Amazon CodeDeploy

用户指南

API 版本 2014-10-06

亚马逊云科技

Amazon CodeDeploy: 用户指南

Copyright © 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或者贬低或诋毁 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。

Table of Contents

什么是 CodeDeploy？ 1
- Amazon CodeDeploy 的优势 2
- CodeDeploy 计算平台概述 2
- CodeDeploy 部署类型概述 4
  - 就地部署概述 5
  - 蓝/绿部署概述 6
- 我们希望听到您的意见和建议 8
- 主要组件 8
- 部署 11
  - 上的部署AmazonLambda 计算平台 11
  - Amazon ECS 计算平台上的部署 13
  - EC2/本地计算平台上的部署 20
- 应用程序规范文件 24
  - Amazon ECS 计算平台上的 AppSpec 文件 24
  - AppSpec 文件Amazon Lambda计算平台 24
  - EC2/本台计算平台上的AppSpec 文件 25
  - CodeDeploy 代理如何使用 AppSpec 文件 25

开始使用 26
- 第 1 步：安装或升级，然后配置Amazon CLI 26
- 第 2 步：创建服务角色 26
  - 创建服务角色（控制台） 28
  - 创建服务角色 (CLI) 28
  - 获取服务角色 ARN（控制台） 29
  - 获取服务角色 ARN (CLI) 29
- 第 3 步：预配 IAM 用户 30
- 第 4 步：创建 IAM 实例配置文件 31
  - 为创建 IAM 实例配置文件 (CLI) 32
  - 为创建 IAM 实例配置文件（控制台） 33

产品和服务集成 36
- 与其他Amazon服务 36
  - Amazon EC2 Auto Scaling 39
  - Elastic Load Balancing 42
- 与合作伙伴的产品和服务集成 45
  - GitHub 47
- 来自社区的集成示例 49
  - 博客帖子 49

教程 51
- 教程：将 WordPress 部署到非 Windows 实例 51
  - 第 1 步：启动 Amazon EC2 实例 52
  - 第 2 步：配置源内容 53
  - 第 3 步：将应用程序上传到 Amazon S3 57
  - 第 4 步：部署您的应用程序 59
  - 第 5 步：更新和重新部署您的应用程序 63
  - 第 6 步：清除 66
- 教程：将 Hello, World! 应用程序部署到 Windows Server 实例 68
  - 第 1 步：启动 Amazon EC2 实例 68
  - 第 2 步：配置源内容 70
  - 第 3 步：将应用程序上传到 Amazon S3 72
  - 第 4 步：部署您的应用程序 75
  - 第 5 步：更新和重新部署您的应用程序 78
  - 第 6 步：清除 80
- 教程：将应用程序部署到本地实例 82
  - Prerequisites 82
  - 第 1 步：配置本地实例 82

第 2 步：创建示例应用程序修订 82
第 3 步：捆绑应用程序修订并上传到 Amazon S3 86
第 4 步：部署您的应用程序修订 86
第 5 步：验证部署 86
第 6 步：清理资源 86
教程：部 Amazon EC2 组 88
Prerequisites 88
第 1 步：创建和配置 Amazon EC2 Auto Scaling 组 89
第 2 步：将应用程序部署到 Amazon EC2 Auto Scaling 组 92
第 3 步：检查您的结果 96
第 4 步：增加 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例的数量 97
第 5 步：再次检查您的结果 98
第 6 步：清除 99
教程：从 GitHub 部署应用程序 101
Prerequisites 101
第 1 步：设置 GitHub 账户 101
第 2 步：创建 GitHub 存储库 101
第 3 步：将示例应用程序上传到 GitHub 存储库 103
第 4 步：置备实例 105
第 5 步：创建应用程序和部署组 105
第 6 步：将应用程序部署到实例 106
步骤 7：监控和验证部署 109
步骤 8：清除 109
教程：部署 Amazon ECS 服务 111
Prerequisites 112
第 1 步：更新您的 Amazon ECS 应用程序 112
第 2 步：创建 AppSpec 文件 113
第 3 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务 114
第 4 步：清除 116
教程：通过验证测试部署 Amazon ECS 服务 117
Prerequisites 118
第 1 步：创建测试侦听器 119
第 2 步：更新您的亚马逊 ECS 应用程序 119
第 3 步：创建生命周期挂钩 Lambda 函数 119
第 4 步：更新 AppSpec 文件 121
第 5 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务 122
第 6 步：在 CloudWatch Logs 中查看您的 Lambda 挂钩函数输出 123
步骤 7：清除 124
教程：使用部署 Lambda 函数AmazonSAM PACKAGE 124
Prerequisites 125
第 1 步：设置您的基础架构 125
第 2 步：更新 Lambda 函数 135
第 3 步：部署更新的 Lambda 函数 136
第 4 步：查看部署结果 138
第 5 步：清除 140
使用 CodeDeploy 代理 141
CodeDeploy 代理支持的操作系统 141
支持的 Amazon EC2 AMI 操作系统 141
支持的本地操作系统 141
CodeDeploy 代理的通信协议和端口 142
CodeDeploy 代理的版本历史记录 142
应用程序修订和日志文件清理 147
由 CodeDeploy 代理安装的文件 147
管理 CodeDeploy 代理操作 149
验证代 CodeDeploy 运行 149
确定 CodeDeploy 代理的版本 150
安 CodeDeploy 151

更新 CodeDeploy 代理 158
卸载 CodeDeploy 161
使用实例 162
将 Amazon EC2 实例与本地实例进行比较 162
CodeDeploy 的实例任务 163
为 CodeDeploy 部署标记实例 163
示例 1：单个标签组，单个标签 164
示例 2：单个标签组，多个标签 165
示例 3：多个标签组，单个标签 166
示例 4：多个标签组，多个标签 168
使用 Amazon EC2 实例 170
为 CodeDeploy 创建 Amazon EC2 实例 170
创建 Amazon EC2 实例 (Amazon CloudFormation模板) 174
配置 Amazon EC2 实例 177
使用本地实例 179
配置本地实例的先决条件 180
注册本地实例 181
管理本地实例操作 198
查看实例详细信息 202
查看实例详细信息（控制台） 202
查看实例详细信息 (CLI) 202
实例运行状况 203
运行状况 203
最小运行正常的实例数 204
使用部署配置 206
EC2/本地计算平台上的部署配置 206
预定义的部署配置 206
Amazon ECS 计算平台上的部署配置 207
Amazon 云服务器的预定义部署配置 208
部署配置Amazon CloudFormation蓝/绿部署（Amazon ECS） 208
上的部署配置Amazon Lambda计算平台 209
Lambda 的预定义部署配置 209
..... 209
创建部署配置 210
查看部署配置详细信息 210
查看部署配置详细信息（控制台） 210
查看部署配置 (CLI) 211
删除部署配置 211
使用应用程序 212
创建应用程序 212
为就地部署创建应用程序（控制台） 213
为蓝/绿部署创建应用程序（控制台） 215
创建 Amazon ECS 服务的应用程序 (控制台) 217
创建一个应用程序Amazon Lambda函数部署（控制台） 218
创建应用程序 (CLI) 219
查看申请详细信息 219
查看应用程序详细信息 220
查看应用程序详细信息 (CLI) 220
创建通知规则 220
重命名应用程序 222
删除应用程序 222
删除应用程序（控制台） 222
删除应用程序 (Amazon CLI） 223
使用部署组 224
Amazon 云服务器计算平台部署中的部署组 224
中的部署组Amazon Lambda计算平台部署 224
EC2/本地计算平台部署中的部署组 224

224
创建部署组 225
为就地部署创建部署组（控制台） 225
为 EC2/本地蓝/绿部署（控制台）创建部署组 227
为 Amazon 云服务器部署创建部署组（控制台） 229
在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 230
为 CodeDeploy Amazon 云服务器部署设置负载均衡器、目标组和侦听器 230
创建部署组 (CLI) 233
查看部署组详细信息 234
查看部署组详细信息（控制台） 234
查看部署组详细信息 (CLI) 234
更改部署组设置 235
更改部署组设置（控制台） 235
更改部署组设置 (CLI) 235
为部署组配置高级选项 236
删除部署组 238
删除部署组（控制台） 238
删除部署组 (CLI) 238
使用应用程序修订 239
计划修订 239
添加 AppSpec 文件 239
为 Amazon ECS 部署添加 AppSpec 文件 240
AppSpecAmazonLambda 部署 242
为 EC2/本地部署部署添加AppSpec 文件 243
选择存储库类型 246
推送修订 247
推送修订版本使用Amazon CLI 248
查看应用程序修订细节 249
查看应用程序修订详细信息 250
查看应用程序修订详细信息 (CLI) 250
注册应用程序修订 251
使用 CodeDeploy (CLI) 在 Amazon S3 中注册修订 251
使用 CodeDeploy (CLI) 在 GitHub 中注册修订 252
使用部署 253
创建部署 254
部署先决条件 254
创建 Amazon ECS 计算平台部署（控制台） 256
创建AmazonLambda 计算平台部署（控制台） 257
创建EC2/本地计算平台部署（控制台） 258
创建 Amazon ECS 计算平台部署 (CLI) 261
创建AmazonLambda 计算平台部署 (CLI) 261
创建EC2/本地计算平台部署(CLI) 262
通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation 264
查看部署详细信息 266
查看部署详细信息（控制台） 267
查看部署详细信息 (CLI) 267
查看部署日志数据 268
在 Amazon CloudWatch 控制台中查看日志文件数据 268
查看实例上的日志文件 268
停止部署 270
停止部署（控制台） 270
停止部署 (CLI) 271
重新部署和回滚部署 271
自动回滚 271
手动回滚 271
回滚和重新部署工作流程 271
对现有内容进行回滚行为 272

将应用程序部署到不同Amazon账户 273
第 1 步：在任一账户中创建 S3 存储桶 274
第 2 步：向 Amazon S3 存储桶授予对生产账户的 IAM 实例配置文件的权限 274
第 3 步：创建资源和跨账户角色 275
第 4 步：将应用程序修订上传到 Amazon S3 存储桶 275
第 5 步：担任跨账户角色并部署应用程序 275
验证本地机器上的部署程序包 276
Prerequisites 276
创建本地部署 278
Examples 279
监控部署 281
自动化工具 281
手动工具 282
使用 Amazon CloudWatch 工具监控部署 283
使用云监控警报监控部署 283
使用 Amazon CloudWatch Events 监控部署 284
使用监视部署Amazon CloudTrail 285
代码在 CloudTrail 中部署信息 286
了解 CodeDeploy 日志文件条目 286
使用 Amazon SNS 事件通知监控部署 287
向 Amazon SNS 授予权限 288
为 CodeDeploy 事件创建触发器 288
编辑部署组中的触发器 293
从部署组中删除触发器 294
用于触发器的 JSON 数据格式 294
安全性 296
数据保护 296
互联网络流量隐私 297
静态加密 297
传输中加密 297
加密密钥管理 297
身份和访问管理 297
Audience 298
使用身份进行身份验证 298
使用策略管理访问 299
Amazon CodeDeploy 如何与 IAM 协同工作 300
基于身份的策略示例 303
故障排除 311
CodeDeploy 权限参考 313
事件响应 318
审计与 CodeDeploy 的所有交互 319
提醒和事件管理 319
合规性验证 319
故障恢复能力 320
基础设施安全性 320
参考 321
AppSpec 文件参考 321
Amazon ECS 计算平台上的 AppSpec 文件 321
AppSpec 文件Amazon Lambda计算平台 322
EC2/本地计算平台上的AppSpec 文件 322
AppSpec 文件结构 322
AppSpec 文件示例 345
AppSpec 文件间距 349
验证您的 AppSpec 文件和文件位置 350
代理配置参考 350
相关主题 353
Amazon CloudFormation模板参考 353

将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用 354
Availability 355
为 CodeDeploy. 355
配置 CodeDeploy 代理和 IAM 权限 356
资源工具包参考 356
按区域列出的资源工具包存储桶 356
资源工具包内容 357
显示资源工具包文件的列表 358
下载资源工具包文件 358
限制 359
Applications 359
应用程序修订 359
Deployments 360
部署配置 360
部署组 361
Instances 361
故障排除 362
一般问题排查 362
一般问题排查核对清单 362
只有部分支持 CodeDeploy 部署资源Amazon区域 363
本指南中的步骤与 CodeDeploy 控制台不匹配 363
所需的 IAM 角色不可用 364
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 364
使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 364
EC2/本地部署问题 364
CodeDeploy 代理无法在 Windows 2016 上启动 365
CodeDeploy 插件 CommandPoller 缺失凭证错误 365
部署失败，显示消息“PKCS7 签名的消息验证失败” 366
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” 366
长时间运行的进程可能会导致部署失败 367
排查 AllowTraffic 生命周期事件失败，但部署日志中未报错的问题 368
排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 368
排查失败的 DownloadBundle 部署生命周期事件的问题，错误为“UnknownError: not opened for reading” 369
排查所有生命周期事件跳过错误 369
默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell 370
故障排除AmazonLambda 部署问题 371
Amazon Lambda在手动停止未配置回滚的 Lambda 部署后，部署失败 371
排查部署组问题 372
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 372
排查实例问题 372
必须正确设置标签 372
Amazon CodeDeploy实例上必须安装并运行代理 372
如果实例在部署期间终止，在最多 1 小时内部署不会失败。 372
分析日志文件以调查针对实例的部署失败 373
如果已意外删除，请创建一个新的 CodeDeploy 日志文件 373
排查“InvalidSignatureException – Signature expired: [time] is now earlier than [time]”部署错误 373
GitHub 令牌问题排查 373
无效 GitHub OAuth 令牌 373
超出了 GitHub OAuth 令牌数量上限 374
解决 Amazon EC2 Auto Scaling 问题 374
Amazon EC2 Auto Scaling 常规故障排除 374
“CodeDeploRole 没有授予您执行以下操作的权限：Amazon服务：AmazonAutoScaling” 错误消息 375
在部署修订之前，Amazon EC2 Auto Scaling 组中的实例不断被预置和终止 375
终止或重启 Amazon EC2 Auto Scaling 实例可能会导致部署失败 376

避免将多个部署组与一个 Amazon EC2 Auto Scaling 组关联 376
Amazon EC2 Auto Scaling 组中的 EC2 实例无法启动，收到错误 “心跳超时” 377
不匹配的 Amazon EC2 Auto Scaling 生命周期挂钩可能导致至 Amazon EC2 Auto Scaling 组的自动部署停止或失败。 378
“部署失败是因为没有为您的部署组找到实例” 错误 379
错误代码 383
相关主题 385
资源 386
参考指南和支持资源 386
Samples 386
Blogs 386
Amazon软件开发工具包和工具 386
文档历史记录 388
早期更新 390
Amazon词汇表 400
..... cdi

什么是 CodeDeploy？

CodeDeploy 是一项部署服务，可以向 Amazon EC2 实例、本地实例、无服务器 Lambda 函数或 Amazon 云服务器服务自动执行应用程序部署。

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 Amazon Lambda 函数
- Web 和配置文件
- 可执行文件
- 软件包
- 脚本
- 多媒体文件

CodeDeploy 可以部署那些在一个服务器上运行并在 Amazon S3 存储桶、GitHub 存储库或 Bitbucket 存储库中存储的应用程序内容。CodeDeploy 还可以部署无服务器 Lambda 函数。您无需更改现有代码即可开始使用 CodeDeploy。

CodeDeploy 能让您更轻松地执行以下操作：

- 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 使用各种系统执行配置管理、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方法，可以快速搜索存储库、构建项目、部署应用程序和管道等资源。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，请参阅适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297)。

主题

- Amazon CodeDeploy 的优势 (p. 2)
- CodeDeploy 计算平台概述 (p. 2)
- CodeDeploy 部署类型概述 (p. 4)
- 我们希望听到您的意见和建议 (p. 8)
- CodeDeploy (p. 8)
- CodeDeploy (p. 11)
- CodeDeploy AppSpec (应用程序规范) 文件 (p. 24)

Amazon CodeDeploy 的优势

CodeDeploy 具备下列优势：

- 服务器、无服务器和容器应用程序. CodeDeploy 既能让您部署服务器上的传统应用程序，又能让您部署用于部署无服务器Amazon Lambda函数版本或 Amazon ECS 应用程序。
- 自动部署. CodeDeploy 可完全自动执行跨您的开发环境、测试环境和生产环境部署应用程序的过程。CodeDeploy 随您的基础设施进行扩展，让您能够部署到一个实例或数千个实例。
- 最大程度减少停机时间。如果您的应用程序使用 EC2/ 本地计算平台，则 CodeDeploy 有助于最大限度地提高应用程序可用性。在就地部署过程中，CodeDeploy 将跨 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。在蓝/绿部署中，最新应用程序修订安装在替换实例上。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Amazon CLI. 您将收到一份报告，其中列出每个应用程序修订的部署时间及其部署到的 Amazon EC2 实例。
- 易于采用. CodeDeploy 与平台无关，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过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/ 本地计算平台，则 CodeDeploy 可同时将其部署到相同的实例组。

CodeDeploy 计算平台概述

CodeDeploy 可以将应用程序部署到三个计算平台：

- EC2/本地：描述可以作为 Amazon EC2 云实例和/或本地服务器的物理服务器实例。使用 EC2/ 本地计算平台创建的应用程序可以包括可执行文件、配置文件和映像等。

  使用 EC2/ 本地计算平台的部署通过使用就地部署或蓝/绿部署类型，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 (p. 4)。
- AmazonLambda：用于部署包含更新后的 Lambda 函数版本的应用程序。Amazon Lambda管理在由高可用性计算结构构成的无服务器计算环境中的 Lambda 函数。计算资源的所有管理工作均由 Amazon Lambda 执行。有关更多信息，请参阅 。无服务器计算和应用. 有关 的更多信息Amazon Lambda和 Lambda 函数，请参阅Amazon Lambda.

  您可以通过选择金丝雀部署、线性方式或一次性配置，管理部署过程中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。

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

下表描述 CodeDeploy 组件如何与每个计算平台一起使用。有关更多信息，请参阅：

- 在 CodeDeploy 中使用部署组 (p. 224)

- 在 CodeDeploy 中使用部署 (p. 253)
- 在 CodeDeploy 中使用部署配置 (p. 206)
- 使用 CodeDeploy 的应用程序修订 (p. 239)
- 使用应 CodeDeploy (p. 212)

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	部署一个包括应用程序和 AppSpec 文件的新修订。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。AppSpec 文件指定要部署哪个 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换任务集。CodeDeploy 将生产流量从具有原始版本的任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
Revision	AppSpec 文件和应用程序文件 (例如，可执行文件、配置文件等) 的组合。	一个 AppSpec 文件，指定要部署的 Lambda 函数以及可在部署生命周期事件挂钩期间运行验证测试的 Lambda 函数。	一个 AppSpec 文件，它指定： • 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可在部署生命周期事件挂钩期间运行验证测试的 Lambda 函数。
应用程序	部署组和修订的集合。EC2/本地应用程序使用EC2/本地计算平台上的计算平台。	部署组和修订的集合。用于Amazon Lambda部署使用无服务器AmazonLambda 计算平台。	部署组和修订的集合。用于Amazon 云服务器部署的应用程序使用Amazon 云服务器计算平台。

CodeDeploy 部署类型概述

CodeDeploy 提供两种部署类型选项：

- 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版，然后启动并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/ 本地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 (p. 5)。

Note

AmazonLambda 和 Amazon ECS 部署无法使用就地部署类型。

- 蓝/绿部署：您的部署行为取决于使用的计算平台：
  - EC2/本地计算平台上的蓝/绿部署：部署组中的实例 (原始环境) 将被不同的实例集 (替代环境) 所代替，步骤如下：
    - 实例是为替换环境预置的。
    - 最新的应用程序修订将安装在替换实例上。
    - 应用程序测试和系统验证等活动将出现可选的等待时间。
    - 替换环境中的实例向弹性负载均衡器注册，从而导致流量重新路由到这些实例。原始环境中的实例将取消注册，可以终止或继续运行以供其他用途。

      Note

      如果您使用 EC2/本地计算平台，请注意蓝/绿色部署仅适用于 Amazon EC2 实例。
  - 蓝/绿部署AmazonLambda 计算平台：流量从当前无服务器环境转移到包含更新后的 Lambda 函数版本的环境。您可以指定执行验证测试的 Lambda 函数并选择流量转移的方式。全部AmazonLambda 计算平台部署是蓝/绿部署。因此，您无需指定部署类型。
  - Amazon ECS 计算平台上的蓝/绿部署：流量从 Amazon ECS 服务中具有原始版本的应用程序任务集转移到同一服务中的替换任务集。您可以通过部署配置将流量转移设置为线性或 Canary。指定负载平衡器侦听器的协议和端口用于重新路由生产流量。在部署期间，在运行验证测试时，可以使用测试侦听器为替换任务集提供流量。
  - 通过 进行蓝/绿部署Amazon CloudFormation：流量从当前资源转移到更新后的资源，作为Amazon CloudFormation堆栈更新。目前，仅支持 ECS 蓝/绿部署。

  有关蓝/绿部署的更多信息，请参阅蓝/绿部署概述 (p. 6)。

Note

利用 CodeDeploy 代理，您可以在已登录的实例上执行部署，而无需应用程序、部署组甚至是Amazonaccount. 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 (p. 276)。

主题

- 就地部署概述 (p. 5)
- 蓝/绿部署概述 (p. 6)

就地部署概述

Note

AmazonLambda 和 Amazon ECS 部署无法使用就地部署类型。

下面是就地部署的工作原理：

1. 首先，您需要在本地开发计算机或类似环境上创建可部署的内容，然后添加应用程序规范文件（AppSpec 文件）。AppSpec 文件对于 CodeDeploy 是唯一的。它定义了您希望 CodeDeploy 执行的部署操作。您可以将可部署的内容和 AppSpec 文件打包到一个存档文件，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，向 CodeDeploy 提供有关您的部署的信息，例如，要从中提取修订的 Amazon S3 存储桶或 GitHub 存储库，以及要将其内容部署到的一组 Amazon EC2 实例。CodeDeploy 调用一组 Amazon EC2 实例部署组. 部署组中包含单独标记的 Amazon EC2 实例和 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例，或两者。

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 代理将轮询 CodeDeploy 以确定从指定的 Amazon S3 存储桶或 GitHub 存储库中提取的内容和提取时间。
4. 最后，每个实例上的 CodeDeploy 代理将从 Amazon S3 存储桶或 GitHub 存储库中提取目标修订，并按照 AppSpec 文件中的说明向实例部署内容。

CodeDeploy 将保留您的部署的记录，以便您可以获取部署状态、部署配置参数、实例运行状况。

蓝/绿部署概述

蓝/绿部署用于更新应用程序，同时最大限度地减少因新应用程序版本更改而造成的中断。CodeDeploy 将您的新应用程序版本与旧版本一起配置，然后再重新路由您的生产流量。

- AmazonLambda：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/本地：流量从原始环境中的一组实例转移到一组替换实例。

全部AmazonLambda 和 Amazon ECS 部署是蓝/绿部署。EC2/本地部署可以是就地部署，也可以是蓝/绿部署。蓝/绿部署相对于就地部署有许多优势：

- 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 EC2/ 本地计算平台，则切换回最新的应用程序版本将更快、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 EC2/ 本地计算平台，则会为蓝/绿部署预置新实例，并且新实例反映最新的服务器配置。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AmazonLambda 计算平台，您可以控制流量如何从原始转移AmazonLambda 函数版本添加到您的新AmazonLambda 函数版本。
- 如果您使用 Amazon ECS 计算平台，则需要控制流量如何从原始任务集转移到新的任务集。

使用 Amazon CloudFormation 进行的蓝/绿部署可以使用以下方法之一：

- Amazon CloudFormation用于部署的模板：将部署配置为Amazon CloudFormation模板时，您的部署将由Amazon CloudFormation更新。当您更改资源并上传模板更改时，Amazon CloudFormation 中的堆栈更新将启动新部署。有关可在 Amazon CloudFormation 模板中使用的资源列表，请参阅Amazon CloudFormation用于 CodeDeploy 参考的模板 (p. 353)。
- 通过 进行蓝/绿部署Amazon CloudFormation：您可以使用Amazon CloudFormation通过堆栈更新来管理蓝/绿部署。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，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移动流量，并删除蓝色资源。有关更多信息，请参阅 。通过 CodeDeploy 使用通过 CodeDeploy 实现 Amazon ECS 蓝/绿部署Amazon CloudFormation中的Amazon CloudFormation用户指南.

  Note

  仅支持 Amazon ECS 蓝/绿部署。

如何配置蓝/绿部署取决于部署使用的计算平台。

蓝/绿部署Amazon Lambda计算平台

如果您使用的是AmazonLambda 计算平台，必须选择下列部署配置类型之一，以指定流量如何从原始Amazon Lambda函数版本添加到新的Amazon Lambda函数版本：

- 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 函数版本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：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。
- 一次性：所有流量均从原始 Lambda 函数一次性地转移到更新后的 Lambda 函数版本。

有关 Amazon Lambda 部署配置的更多信息，请参阅预定义的部署配置Amazon Lambda计算平台 (p. 209)。

Amazon ECS 计算平台上的蓝/绿部署

如果您使用 Amazon ECS 计算平台，则必须选择下列部署配置类型之一，以指定通信如何从原始 Amazon ECS 任务集转移到新的 Amazon 云服务器任务集：

- 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 Amazon 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：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。
- 一次性：所有流量均从原始 Amazon ECS 任务集一次性地转移到更新后的 Amazon ECS 任务集。

有关 Amazon ECS 部署配置的更多信息，请参阅 “”。 Amazon ECS 计算平台上的部署配置 (p. 207).

EC2/本地计算平台上的蓝/绿部署

> **Note**
>
> 您必须在 EC2/本地计算平台上使用 Amazon EC2 实例上的蓝/绿部署。蓝/绿部署类型不支持本地实例。

如果您使用 EC2/ 本地计算平台，则适用以下规则：

您必须有一个或多个带有标识 Amazon EC2 标签的 Amazon EC2 实例或 Amazon EC2 Auto Scaling 组。这些实例必须满足这些额外要求：

- 每个 Amazon EC2 实例必须附加有正确的 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，您还会有一个在原始环境中的实例上运行的应用程序修订，但这对蓝/绿部署来说不是必需的。

当您创建将在蓝/绿部署中使用的部署组时，您可以选择如何指定替换环境：

复制现有 Amazon EC2 Auto Scaling 组：在蓝/绿部署中，CodeDeploy 将在部署期间为替换环境创建实例。利用此选项，CodeDeploy 将使用您指定的 Amazon EC2 Auto Scaling 组作为替换环境的模板，其中包括相同数量的正在运行的实例和许多其他配置选项。

手动选择实例：您可以使用 Amazon EC2 实例标签、Amazon EC2 Auto Scaling 组名称或两者指定要计为替换项的实例。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. 您已有充当原始环境的一些实例或一个 Amazon EC2 Auto Scaling 组。首次运行蓝/绿部署时，您通常使用已在就地部署中使用的实例。
2. 在现有的 CodeDeploy 应用程序中，创建一个蓝/绿部署，在其中，除了就地部署所需的选项之外，您还要指定以下内容：
   - 在蓝/绿部署过程期间，将流量从您原始环境路由到替换环境的负载均衡器。
   - 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 EC2 Auto Scaling 组，则将为您的替换环境预置实例。
   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. 替换环境中的实例向弹性负载均衡器注册，流量开始路由到这些实例。
   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。

通过进行蓝/绿部署Amazon CloudFormation

您可以管理 CodeDeploy 蓝/绿部署，方法是使用Amazon CloudFormation模板。

当您使用 Amazon CloudFormation 模板对蓝/绿资源进行建模时，您可以在更新任务集的 Amazon CloudFormation 中创建堆栈更新。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 CodeDeploy 中查看部署状态和历史记录，但不会在Amazon CloudFormation模板。

> **Note**
>
> 对于通过进行蓝/绿部署Amazon CloudFormation，您不会创建 CodeDeploy 应用程序或部署组。

此方法仅支持 Amazon ECS 蓝/绿部署。有关通过 Amazon CloudFormation 进行蓝/绿部署的更多信息，请参阅通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 264)。

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 (p. 8)
- Deployments (p. 11)
- Application Specification Files (p. 24)

CodeDeploy

您在开始使用此项服务之前，应熟悉 CodeDeploy 部署过程中的主要组件。

应用程序：唯一地标识要部署的应用程序的名称。CodeDeploy 使用此名称，即充当容器，以确保在部署期间引用修订、部署配置和部署组的正确组合。

计算平台：CodeDeploy

- EC2/本地：描述可以作为 Amazon EC2 云实例和/或本地服务器的物理服务器实例。使用 EC2/本地计算平台

  使用 EC2/本地计算平台的部署通过使用就地部署或蓝/绿部署类型，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 (p. 4)。
- AmazonLambda：用于部署包含 Lambda 函数的更新版本的应用程序。Amazon Lambda管理在由高可用性计算结构构构成的无服务器计算环境中的 Lambda 函数。计算资源的所有管理工作均由 Amazon Lambda 执行。有关更多信息，请参阅 。无服务器计算和应用。有关 的更多信息Amazon Lambda和 Lambda 函数，请参阅Amazon Lambda。

  您可以通过选择金丝雀部署、线性方式或一次性配置，管理部署过程中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。

  > **Note**
  >
  > 通过 CodeDeploy 和Amazon CloudFormation。这些部署的详细信息将在后续章节中介绍。

部署配置：CodeDeploy 使用的一组部署规则以及部署成功条件和失败条件。如果您的部署使用 EC2/本地计算平台，您可以为部署指定最少数量运行正常的实例。如果您的部署使用AmazonLambda 计算平台或 Amazon ECS 计算平台，您可以指定流量传输到更新后的 Lambda 函数版本的方式。

有关为使用 EC2/本地计算平台的部署指定最少数量运行正常的主机的详细信息，请参阅最小运行正常的实例数 (p. 204)。

这些部署配置指定流量的路由方式。AmazonLambda 计算平台：

- Canary：流量在两次增量中转移。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 函数版本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：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。
- 一次性：所有流量均从原始 Lambda 函数一次性地转移到更新后的 Lambda 函数版本。

在使用 Amazon 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：流量在两次增量中转移。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 Amazon 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：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。
- 一次性：所有流量均从原始 Amazon ECS 任务集一次性地转移到更新后的 Amazon ECS 任务集。

部署组：一组单独的实例。部署组中包含单独标记的实例和 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例。有关 Amazon EC2 实例标签的信息，请参阅通过控制台使用标签。有关本地实例的信息，请

参阅Working with On-Premises Instances (p. 179)。有关 Amazon EC2 Auto Scaling 的信息，请参阅将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

Deployment type (部署类型)：用于在部署组中的实例上使最新的应用程序修订版本可用的方法。

- 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版，然后启动并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/本地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 (p. 5)。
- 蓝/绿：部署的行为取决于使用的计算平台：
  - EC2/本地计算平台上的蓝/绿：部署组中的实例（原始环境）将被不同的实例集（替换环境）所代替，步骤如下：
    - 为替换环境预置实例。
    - 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，会发生可选的等待时间。
    - 替换环境中的实例向弹性负载均衡器注册，导致流量重新路由到这些实例。原始环境中的实例将取消注册，并且可以终止或保持运行以供其他用途。

      Note

      如果您使用 EC2/本地计算平台，请注意蓝/绿色部署仅适用于 Amazon EC2 实例。
  - 蓝/绿AmazonLambda 计算平台：流量从当前无服务器环境转移到包含更新后的 Lambda 函数版本的环境。您可以指定执行验证测试的 Lambda 函数并选择流量转移的方式。全部AmazonLambda 计算平台部署是蓝/绿部署。因此，您无需指定部署类型。
  - Amazon ECS 计算平台上的蓝/绿：流量从 Amazon ECS 服务中包含原始版本的应用程序的任务集转移到同一服务中的替换任务集。您可以通过部署配置将流量转移设置为线性或 Canary。指定负载平衡器侦听器的协议和端口用于重新路由生产流量。在部署期间，在运行验证测试时，可以使用测试侦听器为替换任务集提供流量。
  - 通过 进行蓝/绿部署Amazon CloudFormation：流量从当前资源转移到更新后的资源。Amazon CloudFormation堆栈更新。目前，仅支持 ECS 蓝/绿部署。

  有关蓝/绿部署的更多信息，请参阅蓝/绿部署概述 (p. 6)。

  Note

  支持使用 CodeDeploy 和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：您附加到 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 GitHub 存储库所需的权限。有关更多信息，请参阅 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。

修订：一个AmazonLambda 部署修订是一个 YAML 或 JSON 格式的文件，指定有关要部署的 Lambda 函数的信息。EC2/本地部署修订是一个存档文件，包含源内容 (源代码、网页、可执行文件和部署脚本) 以及应用程序规范文件 (AppSpec 文件。AmazonLambda 修订可存储在 Amazon S3 存储桶中。EC2/ 本地修订存储在 Amazon S3 存储桶或 GitHub 存储库中。对于 Amazon S3，修订由其 Amazon S3 对象键以及其 ETag 和/或版本唯一标识。对于 GitHub，修订由其提交 ID 唯一标识。

服务角色：一个 IAM 角色，它向Amazon服务，以便它可以访问Amazon资源的费用。附加到服务角色的策略决定了Amazon服务可以访问的资源以及它可以使用这些资源执行的操作。对于 CodeDeploy，服务角色用于以下内容：

- 读取应用于实例的标签或与实例关联的 Amazon EC2 Auto Scaling 组名称。这使 CodeDeploy 能够识别它可以将应用程序部署到的实例。
- 对实例、Amazon EC2 自动扩展组和 Elastic Load Balancing 器执行操作。
- 向 Amazon SNS 主题发布信息，以便在发生指定的部署或实例事件时发送通知。

- 检索有关 CloudWatch 警报的信息，以便为部署设置警报监控。

有关更多信息，请参阅 第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

目标版本：您上传到存储库并且想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。

有关 CodeDeploy 工作流中的其他组件的信息，请参阅以下主题：

- 选择 CodeDeploy 存储库类型 (p. 246)
- Deployments (p. 11)
- Application Specification Files (p. 24)
- Instance Health (p. 203)
- 使用 CodeDeploy 代理 (p. 141)
- Working with On-Premises Instances (p. 179)

CodeDeploy

本主题提供 CodeDeploy 中部署的组件和工作流的相关信息。部署过程因计算平台或部署方法（Lambda、Amazon ECS、EC2/ 本地或通过Amazon CloudFormation），您可以在部署中使用。

> Note
>
> 亚太地区（大阪）区域中不支持 Amazon ECS 计算平台上的部署。

主题

- 上的部署AmazonLambda 计算平台 (p. 11)
- Amazon ECS 计算平台上的部署 (p. 13)
- EC2/本地计算平台上的部署 (p. 20)

上的部署AmazonLambda 计算平台

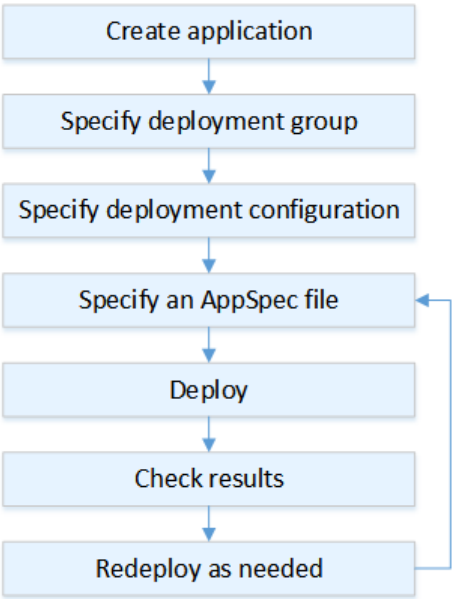
本主题提供了有关使用的 CodeDeploy 部署的组件和工作流程的信息。AmazonLambda 计算平台。

主题

- 上的部署工作流Amazon Lambda计算平台 (p. 11)
- 上传应用程序修订 (p. 12)
- 创建应用程序和部署组 (p. 12)
- 部署您的应用程序修订 (p. 13)
- 更新应用程序 (p. 13)
- 停止和失败的部署 (p. 13)
- 重新部署和部署回滚 (p. 13)

上的部署工作流Amazon Lambda计算平台

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，请选择 AmazonLambda 计算平台，当您创建应用程序。CodeDeploy 在部署期间使用此名称以确保引用正确的部署组件，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 CodeDeploy 创建应用程序 (p. 212)。
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如何将流量从原始Amazon Lambda函数版本更改为新的 Lambda 函数版本。有关更多信息，请参阅View Deployment Configuration Details (p. 210)。
4. 上传应用程序规范文件（AppSpec 文件）到 Amazon S3。AppSpec 文件指定 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，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。
5. 将应用程序修订部署到部署组。Amazon CodeDeploy部署您指定的 Lambda 函数修订。流量使用您在创建应用程序时选择的部署 AppSpec 文件转移到 Lambda 函数修订。有关更多信息，请参阅使用 CodeDeploy 创建部署 (p. 254)。
6. 检查部署结果。有关更多信息，请参阅监控 CodeDeploy 中的部署 (p. 281)。

上传应用程序修订

将 AppSpec 文件放入 Amazon S3 中或将其直接输入到控制台或Amazon CLI. 有关更多信息，请参阅Application Specification Files (p. 24)。

创建应用程序和部署组

上的 CodeDeploy 部署组AmazonLambda 计算平台可以标记一个或多个 AppSpec 文件的集合。每个 AppSpec 文件都可以部署一个 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。

部署您的应用程序修订

现在，您已准备好将 AppSpec 文件中指定的函数修订部署到部署组。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

您可以更新应用程序，然后使用 CodeDeploy 控制台或调用创建部署命令推送修订版本。

停止和失败的部署

可以使用 CodeDeploy 控制台或停止部署命令可停止部署。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 。ErrorInformation和常见错误中的Amazon CodeDeployAPI 参考。

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 (p. 147)和查看代表部署EC2/本地部署的日志数据 (p. 268)。

重新部署和部署回滚

CodeDeploy 通过将以前部署的版本重新部署为新的部署来实现回滚。

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 (p. 271)。

Amazon ECS 计算平台上的部署

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

主题

- 在开始 Amazon ECS 部署之前 (p. 14)
- Amazon ECS 计算平台上的部署工作流（高级） (p. 15)
- 亚马逊云服务器部署过程中会发生什么 (p. 16)
- 上传应用程序修订 (p. 19)
- 创建应用程序和部署组 (p. 19)
- 部署您的应用程序修订 (p. 19)
- 更新应用程序 (p. 19)
- 停止和失败的部署 (p. 19)
- 重新部署和部署回滚 (p. 20)
- 通过 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 20)

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。创建部署组时指定了一些要求，并且某些要求在 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
Application Load Balancer 或 Network Load Balancer	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，您需要指定包含 Amazon ECS 服务的 Amazon ECS 集群。有关更多信息，请参阅 。Amazon ECS 集群中的Amazon Elastic Container Service 用户指南.

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。您的 Amazon ECS 服务必须已启用，以便 CodeDeploy。默认情况下，亚马逊云服务器部署启用 Amazon 云服务器服务。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 ECS 服务中的Amazon Elastic Container Service 用户指南.

Application Load Balancer 或 Network Load Balancer

您必须将 Elastic Load Balancing 与要随 Amazon ECS 部署一起更新的 Amazon ECS 服务配合使用。您可以使用应用程序负载均衡器或 Network Load Balancer。我们建议使用 Application Load Balancer，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。在创建 CodeDeploy 应用程序的部署组时，您需要指定负载均衡器。有关更多信息，请参阅 。为 CodeDeploy Amazon 云服务器部署设置负载均衡器、目标组和侦听器 (p. 230)和创建负载均衡器中的Amazon Elastic Container Service 用户指南.

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 。应用程序负载均衡器的侦听器中的Elastic Load Balancing 用户指南和创建服务中的Amazon Elastic Container Service 用户指南.

两个亚马逊云服务器目标组

目标组 用于将流量路由到一个注册目标。Amazon ECS 部署需要两个目标组：一个用于您的 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 会创建替换任务

集，并将流量从原始任务集重新路由到新的任务集。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 会确定哪个目标组与 Amazon ECS 服务中的任务集相关联，该任务集的状态为PRIMARY（这是原始任务集），并将一个目标组与其关联，然后将另一个目标组与替换任务集关联。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 。应用程序负载均衡器的目标组中的Elastic Load Balancing 用户指南.

Amazon ECS 任务定义

A任务定义需要运行包含 Amazon ECS 应用程序的 Docker 容器。您需要在 CodeDeploy 应用程序的 AppSpec 文件中指定任务定义的 ARN。有关更多信息，请参阅 。Amazon ECS 任务定义中的Amazon Elastic Container Service 用户指南和 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329).

Amazon ECS 应用程序的容器

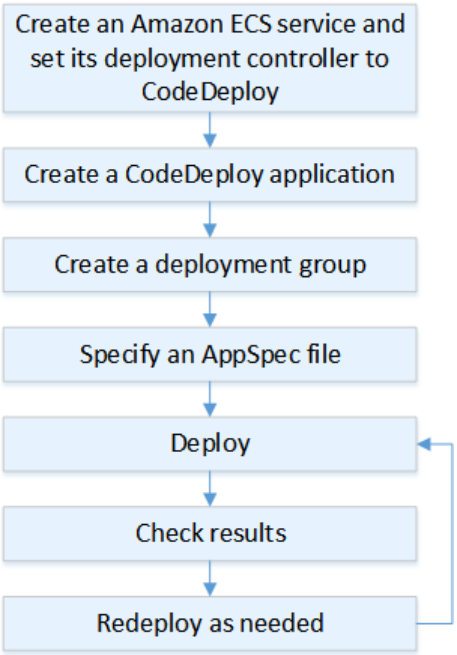
Docker 容器 是一套打包代码及其依赖项以便应用程序运行的软件。容器可以隔离您的应用程序，使其在不同的计算环境中运行。负载均衡器将流量定向到 Amazon ECS 应用程序任务集中的容器。您需要在 CodeDeploy 应用程序的 AppSpec 文件中指定容器的名称。AppSpec 文件中指定的容器必须是 Amazon ECS 任务定义中指定的容器之一。有关更多信息，请参阅 。Amazon Elastic Container Service中的Amazon Elastic Container Service 用户指南和 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329).

您的替换任务集的端口

在您的 Amazon ECS 部署期间，您的负载均衡器会将流量引导到此端口在您的 CodeDeploy 应用程序的 AppSpec 文件中指定的容器上。您需要在 CodeDeploy 应用程序的 AppSpec 文件中指定端口。有关更多信息，请参阅 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329)。

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。



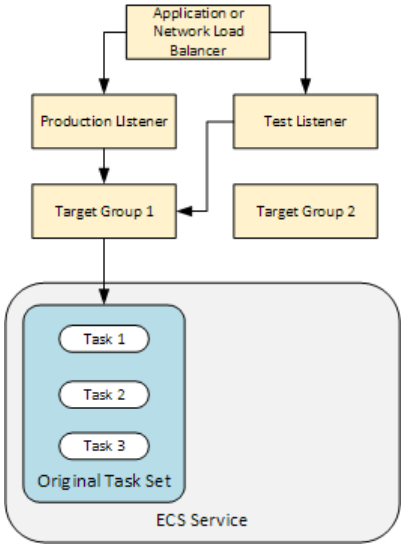
这些步骤包括：

1. 通过指定一个唯一表示要部署的内容的名称，创建 Amazon CodeDeploy 应用程序。要部署 Amazon 云服务器应用程序，请在Amazon CodeDeploy应用程序中，选择 Amazon 云服务器计算平台。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、流量重新路由行为和应用程序修订。有关更多信息，请参阅使用 CodeDeploy 创建应用程序 (p. 212)。
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您的 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）。您可以将它上传到 Amazon S3，以 YAML 或 JSON 格式将它输入控制台，或者使用Amazon CLI或 SDK。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 Lambda 函数在部署生命周期挂钩之后运行。容器名称必须是 Amazon ECS 任务定义中的容器。有关更多信息，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。
4. 部署应用程序修订。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换任务集。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 CodeDeploy 创建部署 (p. 254)。
5. 检查部署结果。有关更多信息，请参阅监控 CodeDeploy 中的部署 (p. 281)。

亚马逊云服务器部署过程中会发生什么

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 (p. 14)。

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。一些生命周期事件是挂钩，仅执行在 AppSpec 文件中指定的 Lambda 函数。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Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。

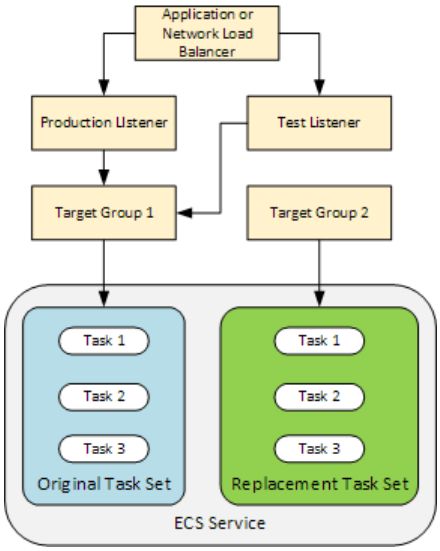
生命周期事件	生命周期事件操作
`BeforeInstall`（Lambda 函数的挂钩）	运行 Lambda 函数。

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

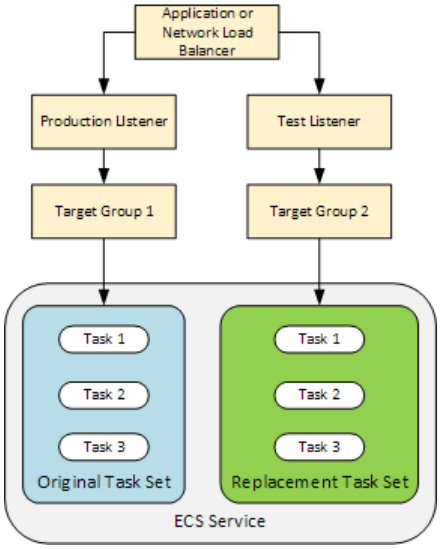
> **Note**
>
> 挂钩中的 Lambda 函数是可选的。

1. Lambda 在`BeforeInstall`挂钩中的 AppSpec 文件。
2. 在 `Install` 生命周期事件期间：
   a. Amazon ECS 服务中会创建替换任务集。
   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. Lambda 在`AfterInstall`挂钩中的 AppSpec 文件。
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. Lambda 在`AfterAllowTestTraffic`挂钩中的 AppSpec 文件。Lambda 函数可以使用测试流量验证部署。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，则您可以配置 CloudWatch 警报，以便在 Lambda 函数中的验证测试失败时触发回滚。

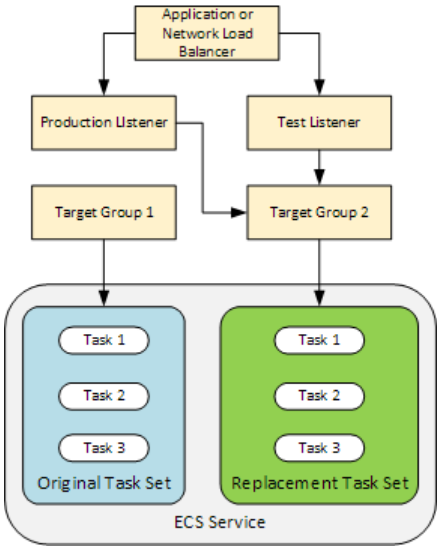
   验证测试完成后，将会发生以下情况之一：

   - 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。

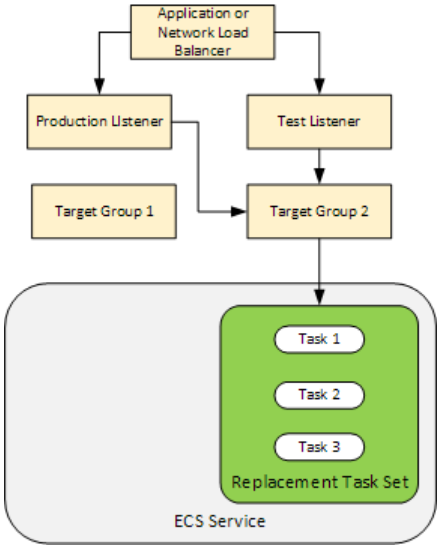
   有关更多信息，请参阅 使用 CodeDeploy 中的 CloudWatch 警报监控部署 (p. 283)、自动回滚 (p. 271) 和 为部署组配置高级选项 (p. 236)。

6. Lambda 在`BeforeAllowTraffic`挂钩中的 AppSpec 文件。

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. Lambda 在`AfterAllowTraffic`挂钩中的 AppSpec 文件。
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 中或将其直接输入到控制台或Amazon CLI. 有关更多信息，请参阅Application Specification Files (p. 24)。

创建应用程序和部署组

Amazon ECS 计算平台上的 CodeDeploy 部署组可以标识侦听器，以便处理传输到更新的 Amazon ECS 应用程序和部署期间使用的两个目标组的流量。部署组还定义一组配置选项，例如警报和回滚配置。

部署您的应用程序修订

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

您可以更新应用程序，然后使用 CodeDeploy 控制台或调用创建部署命令推送修订版本。

停止和失败的部署

您可以使用 CodeDeploy 控制台或停止部署命令可停止部署。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 。错误信息和常见错误中的Amazon CodeDeployAPI 参考。

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 (p. 271)。

通过 Amazon ECS 蓝/绿部署Amazon CloudFormation

您可以使用Amazon CloudFormation通过 CodeDeploy 管理 Amazon ECS 蓝/绿部署。有关更多信息，请参阅通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 264)。

> **Note**
>
> 使用管理 Amazon ECS 蓝/绿部署Amazon CloudFormation不在亚太地区（大阪）区域提供。

EC2/本地计算平台上的部署

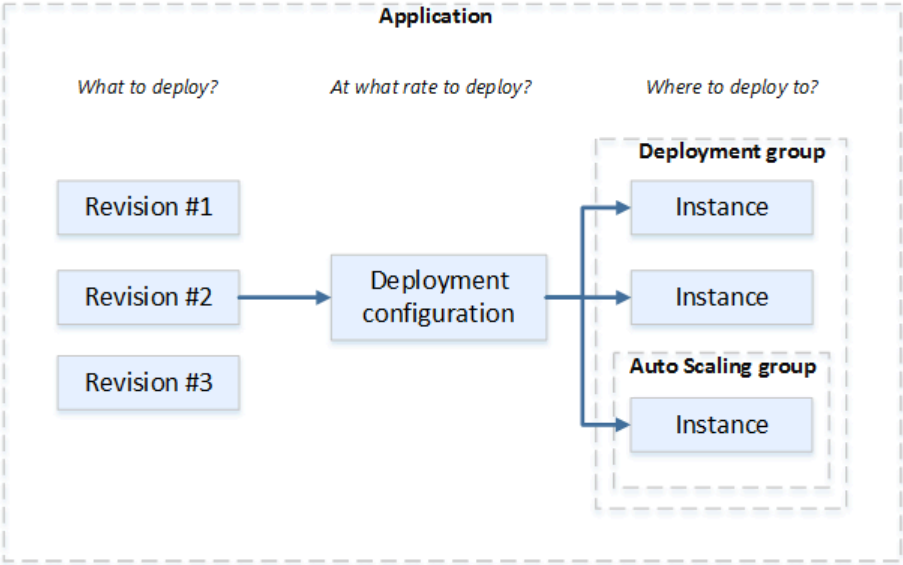
本主题提供了有关使用 EC2/ 本地计算平台的 CodeDeploy 部署的组件和工作流程的信息。有关蓝/绿部署的信息，请参阅蓝/绿部署概述 (p. 6)。

主题

- EC2/本地计算平台上的部署组件 (p. 20)
- EC2/本地计算平台上的部署工作流程 (p. 21)
- 设置实例 (p. 22)
- 上传应用程序修订 (p. 23)
- 创建应用程序和部署组 (p. 23)
- 部署您的应用程序修订 (p. 23)
- 更新应用程序 (p. 23)
- 停止和失败的部署 (p. 23)
- 重新部署和部署回滚 (p. 24)

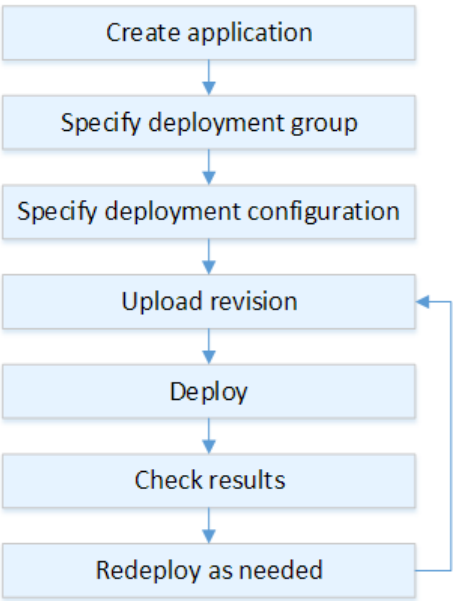
EC2/本地计算平台上的部署组件

下图显示了 EC2/本地计算平台上的 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指定唯一标识要部署的应用程序修订和表示应用程序的计算平台的名称。CodeDeploy 在部署期间使用此名称以确保引用正确的部署组件，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 CodeDeploy 创建应用程序 (p. 212)。
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。蓝/绿部署向负载均衡器注册部署组的一组替换实例并取消注册原始实例。

   您可以指定应用于实例的标签，或 Amazon EC2 Auto Scaling 组名称。

如果在部署组中指定一组标签，CodeDeploy 将部署到至少应用了一个指定标签的实例。如果您指定两个或多个标签组，CodeDeploy 将只部署到满足每个标签组条件的实例。有关更多信息，请参阅Tagging Instances for Deployments (p. 163)。

在所有情况下，实例必须配置为在部署中使用（即，它们必须已标记或属于 Amazon EC2 Auto Scaling 组），并且已安装并运行 CodeDeploy 代理。

我们为您提供Amazon CloudFormation模板，您可使用该模板基于亚马逊 Linux 或 Windows 服务器快速设置 Amazon EC2 实例。我们还提供了独立的 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 CodeDeploy 创建部署组 (p. 225)。

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 (p. 287)。
- 基于警报的部署管理。实现 Amazon CloudWatch 警报监控，以在指标超出或低于 CloudWatch 中设置的阈值时停止部署。
- 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 (p. 210)。
4. 将应用程序修订上传到 Amazon S3 或 GitHub。除了要部署的文件和要在部署期间运行的所有脚本外，您还必须包含应用程序规范文件（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，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。
5. 将应用程序修订部署到部署组。部署组中每个实例的 CodeDeploy 代理将您的应用程序修订从 Amazon S3 或 GitHub 复制到该实例。然后，CodeDeploy 代理将取消捆绑修订，并使用 AppSpec 文件将这些文件复制到指定的位置并执行任何部署脚本。有关更多信息，请参阅使用 CodeDeploy 创建部署 (p. 254)。
6. 检查部署结果。有关更多信息，请参阅监控 CodeDeploy 中的部署 (p. 281)。
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，您需要将修订后的源内容、所有部署脚本和 AppSpec 文件重新捆绑成一个新修订，然后将该修订上传到 Amazon S3 存储桶或 GitHub 存储库。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 CodeDeploy 创建部署 (p. 254)。

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。CodeDeploy 代理可以安装在 Amazon Linux、Ubuntu 服务器、RHEL 和 Windows 服务器实例上。
2. 如果要使用标签来标识部署组中的实例，请启用标记。CodeDeploy 依赖标签来标识实例并将其分组到 CodeDeploy 部署组。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. 启动附加有 IAM 实例配置文件的 Amazon EC2 实例。IAM 实例配置文件必须附加到 Amazon EC2 实例，以便 CodeDeploy 代理验证该实例的身份。
4. 创建服务角色。提供服务访问权，以便 CodeDeploy 可以展开Amazonaccount.

对于初始部署，Amazon CloudFormation 模板将为您完成所有这些操作。它基于已安装代理的 Amazon Linux 或 Windows 服务器创建并配置单个新 Amazon EC2 实例。有关更多信息，请参阅使用 CodeDeploy 的实例 (p. 162)。

Note

对于蓝/绿部署，您可以选择使用您已有的用于替换环境的实例，或者也可以选择让 CodeDeploy 为您预置新实例，以便在部署过程中选择。

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 (p. 24)。

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。上传存档文件（修订）添加到 Amazon S3 存储桶或 GitHub 存储库。

Note

Windows 服务器实例不支持 tar 和压缩 tar 归档文件格式（.tar 和 .tar.gz）。

创建应用程序和部署组

CodeDeploy 部署组基于实例的标签、Amazon EC2 Auto Scaling 组名称标识实例集合。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 应用程序中创建两种不同的部署组，从而使您能够选择应参与部署的一组或两组服务器。

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 (p. 163)。

部署您的应用程序修订

现在，您已准备好将您的应用程序修订从 Amazon S3 或 GitHub 部署到部署组。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

您可以更新应用程序，然后使用 CodeDeploy 控制台或调用创建部署命令推送修订版本。

停止和失败的部署

您可以使用 CodeDeploy 控制台或停止部署命令可停止部署。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，请参阅 。ErrorInformation和常见错误中的Amazon CodeDeployAPI 参考。

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。要查明部署失败的原因，可以使用 CodeDeploy 控制台调用获取部署实例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 (p. 147)和查看代表部署EC2/本地部署的日志数据 (p. 268)。

重新部署和部署回滚

CodeDeploy 通过将以前部署的版本重新部署为新的部署来实现回滚。

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 (p. 271)。

CodeDeploy AppSpec (应用程序规范) 文件

CodeDeploy 唯一的应用程序规范文件（AppSpec 文件）是YAML格式化或JSON格式化的文件。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 (p. 321)。

主题

- Amazon ECS 计算平台上的 AppSpec 文件 (p. 24)
- AppSpec 文件Amazon Lambda计算平台 (p. 24)
- EC2/本台计算平台上的AppSpec 文件 (p. 25)
- CodeDeploy 代理如何使用 AppSpec 文件 (p. 25)

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：

- Amazon ECS 服务的名称以及用于将流量定向到新任务集的容器名称和端口。
- 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Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)、 用于亚马逊云服务器部署的应用程序规格文件结构 (p. 322)和Amazon ECS 部署 AppSpec 文件示例 (p. 346)。

AppSpec 文件Amazon Lambda计算平台

如果您的应用程序使用AmazonLambda 计算平台上的 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：

- 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 用于 AppSpec 的 “hooks” 部分 AmazonLambda 部署 (p. 338)。

EC2/本台计算平台上的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 只运行在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 EC2/本部署的AppSpec “挂钩” 部分 (p. 340)。

CodeDeploy 代理如何使用 AppSpec 文件

在部署过程中，CodeDeploy 代理会在hooks部分中的 AppSpec 文件。如果找不到该事件，则代理 CodeDeploy 移到下一步。如果找到该事件，则 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
>
> CodeDeploy 代理未用于AmazonLambda 或亚马逊云服务器部署。

在安装事件时，CodeDeploy 代理将使用filesAppSpec 文件，以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。

有关代 CodeDeploy 日志文件的信息，请参阅使用 CodeDeploy 代理 (p. 141)。

开始使用 CodeDeploy

在首次使用 Amazon CodeDeploy 之前，必须完成设置步骤。

在开始时，您必须注册 Amazon 账户。要注册，请转至https://aws.amazon.com/然后选择创建Amazon账户.

然后，您可以继续执行本部分中的其余设置步骤。

主题

- 第 1 步：安装或升级，然后配置Amazon CLI (p. 26)
- 第 2 步：为 CodeDeploy 创建服务角色 (p. 26)
- 第 3 步：预配 IAM 用户 (p. 30)
- 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)

第 1 步：安装或升级，然后配置Amazon CLI

从中调用 CodeDeploy 命令Amazon CLI在本地开发计算机上，您必须安装Amazon CLI. 的版本 1.6.1 中的 CodeDeploy 命令首次提供Amazon CLI. 的 1.7.19 中提供了用于处理本地实例的 CodeDeploy 命令Amazon CLI.

如果您有旧版本的Amazon CLI安装完毕，您必须升级它，以使 CodeDeploy 命令可用。调用 aws --version 来查看版本。

安装或升级 Amazon CLI：

1. 按照安装 Amazon Command Line Interface 中的说明操作以安装或升级 Amazon CLI。
2. 配置Amazon CLI，请参阅配置Amazon Command Line Interface和管理 IAM 用户的访问密钥.

   Important

   在配置 Amazon CLI 时，系统将提示您指定 Amazon 区域。选择中列出的受支持区域之一区域和终端节点中的Amazon一般参考.
3. 要确认安装或升级，请从 Amazon CLI 调用以下命令：

   ```
   aws deploy help
   ```

   如果成功，此命令将显示可用的 CodeDeploy 命令的列表。

第 2 步：为 CodeDeploy 创建服务角色

在 Amazon 中，服务角色用于向 Amazon 服务授予权限，以便能访问 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 CodeDeploy 创建的服务角色必须授予对您的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权限。Amazon提供的政策：

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

- 阅读您实例上的标签或通过 Amazon EC2 Auto Scaling 组名称来识别您的 Amazon EC2 实例。
- 阅读、创建、更新和删除 Amazon EC2 Auto Scaling 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。
- 阅读和更新 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，请参阅 。第 2 步：创建服务角色 (p. 26)、为 Auto Scaling 组创建启动模板, 和启动模板支持中的Amazon EC2 Auto Scaling 用户指南.

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`**政策。该策略为您的服务角色提供针以下权限：

- 读取、更新和删除 Amazon ECS 任务集。
- 更新 Elastic Load Balancing) 目标组、侦听器和规则。
- 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`**政策。该策略为您的服务角色提供针以下权限：

- 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。

适用于Amazon对于 Lambda 部署，如果您想要允许向 Amazon SNS 发布，请附加**`AWSCodeDeployRoleForLambda`**政策。该策略为您的服务角色提供针以下权限：

- 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。

适用于Amazon对于 Lambda 部署，如果您想要限制对 Amazon SNS 的访问权限，请附加**`AWSCodeDeployRoleForLambdaLimited`**政策。该策略为您的服务角色提供针以下权限：

- 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 控制台创建服务角色，Amazon CLI，或者 IAM API。

主题

- 创建服务角色（控制台） (p. 28)
- 创建服务角色 (CLI) (p. 28)

- 获取服务角色 ARN（控制台） (p. 29)
- 获取服务角色 ARN (CLI) (p. 29)

创建服务角色（控制台）

1. 登录 Amazon Web Services Management Console，打开 IAM 控制台 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. 在存储库的创建角色页面上，选择Amazon服务，从中找到选择将使用此角色的服务。选择列表，选择CodeDeploy.
4. 在选择您的使用案例下，选择您的使用案例：
   - 对于 EC2/ 本地部署，选择CodeDeploy.
   - 对于 Amazon ECS 部署，请选择CodeDeploy-ECS.
   - 适用于Amazon选择 Lambda 部署，选择针对 Lambda CodeDeploy.
5. 选择 Next:。Permissions (下一步：权限)。
6. 在附加的权限策略页面上，会显示权限策略。选择 Next:。标签。
7. 在存储库的审核在页面上，Role name (角色名称)，输入服务角色的名称（例如，**`CodeDeployServiceRole`**)，然后选择创建角色.

   您还可以在中输入此服务角色的说明。角色描述.
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一些终端节点，则在角色列表中，浏览并选择您创建的角色，然后继续下一步。
9. 在信任关系选项卡上，选择编辑信任关系。
10. 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，请参阅创建角色以将权限委托给Amazon服务中的IAM 用户指南.

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) (p. 29)中的说明操作。

3. 调用attach-role-policy命令来指定名为的服务角色**CodeDeployServiceRole**基于名为的 IAM 托管策略的权限**AWSCodeDeployRole**. 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，请参阅为创建角色Amazon服务中的IAM 用户指南.

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 Management Console，打开 IAM 控制台 https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles（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 Amazon CLI 获取服务角色的 ARN，请对名为 **CodeDeployServiceRole** 的服务角色调用 get-role 命令：

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

返回值是服务角色的 ARN。

第 3 步：预配 IAM 用户

按照以下说明准备 IAM 用户以使用 CodeDeploy：

1. 创建 IAM 用户或使用与您的 IAM 用户关联的 IAM 用户Amazonaccount. 有关更多信息，请参阅 。创建 IAM 用户在IAM 用户指南.
2. 授予 IAM 用户对 CodeDeploy 的访问权限-Amazon通过复制以下策略并将其附加到 IAM 用户，向 IAM 用户提供服务和操作：

```
{
  "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::account-ID:role/CodeDeployServiceRole"
    }
  ]
}
```

在之前的策略中，替换*arn: aws: iam# ## ID: ##/Code ##ServicerRole*使用您在中创建的 CodeDeploy 服务角色的 ARN 值第 2 步：为 CodeDeploy 创建服务角色 (p. 26). 您可以在 IAM 控制台中的服务角色的详细信息页面中找到 ARN 值。

上述策略向 IAM 用户授予部署所需的访问权限：AmazonLambda 计算平台、EC2/本地计算平台和 Amazon ECS 计算平台。

要了解如何将策略附加到 IAM 用户，请参阅使用策略. 要了解如何将用户限制为使用一组有限的 CodeDeploy 操作和资源，请参阅适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297).

