uto Scaling 组时将其部署到新实例。对于已经在 Amazon A EC2 uto Scaling 组中的实例，部署不会自动进行。

“由于未找到您的部署组的实例，部署失败”错误

如果您看到以下 CodeDeploy 错误，请阅读本节：

```
The deployment failed because no instances were found for your deployment
group. Check your deployment group settings to make sure the tags for your
EC2 instances or Auto Scaling groups correctly identify the instances you
want to deploy to, and then try again.
```

导致出现此错误的可能原因是：

1. 您的部署组设置包括不正确的 EC2 实例、本地实例或 Auto Scaling 组的标签。要修复此问题，请检查您的标签是否正确，然后重新部署您的应用程序。
2. 部署开始后，您的实例集已横向扩展。在这种情况下，您会看到实例集中处于 `InService` 状态的正常运行实例，但也会看到上面的错误。要修复此问题，请重新部署您的应用程序。
3. 您的 Auto Scaling 组不包含任何处于 `InService` 状态的实例。在这种情况下，当您尝试在队列范围内进行部署时，部署失败并显示上述错误消息，因为至少 CodeDeploy 需要一个实例才能处于该`InService`状态。可能没有实例处于 `InService` 状态的原因有很多。其中一些原因包括：
   - 您已将 Auto Scaling 组的大小计划（或手动配置）为 `0`。
   - Auto Scaling 检测到坏 EC2 实例（例如，这些 EC2 实例出现硬件故障），因此将其全部取消，没有任何实例`InService`处于状态。
   - 在从`0`到的扩展事件中`1`， CodeDeploy 部署了先前成功的修订（称为上次成功修订），该修订自上次部署以来已变得不健康。这导致在横向扩展实例上的部署失败，进而导致 Auto Scaling 取消该实例，使没有实例处于 `InService` 状态。

     如果您发现没有处于 `InService` 状态的实例，请按照以下过程To troubleshoot the error if there are no instances in the InService state中的说明对问题进行故障排除。

如果没有处于该 InService 状态的实例，则要解决错误

1. 在 Amazon EC2 控制台中，验证 “所需容量” 设置。如果为零，则将其设置为正数。等待实例状态变为 `InService`，这意味着部署成功。您已解决此问题，可以跳过此故障排除过程的其余步骤。有关设置所需容量设置的信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Auto Sc aling 组设置容量限制。
2. 如果 Auto Scaling 一直尝试启动新 EC2 实例以满足所需容量，但永远无法实现横向扩展，则通常是由于 Auto Scaling 生命周期挂钩失败所致。按如下方式排查此问题：
   a. 要查看哪个 Auto Scaling 生命周期挂钩事件失败，请参阅 Amazon A uto Scaling 用户指南中的验证 A EC2 uto Scaling 组的扩展活动。
   b. 如果失败的挂钩的名称为`CodeDeploy-managed-automatic-launch-deployment-hook-`*`DEPLOYMENT_GROUP_NAME`*，请转至 CodeDeploy，找到部署组，然后找到由 Auto Scaling 启动的失败部署。然后调查部署失败的原因。
   c. 如果您了解部署失败的原因（例如，发生了 CloudWatch 警报），并且可以在不更改修订版的情况下修复问题，那么现在就这样做。
   d. 如果经过调查，您确定上次成功修订 CodeDeploy的运行状况不佳，并且您的 Auto Scaling 组中没有正常运行的实例，则说明您处于部署死锁状态。要解决此问题，必须从 Auto Scaling 组中暂时删除 CodeDeploy生命周期挂钩，然后重新安装挂钩并重新部署新的（良好）CodeDeploy 修订版，从而修复错误的修订版。有关说明，请参阅：
      - To fix the deployment deadlock issue (CLI)
      - To fix the deployment deadlock issue (console)

修复部署死锁问题（CLI）

1. （可选）屏蔽导致 CodeDeploy 错误的 CI/CD 管道，这样在修复此问题时就不会发生意外部署。
2. 记下你当前的 Auto Scaling DesiredCapacity设置：

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name ASG_NAME
   ```

   在此过程结束时，您可能需要重新缩减到此数字。
3. 将 Auto Scaling DesiredCapacity设置设置为1。如果您的所需容量大于起始容量 1，则这是可选的。通过将其降至 1，可以缩短实例的配置和部署时间，从而加快故障排除速度。如果您的 Auto Scaling 的所需容量最初设置为 0，则必须将其增加到 1。这是强制性的。

aws 自动缩放 — set-desired-capacity auto-scaling-group-name *ASG_NAME* — 所需容量 1

> **Note**
>
> 此过程的其余步骤假设您已将设置DesiredCapacity为1。

此时，Auto Scaling 会尝试扩展到一个实例。然后，由于 CodeDeploy 添加的挂钩仍然存在，因此 CodeDeploy 会尝试部署；部署失败；Auto Scaling 取消实例；Auto Scaling 尝试重新启动实例以达到所需的容量，但再次失败。您正处于取消-重启的循环中。

4. 从部署组中取消注册 Auto Scaling 组：

> **Warning**
>
> 以下命令将启动一个没有软件的新 EC2 实例。在运行命令之前，请确保不运行任何软件的 Auto Scaling `InService` 实例是可以接受的。例如，确保与实例关联的负载均衡器在没有软件的情况下不会向该主机发送流量。

> **Important**
>
> 使用下面显示的 CodeDeploy 命令移除挂钩。请勿通过 Auto Scaling 服务移除挂钩，因为移除操作不会被识别 CodeDeploy。

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups
```

运行此命令后，将会发生以下情况：

a. CodeDeploy 从部署组中注销 Auto Scaling 组。

b. CodeDeploy 从 Auto Scaling 组中移除 Auto Scaling 生命周期挂钩。

c. 由于导致部署失败的挂钩已不复存在，Auto Scaling 会取消现有 EC2 实例，并立即启动一个新实例以扩展到所需容量。新实例应该很快就会进入 `InService` 状态。新实例不包括软件。

5. 等待 EC2 实例进入`InService`状态。要验证此，请使用以下命令：

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
ASG_NAME --query AutoScalingGroups[0].Instances[*].LifecycleState
```

6. 将挂钩添加回实 EC2 例：

> **Important**
>
> 使用下面显示的 CodeDeploy 命令添加挂钩。请勿使用 Auto Scaling 服务添加挂钩，因为添加的挂钩将无法被识别 CodeDeploy。

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups ASG_NAME
```

运行此命令后，将会发生以下情况：

a. CodeDeploy 将 Auto Scaling 生命周期挂钩重新安装到该实 EC2 例

b. CodeDeploy 将 Auto Scaling 组重新注册到部署组。

7. 使用您知道运行正常且想要使用的 Amazon S3 或 GitHub 修订版创建全队列部署。

例如，如果修订是名为 `my-revision-bucket` 的 Amazon S3 存储桶中的 .zip 文件，对象键为 `httpd_app.zip`，则输入以下命令：

```
aws deploy create-deployment --application-name APPLICATION_NAME
--deployment-group-name DEPLOYMENT_GROUP_NAME --
revision "revisionType=S3,s3Location={bucket=my-revision-
bucket,bundleType=zip,key=httpd_app.zip}"
```

由于 Auto Scaling 组中现在有一个 `InService` 实例，因此此部署应该可以正常工作，而且您不应再看到错误：由于未找到您的部署组的实例，部署失败。

