
AWS CodeDeploy

User Guide

API 版本 2014-10-06



AWS CodeDeploy: User Guide

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

什么是 AWS CodeDeploy ?	1
AWS CodeDeploy 视频简介	1
AWS CodeDeploy 的优势	1
AWS CodeDeploy 计算平台概览	2
AWS CodeDeploy 部署类型概述	2
就地部署概述	3
蓝/绿部署概述	4
我们希望听到您的意见和建议	6
主要组件	6
部署	8
AWS Lambda 计算平台 上的部署	8
EC2/本地 计算平台 上的部署	11
应用程序规范文件	14
AWS Lambda 计算平台上的 AppSpec 文件	15
EC2/本地计算平台上的 AppSpec 文件	15
AWS CodeDeploy 代理如何使用 AppSpec 文件	15
入门	16
步骤 1：预置 IAM 用户	16
步骤 2：安装或升级 AWS CLI，然后对其进行配置	17
步骤 3：创建服务角色	18
创建服务角色（控制台）	18
创建服务角色（CLI）	19
获取服务角色 ARN（控制台）	21
获取服务角色 ARN（CLI）	21
步骤 4：创建 IAM 实例配置文件	22
为 Amazon EC2 实例创建 IAM 实例配置文件（CLI）	22
为 Amazon EC2 实例创建 IAM 实例配置文件（控制台）	24
步骤 5：尝试使用示例部署向导	25
先决条件	26
在 AWS CodeDeploy 中尝试使用示例蓝/绿部署	27
在 AWS CodeDeploy 中尝试示例就地部署	31
产品和服务集成	35
与其他 AWS 服务集成	35
Auto Scaling	38
Elastic Load Balancing	40
与合作伙伴产品和服务集成	42
GitHub	45
来自社区的集成示例	47
博客帖子	47
视频	48
教程	49
教程：将 WordPress 部署到非 Windows 实例	49
步骤 1：启动 Amazon EC2 实例	50
步骤 2：配置源内容	51
步骤 3：将您的应用程序上传到 Amazon S3	55
步骤 4：部署应用程序	58
步骤 5：更新和重新部署您的应用程序	62
第 6 步：清除	64
教程：将 HelloWorld 应用程序部署到 Windows Server 实例	66
步骤 1：启动 Amazon EC2 实例	67
步骤 2：配置源内容	67
步骤 3：将您的应用程序上传到 Amazon S3	70
步骤 4：部署应用程序	72
步骤 5：更新和重新部署您的应用程序	76

第 6 步：清除	78
教程：将应用程序部署到本地实例	79
先决条件	80
步骤 1：配置本地实例	80
步骤 2：创建示例应用程序修订	80
步骤 3：打包并将您的应用程序修订上传到 Amazon S3	84
步骤 4：部署应用程序修订	84
步骤 5：验证您的部署	84
步骤 6：清除资源	84
教程：部署到 Auto Scaling 组	86
先决条件	86
步骤 1：创建和配置 Auto Scaling 组	86
步骤 2：将应用程序部署到 Auto Scaling 组	91
步骤 3：检查结果	96
步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数	97
步骤 5：再次检查结果	98
步骤 6：清除	100
教程：从 GitHub 部署应用程序	101
先决条件	101
步骤 1：设置 GitHub 账户	101
步骤 2：创建 GitHub 存储库	102
步骤 3：将示例应用程序上传到 GitHub 存储库	103
步骤 4：预置实例	106
步骤 5：创建应用程序和部署组	106
步骤 6：将应用程序部署到实例	107
步骤 7：监控和验证部署	110
步骤 8：清除	111
使用 AWS CodeDeploy 代理	113
AWS CodeDeploy 代理支持的操作系统	113
支持的 Amazon EC2 AMI 操作系统	113
支持的本地操作系统	113
AWS CodeDeploy 代理的通信协议和端口	114
面向 AWS CodeDeploy 代理的、适用于 Ruby 的 AWS 开发工具包 (aws-sdk-core) 支持	114
AWS CodeDeploy 代理的版本历史纪录	114
应用程序修订和日志文件清理	116
AWS CodeDeploy 代理安装的文件	116
管理 AWS CodeDeploy 代理操作	118
验证 AWS CodeDeploy 代理是否正在运行	119
确定 AWS CodeDeploy 代理的版本	120
安装或重新安装 AWS CodeDeploy 代理	121
更新 AWS CodeDeploy 代理	127
卸载 AWS CodeDeploy 代理	131
使用实例	133
将 Amazon EC2 实例与本地实例进行比较	133
AWS CodeDeploy 实例任务	134
标记实例，便于 AWS CodeDeploy 部署	134
示例 1：唯一标签组，唯一标签	135
示例 2：唯一标签组，多个标签	136
示例 3：多个标签组，单个标签	137
示例 4：多个标签组，多个标签	139
使用 Amazon EC2 实例	141
创建 Amazon EC2 实例 (AWS CLI 或 Amazon EC2 控制台)	141
创建 Amazon EC2 实例 (AWS CloudFormation 模板)	147
配置 Amazon EC2 实例	153
使用本地实例	156
配置本地实例的先决条件	156
注册本地实例	157

管理本地实例操作	175
查看实例详细信息	179
查看实例详细信息 (控制台)	179
查看实例详细信息 (CLI)	180
实例运行状况	180
运行状况	181
最小运行正常的实例数和部署数	182
使用部署配置	184
EC2/本地计算平台上的部署配置	184
预定义的部署配置	184
AWS Lambda 计算平台上的部署配置	185
预定义的部署配置	186
.....	186
创建部署配置	186
查看部署配置详细信息	187
查看部署配置详细信息 (控制台)	187
查看部署配置 (CLI)	187
删除部署配置	188
使用应用程序	189
创建应用程序	189
为就地部署创建应用程序 (控制台)	190
为蓝/绿部署创建应用程序 (控制台)	192
为 AWS Lambda 函数部署创建应用程序 (控制台)	194
创建应用程序 (CLI)	195
查看应用程序详细信息	195
查看应用程序详细信息 (控制台)	195
查看应用程序详细信息 (CLI)	196
重命名应用程序	196
删除应用程序	196
删除应用程序 (控制台)	196
删除应用程序 (AWS CLI)	197
使用部署组	198
AWS Lambda 计算平台部署中的部署组	198
EC2/本地 计算平台 部署中的部署组	198
.....	198
创建部署组	198
为就地部署创建部署组 (控制台)	199
为蓝/绿部署创建部署组 (控制台)	200
在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer	202
创建部署组 (CLI)	203
查看部署组详细信息	203
查看部署组详细信息 (控制台)	203
查看部署组详细信息 (CLI)	204
更改部署组设置	204
更改部署组设置 (控制台)	204
更改部署组设置 (CLI)	205
为部署组配置高级选项	206
删除部署组	207
删除部署组 (控制台)	207
删除部署组 (CLI)	207
使用应用程序修订	208
计划修订	208
添加 AppSpec File	209
为 AWS Lambda 部署添加 AppSpec 文件	209
为 EC2/本地 部署添加 AppSpec 文件	210
选择存储库类型	213
推送修订	214

使用 AWS CLI 推送修订	215
查看应用程序修订详细信息	216
查看应用程序修订详细信息 (控制台)	216
查看应用程序修订详细信息 (CLI)	217
注册应用程序修订	217
使用 AWS CodeDeploy 在 Amazon S3 中注册修订 (CLI)	218
使用 AWS CodeDeploy 在 GitHub 中注册修订 (CLI)	218
使用部署	220
创建部署	220
部署先决条件	221
创建 AWS Lambda 计算平台 部署 (控制台)	223
创建 EC2/本地 计算平台 部署 (控制台)	223
创建 AWS Lambda 计算平台 部署 (CLI)	226
创建 EC2/本地 计算平台 部署 (CLI)	227
查看部署详细信息	229
查看部署详细信息 (控制台)	229
查看部署详细信息 (CLI)	229
查看部署日志数据	230
在 Amazon CloudWatch 控制台中查看日志文件数据	230
查看实例上的日志文件	230
停止部署	232
停止部署 (控制台)	232
停止部署 (CLI)	233
重新部署和回滚部署	233
自动回滚	233
手动回滚	233
回滚和重新部署工作流程	233
回滚行为与现有内容	234
在其他 AWS 账户中部署应用程序	235
步骤 1：在任一账户中创建 S3 存储桶	236
步骤 2：将 Amazon S3 存储桶权限授予生产账户的 IAM 实例配置文件	236
步骤 3：在生产账户中创建资源和跨账户角色	237
步骤 4：将应用程序修订上传到 Amazon S3 存储桶	237
步骤 5：代入跨账户角色和部署应用程序	238
验证本地机器上的部署程序包	238
先决条件	238
创建本地部署	240
示例	241
监控部署	243
自动化工具	243
手动工具	244
使用 Amazon CloudWatch 工具监控部署	244
使用 CloudWatch 警报监控部署	245
使用 Amazon CloudWatch Events 监控部署	246
使用 AWS CloudTrail 监视部署	248
CloudTrail 中的 AWS CodeDeploy 信息	248
了解 AWS CodeDeploy 日志文件条目	249
使用 Amazon SNS 事件通知监控部署	250
向服务角色授予 Amazon SNS 权限	251
为 AWS CodeDeploy 事件创建触发器	251
在部署组中编辑触发器	255
从部署组中删除触发器	257
触发器的 JSON 数据格式	258
身份验证和访问控制	260
身份验证	260
访问控制	261
访问管理概述	261

资源和操作	261
了解资源所有权	262
管理对资源的访问	263
指定策略元素：操作、效果和委托人	264
在策略中指定条件	265
使用基于身份的策略 (IAM 策略)	265
使用 AWS CodeDeploy 控制台所需的权限	266
适用于 AWS CodeDeploy 的 AWS 托管 (预定义) 策略	266
客户托管策略示例	267
AWS CodeDeploy 权限参考	269
AppSpec 文件参考	275
AWS Lambda 计算平台上的 AppSpec 文件	275
EC2/本地计算平台上的 AppSpec 文件	275
AppSpec File 结构	276
用于 AWS Lambda 部署的 AppSpec 文件结构	276
用于 EC2/本地 部署的 AppSpec 文件结构	276
AppSpec 的“files”部分 (仅限 EC2/本地 部署)	277
AppSpec 的“resources”部分 (仅限 AWS Lambda 部署)	280
AppSpec 的“permissions”部分 (仅限 EC2/本地 部署)	281
AppSpec 的“hooks”部分	284
AppSpec File 示例	293
用于 AWS Lambda 部署的 AppSpec File 示例	293
用于 EC2/本地 部署的 AppSpec File 示例	294
AppSpec File 间距	295
验证您的 AppSpec File 和文件位置	296
代理配置参考	297
相关主题	299
AWS CloudFormation 模板参考	300
资源工具包参考	302
各区域的资源工具包存储桶名称	302
资源工具包内容	302
显示资源工具包文件列表	303
下载资源工具包文件	304
限制	307
应用程序	307
应用程序修订	307
部署	307
部署配置	308
部署组	309
实例	309
故障排除	310
一般问题排查	310
一般问题排查核对清单	310
AWS CodeDeploy 部署资源仅在特定区域中受支持	311
所需的 IAM 角色不可用	312
避免对同一 Amazon EC2 实例进行并行部署	312
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	312
使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败	313
排查 EC2/本地 部署问题	313
部署失败，显示消息“PKCS7 签名的消息验证失败”	313
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”	313
排查 AllowTraffic 生命周期事件失败，但部署日志中未报错的问题	315
对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除	315
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”	316

默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell	316
长时间运行的进程可能会导致部署失败	317
排查 AWS Lambda 部署问题	318
在手动终止未配置回滚的 Lambda 部署后，AWS Lambda 部署失败	318
解决部署组问题	318
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	318
解决实例问题	319
必须正确设置标签	319
必须安装并在实例上运行 AWS CodeDeploy 代理	319
如果实例在部署期间终止，在最多 1 小时内部署不会失败。	319
分析日志文件以调查针对实例的部署失败	319
如果 AWS CodeDeploy 日志文件被意外删除，请创建一个新的日志文件	320
排查“InvalidSignatureException – Signature expired: [time] is now earlier than [time]”部署错误	320
GitHub 令牌问题排查	320
无效 GitHub OAuth 令牌	320
超出了 GitHub OAuth 令牌数量上限	320
解决 Auto Scaling 问题	321
一般 Auto Scaling 问题排查	321
在部署修订之前，Auto Scaling 组中的实例不断预配并终止	321
终止或重启 Auto Scaling 实例可能会导致部署失败	322
避免将多个部署组与一个 Auto Scaling 组关联	322
Auto Scaling 组中的 Amazon EC2 实例无法启动，收到错误“Heartbeat Timeout”	323
不匹配的 Auto Scaling 生命周期钩子可能导致针对 Auto Scaling 组的自动部署停止或失败	323
错误代码	324
相关主题	326
Resources	327
参考指南和支持资源	327
示例	327
博客	327
AWS 软件开发工具包和工具	327
文档历史记录	329
早期更新	329
AWS 词汇表	340

什么是 AWS CodeDeploy ?

AWS CodeDeploy 是一项部署服务，可以向 Amazon EC2 实例、本地实例或无服务器 Lambda 函数自动执行应用程序部署。

您几乎可以部署无限种类的应用程序内容，例如代码、无服务器 AWS Lambda 函数、Web 文件和配置文件、可执行文件、程序包、脚本、多媒体文件，等等。AWS 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。AWS 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 AWS CodeDeploy。

AWS CodeDeploy 能让您更轻松地执行以下操作：

- 快速发布新功能。
- 更新 AWS 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。

AWS 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](#)。

主题

- [AWS CodeDeploy 视频简介 \(p. 1\)](#)
- [AWS CodeDeploy 的优势 \(p. 1\)](#)
- [AWS CodeDeploy 计算平台概览 \(p. 2\)](#)
- [AWS CodeDeploy 部署类型概述 \(p. 2\)](#)
- [我们希望听到您的意见和建议 \(p. 6\)](#)
- [AWS CodeDeploy 主要组件 \(p. 6\)](#)
- [AWS CodeDeploy 部署 \(p. 8\)](#)
- [AWS CodeDeploy 应用程序规范文件 \(p. 14\)](#)

AWS CodeDeploy 视频简介

此简短视频 (2:10) 介绍 AWS CodeDeploy 如何自动执行到 Amazon EC2 实例的代码部署，从而便于您快速发布新功能，消除部署期间的停机时间，并避免执行易于出错的手动操作。

[AWS CodeDeploy 部署的视频演练。](#)

AWS CodeDeploy 的优势

AWS CodeDeploy 具备下列优势：

- 服务器和无服务器应用程序。AWS 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 AWS Lambda 函数版本的应用程序。
- 自动部署。AWS 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。AWS 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。AWS CodeDeploy 可以帮助最大程度地提高应用程序的可用性。在就地部署期间，AWS 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。

态的实例的数量。在蓝/绿部署中，最新应用程序修订将安装到替换实例上，在您选择时，流量会立即路由到这些实例或者在完成新环境测试之后路由。对于两种部署类型，AWS 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 AWS CodeDeploy 控制台或 AWS CLI 启动并跟踪部署状态。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。AWS CodeDeploy 与平台无关，适用于任何应用程序。您可以轻松重用设置代码。AWS CodeDeploy 还能与您的软件发布过程或持续交付工具链集成。

AWS CodeDeploy 计算平台概览

AWS CodeDeploy 可以将应用程序部署到两个 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 [AWS CodeDeploy 部署类型概述 \(p. 2\)](#)。

- AWS Lambda：用于部署包含更新后的 Lambda 函数版本的应用程序。AWS 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 AWS Lambda 执行。有关更多信息，请参阅[无服务器计算和应用程序](#)。有关 AWS Lambda 和 Lambda 函数的更多信息，请参阅 [AWS Lambda](#)。

使用 AWS Lambda 计算平台创建的应用程序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定向到更新后的 Lambda 函数版本的方式。

下表描述 AWS CodeDeploy 组件如何与每个 计算平台 一起使用。

AWS CodeDeploy 组件	EC2/本地	AWS Lambda
部署组	部署一组将要在其上部署新修订的实例。	将 Lambda 函数版本部署到高可用性计算基础设施。
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。	部署一个包括 AppSpec 文件的新修订。AppSpec 指定要部署哪个 Lambda 函数版本。
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。
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。	AppSpec 文件指定要部署和更新哪些 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。	修订的集合。Lambda 应用程序使用 AWS Lambda 计算平台。

AWS CodeDeploy 部署类型概述

AWS 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，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2/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. 3)。

Note

AWS Lambda 计算平台部署无法使用就地部署类型。

- 蓝/绿部署：部署的行为取决于使用的计算平台：
 - 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：
 - 系统将为替代环境配置实例。
 - 替代实例上将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来说，等待时间可选。
 - 替代环境中的实例在 Elastic Load Balancing 负载均衡器中进行注册，使得流量重新路由至这些实例。系统将撤销原始环境中的实例注册，进而终止或因其他使用情形而保持运行。

Note

在使用 EC2/本地 计算平台时，蓝/绿部署只能与 Amazon EC2 实例配合使用。

- AWS Lambda 计算平台上的蓝/绿部署：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方法。所有 AWS Lambda 计算平台部署都是蓝/绿部署。因此，您无需指定部署类型。

有关蓝/绿部署的更多信息，请参阅[蓝/绿部署概述](#) (p. 4)。

Note

使用 AWS 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 AWS 账户。有关信息，请参阅 [使用 AWS CodeDeploy 代理验证本地机器上的部署程序包](#) (p. 238)。

主题

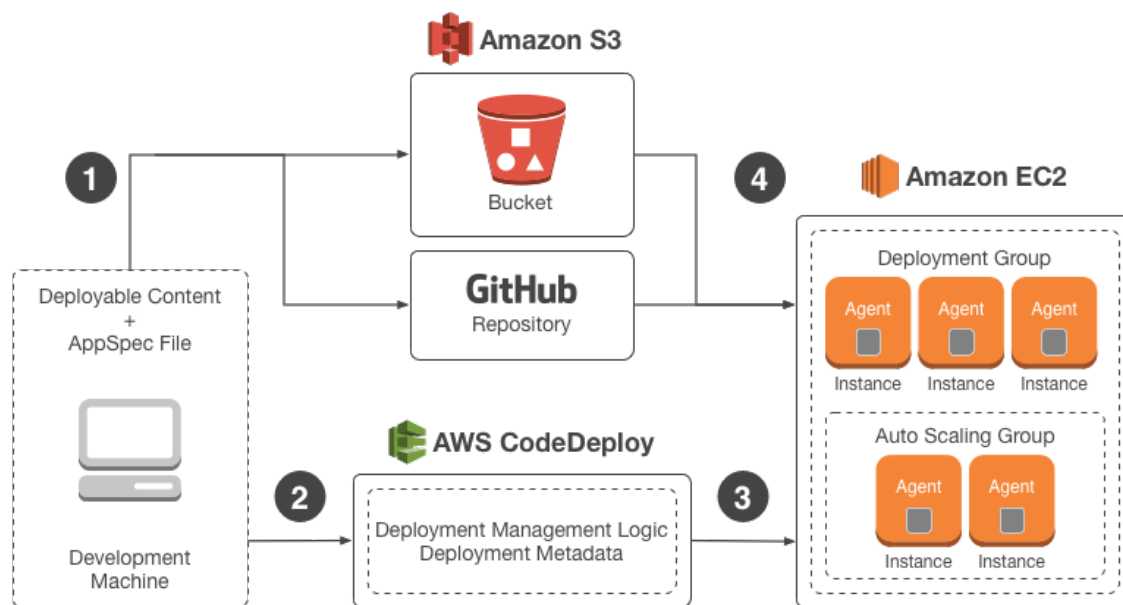
- [就地部署概述](#) (p. 3)
- [蓝/绿部署概述](#) (p. 4)

就地部署概述

下图显示了典型的 AWS CodeDeploy 就地部署流。

Note

AWS Lambda 计算平台部署无法使用就地部署类型。



下面将介绍操作方式：

1. 首先，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application specification file (AppSpec file)。AppSpec file对 AWS CodeDeploy 是唯一的。它定义您希望 AWS CodeDeploy 执行的部署操作。将可部署的内容和 AppSpec file捆绑成一个存档文件，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 (或简称修订)。
2. 接下来，向 AWS 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。AWS CodeDeploy 将一组 Amazon EC2 实例称为一个部署组。部署组中包含单独标记的 Amazon EC2 实例和/或 Auto Scaling 组中的 Amazon EC2 实例。

每次您成功上传要部署到部署组的新应用程序修订时，该捆绑就将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。这也是将为自动部署提取的修订。

3. 接下来，每个实例上的 AWS CodeDeploy 代理将轮询 AWS 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 AWS CodeDeploy 代理将从指定的 Amazon S3 存储桶或 GitHub 存储库中提取目标修订，并按照 AppSpec file中的说明向实例中部署内容。

AWS 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

在蓝/绿部署中将流量从一组实例 (原始环境) 重新路由到另一组实例 (替换环境)，相比就地部署提供了多种优势：

- 可以提前在新实例上安装和测试应用程序，并通过直接将流量切换到新服务器来将应用程序部署到生产中。
- 可以更快、更可靠地切换回最近的应用程序版本，因为只要原始实例没有终止，就可以将流量路由回原始实例。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 AWS Lambda 计算平台，则需要控制流量如何从原始 AWS Lambda 函数版本转移到新的 AWS Lambda 函数版本。

如何配置蓝/绿部署取决于部署使用的计算平台。

AWS Lambda 计算平台上的蓝/绿部署

如果您使用 AWS Lambda 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 AWS Lambda 函数版本转移到新的 AWS Lambda 函数版本：

- **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。
- **线性的**：流量使用相等的增量转移，在每次增量之间的分钟数相同。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。
- **All-at-once**：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 AWS Lambda 部署配置的更多信息，请参阅[AWS Lambda 计算平台的预定义部署配置](#) (p. 186)。

EC2/本地计算平台上的蓝/绿部署

Note

必须对 EC2/本地 计算平台上的蓝/绿部署使用 Amazon EC2 实例。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 标签或 Auto Scaling 组的 Amazon EC2 实例。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 AWS 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 Auto Scaling 组：在蓝/绿部署中，AWS CodeDeploy 将在部署期间自动为替换环境创建实例。使用此选项，AWS CodeDeploy 将使用您指定的 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 实例标签和/或 Auto Scaling 组名称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. 您已有将充当原始环境的一些实例或一个 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 AWS 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 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 Elastic Load Balancing 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。

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 [AWS CodeDeploy 论坛](#)。

主题

- [Primary Components \(p. 6\)](#)
- [Deployments \(p. 8\)](#)
- [Application Specification Files \(p. 14\)](#)

AWS CodeDeploy 主要组件

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

Application: A name that uniquely identifies the application you want to deploy. AWS CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.

计算平台：AWS CodeDeploy 部署应用程序的平台。

- **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 [AWS CodeDeploy 部署类型概述 \(p. 2\)](#)。

- **AWS Lambda：**用于部署包含更新后的 Lambda 函数版本的应用程序。AWS 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 AWS Lambda 执行。有关更多信息，请参阅[无服务器计算和应用程序](#)。有关 AWS Lambda 和 Lambda 函数的更多信息，请参阅 [AWS Lambda](#)。

使用 AWS Lambda 计算平台创建的应用程序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定向到更新后的 Lambda 函数版本的方式。

部署配置：AWS CodeDeploy 在部署期间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的运行正常的实例。如果您的部署使用 AWS Lambda 计算平台，您可以指定流量路由到更新后的 Lambda 函数版本的方式。

有关对使用 EC2/本地 计算平台的部署指定最少数量运行正常的主机的更多信息，请参阅[最小运行正常的实例数和部署数 \(p. 182\)](#)。

在使用 AWS Lambda 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 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。
- **线性的**：流量使用相等的增量转移，在每次增量之间的分钟数相同。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。
- **All-at-once**：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。

部署组：一组单独的实例。部署组中包含单独标记的实例和/或 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 \(p. 156\)](#)。有关 Auto Scaling 的信息，请参阅[将 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。

Deployment type: The method used to make the latest application revision available on instances in a deployment group.

- **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，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2/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 \(p. 3\)](#)。
- **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**：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：
 - 系统将为替代环境配置实例。
 - 替代实例上将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来说，等待时间可选。
 - 替代环境中的实例在 Elastic Load Balancing 负载均衡器中进行注册，使得流量重新路由至这些实例。系统将撤销原始环境中的实例注册，进而终止或因其他使用情形而保持运行。

Note

在使用 EC2/本地 计算平台时，蓝/绿部署只能与 Amazon EC2 实例配合使用。

- **AWS Lambda 计算平台上的蓝/绿部署**：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方法。所有 AWS Lambda 计算平台部署都是蓝/绿部署。因此，您无需指定部署类型。

有关蓝/绿部署的更多信息，请参阅[蓝/绿部署概述 \(p. 4\)](#)。

IAM instance profile: An IAM role that you attach to your Amazon EC2 instances. This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications that will be deployed by AWS CodeDeploy are stored. For more information, see [步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#).

修订：AWS Lambda 部署修订是一种 YAML 格式或 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 application specification file (AppSpec file)。AWS Lambda 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 对象键与 ETag 和/或版本唯一标识。对于 GitHub，修订由其提交 ID 唯一标识。

Service role: An IAM role that grants permissions to an AWS service so it can access AWS resources. The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources. For AWS CodeDeploy, a service role is used for the following:

- To read either the tags applied to the instances or the Amazon EC2 Auto Scaling group names associated with the instances. This enables AWS CodeDeploy to identify instances to which it can deploy applications.

- To perform operations on instances, Auto Scaling groups, and Elastic Load Balancing load balancers.
- To publish information to Amazon SNS topics so that notifications can be sent when specified deployment or instance events occur.
- To retrieve information about CloudWatch alarms in order to set up alarm monitoring for deployments.

For more information, see [步骤 3 : 为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#).

Target revision: The most recent version of the application revision that you have uploaded to your repository and want to deploy to the instances in a deployment group. In other words, the application revision currently targeted for deployment is the target revision. This is also the revision that will be pulled for automatic deployments.