您可以使用Amazon CloudFormation本文档中提供的模板，用于启动与 CodeDeploy 兼容的 Amazon EC2 实例。使用Amazon CloudFormation要创建应用程序、部署组或部署配置的模板，您必须向 IAM 用户授予对的访问权限Amazon CloudFormation——Amazon服务和行动Amazon CloudFormation依赖于— 通过将其他策略附加到 IAM 用户：

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

有关这些语句中列出的其他 Amazon 服务的信息，请参阅：

- 概述Amazon Identity and Access Management策略
- 控制用户对负载均衡器的访问
- 控制对 Auto Scaling 资源的访问
- 控制Amazon CloudFormation使用Amazon Identity and Access Management

第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件

Note

如果您使用的是亚马逊云服务器或AmazonLambda 计算平台，请跳过此步骤。亚马逊云服务器部署部署了一个亚马逊云服务器服务，AmazonLambda 部署会部署一个无服务器的 Lambda 函数版本，因此不需要 Amazon EC2 实例的实例配置文件。

您的 Amazon EC2 实例需要权限才能访问用于存储应用程序的 Amazon S3 存储桶或 GitHub 存储库。要启动与 CodeDeploy 兼容的 Amazon EC2 实例，您必须创建附加 IAM 角色（实例配置文件。这些说明向您介绍如何创建 IAM 实例配置文件以附加到 Amazon EC2 实例。此角色为授予 CodeDeploy 访问用于存储应用程序的 Amazon S3 存储桶或 GitHub 存储库的权限。

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

Note

您可以将 IAM 实例配置文件附加到 Amazon EC2 实例（在启动该实例时）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

- 为创建 IAM 实例配置文件 (CLI) (p. 32)
- 为创建 IAM 实例配置文件（控制台） (p. 33)

为创建 IAM 实例配置文件 (CLI)

在这些步骤中，我们假定您已遵循开始使用 CodeDeploy (p. 26)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，这将允许 Amazon EC2 代表您工作：

```
{
    "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

我们建议您将此策略限制为您的 Amazon EC2 实例必须访问的 Amazon S3 存储桶。请确保授予对包含 CodeDeploy 代理的 Amazon S3 存储桶的访问权限。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:::replace-with-your-s3-bucket-name/*",
                "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
                "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
            ]
```

```
        }
    ]
}
```

Note

如果您想使用IAM 授权或 Amazon Virtual Private Cloud (VPC) 终端节点，您将需要添加更多权限。请参阅将代码与 Amazon Virtual Private Cloud 结合使用了解更多信息。

3. 从同一目录调用create-role命令创建命名为的 IAM 角色**CodeDeployDemo-EC2-Instance-Profile**，根据第一个文件中的信息：

   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. 调用attach-role-policy以授予该角色 Amazon EC2 Systems 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 的更多信息安装 CodeDeploy 代理。

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

6. 调用create-instance-profile命令，后跟add-role-to-instance-profile命令创建命名为的 IAM 实例配置文件**CodeDeployDemo-EC2-Instance-Profile**。实例配置文件允许 Amazon EC2 传递名为**CodeDeployDemo-EC2-Instance-Profile**当该实 Amazon EC2 首次启动时：

```
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，请参阅用于角色的列表实例配置文件的 IAM 部分Amazon CLI参考。

现在，您已创建要附加到 Amazon EC2 实例的 IAM 实例配置文件。有关更多信息，请参阅 。适用于 Amazon EC2 实例的 IAM 角色中的Amazon EC2 用户指南。

为创建 IAM 实例配置文件（控制台）

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：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。（如果 Get Started 按钮出现，选择此按钮，然后选择 Create Policy。）

3. 在创建策略页面上的 JSON 选项卡中粘贴以下内容：

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

Note

我们建议您将此策略限制为您的 Amazon EC2 实例必须访问的 Amazon S3 存储桶。请确保授予对包含 CodeDeploy 代理的 Amazon S3 存储桶的访问权限。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:::replace-with-your-s3-bucket-name/*",
                "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
                "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
            ]
        }
    ]
}
```

Note

如果您想使用IAM 授权或 Amazon Virtual Private Cloud (VPC) 终端节点，您将需要添加更多权限。请参阅将代码与 Amazon Virtual Private Cloud 结合使用了解更多信息。

4. 选择查看策略。
5. 在创建策略页面上，在**CodeDeployDemo-EC2-Permissions**策略名称框中键入 。
6. (可选) 对于描述，键入策略的描述。
7. 选择 Create Policy。
8. 在导航窗格中，选择角色，然后选择创建角色。
9. 在创建角色页面上，选择 Amazon 服务，然后从选择将使用此角色的服务列表中，选择 EC2。
10. 从选择您的使用案例列表中，选择 EC2。
11. 选择 Next:。Permissions (下一步：权限)。
12. 在策略列表中，选中刚创建的策略 (CodeDeployDemo-EC2-Permissions) 旁边的复选框。如有必要，请使用搜索框查找该策略。
13. 要使用 Systems Manager 安装或配置 CodeDeploy 代理，请选中 AmazonSSMManagedInstanceCore 旁边的框。此 Amazon 托管策略使实例能够使用 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 的更多信息安装 CodeDeploy 代理。

14. 选择 Next:。标签
15. 离开添加标签（可选）页面未更改，然后选择后续：审核。
16. 在存储库的审核页, 在Role name (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。

    您还可以输入此服务角色的描述：角色描述。

现在，您已创建要附加到 Amazon EC2 实例的 IAM 实例配置文件。有关更多信息，请参阅 。适用于 Amazon EC2 实例的 IAM 角色中的Amazon EC2 用户指南。

CodeDeploy 产品和服务集成

默认情况下，CodeDeploy 与许多Amazon服务以及合作伙伴产品和服务。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 (p. 36)
- 与合作伙伴的产品和服务集成 (p. 45)
- 来自社区的集成示例 (p. 49)

与其他Amazon服务

CodeDeploy 与以下内容集成Amazon服务：

Amazon CloudWatch	Amazon CloudWatch是面向的监控服务Amazon云资源以及您在上运行的应用程序Amazon. 您可以使用 Amazon CloudWatch 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 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，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 Event 会调用一个或多个目标操作。 了解更多： • 使用 Amazon CloudWatch Events 监控部署 (p. 284) • 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
Amazon EC2 Auto Scaling	CodeDeploy 支持Amazon EC2 Auto Scaling. 该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： • 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39) • 教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88) • 在引擎盖下：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 通过将应用程序的更新版本安装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。在 Amazon ECS 计算平台上进行部署.
Amazon CloudTrail	CodeDeploy 与Amazon CloudTrail. 此服务可在您的中捕获由 CodeDeploy 或代表其发出的 API 调用。Amazon并将日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 过 CodeDeploy 控制台，可捕获从 CodeDeploy 命令通过Amazon CLI，或者直接从 CodeDeploy API 中使用。通过使用 CloudTrail 收集的信息，您可以确定： • 向 CodeDeploy 提出了哪个请求。 • 已从中发出请求的源 IP 地址。 • 请发出了请求。 • 发出请求的时间。 了解更多： • Monitoring Deployments (p. 285)

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。有关使用的更多信息Amazon Cloud9使用 GitHub，请参阅的 GitHub 示例Amazon Cloud9. • 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 。使用Amazon Lambda中的函数Amazon Cloud9集成开发环境 (IDE). 有关 的更多信息Amazon Cloud9，请参阅什么是Amazon Cloud9和开始使用Amazon Cloud9.
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：测试版、伽玛和 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。 您可以配置Amazon CodePipeline要使用 CodeDeploy 进行部署： • 将代码部署到 Amazon EC2 实例和/或本地实例。 • Serverless (无服务器)AmazonLambda 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或在阶段的部署操作中使用创建管道向导。 了解更多： • Amazon有关 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。 • 简单的管道演练（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 (AmazonSAM) 是一个用于定义无服务器应用程序的模型。扩展Amazon CloudFormation提供一种简化的定义方法Amazon Lambda无服务器应用程序所需的函数、Amazon API Gateway API 和 Amazon DynamoDB 表。如果您已经使用AmazonSAM，您可以添加部署首选项，以便开始使用 CodeDeploy 来管理在 AmazonLambda 应用程序部署。 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，是一种服务，可跨多个 Amazon EC2 实例分发传入的应用程序流量。 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 (p. 42)

主题

- 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)
- 将 CodeDeploy 与 Elastic Load Balancing (p. 42)

将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成

CodeDeploy 支持 Amazon EC2 Auto Scaling，Amazon可根据您定义的条件自动启动 Amazon EC2 实例的服务。这些条件可以包括：在指定时间间隔内超过的 CPU 利用率、磁盘读写数或者入站或出站网络流量的限制。当不再需要实例时，Amazon EC2 Auto Scaling 将终止它们。有关更多信息，请参阅 。什么是 Amazon EC2 Auto Scaling？.

当新 Amazon EC2 实例作为 Amazon EC2 Auto Scaling 组一部分启动时，CodeDeploy 可自动将您的修订部署到这些新实例。您还可以将 CodeDeploy 中的部署与已向 Elastic Load Balancing 器注册的 Amazon EC2 Auto Scaling 实例进行协调。有关更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing (p. 42)和在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)。

> **Note**
>
> 请注意，如果您将多个部署组与单个 Amazon EC2 Auto Scaling 组相关联，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 。避免将多个部署组与一个 Amazon EC2 Auto Scaling 组关联 (p. 376)和在引擎盖下：CodeDeploy 和 Amazon EC2 Auto Scaling 集成.

主题

- 向 Amazon EC2 Auto Scaling 组 CodeDeploy 应用程序。 (p. 40)
- Amazon EC2 Auto Scaling 如何与 CodeDeploy 协作 (p. 40)
- 将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 结合使用 (p. 42)

向 Amazon EC2 Auto Scaling 组 CodeDeploy 应用程序。

要将 CodeDeploy 应用程序修订部署到 Amazon EC2 Auto Scaling 组：

1. 创建或找到允许 Amazon EC2 Auto Scaling 组使用 Amazon S3 的 IAM 实例配置文件。

   Note

   您还可以使用 CodeDeploy 将修订从 GitHub 存储库部署到 Amazon EC2 Auto Scaling 组。尽管 Amazon EC2 实例仍需要一个 IAM 实例配置文件，但该配置文件不需要任何额外的权限即可从 GitHub 存储库部署。有关更多信息，请参阅第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。
2. 创建或使用 Amazon EC2 Auto Scaling 组，并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 。Amazon EC2 实例上运行的应用程序的 IAM 角色.
3. 创建或找到允许 CodeDeploy 创建一个包含 Amazon EC2 Auto Scaling 组的部署组的服务角色。
4. 使用 CodeDeploy 创建一个部署组，指定 Amazon EC2 Auto Scaling 组名称和服务角色。
5. 使用 CodeDeploy 将您的修订部署到包含 Amazon EC2 Auto 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 Amazon EC2 Auto Scaling 组 (p. 88)。

Amazon EC2 Auto Scaling 如何与 CodeDeploy 协作

为了让 CodeDeploy 在 Auto Scaling 扩展事件期间将应用程序修订部署到新的 EC2 实例，CodeDeploy 使用了 Auto Scaling 生命周期钩子。生命周期钩子通知 CodeDeploy，Auto Scaling 向外扩展事件正在进行中，CodeDeploy 需要将修订部署到缩小的实例。

Note

在本节中，术语 “生命周期钩子” 和 “挂钩” 是指用于将 CodeDeploy 与 Auto Scaling 集成的钩子。这个钩子与中描述的钩子不同AppSpec 的 “hooks” 部分 (p. 335).

主题

- 生命周期挂钩是如何创建和安装的？ (p. 40)
- CodeDeploy 添加生命周期钩子后，它是如何使用的？ (p. 40)
- CodeDeploy 如何命名 Amazon EC2 Auto Scaling 组 (p. 41)
- 自定义周期挂钩事件的执行顺序 (p. 41)
- 部署期间的横向扩展事件 (p. 41)
- 中的活动顺序Amazon CloudFormationcfn-init 脚本 (p. 42)

生命周期挂钩是如何创建和安装的？

当您创建或更新部署组以包含 Auto Scaling 组时，CodeDeploy 会使用 CodeDeploy 服务角色访问 Auto Scaling 组，然后在 Auto Scaling 组中安装生命周期钩子。

CodeDeploy 添加生命周期钩子后，它是如何使用的？

安装生命周期挂钩后，它将在扩展事件期间使用。扩展事件展开如下：

1. Auto Scaling 服务（或简单地称为 Auto Scaling）确定需要发生扩展事件，并与 EC2 服务联系以启动新的 EC2 实例。
2. EC2 服务启动了一个新的 EC2 实例。该实例移动到`Pending`状态，然后进入`Pending:Wait`状态。
3. 在`Pending:Wait`，Auto Scaling 将运行附加到 Auto Scaling 组的所有生命周期钩子，包括 CodeDeploy 创建的生命周期钩子。

4. 生命周期钩子将通知发送给Amazon SQS 队列这是由 CodeDeploy 调查的。
5. 收到通知后，CodeDeploy 会解析消息，执行一些验证，并开始使用上次成功的修订版将应用程序部署到新的 EC2 实例。
6. 在部署运行期间，CodeDeploy 每五分钟向 Auto Scaling 发送一次心跳，以告知实例仍在处理中。
7. 到目前为止，EC2 实例仍在`Pending:Wait`状态。
8. 部署完成后，CodeDeploy 会向 Auto Scaling 指示`CONTINUE`或者`ABANDON`EC2 启动过程，具体取决于部署是成功还是失败。
   - 如果 CodeDeploy 表示`CONTINUE`，Auto Scaling 将继续启动过程，要么等待其他挂钩完成，或者将实例放入`Pending:Proceed`然后`InService`状态。
   - 如果 CodeDeploy 表示抛弃，Auto Scaling 将终止 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 中所定义）所需容量设置。

CodeDeploy 如何命名 Amazon EC2 Auto Scaling 组

在 EC2/本地计算平台上部署蓝绿期间，您可以通过两个选项将实例添加到替换（绿色）环境：

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

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

```
CodeDeploy_deployment_group_name_deployment_id
```

例如，如果有 ID 的部署`10`部署名为的部署组`alpha-deployments`，预配的 Amazon EC2 Auto Scaling 组命名为`CodeDeploy_alpha-deployments_10`. 有关更多信息，请参阅 为 EC2/本地蓝/绿部署（控制台）创建部署组 (p. 227) 和 GreenFleetProvisioningOption。

自定义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 CodeDeploy 部署到的 Amazon EC2 Auto Scaling 组。但是，这些自定义生命周期挂钩事件的执行顺序无法根据 CodeDeploy 默认部署生命周期事件预先确定。例如，如果添加名为的自定义生命周期钩子`ReadyForSoftwareInstall`对于 Amazon EC2 Auto Scaling 组，则无法提前知道它是在第一个 CodeDeploy 默认部署生命周期事件之前还是在最后一个该事件之后执行。

要了解如何将自定义生命周期挂钩添加到 Amazon EC2 Auto Scaling 组，请参阅。添加生命周期钩.

部署期间的横向扩展事件

如果在部署过程中发生 Amazon EC2 Auto Scaling 横向扩展事件，则新实例将使用最近部署的最新应用程序修订进行更新，而不是当前正在部署的应用程序修订。如果部署成功，则旧实例和最近扩展的实例将会托管不同的应用程序修订。为了使这些实例更新，CodeDeploy 会自动启动后续部署（在第一个实例之后立即开始）以更新任何过时的实例。如果您想更改此默认行为，以便过时的 EC2 实例留在较旧的修订版中，请参阅Automatic updates to outdated instances.

如果您想在部署期间暂停 Amazon EC2 Auto Scaling 横向扩展流程，则可以通过`common_functions.sh`用于使用 CodeDeploy 进行负载平衡的脚本。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 CodeDeploy Default.oneatime 部署配置支持此功能。

有关使用的更多信息`HANDLE_PROCS=true`要避免在使用 Amazon EC2 Auto Scaling 时出现部署问题，请参阅关于处理 AutoScaling 流程的重要在aws-code 部署示例（位于 GitHub 上）。

中的活动顺序Amazon CloudFormationcfn-init 脚本

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。

如果没有精心控制事件的顺序，那么 CodeDeploy 代理可能会在所有脚本运行完成之前启动部署。

要控制事件的顺序，请使用以下最佳实践之一：

- 通过安装 CodeDeploy 代理`cfn-init`脚本，将其放在所有其他脚本之后。
- 在自定义 AMI 中包括 CodeDeploy 代理，使用`cfn-init`脚本启动它，将其放在所有其他脚本之后。

有关使用的信息`cfn-init`，请参阅cfn-init在Amazon CloudFormation用户指南.

将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 结合使用

指定要在 Amazon EC2 Auto Scaling 组中启动新 Amazon EC2 实例时使用的基本 AMI 时，有下面两个选项：

- 您可以指定已安装 CodeDeploy 代理的自定义的基本 AMI。由于已安装代理，因此相比于另一个选项，此选项启动新的 Amazon EC2 实例的速度更快。但是，此选项提高了 Amazon EC2 实例的初始部署失败的可能性，尤其在 CodeDeploy 代理已过期的情况下。如果选择此选项，建议您定期更新自定义的基本 AMI 中的 CodeDeploy 代理。
- 您可以指定未安装 CodeDeploy 代理并在 Amazon EC2 Auto Scaling 组中启动每个新实例时安装代理的基本 AMI。尽管此选项相比于另一个选项启动新 Amazon EC2 实例的速度更慢，但是它提高了实例的初始部署成功的可能性。此选项使用最新版本的 CodeDeploy 代理。

将 CodeDeploy 与 Elastic Load Balancing

在 CodeDeploy 部署期间，负载均衡器会阻止 Internet 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。但是，负载均衡器的具体作用取决于它是用于蓝/绿部署还是就地部署。

> **Note**
>
> Elastic Load Balancing 负载均衡器的使用在蓝/绿部署中为必需，在就地部署中为可选。

Elastic Load Balancing 类型

Elastic Load Balancing 提供了三种类型的负载均衡器可用于 CodeDeploy 部署：经典负载均衡器、应用程序负载均衡器和网络负载均衡器。

传统负载均衡器

路由和负载均衡在传输层 (TCP/SSL) 或应用程序层 (HTTP/HTTPS) 进行。它支持 EC2-Classic 或 VPC。

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 Virtual Private Cloud (VPC) 中每个 EC2 实例或容器实例上的端口。

Note

应用程序负载均衡器目标组必须有一个目标类型：`instance`用于 EC2 实例上的部署，以及`IP`用于 Fargate 部署。有关更多信息，请参阅 。Target type.

网络负载均衡器

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。网络负载均衡器可以处理突发流量、保留客户端的源 IP 以及在负载均衡器的使用寿命内使用固定 IP。

要了解有关 Elastic Load Balancing 负载均衡器的更多信息，请参阅以下主题：

- 什么是 Elastic Load Balancing？
- 什么是 Classic Load Balancer？
- 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 负载均衡器之后重新路由实例流量是 CodeDeploy 蓝/绿部署的基础。

在蓝/绿部署期间，负载均衡器根据您指定的规则，允许将流量路由到已部署最新应用程序修订的部署组中的新实例 (替换环境)，然后阻止运行较早应用程序修订的旧实例的流量 (原始环境)。

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

对于蓝/绿部署，您可以在部署组中指定 Classic Load Balancer 器、应用程序负载均衡器或 Network Load Balancer。您可以使用 CodeDeploy 控制台或Amazon CLI将负载均衡器添加到部署组。

有关在蓝/绿部署中使用负载均衡器的更多信息，请参阅以下主题：

- 在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)
- 为蓝/绿部署创建应用程序（控制台） (p. 215)
- 为 EC2/本地蓝/绿部署（控制台）创建部署组 (p. 227)

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。在将 Elastic 负载均衡器用于就地部署时，部署组中的实例将从负载均衡器取消注册，更新为最新的应用程序修订，并在部署完成后向负载均衡器重新注册同一部署组。已成功。CodeDeploy 将等待长达 1 个小时，以使实例在负载均衡器后面保持运行状态。如果在等待期间，负载均衡器未将实例标记为运行状况良好，则根据部署配置，CodeDeploy 将移动到下一个实例或部署失败。

对于就地部署，可以指定 Classic Load Balancer、Application Load Balancer 或 Network Load Balancer。您可以在配置部署组的过程中指定负载均衡器，或使用 CodeDeploy 提供的脚本实施负载均衡器。

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）(p. 213)
- 为就地部署创建部署组（控制台）(p. 225)
- 在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)

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**
>
> 您应该仅在使用脚本为就地部署设置负载均衡器时使用 CodeDeployDefault.OneAtATime 部署配置。不支持并发运行，并且 CodeDeployDefault.OneAtATime 设置可确保脚本的顺序执行。有关部署配置的更多信息，请参阅在 CodeDeploy 中使用部署配置 (p. 206)。

在 GitHub 上的 CodeDeploy 示例存储库中，我们提供了说明和示例，您可进行调整以使用 CodeDeploy Elastic Load Balancing ad Balancer。这些存储库包括三个示例脚本—`register_with_elb.sh`、`deregister_from_elb.sh`, 和`common_functions.sh`— 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。

要使用已注册到 Elastic Load Balancing 负载均衡器的 Amazon EC2 实例在 CodeDeploy 中设置就地部署，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：
   - 传统负载均衡器
   - 应用程序负载均衡器或 Network Load Balancer（同一脚本可用于任一类型）
2. 请确保每个目标 Amazon EC2 实例均具有Amazon CLI已安装。
3. 请确保您的每个目标 Amazon EC2 实例均已附加至少一个 IAM 实例配置文件，并且具有 elasticload Balancing: * 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`对于应用程序修订版，请提供 CodeDeploy 运行`register_with_elb.sh`在ApplicationStart事件和`deregister_from_elb.sh`在ApplicationStopevent.
6. 如果实例属于 Amazon EC2 Auto Scaling 组，则可跳过此步骤。

   在 `common_functions.sh` 脚本中：

   - 如果您使用传统负载均衡器中指定 Elastic Load Balancing 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任何更改。
   - 如果您使用应用程序负载均衡器或 Network Load Balancer中，指定中的 Elastic Load Balancing 目标组名称的名称`TARGET_GROUP_LIST=""`，并对文件中的其他部署设置进行所需的任何更改。
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。将该修订部署到 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取消注册 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。

与合作伙伴的产品和服务集成

CodeDeploy 内置集成了下列合作伙伴产品和服务：

Ansible	如果您已经有一套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 — 竹子和 Bitbucket	的 CodeDeploy 任务竹将包含 AppSpec 文件的目录压缩为 .zip 文件，将文件上传到 Amazon S3，然后根据 CodeDeploy 应用程序中提供的配置启动部署。 Atlassian Bitbucket 支持 CodeDeploy 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，您无需登录到持续集成 (CI) 平台或 Amazon EC2 实例即可运行手动部署过程。 了解更多： • 为竹子使用 CodeDeploy 任务 • 宣布 Atlassian Bitbucket 支持 CodeDeploy
Chef	Amazon提供两个用于集成的模板示例Chef和 CodeDeploy. 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 第二个示例模板演示如何使用 CodeDeploy 在每个节点上协调说明书和方法的运行。 了解更多： • CodeDeploy
CircleCI	CircleCI 提供了一个自动测试和持续集成以及部署工具集。在中创建 IAM 角色Amazon要使用 CircleCI 并在 circle.yml 文件中配置部署参数，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： • CodeDeploy 持续部署
CloudBees	你可以使用 CodeDeploy Jenkins 插件，可在CloudBeesDEV @cloud，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： • CodeDeploy 詹金斯插件现已在 DEV @cloud 上提供
Codeship	您可以使用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 • CodeDeploy 在 Codeship 上集成
GitHub	可以使用 CodeDeploy 从部署应用程序修订GitHub存储库。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： • GitHub 将 CodeDeploy 集成 (p. 47) • 教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101) • 使用 CodeDeploy 从 GitHub 自动部
HashiCorp Consul	在 CodeDeploy 中部署应用程序时，可以使用开源 HashiCorp Consul 工具来帮助确保应用程序环境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： • CodeDeploy 与 HashiCorp 领事进行部署
Jenkins	CodeDeployJenkins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 设置詹金斯插件
Puppet Labs	Amazon为提供示例模板Puppet和 CodeDeploy. 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 第二个示例模板演示如何使用 CodeDeploy 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SaltStackCodeDeploy 的基础设施。您可以使用 CodeDeploy 模块在您的召唤上安装和运行 CodeDeploy 代理，或使用几个简单的部署挂钩来协调 Salt State 的运行。 了解更多： • SaltStack 和 CodeDeploy
TeamCity	您可以使用 CodeDeploy Runner 插件直接从 TeamCity 部署应用程序。该插件会增加 TeamCity 构建步骤，准备应用程序修订并上传到 Amazon S3 存储桶、在 CodeDeploy 应用程序中注册修订、创建 CodeDeploy 部署，然后等待部署完成（如果您选择）。 了解更多： • CodeDeploy Runner（下载） • CodeDeploy Runner 插件（文档）
Travis CI	您可以配置Travis CI以在成功构建后触发在 CodeDeploy 中进行的部署。 了解更多： • Travis CI 和 CodeDeploy 部署

主题

• GitHub 将 CodeDeploy 集成 (p. 47)

GitHub 将 CodeDeploy 集成

CodeDeploy 支持GitHub，一项基于网络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 CodeDeploy 仅支持 GitHub 进行 EC2/本地部署。

主题

• 从 GitHub 部署 CodeDeploy 修订 (p. 47)
• CodeDeploy 的 GitHub 行为 (p. 48)

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，请按照规划 CodeDeploy 的修订版 (p. 239)和将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)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. 使用创建部署CodeDeploy 控制台中的页面或Amazon CLI create-deployment命令以将您的修订从 GitHub 存储库部署到已配置为在 CodeDeploy 部署中使用的目标实例。

如果你想打电话给create-deployment命令，您必须首先使用创建部署页，用于向 CodeDeploy 授予代表指定应用程序的首选 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 CodeDeploy 创建部署 (p. 254)。

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) (p. 262)。

要了解如何准备实例以在 CodeDeploy 部署中使用，请参阅使用 CodeDeploy 的实例 (p. 162).

有关更多信息，请参阅教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101)。

CodeDeploy 的 GitHub 行为

主题

- 使 GitHub CodeDeploy 中的应用程序进 (p. 48)
- 将 CodeDeploy 与私有和公有 GitHub 存储库 (p. 49)
- CodeDeploy 与组织管理的 GitHub 仓库交互 (p. 49)
- 使用 CodePipeline 自动 CodeDeploy Pipeline (p. 49)

使 GitHub CodeDeploy 中的应用程序进

在向提供 CodeDeploy 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。

在 CodeDeploy 中将 GitHub 帐户链接到应用程序

1. 登录到Amazon Web Services Management Console在上打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在中使用的同一账户或 IAM 用户信息登录开始使用 CodeDeploy (p. 26).
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 CodeDeploy 创建部署组 (p. 225)。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. 请执行下列操作之一：

   - 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。该网页将提示您授权 CodeDeploy 与您的应用程序的 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。该网页将提示您授权 CodeDeploy 与您的应用程序的 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 Authorize application。GitHub 向提供 CodeDeploy 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. 登录到GitHub使用您要撤消的 GitHub 账户的凭证Amazon CodeDeploy许可。
2. GitHub 打开应用程序页面，找到CodeDeploy然后按照 GitHub 过程撤消应用程序的授权。

将 CodeDeploy 与私有和公有 GitHub 存储库

CodeDeploy 支持从私有和公有 GitHub 存储库部署应用程序。当您向提供代表您访问 GitHub 的权限后，CodeDeploy 将拥有对您的 GitHub 账户有权访问的所有私有 GitHub 存储库的读写访问权。但是，CodeDeploy 只能从 GitHub 存储库读取。它将不会对任何私有 GitHub 存储库进行写入。

CodeDeploy 与组织管理的 GitHub 仓库交互

默认情况下，组织（而不是账户自己的私有或公有存储库）的 GitHub 存储库不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 CodeDeploy 权单个账户访问：
  - 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申请.
  - 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。想要了解有关信息，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 。关于第三方申请限制.

使用 CodePipeline 自动 CodeDeploy Pipeline

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帖子、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

- 在中自动执行 CodeDeploy 配置Amazon CloudFormation

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

  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 插件将 Web 应用程序部署到Amazon直接来自 Eclipse 开发环境。

  发布时间：2015 年 2 月
- 使用 CodeDeploy 从 GitHub 自动部

  了解如何使用从 GitHub 到 CodeDeploy 的自动部署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 CodeDeploHelloWorldApp Deploy Deploy Deploy Deploy-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（亚马逊 Linux 或红帽企业 Linux 和 Linux、macOS 或 Unix）(p. 51)
- 教程：部署一个 “你好，世界！” 应用程序与 CodeDeploy（Windows 服务器）(p. 68)
- 教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)
- 教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88)
- 教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101)
- 教程：部署 Amazon ECS 服务 (p. 111)
- 教程：通过验证测试部署 Amazon ECS 服务 (p. 117)
- 教程：使 Lambda 码部署和Amazon无服务器应用程序模型 (p. 124)

教程：将 WordPress 部署到 Amazon EC2 实例（亚马逊 Linux 或红帽企业 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）部署到运行亚马逊 Linux 或红帽企业 Linux (RHEL) 的单个 Amazon EC2 实例。

不是您要找的内容？

- 要改为实施针对运行 Windows 服务器的 Amazon EC2 实例的部署，请参阅教程：部署一个 “你好，世界！” 应用程序与 CodeDeploy（Windows 服务器）(p. 68)。
- 要实施针对本地实例而非 Amazon EC2 实例的部署，请参阅教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)。

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，您必须完成开始使用 CodeDeploy (p. 26)中的先决条件。这包括配置您的 IAM 用户账户、安装或升级Amazon CLI，并创建 IAM 实例配置文件和服务角色。

主题

- 第 1 步：启动和配置 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例 (p. 52)
- 第 2 步：将源内容配置为部署到亚马逊 Linux 或红帽企业 Linux Amazon EC2 实例 (p. 53)
- 第 3 步：将 WordPress 应用程序上传到 Amazon S3 (p. 57)
- 第 4 步：部署您的 WordPress 应用程序 (p. 59)
- 第 5 步：更新和重新部署您的 WordPress 应用程序 (p. 63)
- 第 6 步：清理您的 WordPress 应用程序和相关资源 (p. 66)

第 1 步：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，您需要运行亚马逊 Linux 或红帽企业 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 (p. 170)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，请确保在**Name**的标签值**CodeDeployDemo**。（如果您指定不同的标签密钥或标签值，则第 4 步：部署您的 WordPress 应用程序 (p. 59)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。请勿继续执行使用 CodeDeploy 创建应用程序 (p. 212)作为下一步骤。

Connect 到您的亚马逊 Linux 或 RHEL Amazon EC2 实例

启动新 Amazon EC2 实例之后，请按照下面的说明来练习连接操作。

1. 使用ssh命令（或支持 SSHS 的终端仿真器，如PuTTY）连接到您的亚马逊 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 key pair 的私有密钥。有关更多信息，请参阅 。连接到您的实例。

   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 key pair 名为**codedeploydemo.pem**，您可以键入：

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

   Replace*/path/to/*codedeploydemo.pem的路径.pem文件和示例 DNS 地址及其地址传输到您的 Amazon Linux 或 RHEL Amazon EC2 实例的示例 DNS 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，使用chmod命令、macOS 或 Unix 上，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，它应如下所示：

   ```
   
          __|  __|_  )
          _|  (     /   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。

添加允许 HTTP 流量到您的 Amazon Linux 或 RHEL 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 Management Console，然后通过以下网址打开 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. 在存储库的说明选项卡，在安全组中，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. UNDER安全组下，为您的 Amazon EC2 实例选择安全组。它可能被命名为**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。如果您的实例的安全组配置正确，您应会看到规则具有以下值：

   - 类型：HTTP
   - 协议：TCP
   - 端口范围：80
   - 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将它们添加到新的安全规则中。

第 2 步：将源内容配置为部署到亚马逊 Linux 或红帽企业 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 (p. 53)
- 创建脚本以运行应用程序 (p. 54)
- 添加应用程序规范文件 (p. 56)

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）(p. 54)
- 获取 WordPress 源代码的副本 (Git) (p. 54)

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 `/tmp/WordPress/scripts` 中创建 `change_permissions.sh` 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 可在第 5 步：更新和重新部署您的 WordPress 应用程序 (p. 63)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，添加应用程序规范文件（AppSpec 文件）、YAMLCodeDeploy 使用的格式文件来执行以下操作：

- 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。

必须将 AppSpec 文件命名为`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 文件复制`/tmp/WordPress`文件夹添加到`/var/www/html/WordPress`文件夹中的 Amazon EC2 夹。在部署过程中，CodeDeploy 会将指定的脚本作为`root`中的`/var/www/html/WordPress/scripts`文件夹中的部署生命周期内的指定事件中的文件夹中，Amazon EC2 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 (p. 321)。

> **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 (p. 349)。

第 3 步：将 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 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GitHub 将 CodeDeploy 集成 (p. 47)。

主题

- 配置 Amazon S3 存储桶 (p. 57)
- 为存储桶准备应用程序的文件 (p. 58)
- 将应用程序的文件捆绑到单个存档文件中，然后推送归档文件 (p. 59)

配置 Amazon S3 存储桶

创建存储容器或存储桶（或使用现有存储桶）。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。

您可以将Amazon CLI、Amazon S3 控制台或 Amazon S3 API 创建 Amazon S3 存储桶。创建存储桶后，请确保为存储桶和您的 IAM 用户赋予访问权限。

> **Note**
>
> 存储桶名称在 Amazon S3 中必须是唯一的Amazon账户。如果您无法使用 **`codedeploydemobucket`**，请尝试其他存储桶名称，例如 **`codedeploydemobucket`** 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 **`codedeploydemobucket`** 替换为您的存储桶名称。
> Amazon S3 存储桶必须在同一Amazon区域，该区域启动您的目标 Amazon EC2 实例。例如，如果您在美国东部（弗吉尼亚北部）区域中创建存储桶，则必须在美国东部（弗吉尼亚北部）区域中启动目标 Amazon EC2 实例。

主题

- 创建 Amazon S3 存储桶 (CLI) (p. 57)
- 创建 Amazon S3 存储桶（控制台） (p. 57)
- 向 Amazon S3 存储桶和您的 IAM 用户赋予权限 (p. 58)

创建 Amazon S3 存储桶 (CLI)

调用mb命令创建名为的 Amazon S3 存储桶**`codedeploydemobucket`**：

```
aws s3 mb s3://codedeploydemobucket
```

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。

向 Amazon S3 存储桶和您的 IAM 用户赋予权限

您必须有权上传到 Amazon S3 存储桶。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 Amazon S3 存储桶中的任何名为`codedeploydemobucket`：

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

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 存储桶策略指定此操作。例如，在以下 Amazon S3 存储桶策略中，使用通配符 (*) 允许任何带有包含 ARN 的 IAM 实例配置文件的 Amazon EC2 实例`arn:aws:iam::444455556666:role/CodeDeployDemo`从 Amazon S3 存储桶中任何名为`codedeploydemobucket`：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "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 策略的信息，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...)
```

将应用程序的文件捆绑到单个存档文件中，然后推送归档文件

将 WordPress 应用程序文件和 AppSpec 文件捆绑到存档文件（称为应用程序）中修订)。

> **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 /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 CodeDeploypush命令将文件捆绑在一起，将修订上传到 Amazon S3，然后向 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://codedeploydemobucket/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`**，将修订上传到**`codedeploydemobucket`**存储桶，并在 CodeDeploy 中注册有关上传修订的信息。

第 4 步：部署您的 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。您可以将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 (p. 60)
- 对部署进行监控和排除故障 (p. 62)
- 验证部署 (p. 62)

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) (p. 60)
- 部署应用程序修订（控制台）(p. 61)

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 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) (p. 28)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) (p. 29)。
2. 现在您已拥有服务角色 ARN，请调用create-deployment-group命令创建名为的部署组**WordPress_DepGroup**，与名为的应用程序关联**WordPress_App**，使用名为的 Amazon EC2 标签**CodeDeployDemo**和名为的部署配置**CodeDeployDefault.OneAtATime**：

```
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**
   >
   > 这些区域有：创建部署组命令支持创建触发器，这些触发器将导致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以在满足 Amazon CloudWatch 警报中的监控阈值时停止部署的选项。本教程不包括这些操作的命令。
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，请参阅是什么Amazon Systems Manager。
4. 现在调用 create-deployment 命令，使用名为 **codedeploydemobucket** 的存储桶中名为 **WordPressApp.zip** 的应用程序修订，创建一个与名为 **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=codedeploydemobucket,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 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） (p. 28)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） (p. 29)。
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：

   登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 _ 应用程序。
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. In环境配置中，选择Amazon EC2 实例。
9. In代理配置Amazon Systems Manager，请保留默认值。
10. 在键中，输入 **`Name`**。
11. 在值中，输入 **`CodeDeployDemo`**。

    Note

    在您键入**`CodeDeployDemo`**，a1应显示在匹配的实例以确认 CodeDeploy 找到了一个匹配的 Amazon EC2 实例。
12. In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. 旁边存储库类型中，选择我的应用程序存储在 Amazon S3 中。In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，选择代码部署演示存储桶（或您上传您的应用程序修订的存储桶的名称）。
    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/codedeploydemobucket/WordPressApp.zip`
    e. 返回到 CodeDeploy 控制台，然后在修订位置中，粘贴Link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. Expand部署组覆盖，来自的部署配置中，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。

对部署进行监控和排除故障

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) (p. 62)
- 监视您的部署和故障排除（控制台） (p. 62)

监视您的部署并排除故障 (CLI)

1. 针对名为 **WordPress_App** 的应用程序和名为 **WordPress_DepGroup** 的部署组调用 list-deployments 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`。

   如果整体状态为`Failed`，则您可以调用诸如列表部署实例和获取部署实例排除的故障。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 (p. 373)。

监视您的部署和故障排除（控制台）

在存储库的部署页 CodeDeploy，您可以在状态column.

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 (p. 202)中描述方法类似的方法。您还可以分析 Amazon EC2 实例上的部署日志文件。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 (p. 373)。

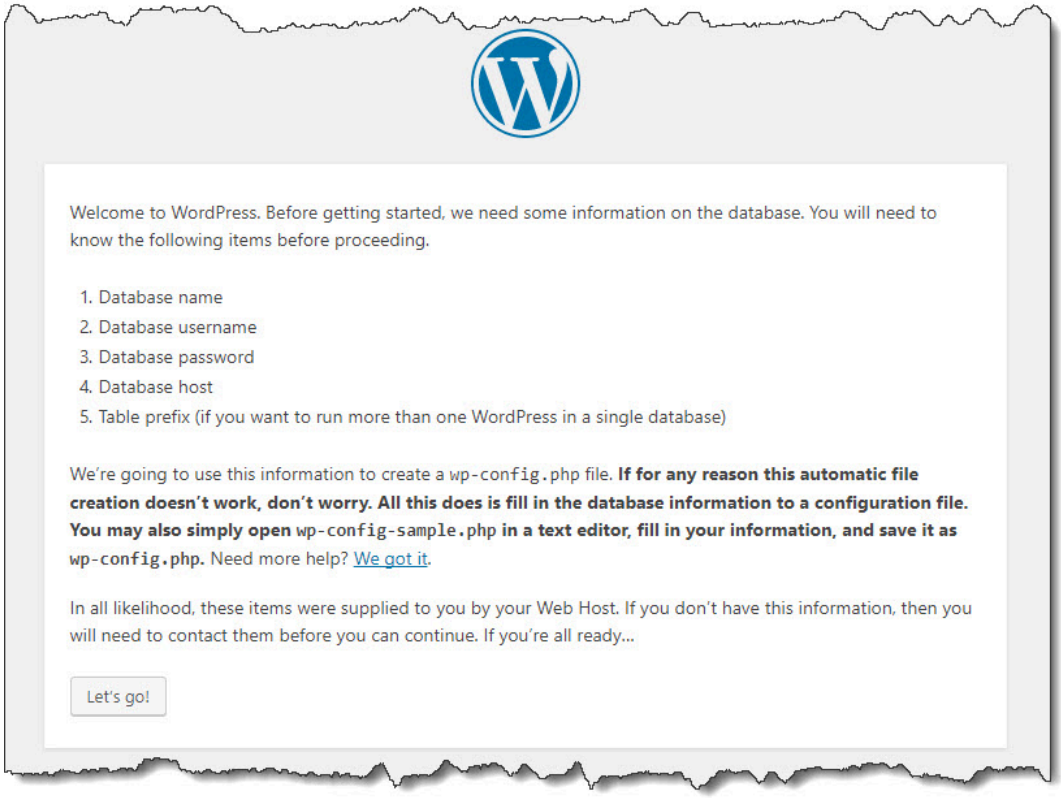
验证部署

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，然后在说明选项卡上，查找公有 DNS。）

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 实例没有添加任何 HTTP 入站规则到其安全组，则不显示 WordPress 欢迎页面。如果您看到说明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 添加允许 HTTP 流量到您的 Amazon Linux 或 RHEL Amazon EC2 实例的入站规则 (p. 53)。

第 5 步：更新和重新部署您的 WordPress 应用程序

既然您已成功部署应用程序修订，那么就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 网站 (p. 63)
- 修改站点 (p. 64)
- 重新部署站点 (p. 64)

设置 WordPress 网站

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扩展名。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**
   - 密码：将 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

既然您已经修改了站点的代码，则可以开始使用 Amazon S3 和代 CodeDeploy y 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捆绑到单个存档文件中，然后推送归档文件 (p. 59)。(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，**codedeploydemobucket**)。

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

主题

- 重新部署站点 (CLI) (p. 65)
- 重新部署站点（控制台） (p. 65)

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 **codedeploydemobucket** 的存储桶中名为 **WordPressApp.zip** 的修订：

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

您可以检查部署的状态，如 对部署进行监控和排除故障 (p. 62) 中所述。

在 CodeDeploy 重新部署站点之后，在 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. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 _ 应用程序。
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，然后将修订版本的 Amazon S3 链接复制到修订位置。要查找链接值，请执行以下操作：

      i. 在单独的浏览器选项卡中：

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

         浏览到并打开代码部署演示存储桶，然后选择您的修订**WordPressApp.zip**。
      ii. 如果属性窗格中不可见，则选择属性按钮。
      iii. 在属性窗格中，复制Link字段中的修订位置框中的 CodeDeploy 控制台中。
   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. Expand部署组覆盖和来自的部署配置中，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 对部署进行监控和排除故障 (p. 62) 中所述。

      在 CodeDeploy 重新部署站点之后，在 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！

第 6 步：清理您的 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)。
- 任何 Amazon S3 存储桶。
- 这些区域有：`WordPress_App`应 CodeDeploy。
- 这些区域有：Amazon Systems ManagerCodeDeploy 代理的状态管理器关联。

您可以将Amazon CLI，Amazon CloudFormation、Amazon S3、Amazon EC2 和 CodeDeploy 控制台或Amazon用于执行清理的 API。

主题

- 清除资源 (CLI) (p. 66)
- 清除资源（控制台）(p. 67)
- 接下来做什么？ (p. 68)

清除资源 (CLI)

1. 如果您为此教程使用了 Amazon CloudFormation 模板，请对名为 **`CodeDeployDemoStack`** 的堆栈调用 delete-stack 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，请调用rm命令，--recursive切换到名为**`codedeploydemobucket`**。这将删除存储桶以及该存储桶中的所有对象：

```
aws s3 rm s3://codedeploydemobucket --recursive
```

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
```

   您可以将`####`调用describe-association命令。

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

如果您没有使用Amazon CloudFormation堆栈，请调用terminate-instances命令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 Management Console，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。

Amazon EC2 止在Amazon CloudFormation堆栈：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 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：

1. 登录到 Amazon Web Services Management Console，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，**`codedeploydemobucket`**)。
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。

删除`WordPress_App`应用程序从 CodeDeploy：

1. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 _ 应用程序。
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

1. 打开Amazon Systems Manager控制台，请访问 https://console.aws.amazon.com/systems-manager。

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：部署一个 “你好，世界！” 应用程序与 CodeDeploy（Windows 服务器）

在本教程中，您会将一个网页部署到一个正在将互联网信息服务 (IIS) 作为其 Web 服务器 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？

- 要练习将部署到 Amazon Linux 或 Red Hat Enterprise Linux (RHEL) Amazon EC2 实例，请参阅教程：将 WordPress 部署到 Amazon EC2 实例（亚马逊 Linux 或红帽企业 Linux 和 Linux、macOS 或 Unix）(p. 51)。
- 要改而部署到本地实例，请参阅教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例(p. 82)。

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）的步骤。`c:\temp`。另外，如果您要连接到 Amazon EC2 实例，您需要一个客户端应用程序，该应用程序可以通过远程桌面协议 (RDP) 连接到正在运行 Windows Server 的 Amazon EC2 实例。（默认情况下，Windows 包含 RDP 连接客户端应用程序。）

在开始本教程前，您必须完成开始使用 CodeDeploy(p. 26)，包括配置 IAM 用户、安装或升级Amazon CLI，并创建 IAM 实例配置文件和服务角色。

主题

- 第 1 步：启动 Windows 服务器 Amazon EC2 实例 (p. 68)
- 第 2 步：将源内容配置为部署到 Windows Server Amazon EC2 实例的源内容 (p. 70)
- 第 3 步：上传您的 “你好，世界！” 应用程序添加到 Amazon S3 (p. 72)
- 第 4 步：部署您的 “世界” 应用程序 (p. 75)
- 第 5 步：更新和重新部署您的 “你好，世界！” application (p. 78)
- 第 6 步：清理你的 “你好，世界！” 应用程序和相关资源 (p. 80)

第 1 步：启动 Windows 服务器 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(p. 170)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，请确保指定标签密钥**Name**和标签值**CodeDeployDemo**。(如果您指定不同的标记键或标签值，则第 4 步：部署您的 “世界” 应用程序(p. 75)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 CodeDeploy 创建应用程序(p. 212)作为下一步骤。

Connect 到 Amazon EC2 实例

在您的 Amazon EC2 实例启动之后，请按照下面的说明执行连接操作。

> **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。想要了解有关信息，请参阅使用 RDP 连接到 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. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 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 Connect (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 key pair，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. 在您连接之后，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 流量到您的 Windows 服务器 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 Management Console，然后通过以下网址打开 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. 在存储库的说明选项卡，在安全组中，选择查看入站规则。

   您应在安全组中看到类似以下的规则列表：

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

4. UNDER安全组下，为您的 Amazon EC2 实例选择安全组。它可能被命名为**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，您应会看到具有以下值的规则：

   - 类型：HTTP
   - 协议：TCP

- 端口范围：80
- 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将它们添加到新的安全规则中。

第 2 步：将源内容配置为部署到 Windows Server Amazon EC2 实例的源内容

现在是时候配置应用程序的源内容以拥有可部署到 Amazon EC2 实例的内容了。在此教程中，您将一个网页部署到运行 Windows Server 的 Amazon EC2 实例，后者将互联网信息服务 (IIS) 作为其 Web 服务器运行。此网页将显示简单的 “Hello，World!” 消息。

主题

- 创建网页 (p. 70)
- 创建脚本以运行应用程序 (p. 71)
- 添加应用程序规范文件 (p. 71)

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>
```

创建脚本以运行应用程序

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YAMLCodeDeploy 使用的格式文件夹：

- 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。

必须将 AppSpec 文件命名为`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 文件复制`index.html`文件夹中的应用程序源代码根文件夹中的`c:\inetpub\wwwroot`文件夹中的 Amazon EC2 夹。在部署过程中，CodeDeploy 将运行`before-install.bat`目标 Amazon EC2 实例上的批处理脚本**`BeforeInstall`**部署生命周期事件。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 (p. 321)。

> **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 (p. 349)。

第 3 步：上传您的 “你好，世界！” 应用程序添加到 Amazon S3

现在，您将准备源内容并将其上传到 CodeDeploy 可从该位置部署源内容的位置。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GitHub 将 CodeDeploy 集成 (p. 47)。

主题

- 配置 Amazon S3 存储桶 (p. 72)
- 为存储桶准备应用程序的文件 (p. 74)
- 将应用程序的文件捆绑到单个存档文件中，然后推送归档文件 (p. 74)

配置 Amazon S3 存储桶

创建存储容器或存储桶在 Amazon S3 中 — 或者使用现有存储桶。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 来创建 Amazon S3 存储桶。在您创建存储桶之后，请确保为存储桶和您的 IAM 用户提供访问权限。

> **Note**
>
> 对于 Amazon S3 中的所有存储桶名称在 Amazon S3 中必须唯一Amazon账户。如果您无法使用 **`codedeploydemobucket`**，请尝试其他存储桶名称，例如 **`codedeploydemobucket`** 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 **`codedeploydemobucket`** 替换为您的存储桶名称。
> Amazon S3 存储桶必须在同一Amazon区域中启动目标 Amazon EC2 实例。例如，如果您在美国东部（弗吉尼亚北部）区域中创建存储桶，则您的目标 Amazon EC2 实例必须在美国东部（弗吉尼亚北部）区域启动。

主题

- 创建 Amazon S3 存储桶 (CLI) (p. 72)
- 创建 Amazon S3 存储桶（控制台） (p. 72)
- 向 Amazon S3 存储桶和您的 IAM 用户授予权限 (p. 73)

创建 Amazon S3 存储桶 (CLI)

调用mb命令创建名为的 Amazon S3 存储桶**`codedeploydemobucket`**：

```
aws s3 mb s3://codedeploydemobucket
```

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。

向 Amazon S3 存储桶和您的 IAM 用户授予权限

您必须具有上传到 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 Amazon S3 存储桶中任何名为的目录中的任何名为`codedeploydemobucket`：

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

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 存储桶策略指定此权限。例如，在以下 Amazon S3 存储桶策略中，使用通配符 (*) 允许任何带有包含 ARN 的 IAM 实例配置文件的 Amazon EC2 实例`arn:aws:iam::444455556666:role/CodeDeployDemo`从 Amazon S3 存储桶中任何名为`codedeploydemobucket`：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "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。

您的账户必须具有将修订上传到 Amazon S3 存储桶的权限。一种方式是通过 IAM 策略指定此权限。以下 IAM 策略允许您的 IAM 用户将修订上传到名为的 Amazon S3 存储桶中的任意位置`codedeploydemobucket`：

```
{
```

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

有关如何创建和附加 IAM 策略的信息，请参阅使用策略。

为存储桶准备应用程序的文件

确保在您的开发计算机上按如下方式整理网页、AppSpec 文件和脚本：

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

将应用程序的文件捆绑到单个存档文件中，然后推送归档文件

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命令注册名为的新应用程序**`HelloWorld_App`**使用 CodeDeploy:

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

3. 调用 CodeDeploypush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 HelloWorld_App --s3-location s3://
   codedeploydemobucket/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）捆绑到一个名为`HelloWorld_App.zip`中，将修订上传到**`codedeploydemobucket`**存储桶，并向 CodeDeploy 注册有关上传修订的信息。

第 4 步：部署您的 “世界” 应用程序

现在，您部署已上传到 Amazon S3 的示例 Hello, World! 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 (p. 75)
- 对部署进行监控和问题排查 (p. 77)
- 验证部署 (p. 77)

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) (p. 75)
- 部署应用程序修订（控制台）(p. 76)

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，服务角色将为提供 CodeDeploy 权限来访问您的 Amazon EC2 实例，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) (p. 28)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) (p. 29)。

2. 现在您已拥有 ARN，可以开始调用create-deployment-group命令创建名为的部署组**HelloWorld_DepGroup**，与名为的应用程序**HelloWorld_App**，使用名为的 Amazon EC2 实例标签**CodeDeployDemo**和部署配置**CodeDeployDefault.OneAtATime**，具有服务角色 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

   这些区域有：创建部署组命令支持创建触发器，这些触发器将导致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以在满足 Amazon CloudWatch 警报中的监控阈值时停止部署的选项。本教程不包括这些操作的命令。

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，请参阅是什么Amazon Systems Manager。

4. 现在调用 create-deployment 命令，使用名为 **codedeploydemobucket** 的存储桶中名为 **HelloWorld_App.zip** 的应用程序修订，创建一个与名为 **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=codedeploydemobucket,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，服务角色将为提供 CodeDeploy 权限来访问您的 Amazon EC2 实例，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） (p. 28)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） (p. 29)。
2. 既然您已拥有 ARN，那么就可以开始使用 CodeDeploy 控制台部署您的应用程序修订。

   登录到Amazon Web Services Management Console，然后在中打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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. In环境配置中，选择Amazon EC2 实例。
10. In代理配置Amazon Systems Manager，请保留默认值。
11. 在键中，输入 **Name**。
12. 在值中，输入 **CodeDeployDemo**。
13. In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. In修订类型中，选择我的应用程序存储在 Amazon S3，然后在修订位置中，输入您之前上传到 Amazon S3 的 Hello, World! 应用程序修订示例的位置。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，选择代码部署演示存储桶（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. 返回到 CodeDeploy 控制台，然后在修订位置中，粘贴Link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。

对部署进行监控和问题排查

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) (p. 77)
- 监视您的部署和故障排除（控制台） (p. 77)

监视您的部署并排除故障 (CLI)

1. 针对名为 **HelloWorld_App** 的应用程序和名为 **HelloWorld_DepGroup** 的部署组调用 list-deployments 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`。

   如果整体状态为`Failed`，则您可以调用诸如列表部署实例和获取部署实例排除的故障。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 (p. 373)。

监视您的部署和故障排除（控制台）

在存储库的部署页 CodeDeploy，您可以在状态column.

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 (p. 202) 的技巧。您还可以分析 Amazon EC2 实例上的部署日志文件。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 (p. 373)。

验证部署

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，然后在说明选项卡上，在公有 DNS。）

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

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

如果成功，您应该看到 Hello, World! 网页。

第 5 步：更新和重新部署您的 “你好，世界！” application

既然您已成功部署您的应用程序修订，那么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 (p. 78)
- 重新部署站点 (p. 78)

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

既然您已修改代码，那么就可以使用 Amazon S3 和 CodeDeploy 重新部署网页。

将更改打包并上传到 Amazon S3，如中所述将应用程序的文件捆绑到单个存档文件中，然后推送归档文件 (p. 74)。（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，**`codedeploydemobucket`**）。

使用Amazon CLI或 CodeDeploy 控制台中的站点。

主题

- 重新部署站点 (CLI) (p. 79)
- 重新部署站点（控制台） (p. 79)

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 **codedeploydemobucket** 的存储桶中名为 **HelloWorld_App.zip** 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=codedeploydemobucket,bundleType=zip,key=HelloWorld_App.zip
```

您可以检查新部署的状态，如对部署进行监控和问题排查 (p. 77)中所述。

在 CodeDeploy 重新部署了站点后，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. 登录到Amazon Web Services Management Console，然后在中打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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. In修订位置中，输入与您的修订相关的 Amazon S3 链接。

      要查找链接值，请执行以下操作：

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

         浏览到并打开代码部署演示存储桶，然后选择您的修订**HelloWorld_App.zip**，在 Amazon S3 控制台中。
      ii. 如果属性窗格中不可见，则在 Amazon S3 控制台中选择属性按钮。
      iii. 在属性窗格中，复制链接字段的值。
      iv. 返回 CodeDeploy 控制台，然后将链接粘贴到修订位置。
      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. Expand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，如对部署进行监控和问题排查 (p. 77)中所述。

      在 CodeDeploy 重新部署了站点后，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！

第 6 步：清理你的 “你好，世界！” 应用程序和相关资源

您现在已成功更新 “Hello, World!” 代码并重新部署站点。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，条件是您在Amazon CloudFormation)。
- 任何 Amazon S3 存储桶。
- 这些区域有：`HelloWorld_App`应用程序中的。
- 这些区域有：Amazon Systems ManagerCodeDeploy 代理的状态管理器关联。

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

主题

- 使用清除资源 (CLI) (p. 80)
- 清除资源（控制台） (p. 81)
- 接下来做什么？ (p. 82)

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 **`CodeDeployDemoStack`** 的堆栈调用 delete-stack 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，请调用rm命令使用--recursive切换到名为**`codedeploydemobucket`**。这将删除存储桶以及该存储桶中的所有对象。

```
aws s3 rm s3://codedeploydemobucket --recursive
```

3. 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
```

   您可以将`####`调用describe-association命令。

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

5. 如果您没有使用Amazon CloudFormation堆栈，请调用terminate-instances命令终止您手动创建的 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 Management Console并打开Amazon CloudFormation控制台位于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。

要终止在外部创建的 Amazon EC2 实例，请执行Amazon CloudFormation堆栈：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 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 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：

1. 登录到 Amazon Web Services Management Console，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，**codedeploydemobucket**)。
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。

删除`HelloWorld_App`应用程序从 CodeDeploy：

1. 登录到Amazon Web Services Management Console，然后在中打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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

1. 打开Amazon Systems Manager控制台，请访问 https://console.aws.amazon.com/systems-manager。
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？

如果您已抵达此处，则表示您已成功完成使用 CodeDeploy 执行的部署。恭喜您!

教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例

本教程通过引导您 CodeDeploy 示例应用程序修订部署到单个本地实例来帮助您体验即，不是 Amazon EC2 实例的物理设备运行 Windows 服务器、Ubuntu 服务器或红帽企业 Linux (RHEL)。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 (p. 179)。

不是您要找的内容？

- 要练习到运行 Amazon Linux 或 RHEL 的 Amazon EC2 实例的部署，请参阅教程：将 WordPress 部署到 Amazon EC2 实例（亚马逊 Linux 或红帽企业 Linux 和 Linux、macOS 或 Unix） (p. 51)。
- 要练习到运行 Windows Server 的 Amazon EC2 实例的部署，请参阅教程：部署一个 “你好，世界！” 应用程序与 CodeDeploy（Windows 服务器） (p. 68)。

主题

- Prerequisites (p. 82)
- 第 1 步：配置本地实例 (p. 82)
- 第 2 步：创建示例应用程序修订 (p. 82)
- 第 3 步：捆绑应用程序修订并上传到 Amazon S3 (p. 86)
- 第 4 步：部署您的应用程序修订 (p. 86)
- 第 5 步：验证部署 (p. 86)
- 第 6 步：清理资源 (p. 86)

Prerequisites

在开始本教程前，您必须完成开始使用 CodeDeploy (p. 26)，其中包括配置 IAM 用户、安装或升级Amazon CLI和创建服务角色。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 (p. 141) 中列出的操作系统之一。

第 1 步：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 (p. 179)中的说明操作，然后返回此页。

安 CodeDeploy

配置本地实例后，请按照安 CodeDeploy并返回此页。

第 2 步：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。

由于很难知道哪些软件和功能已经安装 — 或允许您组织的策略安装 — 在本地实例上，我们提供的示例应用程序修订只使用批处理脚本 (适用于 Windows Server) 或外壳脚本 (适用于 Ubuntu 服务器和 RHEL)将文本文件写入到本地实例上的位置。对于多个 CodeDeploy 部署生命周期事件（包括安装、AfterInstall、ApplicationStart, 和ValidateService。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 代理的用户帐户无权执行脚本。
- 用户账户无权在脚本中列出的位置创建或删除文件夹。
- 用户账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，则可能需要使用第 2 步：将源内容配置为部署到 Windows Server Amazon EC2 实例的源内容 (p. 70)中的教程：部署一个 “你好，世界！” 应用程序与 CodeDeploy（Windows 服务器）(p. 68)教程。
如果您配置了 RHEL 实例并希望部署其他示例，则可能需要在第 2 步：将源内容配置为部署到亚马逊 Linux 或红帽企业 Linux Amazon EC2 实例 (p. 53)中的教程：将 WordPress 部署到 Amazon EC2 实例（亚马逊 Linux 或红帽企业 Linux 和 Linux、macOS 或 Unix）(p. 51)教程。
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，我们假设您将使用`c:\temp`文件夹作为 Windows 服务器的根文件夹或`/tmp`文件夹作为 Ubuntu 服务器和 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：

   `appspec.yml`适用于 Windows Server 的：

   ```
   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
```

`appspec.yml`对于 Ubuntu 服务器和 RHEL：

```
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 的修订版 (p. 239)和CodeDeploy AppSpec 文件参考 (p. 321)。

`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`子文件夹中，使用文本编辑器创建四个名为`before-install.bat`、`after-install.bat`、`application-start.bat`, 和`validate-service.bat`对于 Windows Server，或者`before-install.sh`、`after-install.sh`、`application-start.sh`, 和`validate-service.sh`对于 Ubuntu 服务器和 RHEL：

   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，请确保四个 shell 脚本都具有执行权限：

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

第 3 步：捆绑应用程序修订并上传到 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 CodeDeploy 创建应用程序 (p. 212)和将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署）(p. 247)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GitHub 将 CodeDeploy 集成 (p. 47)。

第 4 步：部署您的应用程序修订

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 CodeDeploy 创建部署 (p. 254)中的说明操作，然后返回此页。

第 5 步：验证部署

要验证部署已经成功，请按照查看 CodeDeploy 部署详细信息 (p. 266)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（适用于 Windows 服务器）或`/tmp/CodeDeployExample`（适用于 Ubuntu 服务器和 RHEL）。

如果部署失败，请按照View Instance Details (p. 202)和 排查实例问题 (p. 372)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。

第 6 步：清理资源

为避免对您为此教程创建的资源持续收费，请删除 Amazon S3 存储桶（如果您不再使用）。您还可以清除关联的资源，例如 CodeDeploy 和本地实例中的应用程序和部署组记录。

您可以将Amazon CLI或 CodeDeploy 和 Amazon S3 控制台以及Amazon CLI清除资源。

清除资源 (CLI)

删除 Amazon S3 存储桶

- 调用RM命令以及`--recursive`切换到存储桶（例如`codedeploydemobucket`）。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
```

删除 CodeDeploy 中的应用程序和部署组记录

- 调用删除应用程序命令（例如，CodeDeploy-OnPrem-App)。将删除部署和部署组的记录。

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

取消注册本地实例并删除 IAM 用户

- 调用取消注册对本地实例和区域的命令：

```
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

- 从本地实例调用卸载命令：

```
aws deploy uninstall
```

现在您已完成清除此教程所用资源的全部步骤。

清除资源（控制台）

删除 Amazon S3 存储桶

1. 登录到 Amazon Web Services Management Console，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，`codedeploydemobucket`），但不要选择存储桶本身。
3. 选择 Actions，然后选择 Delete 。
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. 登录到Amazon Web Services Management Console，CodeDeployhttps://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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。

您无法使用Amazon CodeDeploy控制台注销本地实例或者卸载 CodeDeploy 代理。按照取消注册本地实例并删除 IAM 用户 (p. 87)中的说明进行操作。

教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组

在本教程中，您将使用 CodeDeploy 将应用程序修订部署到 Amazon EC2 Auto Scaling 组。Amazon EC2 Auto Scaling 使用预定义的条件启动 Amazon EC2 实例，然后在不再需要这些实例时终止这些实例。Amazon EC2 Auto Scaling 可确保 CodeDeploy 始终拥有正确数量的 Amazon EC2 实例来处理部署负载。有关 Amazon EC2 Auto Scaling 集 CodeDeploy 的信息，请参阅将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

主题

- Prerequisites (p. 88)
- 第 1 步：创建和配置 Amazon EC2 Auto Scaling 组 (p. 89)
- 第 2 步：将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 92)
- 第 3 步：检查您的结果 (p. 96)
- 第 4 步：增加 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例的数量 (p. 97)
- 第 5 步：再次检查您的结果 (p. 98)
- 第 6 步：清除 (p. 99)

Prerequisites

在本教程中遵循以下步骤操作：

- 完成中的所有步骤开始使用 CodeDeploy (p. 26)，包括设置和配置Amazon CLI并创建 IAM 实例配置文件 (**`CodeDeployDemo-EC2-Instance-Profile`**) 和一个服务角色 (**`CodeDeployDemo`**)。A服务角色是一种特殊类型的 IAM 角色，该角色授予某个服务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 。第 2 步：创建服务角色 (p. 26)、为 Auto Scaling 组创建启动模板, 和启动模板支持中的Amazon EC2 Auto Scaling 用户指南。
- 创建和使用与 Ubuntu 服务器实例和 CodeDeploy 兼容的修订。对于您的修订，可以执行以下操作之一：
  - 创建并使用教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)教程的第 2 步：创建示例应用程序修订 (p. 82)中的示例修订。
  - 自行创建修订，具体请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。
- 创建名为的安全组**`CodeDeployDemo-AS-SG`**使用以下入站规则：
  - 类型: HTTP
  - 源：随时随地

  这是查看应用程序和验证部署成功所必需的。有关如何创建组的信息，请参阅创建安全组中的Amazon EC2 用户指南。

第 1 步：创建和配置 Amazon EC2 Auto Scaling 组

在此步骤中，您将创建一个包含单个 Amazon Linux、RHEL 或 Windows 服务器 Amazon EC2 实例的 Amazon EC2 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 会将您的修订部署到它。

主题

- 创建和配置 Amazon EC2 Auto Scaling 组 (CLI) (p. 89)
- 创建和配置 Amazon EC2 Auto Scaling 组（控制台）(p. 91)

创建和配置 Amazon EC2 Auto Scaling 组 (CLI)

1. 调用create-launch-configuration命令创建 Amazon EC2 Auto Scaling 启动配置。

   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 key pair 的名称 (由占位符表示) 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. 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. 在存储库的快速启动选项卡上的选择一个 Amazon 系统映像页面上，注意 AMI 的 IDAmazon Linux AMI、Red Hat Enterprise Linux 7.1、Ubuntu 服务器 14.04 LTS，或者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 用于 CodeDeploy 和 Amazon EC2 Auto Scaling 的信息，请参阅将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 结合使用 (p. 42)。

   对于 Amazon EC2 实例 key pair，请使用 Amazon EC2 实例 key pair 的名称。

   调用 create-launch-configuration 命令。

   在本地 Linux、macOS 或 Unix 计算机上：

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name CodeDeployDemo-AS-Configuration \
  --image-id image-id \
  --key-name key-name \
  --iam-instance-profile CodeDeployDemo-EC2-Instance-Profile \
  --instance-type t1.micro
```

   在本地 Windows 计算机上：

```
aws autoscaling create-launch-configuration --launch-configuration-name CodeDeployDemo-
AS-Configuration --image-id image-id --key-name key-name --iam-instance-profile
 CodeDeployDemo-EC2-Instance-Profile --instance-type t1.micro
```

   这些命令创建 Amazon EC2 Auto Scaling 启动配置**CodeDeployDemo-AS-Configuration**（根据指定的映像 ID），同时应用指定的 IAM 实例配置文件和 Amazon EC2 实例 key pair。此启动配置基于 t1.micro Amazon EC2 实例类型。

2. 调用create-auto-scaling-group命令创建 Amazon EC2 Auto Scaling 组。您将需要的区域之一中的某个可用区的名称 (在区域和终端节点中的Amazon一般参考，由占位符表示###。

   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）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，请参阅按区域列出的资源工具包存储桶 (p. 356)。

   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-configuration-name CodeDeployDemo-AS-Configuration \
     --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-configuration-name CodeDeployDemo-AS-Configuration --min-size 1
    --max-size 1 --desired-capacity 1 --availability-zones availability-zone --tags
    Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
   ```

   这些命令将创建一个名为的 Amazon EC2 Auto Scaling 组**CodeDeployDemo-AS-Group**基于 Amazon EC2 Auto Scaling 启动配置**CodeDeployDemo-AS-Configuration**。此 Amazon EC2 Auto Scaling 组只有一个 Amazon EC2 实例，并且在指定可用区中创建。此 Amazon EC2 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 **CodeDeployDemo-AS-Group** 调用 describe-auto-scaling-groups 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。安装 CodeDeploy 代理程序，方法是调用create-association命令来自Amazon Systems Manager，这些标签是在 Amazon EC2 Auto Scaling 组创建时添加的。