8. 如果您之前缩减了 Auto Scaling 组的容量，则在部署成功后，将其扩展回原始容量：

```
aws autoscaling set-desired-capacity --auto-scaling-group-name ASG_NAME
--desired-capacity ORIGINAL_CAPACITY
```

修复部署死锁问题（控制台）

1. （可选）屏蔽导致 CodeDeploy 错误的 CI/CD 管道，这样在修复此问题时就不会发生意外部署。
2. 前往亚马逊 EC2 控制台，记下您的 Auto Scaling 所需容量设置。在此过程结束时，您可能需要重新缩减到此数字。有关查找此设置的信息，请参阅在 Auto Scaling 组中设置容量限制。
3. 将所需的 EC2 实例数设置为1：

   如果您的所需容量大于起始容量 1，则这是可选的。通过将其降至 1，可以缩短实例的配置和部署时间，从而加快故障排除速度。如果您的 Auto Scaling 的所需容量最初设置为 0，则必须将其增加到 1。这是强制性的。

   > **Note**
   > 此过程的其余步骤假设您已将所需容量设置为 1。

   a. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
   b. 选择适当的区域。
   c. 转到有问题的 Auto Scaling 组。
   d. 在组详细信息中，选择编辑。
   e. 将所需容量设置为 **1**。
   f. 选择更新。
4. 从部署组中取消注册 Auto Scaling 组：

   > **Warning**
   > 以下子步骤将启动一个没有软件的新 EC2 实例。在运行命令之前，请确保不运行任何软件的 Auto Scaling `InService` 实例是可以接受的。例如，确保与实例关联的负载均衡器在没有软件的情况下不会向该主机发送流量。

   a. 打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。
   b. 选择适当的区域。
   c. 在导航窗格中，选择 应用程序。
   d. 选择您的 CodeDeploy 应用程序的名称。

e. 选择您的 CodeDeploy 部署组的名称。

f. 选择编辑。

g. 在环境配置中，取消选择 Amazon A EC2 uto Scaling 群组。

> **Note**
>
> 如果未定义环境配置，则控制台不允许您保存配置。要绕过检查，请临时添加一个您知道不会解析为任何主机的标签 `EC2` 或 `On-premises`。要添加标签，请选择 Amazon EC2 实例或本地实例，然后添加标签密钥为**`EC2`**或**`On-premises`**。您可以将标签值留空。

h. 选择保存更改。

完成这些子步骤后，将会发生以下情况：

i. CodeDeploy 从部署组中注销 Auto Scaling 组。

ii. CodeDeploy 从 Auto Scaling 组中移除 Auto Scaling 生命周期挂钩。

iii. 由于导致部署失败的挂钩已不复存在，Auto Scaling 会取消现有 EC2 实例，并立即启动一个新实例以扩展到所需容量。新实例应该很快就会进入 `InService` 状态。新实例不包括软件。

5. 等待 EC2 实例进入`InService`状态。要验证其状态，请执行以下操作：

a. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。

b. 在导航窗格中，选择 Auto Scaling Groups。

c. 选择您的 Auto Scaling 组。

d. 在内容窗格中，选择实例管理选项卡。

e. 在 “实例” 下，确保生命周期列显示在实例InService旁边。

6. 使用与移除组相同的方法将 Auto Scaling 组重新注册到 CodeDeploy 部署组：

a. 打开 CodeDeploy 控制台，网址为https://console.aws.amazon.com/codedeploy/。

b. 选择适当的区域。

c. 在导航窗格中，选择 应用程序。

d. 选择您的 CodeDeploy 应用程序的名称。

e. 选择您的 CodeDeploy 部署组的名称。

f. 选择编辑。

g. 在环境配置中，选择 Amazon A EC2 uto Scaling 群组，然后从列表中选择您的 Auto Scaling 群组。

h. 在 Amazon EC2 实例或本地实例下，找到您添加的标签并将其删除。

i. 取消选中 Amazon EC2 实例或本地实例旁边的复选框。

j. 选择保存更改。

此配置将生命周期挂钩重新安装到 Auto Scaling 组中。

7. 使用您知道运行正常且想要使用的 Amazon S3 或 GitHub 修订版创建全队列部署。

例如，如果修订是名为 `my-revision-bucket` 的 Amazon S3 存储桶中的 .zip 文件，对象键为 `httpd_app.zip`，则执行以下操作：

a. 在 CodeDeploy 控制台的部署组页面中，选择创建部署。

b. 对于 Revision type（修订类型），选择 My application is stored in Amazon S3（我的应用程序存储在 Amazon S3 中）。

c. 在修订位置中，选择 `s3://my-revision-bucket/httpd_app.zip`。

d. 对于修订文件类型，选择 `.zip`。

e. 选择 Create deployment（创建部署）。

由于 Auto Scaling 组中现在有一个 `InService` 实例，因此此部署应该可以正常工作，而且您不应再看到错误：由于未找到您的部署组的实例，部署失败。

8. 如果您之前缩减了 Auto Scaling 组的容量，则在部署成功后，将其扩展回原始容量：

a. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。

b. 选择适当的区域。

c. 转到您的 Auto Scaling 组。

d. 在组详细信息中，选择编辑。

e. 将所需容量设置回其原始值。

f. 选择更新。

Amazon CodeDeploy 的错误代码

本主题提供有关 CodeDeploy 错误的参考信息。

错误代码	描述
AGENT_ISSUE	部署因 CodeDeploy 代理出现问题而失败。确保此代理已安装并在该部署组中的所有实例上运行。 了解更多： • 验证 CodeDeploy 代理是否正在运行 • 安装 CodeDeploy 代理 • 与 CodeDeploy 代理合作
AUTO_SCALING_IAM_ROLE_PERMISSIONS	与您的部署组关联的服务角色不具备在以下 Amazon 服务中执行操作所需的权限。 了解更多： • 步骤 2：为创建服务角色 CodeDeploy • 创建向 Amazon 服务委派权限的角色
HEALTH_CONSTRAINTS	总体部署因以下原因而失败：过多的独立实例部署遭失败、可供部署的正常实例太少或您的部署组中的一些实例遇到问题。 了解更多： • Instance Health • 排查实例问题 • 排除 EC2 /本地部署问题
HEALTH_CONSTRAINTS_INVALID	由于部署配置所定义的最小数目的正常运行的实例不可用，因此部署无法开始。您可通过更新部署配置来减少所需的正常运行的实例数或增加此部署组中的实例数。

错误代码	描述
	了解更多： • Instance Health • 使用适用于 CodeDeploy 的实例
IAM_ROLE_MISSING	由于不存在具有为部署组指定的服务角色名称的服务角色，因此部署失败。确保您使用的是正确的服务角色名称。 了解更多： • 步骤 2：为创建服务角色 CodeDeploy • 使用 CodeDeploy 更改部署组设置
IAM_ROLE_PERMISSIONS	CodeDeploy 没有代入角色所需的权限，或者您正在使用的 IAM 角色未为您提供在 Amazon 服务中执行操作的权限。 了解更多： • 步骤 1：设置 • 步骤 2：为创建服务角色 CodeDeploy • 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件

错误代码	描述
NO_INSTANCES	这可能是由于下列原因之一导致的。 • 对于 EC2/本地蓝绿部署，如果您使用 Amazon EC2 标签，可能会不正确地配置它们。在您的 CodeDeploy 部署组中，请确保它们包括在您的蓝色实例和绿色实例中。您可以使用 Amazon EC2 控制台确认您的实例已正确标记。 • 如果您使用 Amazon EC2 Auto Scaling 组，则 Auto Scaling 组可能容量不足。确保您的 Auto Scaling 组具有足够的容量用于部署。您可以使用 Amazon EC2 控制台，通过查看运行正常的实例数来查看您 Amazon EC2 Auto Scaling 组的容量。 • 对于 EC2/本地蓝绿部署，蓝色和绿色实例集的大小可能不同。请确保两个机群大小相同。 了解更多： • Tagging Instances for Deployments • 教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组 • 为蓝绿部署创建应用程序（控制台）