有关 AWS CodeDeploy 工作流程中的其他主要组件的信息，请参阅以下主题：

- [选择 AWS CodeDeploy 存储库类型 \(p. 213\)](#)
- [Deployments \(p. 8\)](#)
- [Application Specification Files \(p. 14\)](#)
- [Instance Health \(p. 180\)](#)
- [使用 AWS CodeDeploy 代理 \(p. 113\)](#)
- [Working with On-Premises Instances \(p. 156\)](#)

AWS CodeDeploy 部署

本主题提供 AWS CodeDeploy 中部署的组件和工作流的相关信息。部署过程因您的用于部署的计算平台 (EC2/本地或 Lambda) 而异。

主题

- [AWS Lambda 计算平台上的部署 \(p. 8\)](#)
- [EC2/本地 计算平台上的部署 \(p. 11\)](#)

AWS Lambda 计算平台上的部署

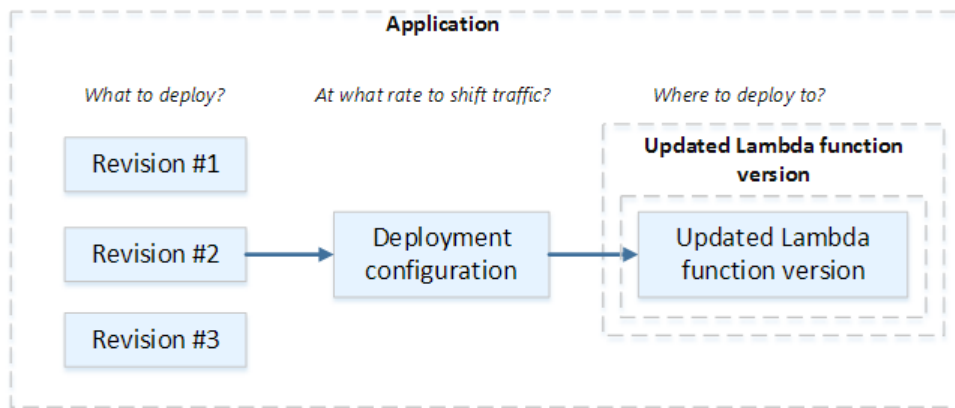
本主题提供了有关使用 AWS Lambda 计算平台的 AWS CodeDeploy 部署的组件和工作流的信息。

主题

- [AWS Lambda 计算平台上的部署组件 \(p. 8\)](#)
- [AWS Lambda 计算平台上的部署工作流程 \(p. 9\)](#)
- [上传应用程序修订 \(p. 10\)](#)
- [创建应用程序和部署组 \(p. 10\)](#)
- [部署应用程序修订 \(p. 10\)](#)
- [更新应用程序 \(p. 10\)](#)
- [停止和失败的部署 \(p. 10\)](#)
- [重新部署和部署回滚 \(p. 10\)](#)

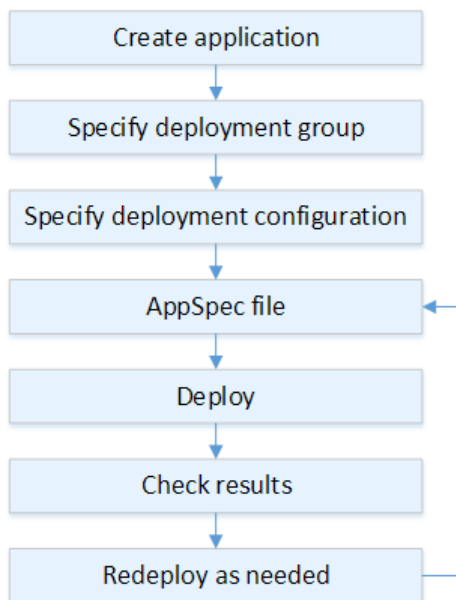
AWS Lambda 计算平台上的部署组件

下图显示了 AWS Lambda 计算平台上的 AWS CodeDeploy 部署中的组件。



AWS Lambda 计算平台上的部署工作流程

下图显示了部署新增和更新 AWS Lambda 函数的主要步骤。



这些步骤包括：

1. 通过指定一个唯一表示要部署的应用程序修订的名称，创建应用程序。要部署 Lambda 函数，您需要在创建应用程序时指定 AWS Lambda 计算平台。AWS CodeDeploy 在部署期间使用此名称以确保它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 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)。
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 AWS Lambda 函数版本转移到新的 Lambda 函数版本。有关更多信息，请参阅 [View Deployment Configuration Details \(p. 187\)](#)。
4. 将 application specification file (AppSpec file) 上传到 Amazon S3。AppSpec file 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 file，则可以使用 JSON 或 YAML 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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。
5. 将应用程序修订部署到部署组。AWS 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 文件转移到您的 Lambda 函数修订。有关更多信息，请参阅 [使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

6. 检查部署结果。有关更多信息，请参阅 [在 AWS CodeDeploy 中监控部署 \(p. 243\)](#)。
7. 重新部署修订。如果需要在部署中修复 Lambda 函数中的错误或解决部署失败的问题，您可能希望这样做。如果您重新部署相同 Lambda 函数版本，则仅需将新部署执行到具有相同 AppSpec file 的部署组。如果您在重新部署之前重命名、添加或删除一个 Lambda 函数，则必须更新您的 AppSpec file。有关更多信息，请参阅 [使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

上传应用程序修订

将 AppSpec file 放入 Amazon S3 中或将其直接输入到控制台或 AWS CLI 中。有关更多信息，请参阅 [Application Specification Files \(p. 14\)](#)。

创建应用程序和部署组

AWS Lambda 计算平台 上的 AWS CodeDeploy 部署组标识一个或多个 AppSpec file。每个 AppSpec file 可以部署一个 Lambda 函数版本。部署组还定义一些用于未来部署的配置选项，例如警报和回滚配置。

部署应用程序修订

现在您已做好准备，可将 AppSpec file 中指定的函数修订部署到部署组。可以使用 AWS CodeDeploy 控制台或 [create-deployment](#) 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 AWS CodeDeploy 控制台或调用 [create-deployment](#) 命令推送修订。

停止和失败的部署

您可以使用 AWS CodeDeploy 控制台或 [stop-deployment](#) 命令停止部署。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 AWS CodeDeploy API Reference 中的 [错误信息](#) 和 [常见错误](#)。

与停止的部署一样，失败的部署可能导致某些部署生命周期事件已在运行。要查明部署失败的原因，可以使用 AWS 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 \(p. 116\)](#) 和 [查看 AWS CodeDeploy 部署日志数据 \(p. 230\)](#)。

重新部署和部署回滚

AWS CodeDeploy 实现回滚的方式是将以前部署的版本重新部署为新的部署。

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 AWS 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 AWS CodeDeploy 重新部署和回滚部署 \(p. 233\)](#)。

EC2/本地 计算平台 上的部署

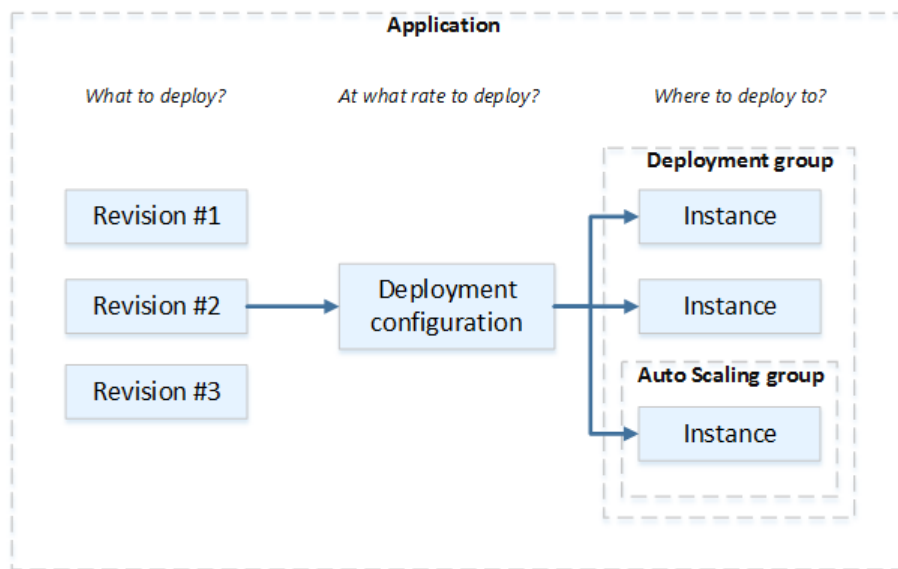
本主题提供了有关使用 EC2/本地 计算平台 的 AWS CodeDeploy 部署的组件和工作流程的信息。有关蓝/绿部署的信息，请参阅[蓝/绿部署概述 \(p. 4\)](#)。

主题

- [EC2/本地计算平台上的部署组件 \(p. 11\)](#)
- [EC2/本地计算平台上的部署工作流程 \(p. 11\)](#)
- [设置实例 \(p. 13\)](#)
- [上传应用程序修订 \(p. 13\)](#)
- [创建应用程序和部署组 \(p. 13\)](#)
- [部署应用程序修订 \(p. 14\)](#)
- [更新应用程序 \(p. 14\)](#)
- [停止和失败的部署 \(p. 14\)](#)
- [重新部署和部署回滚 \(p. 14\)](#)

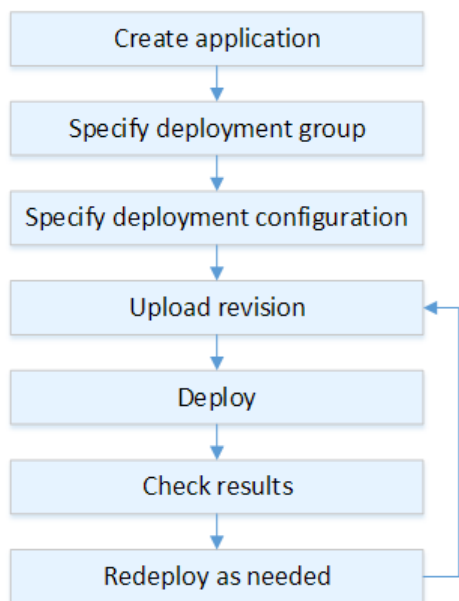
EC2/本地计算平台上的部署组件

下图显示了 EC2/本地计算平台上的 AWS 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. 通过指定唯一地表示要部署的应用程序修订的名称以及应用程序的计算平台，来创建应用程序。AWS CodeDeploy 在部署期间使用此名称以确保它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 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)。
2. 设置部署组时，要指定部署类型和要部署应用程序修订的实例。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。

可以指定应用于实例的标签和/或 Auto Scaling 组名称。

如果在部署组中指定一组标签，AWS 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，AWS 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 AWS CodeDeploy Deployments \(p. 134\)](#)。

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 Auto Scaling 组），并且已安装并运行 AWS CodeDeploy 代理。

我们提供一个 AWS CloudFormation 模板，您可使用该模板基于 Amazon Linux 或 Windows Server 快速设置 Amazon EC2 实例。我们还提供独立的 AWS CodeDeploy 代理，以便您可以将其安装在 Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL) 或 Windows Server 实例上。有关更多信息，请参阅 [使用 AWS CodeDeploy 创建部署组 \(p. 198\)](#)。

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 \(p. 250\)](#)。
 - 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，以指明多少实例要同时部署应用程序修订并描述部署的成功和失败条件。有关更多信息，请参阅 [View Deployment Configuration Details \(p. 187\)](#)。
 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，您还必须包括 application specification file (AppSpec file)。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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

5. 将应用程序修订部署到部署组。部署组中每个实例的 AWS 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，AWS CodeDeploy 代理将取消捆绑修订，使用 AppSpec file 将相应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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。
6. 检查部署结果。有关更多信息，请参阅 [在 AWS CodeDeploy 中监控部署 \(p. 243\)](#)。
7. 重新部署修订。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，可将修订后的源内容、所有部署脚本和 AppSpec file 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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

设置实例

您需要先设置实例，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。

要手动预配实例，请执行以下操作：

1. 在实例上安装 AWS CodeDeploy 代理。AWS CodeDeploy 代理可安装在 Amazon Linux、Ubuntu Server、RHEL 和 Windows Server 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。AWS CodeDeploy 依赖标签来标识实例并将其分组到 AWS 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启动的 Amazon EC2 实例，以便 AWS 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 AWS CodeDeploy 可以展开您的 AWS 账户中的标签。

对于初始部署，AWS CloudFormation 模板将为您完成所有这些操作。它基于已安装 AWS CodeDeploy 代理的 Amazon Linux 或 Windows Server 创建并配置单个新 Amazon EC2 实例。有关更多信息，请参阅 [使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)。

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 AWS CodeDeploy 在部署过程中为您预置新实例。

上传应用程序修订

将 AppSpec file 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 \(p. 14\)](#)。

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

Note

The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

创建应用程序和部署组

AWS CodeDeploy 部署组基于实例的标签和/或 Auto Scaling 组名称标识实例集合。多个应用程序修订可部署到同一个实例。一个应用程序修订可部署到多个实例。

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 AWS 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 AWS CodeDeploy Deployments](#) (p. 134)。

部署应用程序修订

现在，您可以开始将您的应用程序修订从 Amazon S3 或 GitHub 部署到部署组。可以使用 AWS CodeDeploy 控制台或 [create-deployment](#) 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 AWS CodeDeploy 控制台或调用 [create-deployment](#) 命令推送修订。

停止和失败的部署

您可以使用 AWS CodeDeploy 控制台或 [stop-deployment](#) 命令停止部署。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 AWS CodeDeploy API Reference 中的 [错误信息](#) 和 [常见错误](#)。

与停止的部署一样，失败的部署可能导致一些部署生命周期事件已在部署组中的一个或多个实例上运行。要找出部署失败的原因，您可以使用 AWS CodeDeploy 控制台调用 [get-deployment-instance](#) 命令或分析失败部署的日志文件数据。有关更多信息，请参阅 [应用程序修订和日志文件清理](#) (p. 116) 和 [查看 AWS CodeDeploy 部署日志数据](#) (p. 230)。

重新部署和部署回滚

AWS CodeDeploy 实现回滚的方式是将以前部署的版本重新部署为新的部署。

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 AWS 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 AWS CodeDeploy 重新部署和回滚部署](#) (p. 233)。

AWS CodeDeploy 应用程序规范文件

对 AWS CodeDeploy 唯一的 application specification file (AppSpec file) 是 [YAML](#) 格式或 [JSON](#) 格式文件。AppSpec file 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 file 的信息，请参阅 [AWS CodeDeploy AppSpec File 参考](#) (p. 275)。

主题

- [AWS Lambda 计算平台上的 AppSpec 文件 \(p. 15\)](#)
- [EC2/本地计算平台上的 AppSpec 文件 \(p. 15\)](#)
- [AWS CodeDeploy 代理如何使用 AppSpec 文件 \(p. 15\)](#)

AWS Lambda 计算平台上的 AppSpec 文件

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

- 要部署的 AWS 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 AWS Lambda 部署的 AppSpec 的“hooks”部分 \(p. 285\)](#)。

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 file 文件始终是 YAML 格式的。AppSpec file 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。AWS 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 [用于 EC2/本地 部署的 AppSpec 的“hooks”部分 \(p. 287\)](#)。

AWS CodeDeploy 代理如何使用 AppSpec 文件

部署期间，AWS CodeDeploy 代理将在 AppSpec file 的 hooks 部分中查找当前事件的名称。如果找不到该事件，AWS CodeDeploy 代理将移到下一步。如果找到该事件，AWS CodeDeploy 代理将检索要执行的脚本列表。脚本按其出现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 AWS 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

AWS CodeDeploy 代理不用于 AWS Lambda 部署中。

在 Install 事件期间，AWS CodeDeploy 代理使用 AppSpec file 的 files 部分中定义的映射来确定要从修订复制到的实例的文件夹或文件。

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

有关 AWS CodeDeploy 代理日志文件的信息，请参阅 [使用 AWS CodeDeploy 代理 \(p. 113\)](#)。

AWS CodeDeploy 入门

在首次使用 AWS CodeDeploy 之前，必须完成设置步骤。

在开始时，您必须注册 AWS 账户。要注册，请转至 <https://aws.amazon.com/> 并选择 Create an AWS Account。

然后，您可以继续执行本部分中的其余设置步骤。

主题

- [步骤 1：预置 IAM 用户 \(p. 16\)](#)
- [步骤 2：安装或升级 AWS CLI，然后对其进行配置 \(p. 17\)](#)
- [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)
- [步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)
- [步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 \(p. 25\)](#)

步骤 1：预置 IAM 用户

按照以下说明准备 IAM 用户以使用 AWS CodeDeploy：

1. 创建 IAM 用户或使用与您的 AWS 账户关联的现有用户。有关更多信息，请参阅 IAM 用户指南 中的 [创建 IAM 用户](#)。
2. 通过复制以下策略并将其附加到 IAM 用户，向 IAM 用户授予对 AWS CodeDeploy (以及 AWS CodeDeploy 所依赖的 AWS 服务和操作) 的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "autoscaling:*",
        "codedeploy:*",
        "ec2:*",
        "lambda:*",
        "elasticloadbalancing:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam>DeleteInstanceProfile",
        "iam>DeleteRole",
        "iam>DeleteRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
```



```
        "s3:*"
      ],
      "Resource" : "*"
    }
  ]
}
```

上述策略向 IAM 用户授予部署到 AWS Lambda 计算平台和 EC2/本地 计算平台时所需的访问权。

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 AWS CodeDeploy 操作和资源，请参阅[AWS CodeDeploy 的身份验证和访问控制](#) (p. 260)。

您可使用本文档中提供的 AWS CloudFormation 模板启动与 AWS CodeDeploy 兼容的 Amazon EC2 实例。要使用 AWS CloudFormation 模板创建应用程序、部署组或部署配置，您必须向 IAM 用户授予对 AWS CloudFormation 以及 AWS CloudFormation 所依赖的 AWS 服务和操作的访问权，方式是将其他策略附加到 IAM 用户，如下所示：

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

有关这些语句中列出的其他 AWS 服务的信息，请参阅：

- [AWS IAM 策略概述](#)
- [控制用户对您的负载均衡器的访问权限](#)
- [控制对您 Auto Scaling 资源的访问权](#)
- [使用 AWS Identity and Access Management 控制 AWS CloudFormation 访问](#)

步骤 2：安装或升级 AWS CLI，然后对其进行配置

要从本地开发计算机上的 AWS CLI 调用 AWS CodeDeploy 命令，您必须安装 AWS CLI。AWS CLI 的 1.6.1 版最先提供了 AWS CodeDeploy 命令。而 1.7.19 版的 AWS CLI 中提供了用于处理本地实例的 AWS CodeDeploy 命令。

如果您安装的是旧版 AWS CLI，则必须升级它，以使 AWS CodeDeploy 命令可用。您可调用 `aws --version` 来查看版本。

安装或升级 AWS CLI：

1. 按照[安装 AWS Command Line Interface](#) 中的说明操作可安装或升级 AWS CLI。
2. 要配置 AWS CLI，请参阅[配置 AWS Command Line Interface](#) 和[管理 IAM 用户的访问密钥](#)。

Important

在配置 AWS CLI 时，系统将提示您指定 AWS 区域。指定 AWS General Reference 中的[区域和终端节点](#)中列出的某个支持区域。

3. 要确认安装或升级，请从 AWS CLI 调用以下命令：

API 版本 2014-10-06

```
aws deploy help
```

如果成功，此命令将显示可用 AWS CodeDeploy 命令的列表。

步骤 3：为 AWS CodeDeploy 创建服务角色

在 AWS 中，服务角色用于向 AWS 服务授予权限，以便能访问 AWS 资源。附加到服务角色的策略将确定服务可访问的 AWS 资源以及可使用这些资源执行的操作。

对于您为 AWS CodeDeploy 创建的服务角色，必须向其授予对您将应用程序部署到的实例的访问权限。利用这些权限，AWS CodeDeploy 能够读取已应用于实例的标签或与实例关联的 Auto Scaling 组名称。

您添加到服务角色的权限指定了 AWS CodeDeploy 在访问您的 Amazon EC2 实例和 Auto Scaling 组时可执行的操作。要添加这些权限，请将 AWS 提供的策略 `AWSCodeDeployRole` 附加到服务角色。

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、AWS CLI 或 IAM API 创建服务角色。

主题

- [创建服务角色 \(控制台\)](#) (p. 18)
- [创建服务角色 \(CLI\)](#) (p. 19)
- [获取服务角色 ARN \(控制台\)](#) (p. 21)
- [获取服务角色 ARN \(CLI\)](#) (p. 21)

创建服务角色 (控制台)

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 在 Create role (创建角色) 页面上，选择 AWS service (AWS 服务)，然后从 Choose the service that will use this role (选择将使用此角色的服务) 列表中，选择 CodeDeploy。
4. 从选择您的使用案例中，选择 CodeDeploy。
5. 选择 Next: Permissions。
6. 在 Attached permissions policy 页面上，如果有一个框紧挨着 **AWSCodeDeployRole**，选择它，然后选择 Next: Review。

AWSCodeDeployRole 策略为您的服务角色提供针对以下内容的权限：

- 阅读您实例上的标签或通过 Auto Scaling 组名称来识别您的 Amazon EC2 实例。
 - 将信息发布到 Amazon SNS 主题。
 - 检索有关 CloudWatch 警报的信息。
 - 检索有关 Elastic Load Balancing 的信息。
7. 在 Review 页面上的 Role name 中，键入服务角色 (例如，**CodeDeployServiceRole**)，and then choose Create role.

您还可以在 Deployment Description 框中键入此服务角色的说明。

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。

如果您希望限制此服务角色访问一些终端节点，则在角色列表中，浏览并选择您刚刚创建的角色，然后继续下一步。

9. 在 Trust relationships 选项卡上，选择 Edit trust relationship。
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"
    }
  ]
}
```

要仅授予服务角色访问支持的某些终端节点的权限，请使用以下策略替换 Policy Document 框中的内容，删除您希望阻止访问的终端节点的行，然后选择 Update Trust Policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-2.amazonaws.com",
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-3.amazonaws.com",
          "codedeploy.ca-central-1.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.eu-west-2.amazonaws.com",
          "codedeploy.eu-central-1.amazonaws.com",
          "codedeploy.ap-northeast-1.amazonaws.com",
          "codedeploy.ap-northeast-2.amazonaws.com",
          "codedeploy.ap-southeast-1.amazonaws.com",
          "codedeploy.ap-southeast-2.amazonaws.com",
          "codedeploy.ap-south-1.amazonaws.com",
          "codedeploy.sa-east-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关创建服务角色的更多信息，请参阅 IAM 用户指南中的[创建向 AWS 服务委托权限的角色](#)。

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 AWS CodeDeploy 代表您执行操作。

执行以下任一操作：

- 要授予对所有支持区域的访问权限，请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"
    }
  ]
}
```

- 要仅授予对部分支持区域的访问权限，请在文件中键入以下内容，然后删除不授予访问权限的区域的行：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-2.amazonaws.com",
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-3.amazonaws.com",
          "codedeploy.ca-central-1.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.eu-west-2.amazonaws.com",
          "codedeploy.eu-central-1.amazonaws.com",
          "codedeploy.ap-northeast-1.amazonaws.com",
          "codedeploy.ap-northeast-2.amazonaws.com",
          "codedeploy.ap-southeast-1.amazonaws.com",
          "codedeploy.ap-southeast-2.amazonaws.com",
          "codedeploy.ap-south-1.amazonaws.com",
          "codedeploy.sa-east-1.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note

请不要在列表中的最后一个终端节点后使用逗号。

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

Be sure to include `file://` before the file name. It is required in this command.

在命令输出中，记录 `Role` 对象下方 `Arn` 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. 21)中的说明操作。

3. 您使用的托管策略取决于 计算平台。

- 如果您要部署到 EC2/本地 计算平台，请：

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

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

- 如果您要部署到 AWS Lambda 计算平台，请：

调用 `attach-role-policy` 命令，根据名为 **AWSCodeDeployRoleForLambda** 的 IAM 托管策略向名为 **CodeDeployServiceRole** 服务角色授予权限。例如：

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

有关创建服务角色的更多信息，请参阅 IAM 用户指南中的[为 AWS 服务创建角色](#)。

获取服务角色 ARN (控制台)

使用 IAM 控制台获取服务角色的 ARN：

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 **Roles**。
3. 在 Filter 框中，键入 **CodeDeployServiceRole**，然后按 Enter。
4. 选择 **CodeDeployServiceRole**。
5. 记录 **Role ARN** 字段的值。

获取服务角色 ARN (CLI)

要使用 AWS 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。

步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件

Note

如果您正在使用 AWS Lambda 计算平台，请跳过此步骤。AWS Lambda 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将由 AWS CodeDeploy 部署的应用程序的 Amazon S3 存储桶或 GitHub 存储库。要启动与 AWS 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。此角色向 AWS CodeDeploy 授予对存储应用程序的 Amazon S3 存储桶或 GitHub 存储库的访问权限。

您可使用 AWS CLI、IAM 控制台或 IAM API 创建 IAM 实例配置文件。

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 Amazon EC2 实例创建 IAM 实例配置文件 \(CLI\)](#) (p. 22)
- [为 Amazon EC2 实例创建 IAM 实例配置文件 \(控制台\)](#) (p. 24)

为 Amazon EC2 实例创建 IAM 实例配置文件 (CLI)

在这些步骤中，我们假定您已遵循[AWS CodeDeploy 入门](#) (p. 16)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.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

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

```
{  
  "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-us-east-2/*",  
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",  
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",  
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",  
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",  
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",  
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",  
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",  
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",  
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",  
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",  
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",  
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",  
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",  
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"   
      ]  
    }  
  ]  
}
```

3. 从同一目录中调用 `create-role` 命令，以基于第一个文件中的信息创建一个名为 **CodeDeployDemo-EC2-Instance-Profile** 的 IAM 角色：

Important

Be sure to include `file://` before the file name. It is required in this command.

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

4. 从同一目录中调用 `put-role-policy` 命令，以基于第二个文件中的信息向名为 **CodeDeployDemo-EC2-Instance-Profile** 的角色提供权限：

Important

Be sure to include `file://` before the file name. It is required in this command.

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

5. 在调用 `create-instance-profile` 命令后调用 `add-role-to-instance-profile` 命令，以创建名为 **CodeDeployDemo-EC2-Instance-Profile** 的 IAM 实例配置文件。此实例配置文件使 Amazon EC2 能够在 Amazon EC2 实例首次启动时将名为 **CodeDeployDemo-EC2-Instance-Profile** 的 IAM 角色传递给该实例：

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

如您需要获得 IAM 实例配置文件的名称，请参阅 AWS CLI 参考中的 IAM 部分的 [list-instance-profiles-for-role](#)。

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 Amazon EC2 用户指南 中的 [Amazon EC2 的 IAM 角色](#)。

为 Amazon EC2 实例创建 IAM 实例配置文件 (控制台)

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 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

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

```
{
  "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-us-east-2/*",
  "arn:aws:s3:::aws-codedeploy-us-east-1/*",
  "arn:aws:s3:::aws-codedeploy-us-west-1/*",
  "arn:aws:s3:::aws-codedeploy-us-west-2/*",
  "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
  "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
  "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
  "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
  "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
  "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
  "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
  "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
  "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
  "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
  "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
]
}
```

4. 选择查看策略。
5. 在创建策略页面上，在策略名称框中键入 **CodeDeployDemo-EC2-Permissions**。
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 Roles，然后选择 Create Role。
9. 在 Create role (创建角色) 页面上，选择 AWS service (AWS 服务)，然后从 Choose the service that will use this role (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。
12. 在 Attached permissions policy 页面上，如果有一个框紧挨着 **CodeDeployDemo-EC2-Permissions**，选择它，然后选择 Next: Review。
13. 在 Review 页面上的 Role name 中，键入服务角色 (例如，**CodeDeployDemo-EC2-Instance-Profile**)，and then choose Create role。

您还可以在 Deployment Description 框中键入此服务角色的说明。

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 Amazon EC2 用户指南 中的 [Amazon EC2 的 IAM 角色](#)。

步骤 5：尝试使用 AWS CodeDeploy 示例部署向导

Note

AWS CodeDeploy 示例部署向导当前不适用于使用 AWS Lambda 计算平台 的部署。

在您完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的前四个步骤之后，请尝试使用 Sample deployment wizard。它将指导您完成创建 AWS CodeDeploy 部署的步骤。Sample deployment wizard 可让您尝试就地部署和蓝/绿部署。

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订版，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2/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. 3)。
- 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：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：
 - 系统将为替代环境配置实例。
 - 替代实例上将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来说，等待时间可选。
 - 替代环境中的实例在 Elastic Load Balancing 负载均衡器中进行注册，使得流量重新路由至这些实例。系统将撤销原始环境中的实例注册，进而终止或因其他使用情形而保持运行。

Note

在使用 EC2/本地 计算平台时，蓝/绿部署只能与 Amazon EC2 实例配合使用。

- AWS Lambda 计算平台上的蓝/绿部署：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方法。所有 AWS Lambda 计算平台部署都是蓝/绿部署。因此，您无需指定部署类型。

有关蓝/绿部署的更多信息，请参阅[蓝/绿部署概述](#) (p. 4)。

对于这两种部署类型示例，我们都假定您之前未使用过 AWS CodeDeploy 并且尚未创建任何资源，例如 AWS CodeDeploy 中的应用程序、应用程序修订或部署组。

这些主题引用的是 AWS CodeDeploy 独有的资源和概念。要在开始前熟悉这些组件和概念，请参阅[Primary Components](#) (p. 6)。

先决条件

如果您希望 AWS CodeDeploy 创建一些示例 Amazon EC2 实例，则必须具有 Amazon EC2 实例密钥对。要创建 Amazon EC2 实例密钥对，请按照[使用 Amazon EC2 创建密钥对](#)。确保在 AWS General Reference 中的[区域和终端节点](#)中列出的某个区域中创建 Amazon EC2 实例密钥对。在启动向导之前，必须创建 Amazon EC2 实例密钥对。否则，它将不会出现在示例部署向导中的 Key pair name 下拉列表中。

如果您使用 AWS CloudFormation 模板启动 Amazon EC2 实例，则调用的 IAM 用户必须具有对 AWS CloudFormation 以及 AWS CloudFormation 所依赖的 AWS 服务和操作的访问权限。如果您尚未执行[步骤 1：预置 IAM 用户](#) (p. 16)中的步骤来预置调用的 IAM 用户，则必须至少附加以下策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "codedeploy:*",
        "ec2:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam>DeleteInstanceProfile",
        "iam>DeleteRole",
        "iam>DeleteRolePolicy",
        "iam:GetRole",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": "*"    
  }  
]  
}
```

该策略的以下部分向调用 IAM 的用户授予对创建服务角色所需的 IAM 操作的访问权。

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

该策略的以下部分向调用 IAM 的用户授予创建应用程序和部署组以及部署应用程序的权限。

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

不是您要找的内容？

- 要创建使用 AWS CodeDeploy 中的现有应用程序、修订、部署组或自定义部署配置的部署，请按照[使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)中的说明执行操作。
- 要练习部署到本地实例而非 Amazon EC2 实例，请参阅[教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \) \(p. 79\)](#)。

主题

- [在 AWS CodeDeploy 中尝试使用示例蓝/绿部署 \(p. 27\)](#)
- [在 AWS CodeDeploy 中尝试示例就地部署 \(p. 31\)](#)

在 AWS CodeDeploy 中尝试使用示例蓝/绿部署

本部分将指导您完成使用 Sample deployment wizard 对一个或多个 Amazon EC2 实例部署修订所需的步骤，然后运行蓝/绿部署，以将部署组中的原始实例组 (原始环境) 替换为一组不同的实例 (替换环境)。

主题

- [启动向导 \(p. 28\)](#)
- [步骤 1：开始使用 \(p. 28\)](#)
- [步骤 2：选择部署类型 \(p. 28\)](#)

- 步骤 3：创建蓝/绿部署 (p. 28)
- 步骤 4：监控蓝/绿部署 (p. 29)
- 步骤 5：刷新 Web 应用程序窗口 (p. 29)
- 步骤 6：清除示例资源 (p. 30)

启动向导

To start the wizard:

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. If an introductory page appears, choose Get Started Now. If the Applications page appears, in More info, choose Sample deployment wizard.