   ```
   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 Amazon EC2 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 代理的更多信息，请参阅使用 CodeDeploy 代理。有关 Systems Manager 的更多信息，请参阅是什么Amazon Systems Manager。

创建和配置 Amazon EC2 Auto Scaling 组（控制台）

1. 打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在全局导航栏中，确保在中列出的某个区域。区域和终端节点中的Amazon一般参考处于选择状态。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. 在导航栏中的下实例中，选择启动模板。
4. 选择创建启动模板。
5. 在启动模板名称和描述对话框，用于启动模板名称输入，输入**`CodeDeployDemo-AS-Launch-Template`**。对其他字段保留默认值。
6. 在Amazon 系统映像 (AMI)对话框中，单击下拉列表AMI中，选择适用于本教程的 AMI：
   - 在存储库的快速启动选项卡上的AMI下拉列表中，选择以下选项之一：Amazon Linux AMI、Red Hat Enterprise Linux 7.1、Ubuntu 服务器 14.04 LTS，或者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 用于 CodeDeploy 和 Amazon EC2 Auto Scaling 的信息，请参阅将自定义 AMI 与 CodeDeploy 和 Amazon EC2 Auto Scaling 结合使用 (p. 42)。
7. In实例类型，选择下拉菜单，然后选择t1.micro。您可以使用搜索栏更快地查找。
8. 在密钥对（登录）对话框中，选择选择现有 key pair。在选择一个密钥对下拉列表中，选择您在前面步骤中创建或使用的 Amazon EC2 实例 key pair。
9. 在Network settings (网络设置)对话框中，选择Virtual Comvate Cloud (VPC)。

   在安全组下拉列表中，选择您在中创建的安全组教程的先决条件部分(**`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，选择您之前创建的 IAM 角色 (**`CodeDeployDemo-EC2-Instance-Profile`**) 在下IAM 实例配置文件。

    保留其他默认值。
11. 选择创建启动模板。
12. 在后续步骤对话框中，选择创建 Auto Scaling 组。
13. 在存储库的选择启动模板或配置页面，用于Auto Scaling 组名称，键入**`CodeDeployDemo-AS-Group`**。
14. 在启动模板对话框中，启动配置 (**`CodeDeployDemo-AS-Launch-Template`**），如果没有填写，请从下拉菜单中选择它。保留默认值，然后选择下一步。
15. 在存储库的配置设置页面上，在网络对话框中，选择默认 VPC 和默认子网。如果您无法选择默认值，则必须创建 VPC。有关更多信息，请参阅 。Amazon VPC 入门。
16. 选择下一步转至配置高级选项页.
17. 保留默认值，然后选择下一步。
18. 在存储库的配置组大小和扩展策略页面上，请保留默认组大小值为 1。选择 Next (下一步)。
19. 跳过配置通知的步骤，然后选择下一步。
20. 在存储库的添加标签页面上，添加要在稍后安装 CodeDeploy 代理时使用的标签。选择 Add tag。
    a. 在键中，输入 **`Name`**。
    b. 在值中，输入 **`CodeDeployDemo`**。

选择 Next (下一步)。

21. 查看 Auto Scaling 组信息审核页面，然后选择创建 Auto Scaling 组。
22. 在导航栏中，使用Auto Scaling 组选定，请选择**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。
23. 安装 CodeDeploy 代理程序，方法是按照安装 CodeDeploy 代理并使用`Name=CodeDeployDemo`实例标签。

第 2 步：将应用程序部署到 Amazon EC2 Auto Scaling 组

在此步骤中，您将修订部署到 Amazon EC2 Auto Scaling 组中的单个 Amazon EC2 实例。

主题

- 创建部署 (CLI) (p. 92)
- 创建部署（控制台）(p. 94)

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. 您应该已经按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明创建了一个服务角色。该服务角色将向授予 CodeDeploy 访问您的 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) (p. 29) 中的说明操作。
3. 现在您拥有服务角色 ARN，请调用create-deployment-group命令创建组**SimpleDemoDG**，与名为**SimpleDemoApp**，使用名为的 Amazon EC2 Auto Scaling 组**CodeDeployDemo-AS-Group**和名为的部署配置**CodeDeployDefault.OneAtATime**（具有指定服务角色 ARN）。

   Note

   这些区域有：创建部署组命令支持创建触发器，这些触发器将导致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以在满足 Amazon CloudWatch 警报中的监控阈值时停止部署的选项。本教程不包括这些操作的命令。

   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。

   对于亚马逊 Linux 和 RHEL Amazon EC2 实例，可从本地 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
```

   对于从本地 Linux、macOS 或 Unix 计算机调用的 Windows 服务器 Amazon EC2 实例

   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 服务器 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 服务器 Amazon EC2 实例的示例修订。要自己创建修订，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。

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 **SimpleDemoApp** 的应用程序和名为 **SimpleDemoDG** 的部署组调用 list-deployments 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. 您应该已经按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明创建了一个服务角色。该服务角色将向授予 CodeDeploy 访问您的实例以展开（读取）其标签的权限。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。要获取服务角色 ARN，请按照获取服务角色 ARN（控制台） (p. 29) 中的说明操作。
2. 现在您已拥有服务角色 ARN，可以使用 CodeDeploy 控制台部署您的应用程序修订。

   登录到Amazon Web Services Management Console，然后从打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。

3. 在导航窗格中，展开部署，然后选择应用程序。

4. 选择 Create application。
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 Create application。
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. In环境配置选择Amazon EC2 Auto Scaling 组，然后选择**CodeDeployDemo-AS-Group**。
14. In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. In修订类型中，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。

    适用于亚马逊 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，则您可能需要尝试对部署进行监控和排除故障 (p. 62)中的一些方法（使用应用程序名称 **SimpleDemoApp** 和部署组名称 **SimpleDemoDG**）。

第 3 步：检查您的结果

在此步骤中，您将检查是否已在 CodeDeploy 安装**SimpleDemoApp**对 Amazon EC2 Auto Scaling 组中的单个 Amazon EC2 实例进行修订。

主题

- 检查结果 (CLI) (p. 96)
- 检查结果（控制台） (p. 96)

检查结果 (CLI)

首先，您将需要 Amazon EC2 实例的公有 DNS。

使用Amazon CLI获取 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例的公有 DNS，方法是调用describe-instances命令。

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 **CodeDeployDemo-AS-Group** 调用 describe-auto-scaling-groups：

```
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 Amazon EC2 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，您将将 Amazon EC2 实例添加到 Amazon EC2 Auto Scaling 组。Amazon EC2 Auto Scaling 添加 Amazon EC2 实例之后，CodeDeploy 会将您的修订部署到新的实例。

检查结果（控制台）

首先，您将需要 Amazon EC2 实例的公有 DNS。

打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

在 Amazon EC2 导航窗格中的下Auto Scaling中，选择Auto Scaling 组，然后选择**CodeDeployDemo-AS-Group**条目。

在存储库的实例选项卡上，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 Amazon EC2 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，您将 Amazon EC2 实例添加到 Amazon EC2 Auto Scaling 组。在 Amazon EC2 Auto Scaling 添加 Amazon EC2 实例后，CodeDeploy 将将您的修订部署到新的 Amazon EC2 实例。

第 4 步：增加 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例的数量

在此步骤中，您将指示 Amazon EC2 Auto Scaling 组创建额外的 Amazon EC2 实例。Amazon EC2 Auto Scaling 创建实例之后，CodeDeploy 会将您的修订部署到它。

主题

- 扩展 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例数 (CLI) (p. 97)
- 扩展部署组中的 Amazon EC2 实例数（控制台）(p. 97)

扩展 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例数 (CLI)

1. 调用update-auto-scaling-group命令增加 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例，该实例名为**CodeDeployDemo-AS-Group**从一个到二个。

   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 Amazon EC2 Auto Scaling 组现在有两个 Amazon EC2 实例。针对 **CodeDeployDemo-AS-Group** 调用 describe-auto-scaling-groups 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 导航栏中的Auto Scaling中，选择Auto Scaling 组，然后选择**CodeDeployDemo-AS-Group**。

2. 选择 Actions，然后选择 Edit。
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 实例应显示在列表中。（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。

第 5 步：再次检查您的结果

在此步骤中，您将检查 CodeDeploy 是否已在 Amazon EC2 Auto Scaling 组中的新实例上安装简单演示应用程序修订。

主题

- 检查自动部署结果 (CLI) (p. 98)
- 检查自动部署结果（控制台）(p. 99)

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 **SimpleDemoApp** 的应用程序和名为 **SimpleDemoDG** 的部署组调用 list-deployments 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，您还应该看到`autoScaling`在命令输出中。（`autoScaling`意味着 Amazon EC2 Auto Scaling 创建部署。）

请在部署状态显示 `Succeeded` 之后继续。

2. 在您调用describe-instances命令，您将需要新的 Amazon EC2 实例的 ID。要获取此 ID，请针对 **CodeDeployDemo-AS-Group** 再次调用 describe-auto-scaling-groups 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
```

在输出中，describe-instances命令中，记下新的 Amazon EC2 实例的公有 DNS。

3. 使用 Web 浏览器，显示`SimpleDemoApp`修订，Amazon EC2 如下所示的 URL：

```
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 Amazon EC2 Auto Scaling 组中扩展的 Amazon EC2 实例！

检查自动部署结果（控制台）

1. 登录到Amazon Web Services Management Console，然后从打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 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 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 导航窗格中的下Auto Scaling中，选择Auto Scaling 组，然后选择**CodeDeployDemo-AS-Group**条目。
7. 在存储库的实例选项卡上，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 Amazon EC2 Auto Scaling 组中扩展的 Amazon EC2 实例！

第 6 步：清除

在此步骤中，您将删除 Amazon EC2 Auto Scaling 组，以避免对您在本教程中使用的资源持续产生费用。（可选）您可以删除 Amazon EC2 Auto Scaling 配置和 CodeDeploy 部署组件记录。

主题

- 清除资源 (CLI) (p. 99)
- 清除资源（控制台）(p. 100)

清除资源 (CLI)

1. 通过调用 Amazon EC2 Auto Scaling 组delete-auto-scaling-group命令针对**CodeDeployDemo-AS-Group**。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）通过调用 Amazon EC2 Auto Scaling 启动配置，删除方法是调用delete-launch-configuration命令针对名为**CodeDeployDemo-AS-Configuration**：

   ```
   aws autoscaling delete-launch-configuration --launch-configuration-name CodeDeployDemo-AS-Configuration
   ```

3. （可选）从 CodeDeploy 中删除应用程序，方法是调用delete-application命令针对名为**SimpleDemoApp**。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
```

您可以获取####调用describe-association命令。

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

清除资源（控制台）

要删除 Amazon EC2 Auto Scaling 组（该组同时终止 Amazon EC2 实例）：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在 Amazon EC2 导航窗格中的下Auto Scaling中，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）要删除启动配置，请执行以下操作：

1. 在导航栏中的下Auto Scaling中，选择启动配置，然后选择**CodeDeployDemo-AS-Configuration**。
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。从打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。
2. 登录到Amazon Web Services Management Console，然后从打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。

   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，选择简单演示应用程序。
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。

删除 Systems Manager 状态管理器关联：

1. 打开Amazon Systems Manager控制台，请访问 https://console.aws.amazon.com/systems-manager。
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 RHub、一个红帽企业 Linux (RHEL) 实例或一个 Windows 服务器实例。有关 GitHub 与 CodeDeploy 的信息，请参阅GitHub 将 CodeDeploy 集成 (p. 47).

> Note
>
> 您还可使用 CodeDeploy 将应用程序修订从 GitHub 部署到 Ubuntu 服务器实例。您可以使用第 2 步：创建示例应用程序修订 (p. 82)在教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)，也可创建与 Ubuntu 服务器实例和 CodeDeploy 兼容的修订。要创建您自己的修订，请参阅规划 CodeDeploy 的修订版 (p. 239)和将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)。

主题

- Prerequisites (p. 101)
- 第 1 步：设置 GitHub 账户 (p. 101)
- 第 2 步：创建 GitHub 存储库 (p. 101)
- 第 3 步：将示例应用程序上传到 GitHub 存储库 (p. 103)
- 第 4 步：置备实例 (p. 105)
- 第 5 步：创建应用程序和部署组 (p. 105)
- 第 6 步：将应用程序部署到实例 (p. 106)
- 步骤 7：监控和验证部署 (p. 109)
- 步骤 8：清除 (p. 109)

Prerequisites

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.
- 完成开始使用 CodeDeploy (p. 26)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。

第 1 步：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，请向前跳至第 2 步：创建 GitHub 存储库 (p. 101)。

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。

第 2 步：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`**，然后向前跳至第 3 步：将示例应用程序上传到 GitHub 存储库 (p. 103)。

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 Create repository。
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）。想要了解有关信息，请参阅提供您的 2FA 身份验证码.

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。

第 3 步：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：
>
> - 遵循规划 CodeDeploy 的修订版 (p. 239)和将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)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，请向前跳至第 5 步：创建应用程序和部署组 (p. 105)。
> 如果您要部署到 Ubuntu 服务器实例，则需将与 Ubuntu 服务器实例和 CodeDeploy 兼容的修订上传到您的 GitHub 存储库。有关更多信息，请参阅 规划 CodeDeploy 的修订版 (p. 239) 和 将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)。

主题

- 从本地 Linux、macOS 或 Unix 计算机推送示例修订 (p. 103)
- 从本地 Windows 计算机推送示例修订 (p. 104)

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 服务器实例，请将`SampleApp_Windows.zip`对于 来说为`SampleApp_Linux.zip`在命令中。

```
(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 #####`是以下值之一：

- `aws s3 cp s3://aws-codedeploy-cn-north-1/samples/latest/SampleApp_Linux.zip . --region cn-north-1`为中国（北京）区域
- `aws s3 cp s3://aws-codedeploy-cn-northwest-1/samples/latest/SampleApp_Linux.zip . --region cn-northwest-1`中国（宁夏）区域

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，请将`SampleApp_Linux.zip`对于 来说为`SampleApp_Windows.zip`在命令中。

```
(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 #####`是以下值之一：

- `aws s3 cp s3://aws-codedeploy-cn-north-1/samples/latest/SampleApp_Windows.zip . --region cn-north-1`为中国（北京）区域
- `aws s3 cp s3://aws-codedeploy-cn-northwest-1/samples/latest/SampleApp_Windows.zip . --region cn-northwest-1`中国（宁夏）区域

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
```

第 4 步：置备实例

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 (p. 141)。（如果您已有配置为在 CodeDeploy 部署中使用的实例，请跳至下一步。）

预配实例

1. 按照中的说明进行操作启动 Amazon EC2 实例（控制台）(p. 171)来置备实例。
2. 启动实例时，请记住在添加标签页. 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）(p. 171).

验证 CodeDeploy 代理是否正在实例上运行

- 按照中的说明进行操作验证代 CodeDeploy 运行 (p. 149)以验证代理是否正在运行。

在成功预置实例并确认 CodeDeploy 代理正在运行之后，转至下一步。

第 5 步：创建应用程序和部署组

在本步骤中，您将使用 CodeDeploy 控制台或Amazon CLI来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. 登录到Amazon Web Services Management ConsoleCodeDeployhttps://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
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 Create application。
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. In服务角色中，选择 CodeDeploy 服务角色的名称。为 CodeDeploy 创建服务角色 (p. 26).
10. 在部署类型中，选择就地。
11. In环境配置，根据您使用的实例的类型，选择Amazon EC2 实例或者本地实例. 对于键和值，根据第 4 步：置备实例 (p. 105)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命令在 CodeDeploy 中创建一个名为`CodeDeployGitHubDemo-App`：

```
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，请*`EC2 ###`*是 Amazon EC2 实例标签键，该实例应用于 Amazon EC2 实例第 4 步：置备实例 (p. 105).
   - 如果您要部署到 Amazon EC2 实例，请*`EC2 ###`*是 Amazon EC2 实例的 Amazon 实例标签值，该值在第 4 步：置备实例 (p. 105).
   - 如果您要部署到本地实例，则 *`on-premises-tag-key`* 是作为第 4 步：置备实例 (p. 105)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`* 是作为第 4 步：置备实例 (p. 105)的一部分应用于本地实例的本地实例标签值。
   - *`#### arn`*是您在中创建的服务角色的服务角色 ARN为 CodeDeploy 创建服务角色 (p. 26).（按照获取服务角色 ARN (CLI) (p. 29)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**
>
> 这些区域有：创建部署组命令支持创建触发器，这些触发器将导致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以在满足 Amazon CloudWatch 警报中的监控阈值时停止部署的选项。本教程不包括对这些操作的命令。

第 6 步：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 与名为的应用程序的 GitHub 进行交互，`CodeDeployGitHubDemo-App`. 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，位于存储库名称中，输入您用于登录的 GitHub 用户名，然后输入正斜杠（/），然后是您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. In您的存储库中，选择代码部署 GitHub 演示.
   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)

在您可以调用任何Amazon CLI命令与 GitHub 交互（例如create-deployment命令（您接下来将调用的命令），您必须为 CodeDeploy 授权以使用您的 GitHub 用户账户与 GitHub 进行交互，然后才能调用的命令）。`CodeDeployGitHubDemo-App`应用程序. 当前，您必须使用 CodeDeploy 控制台执行此操作。

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 在导航窗格中，展开部署，然后选择应用程序.
3. 选择CodeDeploy.
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，选择代码部署管理演示-深层.

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 与名为的应用程序的 GitHub 进行交互，`CodeDeployGitHubDemo-App`. 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创建部署页面上，选择Cancel.
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. In您的存储库中，选择代码部署 GitHub 演示.
      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。

    如果您正在本地 Linux、macOS 或 Unix 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
```

    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

在本步骤中，您将使用 CodeDeploy 控制台或Amazon CLI以验证部署是否成功。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，查找带有应用程序值CodeDeploy和部署组值代码部署管理演示-深层. 如果Succeeded或者已失败文件不会显示在状态列中，选择Refresh按钮。
3. 如果已失败文件出现在状态列中的说明，请参阅以下说明：查看实例详细信息（控制台）(p. 202)对部署进行故障排除。
4. 如果Succeeded文件出现在状态列中，您现在可通过 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`http://public-dns`对实例（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，可向前跳至步骤 8：清除 (p. 109)。

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）(p. 202)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`http://public-dns`对于 Amazon EC2 实例（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！您已成功使用 Amazon CodeDeploy 从您的 GitHub 存储库部署。

步骤 8：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模板以创建 Amazon EC2 实例）

1. 登录到Amazon Web Services Management Console，然后打开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。
4. 当系统提示时，选择 Delete stack (删除堆栈)。Amazon EC2 实例以及关联的 IAM 实例配置文件和服务角色将被删除。

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调用取消注册命令对本地实例******和关联区域****：

```
aws deploy deregister --instance-name your-instance-name --no-delete-iam-user --
region your-region
```

2. 从本地实例调用卸载命令：

```
aws deploy uninstall
```

手动终止 Amazon EC2 实例（如果您已手动启动 Amazon EC2 实例）

1. 登录到Amazon Web Services Management Console，然后通过以下网址打开 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. 登录到Amazon Web Services Management ConsoleCodeDeployhttps://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 在导航窗格中，展开部署，然后选择应用程序.
3. 选择CodeDeploy.
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。

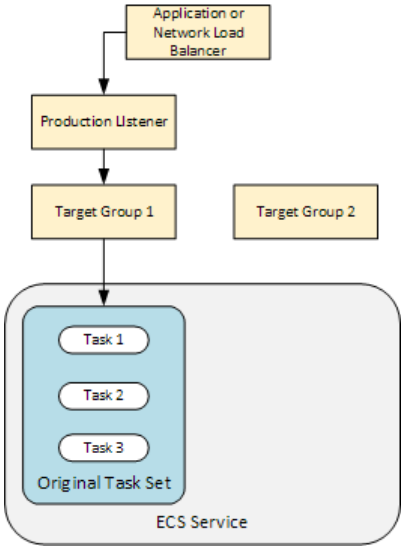
删除 GitHub 存储库

请参阅删除存储库在GitHub 帮助.

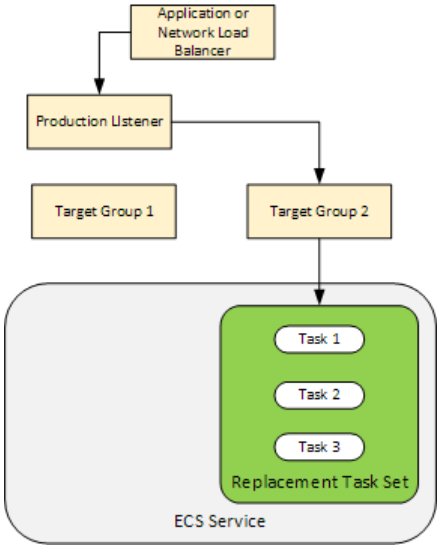
教程：部署 Amazon ECS 服务

在本教程中，您了解如何部署 Amazon ECS 应用程序。从您已创建的 Amazon ECS 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，使用 CodeDeploy 部署更新。在部署过程中，CodeDeploy 会将更新安装到新的替换任务集中。然后，将原始任务集中 Amazon ECS 服务原始版本的生产流量，转移到替换任务集中更新的版本。

在部署过程中，CodeDeploy 使用配置了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。本教程使用的是 Application Load Balancer。您还可以使用 Network Load Balancer。



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，请参阅亚马逊云服务器部署过程中会发生什么 (p. 16)。



有关如何使用Amazon CLI部署 Amazon ECS 服务，请参阅教程：使用蓝/绿部署创建服务. 有关如何使用 CodePipeline ploy 通过检测更改并自动部署到 Amazon ECSDeploy 服务的信息，请参阅教程：使用 Amazon ECR 源和 ECR 到 CodeDeploy 部署创建管道.

完成本教程后，您可以使用创建的 CodeDeploy 应用程序和部署组，在教程：通过验证测试部署 Amazon ECS 服务 (p. 117).

主题

- Prerequisites (p. 112)
- 第 1 步：更新您的 Amazon ECS 应用程序 (p. 112)
- 第 2 步：创建 AppSpec 文件 (p. 113)
- 第 3 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务 (p. 114)
- 第 4 步：清除 (p. 116)

Prerequisites

要完成本教程，您首先必须：

- 完成 开始使用 CodeDeploy (p. 26) 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 Application Load Balancer。有关使用控制台创建负载均衡器的信息，请参阅 为 CodeDeploy Amazon 云服务器部署设置负载均衡器、目标组和侦听器 (p. 230)。有关使用创建Amazon CLI，请参阅第 1 步：创建 Application Load Balancer中的Amazon Elastic Container Service 用户指南. 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。有关详细信息，请参阅教程：使用蓝/绿部署创建服务中的Amazon Elastic Container Service 用户指南. 记录以下内容以用于本教程：
  - Amazon ECS 集群的名称。
  - Amazon ECS 服务所使用的任务定义的 ARN。
  - Amazon ECS 服务所用的容器的名称。
- 为您的 AppSpec 文件创建 Amazon S3 存储桶。

第 1 步：更新您的 Amazon ECS 应用程序

在本部分中，您将使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。In 第 3 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务 (p. 114)，您将部署 Amazon ECS 应用程序的更新版本。

更新任务定义

1. 从打开 Amazon ECS 控制台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复选框。
4. 选择 Create new revision (创建新修订)。
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 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。

第 2 步：创建 AppSpec 文件

在本部分中，您将创建 AppSpec 文件并将其上传到您在Prerequisites (p. 112)部分。Amazon ECS 部署的 AppSpec 文件指定任务定义、容器名称和容器端口。有关更多信息，请参阅 Amazon ECS 部署 AppSpec 文件示例 (p. 346) 和 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329)。

创建 AppSpec 文件

1. 如果要使用 YAML 创建 AppSpec 文件，请创建名为`appspec.yml`. 如果要使用 JSON 创建 AppSpec 文件，请创建名为`appspec.json`.
2. 根据您是使用 YAML 还是 JSON 创建的 AppSpec 文件，选择相应的选项卡，并将其内容复制到您刚刚创建的 AppSpec 文件中。对于 `TaskDefinition` 属性，请使用您在 第 1 步：更新您的 Amazon ECS 应用程序 (p. 112) 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。您可以在 AppSpec 文件中设置可选属性，以覆盖替换任务集的这些值。有关更多信息，请参阅 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329) 和 Amazon ECS 部署 AppSpec 文件示例 (p. 346)。

3. 将 AppSpec 文件上传到作为本教程先决条件而创建的 S3 存储桶。

第 3 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务

在本部分中，您将创建 CodeDeploy 应用程序和部署组，以部署更新的 Amazon ECS 应用程序。在部署过程中，CodeDeploy 将 Amazon ECS 应用程序的生产流量转移到新的替换任务集的新版本。要完成此步骤，您需要以下各项：

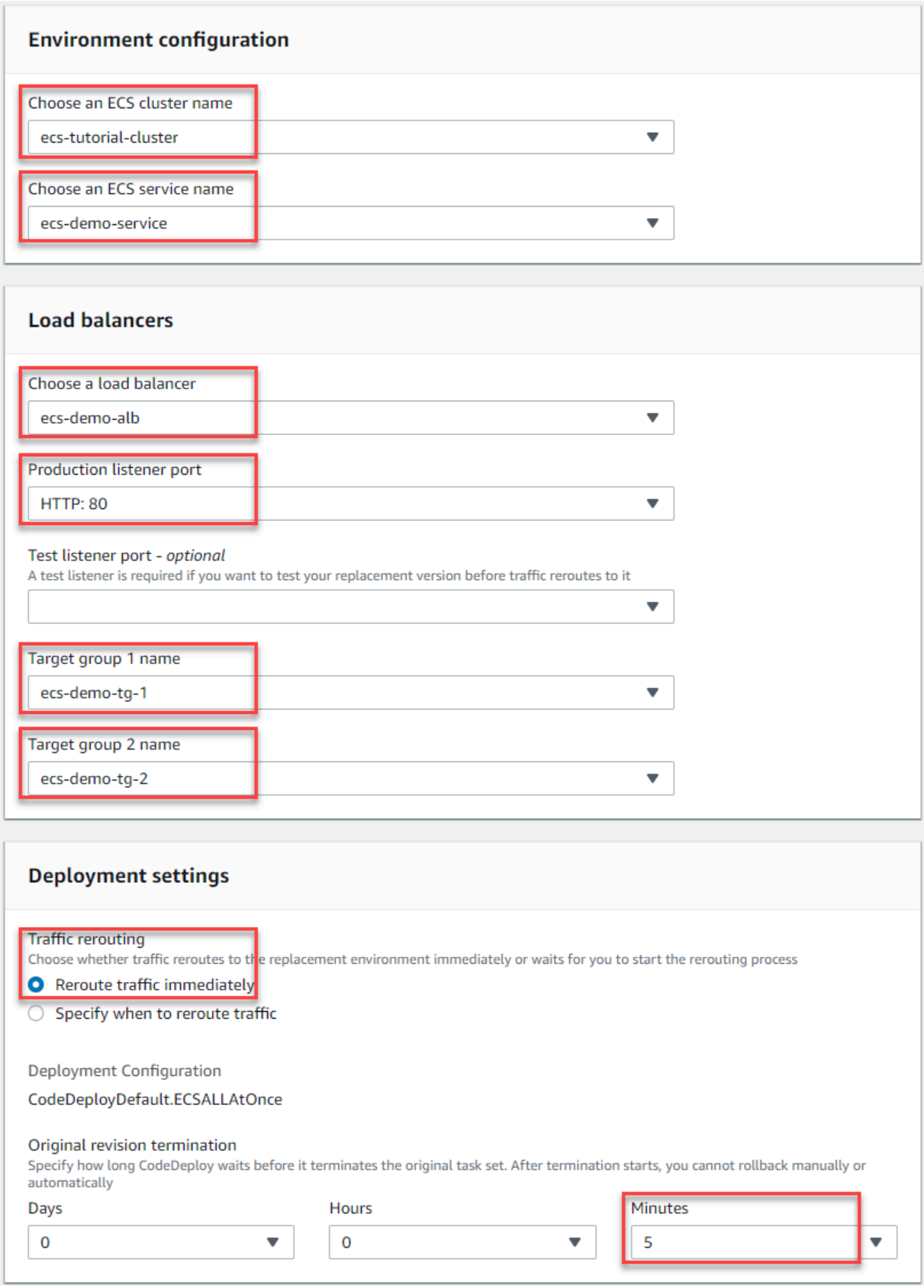
- 您的 Amazon ECS 集群名称。
- 您的 Amazon ECS 服务名称。
- 您的 Application Load Balancer 名称。
- 生产侦听器端口。
- 目标组名称。
- 您创建的 S3 存储桶的名称。

创建 CodeDeploy 应用程序

1. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy/.
2. 选择 Create application。
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 Create application。

创建 CodeDeploy 部署组

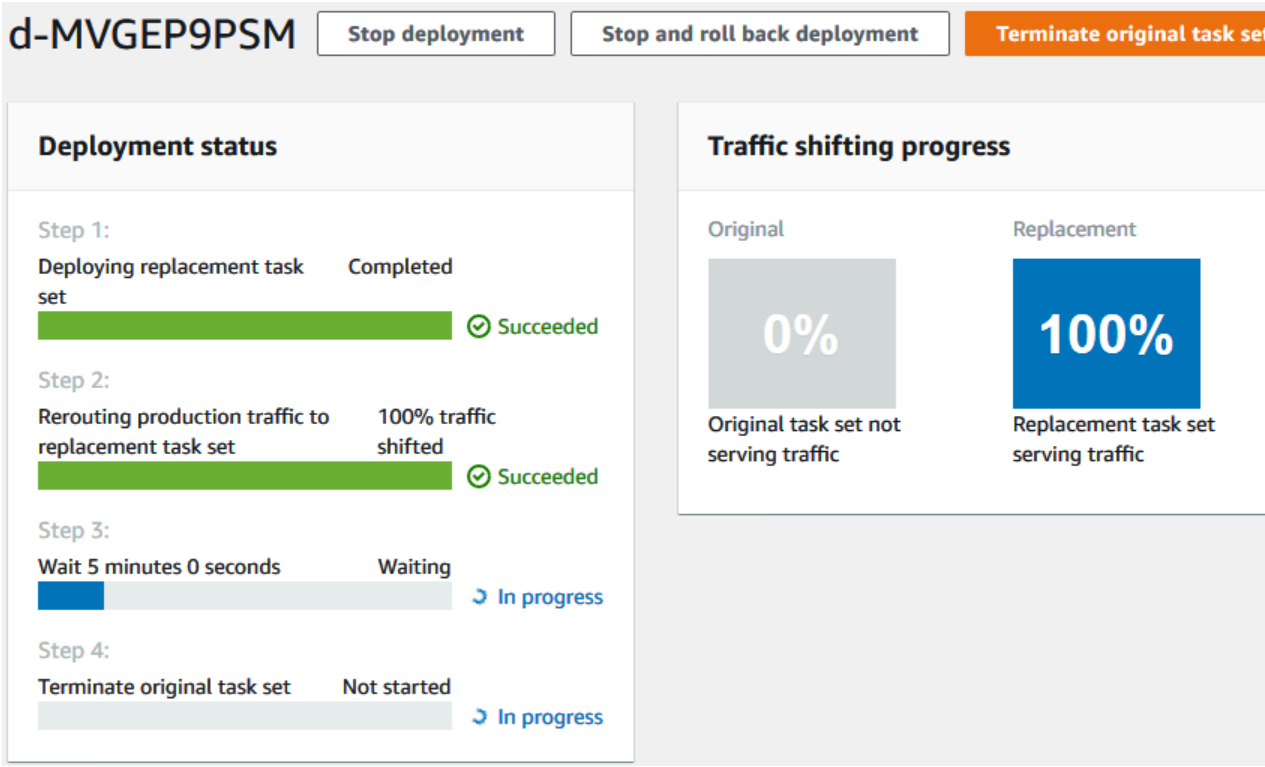
1. 在应用程序页面的 Deployment groups (部署组) 选项卡上，选择 Create deployment group (创建部署组)。
2. 在 Deployment group name (部署组名称) 中，输入 **ecs-demo-dg**。
3. In服务角色中，选择向 CodeDeploy 授予 Amazon ECS 访问权限的服务角色。有关更多信息，请参阅适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297)。
4. In环境配置，选择 Amazon ECS 集群名称和服务名称。
5. 从负载均衡器中，选择将流量路由至您的 Amazon ECS 服务的负载均衡器的名称。
6. 从生产侦听器端口，选择将生产流量传输至您的 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
0
Hours
0
Minutes
5

10. 选择 Create deployment group (创建部署组)。

部署 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 Deployment group (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。



第 4 步：清除

下一个教程教程：通过验证测试部署 Amazon ECS 服务 (p. 117)，建立在本教程的基础上，使用的是 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帐户不会对您创建的 CodeDeploy 资源产生费用。

这些步骤中的资源名称是本教程中建议的名称（例如，**ecs-demo-codedeploy-app**作为 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. 使用删除部署组命令删除 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. 使用删除应用程序命令删除 CodeDeploy 应用程序。

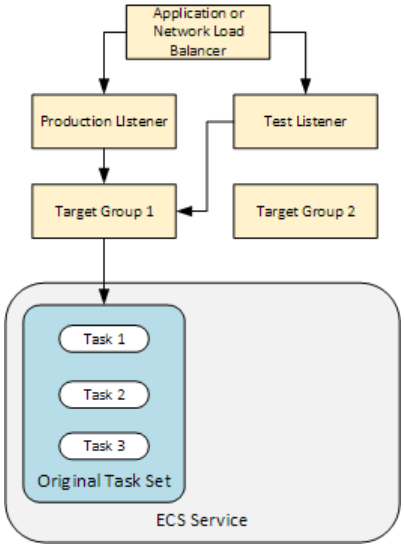
```
aws deploy delete-application --application-name ecs-demo-codedeploy-app --region aws-
region-id
```

教程：通过验证测试部署 Amazon ECS 服务

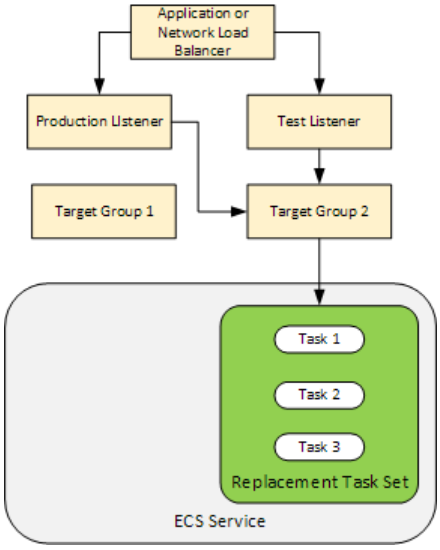
在本教程中，您将学习如何使用 Lambda 函数验证已更新的 Amazon ECS 应用程序的部分部署。本教程使用在中用到的 CodeDeploy 应用程序、CodeDeploy 组以及 Amazon 云服务器应用程序。教程：部署 Amazon ECS 服务 (p. 111)。请先完成上述教程，然后再开始本教程。

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。

在具有验证测试的 Amazon ECS 部署期间，CodeDeploy 使用配置了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 Application Load Balancer。您也可以使用 Network Load Balancer。



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Amazon 云服务器部署的生命周期事件挂钩列表 (p. 335)。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，请参阅亚马逊云服务器部署过程中会发生什么 (p. 16)。



Note

完成本教程可能会对您的 Amazon 账户收费。其中包括可能对 CodeDeploy、Amazon Lambda 和 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

- Prerequisites (p. 118)
- 第 1 步：创建测试侦听器 (p. 119)
- 第 2 步：更新您的亚马逊 ECS 应用程序 (p. 119)
- 第 3 步：创建生命周期挂钩 Lambda 函数 (p. 119)
- 第 4 步：更新 AppSpec 文件 (p. 121)
- 第 5 步：使用 CodeDeploy 控制台部署 Amazon ECS 服务 (p. 122)
- 第 6 步：在 CloudWatch Logs 中查看您的 Lambda 挂钩函数输出 (p. 123)
- 步骤 7：清除 (p. 124)

Prerequisites

要成功完成本教程，您首先必须：

- 满足 Prerequisites (p. 112)（针对教程：部署 Amazon ECS 服务 (p. 111)）中的先决条件。
- 完成 教程：部署 Amazon ECS 服务 (p. 111) 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。
  - 您创建用于 CodeDeploy 的 AppSpec 文件。您可以在本教程中编辑此文件。

第 1 步：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 命令创建第二个侦听器，其默认规则为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
```

第 2 步：更新您的亚马逊 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. 从打开 Amazon ECS 控制台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。

第 3 步：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。Lambda 函数在安装更新的 Amazon ECS 应用程序之前运行验证测试。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中的一个或多个实施 Lambda 测试函数（`BeforeInstall`、`AfterInstall`、`BeforeAllowTraffic`, 和`AfterAllowTraffic`)。有关更多信息，请参阅Amazon 云服务器部署的生命周期事件挂钩列表 (p. 335)。

必须具有 IAM 角色才能创建 Lambda 函数。该角色授予 Lambda 函数写入 CloudWatch Logs 并设置 CodeDeploy 生命周期挂钩状态的权限。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 从导航窗格中选择 Roles (角色)，然后选择 Create role (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。
   - Permissions (权限)：AWSLambdaBasicExecutionRole。这将授予 Lambda 函数写入 CloudWatch Logs 的权限。

- 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。这将授予 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 。添加 IAM 身份权限中的Amazon Identity and Access Management用户指南和PutLifecycleEventHookExecutionStatus中的CodeDeploy API 参考。

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 部署程序包。

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

3. 使用`create-function`命 Lambda 为`AfterAllowTestTraffic`挂钩。

```
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 Lambda 函数 ARN`create-function`响应. 您在下一个步骤中更新 CodeDeploy 部署部署的 AppSpec 文件时，可以使用此 ARN。

第 4 步：更新 AppSpec 文件

在本部分中，您将 AppSpec`Hooks`部分。在`Hooks`部分中，您可以为`AfterAllowTestTraffic`生命周期挂钩。

更新您的 AppSpec 文件

1. 打 AppSpec 您在 第 2 步：创建 AppSpec 文件 (p. 113)的教程：部署 Amazon ECS 服务 (p. 111)。
2. 采用您在 第 2 步：更新您的亚马逊 ECS 应用程序 (p. 119) 中记下的任务定义 ARN 更新 `TaskDefinition` 属性。
3. 复制并粘贴`Hooks`部分添加到您的 AppSpec 文件文件中。更新 ARN 之后`AfterAllowTestTraffic`的 Lambda 函数的 ARN，您在第 3 步：创建生命周期挂钩 Lambda 函数 (p. 119)。

   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 AppSpec 文件并上传至其 S3 存储桶。

第 5 步：使用 CodeDeploy 控制台部署 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 第 1 步：创建测试侦听器 (p. 119) 中创建的侦听器。在部署过程中，CodeDeploy 会在`AfterAllowTestTraffic`部署生命周期挂钩，使用通过测试侦听器提供给替换任务集的测试流量。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`。

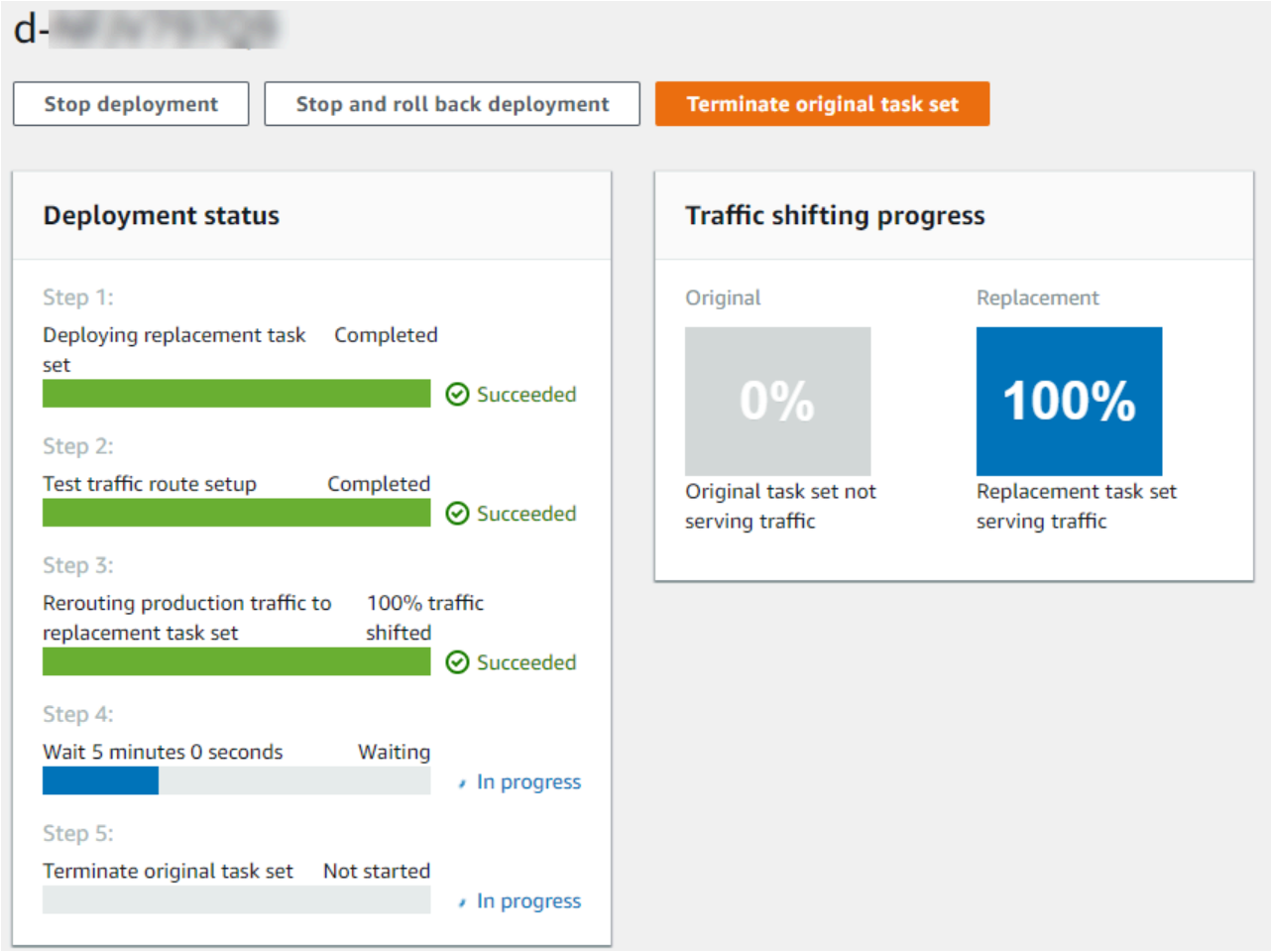
向部署组添加测试侦听器

1. 登录到Amazon Web Services Management ConsoleCodeDeployhttps://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：部署 Amazon ECS 服务 (p. 111) 中创建的应用程序。如果使用的是建议的名称，则该名称为 ecs-demo-codedeploy-app。
4. 在 Deployment group (部署组) 中，选择您在 教程：部署 Amazon ECS 服务 (p. 111) 中创建的部署组。如果使用的是建议的名称，则该名称为 ecs-demo-dg。
5. 选择 Edit。
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。这应为HTTP: 8080。
7. 选择保存更改。

部署 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 Deployment group (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)。In修订位置中，输入 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)。

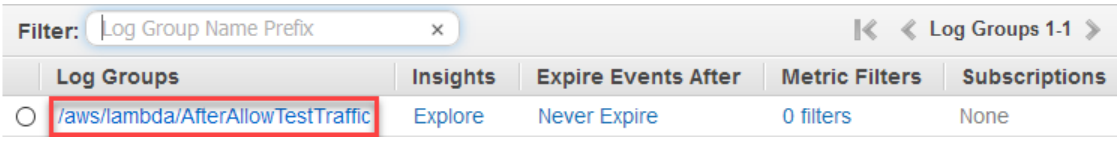
您可以在 Deployment status (部署状态) 中监控部署。在生产流量已全部路由至替换任务集之后，您可以选择终止原始任务集立即终止原始任务集。如果未选择 Terminate original task set (终止原始任务集)，则原始任务集将在您创建部署组时指定的持续时间之后终止。



第 6 步：在 CloudWatch Logs 中查看您的 Lambda 挂钩函数输出

如果部 CodeDeploy 成功，则 Lambda 挂钩函数中的验证测试也会成功。您可以在 CloudWatch log 日志中查看挂钩函数日志以进行确认。

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 从导航窗格中，选择 Logs (日志)。您应该看到您在 AppSpec 文件中指定的 Lambda 挂钩函数的新日志组。



3. 选择新的日志组。应当为 /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`AfterAllowTrafficLambda` 挂钩函数已成功。

Time (UTC +00:00)	Message
2019-09-11	
20:11:20	START RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916 Version: $LATEST
START RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916 Version: $LATEST	
20:11:21	2019-09-11T20:11:21.033Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO Entering AfterAllowTes
2019-09-11T20:11:21.033Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO Entering AfterAllowTestTraffic hook.	
20:11:21	2019-09-11T20:11:21.034Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO This is where AfterAllo
2019-09-11T20:11:21.034Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO This is where AfterAllowTestTraffic validation tests	
20:11:21	2019-09-11T20:11:21.789Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO AfterAllowTestTraffic v
2019-09-11T20:11:21.789Z e875485b-cdb2-4e1e-b4e8-8054e7b7f916 INFO AfterAllowTestTraffic validation tests succeeded	
20:11:21	END RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916
20:11:21	REPORT RequestId: e875485b-cdb2-4e1e-b4e8-8054e7b7f916 Duration: 977.80 ms Billed Duratio

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。本步骤中的资源名称是本教程中建议的名称（例如，**ecs-demo-codedeploy-app**作为 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. 使用删除部署组命令删除 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. 使用删除应用程序命令删除 CodeDeploy 应用程序。

```
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. 使用删除日志组命令删除 CloudWatch 日志组。

```
aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic
```

教程：使 Lambda 码部署和Amazon无服务器应用程序模型

AmazonSAM 是一个开源框架，用于构建无服务器应用程序。它转换和扩展 YAML 语法Amazon SAM将模板转换为Amazon CloudFormation语法构建无服务器应用程序，如 Lambda 函数。有关更多信息，请参阅 。什么是Amazon无服务器应用程序模型？

在本教程中，您将使用AmazonSAM 创建执行以下操作的解决方案：

- 创建 Lambda 函数。
- 创建 CodeDeploy 应用程序和部署组。
- 创建两个 Lambda 函数，以在 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 函数的更新会通过触发部署，从而 CodeDeploy 生产流量从 Lambda 函数的原始版本逐渐转移到更新版本。

> **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。这些费用包括可能对 CodeDeploy、Amazon CloudWatch 控和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

- Prerequisites (p. 125)
- 第 1 步：设置您的基础架构 (p. 125)
- 第 2 步：更新 Lambda 函数 (p. 135)
- 第 3 步：部署更新的 Lambda 函数 (p. 136)
- 第 4 步：查看部署结果 (p. 138)
- 第 5 步：清除 (p. 140)

Prerequisites

要完成本教程，您首先必须：

- 完成 开始使用 CodeDeploy (p. 26) 中的步骤。
- 安装 Amazon Serverless Application Model CLI。想要了解有关信息，请参阅安装AmazonSAM CLI.
- 创建 S3 存储桶。AmazonSAM 会上传AmazonSAM template (SAM 模板)放入此存储桶。

第 1 步：设置您的基础架构

本主题介绍如何使用Amazon SAM创建文件Amazon SAM模板和 Lambda 函数。然后，您将使用 Amazon SAM `package` 和 `deploy` 命令，在基础设施中生成组件。基础设施准备就绪之后，您应具备 CodeDeploy 应用程序和部署组、要更新和部署的 Lambda 函数，以及两个包含在部署 Lambda 函数时运行的验证测试的 Lambda 函数。完成后，您可以使用Amazon CloudFormation以查看您的组件和 Lambda 控制台或Amazon CLI测试 Lambda 函数。

主题

- 创建文件 (p. 125)
- 打包AmazonSAM 应用程序 (p. 133)
- 部署AmazonSAM 应用程序 (p. 133)
- （可选）检查和测试您的基础架构 (p. 134)

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AmazonSAM template (SAM 模板) (p. 126)
- 为您的 Lambda 函数创建一个文件 (p. 128)
- 为您的 “BeforeAllowTraffic” Lambda 函数创建一个文件 (p. 130)
- 为您的 AfterAllowTraffic Lambda 函数创建一个文件 (p. 131)

创建AmazonSAM template (SAM 模板)

创建AmazonSAM 模板文件，用于指定基础架构中的组件。

创建AmazonSAM template (SAM 模板)

1. 创建名为 `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: nodejs10.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: nodejs10.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: nodejs10.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.

Lambda 函数称为**myDateTimeFunction**

发布此 Lambda 函数时，AutoPublishAlias行将其链接到名为的别名live. 在本教程的后面，该函数的更新会通过 Amazon CodeDeploy 触发部署，从而将生产流量从原始版本逐渐转移到更新版本。

两个 Lambda 部署验证函数

以下 Lambda 函数在 CodeDeploy 生命周期挂钩期间执行。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。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。这些区域有：`CodeDeployRoleForLambda`角色允许调用 Lambda`invoke`方法仅在 Lambda 函数中，采用以此前缀开头的名称。有关更多信息，请参阅 。用于 AppSpec 的 “hooks” 部分AmazonLambda 部署 (p. 338)和PutLifecycleEventHookExecutionStatus中的CodeDeploy.

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

部署配置用于确定 CodeDeploy 应用程序将流量从 Lambda 函数的原始版本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AmazonSAM 模板。有关更多信息，请参阅Create a Deployment Configuration (p. 210)。

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`lambda:InvokeFunction`权限授予AmazonSAM 应用程序调用 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。

为您的 Lambda 函数创建一个文件

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. 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` 文件名形式保存到 `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）。由 Amazon SAM 创建的框架会检测并部署更新的函数版本。

> **Note**
>
> 这个 Lambda 函数也用于Amazon Cloud9教程。Amazon Cloud9是一个基于云的集成开发环境。有关如何在Amazon Cloud9，请参阅Amazon Lambda的教程Amazon Cloud9.

为您的 “BeforeAllowTraffic” Lambda 函数创建一个文件

为您创建文件`beforeAllowTraffic`挂钩 Lambda 函数。

1. 创建文本文件，并以 `beforeAllowTraffic.js` 文件名形式保存到 `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

为您创建文件`afterAllowTraffic`挂钩 Lambda 函数。

1. 创建文本文件，并以 `afterAllowTraffic.js` 文件名形式保存到 `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");
         }
        });
       }
      });
     }
```

打包AmazonSAM 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`

现在您已准备就绪，可以使用AmazonSAM PACKAGEsam package命令为您的 Lambda 函数和 CodeDeploy 应用程序创建并打包构件。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。此文件由AmazonSAM PACKAGEsam deploy命令执行该操作。

> **Note**
>
> 有关sam package命令，请参阅Amazon SAMCLI 命令参考中的Amazon Serverless Application Model开发人员指南.

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 your-S3-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。这些区域有：`output-template-file`指定了新文件的名称，该文件使用AmazonSAM PACKAGEsam deploy命令。

部署AmazonSAM 应用程序

使用AmazonSAM PACKAGEsam deploy命令与`package.yml`文件创建 Lambda 函数，以及 CodeDeploy 应用程序和部署组，使用Amazon CloudFormation.

> **Note**
>
> 有关sam deploy命令，请参阅Amazon SAMCLI 命令参考中的Amazon Serverless Application Model开发人员指南.

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
```

这些区域有：`--capabilities CAPABILITY_IAM`参数授权Amazon CloudFormation创建 IAM 角色。

（可选）检查和测试您的基础架构

本主题介绍了如何查看基础架构组件以及测试 Lambda 函数。

在运行 **sam deploy** 后查看堆栈结果

1. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
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`. 这些区域有：实体 ID列包含一个链接，用于查看 Lambda 控制台中的函数。

   Note

   的名称`myDateTimeFunction`Lambda 函数的前面是Amazon CloudFormation堆栈，并添加了标识符，因此看起来类似`my-date-time-app-myDateTimeFunction-123456ABCDEF`.
5. 从打开 CodeDeploy 控制台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. 您应该看到一个由Amazon CloudFormation名称开头为`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 Deployment configuration (部署配置) 下，您应当看到 CodeDeployDefault.LambdaLinear10PercentEvery1Minute。

（可选）测试函数（控制台）

1. 打开 Amazon Lambda 控制台，地址：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 Test (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 Test (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. 运行以下命令。将 *`your-function-arn`* 替换为函数的 ARN。

```
aws lambda invoke \
--function your-function-arn \
--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。

第 2 步：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。这将触发 CodeDeploy 通过将生产流量从 Lambda 函数的当前版本转移到更新版本来进行部署。

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;
  }
};
```

第 3 步：部署更新的 Lambda 函数

在此步骤中，您将使用更新的`myDateTimeFunction.js`更新并启动 Lambda 函数的部署。您可以在 CodeDeploy 或Amazon Lambda控制台。

Amazon SAM 模板中的 `AutoPublishAlias: live` 行使得基础设施会检测使用 `live` 别名的函数的更新。函数的更新会通过 CodeDeploy 触发部署，从而将生产流量从函数的原始版本转移到更新版本。

这些区域有：sam package和sam deploy命令用于更新和触发 Lambda 函数的部署。您已在 打包 AmazonSAM 应用程序 (p. 133) 和 部署AmazonSAM 应用程序 (p. 133) 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 your-S3-bucket
```

   这会创建一组新的构件，这些构件引用 S3 存储桶中的更新的 Lambda 函数。

2. 在 `SAM-Tutorial` 目录中，运行以下命令。

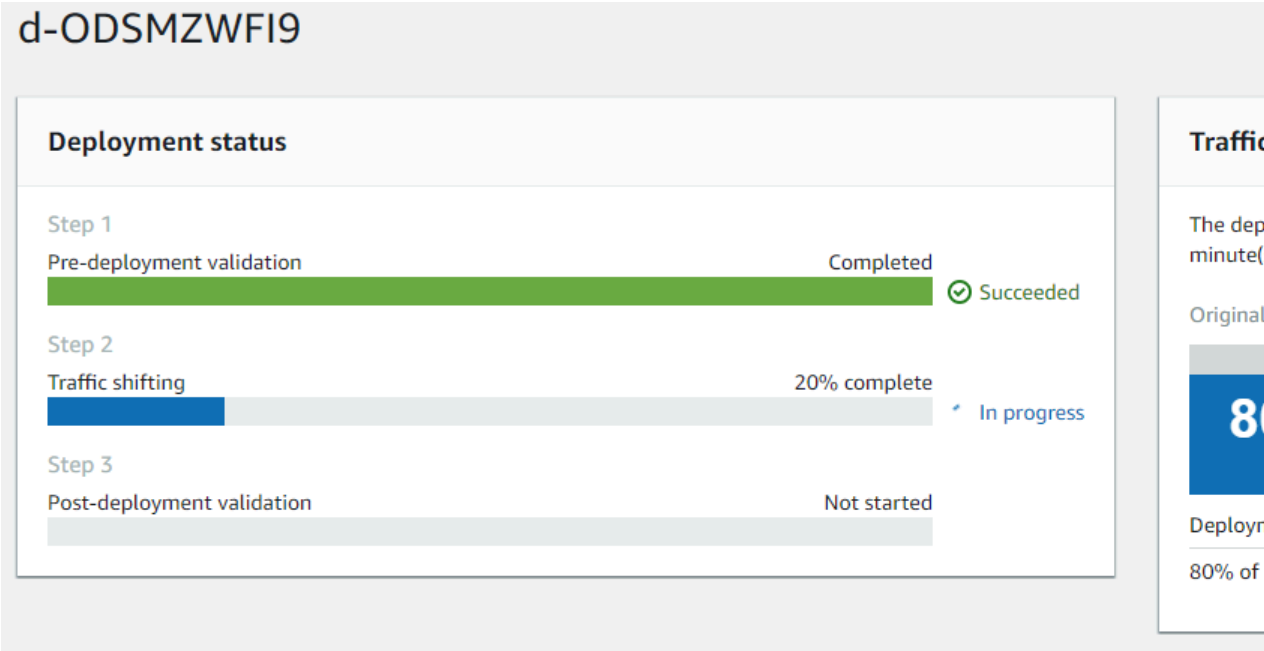
```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

   由于堆栈名称仍为 `my-date-time-app`，Amazon CloudFormation 会识别到这是一个堆栈更新。要查看更新的堆栈，可返回 Amazon CloudFormation 控制台，然后从导航窗格中，选择 Stacks (堆栈)。

（可选）在部署期间查看流量（CodeDeploy 控制台）

1. 从打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy/.
2. 在导航窗格中，展开 Applications (应用程序)，然后选择 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。

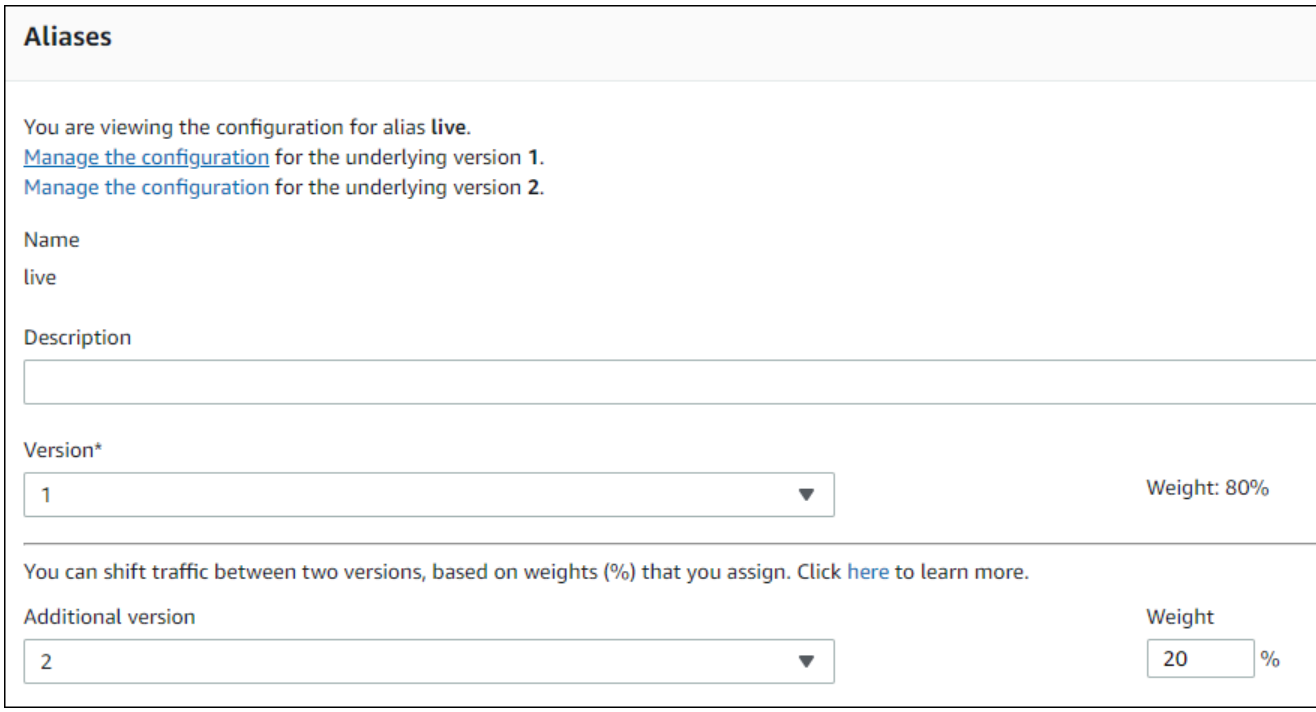
   此页面上的 Traffic shifting (流量转移) 进度条以及 Original (原始) 和 Replacement (替换) 框中的百分比显示了其进度。



（可选）在部署期间查看流量（Lambda 控制台）

1. 打开 Amazon Lambda 控制台，地址：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. 选择别名，然后选择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%。



第 4 步：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，您可以确认更新的 Lambda 函数收到了生产流量。如果部署失败，可以使用 CloudWatch Logs 在部署生命周期挂钩期间运行的 Lambda 函数中查看验证测试的输出。

主题

- 测试部署函数 (p. 138)
- 查看 CloudWatch Logs 中的挂接事件 (p. 139)

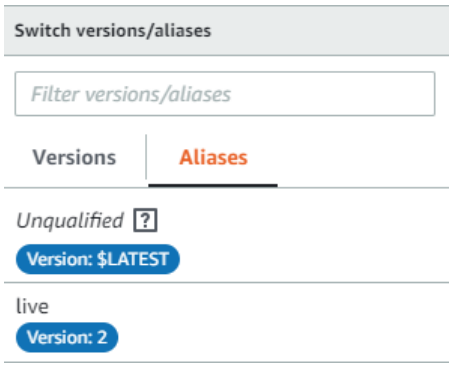
测试部署函数

这些区域有：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，地址：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，您应该看到版本：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 第 2 步：更新 Lambda 函数 (p. 135) 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 测试更新的函数，可运行以下命令，然后打开 `out.txt` 以确认结果中是否包含当前的小时、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 测试函数，您可能会收到意外结果。这是因为 CodeDeploy 会每隔一分钟将 10% 的流量转移到更新版本。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。

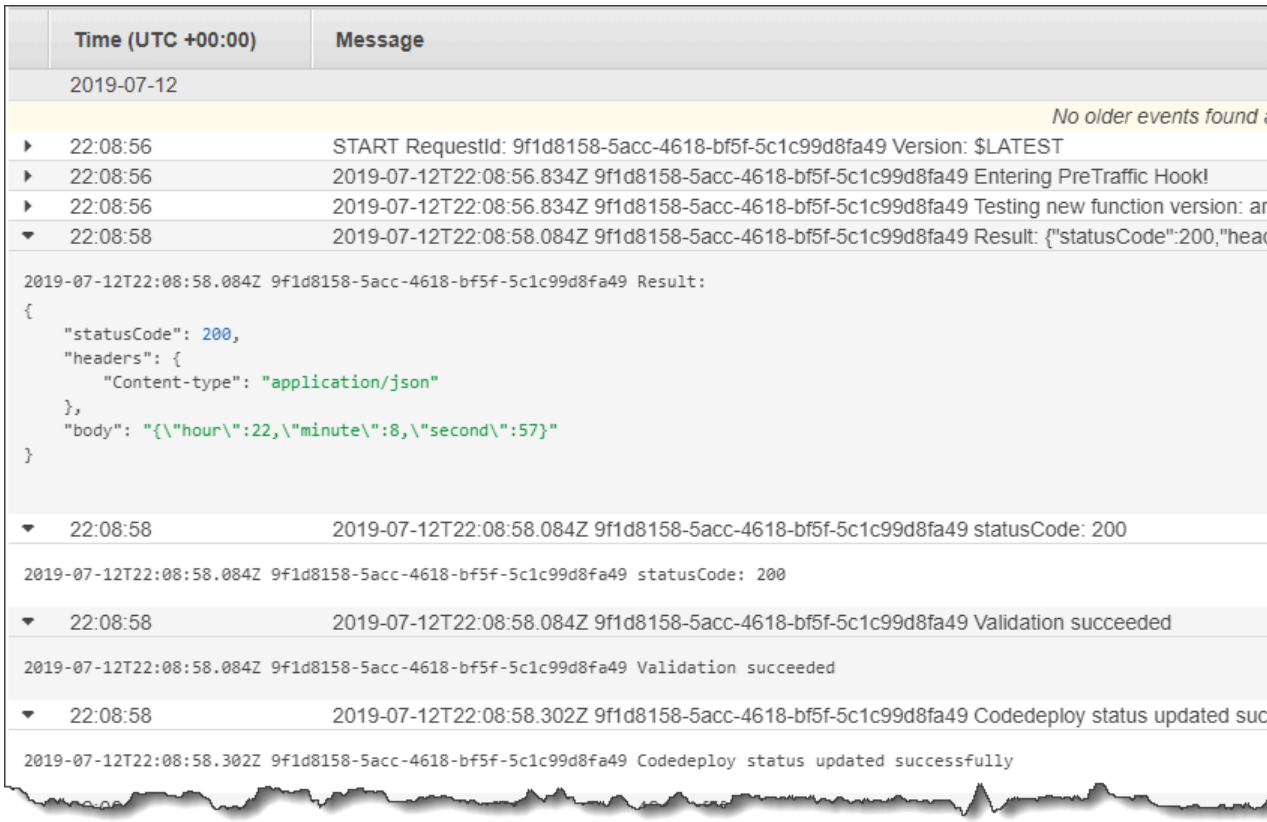
查看 CloudWatch Logs 中的挂接事件

在`BeforeAllowTraffic`挂钩时，CodeDeploy 将执行`CodeDeployHook_beforeAllowTrafficLambda` 函数。在`AfterAllowTraffic`挂钩时，CodeDeploy 将执行`CodeDeployHook_afterAllowTrafficLambda` 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 函数的更新成功，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 Logs 控制台查看验证测试。

查看 CodeDeploy 挂钩事件

1. 通过以下网址打开 CloudWatch 控制台：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：清除

为避免对您在本教程中所用资源收取更多费用，请删除Amazon SAM模板以及由 Lambda 验证函数创建的 CloudWatch 日志。

删除您的 Amazon CloudFormation 堆栈

1. 登录到Amazon Web Services Management Console并打开Amazon CloudFormation从控制台https://console.aws.amazon.com/cloudformation.
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)。Lambda 函数、CodeDeploy 应用程序和部署组以及通过创建的 IAM 角色Amazon SAM将被删除。

删除云监控日志中的日志

1. 通过以下网址打开 CloudWatch 控制台：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 旁的按钮。
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_beforeAllowTraffic 旁的按钮。
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)。

使用 CodeDeploy 代理

CodeDeploy 代理是一个软件包，在某个实例上安装和配置该软件包后，它将支持在 CodeDeploy 部署中使用该实例。

> **Important**
>
> CodeDeploy 代理的支持的最低版本为 1.1.0。使用较早的 CodeDeploy 代理可能会导致部署失败。
>
> **Note**
>
> 仅当您部署到 EC2/本地计算平台时，才需要 CodeDeploy 代理。使用 Amazon 云服务器或 AmazonLambda 计算平台。

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 Amazon CodeDeploy 在与实例交互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，请参阅CodeDeploy 代理配置参考 (p. 350)。

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 (p. 149)。

主题

- CodeDeploy 代理支持的操作系统 (p. 141)
- CodeDeploy 代理的通信协议和端口 (p. 142)
- CodeDeploy 代理的版本历史记录 (p. 142)
- 应用程序修订和日志文件清理 (p. 147)
- 由 CodeDeploy 代理安装的文件 (p. 147)
- 管理 CodeDeploy 代理操作 (p. 149)

CodeDeploy 代理支持的操作系统

支持的 Amazon EC2 AMI 操作系统

CodeDeploy 代理已在以下 Amazon EC2 AMI 操作系统上测试：

- Amazon Linux 2018.03.x、2017.03.x、2016.03.x、2016.03.x 以及 2016.09.x
- Amazon Linux 2（ARM、x86）
- Ubuntu 服务器 20.04 LTS、18.04 LTS、16.04 LTS 和 14.04 LTS
- Microsoft Windows Windows 服务器 2019、2016、2012 R2 以及 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

CodeDeploy 代理可以作为开源系统来满足您的需求。它可与其他 Amazon EC2 AMI 操作系统配合使用。有关更多信息，请转到CodeDeploy 代理存储 GitHub。

支持的本地操作系统

已在以下本地操作系统上测试 CodeDeploy 代理：

- Ubuntu 服务器 20.04 LTS、18.04 LTS、16.04 LTS 和 14.04 LTS
- Microsoft Windows Windows 服务器 2019、2016、2012 R2 以及 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

CodeDeploy 代理可以作为开源系统来满足您的需求。它可与其他本地实例操作系统配合使用。有关更多信息，请转到CodeDeploy 代理存储 GitHub。

CodeDeploy 代理的通信协议和端口

CodeDeploy 代理通过端口 443 使用 HTTPS 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实例相关信息。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 1.1.0。如果您运行的是更早的版本，则针对您实例的部署可能会失败。