错误代码	描述
OVER_MAX_INSTANCES	由于为部署定向的实例数超出您的账户允许的数量，因此部署失败。要减少为此部署定向的实例数，请更新此部署组的标签设置或删除一些定向实例。或者，您可联系 Amazon Web Services 支持 以请求提高限制。 了解更多： • 使用 CodeDeploy 更改部署组设置 • CodeDeploy 配额 • 请求提高限制
THROTTLED	由于 IAM 角色发出的请求的数目超出 Amazon CodeDeploy 允许的数目，因此部署失败。尝试减少请求数。 了解更多： • 查询 API 请求速率
UNABLE_TO_SEND_ASG	由于部署组未使用其 Amazon EC2 Auto Scaling 组正确配置，因此部署失败。在 CodeDeploy 控制台中，从部署组中删除 Amazon EC2 Auto Scaling 组，然后重新添加此组。 了解更多： • 深入剖析：CodeDeploy 和 Auto Scaling 集成

相关主题

故障排除 CodeDeploy

CodeDeploy 资源

下列相关资源在您使用 CodeDeploy 的过程中会有所帮助。

参考指南和支持资源

- Amazon CodeDeploy API 参考 - 有关 CodeDeploy 操作和数据类型的描述、语法和用法示例，包括常见参数和错误代码。
- CodeDeploy 技术常见问题 - 客户提出的有关 CodeDeploy 的热门问题。
- Amazon Support 中心 – 用于创建和管理 Amazon Web Services 支持 案例的中心。还提供指向其他资源的链接，如论坛、技术常见问题、服务运行状况以及 Amazon Trusted Advisor。
- Amazon Support 计划 - 包含有关 Amazon Web Services 支持 计划的信息的主要网页。
- 联系我们 – 用于查询有关 Amazon 账单、账户、事件、滥用和其他问题的中央联系点。
- Amazon 网站条款 – 有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

样本

- GitHub 上的 CodeDeploy 示例 - CodeDeploy 的示例和模板方案。
- CodeDeploy Jenkins 插件 - 适用于 CodeDeploy 的 Jenkins 插件。
- CodeDeploy 代理 - CodeDeploy 代理的开源版本。

博客

- Amazon DevOps 博客 - 面向开发人员、系统管理员和架构师提供深入的见解。

Amazon 软件开发工具包和工具

以下 Amazon 开发工具包和工具支持使用 CodeDeploy 开发解决方案：

- 适用于 .NET 的 Amazon SDK
- 适用于 Java 的 Amazon SDK
- 适用于 JavaScript 的 Amazon SDK
- 适用于 PHP 的 Amazon SDK

- Amazon SDK for Python (Boto)
- 适用于 Ruby 的 Amazon SDK
- Amazon Toolkit for Eclipse - 第 1、2 和 3 部分。
- Amazon Tools for Windows PowerShell - 一组 Windows PowerShell cmdlet，用于在 PowerShell 环境中公开 适用于 .NET 的 Amazon SDK 的功能。
- Amazon Tools for PowerShell 中的 CodeDeploy cmdlet - 一组 Windows PowerShell cmdlet，用于在 PowerShell 环境中公开 CodeDeploy 的功能。
- Amazon Command Line Interface - 用于访问 Amazon 服务的一致的命令行语法。Amazon CLI 使用单一设置过程来启用对所有支持的服务的访问。
- Amazon 开发人员工具 - 指向开发人员工具和资源的链接，其中提供了文档、代码示例、发行说明和有助于您利用 CodeDeploy 和 Amazon 构建创新性应用程序的其他信息。

文档历史记录

下表描述了对此用户指南进行的重大更改，以阐明自上次发布 CodeDeploy 用户指南 以来的新增及增强功能。

- API 版本：2014-10-06

变更	说明	日期
CodeDeploy 代理 v1.8.0 版本	Amazon CodeDeploy 代理已更新至 1.8.0 版。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2025 年 7 月 31 日
CodeDeploy 更新了现有的 Amazon 托管策略	`AWSCodeDeployDeployerAccess` 已更新。有关更多信息，请参阅 Amazon 托管策略更新。	2024 年 12 月 16 日
CodeDeploy 更新了现有的 Amazon 托管策略	`AWSCodeDeployReadOnlyAccess` 已更新。有关更多信息，请参阅 Amazon 托管策略更新。	2024 年 12 月 16 日
CodeDeploy 更新了现有的 Amazon 托管策略	`AWSCodeDeployFullAccess` 已更新。有关更多信息，请参阅 Amazon 托管策略更新。	2024 年 12 月 16 日
CodeDeploy 代理 v1.7.1 版本	Amazon CodeDeploy 代理已更新至版本 1.7.1。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2024 年 11 月 14 日
更新了 Amazon S3 存储桶名称	更新了本指南中的 Amazon S3 存储桶示例，以使用由保留的 Amazon名称。	2024 年 6 月 17 日

添加了替代文本（替代文本）	更新了本指南中的所有图片，使其包含替代文本。屏幕阅读器可以阅读替代文本，使盲人用户更容易访问我们的文档。	2024 年 5 月 22 日
CodeDeploy 代理 v1.7.0 版本	Amazon CodeDeploy 代理已更新至 1.7.0 版。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2024 年 3 月 6 日
更改了命令	不再推荐使用这些`sudo service codedeploy-agent` *`status|start|stop`* 命令，因为它们不`systemd`用于 CodeDeploy 代理进程管理，这是最佳实践。为确保使用 `systemd`，请使用 `systemctl` 命令，如以下示例所示：`sytemctl start codedeploy-agent`。以下主题已使用`systemctl` 命令进行了更新：安装适用于 Amazon Linux 或 RHEL 的 CodeDeploy 代理、为 Ubuntu Server 安装代理、排除所有生命周期事件跳过的错误，以及在意外删除时创建新的 CodeDeploy 日志文件。 CodeDeploy	2024 年 1 月 12 日
添加了主题	在生命周期事件脚本中添加了管理 CodeDeploy 代理流程和引用文件主题。	2024 年 1 月 12 日
CodeDeploy 现在支持区域配置	使用区域配置信息更新了使用CodeDeploy主题创建部署配置。	2023 年 12 月 7 日

CodeDeploy 现在支持终止部署	添加了在 Auto Scaling 横向缩减事件期间启用终止部署主题来描述终止部署功能。还更新了 EC2/Unlide 部署的 “挂钩 AppSpec ” 部分、为就地部署（控制台）创建部署组，以及为 EC2本地部署（控制台）主题创建 blue/green 部署组以说明该功能。	2023 年 12 月 7 日
修复了 JSON 格式	修复了 AppSpec“资源” 部分（仅限 Amazon ECS 和 Amazon Lambda 部署）主题中 JSON 代码示例的格式问题。	2023 年 12 月 3 日
添加了故障排除主题	添加了排查 Amazon ECS 部署问题主题。	2023 年 10 月 24 日
更新了 AppSpec 文件名	更新了CodeDeploy AppSpec 文件引用，以表明必须为 EC2/AppSpec Londest 部署命名`appspec.yml` 该文件。	2023 年 10 月 5 日
CodeDeploy 现在支持多个负载均衡器	更新了为就地部署创建部署组（控制台）、为 EC2 /本地部署（控制台）创建部署组（控制台）和在 ELB for A CodeDeploy ma EC2 zon blue/green 部署中设置负载均衡器主题以表明支持多个负载均衡器。	2023 年 9 月 26 日