步骤 1：开始使用

选择 Sample deployment，然后选择 Next。

步骤 2：选择部署类型

选择 Blue/green deployment，然后选择 Next。

步骤 3：创建蓝/绿部署

在 Step 3: Create blue/green deployment 页面上，我们提供了您进行蓝/绿部署需要的大多数组件的默认值。

1. 在以下字段中接受或更改任意默认名称：

Note

由于这些字段有不同的验证要求，因此为了节约时间，我们建议保留示例部署的默认名称。

字段	描述
应用程序名称	一个充当容器的名称，可确保使用正确的部署选项组合部署正确的内容。
部署组名称	表示要部署到的实例组的名称。
Auto Scaling 组名称	将为部署环境预置 Amazon EC2 实例的 Auto Scaling 组的名称。
负载均衡器名称	Elastic Load Balancing 中的 传统负载均衡器 用于将流量路由至部署环境中的实例。
服务角色名称	服务角色的名称，该角色将为 AWS CodeDeploy 提供在部署过程中访问其他 AWS 服务所需的权限。

2. 从 Key pair name 下拉列表中，选择将用于连接到 Amazon EC2 实例的 Amazon EC2 实例密钥对。

Note

要创建 Amazon EC2 实例密钥对，请按照[使用 Amazon EC2 创建密钥对](#)。请确保在 AWS General Reference 中的[区域和终端节点](#)中列出的某个区域中创建密钥对。新的 Amazon EC2 实例密钥对可能不会显示在 Key pair name 下拉列表中，直至您重新启动向导。

3. 选择 Launch environment。

4. 在环境准备就绪后，查看 Congratulations! Your environment is ready 下的详细信息。

Note

为便于以后参考，我们建议您将 Congratulations! Your environment is ready 区域中的整个文本复制到计算机上的文件中。

- 为您创建的 传统负载均衡器 的名称，例如 BlueGreenLoadBalancer-abcdefh。
- 为您的原始环境创建的 Auto Scaling 组的名称，例如 CodeDeployBGStack-abcdefh-BlueGreenAutoScalingGroup-1IJKLMN234056。
- 在您的原始环境中部署的应用程序的 Web 地址，例如 <http://BlueGreenLoadBalancer-abcdefh-1234567890.us-east-2.elb.amazonaws.com>。

Important

现在，我们建议您在不同的浏览器窗口中打开您的 Web 应用程序，以便稍后可以刷新它来查看蓝/绿部署期间所做的背景颜色更改。

- 用于创建环境的 AWS CloudFormation 堆栈的名称，例如 BlueGreenLoadBalancer-abcdefh。当您准备好从示例蓝/绿部署中清除资源后，将使用 AWS CloudFormation 控制台删除此堆栈。
- 将在原始环境中安装的示例应用程序的位置，例如：

https://s3.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip

5. 选择 Start blue/green deployment。

步骤 4：监控蓝/绿部署

在 Deployment 页面上，您可以以控制面板格式查看蓝/绿部署的进度。

Deployment progress 区域报告部署中四个主要步骤的进度：

- 在替换环境中预置实例。
- 在替换环境中安装新应用程序修订。
- 将流量重新路由到替换环境。
- 在原始环境中终止实例。

Instances receiving traffic 区域报告在原始环境和替换环境中目前向负载均衡器注册的实例的计数。

Deployment details 区域列出有关在替换环境中安装的部署和应用程序修订的识别信息。

Instance activity 区域提供有关原始环境和替换环境中的每个实例的详细信息。

步骤 5：刷新 Web 应用程序窗口

在您之前打开了安装在原始环境中的应用程序的视图的浏览器窗口中（例如 <http://BlueGreenLoadBalancer-abcdefh-1234567890.us-east-2.elb.amazonaws.com>），选择浏览器的 Refresh 按钮。

如果网页的背景颜色从蓝色变为绿色，则表示流量已成功从我们在 [步骤 3：创建蓝/绿部署](#) (p. 28) 中为您创建的实例路由到在 [步骤 4：监控蓝/绿部署](#) (p. 29) 的过程中创建的替换实例。

步骤 6：清除示例资源

要避免将来收费，您必须清除本向导中使用的资源。必须按以下顺序清除资源：

- 替换环境的实例所属的 Auto Scaling 组。(删除 AWS CloudFormation 堆栈时，与原始环境中的实例关联的 Auto Scaling 组也将被删除。)
- Sample deployment wizard 创建的用于为蓝/绿部署提供原始环境的 AWS CloudFormation 堆栈。
- Sample deployment wizard 创建的 AWS CodeDeploy 部署组 and 应用程序。

删除替换环境的 Auto Scaling 组

您将在 Amazon EC2 控制台中看到两个与示例蓝/绿部署关联的 Auto Scaling 组。为了避免错误，请确保在此步骤中删除与替换环境关联的 Auto Scaling 组。您可以按 Auto Scaling 组的格式来区分它们：

- 删除使用此格式的 Auto Scaling：

```
CodeDeploy_BlueGreenDemoFleet-9zyxwvut_d-ZY9XWVUTS8R
```

- 您可以立即删除使用此格式的 Auto Scaling 组，也可以让它随着 AWS CloudFormation 堆栈的删除而被删除：

```
CodeDeployBGStack-abcdefgh-BlueGreenAutoScalingGroup-1IJKLMN234056
```

1. 打开 Amazon EC2 控制台 <https://console.aws.amazon.com/ec2/>。
2. 在导航窗格中的 Auto Scaling 下，选择 Auto Scaling Groups。
3. 在 Auto Scaling 组页面上，选中为替换环境创建的 Auto Scaling 组旁边的框。例如：

```
CodeDeploy_BlueGreenDemoFleet-9zyxwvut_d-ZY9XWVUTS8R。
```

从 Actions 菜单中选择 Delete。

4. 当系统提示进行确认时，选择 Yes, Delete。

删除 AWS CloudFormation 堆栈

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台：<https://console.aws.amazon.com/cloudformation>。
2. 选中为蓝/绿部署创建的堆栈旁边的框。例如：

```
CodeDeployBGStack-abcdefgh-BlueGreenAutoScalingGroup-1IJKLMN234056。
```

从 Actions 菜单中选择 Delete Stack。

3. 在系统提示时，选择 Yes, Delete。在 Amazon EC2、AWS Identity and Access Management、Amazon VPC 和 Elastic Load Balancing 中为此部署创建的其余资源将被删除。

删除 AWS CodeDeploy 蓝/绿部署资源记录

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。

3. 在 Applications 页面上，选择要删除的应用程序，例如 BlueGreenDemoApplication。
4. 在 Application details 页面上的 Deployment groups 中，选择要删除的部署组旁边的按钮，例如 BlueGreenDemoFleet-1abcdef。在 Actions 菜单上，选择 Delete。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。
5. 在 Application details 页的底部，选择 Delete application。
6. 在系统提示时，键入应用程序的名称，然后选择 Delete。

有关应用程序及其关联的部署组、修订和部署的所有记录将被删除。

在 AWS CodeDeploy 中尝试示例就地部署

本部分将指导您使用 Sample deployment wizard 完成将修订部署到一个或多个 Amazon EC2 实例所需的步骤。

主题

- [示例 AWS CodeDeploy 就地部署的视频 \(p. 31\)](#)
- [启动向导 \(p. 31\)](#)
- [步骤 1：开始使用 \(p. 31\)](#)
- [步骤 2：选择部署类型 \(p. 32\)](#)
- [步骤 3：配置实例 \(p. 32\)](#)
- [步骤 4：为您的应用程序命名 \(p. 32\)](#)
- [步骤 5：选择修订 \(p. 32\)](#)
- [步骤 6：创建部署组 \(p. 32\)](#)
- [步骤 7：选择服务角色 \(p. 33\)](#)
- [步骤 8：选择部署配置 \(p. 33\)](#)
- [步骤 9：审核部署详细信息 \(p. 33\)](#)
- [清除示例就地部署资源 \(p. 34\)](#)

示例 AWS CodeDeploy 就地部署的视频

本简短视频 (5:01) 将指导您使用 AWS CodeDeploy 控制台完成示例 AWS CodeDeploy 就地部署。

[AWS CodeDeploy 就地部署的视频。](#)

启动向导

To start the wizard:

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. If an introductory page appears, choose Get Started Now. If the Applications page appears, in More info, choose Sample deployment wizard.

步骤 1：开始使用

选择 Sample deployment，然后选择 Next。

步骤 2：选择部署类型

选择 In-place deployment，然后选择 Next。

步骤 3：配置实例

如果您拥有已配置为在 AWS CodeDeploy 部署中使用的 Amazon EC2 实例，选择 Skip，读取并遵循说明，然后继续[步骤 4：为您的应用程序命名 \(p. 32\)](#)。

如果您希望 AWS CodeDeploy 启动一组新的 Amazon EC2 实例：

1. 在 Operating system 的旁边，选择 Amazon Linux 或 Windows Server。

Important

您可能需要为由 AWS CodeDeploy 启动的 Amazon EC2 实例付费，因此请确保在完成本向导后终止这些实例。在本向导中，使用一个 AWS CloudFormation 模板启动这些 Amazon EC2 实例。要删除创建的 AWS CloudFormation 堆栈以启动 Amazon EC2 实例，请参阅[删除 AWS CloudFormation 控制台中的堆栈](#)。堆栈名称的开头将为 CodeDeploySampleStack。

2. 从 Key pair name 下拉列表中，选择将用于连接到 Amazon EC2 实例的 Amazon EC2 实例密钥对。

Note

要创建 Amazon EC2 实例密钥对，请按照[使用 Amazon EC2 创建密钥对](#)。请确保在 AWS General Reference 中的[区域和终端节点](#)中列出的某个区域中创建密钥对。新的 Amazon EC2 实例密钥对可能不会显示在 Key pair name 下拉列表中，直至您重新启动向导。

3. 保留 Tag key and value 的默认值。AWS CodeDeploy 将在部署期间使用此标签密钥和值来查找实例。

如果您需要覆盖建议的标签密钥和值 (例如，如果您多次运行本向导但未终止任何之前创建的 Amazon EC2 实例)，建议您在 Key 框中保留 Name 的标签密钥，并在 Value 框中键入其他标签值。有关 Amazon EC2 实例标签的信息，请参阅[标记您的 Amazon EC2 资源](#)。

4. 选择 Launch instances。

如果您选择 See more details in AWS CloudFormation，则 AWS CloudFormation 控制台将在单独的 Web 浏览器选项卡中打开。查找以 CodeDeploySampleStack 开始的堆栈。当 CREATE_COMPLETE 出现在 Status 列中时，已启动您的 Amazon EC2 实例。(这可能需要花几分钟的时间。)

5. 要继续，请选择 Next。

步骤 4：为您的应用程序命名

在 Application name 框中，保留建议的应用程序名称或者键入其他名称 (如果您愿意)，然后选择 Next。

步骤 5：选择修订

检查有关示例应用程序修订的信息，并选择 Next。

Note

如果您需要检查示例修订的内容，请选择 Download sample bundle，然后按照 Web 浏览器的说明执行操作来下载和查看内容。

如果您已在 [步骤 3：配置实例 \(p. 32\)](#) 中选择 Skip，请从 Revision type 下拉列表中选择与 Amazon EC2 实例类型 (Amazon Linux 或 Windows Server) 对应的应用程序修订的类型。

步骤 6：创建部署组

1. 在 Deployment group name 框中，保留建议的部署组名称或者键入其他名称 (如果您愿意)。
2. Configure instances 页 (例如，Name 和 CodeDeployDemo) 中指定的键值对的键和值应出现。

如果您已在 [步骤 3：配置实例 \(p. 32\)](#) 中选择 Skip，请在 Add instances 中，用 Amazon EC2 实例的键值对的键和值覆盖 Key 和 Value 框的值。

(可选) 如果 Amazon EC2 实例具有多个密钥值对，则可在空白行中键入这些密钥值对。将显示一个新的空白行，以便能添加其他密钥值对。最多可添加 10 个密钥值对。选择删除图标可从列表中删除密钥值对。

Note

AWS CodeDeploy 显示与每个密钥值对匹配的实例的数目。要在 Amazon EC2 控制台中查看实例，请单击数字。

如果您使用 AWS CloudFormation 模板启动新的 Amazon EC2 实例，并且数字大于您预期的数字，请选择 Cancel，请从头开始向导，并在 [步骤 3：配置实例 \(p. 32\)](#) 中，指定一个不同于默认值的标签值。(请确保删除 AWS CloudFormation 堆栈以终止 Amazon EC2 实例。)

如果您使用自己的 Amazon EC2 实例，则将一个新的标签密钥和值添加到您的 Amazon EC2 实例，然后在 Add instances 中指定一个不同于默认值的标签密钥和值。

3. 如果您拥有一个要添加到部署组的 Auto Scaling 组，请选择 Search by Auto Scaling group names，然后键入 Auto Scaling 组名称。可以添加最多 10 个 Auto Scaling 组。选择删除图标可从列表中删除 Auto Scaling 组。

Note

AWS CodeDeploy 显示与每个 Auto Scaling 组名匹配的 Amazon EC2 实例数。要在 Amazon EC2 控制台中查看实例，请单击数字。

4. 选择 Next。

步骤 7：选择服务角色

选择 Create a service role 或 Use an existing service role。

如果您首次使用本向导，则建议您选择 Create a service role，选择 Next 以接受默认名称，然后继续 [步骤 8：选择部署配置 \(p. 33\)](#)。

如果您已拥有服务角色，则选择 Use an existing service role，从 Role name 下拉列表中选择该角色，然后选择 Next。

步骤 8：选择部署配置

1. 要对此部署使用内置配置，请选择 Use a default deployment configuration。要为此部署创建您自己的配置，请选择 Create a custom deployment configuration。
2. 如果您已选择 Use a default deployment configuration 并希望使用不同于所选配置的其他配置，请在所需配置的旁边选择 Select。选择 Next，然后转至 [步骤 9：审核部署详细信息 \(p. 33\)](#)。
3. 如果您已选择 Create a custom deployment configuration：
 - a. 在 Deployment configuration name 框中，键入配置的唯一名称。
 - b. 使用 Number 或 Percentage 框可键入部署期间可用的总 Amazon EC2 实例数或百分比。
 - c. 选择 Next。

步骤 9：审核部署详细信息

1. 如果您需要进行更改，请选择某个 Edit 链接。在进行更改后，请选择 Next 直至返回到 Review deployment details 页，然后选择 Deploy。
2. 选择表旁边的 Refresh 按钮以获取部署状态。要获取有关部署的信息，请参阅 [查看实例详细信息 \(控制台\) \(p. 179\)](#)。

3. 我们的示例修订为每个实例部署一个网页。通过转至每个实例的 `http://PublicDNS` (例如, `http://ec2-01-234-567-890.compute-1.amazonaws.com`) , 可以使用您的 Web 浏览器来验证部署是否成功。网页将显示一条祝贺消息。

要获取公有 DNS 值, 请在 Amazon EC2 控制台中, 选择 Amazon EC2 实例。在 Description 选项卡上, 在 Public DNS 中查找该值。

清除示例就地部署资源

要避免将来收费, 您必须清除本向导中使用的资源。如果您已使用 AWS CloudFormation 模板启动 Amazon EC2 实例, 请删除 AWS CloudFormation 堆栈。这将终止实例及其关联的资源。

如果您刚刚已为本向导启动您自己的 Amazon EC2 实例, 则应终止这些实例。(可选) 您可以从 AWS CodeDeploy 控制台中删除与本向导关联的部署组件记录。

删除 AWS CloudFormation 堆栈

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台: <https://console.aws.amazon.com/cloudformation>。
2. 选择以 `CodeDeploySampleStack` 开头的堆栈旁边的按钮。在 Actions 菜单上, 选择 Delete Stack。
3. 在系统提示时, 选择 Yes, Delete。这将终止 Amazon EC2 实例。关联的 IAM 实例配置文件和服务角色将被删除。

终止 Amazon EC2 实例

1. 登录 AWS 管理控制台并通过以下网址打开 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. 选中要终止的每个 Amazon EC2 实例的框。
4. 在 Actions 菜单上, 指向 Instance State, 然后选择 Terminate。
5. 在系统提示时, 选择 Yes, Terminate。

删除 AWS CodeDeploy 部署组件记录

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>。

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#)。

2. 如果未显示 Applications 页, 请在 AWS CodeDeploy 菜单上选择 Applications。
3. 在 Applications 页上, 选择要删除的应用程序。
4. 在 Application details 页上, 在 Deployment groups 中, 选择要删除的部署组旁边的按钮。在 Actions 菜单上, 选择 Delete。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。
5. 在 Application details 页的底部, 选择 Delete application。
6. 在系统提示时, 键入应用程序的名称, 然后选择 Delete。

有关应用程序及其关联的部署组、修订和部署的所有记录将被删除。

AWS CodeDeploy 产品和服务集成

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

- [与其他 AWS 服务集成 \(p. 35\)](#)
- [与合作伙伴产品和服务集成 \(p. 42\)](#)
- [来自社区的集成示例 \(p. 47\)](#)

与其他 AWS 服务集成

AWS CodeDeploy 与下列 AWS 服务集成：

Amazon CloudWatch	<p>Amazon CloudWatch 是面向 AWS 云资源以及在 AWS 上运行的应用程序的监控服务。Amazon CloudWatch 可用于收集和跟踪指标，收集和监控日志文件，以及设置警报。AWS CodeDeploy 支持以下 CloudWatch 工具：</p> <ul style="list-style-type: none">• 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，您需要在 CloudWatch 中设置警报，然后在 AWS 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。 <p>了解更多：</p> <ul style="list-style-type: none">• 创建 CloudWatch Logs 警报 • Amazon CloudWatch Events：用于在您的 AWS CodeDeploy 操作中检测实例或部署状态的更改并采取相应的操作。然后 CloudWatch Events 将在部署或实例进入您在规则中指定的状态时，根据您创建的规则调用一个或多个目标操作。 <p>了解更多：</p> <ul style="list-style-type: none">• 使用 Amazon CloudWatch Events 监控部署 (p. 246) • Amazon CloudWatch Logs：用于监控由 AWS CodeDeploy 代理创建的三种类型的日志，而不必逐一登录到实例。 <p>了解更多：</p> <ul style="list-style-type: none">• 在 CloudWatch Logs 控制台中查看 AWS CodeDeploy 日志
Auto Scaling	<p>AWS CodeDeploy 支持 Amazon EC2 Auto Scaling，后者是一种 AWS Web 服务，可根据您指定的条件（例如，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）自动启动 Amazon EC2 实例。这样，您便可根据需要扩展一组 Amazon EC2 实例，然后使</p>

	<p>用 AWS CodeDeploy 自动将应用程序修订部署到这些额外的 Amazon EC2 实例。当不再需要这些 Amazon EC2 实例时，Auto Scaling 会终止它们。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 将 AWS CodeDeploy 与 Auto Scaling 集成 (p. 38)• 教程：使用 AWS CodeDeploy 将应用程序部署到 Auto Scaling 组 (p. 86)• 深入剖析：AWS CodeDeploy 和 Auto Scaling 集成
AWS CloudTrail	<p>AWS CodeDeploy 与 AWS CloudTrail 集成在一起，后者是一种服务，可在 AWS 账户中捕获由 AWS CodeDeploy 自身或代表其发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 从 AWS CodeDeploy 控制台、通过 AWS CLI 从 AWS CodeDeploy 命令或直接从 AWS CodeDeploy API 捕获 API 调用。通过 CloudTrail 收集的信息，您可以确定向 AWS 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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• Monitoring Deployments with AWS CloudTrail (p. 248)
AWS Cloud9	<p>AWS Cloud9 是一个基于云的在线集成开发环境 (IDE)，可用于编写、运行、调试和部署代码 (只需使用连接到 Internet 的计算机中的浏览器即可)。AWS Cloud9 包含代码编辑器、调试程序、终端和基本工具，例如 AWS CLI 和 Git。</p> <ul style="list-style-type: none">• 您可以使用 AWS 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 AWS 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 AWS CodeDeploy 来部署更新。有关将 AWS Cloud9 与 GitHub 结合使用的更多信息，请参阅适用于 AWS Cloud9 的 GitHub 示例。• 可使用 AWS Cloud9 IDE 更新 AWS Lambda 函数。然后，您可以使用 AWS CodeDeploy 创建一个部署，该部署将流量转移到 AWS Lambda 函数的新版本。有关更多信息，请参阅在AWS Cloud9 集成开发环境 (IDE) 中使用 AWS Lambda 函数。 <p>有关 AWS Cloud9 的更多信息，请参阅什么是 AWS Cloud9 和 AWS Cloud9 入门。</p>

AWS CodePipeline	<p>AWS 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 AWS CodePipeline 定义您自己的发布过程，以便服务在每次发生代码更改时构建、测试和部署代码。例如，一个应用程序可以有三个部署组：Beta、Gamma 和 Prod。您可以设置管道，以便每次源代码发生更改时，将更新逐一部署到每个部署组。</p> <p>您可以将 AWS CodePipeline 配置为使用 AWS CodeDeploy 来：</p> <ul style="list-style-type: none">• 将代码部署到 Amazon EC2 实例和/或本地实例。• 部署无服务器 AWS Lambda 函数版本。 <p>在您创建管道之前或在创建管道向导中时，您可以创建 AWS CodeDeploy 应用程序、部署和部署组以在阶段的部署操作中使用。</p> <p>了解更多：</p> <ul style="list-style-type: none">• AWS for DevOps 入门指南 - 了解如何将 AWS CodePipeline 与 AWS CodeDeploy 配合使用以持续将 AWS CodeCommit 存储库中的源代码传输和部署到 Amazon EC2 实例。• 简单管道演练 (Amazon S3 存储桶)• 简单管道演练 (AWS CodeCommit 存储库)• 四阶段管道教程
AWS 无服务器应用程序模型	<p>AWS 无服务器应用程序模型 (AWS SAM) 是定义无服务器应用程序的模型。它能够扩展 AWS CloudFormation，以提供简化方式来定义无服务器应用程序所需的 AWS Lambda 函数、Amazon API Gateway API 和 Amazon DynamoDB 表。如果您已使用 AWS SAM，则可以添加部署首选项，以便开始使用 AWS CodeDeploy 管理 AWS Lambda 应用程序部署期间的流量转移方式。</p> <p>有关更多信息，请参阅 AWS 无服务器应用程序模型。</p>
Elastic Load Balancing	<p>AWS CodeDeploy 支持 Elastic Load Balancing，后者是一种可跨多个 Amazon EC2 实例分发传入的应用程序流量的服务。</p> <p>对于 AWS 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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40)

主题

- [将 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)
- [将 AWS CodeDeploy 与 Elastic Load Balancing 集成 \(p. 40\)](#)

将 AWS CodeDeploy 与 Auto Scaling 集成

AWS CodeDeploy 支持 Auto Scaling，后者是一种 AWS 服务，可根据您定义的条件自动启动 Amazon EC2 实例。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。Auto Scaling 会终止不再需要的实例。有关更多信息，请参阅[什么是 Auto Scaling？](#)

当新 Amazon EC2 实例作为 Auto Scaling 组一部分启动时，AWS CodeDeploy 可自动将您的修订部署到这些新实例。您还可以将 AWS CodeDeploy 中的部署与注册到 Elastic Load Balancing 负载均衡器的 Amazon EC2 实例进行协调。有关更多信息，请参阅[Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 40\)](#) 和 [在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)。

Note

请注意，如果您将多个部署组与单个 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 Auto Scaling 组关联 \(p. 322\)](#)和[深层剖析：AWS CodeDeploy 与 Auto Scaling 集成](#)。

主题

- [将 AWS CodeDeploy 应用程序部署到 Auto Scaling 组 \(p. 38\)](#)
- [使用 AWS CodeDeploy 的 Auto Scaling 行为 \(p. 38\)](#)
- [将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling \(p. 39\)](#)

将 AWS CodeDeploy 应用程序部署到 Auto Scaling 组

要将 AWS CodeDeploy 应用程序修订部署到 Amazon EC2 Auto Scaling 组，请执行以下操作：

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

Note

您还可以使用 AWS CodeDeploy 将修订从 GitHub 存储库部署到 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[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)。

2. 创建或使用 Auto Scaling 组，并指定 IAM 实例配置文件。
3. 创建或找到允许 AWS CodeDeploy 创建一个包含 Auto Scaling 组的部署组的服务角色。
4. 使用 AWS CodeDeploy 创建一个部署组，并指定 Auto Scaling 组名称和服务角色。
5. 使用 AWS CodeDeploy 将您的修订部署到包含 Auto Scaling 组的部署组。

有关更多信息，请参阅[教程：使用 AWS CodeDeploy 将应用程序部署到 Auto Scaling 组 \(p. 86\)](#)。

使用 AWS CodeDeploy 的 Auto Scaling 行为

自定义生命周期挂钩事件的执行顺序无法预先确定

您可以向 AWS CodeDeploy 部署到的 Auto Scaling 组中添加自己的生命周期挂钩。但是，这些自定义生命周期挂钩事件相对于 AWS CodeDeploy 默认部署生命周期事件的执行顺序无法预先确定。例如，如果您将一个名为 `ReadyForSoftwareInstall` 的自定义生命周期挂钩添加到 Auto Scaling 组中，您无法提前知道它是在第一个 AWS 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 Auto Scaling 组中，请参阅[添加生命周期挂钩](#)。

部署期间的纵向扩展事件会产生混合环境

如果在部署过程中发生 Auto Scaling 纵向扩展事件，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，则旧实例和最近纵向扩展的实例将会托管不同的应用程序修订。

要解决发生的此问题，您可以将更新的应用程序修订重新部署到受影响的部署组。

为避免此问题，我们建议您在执行部署时暂停 Auto Scaling 纵向扩展过程。您可以通过用于 AWS CodeDeploy 负载均衡的 `common_functions.sh` 脚本中的设置来完成此操作。如果 `HANDLE_PROCS=true`，则以下 Auto Scaling 事件在部署过程中将会自动暂停：

- AZRebalance
- AlarmNotification
- ScheduledActions
- ReplaceUnhealthy

Important

只有 `CodeDeployDefault.OneAtATime` 部署配置支持此功能。如果您正在使用其他部署配置，部署组的实例可能仍会应用不同的应用程序修订。

有关使用 `HANDLE_PROCS=true` 以避免在使用 Auto Scaling 时出现部署问题的更多信息，请参阅 GitHub 上 [aws-codedeploy-samples](#) 中的[有关处理 AutoScaling 流程的重要通知](#)。

使用 AWS CloudFormation cfn-init 脚本时必须控制事件的顺序

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

该顺序必须是：

1. 新预配的实例启动。
2. 所有 `cfn-init` 引导脚本运行直至完成。
3. AWS 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，则 AWS 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 `cfn-init` 脚本安装 AWS CodeDeploy 代理，将其放在所有其他脚本之后。
- 在定制 AMI 中包括 AWS 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 `cfn-init` 的信息，请参阅 [AWS CloudFormation 用户指南](#) 中的 `cfn-init`。

将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling

指定要在 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：

- 可以指定已安装 AWS 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 Amazon EC2 实例的速度更快。但是，此选项提高了 Amazon EC2 实例的初始部署将失败的可能性，尤其在 AWS CodeDeploy 代理已过时的情况下。如果选择此选项，建议您定期更新自定义的基本 AMI 中的 AWS CodeDeploy 代理。

- 可以指定未安装 AWS CodeDeploy 代理而是在 Auto Scaling 组中启动每个新实例时安装该代理的基本 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将成功的可能性。此选项使用最新版本的 AWS CodeDeploy 代理。

将 AWS CodeDeploy 与 Elastic Load Balancing 集成

Elastic Load Balancing 提供了三种可用于 AWS CodeDeploy 部署的负载均衡器：Classic Load Balancer、Application Load Balancer 和 Network Load Balancer。

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-Classical 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。

Network Load Balancer

路由和负载均衡在传输层 (TCP/UDP 层，即第 4 层) 进行，依据是从 TCP 数据包标头中而非从数据包内容中提取的地址信息。Network Load Balancer 可以处理突发流量、保留客户端的源 IP 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 传统负载均衡器？](#)
- [什么是 应用程序负载均衡器？](#)
- [什么是 Network Load Balancer？](#)

负载均衡器在 AWS CodeDeploy 部署中的作用

在 AWS 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。

蓝/绿部署

依托于 Elastic Load Balancing 负载均衡器来重新路由实例流量是 AWS 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 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer。您可以使用 AWS CodeDeploy 控制台或 AWS 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 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)
- [为蓝/绿部署创建应用程序 \(控制台\) \(p. 192\)](#)
- [为蓝/绿部署创建部署组 \(控制台\) \(p. 200\)](#)

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 Load Balancing 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，并在部署成功后向负载均衡器重新注册同一部署组。

对于就地部署，您可以指定 传统负载均衡器、应用程序负载均衡器 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 AWS CodeDeploy 提供的脚本实施负载均衡器。

可以使用 AWS CodeDeploy 控制台或 AWS CLI 将负载均衡器添加到部署组。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 \(控制台\) \(p. 190\)](#)
- [为就地部署创建部署组 \(控制台\) \(p. 199\)](#)
- [在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)

有关使用脚本在就地部署中指定负载均衡器的信息，请参阅以下主题：

- [使用脚本为就地部署设置负载均衡器 \(p. 41\)](#)

使用脚本为就地部署设置负载均衡器

通过执行以下过程中的步骤，使用部署声明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 AWS CodeDeploy 中使用部署配置 \(p. 184\)](#)。