下表列出了 CodeDeploy 代理的所有版本，以及每个版本所含的功能和增强功能。

版本	发行日期	详细信息
1.3.2	2021 年 5 月 6 日	Important CodeDeploy 1.3.2 地址2018-1000201，这会影响运行代理的 Windows 主机。CVE 引用了 Ruy-ffi，这是 CodeDeploy 代理的依赖关系。如果您的代理安装了 Amazon EC2 Systems Manager (SSM) 并设置为自动更新，则无需执行任何操作。否则，需要执行操作才能手动更新代理。要升级代理，请按照更新 Windows 服务器上的 CodeDeploy 代理。 Fixed：在 Ubuntu 20.04 及更 CodeDeploy 版本上安装代理程序时出现问题。 Fixed：解压缩文件时出现的间歇性问题，因为相对路径未得到正确处理。 添加了：对该项的支持Amazon PrivateLink和 VPC 终端节点的 Windows 实例。 添加了：对乌本图的 Support 20.10. 添加了：AppSpec 文件改进功能，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 (p. 278)。 • AppSpec 文件现在可以具有`.yaml`文件扩展名。 • 现在，您可以使用新的可选`file_exists_behavior`AppSpec 置。有关更多信息，请参阅AppSpec “文件” 部分（仅限 EC2/本部署）(p. 324)。 升级：CodeDeploy 现在使用Amazon适用于 Ruby 3.0 的开发工具包。
1.3.1	2020 年 12 月 22 日	Fixed：1.3.0 阻止本地实例启动的问题。

版本	发行日期	详细信息
1.3.0	2020 年 11 月 10 日	Important 此版本已弃用。 Fixed：已删除不再使用的过期证书。 Fixed：已从代理卸载脚本中删除提示消息Amazon Systems Manager，从而更轻松地将主机或队列降级为先前版本的代理。
1.2.1	2020 年 9 月 23 日	Changed：升级Amazon SDK for Ruby依赖关系从 v2 到 v3。 添加了：Support IMDSv2。如果 IMDSv2 http 请求失败，则包括一个静默回退到 IMDSv1。 Changed：更新了安全修补程序的 Rake 和 Rubyzip 依赖关系。 Fixed：确保空 PID 文件将返回`No CodeDeploy Agent Running`并在代理启动时清理 PID 文件。
1.1.2	2020 年 8 月 4 日	添加了：Support Ubuntu 服务器 19.10 和 20.04。 注意：：19.10 版本已到达其生命周期结束日期，并且不再受 Ubuntu 或 CodeDeploy 的支持。 添加了：Linux 和 Ubuntu 的内存效率提高，以更及时地释放预留内存。 添加了：与 Windows 服务器 “静默清理” 的兼容性导致代理在某些情况下无响应。 添加了：在清理过程中忽略非空目录，以避免部署失败。 添加了：对该项的支持Amazon本地区在洛杉矶（洛杉矶）。 添加了：从实例元数据中提取可用区以提供AmazonLocal Zones。 添加了：用户现在可以在子目录中提供其归档文件，而不需要将其存储在根目录中。 添加了：检测到 Rubyzip 可能导致内存泄漏的问题。更新了解压缩命令，以在使用 Rubyzip 之前首先尝试使用系统安装的解压缩实用程序。 添加了：`:enable_auth_policy:`作为代理配置设置。 Changed：解压缩警告现在会被忽略，因此部署将继续。
1.1.0	2020 年 6 月 30 日	Changed：CodeDeploy 代理的版本控制现在遵循 Ruby 标准版本控制约定。 添加了：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：删 CodeDeploy Linux 和 Ubuntu 的代理自动更新程序。要配置 CodeDeploy 代理的自动更新，请参阅使用安装 CodeDeploy 代理Amazon Systems Manager。

版本	发行日期	详细信息
1.0.1.1597	2018 年 11 月 15 日	Important CodeDeploy 代理的支持的最低版本为 1.1.0。使用较早的 CodeDeploy 代理可能会导致部署失败。 Enhancement：CodeDeploy 支持 Ubuntu 18.04。 Enhancement：CodeDeploy 支持 Ruby 2.5。 Enhancement：CodeDeploy 支持 FIPS 终端节点。有关 FIPS 终端节点的更多信息，请参阅 FIPS 140-2 概述。有关可用于 CodeBuild 的终端节点，请参阅CodeDeploy 区域和终端节点。
1.0.1.1518	2018 年 12 月 6 日	Enhancement：修正了一个问题，该问题导致 CodeDeploy 代理在接受轮询请求的同时关闭时出错。 Enhancement：新增了一项部署跟踪功能，可防止 CodeDeploy 代理在部署进行当中关闭。 Enhancement：改进了删除文件时的性能。
1.0.1.1458	2018 年 3 月 6 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：改进了证书验证，以支持更多受信任的机构。 Enhancement：修复了一个问题，该问题导致本地 CLI 在包含 BeforeInstall 生命周期事件的部署期间失败。 Enhancement：修复了更新 CodeDeploy 代理时，可能导致活动部署失败的问题。
1.0.1.1352	2017 年 11 月 16 日	注意：不再支持此版本。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，用于测试和调试安装了 CodeDeploy 代理的本地计算机或实例 EC2/本地部署。
1.0.1.1106	2017 年 5 月 16 日	注意：不再支持此版本。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。 Enhancement：使 CodeDeploy 代理与版本 2.9.2Amazon SDK for Ruby（aws-sdk-core 2.9.2）。
1.0.1.1095	2017 年 3 月 29 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：在中国（北京）地区引入了对 CodeDeploy 代理的支持。 Enhancement：允许 Puppet 在生命周期事件挂钩进行调用时，在 Windows Server 实例上运行。 Enhancement：改进了对`untar`操作。

版本	发行日期	详细信息
1.0.1.1067	2017 年 1 月 6 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：修改了许多错误消息，以包含导致部署失败的更具体的原因。 Enhancement：修复了导致 CodeDeploy 代理无法在某些部署中指定要部署的正确应用程序修订的问题。 Enhancement：已恢复使用`pushd`和`popd`之前和之后`untar`operation.
1.0.1.1045	2016 年 11 月 21 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：使 CodeDeploy 代理与版本 2.6.11 兼容。Amazon SDK for Ruby（aws-sdk-core 2.6.11）。
1.0.1.1037	2016 年 10 月 19 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 更新了用于 Amazon Linux、RHEL 和 Ubuntu 服务器实例的 CodeDeploy 代理，进行了以下更改。对于 Windows 服务器实例，最新版本仍为 1.0.1.998。 Enhancement：该代理现在可以确定哪个 Ruby 版本安装在实例上，以使它可以调用`codedeploy-agent`脚本使用该版本。
1.0.1.1011.1	2016 年 8 月 17 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：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：不再支持此版本。如果您使用此版本，您的部署可能会失败。 更新了用于 Amazon Linux、RHEL 和 Ubuntu 服务器实例的 CodeDeploy 代理，进行了以下更改。对于 Windows 服务器实例，最新版本仍为 1.0.1.998。 功能：增加了对使用 systemd 初始化系统的操作系统上使用 bash shell 调用 CodeDeploy 代理的支持。 Enhancement：在 CodeDeploy 代理和 CodeDeploy 代理更新程序中启用了对所有 Ruby 2.x 版本的支持。更新的 CodeDeploy 代理不再单纯依赖 Ruby 2.0。（CodeDeploy 代理安装程序的 deb 和 rpm 版本仍然需要 Ruby 2.0。）
1.0.1.998	2016 年 7 月 11 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 Enhancement：修复了对运行 CodeDeploy 代理以外的用户配置文件的支持根。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。

版本	发行日期	详细信息
1.0.1.966	2016 年 6 月 16 日	注意：不再支持此版本。如果您使用此版本，您的部署可能会失败。 功能：引入了用户配置文件运行 CodeDeploy 代理的支持根。 Enhancement：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。 Enhancement：使 CodeDeploy 代理与版本 2.3Amazon SDK for Ruby（aws-sdk-core 2.3）。 Enhancement：修复了与部署期间的 UTF-8 编码有关的问题。 Enhancement：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：不再支持此版本。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。 Enhancement：更新了安装脚本，如果已安装最新版本，则不会下载 CodeDeploy 代理。
1.0.1.934	2016 年 2 月 11 日	注意：不再支持此版本。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。
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。 Enhancement：使 CodeDeploy 代理与版本 2.2 兼容。Amazon SDK for Ruby（aws-sdk-core 2.2）。版本 2.1.2 仍受支持。
1.0.1.854	2015 年 11 月 17 日	注意：不再支持此版本。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：引入了版本跟踪支持`.version`文件。 功能：通过使用环境变量使部署组 ID 可用。 Enhancement：添加了对监视 CodeDeploy 代理日志的支持，使用Amazon CloudWatch Logs。

有关相关信息，请参阅下列内容：

- 确定 CodeDeploy 代理的版本 (p. 150)
- 安 CodeDeploy (p. 151)

有关 CodeDeploy 代理版本的历史记录，请参阅GitHub 上的发布存储库。

应用程序修订和日志文件清理

CodeDeploy 代理将修订和日志文件存档在实例上。CodeDeploy 代理将清理这些项目以节省磁盘空间。

应用程序修订部署日志：您可以使用: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的数目。CodeDeploy 还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。如果不指定任何值，则除了当前部署的修订之外，还将保留五个最新修订。

CodeDeploy 日志：对于 Amazon Linux、Ubuntu 服务器和 RHEL 实例，CodeDeploy 代理将在`/var/log/aws/codedeploy-agent`folder。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`。

由 CodeDeploy 代理安装的文件

CodeDeploy 代理在实例上的根目录中存储修订、部署历史记录和部署脚本。该目录的默认名称和位置是：

`'/opt/codedeploy-agent/deployment-root'`适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例。

`'C:\ProgramData\Amazon\CodeDeploy'`的 Windows Server 实例。

您可以使用root_dir设置来配 CodeDeploy 该目录的名称和位置。有关更多信息，请参阅CodeDeploy 代理配置参考 (p. 350)。

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。

  您可以使用批处理获取部署命令查找部署组 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。此文件列出部署的 AppSpec 文件中指定的所有脚本的输出。

  如果要查找某个部署的文件夹，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 (p. 266)。

  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max_revisions设置来更改该默认值。有关更多信息，请参阅CodeDeploy 代理配置参考 (p. 350)。

  > **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。当 CodeDeploy 代理将部署应用程序中的所有文件复制到实例时，创建此文件。它由 CodeDeploy 代理在下次部署期间使用，以确定哪些`ApplicationStop`和`BeforeInstall`要运行的脚本。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 编译每个部署的 `scripts.log` 文件的内容。`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。何时`codedeploy-agent-deployments.log`达到其最大大小时，CodeDeploy 代理将继续向其中写入内容，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。

> **Important**
>
> CodeDeploy 代理的最低受支持版本为 1.1.0。使用早期的 CodeDeploy 代理可能会导致部署失败。

主题

- 验证代 CodeDeploy 运行 (p. 149)
- 确定 CodeDeploy 代理的版本 (p. 150)
- 安 CodeDeploy (p. 151)
- 更新 CodeDeploy 代理 (p. 158)
- 卸载 CodeDeploy (p. 161)

验证代 CodeDeploy 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

- 验证适用于亚马逊 Linux 的 CodeDeploy 代理程序或 RHEL 是否正在运行 (p. 149)
- 验证 Ubuntu 服务器的 CodeDeploy 代理是否正在运行 (p. 150)
- 验证 Windows 服务器的 CodeDeploy 代理是否正在运行 (p. 150)

验证适用于亚马逊 Linux 的 CodeDeploy 代理程序或 RHEL 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：

```
sudo service codedeploy-agent status
```

如果命令返回错误，则没有安装 CodeDeploy 代理。请按照安装适用于亚马逊 Linux 或 RHEL 的 CodeDeploy 代理 (p. 153)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的消息`The AWS CodeDeploy agent is running.`

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：

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

验证 Ubuntu 服务器的 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：

```
sudo service codedeploy-agent status
```

如果命令返回错误，则没有安装 CodeDeploy 代理。请按照安装 Ubuntu 服务器的 CodeDeploy 代理 (p. 154)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的消息`The AWS CodeDeploy agent is running`.

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：

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

验证 Windows 服务器的 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 服务器的 CodeDeploy 代理 (p. 156)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。

> **Important**
>
> CodeDeploy 代理的最低受支持版本为 1.1.0。使用早期的 CodeDeploy 代理可能会导致部署失败。

首先，从代理版本 1.0.1.854 CodeDeploy 始，您可以在`.version`文件。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 and 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

- 确定亚马逊 Linux 或 RHEL 上的版本 (p. 151)
- 在 Ubuntu 服务器上确定版本 (p. 151)
- 在 Windows 服务器上确定版本 (p. 151)

确定亚马逊 Linux 或 RHEL 上的版本

登录到实例并运行以下命令：

```
sudo yum info codedeploy-agent
```

在 Ubuntu 服务器上确定版本

登录到实例并运行以下命令：

```
sudo dpkg -s codedeploy-agent
```

在 Windows 服务器上确定版本

登录到实例并运行以下命令：

```
sc qdescription codedeployagent
```

安 CodeDeploy

要在 CodeDeploy 实例或本地服务器上使用代理，必须先安装代理。我们建议安装和更新 CodeDeploy 代理程序Amazon Systems Manager，有关 Systems Manager 的详细信息，请参阅是什么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。

主题

- 使用安装 CodeDeploy 代理Amazon Systems Manager (p. 152)
- 使用命令行安装代理 (p. 153)

使用安装 CodeDeploy 代理Amazon Systems Manager

您可以使用Amazon Web Services Management Console或Amazon CLI将 CodeDeploy 代理安装到您的 Amazon EC2 或本地实例中，方法是使用Amazon Systems Manager. 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。

使用Amazon Systems Manager是安装和更新 CodeDeploy 代理的推荐方法。您还可以从 Amazon S3 存储桶安装代理。有关使用 Amazon S3 下载链接的更多信息，请参阅使用命令行安装代理 (p. 153).

主题

- Prerequisites (p. 152)
- 安 CodeDeploy (p. 152)

Prerequisites

按中的步骤操作。开始使用 CodeDeploy (p. 26)来设置 IAM 权限，Amazon CLI.

如果使用使 CodeDeploy Systems Manager 在本地服务器上安装代理，则必须向 Amazon EC2 Systems Manager 器注册本地服务器。有关更多信息，请参阅 。在混合环境中设置 Systems Manager中的Amazon Systems Manager用户指南.

安 CodeDeploy

您必须确保为正确配置实例，然后才能使用 Systems Manager 安装 CodeDeploy 代理。

安装或更新 SSM 代理

在 Amazon EC2 实例上，代 CodeDeploy 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，请执行该操作。

以下 Amazon EC2 AMI 预安装了 SSM 代理：

- 2016 年 11 月或之后发布的 Windows Server 2008-2012 R2 AMI
- Windows Server 2016 和 2019
- Amazon Linux 和 Amazon Linux 2
- Ubuntu Server 16.04 和 18.04
- Amazon ECS 优化

有关在运行 Linux 的实例上安装或更新 SSM 代理的信息，请参阅在 Linux 实例上安装和配置 SSM 代理中的Amazon Systems Manager用户指南.

有关在运行 Windows Server 的实例上安装或更新 SSM 代理的信息，请参阅在 Windows 实例上安装和配置 SSM 代理中的Amazon Systems Manager用户指南.

（可选）验证 Systems Manager 先决条件

在使用 Systems Manager 运行命令安装 CodeDeploy 代理之前，请确认您的实例满足最低 Systems Manager 要求。有关更多信息，请参阅 。设置Amazon Systems Manager中的Amazon Systems Manager用户指南.

安 CodeDeploy

使用 SSM，您可以安装 CodeDeploy 一次或设置计划来安装新版本。

要安装 CodeDeploy 代理，请选择`AWSCodeDeployAgent`软件包，而您按照使用安装或更新软件包Amazon Systems ManagerDistributor.

使用命令行安装代理

> **Note**
>
> 我们建议安 CodeDeploy 用Amazon Systems Manager以便能够配置代理的计划更新。有关更多信息，请参阅使用安装 CodeDeploy 代理Amazon Systems Manager (p. 152)。

可以按照以下主题使用命令行安装和运行 CodeDeploy 代理。

主题

- 安装适用于亚马逊 Linux 或 RHEL 的 CodeDeploy 代理 (p. 153)
- 安装 Ubuntu 服务器的 CodeDeploy 代理 (p. 154)
- 安装适用于 Windows 服务器的 CodeDeploy 代理 (p. 156)

安装适用于亚马逊 Linux 或 RHEL 的 CodeDeploy 代理

登录到实例，并依次运行以下命令。运行命令`sudo yum update`首先被认为是使用`yum`来安装软件包，但如果您不想更新所有软件包，也可以跳过它。

> **Note**
>
> 在第四个命令中,`/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。

```
sudo yum update
```

```
sudo yum install ruby
```

```
sudo yum install wget
```

要清除任何以前的代理缓存信息的 AMI，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
```

```
cd /home/ec2-user
```

在中国（北京）区域中：

- ```
  wget https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com/latest/install
  ```

在中国（宁夏）区域中：

- ```
  wget https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com/latest/install
  ```

```
chmod +x ./install
```

要安装最新版本的代理，请执 CodeDeploy 下操作：

- ```
  sudo ./install auto
  ```

要安装某个特定版本的代理，请执 CodeDeploy 下操作：

- ```
  sudo ./install auto -v releases/codedeploy-agent-###.rpm
  ```

要检查服务是否正在运行，请运行以下命令：

```
sudo service codedeploy-agent status
```

如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于的消息`The AWS CodeDeploy agent is running`.

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：

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

安装 Ubuntu 服务器的 CodeDeploy 代理

> **Note**
>
> 我们建议安 CodeDeploy 用Amazon Systems Manager以便能够配置代理的计划更新。有关更多信息，请参阅使用安装 CodeDeploy 代理Amazon Systems Manager (p. 152)。

在 Ubuntu 服务器上安装代理程序

1. 登录实例。
2. 根据的 Ubuntu Server 版本，执行以下操作之一：
   - 在 Ubuntu 服务器 14.04 上，依次输入以下命令：

     ```
     sudo apt-get update
     ```

     ```
     sudo apt-get install ruby2.0
     ```

     ```
     sudo apt-get install wget
     ```

   - 在 Ubuntu 服务器 16.04 及更高版本上，逐个输入以下命令：

     ```
     sudo apt update
     ```

     ```
     sudo apt install ruby-full
     ```

```
sudo apt install wget
```

3. 输入以下命令：

```
cd /home/ubuntu
```

*/home/ubuntu*代表 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：

在中国（北京）区域中：

-
```
wget https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com/latest/install
```

在中国（宁夏）区域中：

-
```
wget https://aws-codedeploy-cn-northwest-1.s3.cn-northwest-1.amazonaws.com/latest/
install
```

5. 输入以下命令：

```
chmod +x ./install
```

6. 请执行下列操作之一：

- 要在 Ubuntu 14.04、16.04 和 18.04 上安装最新版本的代理，请执行以下操作：

```
sudo ./install auto
```

- 要在 Ubuntu 20.04 上安装最新版本的代 CodeDeploy，请执行以下操作：

  Note

  将输出写入临时日志文件是一种解决方法，当我们使用`install`在 Ubuntu 20.04 上的脚本。

```
sudo ./install auto > /tmp/logfile
```

- 要在 Ubuntu 14.04、16.04 和 18.04 上安装特定版本的代理，请执行以下操作：

```
sudo ./install auto -v releases/codedeploy-agent-###.deb
```

- 要在 Ubuntu 20.04 上安装特定版本的代 CodeDeploy，请执行以下操作：

  Note

  将输出写入临时日志文件是一种解决方法，当我们使用`install`在 Ubuntu 20.04 上的脚本。

```
sudo ./install auto -v releases/codedeploy-agent-###.deb > /tmp/logfile
```

检查服务是否正在运行

1. 输入以下命令：

```
sudo service codedeploy-agent status
```

如果 CodeDeploy 代理已安装且正在运行，您应该会看到一条类似于的消息`The AWS CodeDeploy agent is running.`

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：

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

安装适用于 Windows 服务器的 CodeDeploy 代理

在 Windows 服务器实例上，您可以使用下列方法之一来下载并安装代理：

- 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**
>
> 我们建议安 CodeDeploy 用Amazon Systems Manager以便能够配置代理的计划更新。有关更多信息，请参阅使用安装 CodeDeploy 代理Amazon Systems Manager (p. 152)。

主题

- 使用 Windows PowerShell (p. 156)
- 使用直接链接 (p. 157)
- 使用 Amazon S3 复制命令 (p. 158)

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. 配置Amazon凭证使用`Set-AWSCredential`和`Initialize-AWSDefaultConfiguration`命令。有关更多信息，请参阅 。使用Amazon凭证中的Amazon工具 PowerShell 户指南.
5. 下载 CodeDeploy 装文件。

   针对中国（北京）区域：

要安装最新版本的代理，请执 CodeDeploy 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
  ```

要安装某个特定版本的代理，请执 CodeDeploy 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
  ```

针对中国（宁夏）区域：

要安装最新版本的代理，请执 CodeDeploy 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
  ```

要安装某个特定版本的代理，请执 CodeDeploy 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 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 CodeDeploy 代理刚刚安装了 CodeDeploy 代理并且未启动，则在运行Get-Service命令，下状态，您应该看到**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，下状态，您应该看到**Running**：

```
Status     Name                DisplayName
------     ----                -----------
Running    codedeployagent    CodeDeploy Host Agent Service
```

使用直接链接

如果 Windows 服务器实例上的浏览器安全设置提供权限（例如，对的权限）`http://*.amazonaws.com`），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。

在中国（北京）区域中：

```
https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com.cn/latest/codedeploy-
agent.msi
```

在中国（宁夏）区域中：

```
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Amazon CLI已安装在实例上，则您可以使用 Amazon S3CP 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。

在中国（北京）区域中：

```
aws s3 cp s3://aws-codedeploy-cn-north-1/latest/codedeploy-agent.msi codedeploy-agent.msi
```

在中国（宁夏）区域中：

```
aws s3 cp s3://aws-codedeploy-cn-northwest-1/latest/codedeploy-agent.msi codedeploy-
agent.msi
```

更新 CodeDeploy 代理

您可以使用在所有受支持的操作系统上配置 CodeDeploy 代理的自动计划更新。Amazon Systems Manager. 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

- 在亚马逊 Linux 或 RHEL 上更新 CodeDeploy 代理 (p. 158)
- 更新 Ubuntu 服务器上的 CodeDeploy 代理 (p. 158)
- 更新 Windows 服务器上的 CodeDeploy 代理 (p. 159)

在亚马逊 Linux 或 RHEL 上更新 CodeDeploy 代理

要使用配置 CodeDeploy 代理的自动计划更新，请指定。Amazon Systems Manager，请按照安装 CodeDeploy 代理Amazon Systems Manager.

如果您要强制更新 CodeDeploy 代理，请登录到相应的实例，并运行以下命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Ubuntu 服务器上的 CodeDeploy 代理

要使用配置 CodeDeploy 代理的自动计划更新，请指定。Amazon Systems Manager，请按照安装 CodeDeploy 代理Amazon Systems Manager.

如果您要强制更新 CodeDeploy 代理，请登录到相应的实例，并运行以下命令：

```
sudo /opt/codedeploy-agent/bin/install auto
```

更新 Windows 服务器上的 CodeDeploy 代理

您可以使用启用 CodeDeploy 代理的自动更新。Amazon Systems Manager. 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

- 使用设置 CodeDeploy 代理更新Amazon Systems Manager (p. 159)
- 手动更新 CodeDeploy 代理 (p. 159)
- （已弃用）使用 Windows 服务器更新程序更新代理 (p. 159)

使用设置 CodeDeploy 代理更新Amazon Systems Manager

要配置 Systems Manager 并启用代理 CodeDeploy 自动更新，请按照使用安装 CodeDeploy 代理Amazon Systems Manager.

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。按照中的说明进行操作安 CodeDeploy。建议您卸载旧版本的 CodeDeploy 代理，方法是按照卸载 CodeDeploy.

（已弃用）使用 Windows 服务器更新程序更新代理

> **Note**
>
> 适 CodeDeploy 于 Windows 服务器的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。

要启用自动更新代理的功 CodeDeploy，请在新实例或现有实例上安装适用于 Windows 服务器的代理更新程序。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 服务器的 CodeDeploy 代理 (p. 156).

在 Windows 服务器实例上，您可以通过以下方式下载并安装代理更新程序：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 (p. 159)
- 使用直接链接 (p. 160)
- 使用 Amazon S3 复制命令 (p. 160)

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
```

在中国（北京）区域中：

- ```
  powershell.exe -Command Read-S3Object -BucketName aws-codedeploy-cn-north-1 -Key latest/
  codedeploy-agent-updater.msi -File c:\temp\codedeploy-agent-updater.msi
  ```

在中国（宁夏）区域中：

- ```
  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 服务器实例上的浏览器安全设置提供必需的权限（例如，对的权限）`http://*.amazonaws.com`），则可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。

在中国（北京）区域中：

```
https://aws-codedeploy-cn-north-1.s3.cn-north-1.amazonaws.com.cn/latest/codedeploy-
agent.msi
```

在中国（宁夏）区域中：

```
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Amazon CLI已安装在实例上，则您可以使用 Amazon S3CP 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。

在中国（北京）区域中：

```
aws s3 cp s3://s3-cn-north-1/latest/codedeploy-agent-updater.msi codedeploy-agent-
updater.msi
```

在中国（宁夏）区域中：

```
aws s3 cp s3://s3-cn-northwest-1/latest/codedeploy-agent-updater.msi codedeploy-agent-
updater.msi
```

卸载 CodeDeploy

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。

从亚马逊 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 服务器卸载代理

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 服务器卸载 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。

使用 CodeDeploy 的实例

CodeDeploy 支持对运行 Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL) 和 Windows Server 的实例进行部署。

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 代理并连接到公有的 Amazon EC2 实例以外的任何物理设备Amazon服务终端节点。您可以使用 CodeDeploy 将一个应用程序同时部署到云中的 Amazon EC2 实例以及办公室中的桌面 PC 或您自己的数据中心内的服务器。

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：

Subject	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。	是	是
需要实例能够连接到 CodeDeploy。	是	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 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。	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： • 为每个实例创建一个 IAM 用户，并以纯文本形式将 IAM 用户的账户凭证存储在实例上。 • 创建一个 IAM 角色，该角色可让 IAM 用户在每个实例上都能担任，以获取通过生成的定期刷新的临时凭证。Amazon Security Token Service。	否	是
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，然后才能对其进行部署。	是	是
作为 CodeDeploy 部署的一部分，可以参与 Amazon EC2 Auto Scaling 和 Elastic Load Balancing 方案。	是	否

Subject	Amazon EC2 实例	本地实例
可从 Amazon S3 存储桶和 GitHub 存储库进行部署。	是	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。	是	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：

我要启动新的 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) (p. 174)。 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) (p. 170)。
我要启动新的 Ubuntu 服务器或 RHEL Amazon EC2 实例。	请参阅 为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CLI或 Amazon EC2 控制台) (p. 170)。
我想要配置一个亚马逊 Linux、Windows 服务器、Ubuntu 服务器或 RHEL Amazon EC2 实例。	请参阅 配置 Amazon EC2 实例以使用 CodeDeploy (p. 177)。
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 (p. 179)。
我希望 CodeDeploy 在蓝/绿部署期间预置实例队列。	请参阅 在 CodeDeploy 中使用部署 (p. 253)。

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 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

主题

- Tagging Instances for Deployments (p. 163)
- Working with Amazon EC2 Instances (p. 170)
- Working with On-Premises Instances (p. 179)
- View Instance Details (p. 202)
- Instance Health (p. 203)

在 CodeDeploy 中标记部署组的实例

要帮助管理 Amazon EC2 实例和本地实例，则需使用标签为每个资源分配您自己的元数据。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。每个标签都包含您定义的一个键和一个可选值。有关更多信息，请参阅 。标签您的 Amazon EC2 资源。

要指定 CodeDeploy 部署组中包含的实例，则需在一个或多个标签组。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**
>
> 您还可以在部署组中包含 Amazon EC2 Auto Scaling 组，但这些组是根据其名称识别的，而不是根据实例应用的标签识别。有关信息，请参阅 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

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：单个标签组，单个标签 (p. 164)
- 示例 2：单个标签组，多个标签 (p. 165)
- 示例 3：多个标签组，单个标签 (p. 166)
- 示例 4：多个标签组，多个标签 (p. 168)

示例 1：单个标签组，单个标签

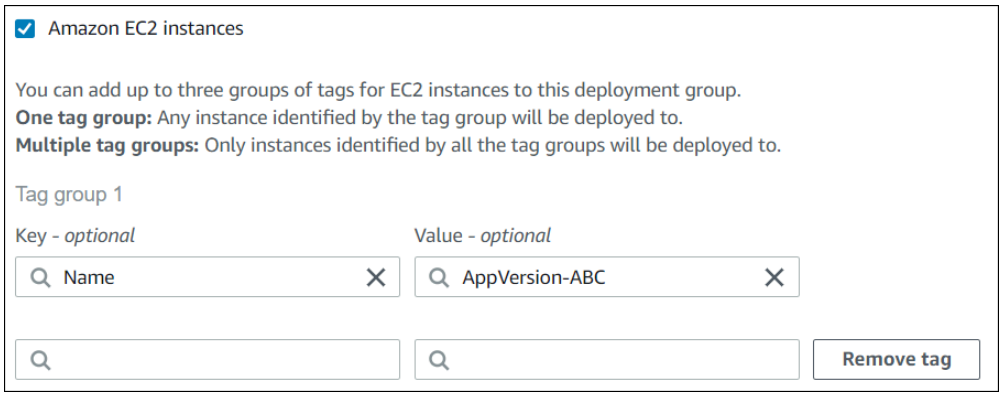
您可以在唯一标签组中指定唯一标签：

标签组 1

密钥	值
名称	AppVersion-ABC

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：单个标签组，多个标签

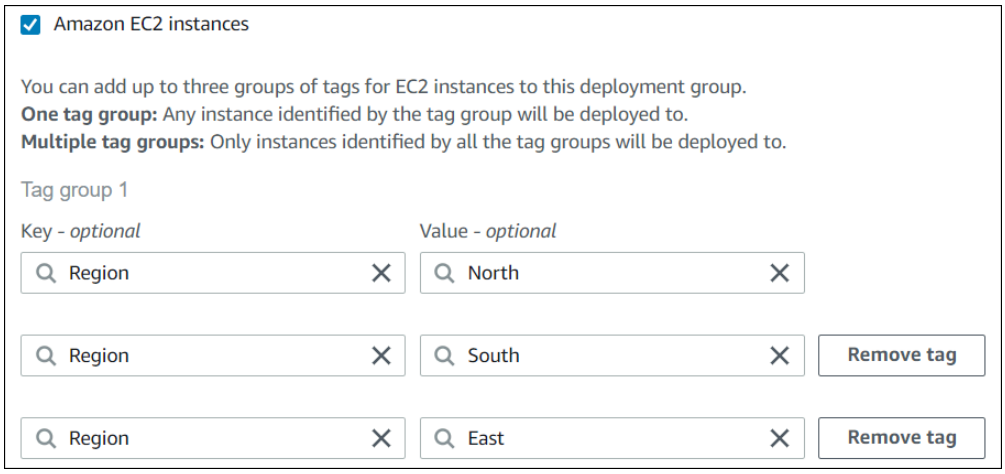
您也可以在唯一标签组中指定多个标签：

标签组 1

密钥	值
区域	北部
区域	南部
区域	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

密钥	值
名称	AppVersion-ABC

标签组 2

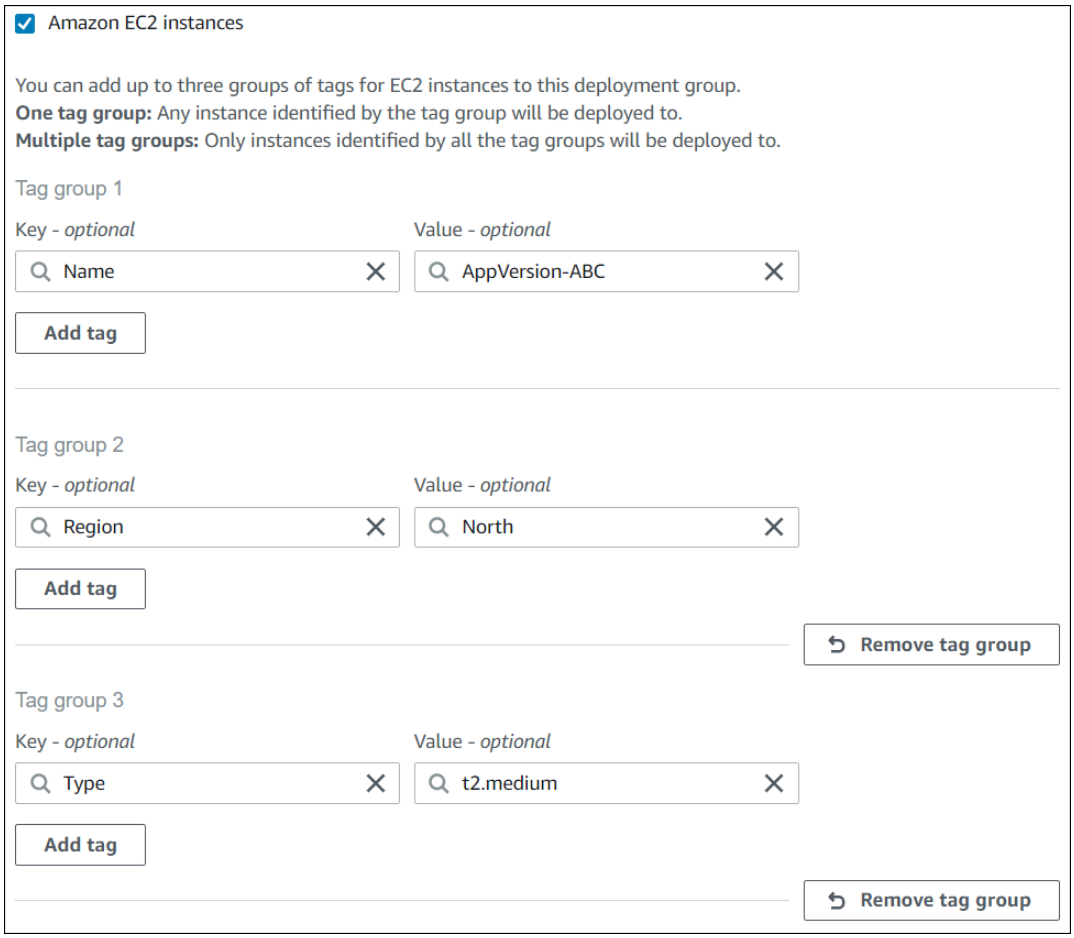
密钥	值
区域	北部

标签组 3

密钥	值
类型	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

密钥	值
Environment	测试版
Environment	生产前调试

标签组 2

密钥	值
区域	北部
区域	南部
区域	东部

标签组 3

密钥	值
类型	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。

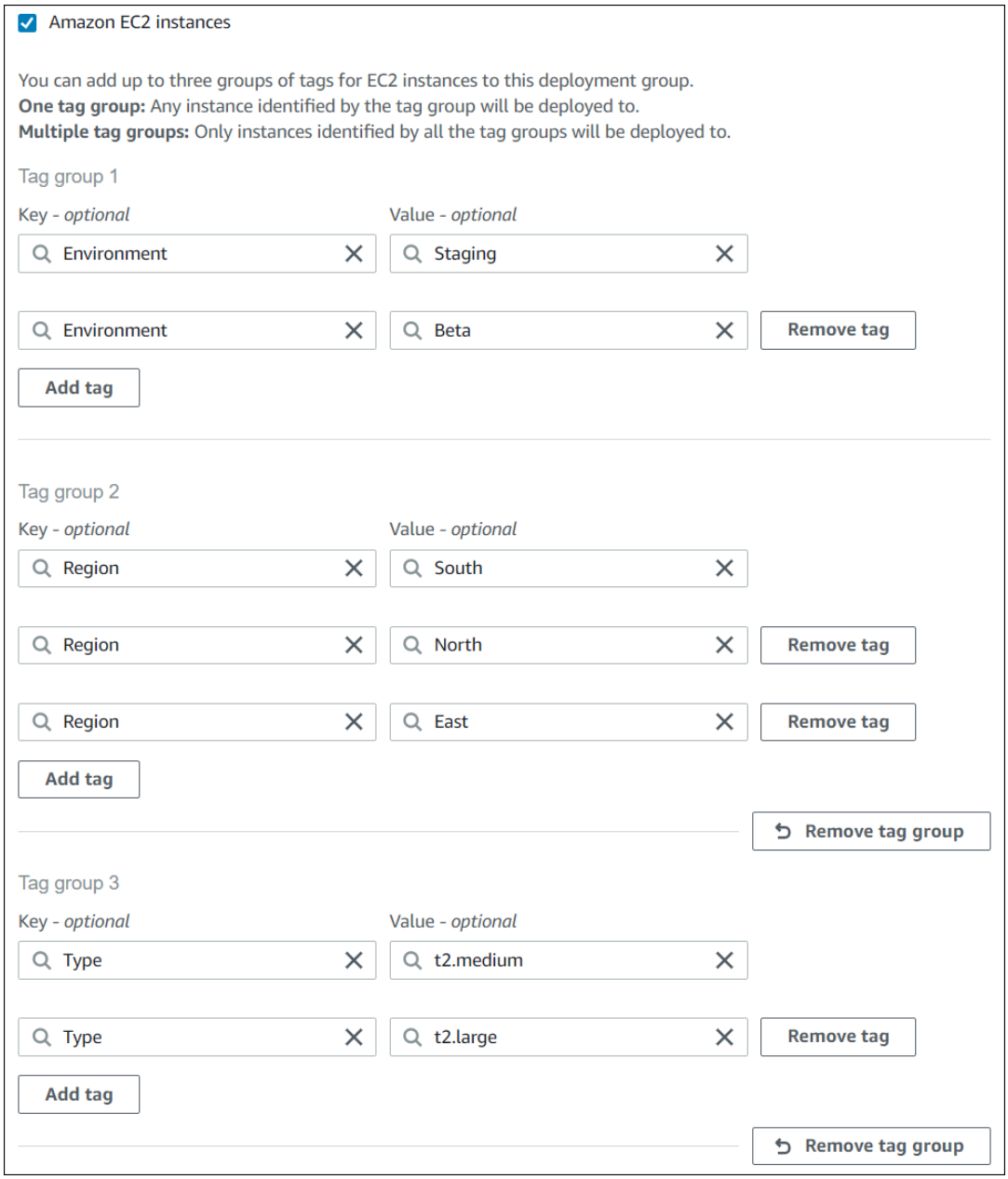
例如，具有以下标签组的实例将会 包含在部署组中：

- `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：



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 Amazon EC2 CodeDeploy 例

Amazon EC2 实例是一种虚拟计算环境，可使用 Amazon Elastic Compute Cloud 创建并配置。Amazon EC2 提供可扩展的计算容量Amazon云。您可以根据需要，使用 Amazon EC2 启动部署 CodeDeploy 的任意数量的虚拟服务器。

有关 Amazon EC2 的更多信息，请参阅Amazon EC2 入门指南.

本节中的说明介绍了如何创建和配置 Amazon EC2 实例，以供在 CodeDeploy 部署中使用。

主题

- 为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CLI或 Amazon EC2 控制台) (p. 170)
- 为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CloudFormation模板) (p. 174)
- 配置 Amazon EC2 实例以使用 CodeDeploy (p. 177)

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。

你可以使用我们的Amazon CloudFormation模板，以启动已配置为在 CodeDeploy 部署中使用的运行 Amazon Linux 或 Windows 服务器的 Amazon EC2 实例。我们不提供Amazon CloudFormation运行 Ubuntu Server 或 Red Hat Enterprise Linux (RHEL) 的 Amazon EC2 实例的模板。有关模板使用的替代方法，请参阅使用 CodeDeploy 的实例 (p. 162)。

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）

Prerequisites

如果您尚未这样做，请按照中的说明操作。开始使用 CodeDeploy (p. 26)设置和配置Amazon CLI创建 IAM 实例配置文件。

启动 Amazon EC2 实例

1. 登录 Amazon Web Services Management Console，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. 在存储库的第 1 步：选择一个Amazon 系统映像(AMI)页面，来自快速启动选项卡中，找到要使用的操作系统和版本，然后选择Select. 您必须选择 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 (p. 141)。
4. 在存储库的步骤 2: 选择一个实例类型页上，选择任何可用的 Amazon EC2 实例类型，然后选择后续：配置实例详细信息.
5. 在存储库的步骤 3: 配置实例详细信息页面，在IAM 角色列表中，选择您在中创建的 IAM 实例角色第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31). 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**
   >
   > 如果启动到 EC2-Classic或者默认的虚拟私有云 (VPC) 不显示在网络列表中，并且您无法选择支持启动到 EC2-Classic 的其他 Amazon EC2 实例类型，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 Next:。添加存储.
7. 离开步骤 4: 添加存储页面不变，然后选择后续：添加标签.
8. 在存储库的第 5 步：添加标签页面，选择添加标签.
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:。配置安全组.
11. 在存储库的步骤 6：配置安全组页面，离开创建新安全组选项已选择。

    为运行 Amazon Linux、Ubuntu 服务器或 RHEL 的 Amazon EC2 实例配置默认 SSH 角色。系统为运行 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，建议您将访问限制到 SSH、RDP 和 HTTP 端口，而不是指定任何地方 0.0.0.0.0.0.0/0. 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 。保护 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. 离开步骤 7：核查实例启动页面不变，然后选择启动.

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 key pair，可在此处选择它。

    如果您尚不拥有 Amazon EC2 实例 key pair，请创建新 key pair给它一个可识别的名称。选择下载密钥对将 Amazon EC2 实例 key pair 下载到您的计算机。

    Important

    如果您需要使用 SSH 或 RDP 访问 Amazon EC2 实例，则必须具有 key pair。
15. 选择 Launch Instances。
16. 选择您的 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 Amazon EC2 实例上安装该代理。有关更多信息，请参阅安 CodeDeploy (p. 151)。

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)

Prerequisites

如果您尚未这样做，请按照中的说明操作。开始使用 CodeDeploy (p. 26)设置和配置Amazon CLI创建 IAM 实例配置文件。

启动 Amazon EC2 实例

1. 仅适用于 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，要创建名为的安全组Code 部署演示-Windows-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，建议您将访问限制到 RDP 和 HTTP 端口。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 。保护 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.

- Amazon EC2 实例的类型名称 (*####*) 你创建的，例如`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，请参阅第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31).
- Amazon EC2 实例 key pair 的名称 (*####*) 以允许针对运行 Amazon Linux、Ubuntu 服务器或 RHEL 或 RDP 的 Amazon EC2 实例的 SSH 访问，以访问运行 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 key pair 名称，请在以下位置打开 Amazon EC2 控制台：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 key pair，请参阅使用 Amazon EC2 创建 key pair. 确保在中列出的其中一个区域中创建 key pair区域和终端节点在Amazon一般参考. 否则，您无法将 Amazon EC2 实例 key pair 与 CodeDeploy 结合使用。

对于 Amazon Linux、RHEL 和 Ubuntu 服务器

调用run-instances命令以启动运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例并附加您在中创建的 IAM 实例配置文件。第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31). 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，建议您将访问限制到 SSH 和 HTTP 端口。CodeDeploy 不需要无限制的端口访问，也不需要 HTTP 端口访问。有关更多信息，请参阅 。保护 Amazon EC2 实例的提示.

适用于 Windows Server

调用run-instances命令以启动运行 Windows Server 的 Amazon EC2 实例并附加您在中创建的 IAM 实例配置文件。第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)，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 IAM 实例配置文件启动一个具有指定的 AMI、key pair 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，可稍后通过调用describe-instances针对 Amazon EC2 实例 key pair 的命令。

```
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，该命令将标记 Amazon EC2 实例，以便 CodeDeploy 可以在部署过程中找到它。在以下示例中，将命名为标签**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 Amazon EC2 实例上安装该代理。有关更多信息，请参阅安 CodeDeploy (p. 151)。

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)

你可以使用我们的Amazon CloudFormation模板，快速启动运行 Amazon Linux 或 Windows Server 的 Amazon EC2 实例。您可以使用Amazon CLI、CodeDeploy 控制台或Amazon使用 API 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。
- 在实例上安装并运行 CodeDeploy 代理。

您不一定要使用我们的Amazon CloudFormation设置 Amazon EC2 实例。有关替代方法，请参阅使用 CodeDeploy 的实例 (p. 162)。

我们不提供Amazon CloudFormation针对运行 Ubuntu Server 或 Red Hat Enterprise Linux (RHEL) 的 Amazon EC2 实例的模板。

主题

- 开始前的准备工作 (p. 175)
- 启动 Amazon EC2 实例Amazon CloudFormation模板（控制台）(p. 175)

- 启动 Amazon EC2 实例Amazon CloudFormation(模板)Amazon CLI) (p. 176)

开始前的准备工作

在您可以使用Amazon CloudFormation模板启动 Amazon EC2 实例，请确保完成以下步骤。

1. 确保您已完成第 3 步：预配 IAM 用户 (p. 30)中的步骤。仔细检查分配给 IAM 用户的策略是否具有以下最低权限，然后添加任何不存在的权限：
   - 云形成：*
   - codedeploy:*
   - ec2:*
   - 我：addroleto 实例配置文件
   - IAM：创建实例配置文件
   - iam:CreateRole
   - 我：删除实例配置文件
   - iam:DeleteRole
   - iam:DeleteRolePolicy
   - iam:GetRole
   - iam:DeleteRolePolicy
   - iam:PutRolePolicy
   - 我：从实例配置文件中删除角色
2. 请确保您有一个实例 key pair，以允许针对运行 Amazon Linux 的 Amazon EC2 实例的 SSH 访问或针对运行 Windows Server 的实例的 RDP 访问。

   要查找 key pair 名称，请在以下位置打开 Amazon EC2 控制台：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 key pair，请参阅使用 Amazon EC2 创建 key pair. 确保 key pair 是在中列出的其中一个区域中创建的区域和终端节点在Amazon一般参考. 否则，您无法将实例 key pair 与 CodeDeploy 结合使用。

启动 Amazon EC2 实例Amazon CloudFormation模板（控制台）

1. 登录到Amazon Web Services Management Console然后打开Amazon CloudFormation控制台https://console.aws.amazon.com/cloudformation.

   > **Important**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)中使用的同一账户登录 Amazon Web Services Management Console。在导航栏的区域选择器中，选择一个在中列出的区域。区域和终端节点在Amazon一般参考. CodeDeploy 仅支持这些区域。
2. 选择 Create Stack。
3. InChoose a template，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）。
- 适用于KeyPairName中，键入实例 key pair 名称。仅键入密钥对名称而不是密钥对文件扩展名。
- 适用于OperatingSystem框，键入**Windows**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

  默认值**0.0.0.0/0**仅用于演示目的。CodeDeploy 不需要 Amazon EC2 实例具有对端口的无限制访问权限。作为最佳实践，建议您限制对 SSH（和 HTTP）端口的访问。
- 适用于TagKey，键入 CodeDeploy 将用于在部署过程中标识实例的实例标签密钥（或保留默认值）名称)。
- 适用于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标签与 CodeDeploy 标签不同。Amazon CloudFormation使用标签来简化基础架构的管理。CodeDeploy 使用标签来识别 Amazon EC2 实例。您在上指定了 CodeDeploy 标签指定参数页.
7. 在存储库的审核在页面上，Capabilities，选择我承认这一点Amazon CloudFormation可能会创建 IAM 资源框，然后选择Create.

   晚于Amazon CloudFormation在中创建堆栈并启动 Amazon EC2 实例Amazon CloudFormation控制台，创建 _ 完成将显示在状态column. 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 (p. 149)，然后执行使用 CodeDeploy 创建应用程序 (p. 212).

启动 Amazon EC2 实例Amazon CloudFormation(模板)Amazon CLI）

1. 在调用 create-stack 命令时使用 Amazon CloudFormation 模板。此堆栈将启动安装 CodeDeploy 代理的新 Amazon EC2 实例。

   要启动运行 Amazon Linux 的 Amazon EC2 实例：

```
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：

```
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
```

*###*是实例 key pair 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）

此命令会创建Amazon CloudFormation名为的堆栈**`CodeDeployDemoStack`**，使用Amazon CloudFormation指定的 Amazon S3 存储桶中的模板。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 **`CodeDeployDemoStack`** 的 Amazon CloudFormation 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 (p. 149)，然后执行使用 CodeDeploy 创建应用程序 (p. 212).

配置 Amazon EC2 实例以使用 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 Amazon EC2 实例，以便在 CodeDeploy 部署中使用。

Note

如果您没有 Amazon EC2 实例，可使用Amazon CloudFormation模板来启动一个运行亚马逊 Linux 或 Windows 服务器的模板。我们不提供 Ubuntu Server 或 RHEL 的模板。

第 1 步：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 Management Console，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 Amazon EC2 实例。
4. 在详细信息窗格中的说明选项卡中的值，请注意IAM 角色字段，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。想要了解有关信息，请参阅将 IAM 角色附加到实例.

第 2 步：验证附加的 IAM 实例配置文件具有正确的访问权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles（角色）。
3. 浏览并选择您在上一部分的步骤 4 中记下的 IAM 角色名称。

   Note

   如果您希望使用由 Amazon CloudFormation 模板生成的服务角色，而不使用您按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明创建的服务角色，请注意以下事项：在我们的某些版本中Amazon CloudFormation模板，生成并附加到 Amazon EC2 实例的 IAM 实例配置文件的显示名称与 IAM 控制台中的显示名称不相同。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，您将看到`CodeDeploySampleStack-expnyi6-InstanceRole`两者都是相同的。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。如果没有进入受信任实体读取选项提供商 ec2.amazonaws.com，则您无法使用此 Amazon EC2 实例。使用中的信息停止并创建 Amazon EC2 实例使用 CodeDeploy 的实例 (p. 162).

   如果有显示的条目选项提供商 ec2.amazonaws.com，并且您只需要将应用程序存储在 GitHub 存储库中，请向前跳到第 3 步：为 Amazon EC2 实例添加标签 (p. 179).

   如果有显示的条目选项提供商 ec2.amazonaws.com，并且您要将应用程序存储在 Amazon S3 存储桶中，选择Permissions (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*"`位于指定操作的列表中。

   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：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          ... Some actions may already be listed here ...
          "s3:Get*",
          "s3:List*"
          ... Some more actions may already be listed here ...
        ],
        ...
      }
    ]
  }
```

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

我们建议您将此策略限制为您的 Amazon EC2 实例必须访问的 Amazon S3 存储桶。请确保授予对包含 CodeDeploy 代理的 Amazon S3 存储桶的访问权限。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:::replace-with-your-s3-bucket-name/*",
                "arn:aws:s3:::aws-codedeploy-cn-north-1/*",
                "arn:aws:s3:::aws-codedeploy-cn-northwest-1/*"
            ]
        }
    ]
}
```

第 3 步：为 Amazon EC2 实例添加标签

有关如何标记 Amazon EC2 实例以便 CodeDeploy 可以在部署过程中找到它的说明，请参阅在控制台中使用标签，然后返回此页。

Note

您可以使用所需的任何密钥和值标记 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。

第 4 步：安装Amazon CodeDeployAmazon EC2 实例上的代理

有关如何在 Amazon EC2 实例上安装 CodeDeploy 代理并验证其是否运行的说明，请参阅管理 CodeDeploy 代理操作 (p. 149)，然后执行使用 CodeDeploy 创建应用程序 (p. 212).

使用本地实例以进行 CodeDeploy

本地实例是可运行 CodeDeploy 代理并连接到公有的 Amazon EC2 实例以外的任何物理设备。Amazon服务终端节点。

将 CodeDeploy 应用程序修订部署到本地实例包含两个主要步骤：

- 步骤 1— 配置每个本地实例，将它注册到 CodeDeploy，然后为它添加标签。
- 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（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实

例 (p. 82)。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 (p. 179)。

如果您不想再在部署中使用本地实例，则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 (p. 198)。

此部分中的说明向您演示如何配置本地实例，然后使用 CodeDeploy 进行注册并为它添加标签，以便在部署时使用。此部分还介绍如何使用 CodeDeploy 获取有关本地实例的信息，以及如何在您不再计划部署到某个本地实例时注销该实例。

主题

- 配置本地实例的先决条件 (p. 180)
- 向 CodeDeploy 注册本地实例 (p. 181)
- 在 CodeDeploy 中管理本地实例操作 (p. 198)

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**
>
> 如果您正在使用注册本地实例命令中生成的定期刷新临时凭证，并通过Amazon Security Token Service(Amazon STS），还有其他先决条件。有关信息，请参阅 IAM 会话 ARN 注册先决条件 (p. 192)。

设备要求

您要使用 CodeDeploy 准备、注册并标记为本地实例的设备必须在受支持的操作系统上运行。有关列表，请参阅CodeDeploy 代理支持的操作系统 (p. 141)。

如果您的操作系统不受支持，则 CodeDeploy 代理可以作为开源代理来适应您的需求。有关更多信息，请参阅 。CodeDeploy CodeDeploy存储 GitHub。

出站通信

本地实例必须能够连接到公有Amazon服务终端节点与 CodeDeploy 进行通信。

CodeDeploy 代理通过端口 443 使用 HTTPS 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帐户或网络帐户必须能够以`sudo`或者`root`（对于 Ubuntu Server）或作为管理员（对于 Windows Server）。

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 (以及在需要时注销本地实例的权限)。

除了开始使用 CodeDeploy (p. 26)，请确保发出调用的 IAM 身份还已经附加了以下其他策略：有关如何附加 IAM 策略的信息，请参阅管理 IAM 策略。

```
{
  "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": "*"
    }
  ]
}
```

向 CodeDeploy 注册本地实例

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 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。最适合只注册一个本地实例的情况。有关信息，请参阅 使用 register 命令（IAM 用户 ARN）注册本地实例。 (p. 181)。
  - 使用注册本地实例命令手动配置大多数注册选项。适合注册少量内部实例的情况。有关信息，请参阅 使用 register ter-on-premises-instance 命令 (IAM 用户 ARN) 注册本地实例 (p. 185)。
- 使用 IAM 角色 ARN 对请求进行身份验证.
  - 使用注册本地实例命令中生成的定期刷新临时凭证，并通过Amazon Security Token Service(Amazon STS）手动配置大多数注册选项。最适合注册大量本地实例的情况。有关信息，请参阅 使用 register ter-on-premises-instance 命令 (IAM 会话 ARN) 注册本地实例 (p. 192)。

主题

- 使用 register 命令（IAM 用户 ARN）注册本地实例。 (p. 181)
- 使用 register ter-on-premises-instance 命令 (IAM 用户 ARN) 注册本地实例 (p. 185)
- 使用 register ter-on-premises-instance 命令 (IAM 会话 ARN) 注册本地实例 (p. 192)

使用 register 命令（IAM 用户 ARN）注册本地实例。

本部分介绍如何以最少的工作量使用 CodeDeploy 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。您可以使用register命令仅当您使用 IAM 用户 ARN 对实例进行身份验证时。您无法使用register命令与 IAM 会话 ARN 进行身份验证。

当您使用register命令，您可以让 CodeDeploy 执行以下操作：

- 在中创建 IAM 用户Amazon Identity and Access Management（如果您未随该命令指定本地实例）。
- 将 IAM 用户的凭证保存到本地实例配置文件中。
- 向 CodeDeploy 注册本地实例。
- 如果您在命令中指定了标签，则向本地实例添加标签。

> **Note**
>
> 这些区域有：注册本地实例命令是register命令。您可以使用register-on-premises-instance命令主要靠自己通过 CodeDeploy 注册和标记该实例，请注册和标记该实例。这些区域有：register-on-premises-instance命令还提供了使用 IAM 会话 ARN 注册实例的选项，而不是使用 IAM 用户 ARN 注册实例。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 ter-on-premises-instance 命令 (IAM 用户 ARN) 注册本地实例 (p. 185) 和 使用 register ter-on-premises-instance 命令 (IAM 会话 ARN) 注册本地实例 (p. 192)。

主题

- 第 1 步：安装和配置Amazon CLI在本地实例上 (p. 182)
- 第 2 步：调用注册命令 (p. 183)
- 第 3 步：调用安装命令 (p. 184)
- 第 4 步：将应用程序修订部署到本地实例 (p. 184)
- 第 5 步：跟踪本地实例的部署 (p. 185)

第 1 步：安装和配置Amazon CLI在本地实例上

1. 在本地实例上安装 Amazon CLI。按照中的说明进行操作开始设置Amazon CLI中的Amazon Command Line Interface用户指南。

   Note

   用于处理本地实例的 CodeDeploy 命令可以在Amazon CLI版本 1.7.19 和更高版本中提供。如果您已安装了 Amazon CLI，请调用 aws --version 来检查其版本。

2. 在本地实例上配置 Amazon CLI。按照中的说明进行操作配置Amazon CLI在Amazon Command Line Interface用户指南。

   Important

   在配置Amazon CLI（例如，通过调用aws configure命令），请确保指定至少具有以下条件的 IAM 用户的私有密钥 ID 和秘密访问密钥。Amazon访问权限除了中指定的权限之外配置本地实例的先决条件 (p. 180)。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：

```
{
  "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:GetTags",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:-cn:s3:::aws-codedeploy-cn-north-1/*",
        "arn:aws:-cn:s3:::aws-codedeploy-cn-northwest-1/*"
```

```
        ]
      }
    ]
}
```

第 2 步：调用注册命令

在这一步中，我们假设您从本地实例本身注册该本地实例。您还可以从安装和配置了 Amazon CLI 的单独设备或实例上注册本地实例，如先前的步骤中所述。

使用Amazon CLI调用，register命令，在命令中指定：

- 向 CodeDeploy 唯一标识本地实例的名称 (使用`--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 限制 (p. 359)。

- （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，或者在用户部分，然后找到用户 ARN值处于摘要部分。如果未指定此选项，则 CodeDeploy 将代表您在Amazon帐户并将其关联到本地实例。

  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，如第 4 步：将配置文件添加到本地实例 (p. 188)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 (使用`--tags`选项），CodeDeploy 将使用这些方法来标识要部署到的 Amazon EC2 实例组。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。要在以后注册标签，请调用将标签添加到本地实例命令。
- 可选，Amazon区域中将在其中向 CodeDeploy 注册本地实例（使用`--region`选项）。此区域必须为区域和终端节点在Amazon一般参考（例如，`us-west-2`)。如果未指定此选项，则默认Amazon区域将使用与发出调用的 IAM 用户关联的区域。

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对 CodeDeploy 进行身份验证和交互。
2. 向 CodeDeploy 注册本地实例。
3. 如果指定，则在代 CodeDeploy 中关联使用`--tags`选项与注册的本地实例名称一起使用。
4. 如果创建了 IAM 用户，则还会在register命令被调用。

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。

第 3 步：调用安装命令

从本地实例，使用Amazon CLI调用，install命令，在命令中指定：

- 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，Amazon区域中将在其中向 CodeDeploy 注册本地实例（使用`--region`选项）。此操作必须为中列出的受支持的区域之一。区域和终端节点在Amazon一般参考（例如，`us-west-2`)。如果未指定此选项，则默认Amazon区域将使用与发出调用的 IAM 用户关联的区域。
- （可选）从中安装 CodeDeploy 代理的自定义位置（使用`--agent-installer`选项）。此选项对于安装 CodeDeploy 不官方支持的自定义 CodeDeploy 代理版本非常有用（例如基于CodeDeploy CodeDeploy存储 GitHub。该值必须是指向包含以下二者之一的 Amazon S3 存储桶的路径：
  - 代理 CodeDeploy 装脚本（对于基于 Linux 或 Unix 的操作系统，类似于CodeDeploy CodeDeploy存储 GitHub。
  - CodeDeploy 代理安装程序包 (.msi) 文件（对于基于 Windows 的操作系统）。

  如果未指定此选项，CodeDeploy 将尽大可能尝试从自己的位置安装与本地实例上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. 检查本地实例是否为 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 代理预期的位置（如果该位置尚没有该文件）。

   对于 Ubuntu Server 和 Red Hat Enterprise Linux (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`。

   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。

第 4 步：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。

将应用程序修订部署到本地实例的方法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 CodeDeploy 创建部署 (p. 254)。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（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)的第 2 步：创建示例应用程序修订 (p. 82)中所述的修订。

> **Important**
>
> 在创建以本地实例为目标的部署组的过程中，如果您重用现有 CodeDeploy 服务角色，则必须将包含`Tag:get*`添加到`Action`部分的服务角色的策略语句。有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

第 5 步：跟踪本地实例的部署

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。

跟踪对本地实例的部署的方法与跟踪对 Amazon EC2 实例的部署的方法类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 (p. 266)。

有关更多选项，请参阅在 CodeDeploy 中管理本地实例操作 (p. 198)。

使用 register ter-on-premises-instance 命令 (IAM 用户 ARN) 注册本地实例

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

- 第 1 步：为本地实例创建 IAM 用户 (p. 185)
- 第 2 步：向 IAM 用户分配权限 (p. 185)
- 第 3 步：获取 IAM 用户凭证 (p. 187)
- 第 4 步：将配置文件添加到本地实例 (p. 188)
- 第 5 步：安装和配置 Amazon CLI (p. 189)
- 第 6 步：设置 AWS_REGION 环境变量（仅针对 Ubuntu Server 和 RHEL）(p. 190)
- 步骤 7：安装 CodeDeploy 代理 (p. 190)
- 步骤 8：向 CodeDeploy 注册本地实例 (p. 190)
- 步骤 9：为本地实例添加标签 (p. 191)
- 步骤 10：将应用程序修订部署到本地实例 (p. 192)
- 步骤 11：跟踪本地实例的部署 (p. 192)

第 1 步：为本地实例创建 IAM 用户

创建一个 IAM 用户，本地实例将用于身份验证和与 CodeDeploy 进行交互。

> **Important**
>
> 您必须为每个参与的本地实例创建单独的 IAM 用户。如果您尝试对多个本地实例重用单个 IAM 用户，则您可能无法成功使用 CodeDeploy 注册或标记这些本地实例。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。

我们建议为 IAM 用户分配一个标识其用途的名称（如代码部署用户 Onprem）。

您可以使用Amazon CLI或 IAM 控制台创建 IAM 用户。想要了解有关信息，请参阅在您的中创建 IAM 用户 Amazon账户。

> **Important**
>
> 无论您使用Amazon CLI或 IAM 控制台创建新的 IAM 用户，请记录为该用户提供的用户 ARN。稍后在第 4 步：将配置文件添加到本地实例 (p. 188) 和步骤 8：向 CodeDeploy 注册本地实例 (p. 190)中您将需要此信息。

第 2 步：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

如果您仅从 GitHub 存储库部署应用程序修订，则跳过此步骤，直接转到第 3 步：获取 IAM 用户凭证 (p. 187)。（您将仍需要有关在中创建的 IAM 用户的信息。第 1 步：为本地实例创建 IAM 用户 (p. 185)。 后面的步骤中将用到此信息。）

分配权限 (CLI)

1. 在您用于调用的 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，请确保还提 Amazon S3 了针对包含Amazon CodeDeploy代理。否则，只要在关联的本地实例上安装或更新 CodeDeploy 代理，就可能出现错误。
例如：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
                "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），以及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. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
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，这些表达式允许Amazon CodeDeploy代表 IAM 用户账户从策略中指定的任意 Amazon S3 存储桶向本地实例部署应用程序修订：

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

6. 选择 Create Policy。
7. 在导航窗格中，选择 Users。
8. 在用户列表中，浏览并选择在中创建的 IAM 用户的名称。第 1 步：为本地实例创建 IAM 用户 (p. 185)。
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)。

第 3 步：获取 IAM 用户凭证

获取 IAM 用户的私有密钥 ID 和私有访问密钥。您在第 4 步：将配置文件添加到本地实例 (p. 188) 中需要使用它们。您可以使用Amazon CLI或 IAM 控制台来获取私有密钥 ID 和秘密访问密钥。

> **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，直接转到第 4 步：将配置文件添加到本地实例 (p. 188)。

获取凭证 (CLI)

1. 调用列表访问密钥命令，指定 IAM 用户的名称（使用`--user-name`选项）并仅查询访问密钥 ID（使用`--query`和`--output`选项）。例如：

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query
 "AccessKeyMetadata[*].AccessKeyId" --output text
```

2. 如果输出中未显示密钥或者输出中只显示一个密钥的相关信息，请调用创建访问密钥命令，指定 IAM 用户的名称（使用`--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。在第 4 步：将配置文件添加到本地实例 (p. 188) 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，则需要按照第 3 步：获取 IAM 用户凭证 (p. 187) 中的步骤生成新的密钥。

3. 如果已经列出了两个访问密钥，则必须通过调用删除访问密钥命令，指定 IAM 用户的名称（使用`--user-name`选项），以及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中所述使用此访问密钥，第 4 步：将配置文件添加到本地实例 (p. 188)，则需要按照第 4 步：将配置文件添加到本地实例 (p. 188)来指定与此 IAM 用户关联的不同访问密钥 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. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   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，浏览并选择在中创建的 IAM 用户的名称。第 1 步：为本地实例创建 IAM 用户 (p. 185)。
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**
   >
   > 如果选择Delete，并且本地实例已经按中所述使用此访问密钥，并且本地实例已经按中所述使用此访问密钥，第 4 步：将配置文件添加到本地实例 (p. 188)，则需要按照第 4 步：将配置文件添加到本地实例 (p. 188)来指定与此 IAM 用户关联的不同访问密钥 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，则需要按照第 3 步：获取 IAM 用户凭证 (p. 187) 中的步骤生成新的密钥。
4. 选择 Close 返回 Users > ***IAM User Name*** 页面。

第 4 步：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 用户凭证和目标Amazon区域以用于 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 和秘密访问密钥，以及目标Amazon区域。该文件必须遵循特定格式。

1. 创建一个名为的文件`codedeploy.onpremises.yml`（对于 Ubuntu 服务器或 RHEL 本地实例）或`conf.onpremises.yml`（对于 Windows 服务器本地实例）在本地实例上的以下位置：

- 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：

- *#### ID*是您在中记录的对应 IAM 用户的私有密钥 ID。第 1 步：为本地实例创建 IAM 用户 (p. 185)或者第 3 步：获取 IAM 用户凭证 (p. 187)。
- *######*是您在中记录的对应 IAM 用户的私有访问密钥。第 1 步：为本地实例创建 IAM 用户 (p. 185)或者第 3 步：获取 IAM 用户凭证 (p. 187)。
- *iam-user-arn* 是您之前在 第 1 步：为本地实例创建 IAM 用户 (p. 185) 中记录的 IAM 用户的 ARN。
- *####*是 CodeDeploy 支持的区域的标识符，您的 CodeDeploy 应用程序、部署组和应用程序修订位于该区域（例如，`us-west-2`)。有关区域列表，请参阅区域和终端节点中的Amazon一般参考。

> **Important**
>
> 如果您选择了Delete在其中一个访问密钥旁边第 3 步：获取 IAM 用户凭证 (p. 187)，并且您的本地实例已经在使用关联访问密钥 ID 和秘密访问密钥，则您需要按照第 4 步：将配置文件添加到本地实例 (p. 188)来指定与此 IAM 用户关联的不同访问密钥 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。

第 5 步：安装和配置 Amazon CLI

在本地实例上安装和配置 Amazon CLI（Amazon CLI将用于步骤 7：安装 CodeDeploy 代理 (p. 190)在本地实例上下载和安装 CodeDeploy 代理。）

1. 安装Amazon CLI，请按照中的说明操作开始设置Amazon CLI中的Amazon Command Line Interface用户指南。

   > **Note**
   >
   > 1.7.19 版中提供了用于处理本地实例的 CodeDeploy 命令。Amazon CLI。如果您已经安装了 Amazon CLI 版本，则可以调用 aws --version 来检查其版本。

2. 配置Amazon CLI，请按照中的说明操作配置Amazon CLI中的Amazon Command Line Interface用户指南。

   > **Important**
   >
   > 在配置Amazon CLI（例如，通过调用aws configure命令），请确保指定至少具有以下条件的 IAM 用户的私有密钥 ID 和秘密访问密钥。Amazon访问权限以及配置本地实例的先决条件 (p. 180)。这使您能够在本地实例上下载和安装 CodeDeploy 代理：

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

```
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:-cn:s3:::aws-codedeploy-cn-north-1/*",
        "arn:aws:-cn:s3:::aws-codedeploy-cn-northwest-1/*"
      ]
    }
  ]
}
```

这些访问权限可以分配给您在第 1 步：为本地实例创建 IAM 用户 (p. 185)或其他 IAM 用户。要将这些权限分配给 IAM 用户，请按照第 1 步：为本地实例创建 IAM 用户 (p. 185)，并使用这些访问权限而不是该步骤中的权限。

第 6 步：设置 AWS_REGION 环境变量（仅针对 Ubuntu Server 和 RHEL）

如果您未在本地实例上运行 Ubuntu 服务器或 RHEL，请跳过此步骤，直接转到步骤 7：安装 CodeDeploy 代理 (p. 190)。

在 Ubuntu 服务器或 RHEL 本地实例上安装 CodeDeploy 代理，并使实例能够在发布新版本时及时更新 CodeDeploy 代理。要实现此目的，需要将`AWS_REGION`环境变量设置为 CodeDeploy 支持的某个区域的标识符。我们建议您将区域的值设置为 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`)。有关区域列表，请参阅区域和终端节点中的Amazon一般参考。

要设置环境变量，请从终端调用以下命令：

```
export AWS_REGION=supported-region
```

其中 *`supported-region`* 是区域标识符（例如，`us-west-2`）。

步骤 7：安装 CodeDeploy 代理

在本地实例上安装代理：

- 对于 Ubuntu Server 本地实例，请按照中的说明操作安装 Ubuntu 服务器的 CodeDeploy 代理 (p. 154)，然后返回此页。
- 对于 RHEL 本地实例，请按照中的说明操作安装适用于亚马逊 Linux 或 RHEL 的 CodeDeploy 代理 (p. 153)，然后返回此页。
- 对于 Windows Server 本地实例，请按照中的说明操作安装适用于 Windows 服务器的 CodeDeploy 代理 (p. 156)，然后返回此页。

步骤 8：向 CodeDeploy 注册本地实例

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。您还可以从安装和配置了 Amazon CLI 的其他设备或实例上注册本地实例，如第 5 步：安装和配置 Amazon CLI (p. 189) 中所述。

使用Amazon CLI向 CodeDeploy 注册本地实例，以便可以在部署中使用该实例。

1. 在您可以使用Amazon CLI，您将需要在中创建的 IAM 用户的用户 ARN。第 1 步：为本地实例创建 IAM 用户 (p. 185)。如果您还没有用户 ARN，请调用get-user命令，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. 调用注册本地实例命令，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 限制 (p. 359)。
   - 您在中创建的 IAM 用户的用户 ARN第 1 步：为本地实例创建 IAM 用户 (p. 185)（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：为本地实例添加标签

您可以使用Amazon CLI或 CodeDeploy 控制台来标记本地实例。（CodeDeploy 使用本地实例标签来标识部署目标。）

标记本地实例 (CLI)

- 调用将标签添加到本地实例命令，在命令中指定：

  - 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)。您必须指定名称和值。CodeDeploy 不允许本地实例标签只具有值。

    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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
2. 从 CodeDeploy 菜单中，依次选择本地实例。
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。

将应用程序修订部署到本地实例的方法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 CodeDeploy 创建部署 (p. 254)。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（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)的第 2 步：创建示例应用程序修订 (p. 82)中所述的修订。

> **Important**
>
> 在创建以本地实例为目标的部署组的过程中，如果您重用 CodeDeploy 服务角色，则必须将包含`Tag:get*`添加到`Action`部分的服务角色的策略语句。有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

步骤 11：跟踪本地实例的部署

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。

跟踪对本地实例的部署的方法与跟踪对 Amazon EC2 实例的部署的方法类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 (p. 266)。

使用 register ter-on-premises-instance 命令 (IAM 会话 ARN) 注册本地实例

为最大程度控制您的本地实例的身份验证和注册，您可以使用注册本地实例命令中生成的定期刷新临时凭证，并通过Amazon Security Token Service(Amazon STS)。实例的一个静态 IAM 角色将担任这些刷新的角色。Amazon STS凭据来执行 CodeDeploy 部署操作。

当您需要注册大量实例时，此方法非常有用。您可以使用 CodeDeploy 来借助自动执行注册过程。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 IAM 会话凭证从服务分发给实例以便用于 CodeDeploy。

> **Note**
>
> 或者，您也可以使用分发给所有本地实例的一个共享 IAM 用户调用 Amazon STS AssumeRole API，以检索本地实例的会话凭证。此方法安全性较低，建议不要用于生产或关键任务型环境。

参考以下主题中的信息，使用 Amazon STS 生成的临时安全凭证来配置本地实例。

主题

- IAM 会话 ARN 注册先决条件 (p. 192)
- 第 1 步：创建本地实例将代入的 IAM 角色 (p. 193)
- 第 2 步：为单个实例生成临时证书，使用Amazon STS (p. 194)
- 第 3 步：将配置文件添加到本地实例 (p. 194)
- 第 4 步：为 CodeDeploy 部署部署的本地实例 (p. 195)
- 第 5 步：向 CodeDeploy 注册本地实例 (p. 196)
- 第 6 步：为本地实例添加标签 (p. 197)
- 步骤 7：将应用程序修订部署到本地实例 (p. 197)
- 步骤 8：跟踪本地实例的部署 (p. 197)

IAM 会话 ARN 注册先决条件

除了在 配置本地实例的先决条件 (p. 180) 中列出的前提条件之外，还必须满足以下要求：

IAM 权限

您用来注册本地实例的 IAM 身份必须有权执行 CodeDeploy 操作。确保AWSC 部署完全访问权限托管策略附加到该 IAM 身份。想要了解有关信息，请参阅Amazon托管策略中的IAM 用户指南。

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 CRAN 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Amazon而无需更改 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 代理可访问的文件中。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。

有关创建和使用的信息Amazon STS凭据，请参阅Amazon Security Token ServiceAPI 参考和使用临时安全证书请求访问Amazonresources。

第 1 步：创建本地实例将代入的 IAM 角色

您可以使用Amazon CLI或 IAM 控制台创建一个 IAM 角色，供您的本地实例在进行身份验证和与 CodeDeploy 进行交互时使用。

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来访问安装 CodeDeploy 代理所需的文件：

```
{
    "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，请确保授予对包含 CodeDeploy 代理的 Amazon S3 存储桶的访问权限。否则，只要在本地实例上安装或更新 CodeDeploy 代理，就可能出现错误。有关如何控制对 Amazon S3 存储桶的访问权限的信息，请参阅管理对 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 AssumeRole API 生成每个实例的短期凭证时，将需要该角色 ARN。

有关创建 IAM 角色的更多信息，请参阅创建向Amazon服务在IAM 用户指南。

有关为现有角色分配权限的信息，请参阅put-role-policy在Amazon CLI命令参考。

第 2 步：为单个实例生成临时证书，使用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。

有关授予`sts:AssumeRole`访问 IAM 身份的权限，请参阅创建向Amazon服务和AssumeRole。

有两种生成临时凭证的方法：

- 使用确保角色命令，并使用Amazon CLI。例如：

```
aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --role-session-
name session-name
```

其中：

- *12345ACCOUNT* 是您的组织的 12 位数账号.
- *role-arn* 是要担任的角色 (在第 1 步：创建本地实例将代入的 IAM 角色 (p. 193)中生成) 的 ARN。
- *session-name* 是您为正在创建的角色会话提供的名称.