更新了“VPC 中的区域”主题	更新了 “ CodeDeploy与 Amazon Virtual Private Cloud 一起使用” 主题中的表格，以显示其他区域支持。具体而言，亚太地区（海得拉巴）、亚太地区（墨尔本）、欧洲地区（米兰）、欧洲（西班牙）和欧洲（苏黎世）区域已更新，以显示对代理端点的支持。	2023 年 9 月 22 日
更新了“资源工具包中的区域”主题	在 “按 Amazon 地区划分的资源包名称” 部分中添加了以下区域：亚太地区（大阪）、亚太地区（海得拉巴）、加拿大（中部）、欧洲（西班牙）、欧洲（苏黎世）、中东（阿联酋）。还更新了包含这些区域以及任何其他缺失区域的 IAM policy。	2023 年 9 月 22 日
缩短了代理安装和更新主题	缩短了 “在 Windows 服务器上安装 CodeDeploy 代理和更新 Windows 服务器上的 CodeDeploy 代理” 主题。删除了多余的 Amazon S3 存储桶 URLs 和 Amazon S3 复制命令。	2023 年 9 月 21 日
添加了亚太地区（雅加达）区域	向各区域的资源工具包存储桶名称添加了亚太地区（雅加达）。	2023 年 9 月 21 日
CodeDeploy 更新了现有的 Amazon 托管策略	AWSCodeDeployRole 托管策略已更新。有关更多信息，请参阅Amazon 对 Amazon 托管策略进行的更新。	2023 年 8 月 16 日

添加了限制	为限制主题添加了CodeDeploy限制。限制是与部署组关联的警报的最大数量。	2023 年 8 月 15 日
修复了与负载均衡器相关的步骤	修复了为 EC2/Un blue/green lide 部署创建部署组（控制台）中的说明。负载均衡器步骤现在标记为可选。	2023 年 8 月 3 日
澄清了 Amazon ECS 主题中的措辞	澄清了教程：将应用程序部署到 Amazon ECS 中的措辞。现在的措辞表示您正在部署应用程序。之前的措辞表明您正在部署 Amazon ECS 服务。	2023 年 8 月 3 日
CodeDeploy 现已在以色列（特拉维夫）地区上市	CodeDeploy 现已在以色列（特拉维夫）地区 (il-central-1) 上市。更新了多个主题，包括包含设置 CodeDeploy 代理说明的主题，以反映这个新区域的可用性。	2023 年 7 月 31 日
主题更新	更新了 “解决 EC2 /本地部署问题” 主题，提供了有关使用运行手册自动执行故障排除任务的提示。	2023 年 7 月 7 日
主题更新	更新了 Amazon ECS 部署主题的 “资源AppSpec ” 部分，其中包含有关任务定义 ARN 的更多信息。	2023 年 7 月 7 日
主题更新	使用故障排除信息更新了 “步骤 1：在本地实例 Amazon CLI 上安装和配置” 主题。	2023 年 7 月 7 日

主题更新	使用有关 Amazon ECS blue/green 部署的信息更新了跨服务混乱的副手 Amazon CloudFormation预防主题。	2023 年 7 月 6 日
主题更新	跨服务混乱的副手预防主题已更新，其中包含有关通过 Amazon CloudFormation的 Amazon ECS blue/green 部署的信息。	2023 年 7 月 6 日
主题更新	更新了 EC2/本地计算平台主题的预定义部署配置。添加了有关 `CodeDeployDefault.HalfAtATime` 预定义部署配置在自动扩缩组中的行为的说明。	2023 年 6 月 29 日
主题更新	更新了 Amazon CodeDeploy主题中的基础设施安全，以指明传输层安全 (TLS) 协议的新最低版本和推荐版本。	2023 年 6 月 28 日
限制更新	更改了以下限制：“ EC2/本地部署可以运行的最大小时数”。有关更多信息，请参阅限制	2023 年 6 月 27 日
主题更新	“步骤 3：限制 CodeDeploy用户的权限” 主题已更新，其中包含详细说明。	2023 年 5 月 31 日
CodeDeploy 更新了现有的 Amazon 托管策略	AWSCodeDeployFullAccess 托管策略已更新。有关更多信息，请参阅 Amazon 对 Amazon 托管策略的更新。	2023 年 5 月 16 日

CodeDeploy 代理 v1.6.0 版本	Amazon CodeDeploy 代理已更新至 1.6.0 版。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2023 年 3 月 30 日
CodeDeploy 代理 v1.5.0 版本	Amazon CodeDeploy 代理已更新至版本 1.5.0。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2023 年 3 月 3 日
Amazon ECS 计算平台更新	亚太地区（雅加达）区域现在支持在 Amazon ECS 计算平台上部署。	2023 年 2 月 8 日
CodeDeploy 更新了现有的 Amazon 托管策略	AWSCodeDeployRole 托管策略已更新。有关更多信息，请参阅Amazon 对 Amazon 托管策略进行的更新。	2023 年 2 月 3 日
主题更新	“使用 CodeDeploy Amazon Virtual Private Cloud” 主题已更新，增加了新的和已更改的 Amazon 区域。	2023 年 2 月 2 日
主题更新	CodeDeploy 现已在亚太地区（墨尔本）区域（ap-southeast-4）推出。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这些新区域的可用性。	2023 年 1 月 26 日
安全最佳实践更新	“入门 CodeDeploy” 部分和其他一些部分已更新，以符合 Amazon 安全最佳实践。	2023 年 1 月 23 日

CodeDeploy 代理 v1.4.1 版本	Amazon CodeDeploy 代理已更新至 1.4.1 版。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2022 年 12 月 6 日
添加了故障排除主题	添加了有关如何解决因与 Windows CodeDeploy 代理一起使用长文件路径而导致的错误的主题。有关更多信息，请参阅长文件路径会导致“没有这样的文件或目录”错误。	2022 年 12 月 6 日
更改了限制	更改了以下限制：“与 Amazon 账户关联的自定义部署配置的最大数量”。现在的限制是 200。有关限制的更多信息，请参阅限制主题。	2022 年 9 月 7 日
CodeDeploy 代理 v1.4.0 版本	Amazon CodeDeploy 代理已更新至版本 1.4.0。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2022 年 8 月 31 日
修复了一些限制。	修复了以下限制：“与 Amazon 账户关联的最大并发部署数量”现在为 1000。“单次部署中的最大实例数”现在为 1000。“正在进行并且与一个账户相关联的并发部署可以使用的最大实例数”现在为 1000。'与一个 Amazon 账户关联的自定义部署配置的最大数量 '现在为 100。有关限制的更多信息，请参阅限制主题。	2022 年 8 月 8 日