在 GitHub 上的 AWS 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 AWS CodeDeploy Elastic Load Balancing 负载均衡器。这些存储库包含三个示例脚本 (`register_with_elb.sh`、`deregister_from_elb.sh` 和 `common_functions.sh`)，这些脚本提供了开始操作所需的全部代码。只需编辑这 3 个脚本中的占位符，然后从 `appspec.yml` 文件中引用这些脚本。

要使用已注册到 Elastic Load Balancing 负载均衡器的 Amazon EC2 实例设置 AWS CodeDeploy 中的就地部署，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. 确保每个目标 Amazon EC2 实例均已安装 AWS CLI。
3. 确保每个目标 Amazon EC2 实例均已附加至少一个 IAM 实例配置文件，并且具有 `elasticloadbalancing:*` 和 `autoscaling:*` 权限。
4. 将部署生命周期事件脚本 (`register_with_elb.sh`、`deregister_from_elb.sh` 和 `common_functions.sh`) 包含在应用程序的源代码目录中。
5. 在应用程序修订的 `appspec.yml` 中，提供有关 AWS CodeDeploy 在 `ApplicationStart` 事件期间运行 `register_with_elb.sh` 脚本以及在 `ApplicationStop` 事件期间运行 `deregister_from_elb.sh` 脚本的说明。

6. 如果实例属于某个 Auto Scaling 组，您可以跳过此步骤。

在 `common_functions.sh` 脚本中：

- 如果您使用的是 [传统负载均衡器](#)，请在 `ELB_LIST=""` 中指定 Elastic Load Balancing 负载均衡器的名称，并对文件中的其他部署设置进行所需的任何更改。
 - 如果您使用的是 [应用程序负载均衡器](#) 或 [Network Load Balancer](#)，请在 `TARGET_GROUP_LIST=""` 中指定 Elastic Load Balancing 目标组名称，然后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。

与合作伙伴产品和服务集成

AWS CodeDeploy 内置集成了下列合作伙伴产品和服务：

Ansible	<p>如果您已有一套 Ansible 操作手册，只需在某个位置运行它们，那么借助适用于 Ansible 和 AWS 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。此外，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 AWS CodeDeploy 代理。</p> <p>了解更多：</p> <ul style="list-style-type: none">• Ansible 和 AWS CodeDeploy
Atlassian – Bamboo 和 Bitbucket	<p>Bamboo 的 AWS CodeDeploy 任务是将包含 AppSpec file 的目录压缩成一个 .zip 文件，将此文件上传到 Amazon S3，然后根据在 AWS CodeDeploy 应用程序中提供的配置开始部署。</p> <p>AWS CodeDeploy 的 Atlassian Bitbucket 支持使您能够直接从 Bitbucket UI 根据需要将代码推送到 Amazon EC2 实例（推送到任何部署组）。这意味着，在您更新 Bitbucket 存储库中的代码之后，您不必登录到持续集成 (CI) 平台或 Amazon EC2 实例即可运行手动部署过程。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 使用 Bamboo 的 AWS CodeDeploy 任务• 推出 AWS CodeDeploy 的 Atlassian Bitbucket 支持
Chef	<p>AWS 提供两个关于将 Chef 与 AWS CodeDeploy 集成的模板示例。第一个是将安装并启动 AWS CodeDeploy 代理的 Chef 说明书。这使您能够在使使用 AWS CodeDeploy 的同时，使用 Chef 继续管理您的主机基础设施。第二个示例模板演示如何使用 AWS CodeDeploy 通过 chef-solo 在每个节点上协调说明书和方法的运行。</p> <p>了解更多：</p>

	<ul style="list-style-type: none">• Chef 和 AWS CodeDeploy
CircleCI	<p>CircleCI 提供一个自动测试和持续集成与部署工具集。在 AWS 中创建 IAM 角色以用于 CircleCI 并在 circle.yml 文件中配置部署参数之后，可以结合使用 CircleCI 和 AWS 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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• AWS CodeDeploy 持续部署
CloudBees	<p>您可以使用 CloudBees DEV@cloud 上提供的 AWS CodeDeploy Jenkins 插件执行构建后操作。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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 目前在 DEV@cloud 上提供 AWS CodeDeploy Jenkins 插件
Codship	<p>您可以使用 Codship 通过 AWS CodeDeploy 部署应用程序修订。您可以使用 Codship UI 将 AWS CodeDeploy 添加到分支的部署管道中。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 部署到 AWS CodeDeploy• 在 Codship 上集成 AWS CodeDeploy
GitHub	<p>您可以使用 AWS CodeDeploy 从 GitHub 存储库部署应用程序修订。只要 GitHub 存储库中的源代码发生了更改，您就可以触发从该存储库的部署。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 将 AWS CodeDeploy 与 GitHub 集成 (p. 45)• 教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序 (p. 101)• 使用 AWS CodeDeploy 自动从 GitHub 部署
HashiCorp Consul	<p>可以使用开源 HashiCorp Consul 工具来帮助确保应用程序环境在 AWS CodeDeploy 中部署应用程序时的运行状况和稳定性。可以使用 Consul 注册要在部署期间发现的应用程序，将应用程序和节点置于维护模式中以将其从部署中排除，并在目标实例变得运行状况不佳时停止部署。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 使用 HashiCorp Consul 执行的 AWS CodeDeploy 部署

Jenkins	<p>AWS CodeDeploy Jenkins 插件提供 Jenkins 项目的构建后步骤。成功构建后，它将压缩工作区，上传到 Amazon S3，并启动新的部署。</p> <p>了解更多：</p> <ul style="list-style-type: none">• AWS CodeDeploy Jenkins 插件• 为 AWS CodeDeploy 设置 Jenkins 插件
Puppet Labs	<p>AWS 提供 Puppet 和 AWS CodeDeploy 的示例模板。第一个是将安装并启动 AWS CodeDeploy 代理的 Puppet 模块。这使您能够在使用 AWS CodeDeploy 的同时，使用 Puppet 继续管理您的主机基础设施。第二个示例模板演示如何使用 AWS CodeDeploy 通过无主 Puppet 在每个节点上协调模块和清单的运行。</p> <p>了解更多：</p> <ul style="list-style-type: none">• Puppet 和 AWS CodeDeploy
SaltStack	<p>您可以将 SaltStack 基础设施与 AWS CodeDeploy 集成。您可以使用 AWS CodeDeploy 模块在自己的设备上安装并运行 AWS CodeDeploy 代理，也可以使用 AWS CodeDeploy 通过几个简单的部署挂钩协调 Salt States 的运行。</p> <p>了解更多：</p> <ul style="list-style-type: none">• SaltStack 和 AWS CodeDeploy
Solano Labs	<p>您的构建在 Solano CI 中通过其测试之后，将运行脚本以准备发布您的应用程序。aws deploy push 命令将通过 AWS CodeDeploy 打包并推送您的应用程序，然后选择性地将应用程序修订部署到部署组并确认其已部署。您还可以通过 CI 构建设置自动 AWS CodeDeploy 部署。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 通过 Solano CI 构建执行 AWS CodeDeploy 部署
TeamCity	<p>可以使用 AWS 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 AWS CodeDeploy 应用程序中注册修订、创建 AWS CodeDeploy 部署，并根据您的需要等待部署完成。</p> <p>了解更多：</p> <ul style="list-style-type: none">• AWS CodeDeploy Runner (下载)• AWS CodeDeploy Runner 插件 (文档)

Travis CI

您可以将 [Travis CI](#) 配置为在成功构建后触发 AWS CodeDeploy 中的部署。

了解更多：

- [Travis CI 和 AWS CodeDeploy 部署](#)

主题

- [将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)

将 AWS CodeDeploy 与 GitHub 集成

AWS CodeDeploy 支持 [GitHub](#)（一种基于 Web 的托管和共享服务）。AWS CodeDeploy 可以将存储在 GitHub 存储库或 Amazon S3 存储桶中的应用程序修订部署到实例。

主题

- [AWS CodeDeploy 与 GitHub 集成视频介绍 \(p. 45\)](#)
- [从 GitHub 部署 AWS CodeDeploy 修订 \(p. 45\)](#)
- [GitHub 与 AWS CodeDeploy 之间的行为 \(p. 46\)](#)

AWS CodeDeploy 与 GitHub 集成视频介绍

本简短视频 (5:20) 说明如何使用 AWS CodeDeploy 从现有的 GitHub 工作流程自动进行应用程序部署。

[AWS CodeDeploy 与 GitHub 集成视频介绍。](#)

从 GitHub 部署 AWS 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 AWS 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[计划 AWS CodeDeploy 的修订 \(p. 208\)](#)和[将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)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[链接 GitHub](#)。要创建 GitHub 存储库，请参阅[创建存储库](#)。

3. 使用 AWS CodeDeploy 控制台中的 Create deployment 页或 AWS CLI create-deployment 命令将修订从 GitHub 存储库部署到已配置为在 AWS CodeDeploy 部署中使用的目标实例。

如果您需要调用 create-deployment 命令，则必须先使用控制台的 Create deployment 页，针对指定的应用程序为 AWS CodeDeploy 授予代表您的首选 GitHub 账户与 GitHub 进行交互的权限。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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

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 \(CLI\) \(p. 227\)](#)。

要了解如何准备实例以在 AWS CodeDeploy 部署中使用，请参阅[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)。

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

GitHub 与 AWS CodeDeploy 之间的行为

主题

- [GitHub 对 AWS CodeDeploy 中的应用程序进行的身份验证](#) (p. 46)
- [AWS CodeDeploy 与私有和公有 GitHub 存储库进行交互](#) (p. 47)
- [AWS CodeDeploy 与组织托管的 GitHub 存储库进行交互](#) (p. 47)
- [使用 AWS CodeDeploy 自动从 GitHub 部署](#) (p. 47)

GitHub 对 AWS CodeDeploy 中的应用程序进行的身份验证

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

将 GitHub 账户链接到 AWS CodeDeploy 中的应用程序

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。
3. 选择 Create deployment。

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。

4. 从 Application 下拉列表中，选择要链接到其他 GitHub 账户的应用程序。
5. 在 Repository type 的旁边，选择 My application is stored in GitHub。
6. 在 Connect to GitHub 中，执行下列操作之一：
 - 要为 AWS 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 AWS 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，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account，然后选择 Connect to GitHub。继续执行步骤 2。
7. 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。
8. 选择 Authorize application。GitHub 向 AWS CodeDeploy 提供代表所选应用程序的已登录 GitHub 账户与 GitHub 交互的权限。
9. 如果您不需要创建部署，请选择 Cancel。

撤销 AWS CodeDeploy 的与 GitHub 交互的权限

1. 使用要撤销 AWS CodeDeploy 权限的 GitHub 账户的凭证登录到 [GitHub](#)。
2. 打开 GitHub [应用程序](#) 页，在已授权应用程序列表中找到 AWS CodeDeploy，然后按照用于撤销应用程序授权的 GitHub 过程执行操作。

AWS CodeDeploy 与私有和公有 GitHub 存储库进行交互

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

AWS CodeDeploy 与组织托管的 GitHub 存储库进行交互

默认情况下，组织托管的 GitHub 存储库（与账户自己的私有或公有存储库相对）未授予对第三方应用程序（包括 AWS 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 AWS CodeDeploy 的访问权。请求此访问权的步骤取决于您是否已授权单个账户访问 AWS CodeDeploy：
 - 如果您已在账户中授予对 AWS CodeDeploy 的访问权，请参阅[请求组织批准您的授权应用程序](#)。
 - 如果您尚未在账户中授予对 AWS CodeDeploy 的访问权，请参阅[请求组织批准第三方应用程序](#)。
- 组织所有者可禁用组织的所有第三方应用程序限制。有关信息，请参阅[禁用组织的所有第三方应用程序限制](#)。

有关更多信息，请参阅[关于第三方应用程序限制](#)。

使用 AWS CodeDeploy 自动从 GitHub 部署

当源代码发生更改时，您可以触发从 GitHub 存储库进行的部署。有关说明，请参阅[使用 AWS CodeDeploy 自动从 GitHub 部署](#)。

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。AWS 对这些内容或外部内容的准确性不承担责任。

博客帖子

- [在 AWS CloudFormation 中自动执行 AWS CodeDeploy 预配](#)

了解如何使用 AWS CloudFormation 在 AWS CodeDeploy 中预配应用程序部署。

发布时间：2016 年 1 月

- [AWS Toolkit for Eclipse 与 AWS CodeDeploy 集成（第 1 部分）](#)

[AWS Toolkit for Eclipse 与 AWS CodeDeploy 集成（第 2 部分）](#)

[AWS Toolkit for Eclipse 与 AWS CodeDeploy 集成（第 3 部分）](#)

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

发布时间：2015 年 2 月

- [使用 AWS CodeDeploy 自动从 GitHub 部署](#)

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

发布时间：2014 年 12 月

视频

- 使用 Docker 和 AWS CodeDeploy 在 AWS 中托管 ASP.NET 5 应用程序

了解如何使用 AWS CodeDeploy 将 ASP.NET 5 应用程序部署到 Microsoft Windows 操作系统上的 Internet Information Services (IIS) 服务器中。

[使用 Docker 和 AWS CodeDeploy 在 AWS 中托管 ASP.NET 5 应用程序](#)

发布时间：2015 年 10 月

持续时间：47 分 37 秒

- 使用 Jenkins 和 Puppet 控制 AWS CodeDeploy

了解如何将开源工具 Jenkins 和 Puppet 用于 AWS CodeDeploy。

[使用 Jenkins 和 Puppet 控制 AWS CodeDeploy](#)

发布时间：2015 年 5 月

持续时间：49 分 31 秒

AWS CodeDeploy 教程

本部分包含的一些教程可帮助您了解如何使用 AWS CodeDeploy。

如果您尚未完成它，建议您从[步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 \(p. 25\)](#)开始。无需具备 AWS CodeDeploy 的使用经验。它指导您完成所需步骤以将其中一个示例应用程序修订部署到 Amazon EC2 实例。

Important

在开始之前，请完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的先决条件。

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 codedeploydemobucket、HelloWorldApp 和 CodeDeployDemo-EC2-Trust.json），但您并不是必须使用它们。在执行这些过程时，请确保替换您的文件位置和名称。

主题

- [教程：将 WordPress 部署到 Amazon EC2 实例 \(Amazon Linux 或 Red Hat Enterprise Linux 和 Linux, macOS, or Unix \) \(p. 49\)](#)
- [教程：使用 部署“Hello, World!”应用程序 AWS CodeDeploy \(Windows Server\) \(p. 66\)](#)
- [教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \) \(p. 79\)](#)
- [教程：使用 AWS CodeDeploy 将应用程序部署到 Auto Scaling 组 \(p. 86\)](#)
- [教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序 \(p. 101\)](#)

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

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

不是您要找的内容？

- 要改为实施针对运行 Windows Server 的 Amazon EC2 实例的部署，请参阅[教程：使用 部署“Hello, World!”应用程序 AWS CodeDeploy \(Windows Server\) \(p. 66\)](#)。
- 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 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \) \(p. 79\)](#)。

本教程建立在[步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 \(p. 25\)](#)中介绍的概念的基础之上。如果您尚未完成该教程，则可能需要先从这里开始。

本教程的步骤是从运行 Linux, macOS, or 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，但您需要改写涉及到命令（例如 chmod 和 wget）、应用程序（例如 sed）和目录路径（例如 /tmp）的步骤。

在开始本教程前，您必须完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的先决条件。这包括配置您的 IAM 用户账户、安装或升级 AWS CLI、创建 IAM 实例配置文件和服务角色。

主题

- [步骤 1：启动和配置 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例 \(p. 50\)](#)
- [步骤 2：配置要部署到 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例的源内容 \(p. 51\)](#)
- [步骤 3：将您的 WordPress 应用程序上传到 Amazon S3 \(p. 55\)](#)
- [步骤 4：部署 WordPress 应用程序 \(p. 58\)](#)
- [步骤 5：更新和重新部署您的 WordPress 应用程序 \(p. 62\)](#)
- [步骤 6：清除 WordPress 应用程序和相关资源 \(p. 64\)](#)

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

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

按照[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)中的说明进行操作。当您进入这些说明中关于向实例分配 Amazon EC2 实例标签的部分时，请确保指定标签密钥 **Name** 和标签值 **CodeDeployDemo**。（如果您指定不同的标签密钥或标签值，则[步骤 4：部署 WordPress 应用程序 \(p. 58\)](#)中的说明可能会产生意外结果。）

在您按照说明启动 Amazon EC2 实例之后，请返回到此页，并继续下一部分。请勿继续[使用 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)作为下一步骤。

连接到 Amazon Linux 或 RHEL Amazon EC2 实例

启动新的 Amazon EC2 实例之后，请按照下面的说明执行连接操作。

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

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 实例密钥对名为 **codedeploydemo.pem**，您可键入：

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

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

Note

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

```
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. 在您设置 Amazon EC2 实例时，通过键入以下内容确认正确安装了 AWS CodeDeploy 代理：

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

有关确定 AWS CodeDeploy 代理状态的更多信息，请参阅[验证 AWS CodeDeploy 代理是否正在运行](#) (p. 119)

如果未安装 AWS CodeDeploy 代理，请按照[安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理](#) (p. 121) 中的说明操作。

4. 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，AWS CodeDeploy 将不能部署到该实例。

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

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

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon EC2 控制台 <https://console.aws.amazon.com/ec2/>。
2. 选择 Instances 并选择您的实例。
3. 在 Security groups 下，选择 view inbound rules。

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	
22	tcp	0.0.0.0/0	launch-wizard- N #

4. 在安全组中，选择您的 Amazon EC2 实例的安全组。它的名称可能为 **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。如果您看到规则具有以下值，则正确配置了实例的安全组：

- Type : HTTP
 - Protocol : TCP
 - Port Range : 80
 - Source : Custom
5. 如果您没有看到包含上一步中列出的值的规则，请使用 [向安全组添加规则](#) 中的过程来将其添加到新安全规则。

步骤 2：配置要部署到 Amazon Linux 或 Red Hat Enterprise Linux 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实例的内容了。

主题

- [获取源代码](#) (p. 52)
- [创建脚本以运行应用程序](#) (p. 53)
- [添加应用程序规范文件](#) (p. 54)

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 \(内置命令行调用 \) \(p. 52\)](#)
- [获取 WordPress 源代码的副本 \(Git\) \(p. 52\)](#)

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 命令执行以下操作：

- 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。AWS 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 /tmp/WordPress/scripts 中创建一个 install_dependencies.sh 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 yum install -y httpd24 php70 mysql56-server php70-mysqlnd
```

3. 在 /tmp/WordPress/scripts 中创建一个 start_server.sh 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
service httpd start
service mysqld start
```

4. 在 /tmp/WordPress/scripts 中创建一个 stop_server.sh 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
    service httpd stop
fi
isExistApp=`pgrep mysqld`
if [[ -n $isExistApp ]]; then
    service mysqld stop
fi
```

5. 在 /tmp/WordPress/scripts 中创建一个 create_test_db.sh 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** 数据库供 WordPress 使用。

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE 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 [步骤 5：更新和重新部署您的 WordPress 应用程序 \(p. 62\)](#) 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 application specification file (AppSpec file) (AWS CodeDeploy 使用的 [YAML](#) 格式的文件) 以：

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

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

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

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

AWS CodeDeploy 使用此 AppSpec file 将开发计算机上的 `/tmp/WordPress` 文件夹中的所有文件复制到目标 Amazon EC2 实例上的 `/var/www/html/WordPress` 文件夹中。在部署过程中，AWS CodeDeploy 会在部署生命周期内的指定事件中作为目标 Amazon EC2 实例上的 `/var/www/html/WordPress/scripts` 文件夹中的 `root` 运行指定脚本，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)，AWS 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 [AWS CodeDeploy AppSpec File 参考 \(p. 275\)](#)。

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，AWS CodeDeploy 会引发可能难以调试的错误。有关更多信息，请参阅 [AppSpec File 间距 \(p. 295\)](#)。

步骤 3：将您的 WordPress 应用程序上传到 Amazon S3

现在，您将准备源内容并将其上传到一个位置（AWS 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 AWS CodeDeploy 将应用程序从 GitHub 存储库部署到实例。有关更多信息，请参阅 [将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)。

主题

- [预置 Amazon S3 存储桶 \(p. 55\)](#)
- [为存储桶准备应用程序的文件 \(p. 57\)](#)
- [将应用程序的文件打包到单个存档文件并推送此存档文件 \(p. 57\)](#)

预置 Amazon S3 存储桶

在 Amazon S3 中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。

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

Note

对于所有 AWS 账户，存储桶名称在 Amazon S3 中必须是唯一的。如果您无法使用 **codedeploydemobucket**，请尝试其他存储桶名称，例如 **codedeploydemobucket**，后跟破折号和您的姓名首字母或某个其他唯一标识符。之后，请确保将本教程中的任何 **codedeploydemobucket** 替换为您的存储桶名称。

您必须在启动目标 Amazon EC2 实例的同一个 AWS 区域中创建 Amazon S3 存储桶。例如，如果您在美国东部（弗吉尼亚北部）地区中创建存储桶，则必须在美国东部（弗吉尼亚北部）地区中启动您的目标 Amazon EC2 实例。

主题

- [创建 Amazon S3 存储桶 \(CLI\) \(p. 55\)](#)
- [创建 Amazon S3 存储桶（控制台）\(p. 55\)](#)
- [提供针对 Amazon S3 存储桶和您的 IAM 用户的权限 \(p. 56\)](#)

创建 Amazon S3 存储桶 (CLI)

调用 mb 命令可创建名为 **codedeploydemobucket** 的 Amazon S3 存储桶：

```
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 Create bucket。
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。

提供针对 Amazon S3 存储桶和您的 IAM 用户的权限

您必须拥有对 Amazon S3 存储桶执行上传操作的权限。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 (*) 可让 AWS 账户 111122223333 将文件上传到名为 codedeploydemobucket 的 Amazon S3 存储桶中的任何目录：

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

To view your AWS account ID, see [Finding Your AWS Account ID](#).

Now is a good time to verify the Amazon S3 bucket will allow download requests from each participating Amazon EC2 instance. You can specify this through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download files from any directory in the Amazon S3 bucket named `codedeploydemobucket`:

```
{
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3::codedeploydemobucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE: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。以下自定义 IAM 用户策略允许您的 IAM 用户将修订上传到名为 `codedeploydemobucket` 的 Amazon S3 存储桶中的任意位置：


```
{
  "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

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

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

将应用程序的文件打包到单个存档文件并推送此存档文件

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

Note

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

1. 在开发计算机上，切换到这些文件存储到的文件夹：

```
cd /tmp/WordPress
```

Note

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

2. 调用 create-application 命令来注册名为 **WordPress_App** 的新应用程序：

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

3. 调用 AWS CodeDeploy [push](#) 命令可将文件打包在一起，将修订上传到 Amazon S3，并向 AWS CodeDeploy 注册有关已上传修订的信息，所有这些步骤通过一个操作完成。

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

此命令将当前目录的文件（不包括任何隐藏文件）打包到名为 **WordPressApp.zip** 的单个存档文件中，将修订上传到 **codedeploydemobucket** 存储桶，然后将有关上传修订的信息注册到 AWS CodeDeploy 中。

步骤 4：部署 WordPress 应用程序

现在，您部署已上传到 Amazon S3 的示例 WordPress 应用程序修订。您可以使用 AWS CLI 或 AWS 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 AWS CodeDeploy 部署您的应用程序修订 \(p. 58\)](#)
- [监控您的部署并排除故障 \(p. 60\)](#)
- [验证您的部署 \(p. 61\)](#)

使用 AWS CodeDeploy 部署您的应用程序修订

主题

- [部署您的应用程序修订 \(CLI\) \(p. 58\)](#)
- [部署应用程序修订（控制台）\(p. 59\)](#)

部署您的应用程序修订 (CLI)

1. 部署需要部署组。不过，在创建部署组之前，您需要服务角色 ARN。服务角色是 IAM 角色，该角色授予某个服务代表您执行操作的权限。在这种情况下，服务角色将提供 AWS 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 \(CLI\) \(p. 19\)](#)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 \(CLI\) \(p. 21\)](#)。

2. 现在您已获得服务角色 ARN，可以调用 `create-deployment-group` 命令以创建一个名为 **WordPress_DepGroup** 的部署组，它与一个名为 **WordPress_App** 的应用程序关联，并且使用名为 **CodeDeployDemo** 的 Amazon EC2 标签和名为 **CodeDeployDefault.OneAtATime** 的部署配置：

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

Note

`create-deployment-group` 命令支持创建触发器，这些触发器导致向主题订阅者发送有关部署和实例中的指定事件的 Amazon SNS 通知。此命令还支持以下选项：自动回滚部署和设置警报以

便在满足 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中未包含适用于这些操作的命令。

3. 现在调用 `create-deployment` 命令以创建一个与名为 **WordPress_App** 的应用程序、名为 **CodeDeployDefault.OneAtATime** 的部署配置和名为 **WordPress_DepGroup** 的部署组关联的部署，它使用名为 **codedeploydemobucket** 的存储桶中的名为 **WordPressApp.zip** 的应用程序修订：