  Note

  如果您使用 CRAN 脚本定期轮询身份和身份验证系统，并将最新会话凭证复制到实例（用于刷新中所述的临时凭证的方法 1）IAM 会话 ARN 注册先决条件 (p. 192)），则可以使用任何受支持的Amazon要调用的 SDKAssumeRole。

- 使用提供的工具Amazon。

  aws-codedeploy-session-helper 工具生成 Amazon STS 凭证并将它们写入您放在实例上的文件。此工具最适用于 IAM 会话 ARN 注册先决条件 (p. 192) 中所述的用于刷新临时凭证的方法 2。在此方法中，aws-codedeploy-session-helper 工具置于每个实例之上，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。在下一个步骤中，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 注册先决条件 (p. 192) 中的“用于刷新临时凭证的系统”。

第 3 步：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 凭证和目标Amazon区域以用于 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 和秘密访问密钥，以及目标Amazon区域。

添加配置文件

1. 创建一个名为的文件`codedeploy.onpremises.yml`（对于 Ubuntu 服务器或 RHEL 本地实例）或`conf.onpremises.yml`（对于 Windows 服务器本地实例）在本地实例上的以下位置：
   - 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：

```
---
iam_session_arn: iam-session-arn
aws_credentials_file: credentials-file
region: supported-region
```

   其中：

   - *`iam-session-arn`* 是您在第 2 步：为单个实例生成临时证书，使用Amazon STS (p. 194)中记录的 IAM 会话 ARN。
   - *`credentials-file`* 是在第 2 步：为单个实例生成临时证书，使用Amazon STS (p. 194)中记录的临时会话 ARN 凭证文件的位置。
   - *`####`*是 CodeDeploy 支持的区域之一，如区域和终端节点在Amazon一般参考。

第 4 步：为 CodeDeploy 部署部署的本地实例

安装和配置 Amazon CLI

在本地实例上安装和配置 Amazon CLI （Amazon CLI将用于在本地实例上下载和安装 CodeDeploy 代理。）

1. 安装Amazon CLI，请按照中的说明操作开始设置Amazon CLI中的Amazon Command Line Interface用户指南。

   > **Note**
   >
   > 1.7.19 版中提供了用于处理本地实例的 CodeDeploy 命令。Amazon CLI。如果您已经安装了 Amazon CLI 版本，则可以调用 aws --version 来检查其版本。

2. 配置Amazon CLI，请按照中的说明操作配置Amazon CLI中的Amazon Command Line Interface用户指南。

   > **Important**
   >
   > 在配置Amazon CLI（例如，通过调用aws configure命令），请确保指定至少具有中所述权限的 IAM 用户的私有密钥 ID 和秘密访问密钥。IAM 会话 ARN 注册先决条件 (p. 192)。

设置 AWS_REGION 环境变量 (仅针对 Ubuntu Server 和 RHEL)

如果您的本地实例上没有运行 Ubuntu 服务器或 RHEL，则跳过此步骤，直接转到 “安装 CodeDeploy 代理”。

在 Ubuntu 服务器或 RHEL 本地实例上安装 CodeDeploy 代理，并使实例能够在发布新版本时及时更新 CodeDeploy 代理。要实现此目的，需要将`AWS_REGION`环境变量设置为 CodeDeploy 支持的某个区域的标识符。我们建议您将区域的值设置为 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`)。有关区域列表，请参阅区域和终端节点中的Amazon一般参考。

要设置环境变量，请从终端调用以下命令：

```
export AWS_REGION=supported-region
```

其中 *supported-region* 是区域标识符（例如，us-west-2）。

安装 CodeDeploy 代理

- 对于 Ubuntu Server 本地实例，请按照中的说明操作安装 Ubuntu 服务器的 CodeDeploy 代理 (p. 154)，然后返回此页。
- 对于 RHEL 本地实例，请按照中的说明操作安装适用于亚马逊 Linux 或 RHEL 的 CodeDeploy 代理 (p. 153)，然后返回此页。
- 对于 Windows 服务器本地实例，请按照安装适用于 Windows 服务器的 CodeDeploy 代理 (p. 156)，然后返回此页。

第 5 步：向 CodeDeploy 注册本地实例

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。您还可以从安装和配置了 Amazon CLI 的其他设备或实例上注册本地实例。

使用Amazon CLI向 CodeDeploy 注册本地实例，以便可以在部署中使用该实例。

您需要您在第 3 步：将配置文件添加到本地实例 (p. 194)中创建的临时会话凭证的 ARN 才能使用 Amazon CLI。例如，对于您指定为 AssetTag12010298EX 的实例：

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

调用注册本地实例命令，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 限制 (p. 359)。

- 您在第 1 步：创建本地实例将代入的 IAM 角色 (p. 193)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，如 AssetTag12010298EX.
- *account-id* 是您的组织的 12 位数账户 ID，如 111222333444.
- *role-to-assume* 是您为实例创建的 IAM 角色的名称，如 CodeDeployInstanceRole.
- *session-name* 是您在 第 2 步：为单个实例生成临时证书，使用Amazon STS (p. 194) 中指定的会话角色的名称。

第 6 步：为本地实例添加标签

您可以使用Amazon CLI或 CodeDeploy 控制台来标记本地实例。（CodeDeploy 使用本地实例标签来标识部署目标。）

标记本地实例 (CLI)

- 调用将标签添加到本地实例命令，在命令中指定：
  - 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)。您必须指定名称和值。CodeDeploy 不允许本地实例标签只具有值。

    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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 Remove (删除)。
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。

将应用程序修订部署到本地实例的方法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 CodeDeploy 创建部署 (p. 254)。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（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)的第 2 步：创建示例应用程序修订 (p. 82)中所述的修订。

Important

在创建以本地实例为目标的部署组的过程中，如果您重用 CodeDeploy 服务角色，则必须将包含`Tag:get*`添加到`Action`部分的服务角色的策略语句。有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

步骤 8：跟踪本地实例的部署

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。

跟踪对本地实例的部署的方法与跟踪对 Amazon EC2 实例的部署的方法类似。有关说明，请参阅查看 CodeDeploy 部署详细信息 (p. 266)。

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注销这些本地实例。

主题

- 获取有关单个本地实例的信息 (p. 198)
- 获取有关多个本地实例的信息 (p. 198)
- 手动删除本地实例中的本地实例标签 (p. 199)
- 自动卸载 CodeDeploy 代理并删除本地实例中的配置文件 (p. 200)
- 自动注销本地实例 (p. 200)
- 手动注销本地实例 (p. 201)

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 (p. 266)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)

- 调用获取本地实例命令，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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 (p. 266)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. 有关本地实例名称的列表，请调用本地实例列表命令，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. 如需更多详细信息，请调用批量获取本地实例命令，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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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。

手动删除本地实例中的本地实例标签

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 (p. 200)。

要手动注销本地实例，请参阅手动注销本地实例 (p. 201)。

要自动卸载 CodeDeploy 代理并删除本地实例中的配置文件，请参阅自动卸载 CodeDeploy 代理并删除本地实例中的配置文件 (p. 200)。

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 (p. 149)。

要手动删除相关联的 IAM 用户，请参阅从您的 IAM 用户中删除Amazon账户。

从本地实例删除本地实例标签 (CLI)

- 调用remove-tags-from-Deploy，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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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。

自动卸载 CodeDeploy 代理并删除本地实例中的配置文件

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 (p. 200)。
要手动注销本地实例，请参阅手动注销本地实例 (p. 201)。
要手动取消任何已关联本地实例标签的关联，请参阅手动删除本地实例中的本地实例标签 (p. 199)。
要从本地实例手动卸载 CodeDeploy 代理，请参阅管理 CodeDeploy 代理操作 (p. 149)。
要手动删除相关联的 IAM 用户，请参阅从您的 IAM 用户中删除Amazon账户。

从本地实例，使用Amazon CLI调用，卸载命令。

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，请参阅手动注销本地实例 (p. 201)。

要手动取消任何已关联本地实例标签的关联，请参阅手动删除本地实例中的本地实例标签 (p. 199)。
要自动卸载 CodeDeploy 代理并删除本地实例中的配置文件，请参阅自动卸载 CodeDeploy 代理并删除本地实例中的配置文件 (p. 200)。
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 (p. 149)。

使用Amazon CLI调用，取消注册命令，在命令中指定：

- 唯一标识的本地实例 CodeDeploy 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。
- 或者，Amazon区域中向 CodeDeploy 注册本地实例（使用`--region`选项）。此操作必须为中列出的受支持的区域之一。区域和终端节点中的Amazon一般参考（例如，`us-west-2`)。如果未指定此选项，则默认Amazon区域将使用与发出调用的 IAM 用户关联的区域。

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 注销本地实例。
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。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，请参阅自动卸载 CodeDeploy 代理并删除本地实例中的配置文件 (p. 200)。

要仅手动卸载 CodeDeploy，请参阅管理 CodeDeploy 代理操作 (p. 149)。

要手动删除相关联的 IAM 用户，请参阅从您的 IAM 用户中删除Amazon账户。

要仅手动删除关联的本地实例标签，请参阅手动删除本地实例中的本地实例标签 (p. 199)。

- 调用取消注册本地实例命令，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

您可以使用 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） (p. 202)
- 查看实例详细信息 (CLI) (p. 202)

查看实例详细信息（控制台）

查看实例详细信息：

1. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在中使用的相同账户或 IAM 用户信息登录开始使用 CodeDeploy (p. 26)。
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。在导航栏上的区域选择器中，选择区域和终端节点中的Amazon一般参考。仅在这些地区支持 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 (p. 372)中找到问题排查提示。
6. 如果要查看有关 Amazon EC2 实例的更多信息，请在实例 IDcolumn.

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-Deploy命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用list-ploy命令。
- 唯一实例 ID。要获取实例 ID，请调用列表部署实例命令。

要查看部署中使用的实例的 ID 列表，请调用列表部署实例命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用list-ploy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 将导致整体部署失败。

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。

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	99%
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

在 在 CodeDeploy 中使用部署配置 (p. 206) 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运行正常的主机值。在使用以下操作时，您可以将这些值定义为整数或百分比：

- 作为`minimum-healthy-hosts`当您使用创建部署配置命令，Amazon CLI。
- 作为`Value`中的MinimumHealthyHosts数据 CodeDeploy。
- 在 Amazon CloudFormation 模板中使用 AWS::CodeDeploy::DeploymentConfig 时作为 `MinimumHealthyHosts`。

主题

- 运行状况 (p. 203)
- 最小运行正常的实例数 (p. 204)

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。

最小运行正常的实例数

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 会将百分比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。对于每次成功部署，CodeDeploy 会将实例的运行状况更改为 “正常”，并将它添加到部署组的运行正常的实例中。CodeDeploy 随后将比较当前的运行正常的实例数与指定的最小运行正常的实例数。

- 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，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。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 一次将部署到一个实例。
- 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 将同时部署到每个实例。

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，CodeDeploy 会立即让整个部署失败，因为运行正常的实例数等于最小运行正常的实例数。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，CodeDeploy 会立即让整个部署失败。此规则的例外情况是，如果最后一个实例是第二个失败的实例，则部署仍将成功。

最小正常的实例数：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。

在 CodeDeploy 中使用部署配置

部署配置是 CodeDeploy 期间使用的一组规则以及成功条件和失败条件。根据您是部署到 EC2/本地计算平台上的计算平台/本地计算平台上的计算平台上的计算平台和条件不同AmazonLambda 计算平台，或者亚马逊云服务器计算平台。

EC2/本地计算平台上的部署配置

通过使用最少正常运行的主机数值，部署配置会通过使用最少正常运行的主机数值，来指定在部署过程中的任意时候保持可用的实例数或实例百分比。

您可以使用由 Amazon 提供的三种预定义的部署配置之一，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 (p. 210)。如果您未指定部署配置，CodeDeployDefault.OneAtATime 部署配置将使用 CodeDeployDefault.OneAtATime 部署配置。

有关 CodeDeploy 如何在部署期间监控和评估实例运行状况的信息，请参阅。Instance Health (p. 203)。要查看已注册到 Amazon 账户的部署配置列表，请参阅View Deployment Configuration Details (p. 210)。

EC2/本地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。

部署配置	描述
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。在包含 9 个实例的示例中，CodeDeployDefault.AllAtOnce 将尝试一次性部署到所有 9 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： • 部署到替换环境：遵循的部署规则与适用于就地部署的 CodeDeployDefault.AllAtOnce 相同。 • 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。
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

部署配置	描述
	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。 蓝/绿部署： • 部署到替换环境：遵循的部署规则与适用于就地部署的 CodeDeployDefault.HalfAtATime 相同。 • 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。
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，CodeDeployDefault.OneAtATime 配置一次只允许使一个实例脱机。 • 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： • 部署到替换环境：遵循的部署规则与适用于就地部署的 CodeDeployDefault.OneAtATime 相同。 • 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

当您部署到 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 任务集。

在部署过程中，有三种可转移流量的方法：

- Canary：流量在两次增量中转移。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 Amazon 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：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。
- 一次性：所有流量均从原始 Amazon ECS 任务集一次性地转移到更新后的 Amazon ECS 任务集。

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 (p. 210)。

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。

部署配置	描述
CodeDeployDefault.ECSLinear10PercentEvery1Minutes	每分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.ECSLinear10PercentEvery3Minutes	每隔 3 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.ECSCanary10Percent5Minutes	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.ECSCanary10Percent15Minutes	在第一次增量中转移 10% 的流量。其余 90% 部署在 15 分钟后进行转移。
CodeDeployDefault.ECSAllAtOnce	所有流量一次性转移到更新后的 Amazon ECS 容器。

部署配置Amazon CloudFormation蓝/绿部署（Amazon ECS）

当您部署到 Amazon ECS 计算平台时，通过Amazon CloudFormation蓝/绿部署，部署配置将指定如何将流量转移到更新后的 Amazon ECS 容器。

Amazon ECS 蓝/绿部署期间可通过以下三种方式进行转移：Amazon CloudFormation：

- Canary：流量在两次增量中转移。您可以选择一个预定义的金丝雀部署选项，该选项指定在第一次增量中转移到更新后的 Amazon 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：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。
- 一次性：所有流量均从原始 Amazon ECS 任务集一次性地转移到更新后的 Amazon ECS 任务集。

对于此部署配置，您无法创建自己的自定义 Canary 或线性部署配置。有关使用的分步指导，Amazon CloudFormation以管理您的 Amazon ECS 蓝/绿部署，请参阅。通过 CodeDeploy 实现 ECS 蓝/绿部署Amazon CloudFormation中的Amazon CloudFormation用户指南。

> **Note**
>
> 使用管理 Amazon ECS 蓝/绿部署Amazon CloudFormation在墨西哥 XP、CPT、KIX 或 BJS/ZHY 区域中不可用。

上的部署配置Amazon Lambda计算平台

当您部署到AmazonLambda 计算平台，部署配置将指定流量转移到您应用程序中的新 Lambda 函数版本的方式。

在部署过程中，有三种可转移流量的方法：

- Canary：流量在两次增量中转移。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 函数版本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：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。
- 一次性：所有流量均从原始 Lambda 函数一次性地转移到更新后的 Lambda 函数版本。

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 (p. 210)。

预定义的部署配置Amazon Lambda计算平台

下表列出了可用于 Amazon Lambda 部署的预定义配置。

部署配置	描述
CodeDeployDefault.LambdaCanary10Percent5Minutes	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.LambdaCanary10Percent10Minutes	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。
CodeDeployDefault.LambdaCanary10Percent15Minutes	在第一次增量中转移 10% 的流量。其余 90% 部署在 15 分钟后进行转移。
CodeDeployDefault.LambdaCanary10Percent30Minutes	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.LambdaLinear10PercentEvery1Minute	每分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.LambdaLinear10PercentEvery2Minutes	每隔 2 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.LambdaLinear10PercentEvery3Minutes	每隔 3 分钟转移 10% 的流量，直到所有流量转移完毕。
CodeDeployDefault.LambdaLinear10PercentEvery10Minutes	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 (p. 210)
- View Deployment Configuration Details (p. 210)
- Delete a Deployment Configuration (p. 211)

使用 CodeDeploy 创建部署配置

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用于 CodeDeploy 参考的模板 (p. 353)。

使用Amazon CLI创建部署配置，请调用创建部署配置命令，并指定：

- 唯一标识部署配置的名称。在使用 CodeDeploy 创建的所有部署配置中，该名称必须是唯一的，并与 Amazonaccount.
- 部署期间应随时可用的运行正常实例的最小数量或百分比。有关更多信息，请参阅Instance Health (p. 203)。

以下示例创建一个名为 TThreeQuartersHealthy 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
```

以下示例创建一个Amazon名为 Canary25Percen45Manute 的 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
```

以下示例创建一个名为 Canary25Percen45Manute 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
```

查看使用 CodeDeploy 的部署配置详细信息

可以使用 CodeDeploy 控制台、Amazon CLI或 CodeDeploy API 查看有关与Amazonaccount. 有关预定义的 CodeDeploy 部署配置的说明，请参阅EC2/本地计算平台的预定义部署配置 (p. 206)。

主题

- 查看部署配置详细信息（控制台）(p. 210)
- 查看部署配置 (CLI) (p. 211)

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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，并在https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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。在导航栏中的区域选择器中，选择区域和终端节点中的Amazon一般参考。CodeDeploy 仅在这些地区受支持。

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-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，请调用列表部署命令。

使用 CodeDeploy 删除部署配置

您可以将Amazon CLI或 CodeDeploy API 删除与您的Amazonaccount. 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。此操作无法撤消。

使用Amazon CLI删除部署配置，请调用删除部署配置命令，并指定部署配置名称。要查看部署配置名称的列表，请调用列表部署配置命令。

以下示例删除名为 ThreeQuartersHealthy 的部署配置。

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

使用应 CodeDeploy

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。一个应用程序只是 CodeDeploy 使用的一个名称或容器，确保在部署过程中引用了正确的修订、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 CodeDeploy 的实例 (p. 162)，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 (p. 163)，然后返回此页。
EC2/本地、AmazonLambda, 和Amazon ECS	我尚未创建应用程序。	请参阅 使用 CodeDeploy 创建应用程序 (p. 212)
EC2/本地、AmazonLambda, 和Amazon ECS	我已创建应用程序，但尚未创建部署组。	请参阅 使用 CodeDeploy 创建部署组 (p. 225)。
EC2/本地、AmazonLambda, 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 使用 CodeDeploy 的应用程序修订 (p. 239)。
EC2/本地、AmazonLambda, 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 CodeDeploy 创建部署 (p. 254)。

主题

- 使用 CodeDeploy 创建应用程序 (p. 212)
- 使用 CodeDeploy 查看应用程序详细信息 (p. 219)
- 创建通知规则 (p. 220)
- 重命名 CodeDeploy 应用程序 (p. 222)
- 删除 CodeDeploy 中的应用程序 (p. 222)

使用 CodeDeploy 创建应用程序

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的应用程序只是 CodeDeploy 使用的名称或容器，确保在部署期间引用正确的修订、部署配置和部署组。您可以使用 CodeDeploy 控制台、Amazon CLI、CodeDeploy API 或Amazon CloudFormation模板创建应用程序。

您的代码（即应用程序修订）将通过一个名为 “部署” 的过程安装到实例。CodeDeploy 支持两种类型的部署：

- 就地部署：部署组中每个实例上的应用程序将停止，安装最新的应用程序修订版，然后启动并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/本地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 (p. 5)。
- 蓝/绿部署：部署的行为取决于使用的计算平台：
  - EC2/本地计算平台上的蓝/绿部署计算平台上的蓝/绿：部署组中的实例 (原始环境) 将被不同的实例集 (替换环境) 所代替，步骤如下：

- 针对替换环境预置实例。
- 替换实例上安装了最新的应用程序修订。
- 应用程序测试和系统验证等活动将出现可选的等待时间。
- 替换环境中的实例向 EElastic Load Balancing cer 注册，导致流量路由到这些实例。原始环境中的实例将取消注册，可终止或继续运行以供其他使用。

  Note

  如果您使用 EC2/本地计算平台，请注意蓝/绿色部署仅适用于 Amazon EC2 实例。

- 蓝/绿部分的蓝/绿AmazonLambda 计算平台：流量将从当前无服务器环境转移到包含更新后的 Lambda 函数版本的环境。您可以指定执行验证测试的 Lambda 函数并选择流量转移的方式。全部 AmazonLambda 计算平台部署是蓝/绿部署。因此，您无需指定部署类型。
- Amazon ECS 计算平台上的蓝/绿：流量将从 Amazon ECS 服务中使用原始版本应用程序的任务集转移到同一服务中的替换任务集。您可以通过部署配置将流量转移设置为线性或 Canary。指定负载平衡器侦听器的协议和端口用于重新路由生产流量。在部署期间，在运行验证测试时，可以使用测试侦听器为替换任务集提供流量。
- 通过 进行蓝/绿部署Amazon CloudFormation：流量将从当前资源转移到更新后的资源，作为Amazon CloudFormation堆栈更新。目前，仅支持 ECS 蓝/绿部署。

有关蓝/绿部署的更多信息，请参阅蓝/绿部署概述 (p. 6)。

当您使用 CodeDeploy 控制台创建应用程序时，可同时配置其首个部署组。当您使用 Amazon CLI 创建应用程序时，可在单独的步骤中创建其首个部署组。

要查看已向您的 Amazon 账户注册的应用程序的列表，请参阅使用 CodeDeploy 查看应用程序详细信息 (p. 219)。有关使用 Amazon CloudFormation 模板创建应用程序的信息，请参阅 Amazon CloudFormation用于 CodeDeploy 参考的模板 (p. 353)。

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	是	是
本地	是	否
Serverless (无服务器)AmazonLambda 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）(p. 213)
- 为蓝/绿部署创建应用程序（控制台）(p. 215)
- 创建 Amazon ECS 服务的应用程序 (控制台) (p. 217)
- 创建一个应用程序Amazon Lambda函数部署（控制台）(p. 218)
- 创建应用程序 (CLI) (p. 219)

为就地部署创建应用程序（控制台）

使用 CodeDeploy 控制台为就地部署创建应用程序：

Warning

以下情况下请勿按照这些步骤操作：

- 您尚未为 CodeDeploy 部署中使用的实例做好准备。要设置您的实例，请按照使用 CodeDeploy 的实例 (p. 162)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 (p. 210)中的说明操作，然后执行本主题中的步骤。
- 您没有信任 CodeDeploy 且具有所需的最低信任和权限的服务角色。要创建和配置具有所需权限的服务角色，请按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明操作，然后返回到本主题中的相应步骤。
- 您需要为就地部署选择 Elastic Load Balancer、应用程序 Load Balancer 或 Network Load Balancer，但尚未创建它。

使用 CodeDeploy 控制台为就地部署创建应用程序：

1. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
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 Create application。
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 EC2/On-premises (EC2/本地)。
6. 选择 Create application。
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；标签和/或 Amazon EC2 Auto Scaling 组名称；部署配置），请在此页上指定这些设置。虽然这个新的部署组与现有部署组同名，仍认为 CodeDeploy 是两个部署组，因为与它们关联的应用程序不同。
9. In服务角色中，选择向授予 CodeDeploy 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 EC2 Auto Scaling 组：输入或选择要部署应用程序修订的 Amazon EC2 Auto Scaling 组的名称。当新的 Amazon EC2 实例作为 Amazon EC2 Auto Scaling 组一部分启动时，CodeDeploy 可自动将您的修订部署到这些新实例。您最多可以将 10 个 Amazon EC2 Auto Scaling 组添加到一个部署组。

    b. Amazon EC2 实例或者本地实例：在密钥和值字段中，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. 您可以在值字段以标识以特定模式标记的所有实例，例如类似的 Amazon EC2 实例、成本中心和组名称等。例如，如果您选择名称中的密钥字段并输入**GRP-*a**中的值字段中，CodeDeploy 标识符合该模式的所有实例，例如**GRP-1a**、**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 查找与每个指定的键值对或 Amazon EC2 Auto Scaling 组名称匹配的实例时，它将显示匹配实例的数量。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 (p. 163)。

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 CodeDeploy 中使用部署配置 (p. 206)。
13. （可选）在负载均衡器中，选择启用负载均衡，然后选择现有 Classic Load Balancer、Application Load Balancer 器或 Network Load Balancer 来在部署过程中管理实例流量。

    在部署期间，每个实例都会从负载均衡器 (Classic Load Balancer) 或目标组（应用程序负载均衡器和网络负载均衡器）注销，以防止流量路由到该实例。部署完成后会重新注册。

    有关 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing (p. 42).
14. （可选）展开Advanced，然后配置您要包含在部署中任何选项，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 (p. 236)。
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，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。

为蓝/绿部署创建应用程序（控制台）

使用 CodeDeploy 控制台为蓝/绿部署创建应用程序：

Note

部署到AmazonLambda 计算平台始终是蓝/绿部署。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：

- 您没有已安装有 CodeDeploy 代理且要在蓝/绿部署过程中替换的实例。要设置您的实例，请按照使用 CodeDeploy 的实例 (p. 162)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 (p. 210)中的说明操作，然后执行本主题中的步骤。
- 您没有至少具有中所述的信任和权限的信任的 CodeDeploy 的服务角色。第 2 步：为 CodeDeploy 创建服务角色 (p. 26). 要创建和配置服务角色，请按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明操作，然后执行本主题中的步骤。
- 您尚未在 Elastic Load Balancer (经典 Load Balancer) 中创建 Classic Load Balancer (应用程序负载均衡器) 或 Network Load Balancer (在替换环境中注册实例)。有关更多信息，请参阅在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)。

1. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

Note

使用您在开始使用 CodeDeploy (p. 26).

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 Compute platform (计算平台) 中，选择 EC2/On-Premises (EC2/本地)。
5. 选择 Create application。
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

   如果您需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 EC2 Auto Scaling 组名称和部署配置），请在此页上选择这些设置。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. In服务角色中，选择向授予 CodeDeploy 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 EC2 Auto Scaling 组：CodeDeploy 将通过复制您指定的扩展组创建 Amazon EC2 Auto Scaling 组。

    b. 手动预置实例：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 EC2 Auto Scaling 组：InAmazon EC2 Auto Scaling 组中，选择或输入要用作替换环境中实例 Amazon EC2 Auto Scaling 组的模板的 Amazon EC2 自动扩展组的名称。您选择的 Amazon EC2 Auto Scaling 组中当前运行正常的实例数将在替换环境中创建。
    - 如果您选择了手动预置实例：启用Amazon EC2 Auto Scaling 组、Amazon EC2 实例，也可以指定要添加到此部署组的实例。输入 Amazon EC2 标签值或 Amazon EC2 Auto Scaling 组名称以标识原始环境中的实例 (即，您要替换的实例或正在运行当前应用程序修订的实例)。
12. （可选）在负载均衡器，启用启用负载均衡，然后选择现有 Classic Load Balancer、Application Load Balancer 器或 Network Load Balancer 来在部署过程中管理实例流量。

    在部署期间，每个实例都会从负载均衡器 (Classic Load Balancer) 或目标组（应用程序负载均衡器和网络负载均衡器）注销，以防止流量路由到该实例。部署完成后会重新注册。

    有关 CodeDeploy 部署的负载均衡器的更多信息，请参阅Integrating CodeDeploy with Elastic Load Balancing (p. 42).
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. 要更改蓝/绿部署的部署设置，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，这些实例将立即自动注册到负载均衡器，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。

设置	选项
	• 我会选择是否重新路由流量：除非您手动重新路由流量，否则替换环境中的实例不会注册到负载均衡器。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	选择在替换环境中实例对于负载均衡器的注册频率，例如每次一个或一次全部。 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 CodeDeploy 中使用部署配置 (p. 206)。
Original instances	• 终止部署组中的原始实例：在流量路由到替换环境后，已从负载均衡器取消注册的实例将在您指定的等待时间后终止。 • 使部署组中的原始实例保持运行：在流量路由到替换环境后，已从负载均衡器取消注册的实例将继续运行。

15. （可选）在Advanced中，配置您要包含在部署中的选项，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 (p. 236)。
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，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。

创建 Amazon ECS 服务的应用程序 (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. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 在导航窗格中，展开部署，然后选择入门.
3. 在 Create application (创建应用程序) 页面上，选择 Use CodeDeploy (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. 从计算平台中，选择Amazon ECS.
6. 选择 Create application。
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的内容的更多信息，请参阅在开始 Amazon ECS 部署之前 (p. 14).
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。虽然这个新的组可能与现有组同名，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。

9. In服务角色中，选择向 CodeDeploy 授予 Amazon ECS 服务的角色。有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。
10. 从负载均衡器名称中，选择向 Amazon ECS 服务提供流量的负载均衡器的名称。
11. 从生产侦听器端口中，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，选择在部署期间将流量路由到 Amazon ECS 服务中的替换任务集的流量的测试侦听器的端口和协议。您可以在 AppSpec 文件中指定一个或多个 Lambda 项目，这些项目在`AfterAllowTestTraffic`挂钩。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，则在`AfterAllowTestTraffic`挂钩。有关更多信息，请参阅Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。
13. 从 Target group 1 name (目标组 1 名称) 和 Target group 2 name (目标组 2 名称) 中，选择用于在部署期间路由流量的目标组。CodeDeploy 会将一个目标组绑定到 Amazon ECS 服务的原始任务集，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者指定何时重新路由流量，以确定何时将流量重新路由到您更新的 Amazon ECS 服务。

    如果您选择立即重新路由流量，则部署会在置备替换任务集后自动重新路由流量。

    如果您选择指定何时重新路由流量，然后选择成功置备替换任务集后等待的天数、小时数和分钟数。在此等待时间内，将执行 AppSpec 文件中指定的 Lambda 函数中的验证测试。如果等待时间在重新路由通信之前过期，则部署状态将更改为`Stopped`.
15. 适用于原始修订终止中，选择在成功部署后在 Amazon ECS 服务中的原始任务集终止之前等待的天数、小时数和分钟数。
16. （可选）在Advanced中，配置您要包含在部署中任何选项，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 (p. 236)。

创建一个应用程序Amazon Lambda函数部署（控制台）

您可以使用 CodeDeploy 控制台为Amazon Lambda函数部署。

1. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 在导航窗格中，展开部署，然后选择入门.
3. 在 Create application (创建应用程序) 页面上，选择 Use CodeDeploy (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 Create application。
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。虽然这个新的部署组可能与现有部署组同名，CodeDeploy 认为它们是两个部署组，因为与它们关联的应用程序不同。

9. In服务角色中，选择向授予 CodeDeploy 访问权限的服务角色Amazon Lambda. 有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。
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 上的部署配置Amazon Lambda计算平台 (p. 209)。
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

       Amazon Lambda 部署的最大长度为两天或 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Advanced中，配置您要包含在部署中任何选项，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 (p. 236)。
13. 选择 Create deployment group (创建部署组)。

创建应用程序 (CLI)

使用Amazon CLI创建应用程序，请调用创建应用程序命令，并指定唯一表示该应用程序的名称。（在Amazon帐户，只能为每个区域使用 CodeDeploy 应用程序名称一次。 您可在不同的区域中重用应用程序名称。）

使用 Amazon CLI 创建应用程序后，下一步是创建部署组，指定部署修订的实例。有关说明，请参阅使用 CodeDeploy 创建部署组 (p. 225)。

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，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。

使用 CodeDeploy 查看应用程序详细信息

可以使用 CodeDeploy 控制台、Amazon CLI，或 CodeDeploy API 查看有关与您的Amazonaccount.

主题

- 查看应用程序详细信息 (p. 220)
- 查看应用程序详细信息 (CLI) (p. 220)

查看应用程序详细信息

使用 CodeDeploy 控制台查看应用程序详细信息：

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
2. 在导航窗格中，展开部署，然后选择入门。
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-get-应用程序命令，指定应用程序名称。

要查看有关多个应用程序的详细信息，请调用批量 get-get-get-get-get-命令，并指定多个应用程序名称。

要查看应用程序名称列表，请调用列出应用程序命令。

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. 登录到Amazon Web Services Management Console，CodeDeployhttps://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. In详细信息类型中，选择基本如果您只想在通知中包含提供给 Amazon EventBridge 的信息。选择FULL（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）。

   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	已失败

类别	事件
	Succeeded 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 将事件发布到主题的策略。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，请在目标下，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 主题和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。您还可以设置通知和之间的集成，Amazon Chatbot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 。配置通知和Amazon Chatbot。

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的通知规则**MyNotificationRule**的应用程序，用于名为*MyDeploymentApplication*在AmazonID 为的账户*123456789012*。包含详细信息类型的通知发送到名为的 Amazon SNS 主题*#####-######*部署成功时：

```
{
    "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

您可以将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调用列出应用程序命令。

有关使用Amazon CLI要更改应用程序名称，请参阅更新应用程序。

有关使用 CodeDeploy API 更改应用程序名称的信息，请参阅API_ 更新应用程序。

删除 CodeDeploy 中的应用程序

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。此操作无法撤消。

主题

- 删除应用程序（控制台）(p. 222)
- 删除应用程序 (Amazon CLI）(p. 223)

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. 登录到Amazon Web Services Management Console，然后在 CodeDeploy 控制台中打开https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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。

   此时将显示一个包含应用程序详细信息的页面
4. 选择删除应用程序在右上角。
5. 出现提示时请输入**delete**以确认您要删除该应用程序，然后选择Delete。

删除应用程序 (Amazon CLI)

使用Amazon CLI删除应用程序，请调用删除应用程序命令，指定应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。

在 CodeDeploy 中使用部署组

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 云服务器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。

中的部署组Amazon Lambda计算平台部署

在AmazonLambda 部署中，部署组定义了一组 CodeDeploy 配置，以便在未来部署Amazon Lambdafunction. 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。单个部署AmazonLambda 部署组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/本地部署中，部署组是一组作为部署目标的单个实例。部署组中包含独标记的实例和/或 Amazon EC2 Auto Scaling 组中的 Amazon EC2 实例。

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。

在蓝/绿部署中，流量将通过以下方式从一组实例重新路由到另一组实例：从负载均衡器取消注册原始实例并注册一组替换实例，这组替换实例通常安装了最新的应用程序修订。

您可以在 CodeDeploy 中将多个部署组与某个应用程序关联。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 (p. 163)。

当您使用 CodeDeploy 控制台创建应用程序时，可同时配置其首个部署组。当您使用 Amazon CLI 创建应用程序时，可在单独的步骤中创建其首个部署组。

要查看已与您的 Amazon 账户关联的部署组的列表，请参阅使用 CodeDeploy 查看部署组详细信息 (p. 234)。

有关 Amazon EC2 实例标签的信息，请参阅通过控制台使用标签。有关本地实例的信息，请参阅Working with On-Premises Instances (p. 179)。有关 Amazon EC2 Auto Scaling 的信息，请参阅将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

主题

- the section called “创建部署组” (p. 225)
- the section called “查看部署组详细信息” (p. 234)
- the section called “更改部署组设置” (p. 235)

- the section called “为部署组配置高级选项” (p. 236)
- the section called “删除部署组” (p. 238)

使用 CodeDeploy 创建部署组

您可以使用 CodeDeploy 控制台、Amazon CLI、CodeDeploy API 或Amazon CloudFormation模板创建部署组。有关使用 Amazon CloudFormation 模板创建部署组的信息，请参阅 Amazon CloudFormation用于 CodeDeploy 参考的模板 (p. 353)。

当您使用 CodeDeploy 控制台创建应用程序时，可同时配置其第一个部署组。当您使用 Amazon CLI 创建应用程序时，可在单独的步骤中创建其首个部署组。

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，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

主题

- 为就地部署创建部署组（控制台） (p. 225)
- 为 EC2/本地蓝/绿部署（控制台）创建部署组 (p. 227)
- 为 Amazon 云服务器部署创建部署组（控制台） (p. 229)
- 在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)
- 为 CodeDeploy Amazon 云服务器部署设置负载均衡器、目标组和侦听器 (p. 230)
- 创建部署组 (CLI) (p. 233)

为就地部署创建部署组（控制台）

使用 CodeDeploy 控制台为就地部署创建部署组：

> Warning
>
> 以下情况下请勿按照这些步骤操作：
>
> - 您没有准备好要在应用程序的第一个 CodeDeploy 部署中使用的实例。要设置您的实例，请按照使用 CodeDeploy 的实例 (p. 162)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 (p. 210)中的说明操作，然后执行本主题中的步骤。
> - 您没有至少具有中所述的信任和权限的信任 CodeDeploy 的服务角色。第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。要创建和配置服务角色，请按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明操作，然后执行本主题中的步骤。
> - 您需要为就地部署选择 Classic Load Balancer、应用程序 Elastic Load Balancing cer 或，但尚未创建它。

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   > Note
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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

如果您需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 EC2 Auto Scaling (或) 和部署配置），请在此页上指定这些设置。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。

6. In服务角色中，选择向授予 CodeDeploy 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 Environment configuration (环境配置) 中，选择以下任一项：

   a. Amazon EC2 Auto Scaling 组：输入或选择要部署应用程序修订的 Amazon EC2 Auto Scaling Groups 的名称。当新的 Amazon EC2 实例作为 Amazon EC2 自动 Auto Scaling 组一部分启动时，CodeDeploy 可自动将您的修订部署到这些新实例。您最多可以将 10 个 Amazon EC2 Auto Scaling Groups 添加到一个部署组。

   b. Amazon EC2 实例或者本地实例：在密钥和值字段中，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. 您可以在值字段标识以特定模式标记的所有实例，例如类似的 Amazon EC2 实例、成本中心和组名称等。例如，如果选择名称中的密钥字段并输入**GRP-*a**中的值字段中，CodeDeploy 标识符合该模式的所有实例，例如**GRP-1a**、**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 查找与每个指定的键值对或 Amazon EC2 Auto Scaling 组名称匹配的实例时，它将显示匹配实例的数量。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 (p. 163)。

9. In代 Systems Manager 置中，指定您希望在部署组中的实例上安装和更新 CodeDeploy 代理的方式。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 Systems Manager？

   a. 从来没有：跳过使用 Systems Manager 配置 CodeDeploy 安装的步骤。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. 只有一次：Systems Manager 将在部署组中的每个实例上安装一次 CodeDeploy 代理。

   c. 现在并安排更新：Systems Manager 将创建与状态管理器的关联，这将根据您配置的计划安装 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 CodeDeploy 中使用部署配置 (p. 206)。
11. （可选）在负载均衡器中，选择启用负载均衡，然后选择现有 Classic Load Balancer、Application Load Balancer 器或 Network Load Balancer 来在部署过程中管理实例流量。

    在部署期间，每个实例都会从负载均衡器（Classic Load Balancer）或目标组（应用程序负载均衡器和网络负载均衡器）注销，以防止流量路由到该实例。部署完成后会重新注册。

    有关 CodeDeploy 部署的负载均衡器的详细信息，请参阅Integrating CodeDeploy with Elastic Load Balancing (p. 42)。
12. （可选）展开Advanced并配置您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 (p. 236)。

13. 选择 Create deployment group (创建部署组)。

为 EC2/本地蓝/绿部署（控制台）创建部署组

使用 CodeDeploy 控制台为蓝/绿部署创建部署组：

> **Warning**
>
> 以下情况下请勿按照这些步骤操作：
>
> - 您没有已安装有 CodeDeploy 代理且要在蓝/绿部署过程中替换的实例。要设置您的实例，请按照使用 CodeDeploy 的实例 (p. 162)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 (p. 210)中的说明操作，然后执行本主题中的步骤。
> - 您没有至少具有中所述的信任和权限的信任 CodeDeploy 的服务角色。第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。要创建和配置服务角色，请按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)中的说明操作，然后执行本主题中的步骤。
> - 您尚未在 Elassic Load Balancer (弹性 Application Load Balancer) 中创建 Classic Load Balancer 来注册替换环境中的实例。有关更多信息，请参阅在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)。

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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**
   >
   > 如果您需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 EC2 Auto Scaling 组名称和部署配置），请在此页上选择这些设置。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。

6. In服务角色中，选择向授予 CodeDeploy 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 Environment configuration 中，选择为替换环境提供实例的方法：

   a. 自动复制 Amazon EC2 Auto Scaling Groups：CodeDeploy 将通过复制您指定的 Amazon EC2 Auto Scaling Groups 来创建 Amazon EC2 Auto Scaling

   b. 手动预配置实例：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。

9. In代 Systems Manager 置中，指定您希望在部署组中的实例上安装和更新 CodeDeploy 代理的方式。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 Systems Manager？

   a. 从来没有：跳过使用 Systems Manager 配置 CodeDeploy 安装的步骤。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. 只有一次：Systems Manager 将在部署组中的每个实例上安装一次 CodeDeploy 代理。

c. 现在并安排更新：Systems Manager 将创建与状态管理器的关联，这将根据您配置的计划安装 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：

- 如果您选择自动复制 Amazon EC2 Auto Scaling Groups：InAmazon EC2 Auto Scaling Groups中，选择或输入要用作 Amazon EC2 Auto Scaling 组的模板的 Amazon EC2 自动扩展组的名称，该组是为替换环境中的实例创建的。您选择的 Amazon EC2 Auto Scaling instance group 中当前运行正常的实例数将在替换环境中创建。
- 如果您选择手动预配置实例：SelectAmazon EC2 Auto Scaling 组、Amazon EC2 Auto Scaling，也可以指定要添加到此部署组的实例。输入 Amazon EC2 Auto Scaling) 标签值或 Amazon EC2 Auto Scaling instance (启用当前应用程序修订的实例)，以标识原始环境中的实例 (即，您要替换的实例或正在运行当前应用程序修订的实例)。

11. （可选）在负载均衡器中，选择启用负载均衡，然后选择现有 Classic Load Balancer、Application Load Balancer 器或 Network Load Balancer 来在部署过程中管理实例流量。

    在部署期间，每个实例都会从负载均衡器（Classic Load Balancer）或目标组（应用程序负载均衡器和网络负载均衡器）注销，以防止流量路由到该实例。部署完成后会重新注册。

    有关 CodeDeploy 部署的负载均衡器的详细信息，请参阅Integrating CodeDeploy with Elastic Load Balancing (p. 42)。

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 13。

13. 要更改蓝/绿部署的部署设置，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，这些实例将立即自动注册到负载均衡器，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。 • 我会选择是否重新路由流量：除非您手动重新路由流量，否则替换环境中的实例不会注册到负载均衡器。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	选择在替换环境中实例对于负载均衡器的注册频率，例如每次一个或一次全部。 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 CodeDeploy 中使用部署配置 (p. 206)。
Original instances	• 终止部署组中的原始实例：将流量路由到替换环境后，已从负载均衡器取消注册的实例将在您指定的一段等待时间后终止。

设置	选项
	• 使部署组中的原始实例保持运行：将流量路由到替换环境后，已从负载均衡器取消注册的实例将继续运行。

14. （可选）在Advanced中，配置您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 (p. 236)。
15. 选择 Create deployment group (创建部署组)。

为 Amazon 云服务器部署创建部署组（控制台）

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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的内容的更多信息，请参阅在开始 Amazon ECS 部署之前 (p. 14)。
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。虽然这个新的组可能与现有组同名，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。
7. In服务角色中，选择向授予 CodeDeploy 访问 Amazon ECS 的权限的服务角色。有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。
8. 从负载均衡器名称中，选择向 Amazon ECS 服务提供流量的负载均衡器的名称。
9. 从生产侦听器端口中，选择将生产流量传输至 Amazon ECS 服务的侦听器的端口和协议。
10. （可选）根据测试侦听器端口中，选择将流量传送到 Amazon ECS 服务中的替换任务集的测试侦听器的端口和协议。您可以在 AppSpec 文件中指定一个或多个 Lambda 模拟项目，在`AfterAllowTestTraffic`挂钩。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，则在`AfterAllowTestTraffic`挂钩。有关更多信息，请参阅Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。
11. 从 Target group 1 name (目标组 1 名称) 和 Target group 2 name (目标组 2 名称) 中，选择用于在部署期间路由流量的目标组。CodeDeploy 将一个目标组绑定到 Amazon ECS 服务的原始任务集，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者指定何时重新路由流量，以确定何时将流量重新路由到您更新的 Amazon ECS 服务。

    如果选择立即重新路由流量，则部署会在置备替换任务集后自动重新路由流量。

    如果选择指定何时重新路由流量，然后选择成功置备替换任务集后等待的天数、小时数和分钟数。在此等待时间内，将执行 AppSpec 文件中指定的 Lambda 函数中的验证测试。如果等待时间在重新路由通信之前过期，则部署状态将更改为`Stopped`。

13. 适用于原始修订终止中，选择在成功部署后在 Amazon ECS 服务中的原始任务集终止之前等待的天数、小时数和分钟数。
14. （可选）在Advanced中，配置您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 (p. 236)。

在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器

在运行任何蓝/绿部署或在地部署 (要为其指定部署组中的可选负载均衡器) 之前，您必须已在 Elassic Load Balancer (启用负载均衡器)、Application Load Balancer 负载均衡器或 Network Load Balancer。对于蓝/绿部署，您使用该负载均衡器注册构成替换环境的实例。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。

要配置 Classic Load Balancer，请按照教程：创建 Classic Load Balancer在Classic Load Balancer 用户指南。请注意以下几点：

- In步骤 2: 定义负载均衡器, 在在内创建 LB中，选择创建实例时所选的同一 VPC。
- In步骤 5: 向 Load Balancer 注册 EC2 实例中，选择当前位于部署组中的实例（就地部署）或已指定位于原始环境中的实例（蓝/绿部署）。
- In步骤 7：创建并验证您的负载均衡器，请记下负载均衡器的 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 Application Load Balancer，请遵循以下某一主题中的说明：

- 创建 Application Load Balancer
- 教程：使用创建 Application Load BalancerAmazon CLI

要配置 Network Load Balancer，请遵循以下某一主题中的说明：

- 创建网络负载均衡器
- 教程：创建 Network Load BalancerAmazon CLI

要配置 Classic Load Balancer，请按照教程：创建 Classic Load Balancer。

为 CodeDeploy Amazon 云服务器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，您必须创建应用 Application Load Balancer 或 Network Load Balancer、两个目标组以及一个或两个侦听器。本主题介绍如何创建 Application Load Balancer。有关更多信息，请参阅在开始 Amazon ECS 部署之前 (p. 14)。

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。在部署过程中，CodeDeploy 会创建替换任务集，并将流量从原始任务集重新路由到新的任务集。CodeDeploy 确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。以下步骤介绍了如何确认默认 VPC、创建 Amazon EC2 Application Load Balancer，然后为负载均衡器创建两个目标组。有关更多信息，请参阅 。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

本主题演示了如何创建 Amazon EC2 Application Load Balancer、两个目标组以及两个端口，以便在 Amazon ECS 取消过程中使用。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. 登录到 Amazon Web Services Management Console 并打开 Amazon VPC 控制台，网址：https://console.aws.amazon.com/vpc/。
2. 验证要使用的默认 VPC。在导航窗格中，选择 Your VPCs。请注意哪些 VPC 显示是中的默认 VPCcolumn. 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)。请记下两个子网的子网 ID，这两个子网显示是中的默认子网column. 在创建负载均衡器时，您可以使用这些 ID。
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. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
7. 从导航窗格中，选择 Security Groups。
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 Amazon EC2 Application Load Balancer、两个目标组和侦听器（控制台）

要使用 Amazon EC2 控制台创建 Amazon EC2 应用 Application Load Balancer，请执行以下操作：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，选择 Load Balancers。
3. 选择 Create Load Balancer。
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 Next:。配置安全设置。
11. 选择 Next:。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 Next:。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 Next:。注册目标。
15. 选择 Next:。审核，然后选择Create。

为您的负载均衡器创建第二个目标组

1. 在预配置您的负载均衡器后，打开 Amazon EC2 控制台。在导航窗格中，选择 Target Groups。
2. 选择 Create target group。
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，您必须已为负载均衡器创建两个目标组。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 。步骤 4: 创建 Amazon ECS 服务中的Amazon ECS 用户指南。

创建 Amazon EC2 Application Load Balancer、两个目标组和侦听器 (CLI)

使用创建 Application Load BalancerAmazon CLI：

1. 使用创建负载均衡器命令创建 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. 使用创建目标组命令创建您的第一个目标组。CodeDeploy 将该目标组的流量路由到您服务中的原始任务集或替换任务集。

```
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. 使用创建目标组命令创建第二个目标组。CodeDeploy 将目标组的流量路由到不由第一个目标组提供的任务集。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)

使用Amazon CLI创建部署组，请调用创建部署组命令，在命令中指定：

- 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
- 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。
- 与用于标识要包含在部署组中的实例的标签、标签组或 Amazon EC2 Auto Scaling Balancer (标签组) 有关的信息。
- 服务角色的 Amazon 资源名称 (ARN) 标识符，该服务角色允许 CodeDeploy 代表您的Amazon帐户与其他Amazon服务。要获取服务角色 ARN，请参阅获取服务角色 ARN (CLI) (p. 29)。有关服务角色的详细信息，请参阅角色术语和概念在IAM 用户指南。

- 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 (p. 210)。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 简单通知服务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 (p. 287)。
- （可选）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出设定的阈值，将激活警报。
- （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。
- 对于就地部署：
  - （可选）Elassic Load Balancer (Elastic Load Balancing cer) 的名称，用于在部署过程中管理实例流量的 Classic Load Balancer。
- 对于蓝/绿部署：
  - 蓝/绿部署配置过程：
    - 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。
  - 用于在替换环境中注册的实例的 Elastic Load Balancing 器或应用程序负载均衡器的名称。

使用 CodeDeploy 查看部署组详细信息

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） (p. 234)
- 查看部署组详细信息 (CLI) (p. 234)

查看部署组详细信息（控制台）

使用 CodeDeploy 控制台查看部署组详细信息：

1. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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。在导航栏中的区域选择器中，选择一个区域和终端节点中的Amazon一般参考。CodeDeploy 仅在这些地区受支持。
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，请调用获取部署组命令，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，请调用列出应用程序命令。
- 部署组名称。要获取部署组名称，请调用列表部署组命令。

要查看部署组名称的列表，请调用列表部署组命令，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，请调用列出应用程序命令。

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 (p. 210)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，而且至少具有第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。要创建和配置具有正确权限的服务角色，请按照第 2 步：为 CodeDeploy 创建服务角色 (p. 26)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）(p. 235)
- 更改部署组设置 (CLI) (p. 235)

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. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在中使用的相同账户或 IAM 用户信息登录开始使用 CodeDeploy (p. 26)。

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。在导航栏上的区域选择器中，选择区域和终端节点中的Amazon一般参考。仅在这些地区支持 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 CodeDeploy 创建部署组 (p. 225)。

7. 选择保存更改。

更改部署组设置 (CLI)

使用Amazon CLI更改部署组设置，请调用更新部署组命令，在命令中指定：

- 对于 EC2/本地和AmazonLambda 部署：
  - 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
  - 当前部署组名称。要查看部署组名称列表，请调用列表部署组命令。
  - （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)，该服务角色允许 CodeDeploy 对Amazon帐户在与其他Amazon服务。要获取服务角色 ARN，请参阅获取服务角色 ARN (CLI) (p. 29)。有关服务角色的详细信息，请参阅角色术语和概念在IAM 用户指南。
  - （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 (p. 210)。（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）用于创建或更新触发器的命令。触发器用于向 Amazon 简单通知服务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 (p. 287)。
- 仅适用于 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：
  - 要部署的亚马逊云服务器服务。
  - 负载均衡器信息，包括 Application Load Balancer 器或 Network Load Balancer、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 CodeDeploy 创建应用程序 (p. 212)
- 使用 CodeDeploy 创建部署组 (p. 225)
- 使用 CodeDeploy 更改部署组设置 (p. 235)

Amazon SNS 通知触发器：您可以将触发器添加到 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 Amazon SNS 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 必须有权从此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 (p. 287)。

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 自动扩展组创建一个警报。对于AmazonLambda 和 Amazon 云服务器部署时，您可以在 Lambda 函数中为错误创建警报。

您可以将部署配置为在 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 CloudWatch 按照中创建的名称输入云监控警报。要查 CloudWatch 警报列表，请在https://console.amazonaws.cn/cloudwatch/，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 检索警报状态时继续，请选择即使警报状态不可用，仍继续部署。

  Note

  此选项对应于ignorePollAlarmFailure中的AlarmConfiguration对象中的 CodeDeploy API。

有关更多信息，请参阅使用 CodeDeploy 中的 CloudWatch 警报监控部署 (p. 283)。

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 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 (p. 271)。

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修订部署到您的 Amazon EC2 实例。例如，如果您的 EC2 实例在 CodeDeploy 部署过程中启动到 Auto Scaling 组 (ASG) 中，则这些实例将接收应用程序的旧版本，而不是最新版本。为了使这些实例处于最新状态，CodeDeploy 会自动启动后续部署（立即在第一个部署之后），以更新任何过时的实例。如果您想更改此默认行为，以便使过时的 EC2 实例保留在旧版本中，则可以通过 CodeDeploy API 或Amazon Command Line Interface(CLI).

要通过 API 配置过时实例的自动更新，请在`outdatedInstancesStrategy`请求参数`UpdateDeploymentGroup`或者`CreateDeploymentGroup`action. 有关详细信息，请参阅。Amazon CodeDeployAPI 参考。

要配置自动更新，请通过Amazon CLI，请使用以下任一命令：

```
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，##替换为部署所需的参数，## | IGNORE替换为UPDATE启用自动更新，或IGNORE以禁用它们。

例如：

```
aws deploy update-deployment-group --application-name "MyApp" --current-
deployment-group-name "MyDG" --region us-east-1 --outdated-instances-strategy
IGNORE
```

有关详细信息Amazon CLI命令中，请参阅。Amazon CLI命令参考。

删除使用 CodeDeploy 的部署组

您可以使用 CodeDeploy 控制台、Amazon CLI或 CodeDeploy API 删除与Amazonaccount.