添加了一个显示每个区域支持的 CodeDeploy 终端节点的表。	有关更多信息，请参阅使用亚马逊 Virtual CodeDeploy Private Cloud。	2022 年 4 月 20 日
为 Amazon ECS blue/green 部署增加了新的限制。	在 Amazon ECS 蓝绿部署过程中，从部署修订到流量转移到替换环境之间的最长小时数现在为 120 小时。有关更多信息，请参阅限制主题中的部署。	2022 年 4 月 12 日
添加了有关如何防止混淆代理人问题的主题	有关更多信息，请参阅适用于 Amazon CodeDeploy的 Amazon Identity and Access Management。	2022 年 3 月 14 日
CodeDeploy 更新了现有的 Amazon 托管策略	`AmazonEC2RoleforAWSCodeDeployLimited` 角色已更新。有关更多信息，请参阅 Amazon 托管策略更新。	2021 年 11 月 22 日
CodeDeploy 更新了现有的 Amazon 托管策略	`Amazon CodeDeployRole` 已更新。有关更多信息，请参阅 Amazon 托管策略更新。	2021 年 5 月 18 日
CodeDeploy 代理 v1.3.2 版本	Amazon CodeDeploy 代理已更新至版本 1.3.2。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2021 年 5 月 6 日
CodeDeploy 支持更新过时的 Amazon EC2 实例	CodeDeploy 现在支持自动更新过时的 Amazon EC2 实例。有关更多信息，请参阅为部署组配置高级选项。	2021 年 2 月 23 日

CodeDeploy 代理 v1.3.1 版本	Amazon CodeDeploy 代理已更新至版本 1.3.1。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2020 年 12 月 22 日
CodeDeploy 代理 v1.3.0 版本	Amazon CodeDeploy 代理已更新至版本 1.3.0。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2020 年 11 月 10 日
CodeDeploy 代理 v1.2.1 版本	Amazon CodeDeploy 代理已更新至版本 1.2.1。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2020 年 9 月 23 日
CodeDeploy 支持由以下设备提供支持的亚马逊 VPC 终端节点 Amazon PrivateLink	如果您使用亚马逊虚拟私有云（Amazon VPC）托管 Amazon 资源，则可以在您的 VPC 和之间建立私有连接 CodeDeploy。您可以使用此连接实现 CodeDeploy 与您的 VPC 上的资源的通信而不用访问公共 Internet。有关更多信息，请参阅使用亚马逊 Virtual CodeDeploy Private Cloud。	2020 年 8 月 11 日
更新了 CodeDeploy 服务限制	将每个账户的应用程序数量和每个应用程序的部署组数量限制更新为 1000。有关 CodeDeploy 服务限制的更多信息，请参阅CodeDeploy 限制。	2020 年 8 月 6 日
CodeDeploy 代理 v1.1.2 版本	Amazon CodeDeploy 代理已更新至 1.1.2 版。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2020 年 8 月 4 日

CodeDeploy 代理 1.1.0 发布并与亚马逊 S EC2 ystems Manager 集成	CodeDeploy 代理版本1.1.0 现已推出，有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。现在，您可以使用 Amazon EC2 Systems Manager 自动管理您的亚马逊 EC2 或本地实例上的 CodeDeploy 代理安装和更新。有关更多信息，请参阅使用 Amazon Syst EC2 ems Manager 安装 CodeDeploy代理。	2020 年 6 月 30 日
CodeDeploy 支持使用管理 Amazon ECS blue/green 部署 Amazon CloudFormation	现在，您可以通过使用 Amazon CloudFormation 来管理 Amazon ECS blue/green 部署 CodeDeploy。通过定义蓝绿资源并指定要在 Amazon CloudFormation中使用的流量路由和稳定设置来生成部署。有关更多信息，请参阅通过创建 Amazon ECS 蓝/绿部署。Amazon CloudFormation	2020 年 5 月 19 日
CodeDeploy 支持 Amazon ECS 蓝/绿部署的加权流量转移	CodeDeploy 现在支持 Amazon ECS 蓝/绿部署的加权流量转移。您选择或创建部署配置，用于指定部署中的流量转移间隔数以及每个间隔转移的流量百分比。以下主题已更新以反映此更改：Amazon ECS 计算平台上的部署配置。	2020 年 2 月 6 日
更新了安全、身份验证和访问控制主题	的安全、身份验证和访问控制信息 CodeDeploy 已整理到新的安全章节中。关更多信息，请参阅安全性。	2019 年 11 月 26 日

CodeDeploy 支持通知规则	现在，您可以使用通知规则向用户通知部署中的重要更改。有关更多信息，请参阅创建通知规则。	2019 年 11 月 5 日
更新的主题	CodeDeploy 现已在亚太地区（香港）区域 (ap-east-1) 区域推出。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。您必须显式启用对此区域的访问。有关更多信息，请参阅管理 Amazon区域。	2019 年 4 月 25 日
更新的主题	Amazon CodeDeploy 现在支持在 Amazon ECS 服务中 blue/green 部署容器化应用程序。使用新 Amazon ECS 计算平台的 CodeDeploy 应用程序会将容器化应用程序部署到同一 Amazon ECS 服务中的新替换任务集中。为了反映这一变化，已经添加和更新了多个主题，包括Amazon CodeDeploy 计算平台概述、Amazon ECS 计算平台上的部署、Amazon ECS 部署AppSpec 的文件结构以及为 Amazon ECS 服务部署创建应用程序（控制台）。	2018 年 11 月 27 日
已更新的 CodeDeploy 代理	Amazon CodeDeploy 代理已更新至版本 1.0.1.1597。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2018 年 11 月 15 日

更新了控制台	本指南中的过程已更新，以匹配最新设计的 CodeDeploy 控制台。	2018 年 10 月 30 日
该 CodeDeploy 代理支持的最低新版本	现在，该 Amazon CodeDeploy 代理支持的最低版本为 1.7.x。有关更多信息，请参阅 CodeDeploy 代理的版本历史记录。	2018 年 8 月 7 日

早期更新

下表描述了 2018 年 6 月之前每次发布 Amazon CodeDeploy 用户指南 时进行的重要更改。

更改	描述	更改日期
主题更新	CodeDeploy 现已在欧洲（巴黎）区域 (eu-west-3) 区域推出。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。	2017 年 12 月 19 日
更新的主题	CodeDeploy 现已在中国（宁夏）区域推出。 要使用中国（北京）区域或中国（宁夏）区域中的服务，您必须拥有特定于这些区域的账户和凭证。其他 Amazon 区域的账户和凭证不适用于北京和宁夏区域，反之亦然。 有关中国区域某些资源的信息， CodeDeploy 例如资源套件存储桶名称 CodeDeploy 和代理安装过程，未包含在本版本的用户指南CodeDeploy 中。 有关更多信息： • CodeDeployin 中国（北京） Amazon 地区入门 • CodeDeploy 中国地区用户指南（英文版 | 中文版）	2017 年 12 月 11 日
更新的主题	CodeDeploy 现在支持部署 Lambda 函数。 Amazon Lambda 部署可将传入流量从现有 Lambda 函数转移到更新的 Lambda 函数版本。您可以选择或创建部署配置来指	2017 年 11 月 28 日