```
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
```

部署应用程序修订 (控制台)

1. 在使用 AWS 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，服务角色将提供 AWS 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. 18)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. 21)。

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

Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

3. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
4. 在应用程序列表中，选择 **WordPress_App**。
5. 在 Deployment groups 下，选择 Create deployment group。
6. 在 Deployment group name 中，键入 **WordPress_DepGroup**。
7. 在 Deployment type 下，选择 In-place deployment。
8. 在 Environment configuration 中，选择 Amazon EC2 instances 选项卡。
9. 在 Key 框中，键入 **Name**。
10. 在 Value 框中，键入 **CodeDeployDemo**。

Note

键入 **CodeDeployDemo** 后，Instances 下方应显示 1 来确认 AWS CodeDeploy 已找到一个匹配的 Amazon EC2 实例。

11. 在 Deployment configuration 下拉列表中，选择 **CodeDeployDefault.OneAtATime**。
12. 在 Service role ARN 下拉列表中，选择服务角色 ARN，然后选择 Create deployment group。
13. 在 Application details 页上，选择新部署组旁边的按钮。从 Actions 菜单中，选择 Deploy new revision。
14. 在 Application 下拉列表中，选择 **WordPress_App**。
15. 在 Deployment group 下拉列表中，选择 **WordPress_DepGroup**。
16. 在 Repository type 的旁边，选择 My application is stored in Amazon S3。在 Revision location 中，键入您之前上传到 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 `codedeploydemobucket` (或将应用程序修订上传到的存储桶的名称)。
- c. 在对象列表中，选择 `WordPressApp.zip`。
- d. 在概述选项卡中，将链接字段的值复制到您的剪贴板。

其内容如下所示：

`https://s3.amazonaws.com/codedeploydemobucket/WordPressApp.zip`

- e. 返回 AWS CodeDeploy 控制台，在 Revision location 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. 从 Deployment configuration 下拉列表中，选择 `CodeDeployDefault.OneAtATime`。
20. 选择 Deploy。有关您新创建的部署的信息将显示在 Deployments 页上。

Note

要获取部署的当前状态，请选择表旁边的 Refresh 按钮。

监控您的部署并排除故障

主题

- [监视您的部署并排除故障 \(CLI\)](#) (p. 60)
- [监视您的部署和故障排除 \(控制台\)](#) (p. 60)

监视您的部署并排除故障 (CLI)

1. 通过对名为 **WordPress_App** 的应用程序和名为 **WordPress_DepGroup** 的部署组调用 `list-deployments` 命令来获取部署的 ID：

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

2. 使用部署 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 `list-deployment-instances` 和 `get-deployment-instance` 等命令来排除故障。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. 319)。

监视您的部署和故障排除 (控制台)

在 AWS CodeDeploy 控制台的 Deployments 页上，可在 Status 列中监视部署的状态。

Note

要获取部署的当前状态，请选择表上方的 Refresh 按钮。

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 Deployments 表中，选择部署 ID 旁的箭头。部署失败后，Details 中将显示描述失败原因的消息。

2. 在 Instances 中，选择 View all instances。此时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。

Note

如果您未看到 Instances，请选择表上方的 Refresh 按钮。在 Status 列从 In progress 更改为 Created 之后，应显示 Instances。

3. 如果您要执行更多问题排查，则可使用与 [View Instance Details \(p. 179\)](#) 中描述方法类似的方法。您还可以分析有关 Amazon EC2 实例的部署日志文件。有关更多信息，请参阅 [分析日志文件以调查针对实例的部署失败 \(p. 319\)](#)。

验证您的部署

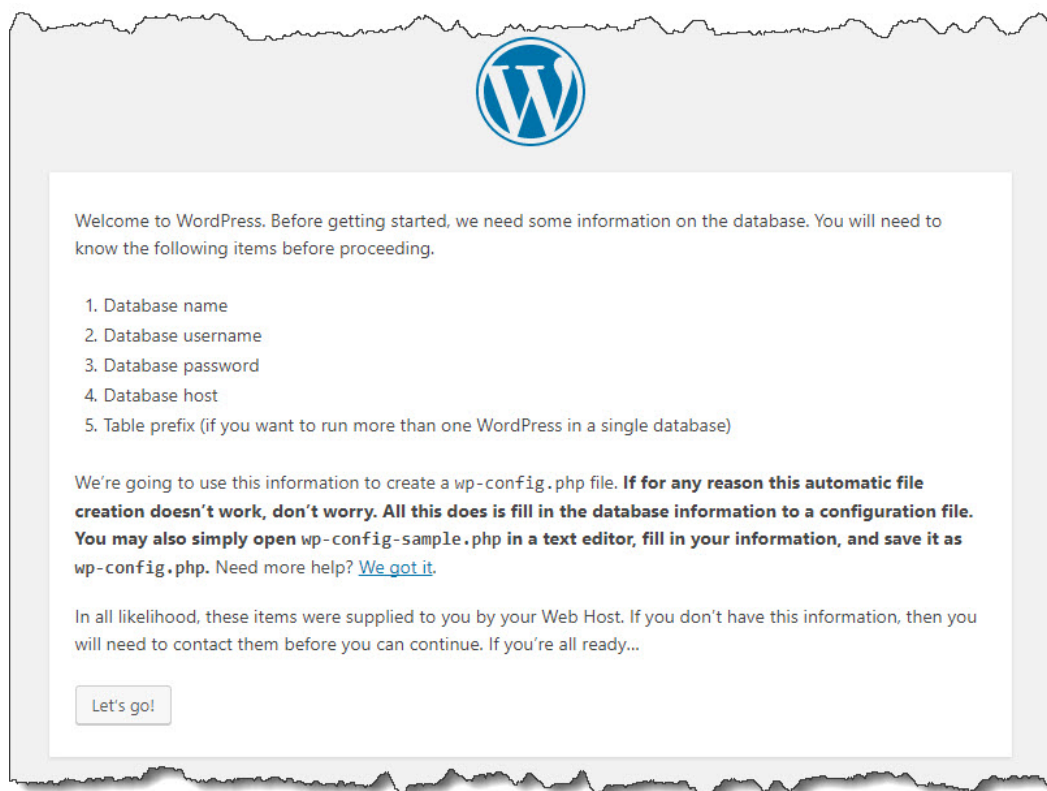
部署成功后，请验证您的 WordPress 安装是否正常工作。使用 Amazon EC2 实例的公有 DNS 地址（后跟 / WordPress）以在 Web 浏览器中查看您的站点。（要获取公有 DNS 值，请在 Amazon EC2 控制台中选择 Amazon EC2 实例，然后在 Description 选项卡上查找 Public DNS 的值。）

例如，如果 Amazon EC2 实例的公共 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 [添加入站规则，允许 HTTP 流量指向您的 Amazon Linux 或 RHEL Amazon EC2 实例 \(p. 51\)](#)。

步骤 5：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 AWS CodeDeploy 重新部署站点。之后，您应该可以看到 Amazon EC2 实例上的代码更改。

主题

- [设置 WordPress 站点 \(p. 62\)](#)
- [修改站点 \(p. 62\)](#)
- [重新部署站点 \(p. 63\)](#)

设置 WordPress 站点

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

1. 将您站点的 URL 键入到 Web 浏览器中。该 URL 是 Amazon EC2 实例的公共 DNS 地址加上 / WordPress 扩展。对于此示例 WordPress 站点（以及示例 Amazon EC2 实例公共 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：
 - Database Name : **test**
 - User Name : **root**
 - Password : 留空。
 - Database Host : **localhost**
 - Table Prefix : **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** 脚本会更新 /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，则可使用 Amazon S3 和 AWS CodeDeploy 重新部署站点。

将更改打包并上传到 Amazon S3，如[将应用程序的文件打包到单个存档文件并推送此存档文件 \(p. 57\)](#)中所述。(在按照这些说明操作时，请记住您不需要创建应用程序。)为新修订提供与之前一样的密钥 (**WordPressApp.zip**)。将其上传到之前创建的同一个 Amazon S3 存储桶（例如，**codedeploydemobucket**）。

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

主题

- [重新部署站点 \(CLI\) \(p. 63\)](#)
- [重新部署站点 \(控制台\) \(p. 63\)](#)

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[监控您的部署并排除故障 \(p. 60\)](#)中所述。

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

重新部署站点 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。
3. 选择 Create deployment。
4. 在 Create deployment 页面上：
 - a. 在 Application 列表中，选择 WordPress_App。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

- b. 在 Deployment group 列表中，选择 WordPress_DepGroup。
- c. 在 Repository type 区域中，选择 My application is stored in Amazon S3，然后将修订的 Amazon S3 链接复制到 Revision location 框中。要查找链接值，请执行以下操作：
 - i. 在单独的浏览器选项卡中：

登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

浏览并打开 codedeploydemobucket，然后选择您的修订 **WordPressApp.zip**。
 - ii. 如果 Properties 窗格在 Amazon S3 控制台中不可见，则选择 Properties 按钮。
 - iii. 在 Properties 窗格中，将 Link 字段的值复制到 AWS CodeDeploy 控制台中的 Revision location 框中。
- d. 如果显示消息说明无法检测文件类型，请选择 .zip。
- e. 将 Deployment description 框留空。
- f. 在 Deployment configuration 列表中，选择 CodeDeployDefault.OneAtATime，然后选择 Deploy。

要更新部署状态，请选择表上方的 Refresh 按钮。

您可以检查部署的状态，如 [监控您的部署并排除故障 \(p. 60\)](#) 中所述。

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

步骤 6：清除 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 AWS CloudFormation 堆栈（或终止任何 Amazon EC2 实例，条件是这些实例是在 AWS CloudFormation 外部创建的）。
- 任何 Amazon S3 存储桶。
- AWS CodeDeploy 中的 WordPress_App 应用程序。

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

主题

- [清除资源 \(CLI\) \(p. 64\)](#)
- [清除资源 \(控制台\) \(p. 65\)](#)
- [接下来做什么？ \(p. 66\)](#)

清除资源 (CLI)

1. 如果您为此教程使用了 AWS 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，请对名为 **codedeploydemobucket** 存储桶调用具有 `--recursive` 开关的 `rm` 命令。这将删除存储桶以及该存储桶中的所有对象：

```
aws s3 rm s3://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
```

如果您未为此教程使用 AWS 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 AWS CloudFormation 模板，请删除关联的 AWS CloudFormation 堆栈。

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台：<https://console.aws.amazon.com/cloudformation>。
2. 在 Filter 框中，键入之前创建的 AWS 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。

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

1. 登录 AWS 管理控制台并通过以下网址打开 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。

删除 Amazon S3 存储桶：

1. 登录 AWS 管理控制台并通过以下网址打开 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，例如 **WordPressApp.zip**。在 Actions 菜单中，选择 Delete。在提示确认删除时，选择 OK。
4. 在清空存储桶后，可以删除存储桶。在存储桶列表中，选择存储桶的行（而不是存储桶名称）。选择 Delete bucket，当系统提示进行确认时，选择 OK。

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

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。
3. 在应用程序列表中，选择 WordPress_App。
4. 在 Application details 页上的 Deployment groups 中，选择部署组旁边的按钮。在 Actions 菜单上，选择 Delete。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。
5. 在 Application details 页的底部，选择 Delete application。
6. 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

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 AWS CodeDeploy 部署，然后已更新您站点的代码并且已重新部署站点。

教程：使用 部署“Hello, World!”应用程序 AWS CodeDeploy (Windows Server)

在本教程中，您将一个网页部署到一个正在将 Internet Information Services (IIS) 作为其 Web 服务器运行的 Windows Server Amazon EC2 实例。此网页将简单地显示“Hello, World!”message。

不是您要找的内容？

- 要改而部署到 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 \(Amazon Linux 或 Red Hat Enterprise Linux 和 Linux, macOS, or Unix \) \(p. 49\)](#)。
- 要改而部署到本地实例，请参阅[教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \) \(p. 79\)](#)。

本教程建立在[步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 \(p. 25\)](#)中介绍的概念的基础之上。如果您尚未完成该教程，则可能需要先学习它。

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, or Unix 的本地计算机上完成其中的大部分步骤，但您需要适应涵盖基于 Windows 的目录路径（例如 c:\temp）的步骤。另外，如果您要连接到 Amazon EC2 实例，您将需要一个客户端应用程序，该应用程序能够通过远程桌面协议 (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。）

在开始本教程之前，您必须完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的先决条件，包括配置您的 IAM 用户、安装或升级 AWS CLI，以及创建 IAM 实例配置文件和服务角色。

主题

- [步骤 1：启动 Windows Server Amazon EC2 实例 \(p. 67\)](#)
- [步骤 2：将您的源内容配置为部署到 Windows Server Amazon EC2 实例 \(p. 67\)](#)
- [步骤 3：将您的“Hello, World!”应用程序上传到 Amazon S3 \(p. 70\)](#)
- [步骤 4：部署您的“Hello, World!”应用程序 \(p. 72\)](#)

- 步骤 5：更新和重新部署“Hello, World!”应用程序 (p. 76)
- 步骤 6：清除“Hello, World!”应用程序和相关资源 (p. 78)

步骤 1：启动 Windows Server Amazon EC2 实例

要使用 AWS 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[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)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，请确保指定标记键##和标签值 **CodeDeployDemo**。(如果您指定不同的标记键或标签值，则[步骤 4：部署您的“Hello, World!”应用程序 \(p. 72\)](#)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 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)作为下一步骤。

连接到您的 Amazon EC2 实例

启动 Amazon EC2 实例之后，请按照说明来练习连接操作。

Note

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

1. 登录 AWS 管理控制台并通过以下网址打开 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。
5. 选择 Get Password。
6. 选择 Browse。浏览并选择与 Windows Server Amazon EC2 实例关联的 Amazon EC2 实例密钥对，然后选择 Open。
7. 选择 Decrypt Password。记录显示的密码。您在步骤 10 中需要它。
8. 选择 Download Remote Desktop File，然后打开文件。
9. 如果系统提示您连接（即使无法确定远程连接的发布程序），请继续。
10. 键入您在步骤 7 中记录的密码，然后继续。（如果 RDP 连接客户端应用程序提示您输入用户名，请键入 **Administrator**。）
11. 如果系统提示您连接（即使无法验证远程计算机的身份），请继续。
12. 在连接之后，将显示运行 Windows Server 的 Amazon EC2 实例的桌面。
13. 您现在可以从运行的 Amazon EC2 实例注销。

Warning

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

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

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

主题

- [创建网页 \(p. 68\)](#)
- [创建运行应用程序的脚本 \(p. 68\)](#)
- [添加应用程序规范文件 \(p. 69\)](#)

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 AWS
CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/codedeploy">AWS CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

创建运行应用程序的脚本

接下来，您将创建一个脚本，AWS CodeDeploy 将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 application specification file (AppSpec file)。AppSpec file 是 [YAML](#) 格式的文件，AWS CodeDeploy 使用该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实例上的目的地。
- 指定在部署期间要在实例上运行的脚本。

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

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

```
notepad appspec.yml
```

2. 将以下 YAML 代码添加到该文件中，然后保存该文件。

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

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

有关这些设置的更多信息，请参阅 [AWS CodeDeploy AppSpec File 参考 \(p. 275\)](#)。

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，AWS CodeDeploy 将引发可能难以调试的错误。有关更多信息，请参阅 [AppSpec File 间距 \(p. 295\)](#)。

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

现在，您将准备源内容并将其上传到一个位置（AWS 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 AWS CodeDeploy 将应用程序从 GitHub 存储库部署到实例。有关更多信息，请参阅 [将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)。

主题

- [预置 Amazon S3 存储桶 \(p. 70\)](#)
- [为存储桶准备应用程序的文件 \(p. 72\)](#)
- [将应用程序的文件打包到单个存档文件并推送此存档文件 \(p. 72\)](#)

预置 Amazon S3 存储桶

在 Amazon S3 中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。

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

Note

对于所有 AWS 账户，存储桶名称在 Amazon S3 中必须是唯一的。如果您无法使用 **codedeploydemobucket**，请尝试其他存储桶名称，例如 **codedeploydemobucket**，后跟破折号和您的姓名首字母或某个其他唯一标识符。之后，请确保将本教程中的任何 **codedeploydemobucket** 替换为您的存储桶名称。

必须在启动目标 Amazon EC2 实例的同一 AWS 区域中创建 Amazon S3 存储桶。例如，如果您在美国东部（弗吉尼亚北部）地区中创建存储桶，则必须在美国东部（弗吉尼亚北部）地区中启动您的目标 Amazon EC2 实例。

主题

- [创建 Amazon S3 存储桶 \(CLI\) \(p. 70\)](#)
- [创建 Amazon S3 存储桶（控制台）\(p. 70\)](#)
- [提供针对 Amazon S3 存储桶和您的 IAM 用户的权限 \(p. 71\)](#)

创建 Amazon S3 存储桶 (CLI)

调用 mb 命令可创建名为 **codedeploydemobucket** 的 Amazon S3 存储桶：

```
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 Create bucket。
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。

提供针对 Amazon S3 存储桶和您的 IAM 用户的权限

您必须拥有对 Amazon S3 存储桶执行上传操作的权限。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 (*) 可让 AWS 账户 111122223333 将文件上传到名为 codedeploydemobucket 的 Amazon S3 存储桶中的任何目录：

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

To view your AWS account ID, see [Finding Your AWS Account ID](#).

Now is a good time to verify the Amazon S3 bucket will allow download requests from each participating Amazon EC2 instance. You can specify this through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download files from any directory in the Amazon S3 bucket named `codedeploydemobucket`:

```
{
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::codedeploydemobucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE: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 codedeploydemobucket 的 Amazon S3 存储桶中的任意位置：

```
{
  "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 file和脚本：

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

将应用程序的文件打包到单个存档文件并推送此存档文件

将这些文件打包到一个存档文件（称为应用程序修订）。

Note

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

1. 在开发计算机上，切换到这些文件存储到的文件夹：

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

Note

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

2. 调用 `create-application` 命令可向 AWS CodeDeploy 注册名为 **HelloWorld_App** 的新应用程序：

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

3. 调用 AWS CodeDeploy [push](#) 命令可将文件打包在一起，将修订上传到 Amazon S3，并向 AWS CodeDeploy 注册有关已上传修订的信息，所有这些步骤通过一个操作完成。

```
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）打包到一个名为 `HelloWorld_App.zip` 的存档文件中，将修订上传到 **codedeploydemobucket** 存储桶，然后向 AWS CodeDeploy 注册有关已上传修订的信息。

步骤 4：部署您的“Hello，World!”应用程序

现在您将部署上传到 Amazon S3 的示例“Hello，World!”应用程序修订。您将使用 AWS CLI 或 AWS 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，您将检查结果。

主题

- [使用 AWS CodeDeploy 部署您的应用程序修订 \(p. 73\)](#)
- [监控您的部署并排除故障 \(p. 74\)](#)
- [验证您的部署 \(p. 75\)](#)

使用 AWS CodeDeploy 部署您的应用程序修订

主题

- [部署您的应用程序修订 \(CLI\) \(p. 73\)](#)
- [部署应用程序修订 \(控制台\) \(p. 73\)](#)

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，服务角色将提供 AWS 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 \(CLI\) \(p. 19\)](#)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 \(CLI\) \(p. 21\)](#)。

2. 您已拥有 ARN，现在可调用 `create-deployment-group` 命令来创建名为 **HelloWorld_DepGroup** 的部署组；这些部署组关联到名为 **HelloWorld_App** 的应用程序，使用名为 **CodeDeployDemo** 的 Amazon EC2 实例标签以及名为 **CodeDeployDefault.OneAtATime** 的部署配置，其服务角色 ARN 为：

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

Note

`create-deployment-group` 命令支持创建触发器，这些触发器导致向主题订阅者发送有关部署和实例中的指定事件的 Amazon SNS 通知。此命令还支持以下选项：自动回滚部署和设置警报以便在满足 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中未包含适用于这些操作的命令。

3. 现在调用 `create-deployment` 命令来创建部署，该部署与名为 **HelloWorld_App** 的应用程序、名为 **CodeDeployDefault.OneAtATime** 的部署配置以及名为 **HelloWorld_DepGroup** 的部署组关联，使用名为 **codedeploydemobucket** 的存储桶中名为 **HelloWorld_App.zip** 的应用程序修订：

```
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 AWS 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，服务角色将提供 AWS 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 \(控制台\) \(p. 18\)](#)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 \(控制台\) \(p. 21\)](#)。

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

Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

3. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
4. 在应用程序列表中，选择 HelloWorld_App。
5. 在 Deployment groups 下，选择 Create deployment group。
6. 在 Deployment group name 中，键入 **HelloWorld_DepGroup**。
7. 在 Environment configuration 中，选择 Amazon EC2 instances 选项卡。
8. 在 Key 框中，键入 **Name**。
9. 在 Value 中，键入 **CodeDeployDemo**。

Note

在键入 **CodeDeployDemo** 后，Matching instances 下方应显示 1 来确认 AWS CodeDeploy 已找到一个匹配的 Amazon EC2 实例。

10. 在 Deployment configuration 下拉列表中，选择 CodeDeployDefault.OneAtATime。
11. 在 Service role ARN 下拉列表中，选择服务角色 ARN，然后选择 Create deployment group。
12. 在 AWS CodeDeploy 菜单上，选择 Deployments。
13. 选择 Create deployment。
14. 在 Application 下拉列表中，选择 HelloWorld_App。
15. 在 Deployment group 下拉列表中，选择 HelloWorld_DepGroup。
16. 在 Repository type 中，选择 My application is stored in Amazon S3，然后在 Revision location 中键入已上传到 Amazon S3 的示例“Hello, World!”应用程序修订的位置。获取位置：
 1. 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，选择 codedeploydemobucket (或您将应用程序修订上传到的存储桶的名称)。
 3. 在对象列表中，选择 HelloWorld_App.zip。
 4. 如果未显示 Properties 窗格，请选择 Properties 按钮。
 5. 在 Properties 窗格中，将 Link 字段的值复制到您的剪贴板。

其内容如下所示：

https://s3.amazonaws.com/codedeploydemobucket/HelloWorld_App.zip

6. 返回 AWS CodeDeploy 控制台，在 Revision Location 中，粘贴 Link 字段值。
17. 如果 File type 列表中显示的消息说明无法检测文件类型，则在文件类型列表中选择 .zip。
18. (可选) 在 Deployment description 中键入备注。
19. 从 Deployment configuration 下拉列表中，选择 CodeDeployDefault.OneAtATime。
20. 选择 Deploy。有关您新创建的部署的信息将显示在 Deployments 页上。

Note

要更新部署的当前状态，请选择表旁边的 Refresh 按钮。

监控您的部署并排除故障

- [监视您的部署并排除故障 \(CLI\) \(p. 75\)](#)
- [监视您的部署和故障排除 \(控制台\) \(p. 75\)](#)

监视您的部署并排除故障 (CLI)

1. 通过对名为 **HelloWorld_App** 的应用程序和名为 **HelloWorld_DepGroup** 的部署组调用 `list-deployments` 命令，来获取部署 ID：

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

2. 使用部署 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 `list-deployment-instances` 和 `get-deployment-instance` 等命令来排除故障。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 \(p. 319\)](#)。

监视您的部署和故障排除 (控制台)

在 AWS CodeDeploy 控制台的 Deployments 页上，可在 Status 列中监视部署的状态。

Note

要更新部署的当前状态，请选择 Refresh 按钮。

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 Deployments 表中，选择部署 ID 旁的箭头。部署失败后，Details 中将显示描述失败原因的消息。
2. 在 Instances 中，选择 View all instances。此时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。

Note

如果您未看到 Instances，请选择表上方的 Refresh 按钮。在 Status 列从 `In progress` 更改为 `Created` 之后，应显示 Instances。

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 \(p. 179\)](#) 的技巧。您还可以分析有关 Amazon EC2 实例的部署日志文件。有关更多信息，请参阅[分析日志文件以调查针对实例的部署失败 \(p. 319\)](#)。

验证您的部署

部署成功后，请验证您的安装是否正常工作。使用 Amazon EC2 实例的公共 DNS 地址，在 Web 浏览器中查看网页。（要获取公共 DNS 值，请在 Amazon EC2 控制台中选择 Amazon EC2 实例，然后在 Description 选项卡上的 Public DNS 中查看值。）

例如，如果 Amazon EC2 实例的公共 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
```

如果成功，您应该看到“Hello, World!”网页。

步骤 5：更新和重新部署“Hello, World!”应用程序

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 AWS CodeDeploy 来重新部署站点。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 \(p. 76\)](#)
- [重新部署站点 \(p. 76\)](#)

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. 修改 `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 AWS CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
    codedeploy">AWS 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 AWS CodeDeploy 重新部署网页。

将更改打包并上传到 Amazon S3，如[将应用程序的文件打包到单个存档文件并推送此存档文件 \(p. 72\)](#)中所述。（在按照这些说明操作时，您不需要创建新的应用程序。）为修订指定与之前相同的密钥 (**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 AWS CLI 或 AWS CodeDeploy 控制台重新部署站点。

主题

- [重新部署站点 \(CLI\) \(p. 77\)](#)
- [重新部署站点 \(控制台\) \(p. 77\)](#)

重新部署站点 (CLI)

调用 `create-deployment` 命令以基于上传的修订创建部署，再次在名为 `codedeploydemobucket` 的存储桶中使用名为 `HelloWorld_App` 的应用程序、名为 `CodeDeployDefault.OneAtATime` 的部署配置、名为 `HelloWorld_DepGroup` 的部署组和名为 `HelloWorld_App.zip` 的修订：

```
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[监控您的部署并排除故障 \(p. 74\)](#)中所述。

在 AWS 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. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。
3. 选择 Create deployment。
4. 在 Create deployment 页面上：
 1. 在 Application 列表中，选择 HelloWorld_App。
 2. 在 Deployment group 列表中，选择 HelloWorld_DepGroup。
 3. 在 Repository type 区域中，选择 My application is stored in Amazon S3，然后将您的修订的 Amazon S3 链接复制到 Revision location 框中。

要查找链接值，请执行以下操作：

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

浏览找到并打开 `codedeploydemobucket`，然后在 Amazon S3 控制台中选择您的修订 **HelloWorld_App.zip**。

2. 如果 Properties 窗格在 Amazon S3 控制台中不可见，则选择 Properties 按钮。
3. 在 Properties 窗格中，将 Link 字段的值复制到 AWS CodeDeploy 控制台中的 Revision location 框中。
4. 在 File type 列表中，如果显示一条消息，告诉您无法检测文件类型，请选择 .zip。
5. 将 Deployment description 框留空。
6. 在 Deployment configuration 列表中，选择 CodeDeployDefault.OneAtATime，然后选择 Deploy。

选择表上方的 Refresh 按钮以获取部署状态。

您可以检查部署的状态，如[监控您的部署并排除故障 \(p. 74\)](#)中所述。

在 AWS 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！

步骤 6：清除“Hello，World!”应用程序和相关资源

现在您已成功更新了“Hello，World!”代码并重新部署了站点。为避免对您创建用于完成本教程的资源继续收费，您应该删除任何 AWS CloudFormation 堆栈（或者终止您在 AWS CloudFormation 外部手动创建的任何 Amazon EC2 实例）。您还应该删除任何刚刚为此教程创建的 Amazon S3 存储桶，以及 AWS CodeDeploy 中的 HelloWorld_App 应用程序。

您可以使用 AWS CLI、AWS CloudFormation、Amazon S3、Amazon EC2 和 AWS CodeDeploy 控制台，或者 AWS API 来清除资源。

主题

- [使用清除资源 \(CLI\) \(p. 78\)](#)
- [清除资源（控制台）\(p. 78\)](#)
- [接下来做什么？\(p. 79\)](#)

使用清除资源 (CLI)

1. 如果您为此教程使用了 AWS 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，请对名为 **codedeploydemobucket** 存储桶调用具有 --recursive 开关的 rm 命令。这将删除存储桶以及该存储桶中的所有对象。

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

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

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

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

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

清除资源（控制台）

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

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台：<https://console.aws.amazon.com/cloudformation>。
2. 在搜索框中，键入 AWS 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 AWS CloudFormation 堆栈外部创建的 Amazon EC2 实例，请执行以下步骤：

1. 登录 AWS 管理控制台并通过以下网址打开 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。

删除 Amazon S3 存储桶：

1. 登录 AWS 管理控制台并通过以下网址打开 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。在存储桶中选择所有文件，例如 **HelloWorld_App.zip**。在 Actions 菜单中，选择 Delete。在提示确认删除时，选择 OK。
4. 在清空存储桶后，可以删除存储桶。在存储桶列表中，选择存储桶的行（而不是存储桶名称）。选择 Delete bucket，当系统提示进行确认时，选择 OK。

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

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。
3. 在 Application details 页上的 Deployment groups 中，选择部署组旁边的按钮。在 Actions 菜单上，选择 Delete。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。
4. 在 Application details 页的底部，选择 Delete application。
5. 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

接下来做什么？

此时，您已成功完成了使用 AWS CodeDeploy 执行的部署。恭喜您！

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

本教程引导您完成将示例应用程序修订部署到运行 Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux (RHEL) 的单个本地实例（即非 Amazon EC2 实例的物理设备）的过程，帮助您获得使用 AWS

CodeDeploy 的经验。有关本地实例以及如何将它们与 AWS CodeDeploy 一起使用的信息，请参阅[Working with On-Premises Instances \(p. 156\)](#)。

不是您要找的内容？

- 要练习如何部署到运行 Amazon Linux 或 RHEL 的 Amazon EC2 实例，请参阅 [教程：将 WordPress 部署到 Amazon EC2 实例 \(Amazon Linux 或 Red Hat Enterprise Linux 和 Linux, macOS, or Unix \) \(p. 49\)](#)。
- 要练习如何部署到运行 Windows Server 的 Amazon EC2 实例，请参阅 [教程：使用部署“Hello, World!”应用程序 AWS CodeDeploy \(Windows Server\) \(p. 66\)](#)。

本教程建立在在 [AWS CodeDeploy 中尝试示例就地部署 \(p. 31\)](#) 中介绍的概念的基础之上。如果您尚未完成就地部署的 Sample deployment wizard，可能需要先完成它。

主题

- [先决条件 \(p. 80\)](#)
- [步骤 1：配置本地实例 \(p. 80\)](#)
- [步骤 2：创建示例应用程序修订 \(p. 80\)](#)
- [步骤 3：打包并将您的应用程序修订上传到 Amazon S3 \(p. 84\)](#)
- [步骤 4：部署应用程序修订 \(p. 84\)](#)
- [步骤 5：验证您的部署 \(p. 84\)](#)
- [步骤 6：清除资源 \(p. 84\)](#)

先决条件

在开始此教程之前，您必须完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的先决条件，包括配置 IAM 用户、安装或升级 AWS CLI 以及创建服务角色。您无需按照先决条件中的所述创建 IAM 实例配置文件。本地实例不使用 IAM 实例配置文件。

您将配置作为本地实例的物理设备必须运行[AWS CodeDeploy 代理支持的操作系统 \(p. 113\)](#) 中列出的操作系统之一。

步骤 1：配置本地实例

您必须先配置本地实例，然后才能在其中进行部署。按照[Working with On-Premises Instances \(p. 156\)](#)中的说明操作，然后返回此页。

步骤 2：创建示例应用程序修订

在这一步中，您将创建要部署到本地实例的示例应用程序修订。

由于很难了解您的本地实例上已经安装了哪些软件和功能或者您的组织策略允许安装哪些软件和功能，因此，我们这里提供的示例应用程序修订仅仅使用批处理脚本（对于 Windows Server）或 shell 脚本（对于 Ubuntu Server 和 RHEL），将文本文件写入到您本地实例上的某个位置。对于多个 AWS CodeDeploy 部署生命周期事件，每个事件写入一个文件，包括 Install、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 AWS CodeDeploy 代理的用户账户无权执行脚本。
- 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 [教程：使用部署“Hello, World!”应用程序 AWS CodeDeploy \(Windows Server\)](#) (p. 66)教程的 [步骤 2：将您的源内容配置为部署到 Windows Server Amazon EC2 实例](#) (p. 67)中的示例。

如果您配置了 RHEL 实例并希望部署其他示例，则可能需要使用 [教程：将 WordPress 部署到 Amazon EC2 实例 \(Amazon Linux 或 Red Hat Enterprise Linux 和 Linux, macOS, or Unix\)](#) (p. 49)教程的 [步骤 2：配置要部署到 Amazon Linux 或 Red Hat Enterprise Linux Amazon EC2 实例的源内容](#) (p. 51)中的示例。

目前，Ubuntu Server 没有替代示例。

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

对于 Windows：

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

对于 Linux, macOS, or 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 (对于 Windows Server)：

```
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 Server 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 AWS CodeDeploy 的修订 \(p. 209\)](#)和[AWS CodeDeploy AppSpec File 参考 \(p. 275\)](#)。

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, or Unix：

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

4. 在 scripts 子文件夹的根中，使用文本编辑器为 Windows Server 创建四个分别名为 before-install.bat、after-install.bat、application-start.bat 和 validate-service.bat 的文件，或为 Ubuntu Server 和 RHEL 创建四个分别名为 before-install.sh、after-install.sh、application-start.sh 和 validate-service.sh 的文件：

对于 Windows Server：

before-install.bat：

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

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

mkdir %FOLDER%
```

after-install.bat：

```
cd %HOMEDRIVE%\temp\CodeDeployExample

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

application-start.bat：