> **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。此操作无法撤消。

主题

- 删除部署组（控制台）(p. 238)
- 删除部署组 (CLI) (p. 238)

删除部署组（控制台）

要使用 CodeDeploy 控制台删除部署组：

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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)

使用Amazon CLI删除部署组，请调用删除部署组命令，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，请调用列表应用程序命令。
- 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，请调用列表部署组命令。

使用 CodeDeploy 的应用程序修订

在 CodeDeploy 中，修订包含将部署到您实例的源文件版本或者 CodeDeploy 将在您的实例上运行的脚本版本。

您 AppSpec 划修订、将添加到修订，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

- 规划 CodeDeploy 的修订版 (p. 239)
- 将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)
- 选择 CodeDeploy 存储库类型 (p. 246)
- 将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署） (p. 247)
- 查看应用程序修订详细信息 (p. 249)
- 使用 CodeDeploy 在 Amazon S3 中注册应用程序修订版 (p. 251)

规划 CodeDeploy 的修订版

良好的计划使部署修订变得轻松多了。

对于到的部署AmazonLambda 或亚马逊云服务器计算平台，修订版本与 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 的修订版 (p. 239)。

对于到 EC2/ 本地计算平台的部署，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`/tmp/`(在 Linux、macOS 或 Unix) 上，或。`c:\temp`根文件夹：

```
/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
```

此根文件夹还应包含应用程序规范文件（AppSpec 文件），如下所示。有关更多信息，请参阅 将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)。

将应用程序规范文件添加到 CodeDeploy 的修订版

本主题介绍如何向部署中添加 AppSpec 文件。它还包括 AppSpec 来为AmazonLambda 和 EC2/ 本地部署。

主题

- 为 Amazon ECS 部署添加 AppSpec 文件 (p. 240)
- AppSpecAmazonLambda 部署 (p. 242)
- 为 EC2/本地部署部署添加AppSpec 文件 (p. 243)

为 Amazon ECS 部署添加 AppSpec 文件

对于 Amazon ECS 计算平台的部署，请执行以下操作：

- 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 格式或 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中。有关更多信息，请参阅 创建 Amazon ECS 计算平台部署（控制台） (p. 256)。

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。

用于亚马逊云服务器部署的 YAML AppSpec 文件模板，附有说明

下面是 Amazon ECS 部署 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Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。

```
# 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 your 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 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Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。

```
{
 "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": ""
  }
 ]
}
```

AppSpecAmazonLambda 部署

对于到的部署AmazonLambda 计算平台：

- 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 格式或 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AmazonLambda 计算平台部署（控制台）(p. 257)。

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。

YAML AppSpec 文件模板Amazon Lambda带说明的部署

有关在 hooks 部分中使用的生命周期事件的信息，请参阅用于 AppSpec 的 “hooks” 部分AmazonLambda 部署 (p. 338)。

```
# 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.
```

JSON AppSpec 文件Amazon Lambda部署模板

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，请参阅用于 AppSpec 的 “hooks” 部分AmazonLambda 部署 (p. 338)。

```
{
  "version": 0.0,
  "Resources": [{
   "MyFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "",
     "Alias": "",
     "CurrentVersion": "",
     "TargetVersion": ""
    }
   }
  }],
  "Hooks": [{
    "LifeCycleEvent": ""
   }
  ]
 }
```

为 EC2/本地部署部署添加AppSpec 文件

如果没有 AppSpec 文件，CodeDeploc 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。

将 AppSpec 文件添加到修订：

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 AppSpec 文件的有效性。
4. 在修订的根目录中将文件另存为 `appspec.yml`。
5. 运行下列命令之一，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
     ```

     如果在其中找不到 AppSpec 文件，则不输出任何内容。
   - 对于 Windows：

     ```
     dir path\to\root\directory\appspec.yml
     ```

A找不到文件如果 AppSpec 文件未存储在那里，则会显示错误。

6. 将修订推送到 Amazon S3 或 GitHub。

   有关说明，请参阅将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署）(p. 247)。

EC2/本地部署的AppSpec 文件模板及说明

Note

部署到 Windows 服务器实例不支持`runas`元素。如果要部署到 Windows 服务器实例，请不要将其包含在 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repository。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。
- AmazonLambda和Amazon ECS：部署需要一个 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 存储桶中。
  - 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AmazonLambda 计算平台部署（控制台）(p. 257) 和 创建 Amazon ECS 计算平台部署（控制台） (p. 256)。
  - 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AmazonLambda 计算平台部署 (CLI) (p. 261) 和 创建 Amazon ECS 计算平台部署 (CLI) (p. 261)。

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解决方案实现安全、可扩展的对象存储。Amazon S3 将数据存储为存储段。对象由文件和描述该文件的任何可选元数据组成。 要将对象存储在 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 • 将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/本地部署）(p. 247) • 使用 CodeDeploy 从 Amazon S3 中自动部署	使用以下计算平台的部署可以将修订存储在Amazon S3存储桶。 • 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	只有 EC2/ 本地部署可以将修订存储在 GitHub 存储库中。

	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： • GitHub 将 CodeDeploy 集成 (p. 47) • 教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101)	
Bitbucket	您可以通过使用CodeDeploy 管道在Bitbucket 管道。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： • 查看 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

一个Amazon Lambda部署仅适用于 Amazon S3 存储库。

将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署）

规划修订后，如规划 CodeDeploy 的修订版 (p. 239)并将 AppSpec 文件添加到修订版中，如将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)，您已准备好捆绑组件文件并将修订推送到 Amazon S3。对于向 Amazon EC2 实例中进行的部署，在您推送修订后，可以使用 CodeDeploy 将修订从 Amazon S3 部署到实例。

Note

CodeDeploy 还可用于将已推送的修订部署到 GitHub。有关更多信息，请参阅 GitHub 文档。

我们假定您已遵循 开始使用 CodeDeploy (p. 26)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。按照中的说明进行操作创建存储桶。

如果要部署到 Amazon EC2 实例，则必须创建目标 Amazon S3 存储桶或者它必须存在于目标实例所在的区域中。例如，如果您要将修订部署到美国东部（弗吉尼亚北部）区域中的某些实例，并将其他实例部署到美国西部（俄勒冈）区域中，则必须有一个带修订的一个副本的存储桶，并且其他存储桶位于美国西部（俄勒冈）区域和另一个带另一个带另一个带另一个带另一个带另一个带另一个带另一个带另一个存储桶。同一修订版的副本。在此方案中，您随后需要分别在美国东部（弗吉尼亚北部）区域和美国西部（俄勒冈）区域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。

您必须具有上传到 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 Amazon S3 存储桶中名为的任何目录`codedeploydemobucket`：

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

查看您的Amazon账户 ID，请参阅查找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 IAM 用户将修订上传到名为的 Amazon S3 存储桶中的任意位置。`codedeploydemobucket`：

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

要了解如何创建和附加 IAM 策略，请参阅使用策略。

推送修订版本使用Amazon CLI

> **Note**
>
> 这些区域有：`push`命令会将应用程序项目和 AppSpec 文件绑定到修订中。此修订的文件格式是压缩的 ZIP 文件。该命令不能与AmazonLambda 或 Amazon ECS 部署，因为这两种部署需要作为 JSON 格式或 YAML 格式的 AppSpec 文件的修订。

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。该键是修订的名称。CodeDeploy 会在上传内容之前压缩内容。采用格式 `s3:`*`//your-S3-bucket-name/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。要部署的内容的位置，以及要压缩并上传到 Amazon S3 的开发计算机上的 AppSpec 文件。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，默认字符串 “上传由Amazon CLI 'time' time 'UTC”，其中，“time (时间)” 是采用协调世界时 (UTC) 的当前系统时间。

您可以将Amazon CLI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://codedeploydemobucket/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://codedeploydemobucket/
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 `codedeploydemobucket` 的存储桶。
- 将根目录中的所有文件绑定到修订。

在推送成功后，可以使用Amazon CLI或 CodeDeploy 控制台来部署 Amazon S3 中的修订版本。使用 Amazon CLI 部署此修订：

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=codedeploydemobucket,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=codedeploydemobucket,key=WordPressApp.zip,bundleType=zip
```

有关更多信息，请参阅使用 CodeDeploy 创建部署 (p. 254)。

查看应用程序修订详细信息

可以使用 CodeDeploy 控制台、Amazon CLI或 CodeDeploy API 查看有关注册到Amazon帐户。

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版 (p. 251)。

主题

- 查看应用程序修订详细信息 (p. 250)
- 查看应用程序修订详细信息 (CLI) (p. 250)

查看应用程序修订详细信息

要查看应用程序修订详细信息，请执行以下操作：

1. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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。在导航栏中，在区域选择器中，选择区域和终端节点中的Amazon一般参考。CodeDeploy 仅在这些地区受支持。
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，请调用获取应用程序修订命令，在命令中指定：

- 应用程序名称。要获取应用程序名称，请调用列出应用程序命令。
- 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。如果在调用注册表应用程序修订，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，请调用列表应用程序修订命令，在命令中指定：

- 应用程序名称。要获取应用程序名称，请调用列出应用程序命令。
- （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版

如果您已经调用了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，请按照以下步骤首先注册修订版本。

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，则还必须遵循以下步骤。

您只能使用Amazon CLI或者 CodeDeploy API，在 Amazon S3 或 GitHub 中注册应用程序修订。

主题

- 使用 CodeDeploy (CLI) 在 Amazon S3 中注册修订 (p. 251)
- 使用 CodeDeploy (CLI) 在 GitHub 中注册修订 (p. 252)

使用 CodeDeploy (CLI) 在 Amazon S3 中注册修订

1. 将修订上传到 Amazon S3。
2. 调用注册表应用程序修订命令，并指定：
   - 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
   - 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。适用于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 服务器实例不支持 tar 和压缩 tar 归档文件格式（.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。

可以在命令行上指定有关 Amazon S3 修订的信息，并将此语法用作register-application-revision调用。(`version`和`eTag`是可选的。)

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
```

对于修订文件，AmazonLambda 部署：

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

使用 CodeDeploy (CLI) 在 GitHub 中注册修订

> **Note**
>
> Amazon Lambda 部署不能用于 GitHub。

1. 将修订上传到 GitHub 存储库。
2. 调用注册表应用程序修订命令，并指定：
   - 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
   - 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 CodeDeploy 中使用部署

在 CodeDeploy 中，部署是指在一个或多个实例上安装内容的过程和过程中涉及的组件。此内容可包含代码、Web 和配置文件、可执行文件、程序包、脚本，等等。CodeDeploy 将根据您指定的配置规则来部署存储在源存储库中的内容。

如果您使用 EC2/本地计算平台，则可以同时运行相同实例组的两个部署。

CodeDeploy 提供了两种部署类型选项，即就地部署和蓝/绿部署。

- 就地部署：部署组中每个实例上的应用程序将停止，安装最新的应用程序修订版，然后启动并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/本地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 (p. 5)。
- 蓝/绿部署：部署的行为取决于使用的计算平台：
  - EC2/本地计算平台上的蓝/绿：部署组中的实例 (原始环境) 将被不同的实例集 (替代环境) 所代替，步骤如下：
    - 将为替换环境预置实例。
    - 替换实例上安装了最新的应用程序修订版。
    - 对于应用程序测试和系统验证等活动，会发生可选的等待时间。
    - 替换环境中的实例向弹性负载均衡器注册，导致流量重新路由到这些实例。原始环境中的实例将取消注册，可以终止，也可以继续运行，以便进行其他用途。

      Note

      如果您使用 EC2/本地计算平台，请注意蓝/绿色部署仅适用于 Amazon EC2 实例。
  - 上的蓝/绿部署AmazonLambda 计算平台：流量从当前无服务器环境转移到包含更新后的 Lambda 函数版本的环境。您可以指定执行验证测试的 Lambda 函数并选择流量转移的方式。全部AmazonLambda 计算平台部署是蓝/绿部署。因此，您无需指定部署类型。
  - Amazon ECS 计算平台上的蓝/绿：流量从 Amazon ECS 服务中采用原始版本应用程序的任务集转移到同一服务中的替换任务集。您可以通过部署配置将流量转移设置为线性或 Canary。指定负载平衡器侦听器的协议和端口用于重新路由生产流量。在部署期间，在运行验证测试时，可以使用测试侦听器为替换任务集提供流量。
  - 通过 进行蓝/绿部署Amazon CloudFormation：流量将从当前资源转移到更新的资源，作为Amazon CloudFormation堆栈更新。目前，仅支持 ECS 蓝/绿部署。

  有关蓝/绿部署的更多信息，请参阅蓝/绿部署概述 (p. 6)。

有关从 Amazon S3 自动部署的信息，请参阅使用 CodeDeploy 从 Amazon S3 自动部署。

主题

- 使用 CodeDeploy 创建部署 (p. 254)
- 查看 CodeDeploy 部署详细信息 (p. 266)
- 查看代表部署EC2/本地部署的日志数据 (p. 268)
- 使用 CodeDeploy 停止部署 (p. 270)
- 使用 CodeDeploy 重新部署和回滚部署 (p. 271)
- 将应用程序部署到不同Amazon账户 (p. 273)
- 使用 CodeDeploy 代理验证本地机器上的部署程序包 (p. 276)

使用 CodeDeploy 创建部署

可以使用 CodeDeploy 控制台、Amazon CLI或 CodeDeploy API 创建一个部署，该部署将安装您已推送到部署组中的实例上的 EC2/本地计算平台(如果您的部署目标是 EC2/本地计算平台(GitHub) 的应用程序修订。

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 (p. 254)
- 创建 Amazon ECS 计算平台部署（控制台） (p. 256)
- 创建AmazonLambda 计算平台部署（控制台）(p. 257)
- 创建EC2/本地计算平台部署（控制台）(p. 258)
- 创建 Amazon ECS 计算平台部署 (CLI) (p. 261)
- 创建AmazonLambda 计算平台部署 (CLI) (p. 261)
- 创建EC2/本地计算平台部署(CLI) (p. 262)
- 通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 264)

部署先决条件

在您开始部署之前，请确保完成以下步骤。

上的部署先决条件Amazon Lambda计算平台

- 创建一个应用程序，其中至少包括一个部署组。有关信息，请参阅 使用 CodeDeploy 创建应用程序 (p. 212) 和 使用 CodeDeploy 创建部署组 (p. 225)。
- 准备应用程序修订，该修订也称为 AppSpec 文件，用于指定您希望部署的 Lambda 函数版本。AppSpec file 还指定 Lambda 函数以验证您的部署。有关更多信息，请参阅使用 CodeDeploy 的应用程序修订 (p. 239).
- 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 (p. 210)。

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 CodeDeploy 的实例 (p. 162)。对于蓝/绿部署，您可使用现有 Amazon EC2 Auto Scaling 组作为替换环境的模板，或指定一个或多个实例或 Amazon EC2 Auto Scaling 组作为原始环境。有关更多信息，请参阅 教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88) 和 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。
- 创建一个应用程序，其中至少包括一个部署组。有关信息，请参阅 使用 CodeDeploy 创建应用程序 (p. 212) 和 使用 CodeDeploy 创建部署组 (p. 225)。
- 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 使用 CodeDeploy 的应用程序修订 (p. 239)。
- 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 (p. 210)。
- 如果您要从 Amazon S3 存储桶部署应用程序修订，则该存储桶应位于相同的Amazon区域为部署组中的实例。
- 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 存储桶策略允许附加 IAM 实例配置文件（其中包含 ARN）的任何 Amazon EC2 实例，`arn:aws:iam::444455556666:role/CodeDeployDemo`从 Amazon S3 存储桶中的任何位置下载名为`codedeploydemobucket`：

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

以下 Amazon S3 存储桶策略允许与关联的 IAM 用户（其中包含 ARN）的任何本地实例`arn:aws:iam::444455556666:user/CodeDeployUser`从 Amazon S3 存储桶中的任何位置下载名为`codedeploydemobucket`：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "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.

- 如果您正在创建蓝/绿部署，或者您已为就地部署指定部署组中的可选 Classic Load Balancer 或，则您已使用包含至少两个子网的 Amazon VPC 创建一个 VPC。（CodeDeploy 使用 Elastic Load Balancing，这需要负载均衡器组中的所有实例都位于单一 VPC 中。）

  如果您尚未创建 VPC，请参阅Amazon VPC 入门指南.

- 如果您要创建蓝/绿部署，则需要提前在 Elastic Load Balancing 中配置 Classic Load Balancer、应用程序负载均衡器或 Network Load Balancer，并使用它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。

  要配置 Classic Load Balancer，请完成教程：创建 Classic Load Balancer在Classic Load Balancer 用户指南. 在操作过程中，记下以下内容：

- In步骤 2: 定义负载均衡器, 在在内部创建 LB中，选择创建实例时所选的同一 VPC。
- In第 5 步：向 Load Balancer 注册 EC2 实例中，选择原始环境中的实例。
- In步骤 7：创建并验证您的负载均衡器，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。

  当您将 DNS 名称粘贴到 Web 浏览器的地址栏中时，您应该会看到您已为原始环境部署的应用程序。

要配置应用程序负载均衡器，请遵循以下某一主题中的说明：

- 创建 Application Load Balancer
- 教程: 使用创建 Application Load BalancerAmazon CLI

要配置 Network Load Balancer，请遵循以下某一主题中的说明：

- 创建网络负载均衡器
- 教程: 使用创建 Network Load BalancerAmazon CLI

通过使用进行蓝/绿部署的部署先决条件Amazon CloudFormation

- 您的模板不需要针对 CodeDeploy 应用程序或部署组为资源建模。
- 您的模板必须包含使用 Amazon VPC 且至少包含两个子网的 VPC 的资源。
- 您的模板必须包含用于将流量定向到目标组的弹性负载平 Classic Load Balancer 器、应用 Application Load Balancer 负载均衡器或 Network Load Balancer 的资源。

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：部署 Amazon ECS 服务 (p. 111) 和 教程：通过验证测试部署 Amazon ECS 服务 (p. 117)。

1. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 请执行下列操作之一：

   - 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。确保计算平台列为您的应用程序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。确保计算平台列为您的部署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，然后才能部署它。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 CodeDeploy 创建部署组 (p. 225)。
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，请参阅指定有关存储在 Amazon S3 存储桶中的修订的信息 (p. 259)，然后返回步骤 6。

- 使用 AppSpec 编辑器— 选择 JSON 或 YAML，然后将您的 AppSpec 文件输入编辑器中。您可以保存 AppSpec 文件，方法是选择另存为文本文件. 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 的修订版 (p. 239).

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 CodeDeploy 中的回滚的信息，请参阅。重新部署和部署回滚 (p. 13)和使用 CodeDeploy 重新部署和回滚部署 (p. 271).

   从以下选项中进行选择：

   - 部署失败时回滚— CodeDeploy 将上一个已知良好的版本重新部署为新的部署。
   - 当警报阈值被回滚— 如果为部署组添加了警报，当一个或多个指定警报激活时，CodeDeploy 将重新部署上一个已知良好的修订。
   - 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 (p. 266)。

创建AmazonLambda 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. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 请执行下列操作之一：

   - 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。确保计算平台列为您的应用程序AmazonLambda.
   - 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。确保计算平台列为您的部署AmazonLambda.
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，然后才能部署它。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 CodeDeploy 创建部署组 (p. 225)。
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，请参阅指定有关存储在 Amazon S3 存储桶中的修订的信息 (p. 259)，然后返回步骤 6。
   - 使用 AppSpec 编辑器— 选择 JSON 或 YAML，然后将您的 AppSpec 文件输入编辑器中。您可以保存 AppSpec 文件，方法是选择另存为文本文件. 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 的修订版 (p. 239).
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不同于在部署组中指定的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 上的部署配置Amazon Lambda计算平台 (p. 209)。
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 CodeDeploy 中的回滚的信息，请参阅。重新部署和部署回滚 (p. 13)和使用 CodeDeploy 重新部署和回滚部署 (p. 271).

   从以下选项中进行选择：

   - 部署失败时回滚— CodeDeploy 将上一个已知良好的版本重新部署为新的部署。
   - 当警报阈值被回滚— 如果为部署组添加了警报，当一个或多个指定警报激活时，CodeDeploy 将重新部署上一个已知良好的修订。
   - 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 (p. 266)。

创建EC2/本地计算平台部署（控制台）

本主题介绍了如何使用控制台将应用程序部署到 Amazon EC2 或本地服务器。

1. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 请执行下列操作之一：

   - 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。确保计算平台列为您的应用程序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。确保计算平台列为您的部署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，然后才能部署它。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 CodeDeploy 创建部署组 (p. 225)。
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，请参阅指定有关存储在 Amazon S3 存储桶中的修订的信息 (p. 259)，然后返回步骤 6。
   - 我的应用程序存储在 GitHub 中— 有关信息，请参阅指定有关存储在 GitHub 存储库中的修订的信息 (p. 260)，然后返回步骤 6。
6. （可选）在 Deployment description (部署描述) 框中，输入此部署的描述。
7. （可选）展开覆盖部署配置以选择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 Amazon EC2 或本地服务器。

   有关更多信息，请参阅在 CodeDeploy 中使用部署配置 (p. 206)。

8. a. Select如果 ApplicationStop 生命周期事件失败，则不要使部署失败如果您希望部署到实例成功，如果`ApplicationStop`生命周期事件失败。

   b. Expand其他部署行为设置以指定 CodeDeploy 如何处理部署目标位置上未作为上一成功部署的一部分的文件。

   从以下选项中进行选择：

   - 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，请参阅对现有内容进行回滚行为 (p. 272)。

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 CodeDeploy 中的回滚的信息，请参阅。重新部署和部署回滚 (p. 24)和使用 CodeDeploy 重新部署和回滚部署 (p. 271).

   从以下选项中进行选择：

   - 部署失败时回滚— CodeDeploy 将上一个已知良好的版本重新部署为新的部署。
   - 当警报阈值被回滚— 如果为部署组添加了警报，当一个或多个指定警报激活时，CodeDeploy 将重新部署上一个已知良好的修订。
   - 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 (p. 266)。

主题

- 指定有关存储在 Amazon S3 存储桶中的修订的信息 (p. 259)
- 指定有关存储在 GitHub 存储库中的修订的信息 (p. 260)

指定有关存储在 Amazon S3 存储桶中的修订的信息

如果您正在按照创建EC2/本地计算平台部署（控制台）(p. 258)，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：

      登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon S3 控制台：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，复制Link字段添加到修订位置框中的 CodeDeploy 内容。

   要将 ETag（文件校验和）指定为修订位置的一部分，请执行以下操作：

   - 如果 Link (链接) 字段值以 **`?versionId=`*`versionId`*** 结尾，请将 **`&etag=`** 和 ETag 添加到 Link (链接) 字段值的结尾。

- 如果 Link (链接) 字段值未指定版本 ID，请将 **?etag=** 和 ETag 添加到 Link (链接) 字段值的结尾。

  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。

指定有关存储在 GitHub 存储库中的修订的信息

如果您正在执行 创建EC2/本地计算平台部署（控制台）(p. 258) 中的步骤，请遵循这些步骤以添加有关存储在 GitHub 存储库中的应用程序修订的详细信息。

1. 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，请在另一个 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 与您的应用程序的 GitHub 进行交互。继续执行步骤 2。
   - 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 4。
   - 要创建与其他 GitHub 账户的连接，请在另一个 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 2。

2. 如果系统提示您登录 GitHub，则按照 Sign in 页面上的说明操作。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 Create deployment (创建部署) 页的 Repository name (存储库名称) 框中，输入包含修订的 GitHub 用户或组织名称，并后跟一个正斜杠 (/) 以及包含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 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 Your repositories (您的存储库) 中不可见，请使用 Search GitHub (搜索 GitHub) 框查找目标存储库名称和 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 仪表板。
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)

在您创建应用程序和修订 (在 Amazon ECS 部署中，这是 AppSpec 文件) 后：

调用创建部署命令，在命令中指定：

- 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
- 部署组名称。要查看部署组名称的列表，请调用列表部署组命令。
- 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，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。)

  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 部署修订模板，请参阅AppSpecAmazonLambda 部署 (p. 242)。有关示例修订，请参阅 AppSpec 文件示例AmazonLambda 部署 (p. 347)。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 (p. 266)。

创建AmazonLambda 计算平台部署 (CLI)

在您创建应用程序和修订 (在AmazonLambda 部署，这是 AppSpec 文件）：

调用创建部署命令，在命令中指定：

- 应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
- 部署组名称。要查看部署组名称的列表，请调用列表部署组命令。
- 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，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。)

  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。要查看部署配置列表，请调用列表部署配置命令。（如果未指定，CodeDeploy 将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 部署修订模板，请参阅AppSpecAmazonLambda 部署 (p. 242)。有关示例修订，请参阅 AppSpec 文件示例AmazonLambda 部署 (p. 347)。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息 (p. 266)。

创建EC2/本地计算平台部署(CLI)

使用Amazon CLI将修订部署到 EC2/本地计算平台上：

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，请立即继续到步骤 2。
   - 如果您希望从 GitHub 存储库部署修订，请首先完成将 CodeDeploy 应用程序 Connect 到 GitHub 存储库 (p. 264)中的步骤，然后继续到步骤 2。
2. 调用创建部署命令，在命令中指定：
   - `--application-name`：应用程序名称。要查看应用程序名称列表，请调用列出应用程序命令。
   - `--deployment-group-name`：Amazon EC2 部署组名称。要查看部署组名称的列表，请调用列表部署组命令。
   - `--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 服务器实例不支持 tar 和压缩 tar 归档文件格式（.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，CodeDeploy 将使用最新的版本。）
- `s3Location --> eTag`：（可选）修订的 ETag。(如果未指定 ETag，则 CodeDeploy 将跳过对象验证。)

对于存储在 GitHub 中的修订：

- `gitHubLocation --> repository`：分配到包含修订的存储库中的 GitHub 用户或组名，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。要查看部署配置列表，请调用列表部署配置命令。（如果未指定，CodeDeploy 将使用特定的默认部署配置。）
- `--ignore-application-stop-failures | --no-ignore-application-stop-failures`：(可选) 您是否希望实例的部署仍继续到`BeforeInstall`部署生命周期事件，如果`ApplicationStop`部署生命周期事件失败。
- `--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由最新部署安装的所有文件。选择当不是上一部署的一部分的文件显示在目标部署位置时，将发生什么情况。
- `--target-instances`：对于蓝/绿部署，属于蓝/绿部署中替换环境的实例的相关信息，包括一个或多个 Amazon EC2 Auto Scaling 组的名称，或者标签筛选键、类型以及用于标识 Amazon EC2 实例的值。

Note

使用此语法作为create-deployment调用，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
```

在 create-deployment 调用中使用此语法，可直接在命令行上指定有关 GitHub 中修订的信息：

```
--github-location repository=string,commitId=string
```

要获取有关已推送修订的信息，请调用列表应用程序修订命令。

要跟踪部署的状态，请参阅查看 CodeDeploy 部署详细信息（p. 266)。

创建部署命令参考

以下是命令结构和选项`create-deployment`命令。有关更多信息，请参阅 。创建部署引用Amazon CLI命令参考.

```
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 Connect 到 GitHub 存储库

在您可以首次使用从 GitHub 存储库部署应用程序之前，您可以使用Amazon CLI，您必须先授予代表您的 GitHub 账户与 GitHub 交互的权限。对于使用 CodeDeploy 控制台的每个应用程序，必须完成一次此步骤。

1. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
2. 选择应用程序。
3. 从 Application (应用程序) 中，选择要链接到您的 GitHub 用户账户的应用程序，然后选择 Deploy application (部署应用程序)。

   Note

   您没有创建部署。这是当前授权代表您的 GitHub 用户账户与 GitHub 交互的唯一方式。
4. 在 Repository type (存储库类型) 旁边，选择 My application revision is stored in GitHub (我的应用程序修订存储在 GitHub 中)。
5. 选择 Connect to GitHub (连接到 GitHub)。

   Note

   如果您看到 Connect to a different GitHub account 链接：
   您可能已针对应用程序授权 CodeDeploy 代表其他 GitHub 账户与 GitHub 进行交互。
   您可能已针对在 CodeDeploy 中链接到的所有应用程序撤消代表已 CodeDeploy 录 GitHub 账户与 GitHub 交互的授权。
   有关更多信息，请参阅使 GitHub CodeDeploy 中的应用程序进 (p. 48)。
6. 如果您尚未登录 GitHub，则按照 Sign in 页面上的说明操作。
7. 在 Authorize application 页上，选择 Authorize application。
8. 现在，CodeDeploy 已具有权限，请选择Cancel，然后继续完成创建EC2/本地计算平台部署 (CLI) (p. 262).

通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation

您可以使用Amazon CloudFormation通过 CodeDeploy 管理 Amazon ECS 蓝/绿部署。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。本主题介绍了由 CodeDeploy 管理的 Amazon ECS 蓝/绿部署与由Amazon CloudFormation.

如需使用分步指导，Amazon CloudFormation管理 Amazon ECS 蓝/绿部署，请参阅。通过 CodeDeploy 使用自动执行 ECS 蓝/绿部署Amazon CloudFormation中的Amazon CloudFormation用户指南.

> Note
>
> 使用管理 Amazon ECS 蓝/绿部署Amazon CloudFormation不在亚太地区（大阪）区域提供。

通过 CodeDeploy 和 Amazon ECS 蓝/绿部署之间的差异Amazon CloudFormation

这些区域有：Amazon CloudFormation堆栈模板对与 Amazon ECS 任务相关的资源和基础设施进行建模，还对部署的配置选项进行建模。因此，标准 Amazon ECS 蓝/绿部署与通过创建的蓝/绿部署与通过Amazon CloudFormation.

与标准 Amazon ECS 蓝/绿部署不同，您不用建模，也不用手动创建以下内容：

- 您不用通过指定一个唯一表示要部署的内容的名称，创建 Amazon CodeDeploy 应用程序。
- 您不用创建 Amazon CodeDeploy 部署组。
- 您不用指定应用程序规范文件（AppSpec 文件）。通常由 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	标准蓝/绿部署	通过 Amazon CloudFormation 进行蓝/绿部署
指定 Amazon ECS 群集、Amazon ECS 服务、Application Load Balancer 或 Network Load Balancer、生产侦听器、测试侦听器和两个目标组。	创建一个用于指定这些资源的 CodeDeploy 部署组。	创建 Amazon CloudFormation 模板对这些资源进行建模。
指定要部署的更改。	创建 CodeDeploy 应用程序。	创建一个用于指定容器映像的 Amazon CloudFormation 模板。
指定 Amazon ECS 任务定义、容器名称和容器端口。	创建一个用于指定这些资源的 AppSpec 文件。	创建 Amazon CloudFormation 模板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。	创建一个用于指定这些选项的 AppSpec 文件。	创建一个使用 `AWS::CodeDeploy::BlueGreen` 挂钩参数指定这些选项的 Amazon CloudFormation 模板。
CloudWatch 警报。	创建触发回滚的 CloudWatch 警报。	配置 CloudWatch 警报Amazon CloudFormation堆栈级别，触发回滚。
回滚/重新部署。	指定回滚和重新部署选项。	取消 Amazon CloudFormation 中的堆栈更新。

通过监控 Amazon ECS 蓝/绿部署Amazon CloudFormation

您可以通过监控蓝/绿部署Amazon CloudFormation和 CodeDeploy。有关通过监控的信息，Amazon CloudFormation，请参阅监控中的蓝/绿事件Amazon CloudFormation中的Amazon CloudFormation用户指南.

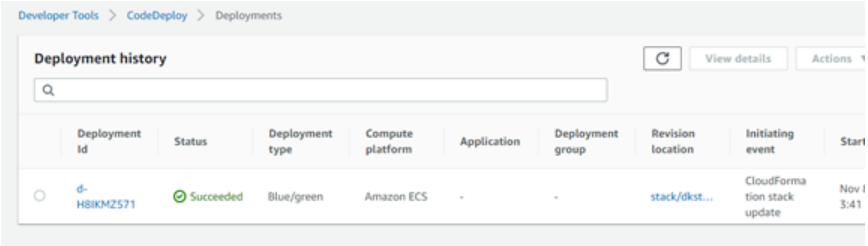
在 CodeDeploy 中查看蓝/绿部署的部署状态

1. 登录到Amazon Web Services Management Console并打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy.

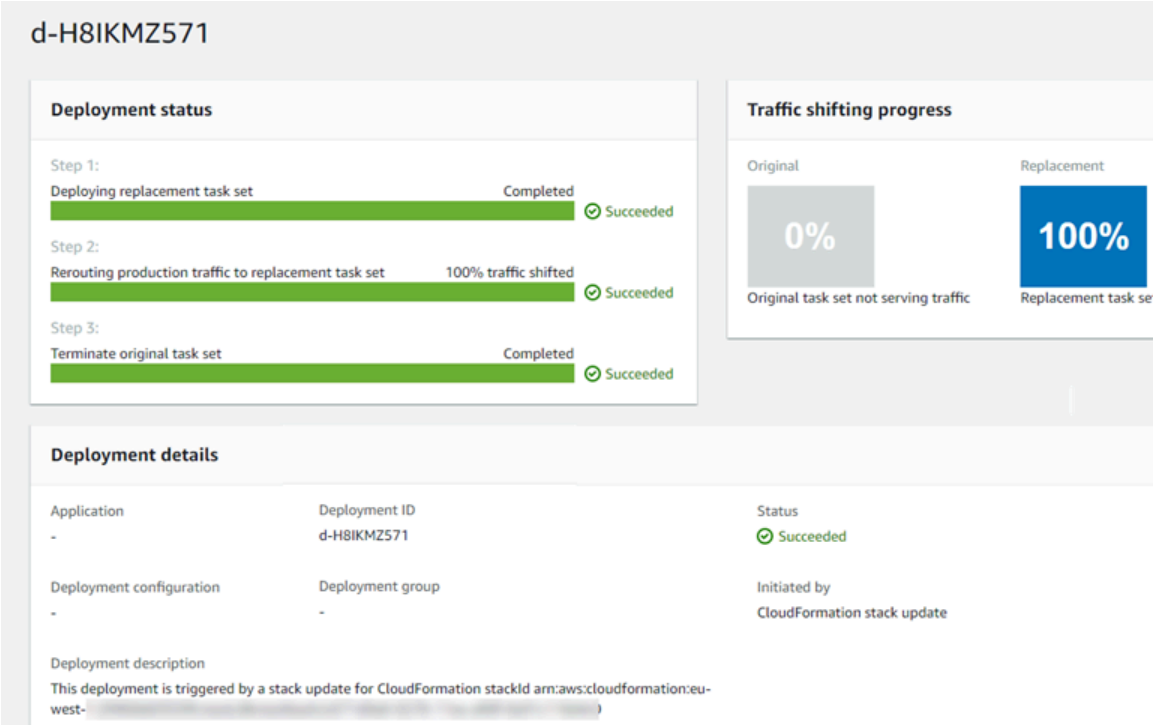
   Note

   使用您在开始使用 CodeDeploy (p. 26).

2. 在 Deployments (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：

   - 成功的部署将显示在 CodeDeploy 中，并显示该部署是由Amazon CloudFormation.
   - 如果要停止并回滚部署，则必须在 Amazon CloudFormation 中取消堆栈更新。

查看 CodeDeploy 部署详细信息

可以使用 CodeDeploy 控制台、Amazon CLI，或者使用 CodeDeploy API 查看有关与Amazonaccount.

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 服务器：`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`
- Windows Server：C:\ProgramData\Amazon\CodeDeploy <DEPLOYMENT-GROUP-ID> <DEPLOYMENT-ID>\ log\ 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 (p. 373)。

主题

- 查看部署详细信息（控制台） (p. 267)
- 查看部署详细信息 (CLI) (p. 267)

查看部署详细信息（控制台）

使用 CodeDeploy 控制台查看部署详细信息：

1. 登录到Amazon Web Services Management Console，然后打开 CodeDeploy 控制台，在https://console.aws.amazon.com/codedeploy.

   Note

   使用您在开始使用 CodeDeploy (p. 26).
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。在导航栏中的区域选择器中，选择区域和终端节点中的Amazon一般参考. CodeDeploy 仅在这些地区受支持。
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，请调用获取部署命令，并指定唯一部署标识符。要获取部署 ID，请调用列出部署命令。

要查看有关多个部署的详细信息，请调用批处理获取部署命令，在命令中指定多个唯一部署标识符。要获取部署 ID，请调用列出部署命令。

要查看部署 ID 的列表，请调用列出部署命令，在命令中指定：

- 与部署关联的应用程序的名称。要查看应用程序名称列表，请调用列出应用程序命令。
- 与部署关联的部署组的名称。要查看部署组名称的列表，请调用列表部署组命令。
- （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。）

查看代表部署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 Logs 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**
>
> 不支持日志Amazon弹性云服务器或亚马逊云服务器部署。只能为 EC2/本地部署创建这些文件。

主题

- 在 Amazon CloudWatch 控制台中查看日志文件数据 (p. 268)
- 查看实例上的日志文件 (p. 268)

在 Amazon CloudWatch 控制台中查看日志文件数据

在实例上安装 Amazon CloudWatch Logs 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日志数据。为简单起见，我们建议您使用 Amazon CloudWatch Logs 集中监控日志文件，而不是逐个实例查看这些文件。有关设置 Amazon CloudWatch Logs 代理的信息，请参阅查看代码在 CloudWatch Logs 控制台中部署日志。

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 服务器实例上的部署日志文件 (p. 268)
- 查看 Windows 服务器实例上的部署日志文件 (p. 269)

查看 Amazon Linux、RHEL 和 Ubuntu 服务器实例上的部署日志文件

在 Amazon Linux、RHEL 和 Ubuntu 服务器实例上，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 服务器实例上的部署日志，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	搜索上一条错误消息。[2] 在一词的前后使用一个空格**ERROR**。

命令	结果
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] 在您输入**/ ERROR** ，键入**n**为下一条错误消息。为上一条错误消息键入 **N**。 [2] 键入后**? ERROR** ，键入**n**为下一条错误消息键入**N**为上一条错误消息。	

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] 在您输入**/stderr**，键入**n**为下一条错误消息键入。为上一条错误消息键入 **N**。 [2] 键入后**?stderr**，键入**n**为下一条错误消息键入。为上一条错误消息键入 **N**。	

查看 Windows 服务器实例上的部署日志文件

CodeDeploy 代理日志文件：在 Windows 服务器实例上，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。

使用 CodeDeploy 停止部署

可以使用 CodeDeploy 控制台、Amazon CLI或 CodeDeploy API 停止与您的Amazonaccount.

> **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 (p. 23)。

您可以停止部署，也可以停止和回滚部署。

- 停止部署（控制台）(p. 270)
- 停止部署 (CLI) (p. 271)

> **Note**
>
> 如果您的部署是通过进行蓝/绿部署，Amazon CloudFormation，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. 登录到Amazon Web Services Management Console，然后在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   > **Note**
   >
   > 使用您在开始使用 CodeDeploy (p. 26)。

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。在导航栏中的区域选择器中，选择区域和终端节点中的Amazon一般参考。CodeDeploy 仅在这些地区受支持。

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 (p. 271)。

   > **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)

调用停止部署命令，并指定部署 ID。要查看部署 ID 的列表，请调用列表部署命令。

使用 CodeDeploy 重新部署和回滚部署

CodeDeploy 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 (p. 271)
- 手动回滚 (p. 271)
- 回滚和重新部署工作流程 (p. 271)
- 对现有内容进行回滚行为 (p. 272)

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**
>
> 每当部署自动回滚时，您都可使用 Amazon Simple Than(使部署自动回滚) 接收通知。有关信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications (p. 287)。

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 (p. 236)。

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 CodeDeploy 创建部署 (p. 254)。

> **Note**
>
> 如果您从部署组中删除某个实例，则 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。CodeDeploy 为此，将检查下面的清除文件：

`/opt/codedeploy-agent/deployment-root/deployment-instructions/`*`deployment-group-ID`*`-cleanup`文件（适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例）

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`*`deployment-group-ID`*`-cleanup`文件（适用于 Windows Server 实例）

如果 CodeDeploy 存在，则将在开始新部署之前，使用清除文件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)
```

清除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`文件包含逻辑，以便不重新创建`e.txt`和`f.txt`文件（如果它们已经存在），那么旧版本`e.txt`和`f.txt`将保持不变，无论何时 CodeDeploy 运行`c.bat`和`d.bat`在后续部署中。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。

对现有内容进行回滚行为

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）(p. 258)。如果使用Amazon CLI，请参阅创建EC2/本地计算平台部署(CLI) (p. 262)。

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	Succeeded	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	已失败	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)。 但是，您想在部署 1 中保留的文件已在部署 2 失败之前被删除，并且无法由Amazon CodeDeploy。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。

将应用程序部署到不同Amazon账户

组织通常会有多个 Amazon 账户用于不同用途（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关联，而另一个与生产环境关联）。

虽然您可能在不同账户中执行相关工作，CodeDeploy 部署组和所部署到的 Amazon EC2 实例严格绑定到创建它们时所使用的账户。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。

假设您有两个 Amazon 账户：开发账户和生产账户。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），它授予您对该帐户的临时访问权限。

第 1 步：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 Amazon S3 存储桶，将在其中存储生产账户的应用程序修订。想要了解有关信息，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。

第 2 步：向 Amazon S3 存储桶授予对生产账户的 IAM 实例配置文件的权限

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 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。

  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您刚刚在生产账户中创建的 IAM 实例配置文件。想要了解有关信息，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。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Cross-account permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::account-id:role/role-name"
         },
         "Action": [
            "s3:Get*",
            "s3:List*"
         ],
         "Resource": [
            "arn:aws:s3:::bucket-name/*"
         ]
      }
   ]
}
```

*account-id*表示您刚刚创建了 IAM 实例配置文件的生产账户的账户编号。

*####*表示您刚刚创建的 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。

第 3 步：创建资源和跨账户角色

在生产账户中：

- 使用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 EC2 实例配置文件、服务角色等）。
- 创建附加角色，即跨账户 IAM 角色，您开发账户中的用户可以代入该角色以在此生产账户中执行 CodeDeploy 操作。

  使用演练：委托跨越Amazon账户使用 IAM 角色作为指南来帮助您创建跨账户角色。您不应将演练中的示例权限添加到策略文档，而是至少应附加以下两个Amazon向角色提供了策略：

  - AmazonS3FullAccess：只有当 S3 存储桶位于开发账户中时才需要。提供对 Amazon S3 服务和资源具有完整访问权限的代入生产账户角色（修订存储在其中）。
  - AWSCodeDeployDeployerAccess：允许 IAM 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 托管策略为 CodeDeploy 任务授予权限的更多信息，请参阅AmazonCodeDeploy 托管（预定义）策略 (p. 305)。

  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 Management Console 来切换角色，您需要提供以下内容之一：
  >
  > - 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 切换角色，您需要提供以下内容：
  >
  > - 您将代入的跨账户角色的 ARN。

第 4 步：将应用程序修订上传到 Amazon S3 存储桶

在您创建 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 将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署） (p. 247)。

第 5 步：担任跨账户角色并部署应用程序

在开发账户中，您可以使用 Amazon CLI 或 Amazon Web Services Management Console 来代入跨账户角色并在生产账户中启动部署。

有关如何使用Amazon Web Services Management Console切换角色并启动部署，请参阅切换到角色 ()Amazon Web Services Management Console）和创建EC2/本地计算平台部署（控制台） (p. 258)。

有关如何使用Amazon CLI以代入跨账户角色并启动部署，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) (p. 262)。

有关通过Amazon STS，请参阅AssumeRole中的Amazon Security Token Service用户指南和确定角色中的Amazon CLI命令参考。

相关主题：

- CodeDeploy: 从开发帐户部署到生产帐户

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，您可以运行codedeploy-local命令，而无需指定任何选项，该命令位于包含 AppSpec 文件和要部署内容的目录中。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。

您可以使用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) 中的部署内容。

Prerequisites

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 (p. 141)。
- 安装代理的 1.0.1.1352 版本或更高版本。有关信息，请参阅 安 CodeDeploy (p. 151)。
- 如果您要部署 Amazon S3 存储桶或 GitHub 存储库中的内容，请预配置一个在 CodeDeploy 中使用的 IAM 用户。有关信息，请参阅 第 3 步：预配 IAM 用户 (p. 30)。
- 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 存储桶策略允许附加的 IAM 实例配置文件（其中包含 ARN）的任何 Amazon EC2 实例`arn:aws:iam::444455556666:role/CodeDeployDemo`从 Amazon S3 存储桶中的任意位置下载，该文件为`codedeploydemobucket`：

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

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

以下 Amazon S3 存储桶策略允许任何本地实例，其中包含 ARN 的关联 IAM 用户。`arn:aws:iam::444455556666:user/CodeDeployUser`从 Amazon S3 存储桶中的任意位置下载，该文件为`codedeploydemobucket`：

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::codedeploydemobucket/*",
            "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，请参阅第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)、为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CLI或 Amazon EC2 控制台） (p. 170)和为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CloudFormation模板) (p. 174)。
- 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 使用 CodeDeploy 的应用程序修订 (p. 239)。
- 对于其他配置选项，如果您希望使用默认值以外的其他值，请创建配置文件并放置在实例中 (`/etc/codedeploy-agent/conf/codedeployagent.yml`对于 Amazon Linux、RHEL 或 Ubuntu Server 实例，或`C:\ProgramData\Amazon\CodeDeploy\conf.yml`）。有关信息，请参阅 CodeDeploy 代理配置参考 (p. 350)。

  Note

  如果您在 Amazon Linux、RHEL 或 Ubuntu Server 实例上使用配置文件，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

在您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://mybucket/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 。fileExistsBehavior在Amazon CodeDeployAPI 参考。

-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。如果您包含DownloadBundle和安装列表`--events`选项，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 的 “hooks” 部分 (p. 335)。

-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 服务器）。有关更多信息，请参阅 CodeDeploy 代理配置参考 (p. 350)。

-A、— 应用程序规范文件名

AppSpec 文件的名称（如果您为其指定了默认名称以外的名称）。默认情况下，AppSpec 文件称为`appspec.yml`或者`appspec.yaml`。

-h、--help

显示帮助内容的摘要。

-v, --version

显示工具的版本号。

Examples

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 部署数据包：

```
codedeploy-local --bundle-location s3://mybucket/bundle.tgz --type tgz
```

```
codedeploy-local --bundle-location s3://mybucket/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 CodeDeploy 中的部署

监控是维护 CodeDeploy 和您的Amazon解决方案。您应从 Amazon 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。不过，在开始监控 CodeDeploy 之前，您应制定监控计划并在计划中回答下列问题：

- 您的监控目标是什么？
- 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，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 CodeDeploy 性能的基准。监控 CodeDeploy 时，应将历史监控数据存储下来以便与当前性能数据进行比较，确定正常性能模式和异常性能表现，并设计出问题解决方法。

例如，如果您使用的是 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更改实例设置或 AppSpec 文件。

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为您提供了各种可用于监控 CodeDeploy 的工具。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化监控工具来监控 CodeDeploy 并在出现错误时报告：

- Amazon CloudWatch 警报 – 按您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。操作是向 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略发送的通知。CloudWatch 警报将不会调用操作，因为这些操作处于特定状态，该状态必须改变并在指定数量的时间段内一直保持。有关更多信息，请参阅 Monitoring Deployments with Amazon CloudWatch Tools (p. 283)。

  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 (p. 283)。有关向您的 CodeDeploy 操作添加 CloudWatch 警报监控的信息，请参阅。使用 CodeDeploy 创建应用程序 (p. 212)、使用 CodeDeploy 创建部署组 (p. 225)，或者使用 CodeDeploy 更改部署组设置 (p. 235)。

- Amazon CloudWatch Logs— 监控、存储和访问来自Amazon CloudTrail或其他来源. 有关更多信息，请参阅 。监控日志文件中的Amazon CloudWatch 用户指南。

有关使用 CloudWatch 控制台查看 CodeDeploy 日志的信息，请参阅。查看代码在 CloudWatch Logs 控制台中部署日志。

- Amazon CloudWatch Events – 匹配事件并将事件传送到一个或多个目标函数或流来进行更改、捕获状态信息和采取纠正措施。有关更多信息，请参阅 。什么是 Amazon CloudWatch Events中的Amazon CloudWatch 用户指南。

  有关在 CodeDeploy 操作中使用 CloudWatch Events 的信息，请参阅。使用 Amazon CloudWatch Events 监控部署 (p. 284)。

- Amazon CloudTrail日志监控— 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 CloudWatch Logs 对它们进行实时监控，使用 Java 编写日志处理应用程序，以及验证您的日志文件是否在被 CloudTrail 交付后未发生更改。有关更多信息，请参阅 。使用 CloudTrail 日志文件中的Amazon CloudTrail用户指南。

  有关将 CloudTrail 与 CodeDeploy 结合使用的信息，请参阅。Monitoring Deployments (p. 285)。

- 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 SMS 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 CodeDeploy 设置 Amazon SNS 通知的信息，请参阅。Monitoring Deployments with Amazon SNS Event Notifications (p. 287)。

手动监控工具

监控 CodeDeploy 的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。CodeDeploy、CloudWatch 和其他Amazon控制台控制面板提供您的Amazon环境。建议您还要查看有关 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：
  - 当前警报和状态
  - 警报和资源的图表
  - 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
  - 搜索并浏览您所有的 Amazon 资源指标
  - 创建和编辑警报以接收有关问题的通知

主题

- Monitoring Deployments with Amazon CloudWatch Tools (p. 283)
- Monitoring Deployments (p. 285)
- Monitoring Deployments with Amazon SNS Event Notifications (p. 287)

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 Events、CloudWatch 警报和 Amazon CloudWatch Logs。

通过查看 CodeDeploy 代理和部署所创建的日志可以帮助您排查部署失败的原因。作为每次查看一个实例的 CodeDeploy 日志的替代方法，您可以使用 CloudWatch Logs 监控一个中心位置的所有日志。

有关使用 CloudWatch 控制台查看 CodeDeploy 日志的信息，请参阅查看代码在 CloudWatch Logs 控制台中部署日志.

有关使用 CloudWatch 警报和 CloudWatch Events 监控 CodeDeploy 部署的信息，请参阅以下主题。

主题

- 使用 CodeDeploy 中的 CloudWatch 警报监控部署 (p. 283)
- 使用 Amazon CloudWatch Events 监控部署 (p. 284)

使用 CodeDeploy 中的 CloudWatch 警报监控部署

您可以在 CodeDeploy 操作中为当前使用的实例或 Amazon EC2 自动扩展组创建 CloudWatch 警报。警报按指定的时间段监控单个指标，并根据相对于给定阈值的指标值在多个时间段执行一项或多项操作。CloudWatch 警报会在动作状态发生变化时调用动作（例如，从`OK`到`ALARM`）。

通过使用本机 CloudWatch 警报功能，您可以指定在部署中所用实例失败时执行的 CloudWatch 支持的任何操作，如发送 Amazon SNS 通知或停止、终止、重启或恢复实例。对于 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。

您可以为 1 个 CodeDeploy 组关联多达 10 个 CloudWatch 警报。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，请参阅创建 Amazon CloudWatch 警报中的Amazon CloudWatch 用户指南.

有关在 CodeDeploy 中为部署组关联 CloudWatch 警报的信息，请参阅使用 CodeDeploy 创建部署组 (p. 225)和使用 CodeDeploy 更改部署组设置 (p. 235).

主题

- 向 CodeDeploy 服务角色授予 CloudWatch 权限 (p. 283)

向 CodeDeploy 服务角色授予 CloudWatch 权限

必须先向您在 CodeDeploy 操作中使用的服务角色授予访问 CloudWatch 资源的权限，然后您才能对部署使用 CloudWatch 警报监控。

向服务角色授予 CloudWatch 权限

1. 登录 Amazon Web Services Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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：

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

8. 选择 Apply Policy。

使用 Amazon CloudWatch Events 监控部署

您可以在 CodeDeploy 操作中使用 Amazon CloudWatch Events 检测实例或部署状态的更改（称为 “事件”）。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 Events 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 Events 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 (`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。
- 当发生指定的部署或实例事件时，使用 CloudWatch 警报操作停止、终止、重启或恢复 Amazon EC2 实例。

本主题的其余内容介绍为 CodeDeploy 创建 CloudWatch Events 规则的基本步骤。然而，在创建用于 CodeDeploy 操作的事件规则之前，您应该执行以下操作：

- 完成 CloudWatch 事件先决条件。想要了解有关信息，请参阅Amazon CloudWatch Events 先决条件.
- 熟悉 CloudWatch Events 中的事件、规则和目标。有关更多信息，请参阅 。什么是 Amazon CloudWatch Events？和新的 CloudWatch 事件 — 跟踪和响应对您的Amazonresources.
- 创建将在您的事件规则中使用的目标。

要为 CodeDeploy 创建 CloudWatch 事件规则，请执行以下操作：

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 Events。
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。
- 要设置仅适用于实例的规则，请选择 Specific detail type，然后选择 CodeDeploy Instance State-change Notification。
- 要设置仅适用于部署的规则，请选择 Specific detail type，然后选择 CodeDeploy Deployment State-change Notification。

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	READY
QUEUED	SUCCESS
READY	
SUCCESS	

6. 指定规则适用的 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。
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。

使用监视部署Amazon CloudTrail

CodeDeploy 与 CloudTrail 集成，后者是一项服务，可用于捕获由 CodeDeploy 或代表 CodeDeploy 发出的 API 调用。Amazon账户，并将日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 可捕获来自 CodeDeploy 控制台的 API 调用，从 CodeDeploy 命令通过Amazon CLI，或直接从 CodeDeploy API 中获取。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，包括如何配置和启用 CloudTrail，请参阅Amazon CloudTrail用户指南。

代码在 CloudTrail 中部署信息

CloudTrail 您在Amazon帐户，则将在日志文件中跟踪对 CodeDeploy 操作执行的 API 调用。CodeDeploy 记录与其他Amazon日志文件中的服务记录。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 操作都会记录并正式记载到Amazon CodeDeploy命令行参考和Amazon CodeDeployAPI参考。例如，对于创建部署、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请求是由根或 IAM 用户凭证发出，通过某个角色或联合用户的临时安全凭证发出，还是由其他 Amazon 服务发出。有关更多信息，请参阅 。userIdentity字段中的CloudTrail 事件参考。

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。

您可以让 CloudTrail 在传送新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅为 CloudTrail 配置 Amazon SNS 通知。

您还可以将 CodeDeploy 日志文件从多个Amazon区域和多个Amazon账户存储到单个 Amazon S3 存储桶。有关更多信息，请参阅 。从多个区域中接收 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

您可以将触发器添加到 CodeDeploy 部署组，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 Amazon SNS 主题的接收人。

您可以在 SMS 消息或电子邮件中接收 CodeDeploy 事件的通知。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。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 (p. 294)。

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。
- 您是系统管理员，需要知道失败的实例数量，以便监控 Amazon EC2 队列的运行状况。
- 您是经理，需要有关部署和实例事件的计数一览表，您可以通过将不同类型的通知传送到您的桌面电子邮件客户端的文件夹中的筛选规则来获取这些信息。

对于以下任意事件类型，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 • 失败 • Started • Stopped • 回滚 • 准备工作 [1] • 所有部署事件	• 成功 • 失败 • Started • 准备工作 [1] • 所有实例事件
[1] 仅适用于蓝/绿部署。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 CodeDeploy 中使用部署 (p. 253)。	

主题

- 向 CodeDeploy 服务角色授予 Amazon SNS 权限 (p. 288)

- 为 CodeDeploy 事件创建触发器 (p. 288)
- 编辑 CodeDeploy 部署组中的触发器 (p. 293)
- 从 CodeDeploy 部署组中删除触发器 (p. 294)
- CodeDeploy 触发器的 JSON 数据格式 (p. 294)

向 CodeDeploy 服务角色授予 Amazon SNS 权限

必须先向您在 CodeDeploy 操作中使用的服务角色授予访问 Amazon SNS 资源的权限，然后您的触发器才能生成通知。

向 Amazon SNS 授予权限

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：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：

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

8. 选择 Apply Policy。

为 CodeDeploy 事件创建触发器

您可以创建一个触发器 Simple Notification Service (Amazon SNS) 为Amazon CodeDeploy部署或实例事件。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 SMS 消息或电子邮件）接收通知。Amazon SNS 提供了多种订阅主题的方法。

创建触发器之前，您必须设置触发器指向的 Amazon SNS 主题。想要了解有关信息，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，并采用这样的格式`Topic-group-us-west-3-deploy-fail`或者`Topic-group-project-2-instance-stop`。

您也必须向 Amazon SNS 授予对 CodeDeploy 服务角色的权限，然后才能为您的触发器发送通知。有关信息，请参阅 向 CodeDeploy 服务角色授予 Amazon SNS 权限 (p. 288)。

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 事件的通知（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 Amazon Web Services Management Console中，打开 Amazon CodeDeploy 控制台。
2. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 Edit。
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. In事件下，选择将触发 Amazon SNS 主题发送通知的事件类型。
   c. InAmazon 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 Create trigger。CodeDeploy 将发送一条测试通知，以确认您已正确配置 CodeDeploy 和 Amazon SNS 主题之间的访问权限。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 事件的通知 (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 文件以配置部署组，然后运行创建部署组命令`--cli-input-json`选项。

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：

```
{
    "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。

   在使用 create-deployment-group 命令时，您必须至少为以下参数提供值：

   - `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`：在您的帐户中为 CodeDeploy 设置的现有服务角色的 ARN。有关信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

   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`：您创建的用于与触发器关联的 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 (仅适用于蓝/绿部署中的替代实例)
     - InstanceStart
     - InstanceSuccess
     - InstanceFailure
     - InstanceReady (仅适用于蓝/绿部署中的替代实例)

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

使用Amazon CLI将 CodeDeploy 事件的触发器添加到现有部署组，创建 JSON 文件以更新部署组，然后运行更新部署组命令`--cli-input-json`选项。

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，请参阅 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. 以 JSON 文件格式保存更新，然后运行更新部署组命令--cli-input-json选项。请务必包含 --current-deployment-group-name 选项并将 *filename* 替换为您的 JSON 文件的名称：

   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。

编辑 CodeDeploy 部署组中的触发器

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)

使用Amazon CLI在更新部署组时更改 CodeDeploy 事件的触发器详细信息，创建 JSON 文件以定义对部署组属性的更改，然后运行更新部署组命令`--cli-input-json`选项。

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，将 *application* 和 *deployment-group* 替换为您的应用程序和部署组的名称：

```
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，请参阅 TriggerConfig。
4. 以 JSON 文件格式保存更新，然后运行更新部署组命令`--cli-input-json`选项。请务必包含 `--current-deployment-group-name` 选项并将 *`filename`* 替换为您的 JSON 文件的名称：

   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（控制台）中删除触发器

1. 登录到Amazon Web Services Management Console并在处打开 CodeDeploy 控制台https://console.aws.amazon.com/codedeploy。

   Note

   使用您在开始使用 CodeDeploy (p. 26)。
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 Edit。
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 (CLI) 中删除触发器

要使用 CLI 删除触发器，请调用更新部署组命令，使用空的触发器配置参数，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，请调用列出应用程序命令。
- 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，请调用列表部署组命令。

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。

> **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 向 Amazon SQS 队列发送消息的信息，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"
    }
}
```

中的安全性AmazonCodeDeploy

Amazon的云安全性的优先级最高。作为Amazon客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。责任共担模型将其描述为云的 安全性和云中 的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。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。要了解适用于的合规性计划AmazonCodeDeploy, 请参阅Amazon合规性计划范围内的服务.
- 云中的安全性 - 您的责任由您使用的Amazon服务决定。你还要对其他因素负责、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。

此文档将帮助您了解如何在使用 CodeDeploy 时应用责任共担模型。以下主题说明如何配置 CodeDeploy 以实现您的安全性和合规性目标。您还将了解如何使用其他Amazon帮助您监控和保护您的 CodeDeploy 资源的服务。

主题

- 中的数据保护AmazonCodeDeploy (p. 296)
- 适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297)
- CodeDeploy 中的记录和监控 (p. 318)
- 的合规性验证AmazonCodeDeploy (p. 319)
- 中的故障恢复能力AmazonCodeDeploy (p. 320)
- 中的基础设施安全性AmazonCodeDeploy (p. 320)

中的数据保护AmazonCodeDeploy

这些区域有：Amazon 责任共担模式适用于中的数据保护AmazonCodeDeploy. 如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的Amazon服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与Amazon资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用Amazon加密解决方案以及Amazon服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。
- 如果在通过命令行界面或 API 访问Amazon时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 终端节点。有关可用的 FIPS 终端节点的更多信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括当您使用 CodeDeploy 或其他时间Amazon使用控制台、API、Amazon CLI，或者Amazon开发工具包。您在用于名称的标签或自由格式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，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

互联网络流量隐私

CodeDeploy 是一种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，基于主机的代理可使用 TLS 与 CodeDeploy 通信。

当前，从代理到服务的通信需要出站互联网连接，以便代理可以与公共 CodeDeploy 和 Amazon S3 服务终端节点进行通信。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。

CodeDeploy 代理支持 HTTP 代理。

由提供支持的 Amazon VPC 终端节点Amazon PrivateLink，在某些地区可用于 CodeDeploy。有关详细信息，请参见将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用 (p. 354)。

> **Note**
>
> 仅当您部署到 Amazon EC2/本地计算平台时，才需要 CodeDeploy 代理。使用 Amazon ECS 的部署不需要代理或Amazon Lambda计算平台。

静态加密

CodeDeploy 中不存储客户代码。作为部署服务，CodeDeploy 会向 EC2 实例或本地服务器上运行的 CodeDeploy 代理分派命令。然后，CodeDeploy 代理会使用 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都存储在 Amazon DynamoDB 中，并使用Amazon 拥有的密钥，由 CodeDeploy 拥有和管理。有关更多信息，请参阅 。Amazon 拥有的密钥s.

传输中加密

CodeDeploy 代理通过端口 443 启动与 CodeDeploy 的所有通信。代理会轮询 CodeDeploy 并侦听命令。CodeDeploy 代理是开源的。所有服务到服务和客户端到服务的通信都使用 TLS 在传输过程中进行加密。这可以保护在 CodeDeploy 和 Amazon S3 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Amazon 拥有的密钥，由 CodeDeploy 拥有和管理。有关更多信息，请参阅 。Amazon 拥有的密钥s.

适用于 Amazon CodeDeploy 的 Identity and Access Management

Amazon Identity and Access Management (IAM) 是一种 Amazon 服务，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制哪些人可以已验证（已登录）和授权（有权限）可以使用 CodeDeploy 资源。IAM 是一个可以免费使用的 Amazon 服务。

主题

- Audience (p. 298)
- 使用身份进行身份验证 (p. 298)
- 使用策略管理访问 (p. 299)
- Amazon CodeDeploy 如何与 IAM 协同工作 (p. 300)

- Amazon CodeDeploy 基于身份的策略示例 (p. 303)
- 对 Amazon CodeDeploy 身份和访问进行故障排除 (p. 311)
- CodeDeploy 权限参考 (p. 313)

Audience

如何使用Amazon Identity and Access Management(IAM) 因您可以在 CodeDeploy 中执行的操作而异。

服务用户— 如果您使用 CodeDeploy 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 CodeDeploy 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 CodeDeploy 中的功能，请参阅对 Amazon CodeDeploy 身份和访问进行故障排除 (p. 311).

服务管理员— 如果您在公司负责管理 CodeDeploy 资源，则您可能具有的完全访问权限。您有责任确定您的员工应访问哪些 CodeDeploy 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 CodeDeploy 搭配使用的更多信息，Amazon CodeDeploy 如何与 IAM 协同工作 (p. 300).

IAM 管理员— 如果您是 IAM 管理员，您可能希望了解有关您可以如何编写策略以管理对 CodeDeploy 的访问的详细信息。要查看您可在 IAM 中使用的基于身份的 CodeDeploy 策略示例，请参阅。Amazon CodeDeploy 基于身份的策略示例 (p. 303).

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。有关使用 Amazon Web Services Management Console 登录的更多信息，请参阅 IAM 用户指南中的以 Amazon Web Services Management Console IAM 用户或根用户身份登录。

您必须作为Amazon Web Services 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到 Amazon）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其他公司的凭证访问 Amazon 时，您间接地代入了角色。

要直接登录到 Amazon Web Services Management Console，请将密码与根用户电子邮件地址或 IAM 用户名一起使用。您可以使用根用户或 IAM 用户访问密钥以编程方式访问Amazon。Amazon提供了开发工具包和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。使用签名版本 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其他安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您首次创建Amazon Web Services 账户时，最初使用的是一个对账户中所有Amazon服务和资源有完全访问权限的单点登录身份。此身份称为 Amazon Web Services 账户 根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，请遵循仅使用根用户创建您的第一个 IAM 用户的最佳实践。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可能具有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理

IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组 是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 Amazon Directory Service、您的企业用户目录或 Web 身份提供商的现有身份。这些用户被称为联合身份用户。在通过身份提供商请求访问权限时，Amazon 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信委托人）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon 服务使用其他 Amazon 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的委托人的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 委托人权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为委托人。策略向委托人授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅服务授权参考中的 Amazon CodeDeploy 的操作、资源和条件键。
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向Amazon服务委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 Amazon 服务关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将Amazon角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南 中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 IAM 身份或Amazon资源，以便控制Amazon中的访问。策略是Amazon中的对象；在与标识或资源相关联时，策略定义它们的权限。您可以通过 root 用户或 IAM 用户身份登录，也可以代入 IAM 角色。随后，当您提出请求时，Amazon会评估相关的基于身份或基于资源的策略。策略中的权限确

定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅。JSON 策略概述中的IAM 用户指南.

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人 可以对什么资源执行操作，以及在什么 条件 下执行。

每个 IAM 实体（用户或角色）最初没有任何权限。换言之，默认情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅创建 IAM 策略中的IAM 用户指南.

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是可以附加到中的多个用户、组和角色的独立策略。Amazon Web Services 账户. 托管策略包括 Amazon 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管策略与内联策略之间进行选择。

其他策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南 中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)— SCP 是指定中的组织或组织单位 (OU) 的最大权限的 JSON 策略。Amazon Organizations.Amazon Organizations是用于分组和集中管理多个服务Amazon Web Services 账户你的企业拥有的。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体的权限，包括每个Amazon Web Services 账户根用户。有关 Organizations 和 SCP 的更多信息，请参阅《Amazon Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 。会话策略中的IAM 用户指南.

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Amazon如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon CodeDeploy 如何与 IAM 协同工作

在使用 IAM 管理对 CodeDeploy 的访问之前，您应了解哪些 IAM 功能可与 CodeDeploy 协同工作。有关更多信息，请参阅 。Amazon使用 IAM 的服务中的IAM 用户指南.

主题

- CodeDeploy 基于身份的策略 (p. 301)
- 基于资源的策略 CodeDeploy (p. 303)
- 基于 CodeDeploy 标签的授权 (p. 303)
- IAM CodeDeploy 角色 (p. 303)

CodeDeploy 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。CodeDeploy 支持操作、资源和条件键。有关您在 JSON 策略中使用的元素的信息，请参阅。IAM JSON 策略元素参考中的IAM 用户指南.

Actions

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人 可以对什么资源执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

CodeDeploy 中的策略操作使用`codedeploy:`操作前缀。例如，`codedeploy:GetApplication` 权限授予用户执行 `GetApplication` 操作的权限。策略语句必须包含 `Action` 或 `NotAction` 元素。CodeDeploy 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

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 操作的列表，请参阅。定义的操作Amazon CodeDeploy中的IAM 用户指南.

有关列出所有 CodeDeploy API 操作及其适用的资源的表，请参阅。CodeDeploy 权限参考 (p. 313).

Resources

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个委托人 可以对什么资源执行操作，以及在什么 条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon Resource Name(ARN)指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

例如，您可以使用某个部署组 (*myDeploymentGroup*) 的 ARN 在语句中指定该规则，如下所示：

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup/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 ARN，请在 `Resource` 元素中使用通配符 (*)，如下所示：

```
"Resource": "*"
```

有些 CodeDeploy API 操作接受多个资源（例如，`BatchGetDeploymentGroups`)。要在单个语句中指定多种资源，请使用逗号将它们隔开，如下所示：

```
"Resource": ["arn1", "arn2"]
```

CodeDeploy 提供一组操作来处理 CodeDeploy 资源。有关可用操作的列表，请参阅 CodeDeploy 权限参考 (p. 313)。

有关 CodeDeploy 资源类型及其 ARN 的列表，请参阅。定义的资源Amazon CodeDeploy中的IAM 用户指南. 有关可指定每个资源的 ARN 的操作的信息，请参阅。定义的操作Amazon CodeDeploy.

CodeDeploy 资源和操作

在 CodeDeploy 中，主要资源是部署组。在策略中，您可以使用 Amazon 资源名称 (ARN) 标识策略应用到的资源。CodeDeploy 支持可与部署组一起使用的其他资源（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相关联的唯一 ARN。有关更多信息，请参阅 。Amazon 资源名称 (ARN)中的Amazon Web Services 一般参考.

资源类型	ARN 格式
部署组	`arn:aws:codedeploy:`*region*`:`*account-id*`:deploymentgroup/`*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:*`
指定账户在指定区域拥有的所有 CodeDeploy 资源	`arn:aws:codedeploy:`*region*`:`*account-id*`:*`

> **Note**
>
> Amazon 中的大多数服务将 ARN 中的冒号 (:) 或正斜杠 (/) 视为相同的字符。不过，CodeDeploy 在资源模式和规则中使用精确匹配。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键，但支持使用某些全局条件键。有关更多信息，请参阅 。Amazon全局条件上下文键中的IAM 用户指南.

Examples

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 (p. 303).

基于资源的策略 CodeDeploy

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示例，请参阅。使用 的基于资源的策略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。

IAM CodeDeploy 角色

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM 角色您的实体Amazon具有特定权限的账户。

将临时凭证用于 CodeDeploy

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 Amazon STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

CodeDeploy 支持使用临时凭证。

服务相关角色

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 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

CodeDeploy 支持服务角色。

在 CodeDeploy 中选择 IAM 角色

在 CodeDeploy 中创建部署组资源时，您必须选择一个角色以允许 CodeDeploy 代表您访问 Amazon EC2。如果您之前已经创建了一个服务角色或服务相关角色，CodeDeploy 会为您提供一个角色列表供您选择。选择一个允许访问以启动和停止 EC2 实例的角色很重要。

Amazon CodeDeploy 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 CodeDeploy 资源的权限。它们还无法使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 执行任务。IAM 管理员必须创建适当 IAM 策

略，为用户和角色授予权限，以便对他们所需的指定资源执行 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 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 CodeDeploy 中，基于身份的策略用于管理对与部署过程相关的多种权限。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	Capabilities
所有	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 Delete resources
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 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。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
```

```
        ]
      }
    ]
}
```

主题

- AmazonCodeDeploy 托管（预定义）策略 (p. 305)
- 客户托管的策略示例 (p. 309)
- 策略最佳实践 (p. 310)
- 使用 CodeDeploy 控制台 (p. 310)
- 允许用户查看他们自己的权限 (p. 311)

AmazonCodeDeploy 托管（预定义）策略

Amazon通过提供由Amazon创建和管理的独立 IAM 策略来满足许多常用案例的要求。这些 Amazon 托管策略授予常用案例的必要权限，这样您不需要调查需要哪些权限。有关更多信息，请参阅 IAM 用户指南.中的 Amazon 托管式策略

主题

- 的列表AmazonCodeDeploy 的托管策略 (p. 305)
- CodeDeploy 将更新到Amazon托管策略 (p. 307)
- CodeDeploy 托管策略和通知 (p. 307)

的列表AmazonCodeDeploy 的托管策略

以下Amazon托管策略（您可以将它们附加到您账户中的用户）特定于 CodeDeploy：

- `AWSCodeDeployFullAccess`：授予对 CodeDeploy 的完全访问权限

  > **Note**
  >
  > AWSCODEDeploy FullAccess 不提供对部署您的应用程序所需的其他服务中的操作的权限（如 Amazon EC2 和 Amazon S3），仅提供对 CodeDeploy 特定的操作的权限。

- `AWSCodeDeployDeployerAccess`：向 IAM 用户授予注册和部署修订的访问权限。

- `AWSCodeDeployReadOnlyAccess`：授予对 CodeDeploy 的只读访问权限

- `AWSCodeDeployRole`：允许 CodeDeploy：
  - 阅读您实例上的标签或通过 Amazon EC2 Auto Scaling 组名称来识别您的 Amazon EC2 实例。
  - 读取、创建、更新和删除 Amazon EC2 Auto Scaling 组、生命周期挂钩、扩展策略和暖池功能
  - 向 Amazon SNS 主题发布信息
  - 检索有关 Amazon CloudWatch 警报的信息
  - 读取和更新 Elastic Load Balancing 服务中的资源

  该策略包含以下代码：

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "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:DescribeInstanceHealth",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- `AWSCodeDeployRoleForLambda`：授予 CodeDeploy 访问的权限Amazon Lambda以及部署所需的任何其他资源。

- `AWSCodeDeployRoleForECS`：授予 CodeDeploy 权限，以访问 Amazon ECS 和部署所需的任何其他资源。

- `AWSCodeDeployRoleForECSLimited`：授予 CodeDeploy 权限以访问 Amazon ECS 和部署所需的任何其他资源，以下情况除外：
  - 在`hooks`在 AppSpec 文件的部分中，只有名称以开头的 Lambda 函数`CodeDeployHook_`可以使用。有关更多信息，请参阅Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。
  - 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 。对象标签.