更改	描述	更改日期
	定部署中的流量转移间隔数以及每个间隔内要转移的流量百分比。 Amazon Lambda Amazon 无服务器应用程序模型 (S Amazon AM) 支持部署，因此您可以使用 S Amazon AM 部署首选项来管理部署期间流量的转移方式。 Amazon Lambda 为反映这一变动，新增和更新了若干个主题，其中包括 CodeDeploy 计算平台概述、Amazon Lambda 计算平台上的部署、创建 Amazon Lambda 计算平台部署（控制台）、 为 Amazon Lambda 函数部署创建应用程序（控制台）和为 Amazon Lambda 部署添加 AppSpec 文件。	
新主题	CodeDeploy 现在支持直接部署到安装 CodeDeploy 代理的本地计算机或实例。您可以在本地测试部署，如果部署有错误，则使用 CodeDeploy 代理错误日志对其进行调试。您还可以使用本地部署来测试应用程序修订版本的完整性、 AppSpec 文件内容等。有关更多信息，请参阅 使用 CodeDeploy 代理验证本地机器上的部署程序包。	2017 年 11 月 16 日
更新的主题	CodeDeploy 部署组中对 ELB 负载均衡器的支持已扩展到包括用于部署和就地 blue/green 部署的网络负载均衡器。现在，您可以为部署组选择应用程序负载均衡器、经典负载均衡器或网络负载均衡器。部署需要负载均衡器，就地 blue/green 部署是可选的。为反映此支持，大量主题已更新，包括 Integrating CodeDeploy with Elastic Load Balancing、为就地部署创建应用程序（控制台）、部署先决条件、Integrating CodeDeploy with Elastic Load Balancing 和 为就地部署创建应用程序（控制台）。	2017 年 9 月 12 日
更新的主题	CodeDeploy 部署组中对 ELB 负载均衡器的支持已扩展到包括用于部署和就地 blue/green 部署的应用程序负载均衡器。现在，您可以为部署组选择应用程序负载均衡器和经典负载均衡器。部署需要负载均衡器，就地 blue/green 部署是可选的。Integrating CodeDeploy with Elastic Load Balancing、使用创建应用程序 CodeDeploy 和 使用创建部署组 CodeDeploy 等主题已更新，以反映这一新增支持。	2017 年 8 月 10 日

更改	描述	更改日期
新增和更新的主题	CodeDeploy 现在支持使用多个标签组来识别要包含在部署组中的实例的并集和交叉点。如果使用唯一标签组，该组中至少一个标签标记的实例即会包括在部署组中。如果使用多个标签组，只有由每个标签组中至少一个标签标记的实例才会包括在内。有关在部署组中添加实例的新方法，请参阅Tagging Instances for Deployments。经过更新以反映此项支持的其他主题包括 为就地部署创建应用程序（控制台）、为蓝绿部署创建应用程序（控制台）、为就地部署创建部署组（控制台）、为 EC2 /本地 blue/green 部署创建部署组（控制台）、Deployments，和 教程：使用 CodeDeploy 从 GitHub 部署应用程序 中的 步骤 5：创建应用程序和部署组。	2017 年 7 月 31 日
更新了主题	中添加了另外两种在 Windows 服务器实例上安装 CodeDeploy 代理的方法安装适用于 Windows Server 的 CodeDeploy 代理。除了 Windows PowerShell 命令外，现在还提供了使用直接 HTTPS 链接和 Amazon S3 复制命令下载安装文件的说明。在将文件下载或复制到实例后，可手动运行安装。	2017 年 7 月 12 日
更新的主题	CodeDeploy 改进了它管理 GitHub 账户和存储库连接的方式。现在，您可以创建和存储最多 25 个 GitHub账户连接，以便将 CodeDeploy 应用程序与 GitHub 存储库相关联。每个连接均可支持多个存储库。您最多可以创建到 25 个不同 GitHub 账户的连接，也可以为单个账户创建多个连接。将应用程序连接到 GitHub账户后，无需您执行任何进一步操作即可 CodeDeploy 管理所需的访问权限。已对指定存储在存储 GitHub 库中的修订的相关信息、将 CodeDeploy 与 GitHub 集成和教程：使用 CodeDeploy 从 GitHub 部署应用程序进行更新以反映此支持。	2017 年 5 月 30 日

更改	描述	更改日期
更新的主题	过去，如果 CodeDeploy 代理在目标位置检测到的文件不是最近成功部署的应用程序修订的一部分，则默认情况下，当前部署将失败。 CodeDeploy 现在为代理如何处理这些文件提供了选项：部署失败、保留内容或覆盖内容。 使用创建部署 CodeDeploy已更新以反映这种支持，并在中添加现有内容的回滚行为了新的章节使用 CodeDeploy 重新部署和回滚部署。	2017 年 5 月 16 日
更新的主题	现在，可以使用 CodeDeploy 控制台或将 Elastic Load Balancing 中的 Classic Load Balancing 分配给部署组 Amazon CLI。在就地部署期间，负载均衡器阻止将 Internet 流量路由到正在部署到的实例，然后在该实例上的部署完成时使实例可供流量使用。已更新多个主题来反映此新的支持，包括与其他 Amazon 服务集成、Integrating CodeDeploy with Elastic Load Balancing、为就地部署创建应用程序（控制台）、为就地部署创建部署组（控制台）和AppSpec “挂钩” 部分。已向故障排除指南添加一个新的部分：对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除。	2017 年 4 月 27 日
更新的主题	现在，可以使用 CodeDeploy 控制台或将 Elastic Load Balancing 中的 Classic Load Balancing 分配给部署组 Amazon CLI。在就地部署期间，负载均衡器阻止将 Internet 流量路由到正在部署到的实例，然后在该实例上的部署完成时使实例可供流量使用。已更新多个主题来反映此新的支持，包括与其他 Amazon 服务集成、Integrating CodeDeploy with Elastic Load Balancing、为就地部署创建应用程序（控制台）、为就地部署创建部署组（控制台）和AppSpec “挂钩” 部分。已向故障排除指南添加一个新的部分：对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除。	2017 年 5 月 1 日

更改	描述	更改日期
更新的主题	CodeDeploy 现已在中国（北京）地区推出。 要使用中国（北京）区域或中国（宁夏）区域中的服务，您必须拥有特定于这些区域的账户和凭证。其他 Amazon 区域的账户和凭证不适用于北京和宁夏区域，反之亦然。 有关中国区域某些资源的信息， CodeDeploy 例如资源套件存储桶名称 CodeDeploy 和代理安装过程，未包含在本版本的用户指南CodeDeploy 中。 有关更多信息： • CodeDeployin 中国（北京） Amazon 地区入门 • CodeDeploy 中国地区用户指南（英文版 | 中文版）	2017 年 3 月 29 日
新增和更新的主题	引入了几个新主题来反映对部署的新 CodeDeploy 支持，blue/green 部署是一种部署方法，在这种部署方法中，部署组（原始环境）中的实例被另一组实例（替换环境）所取代。 蓝绿部署概述提供了对所用 blue/green 方法的高级解释 CodeDeploy。其他新主题包括为蓝绿部署创建应用程序（控制台）、为 EC2 /本地 blue/green 部署创建部署组（控制台）和在 ELB 中为 CodeDeploy Amazon EC2 部署设置负载均衡器。 还更新了许多主题，其中包括 使用创建部署 CodeDeploy、在中使用部署配置 CodeDeploy、使用创建应用程序 CodeDeploy、在中使用部署组 CodeDeploy、在中处理部署 CodeDeploy 和 AppSpec “挂钩” 部分。	2017 年 1 月 25 日
新增和更新的主题	一个新主题介绍如何使用通过 Amazon Security Token Service生成的定期更新的临时证书对本地实例进行身份验证和注册。使用 register-on-premises-instance命令（IAM 会话 ARN）注册本地实例相比对每个实例仅使用一个静态 IAM 用户的凭证，此方法能够为大量本地实例提供更好的支持。Working with On-Premises Instances的内容也已更新，以反映这一新的支持功能。	2016 年 12 月 28 日