```
cd %HOMEDRIVE%\temp\CodeDeployExample

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

validate-service.bat：

```
cd %HOMEDRIVE%\temp\CodeDeployExample

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

对于 Ubuntu Server 和 RHEL：

before-install.sh：

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

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

mkdir -p $FOLDER
```

after-install.sh：

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

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

application-start.sh：

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

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

validate-service.sh：

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

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

unset FOLDER
```

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

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

步骤 3：打包并将您的应用程序修订上传到 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 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)和[将 AWS CodeDeploy 的修订推送到 Amazon S3 \(p. 214\)](#)中的说明操作。（虽然您可以为应用程序和部署组提供任意名称，不过我们建议您使用 CodeDeploy-OnPrem-App 作为应用程序名称，并使用 CodeDeploy-OnPrem-DG 作为部署组名称。）在您完成这些指示的操作之后，返回本页。

Note

或者，您可以将文件包上传到 GitHub 存储库并从该位置部署。有关更多信息，请参阅[将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)。

步骤 4：部署应用程序修订

在您将应用程序修订上传到 Amazon S3 存储桶之后，尝试将其部署到本地实例。按照[使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)中的说明操作，然后返回此页。

步骤 5：验证您的部署

要验证部署已经成功，请按照[使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)中的说明操作，然后返回本页。

如果部署已成功，您将在 c:\temp\CodeDeployExample 文件夹（对于 Windows Server）或 /tmp/CodeDeployExample 文件夹（对于 Ubuntu Server 和 RHEL）中找到四个文本文件。

如果部署失败，请按照[View Instance Details \(p. 179\)](#)和[解决实例问题 \(p. 319\)](#)中的排除故障步骤操作。进行任何必要的修复，重新打包并上传应用程序修订，然后再次尝试部署。

步骤 6：清除资源

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

您可以使用 AWS CLI 或结合使用 AWS CodeDeploy 和 Amazon S3 控制台及 AWS CLI 来清除资源。

清除资源 (CLI)

删除 Amazon S3 存储桶

- 对存储桶（例如，codedeploydemobucket）调用 `rm` 命令以及 `--recursive` 开关。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 AWS CodeDeploy 中的应用程序和部署组记录

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

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

注销本地实例并删除 IAM 用户

- 对本地实例和区域调用 `deregister` 命令：

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

Note

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

卸载 AWS CodeDeploy 代理并从本地实例中删除配置文件

- 从本地实例调用 `uninstall` 命令：

```
aws deploy uninstall
```

现在您已完成清除此教程所用资源的全部步骤。

清除资源（控制台）

删除 Amazon S3 存储桶

- 登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
- 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。
- 选择 Actions，然后选择 Delete。
- 在提示删除存储桶时，选择 OK。

删除 AWS CodeDeploy 中的应用程序和部署组记录

- Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

- 如果未显示应用程序列表，请在 AWS CodeDeploy 菜单上选择 Applications。
- 选择要删除的应用程序的名称（例如，`CodeDeploy-OnPrem-App`）。
- 在 Application details 页的底部，选择 Delete application。
- 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

您不能使用 AWS CodeDeploy 控制台来注销本地实例或者卸载 AWS CodeDeploy 代理。按照[注销本地实例并删除 IAM 用户 \(p. 85\)](#)中的说明进行操作。

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

在本教程中，您将使用 AWS CodeDeploy 向 Auto Scaling 组部署应用程序修订。有关 Auto Scaling 与 AWS CodeDeploy 的集成的信息，请参阅 [将 AWS CodeDeploy 与 Auto Scaling 集成](#) (p. 38)。

主题

- [先决条件](#) (p. 86)
- [步骤 1：创建和配置 Auto Scaling 组](#) (p. 86)
- [步骤 2：将应用程序部署到 Auto Scaling 组](#) (p. 91)
- [步骤 3：检查结果](#) (p. 96)
- [步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数](#) (p. 97)
- [步骤 5：再次检查结果](#) (p. 98)
- [步骤 6：清除](#) (p. 100)

先决条件

对于本教程，我们假设您已经完成[AWS CodeDeploy 入门](#) (p. 16)中的所有步骤，包括安装和配置 AWS CLI 以及创建 IAM 实例配置文件 (**CodeDeployDemo-EC2-Instance-Profile**) 和服务角色 (**CodeDeployDemo**)。服务角色是一种特殊类型的 IAM 角色，用于向服务授予代表您执行操作的权限。

如果您要将应用程序修订部署到 Ubuntu Server Amazon EC2 实例的 Auto Scaling 组，可以在[教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux\)](#) (p. 79)教程的[步骤 2：创建示例应用程序修订](#) (p. 80)中创建和使用示例修订。否则，您需要创建和使用与 Ubuntu Server 实例和 AWS CodeDeploy 兼容的修订。我们还提供适用于 Amazon Linux、Windows Server 和 Red Hat Enterprise Linux (RHEL) Amazon EC2 实例的示例修订。要自己创建修订，请参阅[使用 AWS CodeDeploy 的应用程序修订](#) (p. 208)。

步骤 1：创建和配置 Auto Scaling 组

在此步骤中，您将创建一个包含单个 Amazon Linux、RHEL 或 Windows Server Amazon EC2 实例的 Auto Scaling 组。在后面的步骤中，您将指示 Auto Scaling 再添加一个 Amazon EC2 实例，并且 AWS CodeDeploy 将向其中部署您的修订。

主题

- [创建和配置 Auto Scaling 组 \(CLI\)](#) (p. 86)
- [创建和配置 Auto Scaling 组 \(控制台\)](#) (p. 89)

创建和配置 Auto Scaling 组 (CLI)

1. 调用 create-launch-configuration 命令以创建 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 实例密钥对的名称 (由占位符 **key-name** 表示) 才能访问 Amazon EC2 实例。最后，您需要有关安装最新版本的 AWS CodeDeploy 代理的说明。

AWS CodeDeployTo 获取适用于本教程的 AMI 的 ID：

1. 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，然后选择 Launch Instance。

3. 在 Choose an Amazon Machine Image 页的 Quick Start 选项卡上，记下 Amazon Linux AMI、Red Hat Enterprise Linux 7.1、Ubuntu Server 14.04 LTS 或 Microsoft Windows Server 2012 R2 旁的 AMI 的 ID。

Note

如果您拥有与 AWS CodeDeploy 兼容的 AMI 自定义版本，则在此选择它，而不用浏览 Quick Start 选项卡。有关将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling 的信息，请参阅[将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling \(p. 39\)](#)。

对于 Amazon EC2 实例密钥对，使用 Amazon EC2 实例密钥对的名称。

要在您的开发计算机上安装最新版本的 AWS CodeDeploy 代理，请创建一个包含以下内容的名为 instance-setup.sh (适用于 Amazon Linux、Ubuntu Server 或 RHEL Amazon EC2 实例) 或 instance-setup.txt (适用于 Windows Server Amazon EC2 实例) 的文件。

Note

如果您拥有与 AWS CodeDeploy 兼容的 AMI 自定义版本，则无需创建 instance-setup.sh 或 instance-setup.txt 文件。

在 Amazon Linux 和 RHEL Amazon EC2 实例上

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 **bucket-name** 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称 \(p. 302\)](#)。

在 Ubuntu Server Amazon EC2 实例上

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 **bucket-name** 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称 \(p. 302\)](#)。

在 Windows Server Amazon EC2 实例上

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath API:版本 2014-10-06 -ArgumentList c:\temp\codedeploy-agent.msi -WindowStyle Hidden
```

```
</powershell>
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 **bucket-name** 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

调用 create-launch-configuration 命令。

在本地 Linux, macOS, or Unix 计算机上：

Important

Be sure to include file:// before the file name. It is required in this command.

```
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 \
  --user-data file://path/to/instance-setup.sh
```

在本地 Windows 计算机上：

Important

Be sure to include file:// before the file name. It is required in this command.

```
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 --user-data file://path/
to/instance-setup.txt
```

Note

如果您拥有与 AWS CodeDeploy 兼容的 AMI 自定义版本，则在上述命令中省略 --user-data 选项。

这些命令根据指定的映像 ID 创建名为 **CodeDeployDemo-AS-Configuration** 的 Auto Scaling 启动配置，同时应用指定的 IAM 实例配置文件和 Amazon EC2 实例密钥对，并运行命令以安装最新版本的 AWS CodeDeploy 代理。此启动配置基于 t1.micro Amazon EC2 实例类型。

2. 调用 create-auto-scaling-group 命令以创建 Auto Scaling 组。您将需要 AWS General Reference 的[区域和终端节点](#)所列区域之一的某个可用区的名称 (由占位符 **availability-zone** 表示)。

Note

要查看区域中的可用区列表，请调用：

```
aws ec2 describe-availability-zones --region region-name
```

例如，要查看美国西部（俄勒冈）区域中的可用区列表，请调用：

```
aws ec2 describe-availability-zones --region us-west-2
```

有关区域名称标识符的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

AP-2014-10-08

在本地 Linux, macOS, or 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
```

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
```

这些命令基于名为 **CodeDeployDemo-AS-Configuration** 的 Auto Scaling 启动配置创建一个名为 **CodeDeployDemo-AS-Group** 的 Auto Scaling 组。此 Auto Scaling 组只有一个在指定可用区中创建的 Amazon EC2 实例。

3. 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。

创建和配置 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，确保选中 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。Auto Scaling 资源与您所指定的区域相关联，并且 AWS CodeDeploy 仅在选定区域中受支持。
3. 在导航栏中的 Auto Scaling 下，选择 Launch Configurations。
4. 选择 Create launch configuration。
5. 在 Choose AMI 页的 Quick Start 选项卡上，在 Amazon Linux AMI、Red Hat Enterprise Linux 7.2、Ubuntu Server 14.04 LTS 或 Microsoft Windows Server 2012 R2 Base 旁选择 Select。

Note

如果您拥有已安装 AWS CodeDeploy 代理的 AMI 自定义版本，则改为在此选择该版本。有关将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling 的信息，请参阅[将自定义 AMI 用于 AWS CodeDeploy 和 Auto Scaling \(p. 39\)](#)。

6. 在 Choose Instance Type 页上保留默认值，然后选择 Next: Configure details。
7. 在 Configure details 页上的 Name 中，键入 **CodeDeployDemo-AS-Configuration**。在 IAM role 中，选择您之前创建的 IAM 实例配置文件 (**CodeDeployDemo-EC2-Instance-Profile**)。

展开 Advanced Details，然后在 User data 中键入以下内容。

Note

如果您使用的是已安装 AWS CodeDeploy 代理的 AMI 自定义版本，请跳过此步骤。

对于 Amazon Linux 和 RHEL Amazon EC2 实例

```
#!/bin/bash  
yum -y update
```

```
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Ubuntu Server Amazon EC2 实例

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Windows Server Amazon EC2 实例

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle Hidden
</powershell>
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

保留其余默认值，然后选择 Skip to review。

8. 在 Review 页上选择 Create launch configuration。

Note

在生产环境中，建议您限制对 Amazon EC2 实例的访问。有关更多信息，请参阅[有关保护您的 EC2 实例的提示](#)。

9. 在 Select an existing key pair or create a new key pair 对话框中，选择 Choose an existing key pair。在 Select a key pair 下拉列表中，选择您在前面步骤中创建或使用的 Amazon EC2 实例密钥对。选择 I acknowledge that I have access to the selected private key file (*key-file-name*.pem), and that without this file, I won't be able to log into my instance，然后选择 Create launch configuration。
10. 选择 Create an Auto Scaling group using this launch configuration。
11. 在 Configure Auto Scaling group details 页上的 Group name 中，键入 **CodeDeployDemo-AS-Group**。在 Group size 中，保留默认值。在 Availability Zone(s) 框中，选择 AWS General Reference 的[区域和终端节点](#)所列区域之一中的一个可用区。保留其余默认值，然后选择 Next: Configure scaling policies。

Note

如果 Launch into EC2-Classic 不显示在 Network 列表中，而且您不能选择默认的 Virtual Private Cloud (VPC)，则请选择或创建 VPC 和子网。有关更多信息，请参阅[您的 VPC 和子网](#)。

12. 在 2. Configure scaling policies 页上，保留 Keep this group at its initial size 为选中状态，然后选择 Next: Configure Notifications。
13. 跳过配置通知的步骤，然后选择 Review。
14. 选择 Create Auto Scaling group，然后选择 Close。
15. 在导航栏中，在 Auto Scaling Groups 处于选中状态的情况下，选择 **CodeDeployDemo-AS-Group**，然后选择 Instances 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。

步骤 2：将应用程序部署到 Auto Scaling 组

在此步骤中，您将向 Auto Scaling 组中的单个 Amazon EC2 实例部署修订。

主题

- [创建部署 \(CLI\) \(p. 91\)](#)
- [创建部署 \(控制台\) \(p. 93\)](#)

创建部署 (CLI)

1. 调用 create-application 命令以创建一个名为 **SimpleDemoApp** 的应用程序：

```
aws deploy create-application --application-name SimpleDemoApp
```

2. 您应该已经按照[步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)中的说明创建了一个服务角色。该服务角色将向 AWS CodeDeploy 授予访问 Amazon EC2 实例以展开（读取）其标签的权限。您将需要服务角色 ARN。要获取服务角色 ARN，请按照[获取服务角色 ARN \(CLI\) \(p. 21\)](#)中的说明操作。
3. 现在您已拥有一个服务角色 ARN，请调用 create-deployment-group 命令，使用名为 **CodeDeployDemo-AS-Group** 的 Auto Scaling 组和名为 **CodeDeployDefault.OneAtATime** 的部署配置创建一个与名为 **SimpleDemoApp** 的应用程序相关联的名为 **SimpleDemoDG** 的部署组（具有指定的服务角色 ARN）。

Note

[create-deployment-group](#) 命令支持创建触发器，这些触发器导致向主题订阅者发送有关部署和实例中的指定事件的 Amazon SNS 通知。此命令还支持以下选项：自动回滚部署和设置警报以便在满足 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中未包含适用于这些操作的命令。

在本地 Linux, macOS, or Unix 计算机上：

```
aws deploy create-deployment-group \  
  --application-name SimpleDemoApp \  
  --auto-scaling-groups CodeDeployDemo-AS-Group \  
  --deployment-group-name SimpleDemoDG \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --service-role-arn service-role-arn
```

在本地 Windows 计算机上：


```
aws deploy create-deployment-group --application-name SimpleDemoApp --auto-scaling-groups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG --deployment-config-name CodeDeployDefault.OneAtATime --service-role-arn service-role-arn
```

4. 调用 `create-deployment` 命令，使用指定位置的修订创建一个与名为 **SimpleDemoApp** 的应用程序、名为 **CodeDeployDefault.OneAtATime** 的部署配置、名为 **SimpleDemoDG** 的部署组相关联的部署。

对于 Amazon Linux 和 RHEL Amazon EC2 实例 (从本地 Linux, macOS, or Unix 计算机调用)

```
aws deploy create-deployment \  
  --application-name SimpleDemoApp \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name SimpleDemoDG \  
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/  
SampleApp_Linux.zip
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Amazon Linux 和 RHEL Amazon EC2 实例 (从本地 Windows 计算机调用)

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Linux.zip
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Windows Server Amazon EC2 实例 (从本地 Linux, macOS, or Unix 计算机调用)

```
aws deploy create-deployment \  
  --application-name SimpleDemoApp \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name SimpleDemoDG \  
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/  
SampleApp_Windows.zip
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Windows Server Amazon EC2 实例 (从本地 Windows 计算机调用)

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Windows.zip
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

Note

目前，AWS CodeDeploy 不提供要部署到 Ubuntu Server Amazon EC2 实例的示例修订。要自己创建修订，请参阅[使用 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

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。如果您需要重新获取部署 ID，请对名为 **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，请使用部署 ID 调用 get-deployment 命令：

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

请在返回的值为 Succeeded 之后继续。

创建部署（控制台）

1. 您应该已经按照[步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)中的说明创建了一个服务角色。该服务角色将向 AWS CodeDeploy 授予访问您的实例以展开（读取）其标签的权限。在使用 AWS CodeDeploy 控制台部署应用程序修订之前，您需要服务角色 ARN。要获取服务角色 ARN，请按照[获取服务角色 ARN（控制台） \(p. 21\)](#)中的说明操作。
2. 现在您已拥有服务角色 ARN，可以开始使用 AWS CodeDeploy 控制台部署您的应用程序修订。

Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

3. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
4. 选择 Create application。
5. 在 Application name 框中，键入 **SimpleDemoApp**。
6. 在 Deployment group name 框中，键入 **SimpleDemoDG**。
7. 在 Environment configuration 中的 Auto Scaling groups 选项卡上，键入 **CodeDeployDemo-AS-Group**。
8. 从 Deployment configuration 下拉列表中，选择 CodeDeployDefault.OneAtATime。
9. 从 Service role ARN 下拉列表中，选择服务角色 ARN。
10. 选择 Create application。
11. 在 Application details 页上的 Deployment groups 区域中，选择 SimpleDemoDG 旁的箭头以查看部署组详细信息。
12. 选择 SimpleDemoDG 旁的按钮。在 Actions 菜单中，选择 Deploy new revision。
13. 在 Repository type 区域中，选择 My application is stored in Amazon S3，然后在 Revision location 框中，键入您的操作系统和区域的示例应用程序的位置。

对于 Amazon Linux 和 RHEL Amazon EC2 实例

区域	示例应用程序的位置
美国东部（俄亥俄）区域	http://s3-us-east-2.amazonaws.com/ aws-codedeploy-us-east-2/samples/ latest/SampleApp_Linux.zip
美国东部（弗吉尼亚北部）地区	http://s3.amazonaws.com/aws- codedeploy-us-east-1/samples/latest/ SampleApp_Linux.zip
美国西部（加利福尼亚北部）区域	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us-west-1/samples/ latest/SampleApp_Linux.zip
美国西部（俄勒冈）区域	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us-west-2/samples/ latest/SampleApp_Linux.zip
加拿大（中部）区域	http://s3-ca- central-1.amazonaws.com/aws- codedeploy-ca-central-1/samples/ latest/SampleApp_Linux.zip
欧洲（爱尔兰）区域	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu-west-1/samples/ latest/SampleApp_Linux.zip
欧洲（伦敦）区域	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu-west-2/samples/ latest/SampleApp_Linux.zip
	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu-west-3/samples/ latest/SampleApp_Linux.zip
欧洲（法兰克福）区域	http://s3-eu- central-1.amazonaws.com/aws- codedeploy-eu-central-1/samples/ latest/SampleApp_Linux.zip
亚太区域（东京）	http://s3-ap- northeast-1.amazonaws.com/aws- codedeploy-ap-northeast-1/samples/ latest/SampleApp_Linux.zip
亚太区域（首尔）	http://s3-ap- northeast-2.amazonaws.com/aws- codedeploy-ap-northeast-2/samples/ latest/SampleApp_Linux.zip
亚太区域（新加坡）	http://s3-ap- southeast-1.amazonaws.com/aws- codedeploy-ap-southeast-1/samples/ latest/SampleApp_Linux.zip

区域	示例应用程序的位置
亚太区域 (悉尼)	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip
亚太地区 (孟买) 区域	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip
南美洲 (圣保罗) 区域	http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip

对于 Windows Server Amazon EC2 实例

区域	示例应用程序的位置
美国东部 (俄亥俄) 区域	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip
美国东部 (弗吉尼亚北部) 地区	http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip
美国西部 (加利福尼亚北部) 区域	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip
美国西部 (俄勒冈) 区域	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip
加拿大 (中部) 区域	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip
欧洲 (爱尔兰) 区域	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip
欧洲 (伦敦) 区域	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip
	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip
欧洲 (法兰克福) 区域	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip

区域	示例应用程序的位置
亚太区域 (东京)	<code>http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Windows.zip</code>
亚太区域 (首尔)	<code>http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip</code>
亚太区域 (新加坡)	<code>http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip</code>
亚太区域 (悉尼)	<code>http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip</code>
亚太地区 (孟买) 区域	<code>http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip</code>
南美洲 (圣保罗) 区域	<code>http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip</code>

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。

14. 将 Deployment description 框留空。
15. 在 Deployment configuration 下拉列表选定 CodeDeployDefault.OneAtATime 的情况下，选择 Deploy。

Note

要更新部署的当前状态，请刷新浏览器中的页面。

如果显示 Failed 而非 Succeeded，则您可能需要尝试[监控您的部署并排除故障 \(p. 60\)](#) (使用应用程序名称 **SimpleDemoApp** 和部署组名称 **SimpleDemoDG**) 中的一些技术。

步骤 3：检查结果

在此步骤中，您将检查 AWS CodeDeploy 是否已在 Auto Scaling 组中的单个 Amazon EC2 实例上安装 **SimpleDemoApp** 修订。

主题

- [检查结果 \(CLI\) \(p. 96\)](#)
- [检查结果 \(控制台\) \(p. 97\)](#)

检查结果 (CLI)

首先，您将需要 Amazon EC2 实例的公有 DNS。

可使用 AWS CLI 通过调用 `describe-instances` 命令获取 Auto Scaling 组中的 Amazon EC2 实例的公有 DNS。

在调用此命令之前，您将需要 Amazon EC2 实例的 ID。要获取该 ID，请按之前操作的那样，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 AWS CodeDeploy 将修订部署到 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，您将向 Auto Scaling 组中添加 Amazon EC2 实例。在 Auto Scaling 添加 Amazon EC2 实例之后，AWS CodeDeploy 将向新实例部署您的修订。

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/>。

In the Amazon EC2 navigation pane, under Auto Scaling, choose Auto Scaling Groups, and then choose the **CodeDeployDemo-AS-Group** entry.

在 Instances 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。此值应如下所示：**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 AWS CodeDeploy 将修订部署到 Auto Scaling 组中的单个 Amazon EC2 实例！

接下来，您将向 Auto Scaling 组中添加 Amazon EC2 实例。在 Auto Scaling 添加 Amazon EC2 实例之后，AWS CodeDeploy 将向新 Amazon EC2 实例部署您的修订。

步骤 4：增加 Auto Scaling 组中的 Amazon EC2 实例数

在此步骤中，您将指示 Auto Scaling 组创建其他 Amazon EC2 实例。在 Auto Scaling 创建实例之后，AWS CodeDeploy 将向该实例中部署您的修订。

主题

- [扩展 Auto Scaling 组中的 Amazon EC2 实例数 \(CLI\) \(p. 98\)](#)
- [扩展部署组中的 Amazon EC2 实例数 \(控制台\) \(p. 98\)](#)

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

1. 调用 `update-auto-scaling-group` 命令，以将名为 **CodeDeployDemo-AS-Group** 的 Auto Scaling 组中的 Amazon EC2 实例数从一个增加为两个。

在本地 Linux, macOS, or Unix 计算机上：

```
aws autoscaling update-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --min-size 2 \
  --max-size 2 \
  --desired-capacity 2
```

在本地 Windows 计算机上：

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --min-size 2 --max-size 2 --desired-capacity 2
```

2. 确保 Auto Scaling 组现在有两个 Amazon EC2 实例。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 下，选择 Auto Scaling Groups，然后选择 **CodeDeployDemo-AS-Group**。
2. 选择 Actions，然后选择 Edit。
3. 在 Details 选项卡上，在 Desired、Min 和 Max 框中键入 **2**，然后选择 Save。
4. 选择 Instances 选项卡。新的 Amazon EC2 实例应该会出现列表中。（如果该实例未出现，您可能需要选择几次 Refresh 按钮。）请在 InService 的值出现在 Lifecycle 列中并且 Healthy 的值出现在 Health Status 列中之后继续。

步骤 5：再次检查结果

在此步骤中，您将检查 AWS CodeDeploy 是否已在 Auto Scaling 组中的新实例上安装 SimpleDemoApp 修订。

主题

- [检查自动部署结果 \(CLI\) \(p. 99\)](#)
- [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。要获取该 ID，请对名为 **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 `get-deployment` 命令调用中使用那个未使用过的 ID：

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.[status, creator]" --output text
```

除了部署状态外，在命令输出中，您应该还会看到 `autoScaling`。（`autoScaling` 表示 Auto Scaling 已创建部署。）

请在部署状态显示 `Succeeded` 之后继续。

2. 在调用 `describe-instances` 命令之前，您将需要新 Amazon EC2 实例的 ID。要获取该 ID，请对 **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 `describe-instances` 命令输出中，记下新 Amazon EC2 实例的公有 DNS。

3. 使用如下所示的 URL，在 Web 浏览器中显示部署到该 Amazon EC2 实例的 **SimpleDemoApp** 修订：

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

如果出现恭喜页面，则表示您已使用 AWS CodeDeploy 将修订部署到 Auto Scaling 组中扩展的 Amazon EC2 实例！

检查自动部署结果（控制台）

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。Deployments 页显示有关已创建的部署 Auto Scaling 的信息。通常，您会自己创建部署，但 Auto Scaling 会代表您创建一个部署以将您的修订部署到新 Amazon EC2 实例。

Note

要更新部署的当前状态，请刷新浏览器中的页面。

3. 在显示 `Succeeded` 部署状态之后，请验证实例上的结果。您首先需要获取实例的公有 DNS：

4. In the Amazon EC2 navigation pane, under Auto Scaling, choose Auto Scaling Groups, and then choose the **CodeDeployDemo-AS-Group** entry.
5. 在 Instances 选项卡上，选择新 Amazon EC2 实例的 ID。
6. 在 Instances 页中的 Description 选项卡上，记下 Public DNS 值。此值应如下所示：**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 AWS CodeDeploy 将修订部署到 Auto Scaling 组中扩展的 Amazon EC2 实例！

步骤 6：清除

在此步骤中，您将删除 Auto Scaling 组，以避免对您在教程中使用的资源持续收费。（可选）您可以删除 Auto Scaling 配置和 AWS CodeDeploy 部署组件记录。

主题

- [清除资源 \(CLI\) \(p. 100\)](#)
- [清除资源 \(控制台\) \(p. 100\)](#)

清除资源 (CLI)

1. 通过对 **CodeDeployDemo-AS-Group** 调用 delete-auto-scaling-group 命令，删除 Auto Scaling 组。这同时将终止 Amazon EC2 实例。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete
```

2. （可选）通过对名为 **CodeDeployDemo-AS-Configuration** 的启动配置调用 delete-launch-configuration 命令，删除 Auto Scaling 启动配置：

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

3. （可选）通过对名为 **SimpleDemoApp** 的应用程序调用 delete-application 命令，从 AWS CodeDeploy 中删除应用程序。这同时将删除所有关联的部署、部署组和修订记录。

```
aws deploy delete-application --application-name SimpleDemoApp
```

清除资源 (控制台)

1. 删除 Auto Scaling 组。这同时将终止 Amazon EC2 实例：

登录 AWS 管理控制台并通过以下网址打开 Amazon EC2 控制台 <https://console.aws.amazon.com/ec2/>。

2. In the Amazon EC2 navigation pane, under Auto Scaling, choose Auto Scaling Groups, and then choose the **CodeDeployDemo-AS-Group** entry.
3. 依次选择 Actions、Delete 和 Yes, Delete。

4. （可选）删除启动配置。在导航栏中的 Auto Scaling 下，选择 Launch Configurations，然后选择 **CodeDeployDemo-AS-Configuration**。
5. 依次选择 Actions、Delete launch configuration 和 Yes, Delete。
6. （可选）从 AWS 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。通过以下网址打开 AWS CodeDeploy 控制台：<https://console.aws.amazon.com/codedeploy>。
7. 在 AWS CodeDeploy 菜单上，选择 Applications。
8. 在应用程序列表中，选择 SimpleDemoApp。
9. 在 Application details 页上，选择 Delete application。
10. 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序

在本教程中，您将使用 AWS CodeDeploy 将示例应用程序修订从 GitHub 部署到一个运行 Amazon Linux 的 Amazon EC2 实例、一个 Red Hat Enterprise Linux (RHEL) 实例或一个 Windows Server 实例。有关 GitHub 与 AWS CodeDeploy 的集成的信息，请参阅 [将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)。

Note

您还可使用 AWS CodeDeploy 将应用程序修订从 GitHub 部署到 Ubuntu Server 实例。您可使用[教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \) \(p. 79\)](#)中的[步骤 2：创建示例应用程序修订 \(p. 80\)](#)中描述的示例修订，也可创建与 Ubuntu Server 实例和 AWS CodeDeploy 兼容的修订。要创建您自己的修订，请参阅[计划 AWS CodeDeploy 的修订 \(p. 208\)](#)和将应用程序规范文件添加到 AWS CodeDeploy 的[修订 \(p. 209\)](#)。

主题

- [先决条件 \(p. 101\)](#)
- [步骤 1：设置 GitHub 账户 \(p. 101\)](#)
- [步骤 2：创建 GitHub 存储库 \(p. 102\)](#)
- [步骤 3：将示例应用程序上传到 GitHub 存储库 \(p. 103\)](#)
- [步骤 4：预置实例 \(p. 106\)](#)
- [步骤 5：创建应用程序和部署组 \(p. 106\)](#)
- [步骤 6：将应用程序部署到实例 \(p. 107\)](#)
- [步骤 7：监控和验证部署 \(p. 110\)](#)
- [步骤 8：清除 \(p. 111\)](#)

先决条件

在开始本教程之前，请执行以下操作：

- 在本地计算机上安装 Git。要安装 Git，请参阅 [Git 下载](#)。
- 完成[AWS CodeDeploy 入门 \(p. 16\)](#)中的步骤，包括安装和配置 AWS CLI。这在您需要使用 AWS CLI 将修订从 GitHub 部署到实例时特别重要。

步骤 1：设置 GitHub 账户

您需要拥有 GitHub 账户才能创建用于存储修订的 GitHub 存储库。如果您已拥有 GitHub 账户，请向前跳至[步骤 2：创建 GitHub 存储库 \(p. 102\)](#)。

1. 转至 <https://github.com/join>。
2. 键入用户名、您的电子邮件地址和密码。
3. 选择 Sign up for GitHub，然后按照说明执行操作。

步骤 2：创建 GitHub 存储库

您将需要 GitHub 存储库来存储修订。

如果您已拥有 GitHub 存储库，请确保将本教程中的任何 **CodeDeployGitHubDemo** 替换为该存储库的名称，然后向前跳至 [步骤 3：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 Repository name 框中，键入 **CodeDeployGitHubDemo**。
- 选择 Public。

Note

选择默认的 Public 选项意味着任何人都可查看此存储库。虽然可选择 Private 选项来限制可对此存储库进行查看和提交操作的人员，但此选项可能会产生额外的 GitHub 费用。

- 清除 Initialize this repository with a README 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）。有关信息，请参阅[提供您的双重验证代码](#)。

在本地 Linux, macOS, or Unix 计算机上：

1. 从终端中运行以下命令（一次运行一条命令），其中 **user-name** 是您的 GitHub 用户名：