部署过程某些方面的权限已授给两个代表 CodeDeploy 行事的其他角色类型，而不是 IAM 用户：

- IAM 实例配置文件：您附加到 Amazon EC2 实例的 IAM 角色。此配置文件包括访问用于存储应用程序的 Amazon S3 存储桶或 GitHub 存储库所需的权限。有关更多信息，请参阅第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。
- 服务角色：向向授予权限的 IAM 角色Amazon服务所以它可以访问Amazon资源的费用。您附加到服务角色的策略决定了哪些策略Amazon服务可以访问的资源以及它可以使用这些资源执行的操作。对于 CodeDeploy，服务角色用于以下内容：
  - 读取已应用于实例的标签或与实例关联的 Amazon EC2 Auto Scaling 组名称。这使 CodeDeploy 能够识别可以将应用程序部署到的实例。
  - 要对实例执行操作，请使用 Amazon EC2 Auto Scaling 组和 Elastic Load Balancing 器负载均衡器。
  - 向 Amazon SNS 主题发布信息，以便在发生指定的部署或实例事件时可以发送通知。
  - 检索有关 CloudWatch 警报的信息，以便为部署设置警报监控。

  有关更多信息，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。

您还可以创建自定义 IAM 策略，以授予对 CodeDeploy 操作和资源的权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

CodeDeploy 将更新到Amazon托管策略

查看有关更新的详细信息AmazonCodeDeploy 的托管策略（从该服务开始跟踪这些更改起）。要获得有关此页面更改的自动提示，请订阅 CodeDeploy 上的 RSS 源。文档历史记录 (p. 388).

更改	描述	日期
`AWSCodeDeployRole`托管策略 — 对现有策略的更新	添加了`autoscaling:PutWarmPool`支持的行动向 Amazon EC2 Auto Scaling 组添加温池适用于蓝/绿部署。 删除了不必要的重复操作。	2021 年 5 月 18 日
CodeDeploy 开启跟踪更改	CodeDeploy 为其开启了跟踪更改Amazon托管策略。	2021 年 5 月 18 日

CodeDeploy 托管策略和通知

CodeDeploy 支持通知，以使用户了解部署的重要更改。CodeDeploy 的托管策略包含通知功能的策略语句。有关更多信息，请参阅什么是通知？。

完全访问托管策略中的通知的权限

`AWSCodeDeployFullAccess` 托管策略包含以下语句，以允许对通知进行完全访问。已应用此托管策略的用户还可以创建和管理 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" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*"}
        }
    },
    {
        "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": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    }
```

只读托管策略中的通知的权限

`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" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules"
        ],
        "Resource": "*"
    }
```

有关更多信息，请参阅 。AWS CodeStar 通知的身份和访问管理.

客户托管的策略示例

本节的用户策略示例介绍如何授予各 CodeDeploy 操作的权限。当您使用 CodeDeploy API 时，可以使用这些策略。AmazonSDK，或者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 (p. 310)。

> **Note**
>
> 所有示例都使用美国西部 (俄勒冈) 区域 (`us-west-2`) 并且包含虚构的账户 ID。

示例

- 示例 1：允许用户在单个区域中执行 CodeDeploy 操作 (p. 309)
- 示例 2：允许用户为单个应用程序注册修订 (p. 309)
- 示例 3：允许用户为单个部署组创建部署 (p. 309)

示例 1：允许用户在单个区域中执行 CodeDeploy 操作

以下示例授予在中执行 CodeDeploy 操作的权限。**`us-west-2`**仅限地区：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:*"
      ]
    }
  ]
}
```

示例 2：允许用户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：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:application:Test*"
      ]
    }
  ]
}
```

示例 3：允许用户为单个部署组创建部署

以下示例允许指定用户为与名为 **`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。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:444455556666:application:WordPress_App"
      ]
    }
  ]
}
```

策略最佳实践

基于身份的策略非常强大。它们确定某个人是否可以创建、访问或删除您账户中的 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，请遵循以下准则和建议：

- 开始使用Amazon托管策略— 要快速开始使用 CodeDeploy，请使用Amazon为您的员工授予他们所需的权限的托管策略。这些策略已在您的账户中提供，并由 Amazon 维护和更新。有关更多信息，请参阅 IAM 用户指南中的开始使用 Amazon 托管策略中的权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其他权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最低权限。
- 为敏感操作启用 MFA – 为了提高安全性，要求 IAM 用户使用多重验证 (MFA) 访问敏感资源或 API 操作。有关更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 。IAM JSON 策略元素：Condition中的IAM 用户指南.

使用 CodeDeploy 控制台

如果您使用 CodeDeploy 控制台，则必须具有一组最小权限，以允许您描述其他权限。Amazon您的资源Amazonaccount. 要在 CodeDeploy 控制台中使用 CodeDeploy，您必须拥有来自以下服务的权限：

- Amazon EC2 Auto 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

如果创建比必需的最低权限更为严格的 IAM 策略，对于附加了该 IAM 策略的用户，控制台将无法按预期正常运行。为确保这些用户仍可使用 CodeDeploy 控制台，还请将`AWSCodeDeployReadOnlyAccess`向用户提供托管策略，如中所述AmazonCodeDeploy 托管（预定义）策略 (p. 305).

对于仅对Amazon CLI或者 CodeDeploy API。

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策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
    "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": "*"
        }
    ]
}
```

对 Amazon CodeDeploy 身份和访问进行故障排除

可以使用以下信息，以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 (p. 312)

- 我想要查看我的访问密钥 (p. 312)
- 我是管理员并希望允许其他人访问 CodeDeploy (p. 312)
- 我想要允许我之外的人Amazon账户以访问我的 CodeDeploy 资源 (p. 312)

我无权执行 iam:PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。请求该人员更新您的策略，以便允许您将角色传递给 CodeDeploy。

有些 Amazon 服务允许您将现有角色传递到该服务，而不是创建新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。

当名为的 IAM 用户时，会发生以下示例错误`marymajor`尝试使用控制台在 CodeDeploy 中执行操作。但是，服务必须具有服务角色所授予的权限才可执行操作。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，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南 中的管理访问密钥。

我是管理员并希望允许其他人访问 CodeDeploy

要允许其他人访问 CodeDeploy，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 CodeDeploy 中向其授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我想要允许我之外的人Amazon账户以访问我的 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 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 CodeDeploy 是否支持这些功能，请参阅Amazon CodeDeploy 如何与 IAM 协同工作 (p. 300).

- 要了解如何为您拥有的 Amazon Web Services 账户 中的资源提供访问权限，请参阅IAM 用户指南中的为您拥有的另一个 Amazon Web Services 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方Amazon Web Services 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的Amazon Web Services 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南 中的为经过外部身份验证的用户（联合身份验证）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 角色与基于资源的策略有何不同。

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。该表列出每个 CodeDeploy API 操作、您可授予执行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。

您可以使用AmazonCodeDeploy 策略中的范围的条件键以表示条件。有关 Amazon 范围内的键的完整列表，请参阅 《IAM 用户指南》中的可用键。

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 操作和`codedeploy:Get*`指定以单词开头的所有 CodeDeploy 操作`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`*

通配符无法与 *`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 。ARN CodeDeploy 示例.

CodeDeploy API 操作和必需的操作权限

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 IAM 用户关联的多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`*

BatchGetApplications

操作：`codedeploy:BatchGetApplications`

获取有关与 IAM 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 IAM 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 IAM 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`

在蓝/绿部署期间，对于开始将替换环境中的实例注册到 Elastic 负载均衡器是必需的。

资源： `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

CreateApplication

操作：`codedeploy:CreateApplication`

创建与 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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

获取有关与 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 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 IAM 用户关联的所有实例的信息所必需的，或Amazonaccount.

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 IAM 用户关联的部署组的所有部署的信息所必需的，或获取与 IAM 用户关联的所有部署所必需的。Amazonaccount.

资源： `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

ListGitHubAccountTokenNames

操作：`codedeploy:ListGitHubAccountTokenNames`

对于获取已存储的 GitHub 账户连接的名称的列表是必需的。

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 IAM 用户关联的应用程序的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`

向 CodeDeploy 注册本地实例所必需的。

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`

这是在蓝/绿部署期间成功流量路由之后，立即在原始环境中覆盖指定的等待时间并开始终止实例所必需的。

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 IAM 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 IAM 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 IAM 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] 当你指定时`CreateDeployment`权限，还必须指定`GetDeploymentConfig`部署配置的权限和`GetApplicationRevision`或者`RegisterApplicationRevision`应用程序修订的权限。

[2] 有效期为`ListDeployments`在提供某个部署组时无效，但在列出所有与 IAM 用户关联的部署时无效。

[3] 对于`UpdateApplication`，您必须具有`UpdateApplication`对旧应用程序名称和新应用程序名称的权限。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。

CodeDeploy 中的记录和监控

本节概述 CodeDeploy 中的监控、日志记录和事件响应。

审计与 CodeDeploy 的所有交互

CodeDeploy 与Amazon CloudTrail，在您的中捕获由 CodeDeploy 或代表您的身份发出的 API 调用的服务。Amazon并将日志文件传送到您指定的 S3 存储桶。捕获来 CloudTrail CodeDeploy 控制台的 API 调用，可捕获从 CodeDeploy 命令通过Amazon CLI，或者直接从 CodeDeploy API 中使用。通过使用 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。要详细了解 CloudTrail，请参阅。使用 CloudTrail 日志文件中的Amazon CloudTrail用户指南.

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 CloudWatch Logs 代理以便在 CloudWatch 控制台中查看聚合数据或者登录到某个实例以审核日志文件。有关更多信息，请参阅 。在 CloudWatch Logs 控制台中查看 CodeDeploy.

提醒和事件管理

您可以使用 Amazon CloudWatch Events 检测实例或部署状态的更改（event) 在你的 CodeDeploy 操作中。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 Events 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 Events 时，您可以选择以下类型的目标：

- Amazon Lambda 函数
- Kinesis Streams
- Amazon SQS SQS 队列
- 内置目标（CloudWatch 警报操作）
- 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。
- 当发生指定的部署或实例事件时，使用 CloudWatch 警报操作停止、终止、重启或恢复 EC2 实例。

有关更多信息，请参见 Amazon CloudWatch 用户指南的什么是 Amazon CloudWatch Events。

的合规性验证AmazonCodeDeploy

第三方审计员评估的安全性和合规性AmazonCodeDeploy 作为多个组成部分Amazon合规性计划。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

要查看列表Amazon在特定合规性计划范围内的服务，请参阅合规性计划范围内的服务. 有关常规信息，请参阅 Amazon 合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅在 Amazon Artifact 中下载报告。

您在使用 CodeDeploy 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。Amazon提供以下资源来帮助实现合规性：

- 安全性与合规性 Quick Start 指南安全性与合规性 Quick Start 指南 – 这些部署指南讨论了架构注意事项，并提供了在 Amazon 上部署关注安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用 Amazon 创建符合 HIPAA 标准的应用程序。
- Amazon 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。

- 《Amazon Config 开发人员指南》中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub – 此Amazon服务提供了Amazon中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

中的故障恢复能力AmazonCodeDeploy

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施.

中的基础设施安全性AmazonCodeDeploy

作为托管服务，AmazonCodeDeploy 受Amazon中描述的全局网络安全程序Amazon Web Services：安全流程概述白皮书。

你用Amazon发布的 API 调用通过网络访问 CodeDeploy。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 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 委托人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。

Reference

参考

主题

- CodeDeploy AppSpec 文件参考 (p. 321)
- CodeDeploy 代理配置参考 (p. 350)
- Amazon CloudFormation用于 CodeDeploy 参考的模板 (p. 353)
- 将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用 (p. 354)
- CodeDeploy 资源工具包参考 (p. 356)
- CodeDeploy 限制 (p. 359)

CodeDeploy AppSpec 文件参考

本部分仅供参考。有关 AppSpec 文件的概念性介绍，请参阅Application Specification Files (p. 24)。

应用程序规范文件（AppSpec 文件）是YAMLCodeDeploy 用于管理部署的格式或 JSON 格式文件。

> Note
>
> EC2/本部署的AppSpec 文件必须命名为`appspec.yml`或者`appspec.yaml`，除非您正在执行本地部署。有关更多信息，请参阅创建本地部署 (p. 278)。

主题

- Amazon ECS 计算平台上的 AppSpec 文件 (p. 321)
- AppSpec 文件Amazon Lambda计算平台 (p. 322)
- EC2/本地计算平台上的AppSpec 文件 (p. 322)
- AppSpec 文件结构 (p. 322)
- AppSpec 文件示例 (p. 345)
- AppSpec 文件间距 (p. 349)
- 验证您的 AppSpec 文件和文件位置 (p. 350)

Amazon ECS 计算平台上的 AppSpec 文件

对于 Amazon 云服务器计算平台应用程序，CodeDeploy 使用 AppSpec 文件来确定：

- 您的 Amazon ECS 任务定义文件。此值由其 ARN 在`TaskDefinition`指 AppSpec。
- 替换任务集中的容器和端口，或 Application Load Balancer 部署期间会在其中重新路由流量。这是使用`LoadBalancerInfo`指 AppSpec。
- 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，用于在与 Amazon ECS 部署期间生命周期事件对应的挂钩期间运行。有关更多信息，请参阅Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)。

AppSpec 文件Amazon Lambda计算平台

适用于AmazonLambda 计算平台应用程序，CodeDeploy 使用 AppSpec 文件来确定：

- 要部署哪个 Lambda 函数版本。
- 要用作验证测试的 Lambda 函数。

AppSpec 文件可以采用 YAML 格式或 JSON 格式。在创建部署时，您还可以将 AppSpec 文件的内容直接输入到 CodeDeploy 控制台。

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 文件必须是名为的 YAML 格式的文件。`appspec.yml`并且必须放置在应用程序源代码目录结构的根目录中。否则，部署会失败。CodeDeploy 使用它来确定：

- 应从 Amazon S3 或 GitHub 中的应用程序修订安装到您的实例的内容。
- 为响应部署生命周期事件而要运行的生命周期事件挂钩。

完成的 AppSpec 文件之后，将此文件与要部署的内容一起捆绑到存档文件（zip、tar 或压缩的 tar）。有关更多信息，请参阅使用 CodeDeploy 的应用程序修订 (p. 239)。

> **Note**
>
> Windows 服务器实例不支持 tar 和压缩 tar 归档文件格式（.tar 和 .tar.gz）。

有了捆绑的存档文件（在 CodeDeploy 中称为修订），则您可以将其上传至 Amazon S3 存储桶或 Git 存储库。然后使用 CodeDeploy 部署修订。有关说明，请参阅使用 CodeDeploy 创建部署 (p. 254)。

EC2/本地计算平台部署的 appspec.yml 保存在修订的根目录中。有关更多信息，请参阅 为 EC2/本地部署部署添加AppSpec 文件 (p. 243) 和 规划 CodeDeploy 的修订版 (p. 239)。

AppSpec 文件结构

下面是 AppSpec 文件的高级结构，用于部署到AmazonLambda 和 EC2/ 本地计算平台。

YAML 格式的 AppSpec 文件中属于字符串的值一定不能包含在引号 (“”) 中，除非另行指定。

用于亚马逊云服务器部署的应用程序规格文件结构

> **Note**
>
> 此 AppSpec 文件以 YAML 编写，但您可以使用相同的结构，以 JSON 编写一个。JSON 格式的 AppSpec 文件中的字符串总是包含在引号 (“”) 内。

```
version: 0.0
resources:
  ecs-service-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

版本

此部分指定 AppSpec 文件的版本。请勿更改此值。版本是必需的。当前，允许的唯一值为 **0.0**。此值由 CodeDeploy 保留，供将来使用。

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 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329)。

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Amazon 云服务器部署的生命周期事件挂钩列表 (p. 335)。

用于的 AppSpec 文件结构AmazonLambda 部署

Note

此 AppSpec 文件以 YAML 编写，但您可以使用相同的结构，在 JSON 中为 Lambda 部署编写 AppSpec 文件。JSON 格式的 AppSpec 文件中的字符串总是包含在引号 (“”) 内。

```
version: 0.0
resources:
  lambda-function-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

版本

此部分指定 AppSpec 文件的版本。请勿更改此值。版本是必需的。当前，允许的唯一值为 **0.0**。此值由 CodeDeploy 保留，供将来使用。

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 的 “resource” 部分（Amazon ECS 和Amazon Lambda仅适用于部署） (p. 328)。

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 的 “hooks” 部分 (p. 335)。

EC2/本地部署的AppSpec 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

此部分指定 AppSpec 文件的版本。请勿更改此值。版本是必需的。当前，允许的唯一值为 **0.0**。此值由 CodeDeploy 保留，供将来使用。

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。版本是必需的。可以指定以下值：

- linux— 实例是亚马逊 Linux、Ubuntu 服务器或 RHEL 实例。
- Windows— 实例为 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（仅限 EC2/本部署）(p. 324)。

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）。此部分仅适用于 Amazon Linux、Ubuntu 服务器和 Red Hat Enterprise Linux (RHEL) 实例。

有关更多信息，请参阅AppSpec “权限” 部分（仅限 EC2/本部署）(p. 331)。

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 的 “hooks” 部分 (p. 335)。

主题

- AppSpec “文件” 部分（仅限 EC2/本部署）(p. 324)
- AppSpec 的 “resource” 部分（Amazon ECS 和Amazon Lambda仅适用于部署） (p. 328)
- AppSpec “权限” 部分（仅限 EC2/本部署）(p. 331)
- AppSpec 的 “hooks” 部分 (p. 335)

AppSpec “文件” 部分（仅限 EC2/本部署）

向 CodeDeploy 提供以下相关信息：部署的安装event. 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 服务器实例，为 “/”；对于 Windows 服务器实例，为 “\”），则修订中的所有文件都将复制到实例。

中使用的路径source是相对于appspec.yml文件，该文件应该位于修订的根目录。有关修订版的文件结构的详细信息，请参阅规划 CodeDeploy 的修订版 (p. 239)。

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，如/root/destination/directory（在 Linux、RHEL 和乌本图上）或c:\destination\folder(在 Windows 上)。

source 和 destination 分别使用字符串指定。

这些区域有：file_exists_behavior指令是可选的，并指定 CodeDeploy 如何处理部署目标位置上已存在的文件 (但并不是之前的成功部署放置的文件)。此设置可接受以下任意值：

- 不允许：部署失败。这也是未指定选项时的默认行为。
- 覆盖: 来自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。
- 保留: 实例上已有的文件版本将保存在新部署过程中。

使用file_exists_behavior设置，请了解此设置：

- 只能指定一次，并应用于files:。
- 优先于--file-exists-behavior Amazon CLI选项和fileExistsBehaviorAPI 选项（两者都是可选的）。

示例如下：files部分，用于亚马逊 Linux、Ubuntu 服务器或 RHEL 实例。

```
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。

“文件” 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 服务器和 RHEL 实例。

> **Note**
>
> 只有 EC2/本地部署使用files部分。它不适用于AmazonLambda 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 的 “resource” 部分（Amazon ECS 和Amazon Lambda 仅适用于部署）

中的内容`'resources'`部分会随 AppSpec 部署的计算平台而改变。这些区域有：`'resources'`部分包含您的 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。这些区域有：`'resources'`部分的Amazon Lambda部署包含 Lambda 函数的名称、别名、当前版本和目标版本。

主题

- 用于的 AppSpec 的 “资源” 部分AmazonLambda 部署 (p. 328)
- 用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分 (p. 329)

用于的 AppSpec 的 “资源” 部分AmazonLambda 部署

这些区域有：`'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"
        }
    }
}]
```

每个属性都是使用字符串指定的。

- `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。
- `targetversion`— 必需。这是流量要转移到的 Lambda 函数版本。

用于 Amazon ECS 部署部署的 AppSpec 的 “资源” 部分

这些区域有：`'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-for-your-ECS-application"
          ContainerPort: port-used-by-your-ECS-application
# 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-or-DISABLED"
        CapacityProviderStrategy:
          - Base: Integer
            CapacityProvider: "capacityProviderA"
            Weight: Integer
          - Base: Integer
            CapacityProvider: "capacityProviderB"
            Weight: Integer
```

JSON:

```
"Resources": [
  {
    "TargetService": {
      "Type": "AWS::ECS::Service",
          "Properties": {
              "TaskDefinition": "",
              "LoadBalancerInfo": {
                  "ContainerName": "",
                  "ContainerPort":
              },
              "PlatformVersion": "",
              "NetworkConfiguration": {
                  "AwsvpcConfiguration": {
                      "Subnets": [
                          "",
                          ""
                      ],
                      "SecurityGroups": [
                          "",
                          ""
                      ],
                      "AssignPublicIp": ""
                  }
              },
             "CapacityProviderStrategy": [
                {
                    "Base" : Integer,
```

```
                    "CapacityProvider" : "capacityProviderA",
                    "Weight" : Integer
                },
                {
                    "Base" : Integer,
                    "CapacityProvider" : "capacityProviderB",
                    "Weight" : Integer
                }
            ]
        }
    }
  }
]
```

每个属性都是使用字符串指定的，除了ContainerPort，这是一个数字。

- 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 (ARN) 和Amazon服务命名空间。
- ContainerName— 必需。这是包含您的 Amazon ECS 应用程序的 Amazon ECS 容器的名称。它必须是在 Amazon ECS 任务定义中指定的容器。
- ContainerPort— 必需。这是将流量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。有关更多信息，请参阅 。Amazon Fargate平台版本。如果未指定，LATEST默认使用 。
- NetworkConfiguration：可选。在 AwsvpcConfiguration 下，您可以指定以下设置。有关更多信息，请参阅 。AwsVpcConfiguration中的Amazon ECS 容器服务 API 参考。
  - 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。您的 Amazon 弹性容器服务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 elastic network interface 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。如果一个设置也不指定，则 CodeDeploy 将使用当前网络的 Amazon ECS 设置。
- CapacityProviderStrategy：可选。您要用于部署的 Amazon ECS 容量提供商列表。有关更多信息，请参阅 。Amazon ECS 容量提供程序中的Amazon Elastic Container Service Container Service。对于每个容量提供者，您可以指定以下设置。有关这些设置的详细信息，请参阅aws-properties-ecs-service-capacityproviderstrategyitem.html中的Amazon CloudFormation用户指南
  - 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（仅限 EC2/本部署）

`'permissions'` 部分指定应如何向已复制到实例的 `'files'` 部分中的文件和目录/文件夹应用特殊权限（如果有）。您可以指定多个 `object` 指令。此部分是可选的。它仅适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例。

> **Note**
>
> 这些区域有：`'permissions'`部分仅用于 EC2/ 本地部署。它不会用于Amazon弹性云服务器或亚马逊云服务器部署。

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，指定要应用于`object`。模式设置遵循Linux chmod 命令语法。

  > **Important**
  >
  > 如果值包含前导零，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**
>
> 符号符号，例如**u+x**不支持mode设置。

示例：

- `mode: "0644"`向对象所有者 (6) 授予读取和写入权限，组具有只读权限 (4)，所有其他用户具有只读权限 (4)。
- `mode: 644`授予与`mode: "0644"`。
- `mode: 4755`设置 setuid 属性 (4)，向所有者授予完全控制权限 (7)，向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。

  如果未指定模式，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`，则在执行复制操作之后，应用于原始文件或目录/文件夹结构的所有现有 ACL 将保持不变。这些条目将替换任何现有的 ACL。

  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、未命名的组或其他类似的 ACL 条目将导致 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 Security-Enhanced Linux (SELinux) 的实例，为一个应用于已复制对象的安全相关上下文标签的列表。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，则在执行复制操作之后，应用于原始文件或目录/文件夹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）。如果指定了 **`directory`**，则在执行复制操作之后，权限将以递归方式应用于 `object` 中任何位置的所有目录/文件夹（但不应用于 `object` 自身）。

  使用短划线 (-) 后跟空格和字符串的形式指定 `type`（例如，`- file`）。

“权限” 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 服务器和 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 文件显示如何在复制这些文件和文件夹之后设置权限：

```
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 服务器和 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 的 “hooks” 部分

中的内容`'hooks'`部分会随 AppSpec 部署的计算平台而改变。这些区域有：`'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。这些区域有：`'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

- Amazon ECS 部署的 AppSpec 的 “hooks” 部分 (p. 335)
- 用于 AppSpec 的 “hooks” 部分AmazonLambda 部署 (p. 338)
- EC2/本部署的AppSpec “挂钩” 部分 (p. 340)

Amazon ECS 部署的 AppSpec 的 “hooks” 部分

主题

- Amazon 云服务器部署的生命周期事件挂钩列表 (p. 335)
- Amazon ECS 部署中挂钩的运行顺序 (p. 335)
- “挂钩” 部分的结构 (p. 336)
- 示例 Lambda “hooks” 函数 (p. 337)

Amazon 云服务器部署的生命周期事件挂钩列表

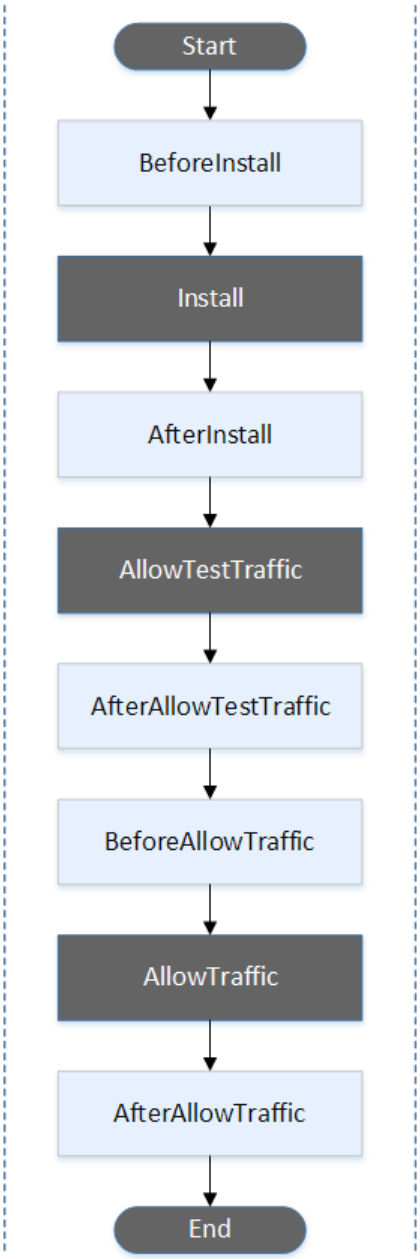
一个AmazonLambda 挂钩是一个 Lambda 函数，该函数在生命周期事件名称之后的新行中使用字符串指定。对于每次部署，每个挂钩将执行一次。以下是在 Amazon ECS 部署期间可以运行挂钩的生命周期事件的描述。

- `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 亚马逊云服务器部署过程中会发生什么 (p. 16) 和 教程：通过验证测试部署 Amazon ECS 服务 (p. 117)。

Amazon ECS 部署中挂钩的运行顺序

在 Amazon ECS 部署中，事件挂钩按以下顺序运行：



Note

部署中的 Start、Install、TestTraffic、AllowTraffic 和 End 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。

“挂钩” 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"
  }
 ]
}
```

示例 Lambda “hooks” 函数

使用`'hooks'`部分可以指定 CodeDeploy 可以调用的用于验证 Lambda 部署的 Lambda 函数。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函数。验证测试完成后，Lambda`AfterAllowTraffic`函数回调 CodeDeploy 并提供`Succeeded`或者`Failed`。

> **Important**
>
> 如果 Lambda 验证函数未在一个小时内通知 CodeDeploy，则部署将被视为已失败。

在调用 Lambda 挂钩函数之前，必须使用`putLifecycleEventHookExecutionStatus`命令。

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');
        }
    });
};
```

用于 AppSpec 的 “hooks” 部分AmazonLambda 部署

主题

- 生命周期事件挂接列表AmazonLambda 部署 (p. 338)
- Lambda 函数版本部署中挂钩的运行顺序 (p. 338)
- “挂钩” 部分的结构 (p. 338)
- 示例 Lambda “hooks” 函数 (p. 339)

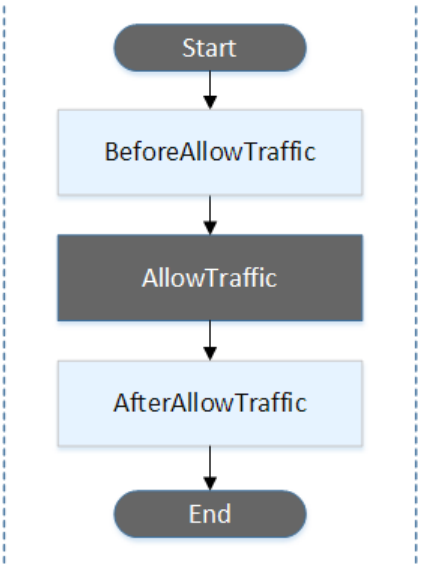
生命周期事件挂接列表AmazonLambda 部署

一个AmazonLambda 挂钩是一个 Lambda 函数，该函数在生命周期事件名称之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。以下是对可用于您的 AppSpec 文件中的挂钩的描述。

- BeforeAllowTraffic— 用于在将流量转移到部署的 Lambda 函数版本之前运行任务。
- AfterAllowTraffic— 用于在将流量转移到部署的 Lambda 函数版本之后运行任务。

Lambda 函数版本部署中挂钩的运行顺序

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 Start、AllowTraffic 和 End 事件无法脚本化，这就是为什么它们在此图中灰显。

“挂钩” 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"
}]
```

示例 Lambda “hooks” 函数

使用 “hooks” 部分可以指定 CodeDeploy 可以调用的用于验证 Lambda 部署的 Lambda 函数。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函数。验证测试完成后，Lambda 验证函数将回调 CodeDeploy 并交付`Succeeded`或者`Failed`。

> **Important**
>
> 如果 Lambda 验证函数未在一个小时内通知 CodeDeploy，则部署将被视为已失败。

在调用 Lambda 挂钩函数之前，必须使用`putLifecycleEventHookExecutionStatus`命令。

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');
        }
    });
```

```
};
```

EC2/本部署的AppSpec “挂钩” 部分

主题

- 生命周期事件挂接列表 (p. 340)
- 生命周期事件挂钩可用 (p. 341)
- 部署中挂接的运行顺序 (p. 342)
- “挂钩” 部分的结构 (p. 343)
- 挂钩的环境变量可用性 (p. 344)
- 挂钩示例 (p. 345)

生命周期事件挂接列表

对于实例的每次部署，EC2/本地部署挂钩执行一次。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。以下是对可用于您的 AppSpec 文件中的挂钩的描述。

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 (p. 341)。

- `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的 AppSpec 文件和脚本来自于上一个成功部署的应用程序修订。

  > **Note**
  >
  > 在您部署实例之前，实例上不存在 AppSpec 文件。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。

  为确定上次成功部署的应用程序修订的位置，CodeDeploy 代理将查看*`deployment-group-id`*`_last_successful_install`文件。此文件位于：

  `/opt/codedeploy-agent/deployment-root/deployment-instructions`文件夹、Ubuntu 服务器和 RHEL Amazon EC2 实例上的文件夹。

  `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions`文 Amazon EC2。

  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 排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)。
- `DownloadBundle`-在此部署生命周期事件期间，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`文件夹、Ubuntu 服务器和 RHEL Amazon EC2 实例上的文件夹。

  `C:\ProgramData\Amazon\CodeDeploy\`*`deployment-group-id`*`\`*`deployment-id`*`\deployment-archive`文 Amazon EC2。

  此事件是为 CodeDeploy 代理预留的，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 排查失败的 DownloadBundle 部署生命周期事件的问题，错误为“UnknownError: not opened for reading” (p. 369)。
- `BeforeInstall`— 您可以使用此部署生命周期事件执行预安装任务，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期间，CodeDeploy 代理会将修订文件从临时位置复制到最终目标文件夹中。此事件是为 CodeDeploy 代理预留的，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`-您通常使用此部署生命周期事件来重新启动在期间停止的服务。`ApplicationStop`。
- `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 排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)。
- `BlockTraffic`-在此部署生命周期事件期间，阻止 Internet 流量访问当前正在处理流量的实例。此事件是为 CodeDeploy 代理预留的，不能用于运行脚本。
- `AfterBlockTraffic`-在从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 排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)。
- `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 Internet 流量在部署后访问实例。此事件是为 CodeDeploy 代理预留的，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

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	就地部署 [1]	蓝/绿部署：Original instances	蓝/绿部署：替换实例	蓝/绿部署回滚：Original instances	蓝/绿部署回滚：替换实例
ApplicationStop	✓		✓		
DownloadBundle[2]	✓		✓		
BeforeInstall	✓		✓		
安装[2]	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic[2]	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic[2]	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	

[1] 也适用于就地部署的回滚。

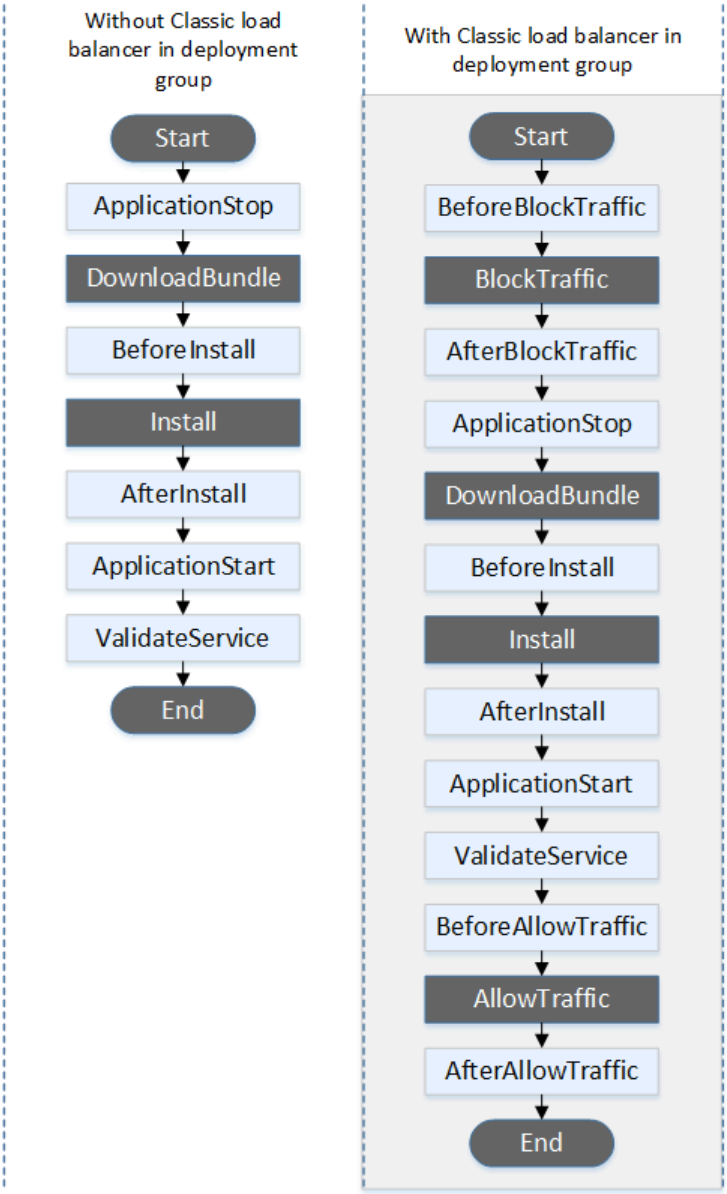
生命周期事件名称	就地部署 [1]	蓝/绿部署：Original instances	蓝/绿部署：替换实例	蓝/绿部署回滚：Original instances	蓝/绿部署回滚：替换实例
[2] 为 CodeDeploy 操作保留。不能用于运行脚本。					

部署中挂接的运行顺序

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，与阻止并允许流量相关的六个挂钩仅当您在部署组中指定弹性 Classic Load Balancer 平衡器、应用程序负载平衡器或网络负载平衡器时适用。

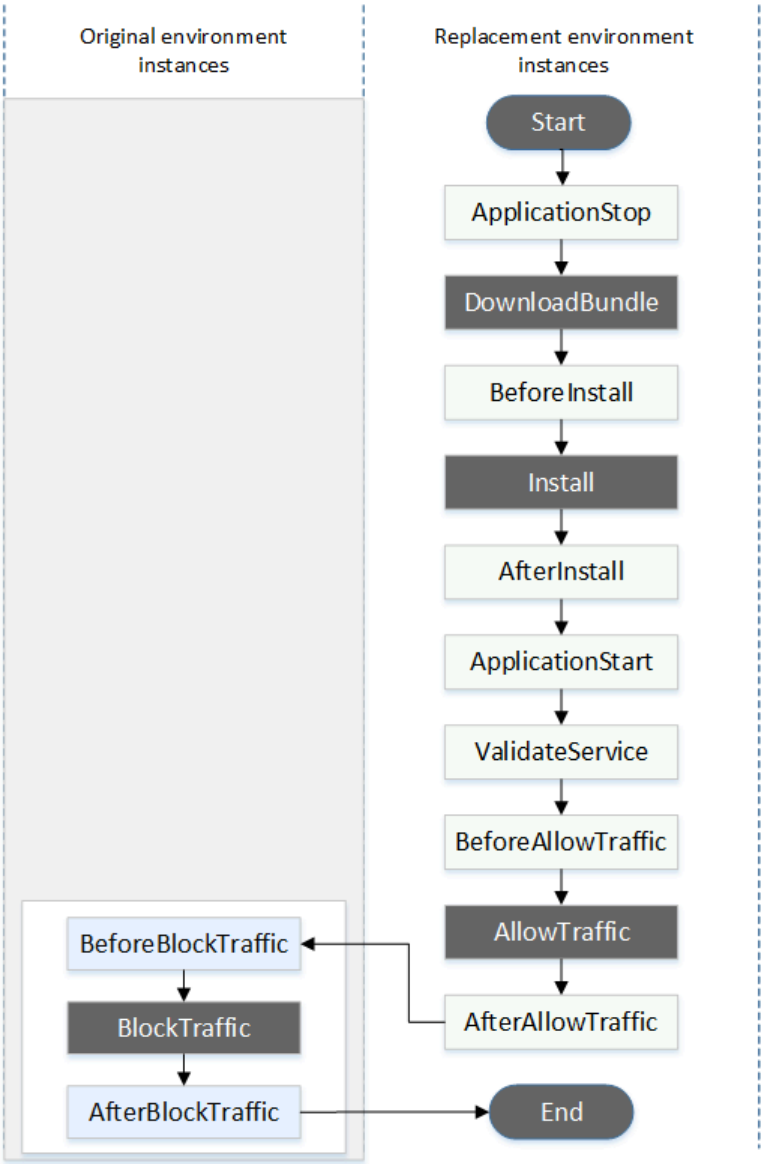


Note

部署中的 Start、DownloadBundle、Install 和 End 事件无法脚本化，这就是为什么它们在此图中灰显。但是，您可以编辑`'files'`部分指定在 AppSpec 文件期间安装的内容。安装event.

蓝/绿部署

在蓝/绿部署中，事件挂钩按以下顺序运行：



Note

部署中的 Start、DownloadBundle、Install、BlockTraffic、AllowTraffic 和 End 事件无法脚本化，这就是它们在此图中灰显的原因。不过，您可以编辑 AppSpec 文件的 “files” 部分，以指定在安装event.

“挂钩” 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：

位置

必填项。修订的脚本文件包的位置。指定脚本的位置在hooks部分相对于应用程序修订包的根目录指定。有关更多信息，请参阅规划 CodeDeploy 的修订版 (p. 239)。

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 不会存储密码，因此，如果runas用户需要密码。此元素仅适用于 Amazon Linux 和 Ubuntu 服务器实例。

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

CodeDeploy 中属于当前部署的应用程序的名称（例如，`WordPress_App`)。

DEPLOYMENT_ID

已分配给当前部署的 ID CodeDeploy（例如，`d-AB1CDEF23`)。

DEPLOYMENT_GROUP_NAME

CodeDeploy 中属于当前部署的部署组的名称（例如，`WordPress_DepGroup`)。

DEPLOYMENT_GROUP_ID

CodeDeploy 中属于当前部署的部署组的 ID（例如，`b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE`)。

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。

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，请参阅规划 CodeDeploy 的修订版 (p. 239)。

AppSpec 文件示例

此主题 AppSpec 用于AmazonLambda 和 EC2/ 本地部署。

主题

- Amazon ECS 部署 AppSpec 文件示例 (p. 346)
- AppSpec 文件示例AmazonLambda 部署 (p. 347)
- EC2/本地部署的AppSpec 文件示例 (p. 348)

Amazon ECS 部署 AppSpec 文件示例

下面是使用 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. 在更换任务集上安装更新的 Amazon ECS 应用程序之前，名为的 Lambda 函数。`LambdaFunctionToValidateBeforeInstall`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 Lambda 接收任何流量之前，名为`LambdaFunctionToValidateAfterInstall`运行。
3. 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后，名为的 Lambda 函数。`LambdaFunctionToValidateAfterTestTrafficStarts`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`AfterAllowTestTraffic`挂钩，并且在将生产流量提供给更新的 Amazon ECS 应用程序之前，名为的 Lambda 函数。`LambdaFunctionToValidateBeforeAllowingProductionTraffic`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，名为的 Lambda 函数。`LambdaFunctionToValidateAfterAllowingProductionTraffic`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 文件示例AmazonLambda 部署

下面是使用 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. 在从 Lambda 为`myLambdaFunction`到版本 2，运 Lambda 个名为`LambdaFunctionToValidateBeforeTrafficShift`，验证部署是否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. 在将流量从 Lambda 为`myLambdaFunction`到版本 2 已完成，请运行一个名为`LambdaFunctionToValidateAfterTrafficShift`验证部署已成功完成。

EC2/本地部署的AppSpec 文件示例

下面是对 Amazon Linux、Ubuntu 服务器或 RHEL 实例进行就地部署的 AppSpec 文件示例。

> **Note**
>
> 部署到 Windows 服务器实例不支持`runas`元素。如果要部署到 Windows 服务器实例，请不要将其包含在您的 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，更改`os: linux`到`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

下面是 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]`意味着在项目之间插入四个空格。CodeDeploy 引发可能难以调试的错误，如果 AppSpec 文件中的位置和空格数不正确。

```
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
```

下面是间距正确的 AppSpec 文件的示例：

```
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

可以使用Amazon提供的 AppSpec 助手脚本，用于验证 AppSpec 文件的内容。您 AppSpec 在GitHub。

您还可以使用基于浏览器的工具，如YAML 绒或者YAML 在线解析器来帮助您检查 YAML 语法。

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 文件不在该根目录中，则将显示 “No or file or directory” 错误。

在本地 Windows 实例上：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec 文件不在该根目录中，则将显示 “File Not Found” 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 在与实例交互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。

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:	设置为`true`，以从 Amazon S3 捕获线路日志并将它们写入名为`codedeploy-agent.wire.log`指向的位置:log_dir:设置。 **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，包括在此设置设为时传入或传出 Amazon S3 的文件的纯文本内容。`true`。线路日志包含有关所有 Amazon S3 活动的信息，这些活动与Amazon帐户，而此设置设置为`true`，而不仅仅是与 CodeDeploy 部署相关的活动。 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。 默认设置为`'/var/log/aws/codedeploy-agent'`适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例，`C:\ProgramData\Amazon\CodeDeploy\log`对于 Windows Server 实例。
: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'`. 此设置仅适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例。
:program_name:	CodeDeploy 代理程序名称。 默认设置为 `codedeploy-agent`. 此设置仅适用于亚马逊 Linux、Ubuntu 服务器和 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。 默认设置为`'/opt/codedeploy-agent/deployment-root'`适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例，`C:\ProgramData\Amazon\CodeDeploy`对于 Windows Server 实例。

:verbose:	设置为`true`以在实例上打印调试消息日志文件。 默认设置为`false`适用于亚马逊 Linux、Ubuntu 服务器和 RHEL 实例，`true`对于 Windows Server 实例。
:wait_between_runs:	对于挂起的部署，CodeDeploy 代理轮询代理之间的时间间隔（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`（适用于 Ubuntu 服务器和 RHEL）或`conf.onpremises.yml`（适用于 Windows Server 的）。 默认情况下，这些文件存储在`/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`适用于 Ubuntu 服务器和 RHELC:`\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`对于 Windows Server。 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 代理通过其连接到的 HTTP 代理Amazon，用于您的 CodeDeploy 操作。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。
: 启用授权策略:	（可选）设置为`true`如果你想使用IAM 授权配置访问控制并限制 CodeDeploy 代理正在使用的 IAM 角色或用户的权限。目的将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用 (p. 354)，此值必须为`true`。 默认设置为 `false`.

相关主题

使用 CodeDeploy 代理 (p. 141)

管理 CodeDeploy 代理操作 (p. 149)

Amazon CloudFormation用于 CodeDeploy 参考的模板

此部分介绍的Amazon CloudFormation资源、转换和挂钩设计用于 CodeDeploy 部署。有关为创建由Amazon CloudFormation挂钩 CodeDeploy 参阅通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 264)

> **Note**
>
> Amazon CloudFormation挂钩是Amazon CloudFormation的组件Amazon并且与 CodeDeploy 生命周期事件挂钩不同。

除了 CodeDeploy 中可供您使用的其他方法外，您还可以使用Amazon CloudFormation模板执行以下任务：

- 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，可帮助您使用模板来建模和设置 Amazon 资源。Amazon CloudFormation 模板是一个文本文件，其格式符合 JSON 格式标准。您可创建一个模板来描述所需的所有 Amazon 资源，而 Amazon CloudFormation 负责为您预配和配置这些资源。

有关更多信息，请参阅 。是什么Amazon CloudFormation?和使用Amazon CloudFormation模板在Amazon CloudFormation用户指南。

如果您计划使用Amazon CloudFormation模板与 CodeDeploy 兼容，您作为管理员，必须授予对Amazon CloudFormation和Amazon服务和操作Amazon CloudFormation取决于。要授予创建应用程序、部署组和部署配置的权限，请将以下策略附加到将使用Amazon CloudFormation：

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

有关托管策略的更多信息，请参阅以下主题：

- 要查看必须附加到将创建 Amazon EC2 实例的 IAM 用户的策略，请参阅为 CodeDeploy 创建 Amazon EC2 实例 (Amazon CloudFormation模板) (p. 174)。
- 有关将策略附加到 IAM 用户的信息，请参阅使用托管策略在IAM 用户指南。

- 要了解如何仅允许用户使用一组有限的 CodeDeploy 操作和资源，请参阅AmazonCodeDeploy 托管（预定义）策略 (p. 305)。

下表显示 Amazon CloudFormation 模板可代表您执行的操作，并且包含一些链接指向您可添加到 Amazon CloudFormation 模板的 Amazon 资源类型及其属性类型的相关详细信息。

操作	Amazon CloudFormation参考	参考类型
创建 CodeDeploy 应用程序。	aws-resource-codedeploy-application.html	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
创建一组供 CodeDeploy 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 Amazon EC2 实例。[2]	aws-properties-ec2-instance.html	Amazon CloudFormation 资源
使用Amazon CloudFormation `AWS::CodeDeployBlueGreen`转换和`AWS::CodeDeploy::BlueGreen`挂钩管理堆栈更新、创建资源和转移流量以进行 CodeDeploy 蓝/绿部署。[3]	AWS::CodeDeployBlueGreen	`AWS::CodeDeployBlueGreen` 转换是一个由 Amazon CloudFormation 托管的宏
	AWS::CodeDeploy::BlueGreen	`AWS::CodeDeploy::BlueGreen` 挂钩在 Amazon CloudFormation 中构建为 Hook 资源。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CodeDeploymentGroup DeploymentGroup S3Location和CodeDeploy DeploymentGroup 部署修订 GitHubLocation中的Amazon CloudFormation用户指南。 [2] 我们提供了您可用来在支持 CodeDeploy 的区域中创建 Amazon EC2 实例的模板。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) (p. 174)。 [3]O此部署配置支持使用 Amazon ECS 蓝/绿部署。有关通过Amazon CloudFormation，请参阅 部署配置Amazon CloudFormation蓝/绿部署（Amazon ECS） (p. 208)。有关通过Amazon CloudFormation以及如何在 CodeDeploy 中查看您的部署，请参阅通过创建 Amazon ECS 蓝/绿部署Amazon CloudFormation (p. 264)。		

将 CodeDeploy 与 Amazon Virtual Private Cloud 结合使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管您的Amazon资源，则您可以在 VPC 和 CodeDeploy 之间建立私有连接。您可以使用此连接实现 CodeDeploy 与 VPC 上的资源的通信而不用访问公共 Internet。

Amazon VPC 是一个Amazon服务，可用来启动Amazon您定义的虚拟网络中的资源。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。对于 VPC 终端节点，VPC 和之间的路由Amazon服务由Amazon网络，并且您可以使用 IAM 策略来控制对服务资源的访问。

要将 VPC 连接到 CodeDeploy，请定义一个接口 VPC 终端节点（对于 CodeDeploy. 接口终端节点是具有私有 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。该终端节点提供了到 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 。什么是 Amazon VPC中的Amazon VPC 用户指南。

接口 VPC 终端节点由Amazon PrivateLink，一个Amazon技术，使之间的私人通信Amazon服务，使用具有私有 IP 地址的 elastic network interface。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 。开始使用中的Amazon VPC 用户指南。

Availability

CodeDeploy 中的 VPC 终端节点支持以下内容中的Amazon区域：

- 美国东部 (俄亥俄)
- 美国东部 (弗吉尼亚北部)
- 美国西部 (加利福尼亚北部)
- 美国西部 (俄勒冈)
- Africa (Cape Town)
- 亚太地区（香港）
- 亚太地区 (孟买)
- Asia Pacific (Osaka)
- 亚太地区 (首尔)
- 亚太地区 (新加坡)
- 亚太地区 (悉尼)
- 亚太地区 (东京)
- 加拿大 (中部)
- 中国（北京）
- 中国 (宁夏)
- 欧洲（法兰克福）
- 欧洲（爱尔兰）
- 欧洲（伦敦）
- Europe (Milan)
- 欧洲（巴黎）
- 欧洲（斯德哥尔摩）
- 中东（巴林）
- 南美洲（圣保罗）

为 CodeDeploy.

要开始将 CodeDeploy 与您的 VPC 结合使用，请为 CodeDeploy 创建一个接口 VPC 终端节点。CodeDeploy 需要用于 Git 操作和 CodeDeploy API 操作的单独终端节点。根据您的业务需求，您可能需要创建多个 VPC 终端节点。当您为 CodeDeploy 创建 VPC 终端节点时，请选择Amazon服务，并在服务名称，请从以下选项中进行选择：

- com.amazonaws.##.codeDeploy：如果要为 CodeDeploy API 操作创建 VPC 终端节点，请选择此选项。例如，如果您的用户使用Amazon CLI、CodeDeploy API 或Amazon用于与 CodeDeploy 进行交互的软件开发工具包，例如`CreateApplication`、`GetDeployment`, 和`ListDeploymentGroups`。
- com.amazonaws.##.code部署/命令-安全：如果要为 CodeDeploy 代理操作创建 VPC 终端节点，请选择此选项。您还需要将`:enable_auth_policy:`到`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 (p. 356)。

如果使用 Lambda 或 ECS 部署，则只需为创建 VPC 终端节点。com.amazonaws.##.codeDeploy。使用 Amazon EC2 部署的客户将需要com.amazonaws.##.codeDeploy和com.amazonaws.##.code部署/命令-安全。

配置 CodeDeploy 代理和 IAM 权限

要将 Amazon VPC 终端节点与 CodeDeploy 一起使用，您必须将`:enable_auth_policy:`到`true`位于 EC2 或本地实例上的代理配置文件中。有关代理配置文件的更多信息，请参阅CodeDeploy 代理配置参考 (p. 350)。

如果您正在使用本地实例，您还必须向 Amazon EC2 实例配置文件或 IAM 用户或角色添加以下 IAM 权限。

```
{
  "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 资源工具包参考

CodeDeploy 依赖的许多文件都存储在公共使用的Amazon区域特定的 Amazon S3 存储桶。这些文件包含 CodeDeploy 代理的安装文件、模板以及示例应用程序文件。我们将此文件集合称为 CodeDeploy 资源工具包。

主题

- 按区域列出的资源工具包存储桶 (p. 356)
- 资源工具包内容 (p. 357)
- 显示资源工具包文件的列表 (p. 358)
- 下载资源工具包文件 (p. 358)

按区域列出的资源工具包存储桶

此表列出了指南中某些过程所需的 *bucket-name* 替换名称。这些是包含 CodeDeploy 资源工具包文件的 Amazon S3 存储桶的名称。

Note

要访问亚太地区（香港）区域中的 Amazon S3 存储桶，您必须启用Amazonaccount. 有关更多信息，请参阅 。管理Amazon区域。

区域名称	#####replacement	区域标识符
中国（北京）	aws-代码-cn-nor1	cn-north-1
中国 (宁夏)	aws-codeal-cn-west-1	cn-northwest-1

资源工具包内容

下表列出了 CodeDeploy 资源工具包中的文件。

File	描述
`LATEST_VERSION`	Amazon EC2 系统管理器等更新机制用于确定 CodeDeploy 代理的最新版本的文件。
`VERSION`	自动更新机制已在 CodeDeploy 代理版本 1.1.0 中删除，并且不再使用此文件。 CodeDeploy 代理运行在实例上时用来更新自身的文件。
`codedeploy-agent.noarch.rpm`	适用于 Amazon Linux 和 Red Hat Enterprise 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 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`)。
`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`	您可以用来更轻松地安装 CodeDeploy 代理的文件。
`CodeDeploy_SampleCF_Template.json`	一个Amazon CloudFormation模板，您可用来启动运行 Amazon Linux 或 Windows 服务器的一至三个 Amazon 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`	一个 Amazon CloudFormation 模板，可用于创建 Apache Web 服务器上运行的负载均衡示例网站。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。此模板将创建三个 Amazon EC2 实例和 IAM 实例配置文件，以为实例授予对 Amazon S3、Amazon EC2 Auto Scaling、Amazon S3、Amazon EC2 Auto Scaling、Amazon CloudFormation和 Elastic Load Balancing。它还会创建负载均衡器和 CodeDeploy 服务角色。

File	描述
`SampleApp_ELB_Integration.zip`	一个示例应用程序修订，可部署到已注册 Elastic Load Balancing 负载均衡器的 Amazon EC2 实例。
`SampleApp_Linux.zip`	一个示例应用程序修订，您可以将它们部署到运行 Amazon Linux 的 Amazon EC2 实例，或者部署到 Ubuntu 服务器或 RHEL 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`)。
`SampleApp_Windows.zip`	一个示例应用程序修订，可部署到 Windows 服务器实例。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`)。

显示资源工具包文件的列表

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 限制

下表描述了 CodeDeploy 中的限制。

> **Note**
>
> 您可以请求提高限制中列出的 CodeDeploy 限制Amazon服务限制中的Amazon Web Services 一般参考。不能增加对部署可运行的小时数的限制。

主题

- Applications (p. 359)
- 应用程序修订 (p. 359)
- Deployments (p. 360)
- 部署配置 (p. 360)
- 部署组 (p. 361)
- Instances (p. 361)

Applications

单个区域中与一个 Amazon 账户关联的最大应用程序数量	1000
应用程序名称中的最大字符数	100
应用程序名称中允许包含的字符	字母（a-z、A-Z）、数字 (0-9)、句点 (.)、下划线 (_)、+（加号）、=（等号）、,（逗号）、@ 符号、-（减号）。
可传递给 BatchGetApplications API 操作的应用程序的最大数目	100
单个 GitHub 连接令牌的最大数量Amazon账户	25

应用程序修订

应用程序修订名称中的最大字符数	100
EC2/本地应用程序修订的允许的文件类型	扩展名为 `.zip` 或 `.tar` 的存档文件，以及扩展名为 `.tar.gz` 的压缩存档文件。 与 CodeDeploy 兼容的存档文件或压缩存档文件必须包含单个文件名为的应用程序规范文件（AppSpec 文件）。`appspec.yml`。
的允许的文件类型AmazonLambda 和亚马逊云服务器应用程序修订版	具有文件名的单个 AppSpec 文件`appspec.yaml`，或带有扩展名的压缩文件`.zip`或者`.tar.gz`，其中包含一个具有文件名的`appspec.yaml`。

Deployments

针对一个部署组的并发部署的最大数目[1]	1
与一个 Amazon 账户关联的最大并发部署数量[2]	100
EC2/本地部署可以运行的最大小时数	8
在 EC2/本地或Amazon ECS 蓝/绿部署期间，在部署修订和将流量转移到替换环境间的最大小时数	48
在 EC2/本地或Amazon ECS 蓝/绿部署期间，在完成部署和中止原始环境间的最大小时数	48
EC2/本地蓝/绿部署可以运行的最大小时数	109 (对于上述两个限制中的每一个为 48) 加上每个可能生命周期事件 (共 13 个) 1 小时
的最大小时数AmazonLambda 部署可以运行	50 (对于在第一次流量转移和最后一次流量转移之间的最长时间为 48 小时，加上对于每个可能生命周期挂钩 (共计两个) 1 小时)
未完成的部署生命周期事件失败之前经过的最大秒数	3600
部署描述中的最大字符数	256
可传递给 BatchGetDeployments API 操作的部署的最大数目	100
如果生命周期事件未在以下事件后启动，部署失败前经过的最大分钟数： • 使用控制台或Amazon CLI 创建部署命令。 • 前一个生命周期事件已完成。	5
蓝/绿部署成功后可以等待的分钟数上限，之后将终止原始部署的实例	2800
Amazon ECS 部署期间流量路由上的最大侦听器数	1
[1] 此限制旨在防止意外地将同一应用程序并发部署到同一部署组。 [2] Amazon EC2 Auto Scaling 组中的扩展 Amazon EC2 实例的每个部署计为单个并发部署。如果扩展的 Amazon EC2 实例与多个应用程序相关联，则会为每个应用程序生成额外的并发部署。例如，如果 Amazon EC2 Auto Scaling 组扩展五个 Amazon EC2 实例并与单个应用程序相关联，则会生成五个并发部署。如果同五个扩展的 Amazon EC2 实例与两个额外的应用程序相关联，则会生成十个额外的并发部署。	

部署配置

与一个 Amazon 账户关联的自定义部署配置的最大数目	50
允许的最小正常运行的实例数设置值 HOST_COUNT	任何正整数或 0（零）。零 (0) 将导致一次部署到所有实例。

允许的最小正常运行的实例数设置值 FLEET_PERCENT	任何小于 100 的正整数或 0（零）。零 (0) 将导致一次部署到所有实例。
自定义部署配置名称中的最大字符数	100
自定义部署配置名称中允许包含的字符	字母（a-z、A-Z）、数字 (0-9)、句点 (.)、下划线 (_)、+（加号）、=（等号）、,（逗号）、@ 符号、-（减号）。
自定义部署配置名称中不允许使用的前缀	`CodeDeployDefault.`
在第一次流量转移和最后一次流量转移之间的最大分钟数AmazonLambda 金丝雀或线性部署	2880
在一次递增中可以转移的最大流量百分比Amazon使用金丝雀部署或线性部署配置的 Lambda 部署。	99

部署组

与单个应用程序关联的部署组的最大数量	1000
部署组中的最大标签数	10
部署组中的 Amazon EC2 Auto Scaling 组的最大数量	10
部署组名称中的最大字符数	100
部署组名称中允许包含的字符	字母（a-z、A-Z）、数字 (0-9)、句点 (.)、下划线 (_)、+（加号）、=（等号）、,（逗号）、@ 符号、-（减号）。
一个部署组中事件通知触发器的最大数量	10
可与 Amazon ECS 服务关联的部署组的最大数量	1

Instances

单次部署中的最大实例数量	500
标签键中的最大字符数	128
标签值中的最大字符数	256
可传递给 BatchGetOnPremisesInstances API 操作的实例的最大数目	100
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	500
的必需版本Amazon SDK for Ruby。	2.1.2或更低版本（适用于早于 1.0.1.880 的 CodeDeploy 版本）。 2.2或更低版本（适用于 CodeDeploy 版本 1.0.1.880 及更高版本）。

排查 CodeDeploy

使用本部分中的主题可帮助解决您在使用 CodeDeploy 时可能遇到的问题和错误。

> **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，我们建议您使用 Amazon CloudWatch Logs 集中监控日志文件，而不是逐个实例查看这些文件。想要了解有关信息，请参阅在 CloudWatch Logs 控制台中查看 CodeDeploy.

主题

- 一般问题排查 (p. 362)
- EC2/本地部署问题 (p. 364)
- 故障排除AmazonLambda 部署问题 (p. 371)
- 排查部署组问题 (p. 372)
- 排查实例问题 (p. 372)
- GitHub 令牌问题排查 (p. 373)
- 解决 Amazon EC2 Auto Scaling 问题 (p. 374)
- 的错误代码Amazon CodeDeploy (p. 383)

一般问题排查

主题

- 一般问题排查核对清单 (p. 362)
- 只有部分支持 CodeDeploy 部署资源Amazon区域 (p. 363)
- 本指南中的步骤与 CodeDeploy 控制台不匹配 (p. 363)
- 所需的 IAM 角色不可用 (p. 364)
- 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 (p. 364)
- 使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 (p. 364)

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 (p. 266)和View Instance Details (p. 202)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：
   - 是否已使用指定的 EC2 key pair 启动实例？ 有关更多信息，请参阅 。EC2 密钥对在适用于 Linux 实例的 Amazon EC2 用户指南.
   - 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 Amazon EC2 实例以使用 CodeDeploy (p. 177)和第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)。
   - 是否已标记实例？ 有关更多信息，请参阅 。使用控制台中的标签在适用于 Linux 实例的 Amazon EC2 用户指南.
   - CodeDeploy 理是否已安装、更新并在实例上运行？ 有关更多信息，请参阅管理 CodeDeploy 代理操作 (p. 149)。要检查安装了哪个版本的代理，请参阅确定 CodeDeploy 代理的版本 (p. 150).
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 (p. 219)。
- 要检查部署组设置，请参阅使用 CodeDeploy 查看部署组详细信息 (p. 234)。

4. 确认已正确配置应用程序修订：
   - 检查 AppSpec 文件的格式。有关更多信息，请参阅将应用程序规范文件添加到 CodeDeploy 的修订版 (p. 239)和CodeDeploy AppSpec 文件参考 (p. 321)。
   - 检查 Amazon S3 存储桶或 GitHub 存储库以验证应用程序修订是否在预期位置。
   - 查看 CodeDeploy 应用程序修订的详细信息以确保它已正确注册。有关信息，请参阅 查看应用程序修订详细信息 (p. 249)。
   - 如果您正在从 Amazon S3 部署，请检查您的 Amazon S3 存储桶以验证 CodeDeploy 已获得下载应用程序修订的权限。有关存储桶策略的信息，请参阅部署先决条件 (p. 254)。
   - 如果要从 GitHub 部署，请检查您的 GitHub 存储库以验证 CodeDeploy 已获得下载应用程序修订的权限。有关更多信息，请参阅使用 CodeDeploy 创建部署 (p. 254)和使 GitHub CodeDeploy 中的应用程序进 (p. 48)。
5. 确保已正确配置服务角色。有关信息，请参阅 第 2 步：为 CodeDeploy 创建服务角色 (p. 26)。
6. 确认您已执行开始使用 CodeDeploy (p. 26)中的步骤，以便：
   - 将策略附加到 IAM 用户。
   - 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，请参阅适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297)。
7. 确认您正在使用 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。

只有部分支持 CodeDeploy 部署资源Amazon区域

如果您看不到或无法访问应用程序、部署组、实例或其他部署资源Amazon CLI或 CodeDeploy 控制台，请确保您引用的是其中一个Amazon中列出的区域区域和终端节点在Amazon一般参考.

必须在这些实例之一中启动和创建在 CodeDeploy 部署中使用的 EC2 实例和 Amazon EC2 Auto Scaling 组（在 CodeDeployAmazon地区。

如果您使用的是 Amazon CLI，请从 Amazon CLI 运行 aws configure 命令。然后，您可以查看和设置默认设置Amazon区域。

如果您使用的是 CodeDeploy 控制台，请从导航栏上的区域选择器中，选择一个受支持的Amazon地区。

> **Important**
>
> 要使用中国 (北京) 区域或中国 (宁夏) 区域中的服务，您必须拥有这些区域的帐户和凭证。其他账户和凭证Amazon地区不适用于北京和宁夏地区，反之亦然。
> 有关中国区域的某些资源的信息，例如 CodeDeploy 资源套件存储桶名称和 CodeDeploy 代理安装过程，不包括在本版本的CodeDeploy 用户指南.
> 有关更多信息：
>
> - CodeDeploy在入门Amazon在中国（北京）区域
> - 中国地区 CodeDeploy 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

如果您依赖的 IAM 实例配置文件或服务角色是作为Amazon CloudFormation堆栈中，如果您删除该堆栈，还将删除所有 IAM 角色。这可能是 IAM 角色不再在 IAM 控制台中显示以及 CodeDeploy 将不再按预期运行的原因。要解决此问题，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要在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的 AppSpec 文件或 Shell 脚本文件，则依赖这些文件的任何部署均可能失败。当 CodeDeploy 在部署过程中使用这些文件时，存在这些字符可能导致难以排查 AppSpec 文件验证故障和脚本执行故障。

在 CodeDeploy 控制台中，选择在部署的事件详细信息页上，选择查看日志.（或者您将Amazon CLI调用获取部署实例命令。）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：

- 请不要使用将非打印字符（例如，回车 `^M` 字符）引入 AppSpec 文件和 Shell 脚本文件中的文本编辑器。
- 使用在您的 AppSpec 文件和 Shell 脚本文件中显示非打印字符（例如回车）的文本编辑器，以便查找并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 服务器或 RHEL 实例上运行的文本编辑器来创建在 Amazon Linux、Ubuntu 服务器或 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 服务器或 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) 应用程序将某个 AppSpec 文件及相关文件及脚本捆绑（压缩）到一个应用程序修订存档 (.zip) 文件中，则部署可能会失败。这是因为 Finder 创建了一个中间`__MACOSX`文件夹放入 .zip 文件中，然后将组件文件放入该文件夹中。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Amazon CLI调用push命令，将组件文件压缩为预期结构。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。

EC2/本地部署问题

主题

- CodeDeploy 代理无法在 Windows 2016 上启动 (p. 365)
- CodeDeploy 插件 CommandPoller 缺失凭证错误 (p. 365)
- 部署失败，显示消息“PKCS7 签名的消息验证失败” (p. 366)

- 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” (p. 366)
- 长时间运行的进程可能会导致部署失败 (p. 367)
- 排查 AllowTraffic 生命周期事件失败，但部署日志中未报错的问题 (p. 368)
- 排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)
- 排查失败的 DownloadBundle 部署生命周期事件的问题，错误为“UnknownError: not opened for reading” (p. 369)
- 排查所有生命周期事件跳过错误 (p. 369)
- 默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell (p. 370)

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，我们建议您使用 Amazon CloudWatch Logs 集中监控日志文件，而不是逐个实例查看这些文件。想要了解有关信息，请参阅在 CloudWatch Logs 控制台中查看 CodeDeploy.