更改	描述	更改日期
更新的主题	CodeDeploy 现已在欧洲（伦敦）区域 (eu-west-2) 上市。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。	2016 年 12 月 13 日
更新的主题	CodeDeploy 现已在加拿大（中部）区域 (ca-central-1) 上市。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。	2016 年 12 月 8 日
更新的主题	CodeDeploy 现已在美国东部（俄亥俄州）区域 (us-east-2) 上市。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。	2016 年 10 月 17 日
新主题	新的章节 “身份验证和访问控制” 提供了有关使用 Amazon Identity and Access Management (IAM) 的全面信息，以及 CodeDeploy 如何通过使用证书来帮助保护对资源的访问。这些证书提供访问 Amazon 资源所需的权限，例如从 Amazon S3 存储桶检索应用程序修订版和读取 Ama EC2 zon 实例上的标签。	2016 年 10 月 11 日
更新了主题	更新 Windows Server 上的 CodeDeploy 代理已更新，以反映适用于 Windows CodeDeploy 服务器的新代理更新程序的可用性。在 Windows Server 实例上安装后，更新程序将定期检查新版本。当检测到新版本时，更新程序将在安装最新版本之前，卸载当前版本的代理（如果已安装）。	2016 年 10 月 4 日

更改	描述	更改日期
更新的主题	CodeDeploy 现在与 Amazon CloudWatch 警报集成，如果指定警报的状态连续发生变化（如警报阈值中所指定），则可以停止部署。 CodeDeploy 现在还支持在满足某些条件（例如部署失败或警报已激活）时自动回滚部署。 已更新了大量主题来反映这些更改，包括使用创建应用程序 CodeDeploy、使用创建部署组 CodeDeploy、使用 CodeDeploy 更改部署组设置、Deployments、使用 CodeDeploy 重新部署和回滚部署和与 CodeDeploy 集成的产品和服务，此外新增了主题使用 CloudWatch 警报监控部署 CodeDeploy。	2016 年 9 月 15 日
新增和更新的主题	CodeDeploy 现在提供与 Amazon Ev CloudWatch ents 的集成。现在，当检测到部署状态或属于 CodeDeploy 部署组的实例的状态发生变化时，您可以使用 CloudWatch 事件启动一项或多项操作。您可以合并调用 Amazon Lambda 函数、发布到 Kinesis 直播或 Amazon SNS 主题的操作、将消息推送到 Amazon SQS 队列的操作，或者反过来触发警报操作的操作。 CloudWatch 有关更多信息，请参阅 使用 Amazon CloudWatch 事件监控部署。	2016 年 9 月 9 日
主题更新	主题Integrating CodeDeploy with Elastic Load Balancing 和与其他 Amazon 服务集成已更新，以反映额外的负载平衡选项。 CodeDeploy 现在支持 Elastic Load Balancing 中提供的经典负载均衡器和应用程序负载均衡器。	2016 年 8 月 11 日
主题更新	CodeDeploy 现已在亚太地区（孟买）区域 (ap-south-1) 推出。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。	2016 年 6 月 27 日

更改	描述	更改日期
主题更新	CodeDeploy 现已在亚太地区（首尔）区域 (ap-northeast-2) 上市。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来反应这一新近可用的区域。 已重新组织内容表，来包含针对实例、部署配置、应用程序、部署组、修订和部署的各个部分。已为 CodeDeploy 教程添加新的部分。为了提高可用性，已将几个较长的主题（包括 CodeDeploy AppSpec 文件参考 和 故障排除 CodeDeploy）划分为多个较短的主题。 CodeDeploy 代理的配置信息已移至新主题 CodeDeploy 代理配置参考。	2016 年 6 月 15 日
新增和更新的主题	Amazon CodeDeploy 的错误代码提供了有关 CodeDeploy 部署失败时可能显示的一些错误消息的信息。 故障排除 CodeDeploy中的下列部分已更新，以能够更好地帮助解决部署问题： • EC2 Amazon A EC2 uto Scaling 组中的实例无法启动并收到 “心跳超时” 错误 • 避免将多个部署组与单个 Amazon A EC2 uto Scaling 组相关联	2016 年 4 月 20 日
主题更新	CodeDeploy 现已在南美洲（圣保罗）区域 (sa-east-1) 上市。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来反应这一新近可用的区域。 与 CodeDeploy 代理合作已更新，以反映新的:max_revisions: 配置选项，您可以使用该选项为要代理存档的部署组指定应用程序修订的数量。 CodeDeploy	2016 年 3 月 10 日

更改	描述	更改日期
新增和更新的主题	CodeDeploy 现在支持向部署组添加触发器，以接收与该部署组中的部署或实例相关的事件的通知。对于您加入到该触发器操作中的 Amazon Simple Notification Service 主题，通知将发送到已订阅该主题的接收人。您也可以使用当触发器在您自己的自定义通知工作流中触发时创建的 JSON 数据。有关更多信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications。 对过程进行了更新，来说明重新设计的应用程序详细信息页面。 故障排除 CodeDeploy中的如果实例在部署期间终止，在最多 1 小时内部署不会失败。部分已更新。 对CodeDeploy 配额进行了更新，以反映可与单个应用程序关联的部署组数量的修订后限制、正常运行的最少实例数设置所允许的值以及 适用于 Ruby 的 Amazon SDK要求的版本。	2016 年 2 月 17 日
新增和更新的主题	CodeDeploy 现已在美国西部（加利福尼亚北部）区域 (us-west-1) 上线。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新增的区域。 选择 CodeDeploy 存储库类型列出并描述了 CodeDeploy 目前支持的存储库类型。在引入对其他存储库的支持时，这一新主题将会进行更新。 管理 CodeDeploy 代理操作已更新，其中包含有关添加到实例以报告 CodeDeploy 代理当前版本的新`.version`文件的信息，以及有关代理支持的版本的信息。 用户指南中增加了代码示例的语法突出显示，包括 JSON 和 YAML 示例。 将应用程序规范文件添加到 CodeDeploy 的修订已按 step-by-step说明进行重组。	2016 年 1 月 20 日

更改	描述	更改日期
新主题	在其他 Amazon 账户中部署应用程序描述了在不需要其他账户的完整凭证集的情况下，用于启动属于您组织中其他账户的部署时的设置要求和过程。这对于将多个账户用于不同用途的组织尤为有用，例如一个账户与开发和测试环境关联，另一个账户与生产环境关联。	2015 年 12 月 30 日
主题更新	与 CodeDeploy 集成的产品和服务主题进行了重新设计。该主题现在包括来自社区的集成示例部分，并提供与 CodeDeploy 集成相关的博客文章和视频示例列表。	2015 年 12 月 16 日
主题更新	CodeDeploy 现已在亚太地区（新加坡）区域 (ap-southeast-1) 推出。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来反应这一新近可用的区域。	2015 年 12 月 9 日
主题更新	与 CodeDeploy 代理合作进行了更新，以反映 CodeDeploy 代理配置文件中的新 `:proxy_uri:` 选项。CodeDeploy AppSpec 文件参考中更新了有关使用新环境变量 `DEPLOYMENT_GROUP_ID` 的信息；在部署生命周期事件期间，挂钩脚本可以访问该变量。	2015 年 12 月 1 日
主题更新	步骤 2：为创建服务角色 CodeDeploy已更新，以反映为其创建服务角色的新程序， CodeDeploy 并纳入了其他改进。	2015 年 11 月 13 日
主题更新	CodeDeploy 现已在欧洲（法兰克福）区域（eu-central-1）上市。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这个新区域的可用性。 故障排除 CodeDeploy主题更新了有关确保实例的时间设置准确无误的信息。	2015 年 10 月 19 日
新主题	Amazon CloudFormation 模板供 CodeDeploy参考已发布，以反映对 CodeDeploy 行动的新 Amazon CloudFormation 支持。 创建了Primary Components主题，并引入了目标修订的定义。	2015 年 10 月 1 日