```
mkdir /tmp/CodeDeployGitHubDemo
```

```
cd /tmp/CodeDeployGitHubDemo
```

```
touch README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

2. 在 /tmp/CodeDeployGitHubDemo 位置保持终端打开。

在本地 Windows 计算机上：

1. 以管理员身份从命令提示符运行以下命令（一次运行一条命令）：

```
mkdir c:\temp\CodeDeployGitHubDemo
```

```
cd c:\temp\CodeDeployGitHubDemo
```

```
notepad README.md
```

2. 在记事本中，保存 README.md 文件。关闭记事本。运行以下命令（一次运行一条命令），其中 **user-name** 是您的 GitHub 用户名：

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

3. 在 c:\temp\CodeDeployGitHubDemo 位置保持终端打开。

步骤 3：将示例应用程序上传到 GitHub 存储库

在此步骤中，您将示例修订从公有 Amazon S3 存储桶中复制到 GitHub 存储库。(为简单起见，为本教程提供的示例修订为单一网页。)

Note

如果您使用您的某个修订而不是我们的示例修订，则您的修订必须：

- 遵循[计划 AWS CodeDeploy 的修订 \(p. 208\)](#)和[将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)中的准则。
- 使用相应的实例类型。
- 可从您的 GitHub 仪表板访问。

如果您的修订满足这些要求，请向前跳至[步骤 5：创建应用程序和部署组 \(p. 106\)](#)。

如果您要部署到 Ubuntu Server 实例，则需将与 Ubuntu Server 实例和 AWS CodeDeploy 兼容的修订上传到您的 GitHub 存储库。有关更多信息，请参阅[计划 AWS CodeDeploy 的修订 \(p. 208\)](#)和[将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)。

主题

- [从本地 Linux, macOS, or Unix 计算机推送示例修订 \(p. 104\)](#)
- [从本地 Windows 计算机推送示例修订 \(p. 105\)](#)

从本地 Linux, macOS, or Unix 计算机推送示例修订

例如，在终端在 /tmp/CodeDeployGitHubDemo 位置仍处于打开状态的情况下，运行以下命令（一次运行一条命令）：

Note

如果您计划部署到 Windows Server 实例，请在命令中使用 SampleApp_Windows.zip 替换 SampleApp_Linux.zip。

```
(Amazon S3 copy command)
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中，*(Amazon S3 copy command)* 为下列项之一：

- `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2` (对于美国东部 (俄亥俄州) 区域)
- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1` (对于美国东部 (弗吉尼亚北部) 区域)
- `aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region us-west-1` (对于美国西部 (加利福尼亚北部) 区域)
- `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2` (对于美国西部 (俄勒冈) 区域)
- `aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1` (对于加拿大 (中部) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1` (对于欧洲 (爱尔兰) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2` (对于欧洲 (伦敦) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3` (对于欧洲 (巴黎) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1` (对于欧洲 (法兰克福) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1` (对于亚太区域 (东京) 区域)

- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2` (对于亚太区域 (首尔) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1` (对于亚太区域 (新加坡))
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2` (对于亚太区域 (悉尼) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1` (对于亚太地区 (孟买) 区域)
- `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1` (对于南美洲 (圣保罗) 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，请在命令中使用 `SampleApp_Linux.zip` 替换 `SampleApp_Windows.zip`。

(Amazon S3 copy command)

将 ZIP 文件的内容直接解压缩到本地目录 (例如 `c:\temp\CodeDeployGitHubDemo`)，而不要解压缩到一个新的子目录中。

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

其中，(Amazon S3 copy command) 为下列项之一：

- `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip . --region us-east-2` (对于美国东部 (俄亥俄州) 区域)
- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip . --region us-east-1` (对于美国东部 (弗吉尼亚北部) 区域)
- `aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip . --region us-west-1` (对于美国西部 (加利福尼亚北部) 区域)
- `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip . --region us-west-2` (对于美国西部 (俄勒冈) 区域)
- `aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip . --region ca-central-1` (对于加拿大 (中部) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip . --region eu-west-1` (对于欧洲 (爱尔兰) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip . --region eu-west-2` (对于欧洲 (伦敦) 区域)
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip . --region eu-west-3` (对于欧洲 (巴黎) 区域)

- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip . --region eu-central-1` (对于欧洲 (法兰克福) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Windows.zip . --region ap-northeast-1` (对于亚太区域 (东京) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip . --region ap-northeast-2` (对于亚太区域 (首尔) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip . --region ap-southeast-1` (对于亚太区域 (新加坡))
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip . --region ap-southeast-2` (对于亚太区域 (悉尼) 区域)
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip . --region ap-south-1` (对于亚太地区 (孟买) 区域)
- `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip . --region sa-east-1` (对于南美洲 (圣保罗) 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
```

步骤 4：预置实例

在此步骤中，您将创建或配置示例应用程序将部署到的实例。您可以部署到 Amazon EC2 实例或本地实例 (运行 AWS CodeDeploy 支持的操作系统之一)。有关信息，请参阅[AWS CodeDeploy 代理支持的操作系统 \(p. 113\)](#)。(如果您已有配置为在 AWS CodeDeploy 部署中使用的实例，请跳至下一步。)

要创建或配置实例，请参阅[使用适用于 AWS CodeDeploy 的实例 \(p. 141\)](#)，然后返回此页。

Note

要快速创建针对本教程的新实例，建议您使用 Amazon EC2 控制台。请参阅 [启动 Amazon EC2 实例 \(控制台\) \(p. 141\)](#)。

要验证 AWS CodeDeploy 代理是否正在实例上运行，请参阅[验证 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)。

在成功启动或配置实例并确认 AWS CodeDeploy 代理正在运行之后，转至下一步。

步骤 5：创建应用程序和部署组

在此步骤中，您将使用 AWS CodeDeploy 控制台或 AWS CLI 创建应用程序和部署组 (用于部署来自 GitHub 存储库的示例修订)。

创建应用程序和部署组 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 Applications 页面上，选择 Create application。

Note

如果您尚未创建任何应用程序，并且已显示 AWS CodeDeploy 开始页面，请选择 Get Started Now，使用 Sample deployment wizard 完成部署，然后返回本主题。

3. 在 Application name 框中，键入 **CodeDeployGitHubDemo-App**。
4. 在 Deployment group name 框中，键入 **CodeDeployGitHubDemo-DepGrp**。
5. 在 Deployment type 中，选择 In-place deployment。
6. 在 Environment configuration 中，根据您使用的实例类型，选择 Amazon EC2 instances 选项卡或 On-premises instances 选项卡。在 Key 和 Value 框中，根据[步骤 4：预置实例 \(p. 106\)](#)的介绍，键入应用于实例的实例标签键和值。
7. 在 Service role ARN 下拉列表中，选择服务角色 ARN。(如果您需要查找您的服务角色 ARN，请按照[获取服务角色 ARN \(控制台\) \(p. 21\)](#)中的说明执行操作。)
8. 选择 Create application，并继续下一步。

创建应用程序和部署组 (CLI)

1. 调用 create-application 命令以在 AWS CodeDeploy 中创建一个名为 CodeDeployGitHubDemo-App 的应用程序：

```
aws deploy create-application --application-name CodeDeployGitHubDemo-App
```

2. 调用 create-deployment-group 命令以创建一个名为 CodeDeployGitHubDemo-DepGrp 的部署组：
 - 如果您要部署到 Amazon EC2 实例，则 **ec2-tag-key** 是作为[步骤 4：预置实例 \(p. 106\)](#)的一部分应用于您的 Amazon EC2 实例的 Amazon EC2 实例标签密钥。
 - 如果您要部署到 Amazon EC2 实例，则 **ec2-tag-value** 是作为[步骤 4：预置实例 \(p. 106\)](#)的一部分应用于您的 Amazon EC2 实例的 Amazon EC2 实例标签值。
 - 如果您要部署到本地实例，则 **on-premises-tag-key** 是作为[步骤 4：预置实例 \(p. 106\)](#)的一部分应用于本地实例的本地实例标签密钥。
 - 如果您要部署到本地实例，则 **on-premises-tag-value** 是作为[步骤 4：预置实例 \(p. 106\)](#)的一部分应用于本地实例的本地实例标签值。
 - **service-role-arn** 为服务角色 ARN。(按照[获取服务角色 ARN \(CLI\) \(p. 21\)](#)中的说明执行操作可查找服务角色 ARN。)

```
aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-App --  
ec2-tag-filters Key=ec2-tag-key,Type=KEY_AND_VALUE,Value=ec2-tag-value --on-premises-  
tag-filters Key=on-premises-tag-key,Type=KEY_AND_VALUE,Value=on-premises-tag-value --  
deployment-group-name CodeDeployGitHubDemo-DepGrp --service-role-arn service-role-arn
```

Note

create-deployment-group 命令支持创建触发器，这些触发器导致向主题订阅者发送有关部署和实例中的指定事件的 Amazon SNS 通知。此命令还支持以下选项：自动回滚部署和设置警报以便在满足 Amazon CloudWatch 警报中的监控阈值时停止部署。本教程中未包含适用于这些操作的命令。

步骤 6：将应用程序部署到实例

在此步骤中，您将使用 AWS CodeDeploy 控制台或 AWS 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 Application details 页上的 Deployment groups 中，选择 CodeDeployGitHubDemo-DepGrp 旁边的按钮。
2. 在 Actions 菜单中，选择 Deploy new revision。
3. 在 Create deployment 页上的 Repository type 区域中，选择 My application is stored in GitHub。
4. 在 Connect to GitHub 中，执行下列操作之一：
 - 要为 AWS 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 AWS CodeDeploy 与名为 CodeDeployGitHubDemo-App 的应用程序的 GitHub 进行交互。继续执行步骤 5。
 - 要使用已创建的连接，请在 GitHub account 中，选择其名称，然后选择 Connect to GitHub。继续执行步骤 7。
 - 要创建与其他 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account，然后选择 Connect to 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 AWS CodeDeploy Create deployment 页上的 Repository name 框中，键入用于登录的 GitHub 用户名，并且后跟正斜杠 (/) 和已在其中推送应用程序修订的存储库的名称 (例如，**my-github-user-name/CodeDeployGitHubDemo**)。

如果您不确定要键入的值，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 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，依次后跟正斜杠字符 (/) 和存储库的名称。将此显示的值键入 Repository name 框中。

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。

8. 在 Commit ID 框中，键入与将应用程序修订推送到 GitHub 这一操作关联的提交的 ID。

如果您不确定要键入的值，请执行以下步骤：

1. 在单独的 Web 浏览器选项卡中，转到您的 [GitHub 仪表板](#)。
 2. 在 Your repositories 中，选择 CodeDeployGitHubDemo。
 3. 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。)
 4. 将提交 ID 粘贴到 Commit ID 框中。
9. 选择 Deploy，然后继续执行下一步。

部署修订 (CLI)

您必须先为 AWS CodeDeploy 提供权限以使用您的 GitHub 用户账户与 CodeDeployGitHubDemo-App 应用程序的 GitHub 进行交互，然后才能调用任何与 GitHub 交互的 AWS CLI 命令 (例如，您接下来将调用的 create-deployment 命令)。当前，您必须使用 AWS CodeDeploy 控制台执行此操作。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。
3. 选择 Create deployment。

Note

您不会创建新的部署。这是当前授权 AWS CodeDeploy 代表您的 GitHub 用户账户与 GitHub 交互的唯一方式。

4. 从 Application 下拉列表中，选择 CodeDeployGitHubDemo-App。
5. 从 Deployment group 下拉列表中，选择 CodeDeployGitHubDemo-DepGrp。
6. 在 Repository type 区域中，选择 My application is stored in GitHub。
7. 在 Connect to GitHub 中，执行下列操作之一：
 - 要为 AWS 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 AWS CodeDeploy 与名为 CodeDeployGitHubDemo-App 的应用程序的 GitHub 进行交互。继续执行步骤 8。
 - 要使用已创建的连接，请在 GitHub account 中，选择其名称，然后选择 Connect to GitHub。继续执行步骤 10。
 - 要创建与其他 GitHub 账户的连接，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account，然后选择 Connect to 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 AWS CodeDeploy Create deployment 页上，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 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，依次后跟正斜杠 (/) 和存储库的名称。这是要使用的值。

Note

如果目标存储库名称未显示在 Your repositories 中，请使用 Search GitHub 框查找目标存储库和对应的 GitHub 用户或组织名。

- **commit-id** 是与您已推送到存储库的应用程序修订的版本关联的提交 (例如，f835159a...528eb76f)。

如果您不确定要使用的值，请执行以下步骤：

1. 在单独的 Web 浏览器选项卡中，转到您的 [GitHub 仪表板](#)。
2. 在 Your repositories 中，选择 CodeDeployGitHubDemo。
3. 在提交列表中，查找与将应用程序修订推送到 GitHub 这一操作关联的提交 ID。此 ID 的长度通常为 40 个字符并包含字母和数字。(请不要使用提交 ID 的较短版本，它通常是较长版本的前 10 个字符。) 请使用此值。

如果您正在本地 Linux, macOS, or 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

在此步骤中，您将使用 AWS CodeDeploy 控制台或 AWS CLI 验证部署是否成功。您将使用 Web 浏览器来查看已部署到您已创建或配置的实例的网页。

Note

如果您要部署到 Ubuntu Server 实例，请使用您自己的测试策略来确定已部署的修订是否在实例上按预期运行，然后转至下一步。

监控和验证部署（控制台）

1. 如果未显示 Deployments 页，请在 AWS CodeDeploy 菜单上，选择 Deployments。
2. 在部署列表中，查找 Application 值为 CodeDeployGitHubDemo-App 且 Deployment group 值为 CodeDeployGitHubDemo-DepGrp 的行。如果 Status 列中未显示 Succeeded 或 Failed，请定期选择 Refresh 按钮。
3. 如果 Status 列中显示 Failed，请按照[查看实例详细信息（控制台）](#) (p. 179) 中的说明执行操作以排查部署的问题。
4. 如果 Status 列中显示 Succeeded，则现在可通过 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，转至该实例的 [http://*public-dns*](http://<i>public-dns</i>)（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 AWS CodeDeploy 从 GitHub 部署修订，可向前跳至[步骤 8：清除](#) (p. 111)。

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，并通过 list-deployments 命令提供输出中的部署 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. 179) 中的说明执行操作以排查部署的问题。
4. 如果返回 Succeeded，则可立即尝试通过 Web 浏览器验证部署。我们的示例修订是已部署到实例的单个网页。如果您要部署到 Amazon EC2 实例，可通过转至面向 Amazon EC2 实例的 [http://*public-*](http://<i>public-</i>)

`dns` (例如, `http://ec2-01-234-567-890.compute-1.amazonaws.com`) 在 Web 浏览器中查看此页。

5. 如果您能看到此网页, 那么恭喜您! 您已成功使用 AWS 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 AWS 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 AWS CloudFormation 堆栈 (如果您已使用 AWS CloudFormation 模板创建 Amazon EC2 实例)

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台: <https://console.aws.amazon.com/cloudformation>。
2. 在 Stack Name 列中, 选中以 `CodeDeploySampleStack` 开头的堆栈旁的框。
3. 选择 Delete Stack。
4. 在系统提示时, 选择 Yes, Delete。这将删除 Amazon EC2 实例和关联的 IAM 实例配置文件以及服务角色。

手动取消注册并清除本地实例 (如果您已预置本地实例)

1. 使用 AWS CLI 对本地实例 (此处由 `your-instance-name` 表示) 和关联区域 (由 `your-region` 表示) 调用 `deregister` 命令:

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

2. 从本地实例调用 `uninstall` 命令:

```
aws deploy uninstall
```

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

1. 登录 AWS 管理控制台并通过以下网址打开 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 AWS CodeDeploy 部署组件记录

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
3. 选择 CodeDeployGitHubDemo-App。
4. 在 Application details 页上的 Deployment groups 中，选择部署组旁边的按钮。在 Actions 菜单上，选择 Delete。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。
5. 在 Application details 页的底部，选择 Delete application。
6. 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

删除 GitHub 存储库

请参阅 [GitHub 帮助](#) 中的 [删除存储库](#)。

使用 AWS CodeDeploy 代理

AWS 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 AWS CodeDeploy 部署中使用该实例。

Note

只有当您部署到 EC2/本地 计算平台 时，才需要 AWS CodeDeploy 代理。使用 AWS Lambda 计算平台的部署不需要代理。

安装该代理时，将在实例上放置一个配置文件。此文件用于指定代理的工作方式。此配置文件指定 AWS 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 AWS CodeDeploy 代理配置文件的信息，请参阅[AWS CodeDeploy 代理配置参考 \(p. 297\)](#)。

有关使用 AWS 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 AWS CodeDeploy 代理操作 \(p. 118\)](#)。

主题

- [AWS CodeDeploy 代理支持的操作系统 \(p. 113\)](#)
- [AWS CodeDeploy 代理的通信协议和端口 \(p. 114\)](#)
- [面向 AWS CodeDeploy 代理的、适用于 Ruby 的 AWS 开发工具包 \(aws-sdk-core\) 支持 \(p. 114\)](#)
- [AWS CodeDeploy 代理的版本历史纪录 \(p. 114\)](#)
- [应用程序修订和日志文件清理 \(p. 116\)](#)
- [AWS CodeDeploy 代理安装的文件 \(p. 116\)](#)
- [管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)

AWS CodeDeploy 代理支持的操作系统

支持的 Amazon EC2 AMI 操作系统

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

- Amazon Linux 2017.03.x、2016.09.0、2016.03.1、2016.03.0、2015.03、2014.09.1
- Ubuntu Server 16.04 LTS 和 14.04 LTS
- Microsoft Windows Server 2016、2012 R2 以及 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

AWS 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，请转至 GitHub 中的 [AWS CodeDeploy 代理存储库](#)。

支持的本地操作系统

已在以下本地操作系统上测试 AWS CodeDeploy 代理：

- Ubuntu Server 14.04 LTS
- Microsoft Windows Server 2016、2012 R2 以及 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

AWS 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，请转至 GitHub 中的 [AWS CodeDeploy 代理存储库](#)。

AWS CodeDeploy 代理的通信协议和端口

AWS CodeDeploy 代理通过端口 443 使用 HTTPS 进行出站通信。

面向 AWS CodeDeploy 代理的、适用于 Ruby 的 AWS 开发工具包 (aws-sdk-core) 支持

1.0.1.880 版本之前的 AWS CodeDeploy 代理版本只能与版本 2.1.2 的适用于 Ruby 的 AWS 开发工具包 (aws-sdk-core 2.1.2) 及早期版本兼容。如果您使用的是 1.0.1.880 版本之前的 AWS CodeDeploy 代理版本，建议您更新到最新版本。有关信息，请参阅以下内容：

- [确定 AWS CodeDeploy 代理的版本 \(p. 120\)](#)
- [安装或重新安装 AWS CodeDeploy 代理 \(p. 121\)](#)

可与 AWS CodeDeploy 代理兼容的、最新版本的适用于 Ruby 的 AWS 开发工具包为 aws-sdk-core 2.3。

AWS CodeDeploy 代理的版本历史纪录

您的实例必须运行支持的 AWS CodeDeploy 代理版本。当前支持的最低版本为 。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 AWS 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
1.0.1.1518	2018 年 12 月 6 日	<p>增强功能：修正了一个问题，该问题导致 AWS CodeDeploy 代理在接受轮询请求的同时关闭时出错。</p> <p>增强功能：添加了一项部署跟踪功能，可防止 AWS CodeDeploy 代理在部署进行中关闭。</p> <p>增强功能：改进了删除文件时的性能。</p>
1.0.1.1458	2018 年 3 月 6 日	<p>增强功能：改进了证书验证，以支持更多受信任的机构。</p> <p>增强功能：修复了当部署包含 BeforeInstall 生命周期事件时导致本地 CLI 失败的问题。</p> <p>增强功能：修复了在更新 AWS CodeDeploy 代理时，可能导致活动部署失败的问题。</p>
1.0.1.1352	2017 年 11 月 16 日	<p>功能：引入了一项新功能，用于测试和调试安装了 AWS CodeDeploy 代理的本地计算机或实例 EC2/本地 部署。</p>
1.0.1.1106	2017 年 5 月 16 日	<p>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。</p> <p>增强功能：使 AWS CodeDeploy 代理与适用于 Ruby 的 AWS 开发工具包版本 2.9.2 (aws-sdk-core 2.9.2) 兼容。</p>

版本	发行日期	详细信息
1.0.1.1095	2017 年 3 月 29 日	<p>增强功能：在 中国（北京）区域 中引入了对 AWS CodeDeploy 代理的支持。</p> <p>增强功能：生命周期事件挂钩进行调用时，允许 Puppet 在 Windows Server 实例上运行。</p> <p>增强功能：改进对 <code>untar</code> 操作的处理。</p>
1.0.1.1067	2017 年 1 月 6 日	<p>增强功能：修改了很多错误消息，以包含导致部署故障的更具体的原因。</p> <p>增强功能：修复了导致 AWS CodeDeploy 代理无法在一些部署中指定要部署的正确应用程序版本的问题。</p> <p>增强功能：恢复在 <code>untar</code> 操作之前和之后使用 <code>pushd</code> 和 <code>popd</code>。</p>
1.0.1.1045	2016 年 11 月 21 日	<p>增强功能：使 AWS CodeDeploy 代理与适用于 Ruby 的 AWS 开发工具包版本 2.6.11 (aws-sdk-core 2.6.11) 兼容。</p>
1.0.1.1037	2016 年 10 月 19 日	<p>更新了用于 Amazon Linux、RHEL 和 Ubuntu Server 实例的 AWS CodeDeploy 代理，进行了以下更改。对于 Windows Server 实例，最新版本仍为 1.0.1.998。</p> <p>增强：该代理现在可以确定哪个 Ruby 版本安装在实例上，以使它可以使用该版本调用 <code>codedeploy-agent</code> 脚本。</p>
1.0.1.1011.1	2016 年 8 月 17 日	<p>增强功能：删除了由于外壳支持问题而在 1.0.1.1011 版中引入的更改。此版本代理在功能上与 2016 年 7 月 11 日发布的 1.0.1.998 版相同。</p>
1.0.1.1011	2016 年 8 月 15 日	<p>更新了用于 Amazon Linux、RHEL 和 Ubuntu Server 实例的 AWS CodeDeploy 代理，进行了以下更改。对于 Windows Server 实例，最新版本仍为 1.0.1.998。</p> <p>功能：增加了在使用 <code>systemd</code> 初始化系统的操作系统上使用 <code>bash</code> shell 调用 AWS CodeDeploy 代理的支持。</p> <p>增强功能：在 AWS CodeDeploy 代理和 AWS 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 AWS CodeDeploy 代理不再单纯依赖 Ruby 2.0。（AWS CodeDeploy 代理安装程序的 <code>deb</code> 和 <code>rpm</code> 版本仍然需要 Ruby 2.0。）</p>
1.0.1.998	2016 年 7 月 11 日	<p>增强功能：修复了对使用 <code>root</code> 之外的用户配置文件运行 AWS CodeDeploy 代理的支持。名为 <code>USER</code> 的变量将被替换为 <code>CODEDEPLOY_USER</code>，以避免与环境变量发生冲突。</p>
1.0.1.966	2016 年 6 月 16 日	<p>功能：推出了对使用 <code>root</code> 之外的用户配置文件运行 AWS CodeDeploy 代理的支持。</p> <p>增强功能：修复了对指定您希望 AWS CodeDeploy 代理为部署组存档的应用程序修订数量的支持。</p> <p>增强功能：使 AWS CodeDeploy 代理与适用于 Ruby 的 AWS 开发工具包版本 2.3 (aws-sdk-core 2.3) 兼容。</p> <p>增强功能：修复了与部署期间的 UTF-8 编码有关的问题。</p> <p>增强功能：提高了标识进程名称时的准确性。</p>

版本	发行日期	详细信息
1.0.1.950	2016 年 3 月 24 日	功能：添加了安装代理支持。 增强功能：更新了安装脚本，如果已安装最新版本，则不会下载 AWS CodeDeploy 代理。
1.0.1.934	2016 年 2 月 11 日	功能：引入了对指定您要 AWS 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	增强功能：使 AWS CodeDeploy 代理与适用于 Ruby 的 AWS 开发工具包版本 2.2 (aws-sdk-core 2.2) 兼容。版本 2.1.2 仍受支持。
1.0.1.854	2015 年 11 月 17 日	功能：引入了对 SHA-256 哈希算法的支持。 Important 2016 年 10 月 17 日后，AWS CodeDeploy 代理的所有安装必须至少更新到版本 1.0.1.854，否则部署将失败。有关更多信息，请参阅 部署失败，显示消息“PKCS7 签名的消息验证失败” (p. 313)。 功能：在 .version 文件中引入了版本跟踪支持。 功能：通过使用环境变量使部署组 ID 可用。 增强功能：添加了对使用 Amazon CloudWatch Logs 监控 AWS CodeDeploy 代理日志的支持。

有关相关信息，请参见下列内容：

- [确定 AWS CodeDeploy 代理的版本](#) (p. 120)
- [安装或重新安装 AWS CodeDeploy 代理](#) (p. 121)

有关 AWS CodeDeploy 代理版本的历史记录，请参阅 [GitHub 上的版本存储库](#)。

应用程序修订和日志文件清理

AWS CodeDeploy 代理将修订和日志文件存档在实例上。AWS 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：可以使用代理配置文件中的 `:max_revisions:` 选项，通过输入任何正整数来指定要存档的应用程序修订的数目。AWS CodeDeploy 还对这些修订的日志文件进行存档。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，AWS CodeDeploy 还将保留五个最新修订。

AWS CodeDeploy 日志：对于 Amazon Linux、Ubuntu Server 和 RHEL 实例，AWS CodeDeploy 代理在 `/var/log/aws/codedeploy-agent` 文件夹下轮换日志文件。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`。

AWS CodeDeploy 代理安装的文件

AWS 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' (对于 Amazon Linux、Ubuntu Server 和 RHEL 实例)。

'C:\ProgramData\Amazon\CodeDeploy' (对于 Windows Server 实例)。

您可以使用 AWS CodeDeploy 代理配置文件中的 `root_dir` 设置来配置该目录的名称和位置。有关更多信息，请参阅 [AWS CodeDeploy 代理配置参考](#) (p. 297)。

下面是根目录下文件和目录结构的示例。该结构假定有 N 个部署组，每个部署组包含 N 个部署。

```
--deployment-root/
-- deployment group 1 ID
|   |-- deployment 1 ID
|   |   |-- Contents and logs of the deployment's revision
|   |   |-- deployment 2 ID
|   |   |   |-- Contents and logs of the deployment's revision
|   |   |-- deployment N ID
|   |   |   |-- Contents and logs of the deployment's revision
-- deployment group 2 ID
|   |-- deployment 1 ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
|   |-- deployment 2 ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
|   |-- deployment N ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
-- deployment group N ID
|   |-- deployment 1 ID
|   |   |-- Contents and logs of the deployment's revision
|   |   |-- deployment 2 ID
|   |   |   |-- Contents and logs of the deployment's revision
|   |   |-- deployment N ID
|   |   |   |-- Contents and logs of the deployment's revision
-- deployment-instructions
|   |-- [deployment group 1 ID]_cleanup
|   |-- [deployment group 2 ID]_cleanup
|   |-- [deployment group N ID]_cleanup
|   |-- [deployment group 1 ID]_install.json
|   |-- [deployment group 2 ID]_install.json
|   |-- [deployment group N ID]_install.json
|   |-- [deployment group 1 ID]_last_successful_install
|   |-- [deployment group 2 ID]_last_successful_install
|   |-- [deployment group N ID]_last_successful_install
|   |-- [deployment group 1 ID]_most_recent_install
|   |-- [deployment group 2 ID]_most_recent_install
|   |-- [deployment group N ID]_most_recent_install
-- deployment-logs
|   |-- codedeploy-agent-deployments.log
```

- Deployment Group ID 文件夹代表您的每个部署组。部署组目录的名称是其 ID (例如，acde1916-9099-7caf-fd21-012345abcdef)。每个部署组目录都包含该部署组中每次尝试的部署的一个子目录。