CodeDeploy 代理无法在 Windows 2016 上启动

在运行 CodeDeploy 代理安装程序文件时 (`CodeDeploy-agent.msi`) 在 Windows 2016 上，.msi 文件无法启动，你会在中看到以下错误`C:\Windows\TEMP\CodeDeploy-agent.msi_installer.log`：

```
Error 1920 Service 'CodeDeploy Host Agent Service' (CodeDeployagent) failed to
start. Verify that you have sufficient privileges to start system services.
```

出现此错误的原因是 Windows Defender 防病毒软件阻止`CodeDeploy-agent.msi`文件无法运行。

要解决此问题，您必须在运行代理安装程序之前添加 Windows Defender 排除项。

添加 Windows Defender 排除项

1. 使用以下 Powershell 语句添加排除项。确保以管理员身份运行它们。

```
Write-Host ("Adding Windows Defender exclude for CodeDeploy...")
Add-MpPreference -ExclusionPath ("C:\ProgramData\Amazon\CodeDeploy","$env:windir\Temp")
```

2. 在代理安装之前添加排除项，作为自定义 AMI 的一部分或使用 Systems Manager 添加排除项。如果要将代理安装为用户数据的一部分，请在代理安装脚本之前一步添加排除项。

   有关更多信息，请参阅 。安装适用于 Windows 服务器的 CodeDeploy 代理 (p. 156)、使用 Systems Manager Run Command 运行命令, 和启动时在 Windows 实例上运行命令.

CodeDeploy 插件 CommandPoller 缺失凭证错误

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 CodeDeploy 通信以及从 Amazon S3 下载修订的权限。对于 EC2 实例，请参阅 适用于 Amazon CodeDeploy 的 Identity and Access Management (p. 297)。对于本地实例，请参阅Working with On-Premises Instances (p. 179)。

部署失败，显示消息“PKCS7 签名的消息验证失败”

此错误消息指示实例正在运行仅支持 SHA-1 哈希算法的 CodeDeploy 代理版本。对 SHA-2 哈希算法的 Support 已在 2015 年 11 月发布的 CodeDeploy 代理 1.0.1.854 版（在 2015 年 11 月发布）中引入的。自 2016 年 10 月 17 起，如果安装了低于 1.0.1.854 版本的 CodeDeploy 代理，部署将会失败。有关更多信息，请参阅 。Amazon切换为 SSL 证书的 SHA256 哈希算法、请注意：正在退出版本 1.0.1.85 之前的 CodeDeploy 主机代理, 和更新 CodeDeploy 代理 (p. 158).

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，但指定目标位置已存在同名文件时，针对该实例的部署可能会失败。您可能会收到错误消息“The deployment failed because a specified file already exists at this location: *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此清除文件中未列出的文件，则默认情况下，CodeDeploy 代理会将其视为错误，并且部署将失败。

> Note
>
> 在 Amazon Linux、RHEL 和 Ubuntu 服务器实例上，清除文件位于`/opt/CodeDeploy-agent/deployment-root/deployment-instructions/`. 在 Windows 服务器实例上，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在生产环境中，并且您首次需要使用 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）(p. 258)和对现有内容进行回滚行为 (p. 272)。

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。

当您尝试重新部署名称和位置都相同的文件时，如果指定应用程序名称和具有之前使用的相同基础部署组 ID 的部署组，则重新部署更可能成功。CodeDeploy 使用底层部署组 ID 来识别在重新部署之前要删除的文件。

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。重新部署将失败，因为即使部署组名称相同，CodeDeploy 将引用不同的基础部署组 ID。
- 您在 CodeDeploy 中删除了某个应用程序和部署组，然后创建了与已删除的应用程序和部署组同名的应用程序和部署组。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，因为即使应用程序和部署组的名称相同，CodeDeploy 仍将引用已删除的部署组的 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。第二次部署将失败，因为 CodeDeploy 在第二次部署开始之前，将不会删除现有文件。两个部署都将引用不同的部署组 ID。
- 您在 CodeDeploy 中部署了一个修订，但至少有一个文件具有相同名称且位于相同位置。部署将失败，因为默认情况下，CodeDeploy 在部署开始之前，将不会删除现有文件。

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。
- 在修订的 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任何位置中与您的修订即将安装的文件匹配的文件。
- 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。
- 在您删除应用程序或部署组之前，部署一个包含未指定要复制到实例的文件的 AppSpec 文件的修订。对于该部署，指定使用您即将删除的基础应用程序和部署组的 ID 的应用程序名称和部署组名称。（您可以将获取部署组命令来检索部署组 ID。）CodeDeploy 使用基础部署组 ID 和 AppSpec 文件删除它在之前的成功部署中安装的所有文件。

长时间运行的进程可能会导致部署失败

对于至 Amazon Linux、Ubuntu Server 和 RHEL 实例的部署，如果您的部署脚本启动了长时间运行的进程，CodeDeploy 可能会在部署生命周期事件中等待较长的时间，然后使部署失败。这是因为，如果进程运行的时间比该事件中的前台和后台进程的预期所需时间长，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 文件包含以下说明：

```
version: 0.0
os: linux
files:
  - source: ./sleep.sh
    destination: /tmp
hooks:
  AfterInstall:
    - location: after-install.sh
      timeout: 60
```

`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
```

何时`after-install.sh`调用`sleep.sh`、`sleep.sh`启动并运行 3 分钟 (180 秒)，它比 CodeDeploy 预期的时间晚 2 分钟 (120 秒)。`sleep.sh`（而且，按照关系，`after-install.sh`) 停止运行。在超时 1 分钟（60 秒）后，CodeDeploy 将在 AfterInstall 应用程序生命周期事件处停止部署并使其失败，即使`sleep.sh`继续按预期运行。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 &
```

这样做可能使部署在长达 1 小时（默认值）的部署生命周期事件超时时段内保持挂起状态，随后 CodeDeploy 将像之前一样在 AfterInstall 应用程序生命周期事件处停止部署并使其失败。

In`after-install.sh`调用`sleep.sh`如下所示，后者支持 CodeDeploy 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`。

排查 AllowTraffic 生命周期事件失败，但部署日志中未报错的问题

在某些情况下，在 AllowTraffic 生命周期事件的过程中，蓝/绿部署会失败，但部署日志未指明失败的原因。

此失败通常是由于用于管理部署组流量的 Classic Load Balancer、应用程序负载均衡器或 Network Load Balancer 的弹性负载均衡器中配置了错误的运行状况检查。

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，请参阅配置运行状况检查中的Classic Load Balancer 用户指南和ConfigureHealthCheck中的Elastic Load Balancing API 参考版本 2012-06-01.

有关应用程序负载均衡器，请参阅目标组的运行状况检查中的适用于应用程序负载均衡器的用户指南.

有关网络负载均衡器，请参阅目标组的运行状况检查中的Network Load Balancer 用户指南.

排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署

在部署期间，CodeDeploy 代理运行在之前的成功部署中的 AppSpec 文件中为 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 指定的脚本。（在当前部署中，所有其他脚本从 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 代理会找到*`deployment-group-id`*`_last_successful_install`文件位于正确位置，但是*`deployment-group-id`*`_last_successful_install`文件不存在。

  在 Amazon Linux、Ubuntu 服务器和 RHEL 实例上，此文件必须存在于`/opt/codedeploy-agent/deployment-root/deployment-instructions`.

  在 Windows Server 实例上，此文件必须存储在`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions`folder。

- 在中列出的位置*deployment-group-id*_last_successful_install文件中，AppSpec 文件无效或脚本无法成功运行。
- 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 控制台调查部署在这些事件中的任一事件期间失败的可能原因。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上，在 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 行中，选择查看日志。或者使用Amazon CLI调用获取部署实例命令。

如果失败的原因是上一个成功部署中从未成功运行的脚本，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。在创建部署页上的 ApplicationStop 生命周期事件失败下，选择如果实例上的此生命周期事件失败，就不要将此部署到实例上。
- 使用 Amazon CLI 调用 create-deployment 命令并包含 `--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。

排查失败的 DownloadBundle 部署生命周期事件的问题，错误为“UnknownError: not opened for reading”

如果您正在尝试从 Amazon S3 部署应用程序修订，并且部署在 DownloadBundle 部署生命周期事件期间失败，`UnknownError: not opened for reading`错误：

- Amazon S3 服务发生内部错误。请重新部署应用程序修订。
- 您的 EC2 实例上的 IAM 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，请参阅将对于 CodeDeploy 的修订推送到 Amazon S3（仅限 EC2/ 本地部署）(p. 247)和部署先决条件(p. 254).
- 您部署到的实例与一个实例关联Amazon区域（例如，美国西部（俄勒冈）），但包含应用程序修订的 Amazon S3 存储桶与另一个存储桶关联。Amazon区域（例如，美国东部（弗吉尼亚北部））。请确保存储桶中的应用程序修订与相同的存储 Amazon S3 关联Amazon区域作为实例。

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调用获取部署实例命令。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记录文件以及启用和禁用线路日志记录的信息，请参阅`:log_aws_wire:`在CodeDeploy 代理配置参考.

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，您可能会收到类似于的错误。`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)`. 这里介绍了一些可能的原因和解决方案：

- 实例上可能未安装或运行 CodeDeploy 代理。要确定 CodeDeploy 代理是否正在运行：
  - 对于 Amazon Linux RHEL 或 Ubuntu 服务器，请运行以下命令：

    ```
    sudo service CodeDeploy-agent status
    ```

  - 对于 Windows，请运行以下命令：

    ```
    powershell.exe -Command Get-Service -Name CodeDeployagent
    ```

  如果未安装或运行 CodeDeploy 代理，请参阅验证代 CodeDeploy 运行 (p. 149).

  使用端口 443，您的实例可能无法访问 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 (p. 142)。
  - 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。
- CodeDeploy 的服务角色可能没有所需权限。要配置 CodeDeploy 服务角色，请参阅第 2 步：为 CodeDeploy 创建服务角色 (p. 26).
- 如果您使用 HTTP 代理，请在`:proxy_uri:`在 CodeDeploy 代理配置文件中的设置。有关更多信息，请参阅CodeDeploy 代理配置参考 (p. 350)。
- 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。查找类似于`Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature expired`在你的 CodeDeploy 代理日志文件中。如果您看到此错误，请按照排查“InvalidSignatureException – Signature expired: [time] is now earlier than [time]”部署错误 (p. 373)中的步骤进行操作。有关更多信息，请参阅查看代表部署EC2/本地部署的日志数据 (p. 268)。
- CodeDeploy 代理可能会由于实例内存或硬盘空间不足而停止运行。尝试更新实例上的存档部署数量。`max_revisions`在 CodeDeploy 代理配置中设置。如果您为 EC2 实例执行了此操作但问题仍然存在，请考虑使用较大的实例。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 。由 CodeDeploy 代理安装的文件 (p. 147)、CodeDeploy 代理配置参考 (p. 350), 和Amazon EC2 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 实例配置文件未附加到您的实例，请创建具有所需权限的实例配置文件，然后附加。
  - 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 。第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)和适用于 Amazon EC2 的 IAM 角色中的Amazon EC2 用户指南.

默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell

如果作为部署的一部分运行的某个 Windows PowerShell 脚本依赖 64 位功能（例如，因为它占用的内存多于 32 位应用程序允许的内存，或调用的库仅在 64 位版本中提供），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 使用 32 位版本的 Windows PowerShell 运行作为应用程序修订的一部分的 Windows PowerShell 脚本。

将类似于以下内容的代码添加到任何必须使用 64 位版本的 Windows PowerShell 运行的脚本的开头：

```
# 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似乎违反直觉，但 32 位 Windows PowerShell 使用与以下内容类似的路径：

```
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

64 位 Windows PowerShell 使用与以下内容类似的路径：

```
c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

故障排除AmazonLambda 部署问题

主题

- Amazon Lambda在手动停止未配置回滚的 Lambda 部署后，部署失败 (p. 371)

Amazon Lambda在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 控制台确保函数不配置为将在两个版本之间转移流量：

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在左侧窗格中，选择函数。
3. 选择位于您的 CodeDeploy 部署中的 Lambda 函数的名称。
4. 从限定词中，选择在 CodeDeploy 部署中使用的别名。
5. 从其他版本中，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. 选择保存并测试。
7. 打开 CodeDeploy 控制台并尝试部署在步骤 5 中的下拉菜单内显示的版本。

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 CodeDeploy 实现到 Amazon EC2 Auto Scaling 组中的新 EC2 实例的自动部署。有关更多信息，请参阅将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39)。

排查实例问题

主题

- 必须正确设置标签 (p. 372)
- Amazon CodeDeploy实例上必须安装并运行代理 (p. 372)
- 如果实例在部署期间终止，在最多 1 小时内部署不会失败。 (p. 372)
- 分析日志文件以调查针对实例的部署失败 (p. 373)
- 如果已意外删除，请创建一个新的 CodeDeploy 日志文件 (p. 373)
- 排查“InvalidSignatureException – Signature expired: [time] is now earlier than [time]”部署错误 (p. 373)

必须正确设置标签

使用列出部署实例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 。使用控制台中的标签中的适用于 Linux 实例的 Amazon EC2 用户指南.

> Note
>
> 如果您标记一个实例并立即使用 CodeDeploy 将一个应用程序部署到该实例，则该实例可能不会包括在部署中。这是因为可能要在几分钟后才能读取标签。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实例上必须安装并运行代理

要验证 CodeDeploy 代理是否已安装并在实例上运行，请参阅验证代 CodeDeploy 运行 (p. 149).

要安装、卸载或重新安装 CodeDeploy 代理，请参阅安 CodeDeploy (p. 151).

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 1 小时的时段来完成。这为长时间运行的脚本提供了足够的时间。

如果生命周期事件正在进行时脚本未运行完成（例如，如果实例终止或 CodeDeploy 代理关闭），部署状态可能需 1 小时后才显示为失败。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后，CodeDeploy 代理将关闭，并且无法处理更多脚本。

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，请参阅查看代表部署EC2/本地部署的日志数据 (p. 268)。

如果已意外删除，请创建一个新的 CodeDeploy 日志文件

如果您意外删除了实例的部署日志文件，CodeDeploy 将不会创建替代日志文件。要创建新的日志文件，请登录到实例，然后运行以下命令：

对于亚马逊 Linux、Ubuntu 服务器或 RHEL 实例，按此顺序运行以下命令（一次运行一个）：

```
sudo service codedeploy-agent stop
```

```
sudo service codedeploy-agent start
```

对于 Windows Server 实例：

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

排查“InvalidSignatureException – Signature expired: [time] is now earlier than [time]”部署错误

CodeDeploy 需要准确的时间引用以执行其操作。如果实例上的日期和时间设置不正确，它们可能与 CodeDeploy 拒绝的部署请求的签名日期不匹配。

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

GitHub 令牌问题排查

无效 GitHub OAuth 令牌

2017 年 6 月之后创建的 CodeDeploy 应用 GitHub 序在每个应用程序中均使用Amazon区域。使用与特定关系的代币Amazon使您能够更好地控制哪些 CodeDeploy 应用程序可以访问 GitHub 存储库。

如果您收到 GitHub 令牌错误，可能是因为您拥有的令牌较旧，不再有效。

修复无效的 GitHub OAuth 令牌

1. 使用以下方法之一删除旧令牌：
   - 要使用 API 删除旧令牌，请使用DeleteGitHubAccountToken.
   - 使用删除旧令牌Amazon Command Line Interface：
     a. 转到令牌所在的计算机。
     b. 确保Amazon CLI已安装在此计算机上。有关安装说明，请参阅安装、更新和卸载Amazon CLI中的Amazon Command Line Interface用户指南
     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，请参阅删除 git-hub 账户令牌.
2. 添加新的 OAuth 令牌。有关更多信息，请参阅GitHub 将 CodeDeploy 集成 (p. 47)。

超出了 GitHub OAuth 令牌数量上限

您在创建 CodeDeploy 部署时，允许的 GitHub 令牌数量上限为 10 个。如果您收到 GitHub OAuth 令牌错误，请确保令牌的数量少于或等于 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 删除旧令牌的信息，请参阅DeleteGitHubAccountToken.

解决 Amazon EC2 Auto Scaling 问题

主题

- Amazon EC2 Auto Scaling 常规故障排除 (p. 374)
- “CodeDeploRole 没有授予您执行以下操作的权限：Amazon服务：AmazonAutoScaling” 错误消息 (p. 375)
- 在部署修订之前，Amazon EC2 Auto Scaling 组中的实例不断被预置和终止 (p. 375)
- 终止或重启 Amazon EC2 Auto Scaling 实例可能会导致部署失败 (p. 376)
- 避免将多个部署组与一个 Amazon EC2 Auto Scaling 组关联 (p. 376)
- Amazon EC2 Auto Scaling 组中的 EC2 实例无法启动，收到错误 “心跳超时” (p. 377)
- 不匹配的 Amazon EC2 Auto Scaling 生命周期挂钩可能导致至 Amazon EC2 Auto Scaling 组的自动部署停止或失败。 (p. 378)
- “部署失败是因为没有为您的部署组找到实例” 错误 (p. 379)

Amazon EC2 Auto Scaling 常规故障排除

到 Amazon EC2 Auto Scaling 组中的 EC2 实例的部署可能因以下原因而失败：

- Amazon EC2 Auto Scaling 实例会持续启动和终止。如果 CodeDeploy 无法自动部署应用程序修订，Amazon EC2 Auto Scaling 将持续启动和终止 EC2 实例。

  解除 Amazon EC2 Auto Scaling 组与 CodeDeploy 部署组的关联或更改 Amazon EC2 Auto Scaling 组的配置，使所需的实例数量与当前实例数量匹配（从而防止 Amazon EC2 Auto Scaling 启动任何 EC2 实例）。有关更多信息，请参阅 。使用 CodeDeploy 更改部署组设置 (p. 235)或者Amazon EC2 Auto Scaling 的手动扩展.

- CodeDeploy 代理没有响应。如果 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）需要 1 个小时以上的时间才能运行，则可能无 CodeDeploy 安装。CodeDeploy 代理有 1 小时的超时以便代理响应挂起的部署。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。
- 在部署期间，Amazon EC2 Auto Scaling 组中的 EC2 实例会重启。如果 EC2 实例在部署期间重启或 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 EC2 Auto Scaling 实例可能会导致部署失败 (p. 376)。
- 同时向 Amazon EC2 Auto Scaling 组中的相同 EC2 实例部署多个应用程序修订。同时将多个应用程序修订部署到 Amazon EC2 Auto Scaling 组中的同一个 EC2 实例，如果其中一个部署具有运行几分钟以上的脚本，则部署多个应用程序修订可能会失败。请勿将多个应用程序修订部署到 Amazon EC2 Auto Scaling 组中的相同 EC2 实例。
- 对于作为 Amazon EC2 Auto Scaling 组的一部分启动的新 EC2 实例，部署将失败。这种情况下，在部署中运行脚本可能会阻止 Amazon EC2 Auto Scaling 组中启动 EC2 实例。（Amazon EC2 Auto 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：
  - 您的 AMI 中不包含 CodeDeploy 代理：如果您将cfn-init请将代理安装脚本置于启动新实例时安装 CodeDeploy 代理，请将代理安装脚本置于`cfn-init`你的部分Amazon CloudFormation模板。
  - 您的 AMI 中包含 CodeDeploy 代理：配置 AMI 以便代理位于`Stopped`在创建实例时进行状态，然后在您的最后一步中包括一个将启动代理作为最后一步的脚本。`cfn-init`脚本库。

“CodeDeploRole 没有授予您执行以下操作的权限：Amazon服务：AmazonAutoScaling” 错误消息

使用启动模板创建的 Auto Scaling 组的部署需要以下权限。这些是除了 `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，请参阅 。教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88)、为 Auto Scaling 组创建启动模板, 和启动模板支持中的Amazon EC2 Auto Scaling 用户指南.

在部署修订之前，Amazon EC2 Auto Scaling 组中的实例不断被预置和终止

有时，一个错误可能导致无法在 Amazon EC2 Auto Scaling 组中成功部署新预置的实例。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。Amazon EC2 Auto Scaling 组配置将预置另一组实例，试图达到正常运行的主机数的最低要求。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 EC2 Auto Scaling 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 EC2 Auto Scaling 组的 EC2 实例。用唯一的 EC2 实例标签标记该实例。
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 EC2 Auto Scaling 组将此应用程序修订部署到 Amazon EC2 Auto Scaling 组中将来的实例。

> **Note**
>
> 确认部署成功后，请删除所创建的实例，避免继续向您收取相应费用。Amazonaccount.

终止或重启 Amazon EC2 Auto Scaling 实例可能会导致部署失败

如果通过 Amazon EC2 Auto Scaling 启动了 EC2 实例，然后终止或重启该实例，则至该实例的部署可能会因以下原因而失败：

- 在部署正在进行时，缩小事件或任何其他终止事件将导致实例与 Amazon EC2 Auto Scaling 组分离并终止。由于无法完成部署，因此它将失败。
- 实例已重启，但需要 5 分钟以上的时间才能启动。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 EC2 Auto Scaling 配置指定基于 Windows 服务器的亚马逊系统映像 (AMI)，并使用 EC2Config 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 服务器基础 AMI 中的常规的选项卡EC2 服务属性，清除Set Computer Name. 清除此复选框后，将对使用该 Windows 服务器基础 AMI 启动的所有新的 Windows Server Amazon EC2 Auto Scaling 实例禁用此行为。对于已启用此行为的 Windows Server Amazon EC2 Auto Scaling 实例，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一个 Amazon EC2 Auto Scaling 组关联

作为最佳实践，应仅为每个 Amazon EC2 Auto Scaling 组关联每个 Amazon EC2 Auto Scaling 组。

这是因为，如果 Amazon EC2 Auto Scaling 向上扩展一个具有与多个部署组关联的钩子的实例，它将一次性为所有钩子发送通知。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前一个生命周期事件结束之间的五分钟超时值。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**
>
> 在生命周期事件中，脚本的默认超时时间为 30 分钟。您可以在 AppSpec 文件中将超时更改为其他值。有关更多信息，请参阅为 EC2/本地部署部署添加AppSpec 文件 (p. 243)。

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，如果至任一实例的部署失败，Amazon EC2 Auto Scaling 将立即终止该实例。当第一个实例关闭时，正在运行的其他部署将开始失败。由于 CodeDeploy 具有 1 个小时的超时时间，以便 CodeDeploy 代理响应挂起的部署，因此每个实例的最长超时时间为 60 分钟。

有关 Amazon EC2 Auto Scaling 的更多信息，请参阅在引擎盖下：CodeDeploy 和 Auto Scaling 集成.

Amazon EC2 Auto Scaling 组中的 EC2 实例无法启动，收到错误 “心跳超时”

Amazon EC2 Auto Scaling 组可能无法启动新的 EC2 实例，并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：

- 与一个关联的最大并发部署数量Amazon已到达账户。有关部署限制的更多信息，请参阅CodeDeploy 限制 (p. 359)。
- Auto Scaling 组试图太快启动太多 EC2 实例。将 API 调用到RecordLifecycleActionHeartbeat或者CompleteLifecycleAction对于每个新实例都受到限制。
- CodeDeploy 中的一个应用程序已在其关联的部署组更新或删除之前被删除。

  当您删除某个应用程序或部署组时，CodeDeploy 将尝试清除与其关联的任何 Amazon EC2 Auto Scaling 挂钩，但某些挂钩可能会保留。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. 请执行下列操作之一：

   - 调用删除部署组命令可删除与 Auto Scaling 组关联的部署部署，导致检测信号超时。
   - 调用更新部署组命令带有 Auto Scaling 组名称的非空列表，以分离所有 CodeDeploy 管理的 Auto Scaling 生命周期挂钩。

     例如，输入以下命令Amazon CLI命令：

     ```
     aws deploy update-deployment-group --application-name my-example-app --
     current-deployment-group-name my-deployment-group --auto-scaling-groups
     ```

     作为另一个例子，如果你将 CodeDeploy API 与 Java 结合使用，请调用`UpdateDeploymentGroup`并设置`autoScalingGroups`到`new ArrayList<String>()`. 这套`autoScalingGroups`转换为空列表，然后删除现有列表。请勿使用`null`，这是默认设置，因为这会离开`autoScalingGroups`原样，这不是你想要的。

   检查调用的输出。如果输出包含`hooksNotCleanedUp`结构包含 Amazon EC2 Auto Scaling 生命周期钩子列表，有剩余的生命周期钩子。
2. 调用描述生命周期挂钩命令，指定与启动失败的 EC2 实例关联的 Amazon EC2 Auto Scaling 组的名称。在输出中，查找以下内容之一：

   - 对应的 Amazon EC2 Auto Scaling 生命周期挂钩名称。`hooksNotCleanedUp`您在步骤 1 中确定的结构。
   - Amazon EC2 Auto 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。
   - Amazon EC2 Auto Scaling 生命周期挂钩名称可能导致 CodeDeploy 部署的心跳超时。

3. 如果挂钩属于步骤 2 中列出的其中一个类别，请调用删除生命周期挂钩命令删除它。在调用中指定 Amazon EC2 Auto Scaling 组和生命周期挂钩。

   Important

   如步骤 2 所述，只删除导致问题的钩子。如果删除可行的挂钩，部署可能会失败，或者 CodeDeploy 可能无法部署应用程序修订以扩展 EC2 实例。

4. 打电话给更新部署组或者创建部署组带有所需的 Auto Scaling 组名称的命令。CodeDeploy 使用新的 UUID 重新安装 Auto Scaling 挂钩。

   Note

   如果您将 Auto Scaling 组与 CodeDeploy 部署组分离，则对 Auto Scaling 组的任何正在进行的部署都可能会失败，并且 Auto Scaling 组扩展的新 EC2 实例将不会从 CodeDeploy 收到您的应用程序修订版本。要使 Auto Scaling 再次与 CodeDeploy 一起工作，您需要将 Auto Scaling 组重新附加到部署组并调用一个新的`CreateDeployment`开始整个车队的部署。

不匹配的 Amazon EC2 Auto Scaling 生命周期挂钩可能导致至 Amazon EC2 Auto Scaling 组的自动部署停止或失败。

Amazon EC2 Auto Scaling 和 CodeDeploy 使用生命周期挂钩确定应将哪些应用程序修订部署到哪些已在 Amazon EC2 Auto Scaling 组中启动的 EC2 实例。如果 Amazon EC2 Auto Scaling 和 CodeDeploy 中的生命周期挂钩及其相关信息未准确匹配，则自动部署可能会停止或失败。

如果至 Amazon EC2 Auto Scaling 组的部署失败，请检查 Amazon EC2 Auto Scaling 和 CodeDeploy 中的生命周期挂钩名称是否匹配。如果不匹配，请使用这些 Amazon CLI 命令调用。

首先，获取 Amazon EC2 Auto Scaling 组和部署组的生命周期挂钩名称的列表：

1. 调用描述生命周期挂钩命令中指定与 CodeDeploy 中的部署组关联的部署 Amazon EC2 Auto Scaling 组的名称。在输出中，在 `LifecycleHooks` 列表中，记下每个 `LifecycleHookName` 值。
2. 调用获取部署组命令中指定与 Amazon EC2 Auto Scaling 组关联的部署部署部署部署部署部署组的名称。在输出中，在`autoScalingGroups`列表中，查找名称值与 Amazon EC2 Auto 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。您可能需要尝试本部分中的其他位置描述的其他 Amazon EC2 Auto Scaling 排查步骤。

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 delete-lifecycle-hook 命令。

   b. 调用更新部署组命令，指定原始 Amazon EC2 Auto Scaling 组的名称。CodeDeploy 在 Amazon EC2 Auto Scaling 组中创建新的替代生命周期挂钩，并将这些生命周期挂钩与部署组关联。现在，自动部署应恢复，因为新的实例已添加到 Amazon EC2 Auto Scaling 组。

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. 调用更新部署组命令，但不指定原始 Amazon EC2 Auto Scaling 组的名称。

b. 调用update-deployment-group再次指定原始 Amazon EC2 Auto Scaling 组的名称。CodeDeploy 在 Amazon EC2 Auto Scaling 组中重新创建缺失的生命周期挂钩。现在，自动部署应恢复，因为新的实例已添加到 Amazon EC2 Auto Scaling 组。

在您将两组生命周期挂钩名称完全匹配后（字符对字符）后，应用程序修订应重新部署，但仅重新部署到新的实例，因为它们已添加到 Amazon EC2 Auto Scaling 组。部署不会在 Amazon EC2 Auto Scaling 组中已存在的实例上自动进行。

“部署失败是因为没有为您的部署组找到实例” 错误

如果你看到以下 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，请检查标签是否正确，然后重新部署应用程序。
2. 部署开始后，您的车队已扩展。在这种情况下，您将在`InService`你的车队中的状态，但你也会看到上面的错误。要解决此问题，请重新部署您的应用程序。
3. 您的 Auto Scaling 组不包括`InService`状态。在这种情况下，当您尝试在队列范围内进行部署时，部署将失败并显示上面的错误消息，因为 CodeDeploy 至少需要一个实例才能在`InService`状态。你可能没有实例的原因有很多`InService`状态。其中一些包括：
   - 您已计划（或手动配置）Auto Scaling 组大小为`0`.
   - Auto Scaling 检测到坏的 EC2 实例（例如，EC2 实例出现硬件故障），因此全部取消了它们，在`InService`状态。
   - 在扩展活动期间`0`到`1`，CodeDeploy 部署了之前成功的修订版（称为上次成功修订）自上次部署以来，这已经变得不健康。这导致扩展实例上的部署失败，这反过来又导致 Auto Scaling 取消该实例，在`InService`状态。

     如果你发现自己在`InService`状态，请按照以下过程中的描述对问题进行故障排除，To troubleshoot the error if there are no instances in the InService state.

如果没有实例处于 inService 状态，请对错误进行故障排除

1. 在 Amazon EC2 控制台中，验证所需容量设置设置。如果为零，请将其设置为正数。等待实例`InService`，这意味着部署成功。您已修复了此问题，可以跳过此故障排除过程的剩余步骤。有关设置所需容量设置，请参阅为 Auto Scaling 组设置容量限制中的Amazon EC2 Auto Scaling 用户指南.
2. 如果 Auto Scaling 继续尝试启动新的 EC2 实例以满足所需容量，但无法实现扩展，则通常是由于 Auto Scaling 生命周期挂钩失败造成的。如下所示解决此问题：

   a. 要检查哪个 Auto Scaling 生命周期挂钩事件失败，请参阅验证 Auto Scaling 组的扩展活动在 Amazon EC2 Auto Scaling 用户指南中。

   b. 如果失败的挂钩的名字是`CodeDeploy-managed-automatic-launch-deployment-hook-`*`DEPLOYMENT_GROUP_NAME`*，转到 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，你可以确定 CodeDeploy上次成功修订不再运行正常，您的 Auto Scaling 组中没有运行状况良好的实例，因此您处于部署死锁状态。要解决此问题，必须通过从 Auto Scaling 组中

临时删除 CodeDeploy 的生命周期钩子，然后重新安装挂接并重新部署新的（好）修订版来修复（坏）CodeDeploy 修订版本。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）阻止导致 CodeDeploy 错误的 CI/CD 管道，以便在修复此问题时不会发生意外部署。
2. 记下您当前的 Auto ScalingDesiredCapacity设置设置：

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name
   ASG_NAME
   ```

   在此过程结束时，您可能需要缩减为此数字。
3. 设置 Auto ScalingDesiredCapacity将设置为1. 如果您想要的容量大于，则这是可选的1以开头。通过将其减少到1，稍后预配和部署实例所需的时间减少，从而加快了故障排除速度。如果您的 Auto Scaling 所需容量最初设置为0，你必须将其增加到1. 这是强制要求的。

   aws 自动扩展集所需容量 — 自动扩展组名称*ASG_NAME*— 所需容量 1

   > **Note**
   >
   > 此过程的剩余步骤假设你已经设置了DesiredCapacity到1.

   此时，Auto Scaling 尝试扩展到一个实例。然后，由于 CodeDeploy 添加的钩子仍然存在，因此 CodeDeploy 会尝试部署，部署失败，Auto Scaling 会取消实例，Auto Scaling 尝试重新启动实例以达到所需容量的实例，这再次失败。你正处于取消重新启动循环中。
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 Auto ScalingInService可以接受运行任何软件的实例。例如，确保负载均衡器没有软件不会向此主机发送流量。
   >
   > **Important**
   >
   > 使用下面显示的 CodeDeploy 命令删除挂接。不要通过 Auto Scaling 服务删除挂钩，因为 CodeDeploy 无法识别删除操作。

   ```
   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，会出现以下情况：

   a. CodeDeploy 从部署组中注销 Auto Scaling 组的注册。
   b. CodeDeploy 将 Auto Scaling 组中删除 Auto Scaling 生命周期挂钩。
   c. 由于导致部署失败的挂钩不再存在，因此 Auto Scaling 将取消现有的 EC2 实例，并立即启动一个新实例以扩展到所需容量。新的实例应该很快移动到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 EC2 实例：

> **Important**
>
> 使用下面显示的 CodeDeploy 命令添加挂接。不要使用 Auto Scaling 服务添加挂接，因为 CodeDeploy 无法识别添加该挂钩。

```
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，会出现以下情况：

a. CodeDeploy 将 Auto Scaling 生命周期钩子重新安装到 EC2 实例

b. CodeDeploy 将 Auto Scaling 组重新注册到部署组。

7. 使用您知道健康且想要使用的 Amazon S3 或 GitHub 修订版创建队列范围内的部署。

   例如，如果修订是 Amazon S3 存储桶中的 .zip 文件，名为`my-revision-bucket`使用对象密钥`httpd_app.zip`中，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}"
```

   因为现在有一个`InService`实例在 Auto Scaling 组中，此部署应该起作用，您应该不再看到错误部署失败，原因是没有为您的部署组找到实例.

8. 部署成功后，如果您之前将 Auto Scaling 组扩展到原始容量，如果您之前将其扩展到原始容量：

```
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）阻止导致 CodeDeploy 错误的 CI/CD 管道，以便在修复此问题时不会发生意外部署。
2. 转到 Amazon EC2 控制台，然后记下您的 Auto Scaling所需容量设置设置。在此过程结束时，您可能需要缩减为此数字。有关查找此设置的信息，请参阅为 Auto Scaling 组设置容量限制.
3. 将所需的 EC2 实例数设置为`1`：

   如果您想要的容量大于，则这是可选的`1`以开头。通过将其减少到`1`，稍后预配和部署实例所需的时间减少，从而加快了故障排除速度。如果您的 Auto Scaling所需容量最初设定为`0`，你必须将其增加到1. 这是强制要求的。

   > **Note**
   >
   > 此过程的剩余步骤假设你已经设置了所需容量到1.

   a. 在以下网址打开 Amazon EC2 Auto Scaling 控制https://console.aws.amazon.com/ec2autoscaling/.

   b. 选择适当的区域。

   c. 转到有问题的 Auto Scaling 组。

   d. In组详细信息，选择编辑.

   e. Set所需容量到**1**.

   f. 选择 Update。

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 Auto ScalingInService可以接受运行任何软件的实例。例如，确保负载均衡器没有软件不会向此主机发送流量。

a. 从 https://console.aws.amazon.com/codedeploy/ 打开 CodeDeploy 控制台。
b. 选择适当的区域。
c. 在导航窗格中，选择 Applications (应用程序)。
d. 选择 CodeDeploy 应用程序的名称。
e. 选择 CodeDeploy 部署组的名称。
f. 选择 Edit。
g. In环境配置，取消选择Amazon EC2 Auto Scaling 组.

Note

如果没有定义环境配置，控制台不允许您保存配置。要绕过支票，请临时添加EC2或者On-premises你知道不会解决任何房东。要添加标签，请选择Amazon EC2 实例或者本地实例，然后添加标签密钥的**EC2**或者**On-premises**. 你可以留下标签值空。

h. 选择保存更改。

完成这些子步骤后，将出现以下情况：

i. CodeDeploy 从部署组中注销 Auto Scaling 组的注册。
ii. CodeDeploy 将 Auto Scaling 组中删除 Auto Scaling 生命周期挂钩。
iii. 由于导致部署失败的挂钩不再存在，因此 Auto Scaling 将取消现有的 EC2 实例，并立即启动一个新实例以扩展到所需容量。新的实例应该很快移动到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：

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，选择 Auto Scaling Groups。
c. 选择 Auto Scaling 组。
d. 在内容窗格中，选择实例管理选项卡。
e. 在实例，请确保生命周期列表示InService在实例旁边。

6. 使用与删除 Auto Scaling 组相同的方法将 Auto Scaling 组重新注册到 CodeDeploy 部署组：

a. 从 https://console.aws.amazon.com/codedeploy/ 打开 CodeDeploy 控制台。
b. 选择适当的区域。
c. 在导航窗格中，选择 Applications (应用程序)。
d. 选择 CodeDeploy 应用程序的名称。
e. 选择 CodeDeploy 部署组的名称。
f. 选择 Edit。
g. In环境配置，选择Amazon EC2 Auto Scaling 组然后从列表中选择您的 Auto Scaling 组。
h. 在Amazon EC2 实例或者本地实例，找到你添加的标签并将其删除。
i. 取消选中旁边的复选框Amazon EC2 实例或者本地实例.
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. 使用您知道健康且想要使用的 Amazon s3or GitHub 修订版创建队列范围内的部署。

例如，如果修订是 Amazon S3 存储桶中的 .zip 文件，名为`my-revision-bucket`使用对象密钥`httpd_app.zip`中，执行以下操作：

a. 在 CodeDeploy 控制台中，在部署组页面上，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. 适用于修订位置，选择`s3://my-revision-bucket/httpd_app.zip`.
d. 适用于修订文件类型，选择`.zip`.
e. 选择 Create deployment (创建部署)。

因为现在有一个`InService`实例在 Auto Scaling 组中，此部署应该起作用，您应该不再看到错误部署失败，原因是没有为您的部署组找到实例.

8. 部署成功后，如果您之前将 Auto Scaling 组扩展到原始容量，如果您之前将其扩展到原始容量：

a. 在以下网址打开 Amazon EC2 Auto Scaling 控制https://console.aws.amazon.com/ec2autoscaling/.
b. 选择适当的区域。
c. 转到 Auto Scaling 组。
d. In组详细信息，选择编辑.
e. Set所需容量回到其原始值。
f. 选择 Update。

的错误代码Amazon CodeDeploy

本主题提供有关 CodeDeploy 错误的参考信息。

错误代码	描述
`AGENT_ISSUE`	由于 CodeDeploy 代理出现问题，因此部署失败。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： • 验证代 CodeDeploy 运行 (p. 149) • 安 CodeDeploy (p. 151) • 使用 CodeDeploy 代理 (p. 141)
`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。 了解更多： • 第 2 步：为 CodeDeploy 创建服务角色 (p. 26) • 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 (p. 203)

错误代码	描述
	• 排查实例问题 (p. 372) • EC2/本地部署问题 (p. 364)
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。 了解更多： • Instance Health (p. 203) • 使用 CodeDeploy 的实例 (p. 162)
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： • 第 2 步：为 CodeDeploy 创建服务角色 (p. 26) • 使用 CodeDeploy 更改部署组设置 (p. 235)
IAM_ROLE_PERMISSIONS	CodeDeploy 不具备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。 了解更多： • 第 3 步：预配 IAM 用户 (p. 30) • 第 2 步：为 CodeDeploy 创建服务角色 (p. 26) • 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31)
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，则您的自动扩展组可能容量不足。请确保您的 Auto Scaling 组具有足够的容量用于部署。您可以通过使用 Amazon EC2 控制台查看运行正常的实例数，查看您的 Amazon EC2 Auto Scaling 组的容量。 • 对于 EC2/ 本地蓝/绿部署，蓝色和绿色机群的大小可能不同。请确保两个机群大小相同。 了解更多： • Tagging Instances for Deployments (p. 163) • 教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88) • 为蓝/绿部署创建应用程序（控制台）(p. 215)

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，您也可以联系Amazon Web Services Support来请求提高限制。 了解更多： • 使用 CodeDeploy 更改部署组设置 (p. 235) • CodeDeploy 限制 (p. 359) • 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	由于部署组未与其 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： • 深入析：CodeDeploy 和 Auto Scaling 集成

相关主题

排查 CodeDeploy (p. 362)

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 CodeDeployAPI 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支持中心— 用于创建和管理您的Amazon Web Services Support案例。还包括指向其他资源的链接，如论坛、技术常见问题、服务运行状况和Amazon Trusted Advisor。
- Amazon支持计划— 提供相关信息的主要网页Amazon Web Services SupportP划y。
- 联系我们— 用于查询相关信息的中心联络点Amazon账单、账户、事件、滥用和其他问题。
- Amazonsite 术语— 有关我们的版权和商标、您的账户、许可证和网站访问以及其他主题的详细信息。

Samples

- CodeDeploy 上的 CodeDeploy 示例— CodeDeploy 的示例和模板方案。
- CodeDeploy 插件— Jenkins 插件的 Jenkins 插 CodeDeploy。
- CodeDeploy 代理— CodeDeploy 代理的开源版本。

Blogs

- AmazonDevOps 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：

- Amazon SDK for .NET
- Amazon SDK for Java
- Amazon SDK for JavaScript
- Amazon SDK for PHP
- Amazon SDK for Python (Boto)
- Amazon SDK for Ruby
- Amazon Toolkit for Eclipse— 零件1、2, 和3。
- Amazon Tools for Windows PowerShell— 一组 Windows PowerShell cmdlet，用于在Amazon SDK for .NET在 PowerShell 环境中。
- CodeDeployAmazon Tools for PowerShell— 一组 Windows PowerShell cmdlet，用于在 PowerShell 环境中公开 CodeDeploy 的功能。
- Amazon Command Line Interface— 用于访问的一致的命令行语法Amazon服务。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。

文档历史记录

下表描述了对此用户指南进行的重大更改，以阐明自上次发布用户指南以来的新增及增强功能。CodeDeploy。

- API 版本：2014-10-06
- 最近文档更新：2021 年 5 月 18 日

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
CodeDeployAmazon管理的策略 (p. 388)	这些区域有：`AmazonCodeDeployRole`更新后的。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 (p. 388)	这些区域有：Amazon CodeDeploy代理已更新到版本 1.3.2。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2021 年 5 月 6 日
CodeDeploy 支持更新过时的 Amazon EC2 实例 (p. 388)	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 (p. 388)	这些区域有：Amazon CodeDeploy代理已更新到版本 1.3.1。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2020 年 12 月 22 日
CodeDeploy 代理 v1.3.0 版本 (p. 388)	这些区域有：Amazon CodeDeploy代理已更新到版本 1.3.0。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2020 年 11 月 10 日
CodeDeploy 代理 v1.2.1 版本 (p. 388)	这些区域有：Amazon CodeDeploy代理已更新到版本 1.2.1。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2020 年 9 月 23 日
CodeDeploy 支持由 Amazon PrivateLink (p. 388)	如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管您的Amazon资源后，您可以在 VPC 和 CodeDeploy 之间建立私有连接。您可以使用此连接实现 CodeDeploy 与 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 。将 CodeDeploy 与 Amazon Virtual Private Cloud 一起使用。	2020 年 8 月 11 日

更新了 CodeDeploy (p. 388)	将每个帐户的应用程序数量和每个应用程序的部署组数的限制更新为 1000。有关 CodeDeploy 服务限制的更多信息，请参阅CodeDeploy。	2020 年 8 月 6 日
CodeDeploy 代理 v1.1.2 版本 (p. 388)	这些区域有：Amazon CodeDeploy代理已更新到版本 1.1.2。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2020 年 8 月 4 日
CodeDeploy 代理 1.1.0 版本以及与 Amazon EC2 Systems Manager 的集成 (p. 388)	代理版本 1.1.0 现已推出，有关详细信息，请参阅CodeDeploy 代理的版本历史纪录。现在，您可以使用 Amazon EC2 系统管理器自动管理您的 Amazon EC2 或本地实例上代理的安装和更新。有关更多信息，请参阅 。使用 Amazon EC2 Systems Manager 安装 CodeDeploy 代理。	2020 年 6 月 30 日
CodeDeploy 支持使用管理 Amazon ECS 蓝/绿部署 Amazon CloudFormation (p. 388)	您现在可以使用Amazon CloudFormation通过 CodeDeploy 管理 Amazon ECS 蓝/绿部署。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 支持使用加权流量转移来进行 Amazon ECS 蓝/绿部署 (p. 388)	CodeDeploy 现在支持使用加权流量转移来进行 Amazon ECS 蓝/绿部署。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 (p. 388)	有关 CodeDeploy 的安全、身份验证和访问控制信息已被整理至新增安全性章节中。关更多信息，请参阅安全性。	2019 年 11 月 26 日
CodeDeploy 支持通知规则 (p. 388)	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 (p. 388)	CodeDeploy 现已在亚太地区（香港）区域 (ap-east-1) 区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。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 (p. 388)	Amazon CodeDeploy现在支持 Amazon ECS 服务中容器化应用程序的蓝/绿部署。使用新的 Amazon ECS 计算平台的 CodeDeploy 应用程序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。为反映这一变动，新增和更新了若干个主题，其中包括概述Amazon CodeDeploy计算平台、Amazon ECS 计算平台上的部署、用于亚马逊云服务器部署的应用程序规格文件结构, 和为 Amazon 云服务部署创建应用程序（控制台）。	2018 年 11 月 27 日
更新了 CodeDeploy (p. 388)	Amazon CodeDeploy 代理已更新到版本 1.0.1.1597。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	2018 年 11 月 15 日
更新的控制台 (p. 388)	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
CodeDeploy 代理最新的最低支持版本 (p. 388)	支持的最低支持版本Amazon CodeDeploy代理现在为 1.1.0。有关更多信息，请参阅 。CodeDeploy 代理的版本历史纪录。	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。

变更	描述	更改日期
主题更新	CodeDeploy 现已在欧洲（巴黎）区域（eu-west-3）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2017 年 12 月 19 日
更新的主题	CodeDeploy 现已在中国（宁夏）区域中提供。 要使用中国（北京）区域或中国（宁夏）区域中的服务，您必须拥有这些区域的账户和凭证。其他帐户和凭据Amazon地区不适用于北京和宁夏地区，反之亦然。 有关中国地区某些资源的信息（如 CodeDeploy 资源工具包存储桶名称和 CodeDeploy 代理安装过程）不包括在此版本的CodeDeploy。 有关更多信息： • CodeDeploy在入门Amazon在中国（北京）区域 • 中国地区 CodeDeploy 用户指南(构建可能会失|中文版本）	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	2017 年 11 月 28 日

变更	描述	更改日期
	函数版本。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。Amazon Lambda 部署受Amazon无服务器应用程序模型 (AmazonSAM ），以便您可以使用AmazonSAM 部署首选项，用于管理在流量期间转移的方式Amazon Lambda部署。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 (p. 2)、上的部署AmazonLambda 计算平台 (p. 11)、创建AmazonLambda 计算平台部署（控制台）(p. 257)、 创建一个应用程序Amazon Lambda函数部署（控制台）(p. 218)和AppSpecAmazonLambda 部署 (p. 242)。	
新主题	CodeDeploy 现已支持直接部署到本地计算机或安装了 CodeDeploy 代理的实例。您可以在本地测试部署，如果有错误，请使用 CodeDeploy 代理错误日志来进行调试。您还可以使用本地部署测试应用程序修订的完整性，测试 AppSpec 文件的内容以及更多内容。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 (p. 276)。	2017 年 11 月 16 日
更新的主题	对部署组中 Elastic Load Balancing CodeDeploy 负载均衡器的支持已扩展到蓝/绿部署和就地部署均包括。您现在可以为部署组选择 Application Load Balancer 负载均衡器、Classic Load Balancer 或 Network Load Balancer。负载均衡器在蓝/绿部署中为必备项，在就地部署中为可选项。为反映此支持，大量主题已更新，包括 Integrating CodeDeploy with Elastic Load Balancing (p. 42)、为就地部署创建应用程序（控制台）(p. 213)、部署先决条件 (p. 254)、Integrating CodeDeploy with Elastic Load Balancing (p. 42) 和 为就地部署创建应用程序（控制台）(p. 213)。	2017 年 9 月 12 日
更新的主题	对部署组中 Elastic Load Balancing CodeDeploy 负载均衡器的支持已扩展到蓝/绿部署和就地部署均包括。现在，您可以在部署组的 Application Load Balancer 和 Classic Load Balancer 之间进行选择。负载均衡器在蓝/绿部署中为必备项，在就地部署中为可选项。Integrating CodeDeploy with Elastic Load Balancing (p. 42)、使用 CodeDeploy 创建应用程序 (p. 212) 和 使用 CodeDeploy 创建部署组 (p. 225) 等主题已更新，以反映这一新增支持。	2017 年 8 月 10 日
新增和更新的主题	CodeDeploy 现已支持使用多个标签组来标记部署组中所包含实例的组合与交叉。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 (p. 163)。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）(p. 213)、为蓝/绿部署创建应用程序（控制台）(p. 215)、为就地部署创建部署组（控制台）(p. 225)、为 EC2/本地蓝/绿部署（控制台）创建部署组 (p. 227)、Deployments (p. 11)，和 教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101) 中的 第 5 步：创建应用程序和部署组 (p. 105)。	2017 年 31 月 7 日
对主题进行了更新	用于在 Windows 服务器实例上安装代理的另外两个方法已添加到安装适用于 Windows 服务器的 CodeDeploy 代理 (p. 156)。除了 Windows PowerShell 命令之外，还提供了使用直接 HTTPS 链接和使用 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 12 月 7 日

变更	描述	更改日期
更新的主题	CodeDeploy 改进了管理与 GitHub 账户和存储库的连接的方法。您现在可以创建和存储最多 25 个与 GitHub 账户的连接以便将 CodeDeploy 应用程序与 GitHub 存储库关联。每个连接均可支持多个存储库。您可以创建与最多 25 个不同的 GitHub 账户的连接，或创建与单个账户的多个连接。在将应用程序连接到 GitHub 账户后，CodeDeploy 管理所需的访问权限，而无需您执行任何进一步的操作。已对指定有关存储在 GitHub 存储库中的修订的信息 (p. 260)、GitHub 将 CodeDeploy 集成 (p. 47)和教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101)进行更新以反映此支持。	2017 年 5 月 30 日
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来自最新的成功部署的应用程序修订的一部分，则默认情况下，将无法进行当前部署。CodeDeploy 现在提供了针对代理处理这些文件的方式的选项：使部署失败，保留内容或覆盖内容。使用 CodeDeploy 创建部署 (p. 254)更新了以反映这一支持，新的部分对现有内容进行回滚行为 (p. 272)将添加到使用 CodeDeploy 重新部署和回滚部署 (p. 271)。	2017 年 5 月 16 日
更新的主题	弹性负载均衡器中的 Classic Load Balancer 现已分配给使用 CodeDeploy 控制台或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 (p. 36)、Integrating CodeDeploy with Elastic Load Balancing (p. 42)、为就地部署创建应用程序（控制台）(p. 213)、为就地部署创建部署组（控制台）(p. 225)和AppSpec 的 “hooks” 部分 (p. 335)。已向故障排除指南添加一个新的部分：排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)。	2017 年 4 月 27 日
更新的主题	弹性负载均衡器中的 Classic Load Balancer 现已分配给使用 CodeDeploy 控制台或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 (p. 36)、Integrating CodeDeploy with Elastic Load Balancing (p. 42)、为就地部署创建应用程序（控制台）(p. 213)、为就地部署创建部署组（控制台）(p. 225)和AppSpec 的 “hooks” 部分 (p. 335)。已向故障排除指南添加一个新的部分：排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368)。	2017 年 5 月 1 日

变更	描述	更改日期
更新的主题	CodeDeploy 现已在中国（北京）区域中提供。 要使用中国（北京）区域或中国（宁夏）区域中的服务，您必须拥有这些区域的账户和凭证。其他帐户和凭据Amazon地区不适用于北京和宁夏地区，反之亦然。 有关中国地区某些资源的信息（如 CodeDeploy 资源工具包存储桶名称和 CodeDeploy 代理安装过程）不包括在此版本的CodeDeploy。 有关更多信息： • CodeDeploy在入门Amazon在中国（北京）区域 • 中国地区 CodeDeploy 用户指南(构建可能会失|中文版本）	2017 年 3 月 29 日
新增和更新的主题	推出了多个新主题，用于说明对蓝/绿部署的新增支持。蓝/绿部署是一种部署方法，在此方法中，部署组（原始环境）中的实例将由一组不同的实例（替换环境）替代。蓝/绿部署概述 (p. 6)提供了对 CodeDeploy 使用的蓝色/绿色方法的高级说明。其他新主题包括为蓝/绿部署创建应用程序（控制台） (p. 215)、为 EC2/本地蓝/绿部署（控制台）创建部署组 (p. 227)和在 Elastic Load Balancing 中为 CodeDeploy Amazon EC2 部署设置负载均衡器 (p. 230)。 还更新了许多主题，其中包括 使用 CodeDeploy 创建部署 (p. 254)、在 CodeDeploy 中使用部署配置 (p. 206)、使用 CodeDeploy 创建应用程序 (p. 212)、在 CodeDeploy 中使用部署组 (p. 224)、在 CodeDeploy 中使用部署 (p. 253) 和 AppSpec 的 “hooks” 部分 (p. 335)。	2017 年 1 月 25 日
新增和更新的主题	新增主题使用 register ter-on-premises-instance 命令 (IAM 会话 ARN) 注册本地实例 (p. 192)，介绍了如何使用通过 Amazon Security Token Service 生成的定期刷新的临时凭证对本地实例进行身份验证和注册。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 (p. 179)的内容也已更新，以反映这一新的支持功能。	2016 年 12 月 28 日
更新的主题	CodeDeploy 现已在欧洲（伦敦）区域（eu-west-2）中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	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）中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2016 年 12 月 8 日
更新的主题	CodeDeploy 现已在美国东部（俄亥俄）区域（美国东部 2）中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2016 年 10 月 17 日
新主题	新增的 “身份验证和访问控制” 章节提供了有关使用Amazon Identity and Access Management(IAM)和 CodeDeploy，以帮助保护通过使用凭证帮助保护对您资源的访问。这些凭据提供访问Amazon资源，例如从 Amazon S3 存储桶检索应用程序修订和读取 Amazon EC2 实例上的标签。	2016 年 10 月 11 日

变更	描述	更改日期
对主题进行了更新	更新 Windows 服务器上的 CodeDeploy 代理 (p. 159)进行了更新，以反映 Windows 服务器的新 CodeDeploy 代理更新程序的可用性。在 Windows 服务器实例上安装后，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
更新的主题	CodeDeploy 现已与 Amazon CloudWatch 警报集成，使您能够在指定警报状态在多个持续时间段发生更改时停止部署，如警报阈值所指定。 CodeDeploy 现在还支持在满足特定条件（例如，部署失败或激活警报）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 CodeDeploy 创建应用程序 (p. 212)、使用 CodeDeploy 创建部署组 (p. 225)、使用 CodeDeploy 更改部署组设置 (p. 235)、Deployments (p. 11)、使用 CodeDeploy 重新部署和回滚部署 (p. 271)和CodeDeploy 产品和服务集成 (p. 36)，此外新增了主题使用 CodeDeploy 中的 CloudWatch 警报监控部署 (p. 283)。	2016 年 9 月 15 日
新增和更新的主题	CodeDeploy 现已提供与 Amazon CloudWatch Events 的集成。现在，当检测到部署组下的部署状态或实例状态发生更改时，您可以使用 CloudWatch 事件启动一个或多个操作。您可以合并调用Amazon Lambda函数，向 Kinesis 流或 Amazon SNS 主题发布内容，向 Amazon SQS 队列推送消息，或者反过来触发 CloudWatch 警报操作。有关更多信息，请参阅使用 Amazon CloudWatch Events 监控部署 (p. 284)。	2016 年 9 月 9 日
主题更新	主题Integrating CodeDeploy with Elastic Load Balancing (p. 42)和与其他Amazon服务 (p. 36)更新了以反映一个额外的负载均衡选项。CodeDeploy 现已支持弹性负载均衡器中提供的 Classic Load Balancer 和 Application Load Balancer。	2016 年 8 月 11 日
主题更新	CodeDeploy 现已在亚太地区（孟买）区域（ap-southea1）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2016 年 6 月 27 日
主题更新	CodeDeploy 现已在亚太地区（首尔）区域（ap-northeast-2）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。 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。为 CodeDeploy 教程添加了一个新的部分。为了提高可用性，已将几个较长的主题（包括 CodeDeploy AppSpec 文件参考 (p. 321) 和 排查 CodeDeploy (p. 362)）划分为多个较短的主题。代理的配置信息已移至新主题。CodeDeploy 代理配置参考 (p. 350)。	2016 年 6 月 15 日

变更	描述	更改日期
新增和更新的主题	的错误代码Amazon CodeDeploy (p. 383)提供了有关当 CodeDeploy 部署失败时可能显示的一些错误消息的信息。 排查 CodeDeploy (p. 362)中的下列部分已更新，以能够更好地帮助解决部署问题： • Amazon EC2 Auto Scaling 组中的 EC2 实例无法启动，收到错误 “心跳超时” (p. 377) • 避免将多个部署组与一个 Amazon EC2 Auto Scaling 组关联 (p. 376)	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) 中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。 使用 CodeDeploy 代理 (p. 141)进行了更新，以反映新:max_revisions:配置选项，您可以使用此选项来指定希望 CodeDeploy 代理对一个部署组进行归档的应用程序修订数。	2016 年 3 月 10 日
新增和更新的主题	CodeDeploy 现在支持将触发器添加到部署组中，以接收与该部署组中的部署或实例相关的事件的通知。这些通知将发送到您在触发器操作中订阅了 Amazon 简单通知服务主题的接收人。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 (p. 287)。 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。 排查 CodeDeploy (p. 362)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。 (p. 372)部分已更新。 CodeDeploy 限制 (p. 359)对进行了更新，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Amazon SDK for Ruby。	2016 年 2 月 17 日
新增和更新的主题	CodeDeploy 现已在美国西部（加利福利亚北部）区域（us-west-1）提供。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 (p. 246)列出并描述 CodeDeploy 现已支持的存储库类型。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 (p. 149)更新了有关新`.version`文件中添加到实例中以报告当前 CodeDeploy 代理的版本；此外还报告有关所支持代理版本的信息。 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 的修订版 (p. 239)已按照分步说明的方式重新整理。	2016 年 1 月 20 日
新主题	将应用程序部署到不同Amazon账户 (p. 273)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

变更	描述	更改日期
主题更新	CodeDeploy 产品和服务集成 (p. 36)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）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2015 年 12 月 9 日
主题更新	使用 CodeDeploy 代理 (p. 141)进行了更新，以反映新`:proxy_uri:`选项中的部 CodeDeploy 代理配置文件中的选项。 CodeDeploy AppSpec 文件参考 (p. 321)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	第 2 步：为 CodeDeploy 创建服务角色 (p. 26)进行了更新，以反映为 CodeDeploy 创建服务角色的新过程并纳入其他改进。	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）中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。 排查 CodeDeploy (p. 362)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用于 CodeDeploy 参考的模板 (p. 353)发布以反映新Amazon CloudFormation对 CodeDeploy 操作的支持。 创建了Primary Components (p. 8)主题，并引入了目标修订的定义。	2015 年 10 月 1 日
主题更新	使用 CodeDeploy 创建部署组 (p. 225)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 (p. 203)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）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这一新的可用区域。	2015 年 8 月 19 日
主题更新	CodeDeploy 现已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 (p. 141) • 使用 CodeDeploy 的实例 (p. 162) • 教程：将 WordPress 部署到 Amazon EC2 实例（亚马逊 Linux 或红帽企业 Linux 和 Linux、macOS 或 Unix）(p. 51) • 教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82)	2015 年 6 月 23 日

变更	描述	更改日期
主题更新	CodeDeploy 现在提供了一组环境变量，在部署期间您的部署脚本可以使用这些变量。这些环境变量包括多种信息，如当前 CodeDeploy 应用程序、部署组和部署生命周期事件的名称，以及当前 CodeDeploy 部署标识符。有关更多信息，请参阅CodeDeploy AppSpec 文件参考 (p. 321)中AppSpec 的 “hooks” 部分 (p. 335)部分的结尾。	2015 年 5 月 29 日
主题更新	CodeDeploy 现在提供了一组Amazon托管策略，您可以使用这些策略而无需自行手动创建等效策略。包括： • 一个用于允许 IAM 用户仅将修订注册到 CodeDeploy 并随后通过 CodeDeploy 进行部署的策略。 • 一个用于向 IAM 用户提供对 CodeDeploy 资源的完整访问权限的策略。 • 一个用于向 IAM 用户提供对 CodeDeploy 资源的只读访问权限的策略。 • 一个要附加到服务角色的策略，以便 CodeDeploy 可以通过 Amazon EC2 标签、本地实例标签或 Amazon EC2 Auto Scaling 组名来识别 Amazon EC2 实例，并相应对其部署应用程序修订。 有关更多信息，请参阅“身份验证和访问控制”中的 客户托管的策略示例 (p. 309) 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）区域中提供。有多个主题进行了更新，包括介绍如何设置 CodeDeploy 代理的主题，来说明这些新的可用区域。	2015 年 5 月 7 日
新主题	CodeDeploy 现已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 (p. 179) • 教程：使用 CodeDeploy（Windows 服务器、Ubuntu 服务器或红帽企业 Linux）将应用程序部署到本地实例 (p. 82) • Working with On-Premises Instances (p. 179)	2015 年 4 月 2 日
新主题	增加了CodeDeploy 资源 (p. 386)。	2015 年 4 月 2 日
主题更新	更新了排查 CodeDeploy (p. 362)： • 新的长时间运行的进程可能会导致部署失败 (p. 367)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。 • 这些区域有：Amazon EC2 Auto Scaling 常规故障排除 (p. 374)部分进行了更新，以说明 CodeDeploy 将其针对 CodeDeploy 代理的 Amazon EC2 Auto Scaling 超时逻辑从 5 分钟增加到 1 小时。 • 一个新不匹配的 Amazon EC2 Auto Scaling 生命周期挂钩可能导致至 Amazon EC2 Auto Scaling 组的自动部署停止或失败。 (p. 378)部分描述了一些步骤，您可以使用这些步骤来确定和解决自动部署到 Amazon EC2 Auto Scaling 组时的故障。	2015 年 4 月 2 日

变更	描述	更改日期
主题更新	下列主题进行了更新，以提供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 Amazon EC2 实例以使用 CodeDeploy (p. 177) • 第 4 步：为您的 Amazon EC2 实例创建 IAM 实例配置文件 (p. 31) • 第 2 步：为 CodeDeploy 创建服务角色 (p. 26) 排查 CodeDeploy (p. 362)中增加了两个部分： • 一般问题排查核对清单 (p. 362) • 默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell (p. 370) CodeDeploy AppSpec 文件参考 (p. 321)中的AppSpec 的 “hooks” 部分 (p. 335)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
主题更新	排查 CodeDeploy (p. 362)：Amazon EC2 Auto Scaling 组中的 EC2 实例无法启动，收到错误 “心跳超时” (p. 377)中增加了新的部分。 CodeDeploy 产品和服务集成 (p. 36)中增加了 CloudBees 部分。	2015 年 1 月 28 日
主题更新	排查 CodeDeploy (p. 362)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 (p. 364) • 使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 (p. 364) • 排查失败的 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 部署 (p. 368) • 排查失败的 DownloadBundle 部署生命周期事件的问题，错误为“UnknownError: not opened for reading” (p. 369) • Amazon EC2 Auto Scaling 常规故障排除 (p. 374)	2015 年 1 月 20 日
新主题	CodeDeploy 产品和服务集成 (p. 36)部分进行了更新，现在包括下列主题：: • 将 CodeDeploy 与 Amazon EC2 Auto Scaling 集成 (p. 39) • 教程：使用 CodeDeploy 将应用程序部署到 Amazon EC2 Auto Scaling 组 (p. 88) • Monitoring Deployments (p. 285) • Integrating CodeDeploy with Elastic Load Balancing (p. 42) • GitHub 将 CodeDeploy 集成 (p. 47) • 教程：使用 CodeDeploy 从 GitHub 部署应用程序 (p. 101)	2015 年 1 月 9 日

变更	描述	更改日期
主题更新	• GitHub 将 CodeDeploy 集成 (p. 47)中增加了使用 CodePipeline 自动 CodeDeploy Pipeline (p. 49)部分。现在，只要某个 GitHub 存储库中的源代码发生了更改，您就可以自动触发从该存储库执行部署的操作。 • 排查 CodeDeploy (p. 362)中增加了解决 Amazon EC2 Auto Scaling 问题 (p. 374)部分。这一新部分描述了如何排除在部署到 Amazon EC2 Auto Scaling 组时遇到的常见问题。 • CodeDeploy AppSpec 文件参考 (p. 321)的AppSpec “文件” 部分（仅限 EC2/本部署） (p. 324)部分中增加了新的子部分“文件示例”。这个新的小节包括若干个例子，说明如何使用`files`部分，指示 CodeDeploAppSpec 在部署期间将特定文件或文件夹复制到 Amazon EC2 实例上的特定位置。	2015 年 1 月 8 日
新主题	增加了Monitoring Deployments (p. 285)。CodeDeploy 与 Amazon CloudTrail，该服务可用于捕获由 CodeDeploy 或代表 CodeDeploy 发出的 API 调用。Amazon帐户，并将日志文件传送到您指定的 Amazon S3 存储桶。	2014 年 12 月 17 日
第一个公开发布版	这是第一个公开发布版CodeDeploy。	2014 年 11 月 12 日

Amazon词汇表

有关最新Amazon术语，请参阅Amazon一般参考中的Amazon术语表。

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。