更改	描述	更改日期
主题更新	使用创建部署组 CodeDeploy进行了更新，以说明使用通配符搜索为部署组查找实例的功能。 Instance Health进行了更新，以澄清正常运行的最少实例数概念。	2015 年 8 月 31 日
主题更新	CodeDeploy 现已在亚太地区（东京）区域 (ap-northeast-1) 上市。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来反应这一新近可用的区域。	2015 年 8 月 19 日
主题更新	CodeDeploy 现在支持部署到红帽企业 Linux (RHEL) 本地实例和 Amazon EC2 实例。有关更多信息，请参阅以下主题： • CodeDeploy 代理支持的操作系统 • 使用适用于 CodeDeploy 的实例 • 教程：将 WordPress 部署到 Amazon EC2 实例（Amazon Linux 或 Red Hat Enterprise Linux 和 Linux、macOS 或 Unix） • 教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux）	2015 年 6 月 23 日
主题更新	CodeDeploy 现在提供了一组环境变量，您的部署脚本可以在部署期间使用。这些环境变量包括诸如当前 CodeDeploy 应用程序的名称、部署组和部署生命周期事件以及当前 CodeDeploy 部署标识符之类的信息。有关更多信息，请参阅CodeDeploy AppSpec 文件参考中AppSpec “挂钩” 部分部分的结尾。	2015 年 5 月 29 日

更改	描述	更改日期
主题更新	CodeDeploy 现在在 IAM 中提供了一组 Amazon 托管策略，您可以使用这些策略，而不必自己手动创建等效的策略。这些方法包括： • 一项政策，允许用户 CodeDeploy 仅向其注册修订版，然后通过进行部署 CodeDeploy。 • 为用户提供对 CodeDeploy资源的完全访问权限的策略。 • 为用户提供对 CodeDeploy资源的只读访问权限的策略。 • 附加到服务角色的策略， CodeDeploy 以便通过亚马逊 EC2 标签、本地 EC2 实例标签或 Amazon A EC2 uto Scaling 组名来识别亚马逊实例，并相应地向它们部署应用程序修订。 有关更多信息，请参阅“身份验证和访问控制”中的 客户管理型策略示例 部分。	2015 年 5 月 29 日
主题更新	CodeDeploy 现已在欧洲（爱尔兰）区域（eu-west-1）和亚太地区（悉尼）区域（ap-southeast-2）推出。更新了多个主题，包括包含 CodeDeploy 代理设置说明的主题，以反映这些新区域的可用性。	2015 年 5 月 7 日
新主题	CodeDeploy 现在支持部署到本地实例和 Amazon EC2 实例。增加了以下主题来描述这一新的支持： • Working with On-Premises Instances • 教程：使用 CodeDeploy 将应用程序部署到本地实例（Windows 服务器、Ubuntu Server 或 Red Hat Enterprise Linux） • Working with On-Premises Instances	2015 年 4 月 2 日
新主题	增加了CodeDeploy 资源。	2015 年 4 月 2 日

更改	描述	更改日期
主题更新	更新了故障排除 CodeDeploy： • 新的长时间运行的进程可能会导致部署失败部分描述了一些步骤，您可以使用这些步骤来确定和解决长时间运行进程造成的部署故障。 • 该Amazon A EC2 uto Scaling 常规疑难解答部分已更新，显示 CodeDeploy 已将 CodeDeploy 代理的 Amazon A EC2 uto Scaling 超时逻辑从五分钟增加到一小时。 • 新的不匹配的 Amazon A EC2 uto Scaling 生命周期挂钩可能会导致对 Amazon A EC2 uto Scaling 群组的自动部署停止或失败章节介绍了您可以采取的步骤，以识别和解决自动部署到 Amazon A EC2 uto Scaling 群组失败的问题。	2015 年 4 月 2 日
主题更新	下列主题进行了更新，以反映创建您自己的自定义策略并将其附加到 IAM 中的用户和角色的新建议： • 配置要 EC2 使用的亚马逊实例 CodeDeploy • 步骤 4：为您的 Amazon 实例创建 IAM EC2 实例配置文件 • 步骤 2：为创建服务角色 CodeDeploy 故障排除 CodeDeploy中增加了两个部分： • 一般问题排查核对清单 • 默认情况下，Windows PowerShell 脚本无法使用 64 位版本 PowerShell 的 Windows CodeDeploy AppSpec 文件参考中的AppSpec “挂钩” 部分部分进行了更新，更加准确地描述了可用的部署生命周期事件。	2015 年 2 月 12 日

更改	描述	更改日期
主题更新	故障排除 CodeDeploy：EC2 Amazon A EC2 uto Scaling 组中的实例无法启动并收到 “心跳超时” 错误中增加了新的部分。 已向中添加了一个 CloudBees 部分与 CodeDeploy 集成的产品和服务。	2015 年 1 月 28 日
主题更新	故障排除 CodeDeploy中增加了以下部分： • 使用某些文本编辑器创建 AppSpec文件和 shell 脚本可能会导致部署失败 • 使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 • 对失败 ApplicationStop BeforeBlockTraffic、或 AfterBlockTraffic 部署生命周期事件进行故障排除 • 使用以下命令对失败的 DownloadBundle 部署生命周期事件进行故障排除 UnknownError：未打开供读取 • Amazon A EC2 uto Scaling 常规疑难解答	2015 年 1 月 20 日
新主题	与 CodeDeploy 集成的产品和服务部分进行了更新，现在包括下列主题：： • CodeDeploy 与 Amazon A EC2 uto Scaling 集成 • 教程：使用 CodeDeploy 将应用程序部署到 Auto Scaling 组 • Monitoring Deployments • Integrating CodeDeploy with Elastic Load Balancing • 将 CodeDeploy 与 GitHub 集成 • 教程：使用 CodeDeploy 从 GitHub 部署应用程序	2015 年 1 月 9 日

更改	描述	更改日期
主题更新	• 将 CodeDeploy 与 GitHub 集成中增加了使用 CodeDeploy 从 CodePipeline 自动部署部分。现在，只要 GitHub 仓库中的源代码发生更改，您就可以自动触发该存储库中的部署。 • 故障排除 CodeDeploy中增加了解决 Amazon A EC2 uto Scaling 问题部分。这个新章节介绍如何解决部署到 Amazon A EC2 uto Scaling 群组时遇到的常见问题。 • CodeDeploy AppSpec 文件参考的AppSpec “文件” 部分（仅EC2限本地部署）部分中增加了新的子部分“文件示例”。这个新的小节包括几个示例，说明如何在部署期间使用 AppSpec 文件`files`部分指示将特定文件或文件夹复制 CodeDeploy 到 Amazon EC2 实例上的特定位置。	2015 年 1 月 8 日
新主题	Monitoring Deployments已添加。 CodeDeploy 与一项服务集成 Amazon CloudTrail，该服务可捕获由您的账户或代表您的 Amazon 账户发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。 CodeDeploy	2014 年 12 月 17 日
第一个公开发布版	这是 CodeDeploy 用户指南 的第一个公开发行版。	2014 年 11 月 12 日

Amazon 术语表

有关最新的 Amazon 术语，请参阅 Amazon Web Services 词汇表 参考中的 Amazon 词汇表。

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。