您可以使用 [batch-get-deployments](#) 命令查找部署组 ID。

- Deployment ID 文件夹代表部署组中的每个部署。每个部署目录的名称都是其 ID。每个文件夹都包含：
 - bundle.tar，一个压缩文件，其中包含部署修订的内容。
 - 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 file 中指定的所有脚本的输出。

如果您想要查找部署文件夹，但不知道其部署 ID 或部署组 ID，则可以使用 [AWS CodeDeploy 控制台](#) 或 [AWS CLI](#) 来查找它们。有关更多信息，请参阅 [使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)。

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 AWS CodeDeploy 代理配置文件中的 max_revisions 设置来更改该默认值。有关更多信息，请参阅 [AWS CodeDeploy 代理配置参考 \(p. 297\)](#)。

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，例如一个或两个。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销版本。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。该文件在 AWS CodeDeploy 代理将部署应用程序中的所有文件复制到实例时创建。它由 AWS CodeDeploy 代理在下次部署期间使用，以确定要运行哪些 ApplicationStop 和 BeforeInstall 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 Deployment ID 文件夹下的 logs 子文件夹中。此文件中的条目前面带有部署 ID。例如，“[d-ABCDEF123]LifecycleEvent - BeforeInstall”可能是在 ID 为 d-ABCDEF123 的部署期间写入的。当 codedeploy-agent-deployments.log 达到其最大大小时，AWS CodeDeploy 代理将继续向其中写入内容，同时删除旧内容。

管理 AWS 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 AWS CodeDeploy 代理，以及如何验证 AWS CodeDeploy 代理是否正在运行。

主题

- [验证 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)
- [确定 AWS CodeDeploy 代理的版本 \(p. 120\)](#)

- [安装或重新安装 AWS CodeDeploy 代理 \(p. 121\)](#)
- [更新 AWS CodeDeploy 代理 \(p. 127\)](#)
- [卸载 AWS CodeDeploy 代理 \(p. 131\)](#)

验证 AWS CodeDeploy 代理是否正在运行

本部分介绍在您怀疑 AWS CodeDeploy 代理已停止在某个实例上运行时要运行的命令。

主题

- [验证适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)
- [验证适用于 Ubuntu Server 的 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)
- [验证适用于 Windows Server 的 AWS CodeDeploy 代理是否正在运行 \(p. 120\)](#)

验证适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理是否正在运行

要查看 AWS 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 AWS CodeDeploy 代理。请按照[安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理 \(p. 121\)](#)中所述进行安装。

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

如果您看到类似于 error: No AWS CodeDeploy agent running 的消息，请启动该服务并依次运行以下两个命令：

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

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

验证适用于 Ubuntu Server 的 AWS CodeDeploy 代理是否正在运行

要查看 AWS 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 AWS CodeDeploy 代理。请按照[安装或重新安装适用于 Ubuntu Server 的 AWS CodeDeploy 代理 \(p. 122\)](#)中所述进行安装。

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

如果您看到类似于 error: No AWS CodeDeploy agent running 的消息，请启动该服务并依次运行以下两个命令：

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



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

验证适用于 Windows Server 的 AWS CodeDeploy 代理是否正在运行

要查看 AWS 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 AWS CodeDeploy 代理。请按照[安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理](#) (p. 123)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 AWS 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 AWS CodeDeploy 代理的版本。

首先，从 AWS CodeDeploy 代理版本 1.0.1.854 开始，您可以在实例上的 .version 文件中查看版本号。下表显示了每个受支持的操作系统的位置和示例版本字符串。

操作系统	文件位置	示例 agent_version 字符串
Amazon Linux 和 Red Hat Enterprise Linux (RHEL)	/opt/codedeploy-agent/.version	OFFICIAL_1.0.1.854_rpm
Ubuntu Server	/opt/codedeploy-agent/.version	OFFICIAL_1.0.1.854_deb
Windows Server	C:\ProgramData\Amazon\CodeDeploy\.version	OFFICIAL_1.0.1.854_msi

其次，您可以在实例上运行命令来确定 AWS CodeDeploy 代理的版本。

主题

- 在 [Amazon Linux 或 RHEL 上确定版本](#) (p. 121)
- 在 [Ubuntu Server 上确定版本](#) (p. 121)
- 在 [Windows Server 上确定版本](#) (p. 121)

在 Amazon Linux 或 RHEL 上确定版本

登录到实例并运行以下命令：

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

在 Ubuntu Server 上确定版本

登录到实例并运行以下命令：

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

在 Windows Server 上确定版本

登录到实例并运行以下命令：

```
sc qdescription codedeployagent
```

安装或重新安装 AWS CodeDeploy 代理

如果您怀疑 AWS CodeDeploy 代理缺失或不起作用，您可以在实例上运行命令来安装或重新安装它。

Important

每个 Amazon EC2 实例必须附加一个 IAM 实例配置文件，该文件有权访问 Amazon S3 存储桶，其中包含您的区域的代理安装文件。如果没有 IAM 实例配置文件提供的权限，实例将无法下载 AWS CodeDeploy 代理安装文件。有关信息，请参阅 [步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件](#) (p. 22) 和 [配置用于 AWS CodeDeploy 的 Amazon EC2 实例](#) (p. 153)。

主题

- [安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理](#) (p. 121)
- [安装或重新安装适用于 Ubuntu Server 的 AWS CodeDeploy 代理](#) (p. 122)
- [安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理](#) (p. 123)

安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理

登录到实例，并依次运行以下命令。

Note

在第四个命令中，`/home/ec2-user` 代表 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
```

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

```
wget https://bucket-name.s3.amazonaws.com/latest/install
```

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

```
sudo ./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 `s3-latest-bucket-name` 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅 [各区域的资源工具包存储桶名称](#) (p. 302)。

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

如果您看到类似于 `error: No AWS CodeDeploy agent running` 的消息，请启动该服务并依次运行以下两个命令：

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

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

安装或重新安装适用于 Ubuntu Server 的 AWS CodeDeploy 代理

登录到实例，并依次运行以下命令。

Note

在第五个命令中，`/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。

```
sudo apt-get update
```

对于 Ubuntu Server 14.04：

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

对于 Ubuntu Server 16.04：

- ```
sudo apt-get install ruby
```

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

```
cd /home/ubuntu
```

```
wget https://bucket-name.s3.amazonaws.com/latest/install
```

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

```
sudo ./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 `aws-codedeploy-us-east-2`。有关存储桶名称的列表，请参阅 [各区域的资源工具包存储桶名称](#) (p. 302)。

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

如果您看到类似于 error: No AWS CodeDeploy agent running 的消息，请启动该服务并依次运行以下两个命令：

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

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

安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理

在 Windows Server 实例上，您可以使用下列方法之一来下载并安装 AWS CodeDeploy 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

在新实例和现有实例上，我们建议安装 Windows Server 的 AWS CodeDeploy 代理更新程序。更新程序会定期检查代理的新版本，当有新版本可用时，会安装该新版本。在新实例上，您可以安装更新程序而不是代理，在安装更新程序后会立即安装代理的最新版本。有关更多信息，请参阅 [更新 Windows Server 上的 AWS CodeDeploy 代理](#) (p. 128)。

主题

- [使用 Windows PowerShell](#) (p. 124)
- [使用直接链接](#) (p. 124)
- [使用 Amazon S3 复制命令](#) (p. 126)

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 适用于 Windows PowerShell 的 AWS 工具。

```
Import-Module AWSPowerShell
```

3. 创建目录，将 AWS CodeDeploy 代理安装文件下载到其中。

```
New-Item -Path "c:\temp" -ItemType "directory" -Force
```

4. 下载 AWS CodeDeploy 代理安装文件。

```
powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
```

5. 运行 AWS CodeDeploy 代理安装文件。

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

要检查服务是否正在运行，请运行以下命令：

```
powershell.exe -Command Get-Service -Name codedeployagent
```

如果刚刚安装了 AWS CodeDeploy 代理并且未启动，则在运行 Get-Service 命令之后，在状态下应看到# #...：

Status	Name	DisplayName
Start...	codedeployagent	CodeDeploy Host Agent Service

如果 AWS CodeDeploy 代理已经运行，在运行 Get-Service 命令之后，在状态 下，您应看到####：

Status	Name	DisplayName
Running	codedeployagent	CodeDeploy Host Agent Service

使用直接链接

如果 Windows Server 实例上的浏览器安全设置提供权限 (例如，对 http://*.s3.amazonaws.com 的权限)，则您可以使用区域的直接链接来下载 AWS CodeDeploy 代理，然后手动运行安装程序。

区域名称	下载链接
美国东部 (俄亥俄州)	https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/codedeploy-agent.msi
美国东部 (弗吉尼亚北部)	https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/codedeploy-agent.msi
美国西部 (加利福尼亚北部)	https://aws-codedeploy-us-west-1.s3.amazonaws.com/latest/codedeploy-agent.msi
美国西部 (俄勒冈)	https://aws-codedeploy-us-west-2.s3.amazonaws.com/latest/codedeploy-agent.msi
加拿大 (中部)	https://aws-codedeploy-ca-central-1.s3.amazonaws.com/latest/codedeploy-agent.msi
欧洲 (爱尔兰)	https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/codedeploy-agent.msi
欧洲 (伦敦)	https://aws-codedeploy-eu-west-2.s3.amazonaws.com/latest/codedeploy-agent.msi
欧洲 (巴黎)	https://aws-codedeploy-eu-west-3.s3.amazonaws.com/latest/codedeploy-agent.msi
欧洲 (法兰克福)	https://aws-codedeploy-eu-central-1.s3.amazonaws.com/latest/codedeploy-agent.msi
亚太区域 (东京)	https://aws-codedeploy-ap-northeast-1.s3.amazonaws.com/latest/codedeploy-agent.msi
亚太区域 (首尔)	https://aws-codedeploy-ap-northeast-2.s3.amazonaws.com/latest/codedeploy-agent.msi
亚太区域 (新加坡)	https://aws-codedeploy-ap-southeast-1.s3.amazonaws.com/latest/codedeploy-agent.msi
亚太区域 (悉尼)	https://aws-codedeploy-ap-southeast-2.s3.amazonaws.com/latest/codedeploy-agent.msi
亚太地区 (孟买)	https://aws-codedeploy-ap-south-1.s3.amazonaws.com/latest/codedeploy-agent.msi

区域名称	下载链接
南美洲 (圣保罗)	https://aws-codedeploy-sa-east-1.s3.amazonaws.com/latest/codedeploy-agent.msi

使用 Amazon S3 复制命令

如果在实例上安装 AWS CLI，则可使用 Amazon S3 `cp` 命令下载 AWS 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 AWS Command Line Interface](#)。

区域名称	Amazon S3 复制命令
美国东部 (俄亥俄州)	<pre>aws s3 cp s3://aws-codedeploy-us-east-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
美国东部 (弗吉尼亚北部)	<pre>aws s3 cp s3://aws-codedeploy-us-east-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
美国西部 (加利福尼亚北部)	<pre>aws s3 cp s3://aws-codedeploy-us-west-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
美国西部 (俄勒冈)	<pre>aws s3 cp s3://aws-codedeploy-us-west-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
加拿大 (中部)	<pre>aws s3 cp s3://aws-codedeploy-ca-central-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
欧洲 (爱尔兰)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
欧洲 (伦敦)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
欧洲 (巴黎)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-3/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
欧洲 (法兰克福)	<pre>aws s3 cp s3://aws-codedeploy-eu-central-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>

区域名称	Amazon S3 复制命令
亚太区域 (东京)	<pre>aws s3 cp s3://aws-codedeploy-ap-northeast-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
亚太区域 (首尔)	<pre>aws s3 cp s3://aws-codedeploy-ap-northeast-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
亚太区域 (新加坡)	<pre>aws s3 cp s3://aws-codedeploy-ap-southeast-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
亚太区域 (悉尼)	<pre>aws s3 cp s3://aws-codedeploy-ap-southeast-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
亚太地区 (孟买)	<pre>aws s3 cp s3://aws-codedeploy-ap-south-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
南美洲 (圣保罗)	<pre>aws s3 cp s3://aws-codedeploy-sa-east-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>

更新 AWS CodeDeploy 代理

对于 Amazon Linux、RHEL 和 Ubuntu Server 操作系统，AWS CodeDeploy 代理将在发布新版本时自动更新。对于 Windows Server，您可以在安装 AWS CodeDeploy 代理后安装 Windows Server 的 AWS CodeDeploy 代理更新程序，或者使用更新程序来取代该代理。只要检测到新版本，就会更新代理。您也可以通过在实例上运行命令来在所有受支持的操作系统上进行强制更新。

主题

- [更新 Amazon Linux 或 RHEL 上的 AWS CodeDeploy 代理 \(p. 127\)](#)
- [更新 Ubuntu Server 上的 AWS CodeDeploy 代理 \(p. 128\)](#)
- [更新 Windows Server 上的 AWS CodeDeploy 代理 \(p. 128\)](#)

更新 Amazon Linux 或 RHEL 上的 AWS CodeDeploy 代理

在实例上安装 AWS CodeDeploy 代理 (codedeploy-agent.noarch.rpm) 后，它会在新版本发布 24 小时内自动更新。更新时间无法轻易取消或重新安排。如果在更新期间某个部署正在进行，则当前部署生命周期事件将首先完成。更新完成后，部署将继续下一个部署生命周期事件。

如果您要强制更新 AWS CodeDeploy 代理，请登录到实例，并运行以下命令：

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


更新 Ubuntu Server 上的 AWS CodeDeploy 代理

在实例上安装 AWS CodeDeploy 代理 (codedeploy-agent_all.deb) 后，它会在新版本发布 24 小时内自动更新。更新时间无法轻易取消或重新安排。如果在更新期间某个部署正在进行，则当前部署生命周期事件将首先完成。更新完成后，部署将继续下一个部署生命周期事件。

如果您要强制更新 AWS CodeDeploy 代理，请登录到实例，并运行以下命令：

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

更新 Windows Server 上的 AWS CodeDeploy 代理

要启用只要发布新版本就自动更新 AWS CodeDeploy 代理的功能，请在新实例或现有实例上安装 Windows Server 的 AWS CodeDeploy 代理更新程序。更新程序定期检查新版本。当检测到新版本时，更新程序将在安装最新版本之前，卸载当前版本的代理 (如果已安装)。

如果在更新程序检测到新版本时部署操作已在进行中，则部署操作将会继续完成。如果尝试在更新过程中启动部署操作，则部署操作将失败。

如果您希望强制更新 AWS CodeDeploy 代理，请按照[安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理 \(p. 123\)](#)中的说明操作。

在 Windows Server 实例上，您可以通过以下方式下载并安装 AWS CodeDeploy 代理更新程序：运行一系列 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 \(p. 128\)](#)
- [使用直接链接 \(p. 129\)](#)
- [使用 Amazon S3 复制命令 \(p. 130\)](#)

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
```

```
powershell.exe -Command Read-S3Object -BucketName bucket-name -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
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 **bucket-name** 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅 [各区域的资源工具包存储桶名称](#) (p. 302)。

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 AWS CodeDeploy 代理更新程序日志文件：

```
notepad C:\ProgramData\Amazon\CodeDeployUpdater\log\codedeploy-agent.updater.log
```

使用直接链接

如果 Windows Server 实例上的浏览器安全设置提供必要权限 (例如，对 http://*.s3.amazonaws.com 的权限)，您可以使用直接链接下载 AWS CodeDeploy 代理更新程序，方式是在浏览器的地址栏中输入以下内容，然后下载并手动运行安装程序。

区域名称	下载链接
美国东部（俄亥俄州）	https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
美国东部（弗吉尼亚北部）	https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
美国西部（加利福尼亚北部）	https://aws-codedeploy-us-west-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
美国西部（俄勒冈）	https://aws-codedeploy-us-west-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
加拿大（中部）	https://aws-codedeploy-ca-central-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
欧洲（爱尔兰）	https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
欧洲（伦敦）	https://aws-codedeploy-eu-west-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
欧洲（巴黎）	https://aws-codedeploy-eu-west-3.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
欧洲（法兰克福）	https://aws-codedeploy-eu-central-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
亚太区域（东京）	https://aws-codedeploy-ap-northeast-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
亚太区域（首尔）	https://aws-codedeploy-ap-northeast-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi

区域名称	下载链接
亚太区域 (新加坡)	https://aws-codedeploy-ap-southeast-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
亚太区域 (悉尼)	https://aws-codedeploy-ap-southeast-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
亚太地区 (孟买)	https://aws-codedeploy-ap-south-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
南美洲 (圣保罗)	https://aws-codedeploy-sa-east-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi

使用 Amazon S3 复制命令

如果实例上已安装 AWS CLI，则可使用 Amazon S3 `cp` 命令下载 AWS 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 AWS Command Line Interface](#)。

区域名称	Amazon S3 复制命令
美国东部 (俄亥俄州)	<pre>aws s3 cp s3://aws-codedeploy-us-east-2/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
美国东部 (弗吉尼亚北部)	<pre>aws s3 cp s3://aws-codedeploy-us-east-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
美国西部 (加利福尼亚北部)	<pre>aws s3 cp s3://aws-codedeploy-us-west-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
美国西部 (俄勒冈)	<pre>aws s3 cp s3://aws-codedeploy-us-west-2/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
加拿大 (中部)	<pre>aws s3 cp s3://aws-codedeploy-ca-central-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
欧洲 (爱尔兰)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
欧洲 (伦敦)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-2/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>

区域名称	Amazon S3 复制命令
欧洲 (巴黎)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-3/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
欧洲 (法兰克福)	<pre>aws s3 cp s3://aws-codedeploy-eu-central-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
亚太区域 (东京)	<pre>aws s3 cp s3://aws-codedeploy-ap- northeast-1/latest/codedeploy-agent- updater.msi codedeploy-agent-updater.msi</pre>
亚太区域 (首尔)	<pre>aws s3 cp s3://aws-codedeploy-ap- northeast-2/latest/codedeploy-agent- updater.msi codedeploy-agent-updater.msi</pre>
亚太区域 (新加坡)	<pre>aws s3 cp s3://aws-codedeploy-ap- southeast-1/latest/codedeploy-agent- updater.msi codedeploy-agent-updater.msi</pre>
亚太区域 (悉尼)	<pre>aws s3 cp s3://aws-codedeploy-ap- southeast-2/latest/codedeploy-agent- updater.msi codedeploy-agent-updater.msi</pre>
亚太地区 (孟买)	<pre>aws s3 cp s3://aws-codedeploy-ap-south-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
南美洲 (圣保罗)	<pre>aws s3 cp s3://aws-codedeploy-sa-east-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>

卸载 AWS CodeDeploy 代理

当您不再需要 AWS CodeDeploy 代理或者希望执行全新安装时，可以从实例中移除该代理。

从 Amazon Linux 或 RHEL 卸载 AWS CodeDeploy 代理

要卸载 AWS CodeDeploy 代理，请登录到实例并运行以下命令：

```
sudo yum erase codedeploy-agent
```

从 Ubuntu Server 卸载 AWS CodeDeploy 代理

要卸载 AWS CodeDeploy 代理，请登录到实例并运行以下命令：

```
sudo dpkg --purge codedeploy-agent
```

从 Windows Server 卸载 AWS CodeDeploy 代理

要卸载 AWS 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
```

或者，登录到实例，在 Control Panel 中，打开 Programs and Features，选择 CodeDeploy Host Agent，然后选择 Uninstall。

使用适用于 AWS CodeDeploy 的实例

AWS CodeDeploy 支持针对运行 Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL) 和 Windows Server 的实例的部署。

您可以使用 AWS CodeDeploy 部署至 Amazon EC2 实例和本地实例。本地实例是可运行 AWS CodeDeploy 代理并连接到公有 AWS 服务终端节点的 Amazon EC2 实例以外的任何物理设备。使用 AWS CodeDeploy，您可以将一个应用程序同时部署到云中运行的 Amazon EC2 实例，以及办公室中运行的桌面 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：

主题	Amazon EC2 实例	本地实例
需要您安装并运行可与实例上运行的操作系统兼容的 AWS CodeDeploy 代理的版本。	是	是
需要实例能够连接到 AWS CodeDeploy 服务。	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 AWS CodeDeploy 部署。有关信息，请参阅 步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 (p. 22)。	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： <ul style="list-style-type: none">为每个实例创建一个 IAM 用户，并以纯文本形式将该 IAM 用户的账户凭证存储在实例上。创建一个可让 IAM 用户在每个实例上都能担任的 IAM 角色，以获取通过 AWS Security Token Service 生成的定期刷新的临时凭证。	否	是
需要您先向 AWS 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 AWS CodeDeploy 才能对其进行部署。	是	是
作为 AWS CodeDeploy 部署的一部分，可参与 Auto Scaling 和 Elastic Load Balancing 方案。	是	否

主题	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

AWS CodeDeploy 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：

我需要启动新的 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 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CloudFormation 模板) (p. 147) 。 要主要靠自己手动启动 Amazon EC2 实例，请参阅 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CLI 或 Amazon EC2 控制台) (p. 141) 。
我需要启动新的 Ubuntu Server 或 RHEL Amazon EC2 实例。	请参阅 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CLI 或 Amazon EC2 控制台) (p. 141) 。
我需要配置 Amazon Linux、Windows Server、Ubuntu Server 或 RHEL Amazon EC2 实例。	请参阅 配置用于 AWS CodeDeploy 的 Amazon EC2 实例 (p. 153) 。
我需要配置 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. 156) 。
我希望 AWS CodeDeploy 在蓝/绿部署期间预置实例队列。	请参阅 在 AWS CodeDeploy 中使用部署 (p. 220) 。

要在 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 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。

主题

- [Tagging Instances for AWS CodeDeploy Deployments \(p. 134\)](#)
- [Working with Amazon EC2 Instances \(p. 141\)](#)
- [Working with On-Premises Instances \(p. 156\)](#)
- [View Instance Details \(p. 179\)](#)
- [Instance Health \(p. 180\)](#)

在 AWS CodeDeploy 中为部署组标记实例

可以使用标签为每个资源分配您自己的元数据，这有助于管理您的 Amazon EC2 实例和本地实例。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[标记您的 Amazon EC2 资源](#)。

要指定 AWS CodeDeploy 部署组中包含哪些实例，需要在一个或多个标签组中指定标签。针对部署组创建部署时，满足标签条件的实例上会安装最新的应用修订。

Note

您还可以在部署组中包含 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 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。

部署组中最简单的实例条件可以是唯一标签组中的唯一标签，复杂的条件可以在最多三个标签组中每组添加 10 个标签。

如果使用唯一标签组，该组中至少一个标签标记的实例即会包括在部署组中。如果使用多个标签组，只有由每个标签组中至少一个标签标记的实例才会包括在内。

以下示例说明如何使用标签和标签组为部署组选择实例。

主题

- [示例 1：唯一标签组，唯一标签 \(p. 135\)](#)
- [示例 2：唯一标签组，多个标签 \(p. 136\)](#)
- [示例 3：多个标签组，单个标签 \(p. 137\)](#)
- [示例 4：多个标签组，多个标签 \(p. 139\)](#)

示例 1：唯一标签组，唯一标签

您可以在唯一标签组中指定唯一标签：

标签组 1

键	值
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。

AWS CodeDeploy 控制台设置视图：

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name	AppVersion-ABC	7	✕
2				✕

Add tag group

JSON 结构：

```
"ec2TagFilters": [
  {
    "Type": "KEY_AND_VALUE",
    "Key": "Name",
    "Value": "AppVersion-ABC"
  },
],
```

示例 2：唯一标签组，多个标签

您也可以在唯一标签组中指定多个标签：

标签组 1

键	值
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。

AWS CodeDeploy 控制台设置视图：

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Region	North	4	✕
2	Region	South	2	✕
3	Region	East	2	✕
4				✕

[Add tag group](#)

JSON 结构：

```
"ec2TagFilters": [
  {
    "Type": "KEY_AND_VALUE",
    "Key": "Region",
    "Value": "North"
  },
],
```

```
    },
    {
      "Type": "KEY_AND_VALUE",
      "Key": "Region",
      "Value": "South"
    },
    {
      "Type": "KEY_AND_VALUE",
      "Key": "Region",
      "Value": "East"
    }
  ],
```

示例 3：多个标签组，单个标签

您还可以使用多组标签组，每个标签组中只有唯一的键值对，指定部署组中实例的条件。如果在部署组中使用多个标签组，只有所有标签组均标记出的实例才会包含在部署组中。

标签组 1

键	值
名称	AppVersion-ABC

标签组 2

键	值
区域	北部

标签组 3

键	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，但在此示例中，部署组中只包含也具有 Region=North 和 Type=t2.medium 标签的实例。

AWS CodeDeploy 控制台设置视图：

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name	AppVersion-ABC	7	✕
2				✕

Tag group 2

	Key	Value	Instances	
1	Region	North	4	✕
2				✕

✖ Remove tag group

Tag group 3

	Key	Value	Instances	
1	Type	t2.medium	3	✕
2				✕

✖ Remove tag group

JSON 结构：

```
"ec2TagSet": {
  "ec2TagSetList": [
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Name",
        "Value": "AppVersion-ABC"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Region",
        "Value": "North"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "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，才能包含在部署组中。

标签组 1

键	值
Environment	测试版
Environment	生产前调试

标签组 2

键	值
区域	北部
区域	南部
区域	东部

标签组 3

键	值
类型	t2.medium
类型	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、Region=**West**、Type=t2.medium
- Environment=Staging、Region=East、Type=**t2.micro**
- Environment=**Production**、Region=South、Type=t2.large

AWS CodeDeploy 控制台设置视图：

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Environment	Staging	4	✕
2	Environment	Beta	6	✕
3				✕

Tag group 2

	Key	Value	Instances	
1	Region	South	2	✕
2	Region	North	2	✕
3	Region	East	2	✕
4				✕

✖ Remove tag group

Tag group 3

	Key	Value	Instances	
1	Type	t2.medium	3	✕
2	Type	t2.large	4	✕
3				✕

✖ Remove tag group

JSON 结构：

```
"ec2TagSet": {
  "ec2TagSetList": [
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Environment",
        "Value": "Beta"
      },
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Environment",
        "Value": "Staging"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Region",
        "Value": "North"
      }
    ]
  ]
}
```

```
{
  "Key": "KEY_AND_VALUE",
  "Type": "Region",
  "Value": "South"
},
{
  "Key": "KEY_AND_VALUE",
  "Type": "Region",
  "Value": "East"
}
],
[
  {
    "Key": "KEY_AND_VALUE",
    "Type": "Type",
    "Value": "t2.medium"
  },
  {
    "Key": "KEY_AND_VALUE",
    "Type": "Type",
    "Value": "t2.large"
  }
],
]
```

使用适用于 AWS CodeDeploy 的 Amazon EC2 实例

Amazon EC2 实例是一种虚拟计算环境，可使用 Amazon Elastic Compute Cloud 服务创建并配置。Amazon EC2 在 Amazon Web Services (AWS) 云中提供可扩展的计算容量。您可以根据需要，使用 Amazon EC2 为 AWS CodeDeploy 部署启动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](#)。

本部分说明了如何创建和配置用于 AWS CodeDeploy 部署的 Amazon EC2 实例。

主题

- [为 AWS CodeDeploy 创建 Amazon EC2 实例 \(AWS CLI 或 Amazon EC2 控制台\) \(p. 141\)](#)
- [为 AWS CodeDeploy 创建 Amazon EC2 实例 \(AWS CloudFormation 模板\) \(p. 147\)](#)
- [配置用于 AWS CodeDeploy 的 Amazon EC2 实例 \(p. 153\)](#)

为 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CLI 或 Amazon EC2 控制台)

这些说明向您介绍如何启动配置为在 AWS CodeDeploy 部署中使用的新 Amazon EC2 实例。

您可使用 AWS CloudFormation 模板启动已配置为在 AWS CodeDeploy 部署中使用的运行 Amazon Linux 或 Windows Server 的 Amazon EC2 实例。我们未为运行 Ubuntu Server 或 Red Hat Enterprise Linux (RHEL) 的 Amazon EC2 实例提供 AWS CloudFormation 模板。有关模板使用的替代方法，请参阅[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)。

您可使用 Amazon EC2 控制台、AWS CLI 或 Amazon EC2 API 启动 Amazon EC2 实例。

启动 Amazon EC2 实例 (控制台)

如果您尚未这样做，请按照[AWS CodeDeploy 入门 \(p. 16\)](#)中的说明进行操作来设置和配置 AWS CLI 并创建 IAM 实例配置文件。

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon EC2 控制台 <https://console.aws.amazon.com/ec2/>。
2. 在导航窗格中，选择 Instances，然后选择 Launch Instance。
3. 在 Step 1: Choose an Amazon Machine Image (AMI) 页上，从 Quick Start 选项卡中，找到要使用的操作系统和版本，然后选择 Select。
4. 在 Step 2: Choose an Instance Type 页上，选择任何可用的 Amazon EC2 实例类型，然后选择 Next: Configure Instance Details。
5. 在步骤 3：配置实例详细信息页上的 IAM 角色列表中，选择您在[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)中创建的 IAM 实例角色。如果您使用建议的角色名称，则选择 **CodeDeployDemo-EC2-Instance-Profile**。如果您创建了自己的角色名称，请选择该名称。

Note

如果 Launch into EC2-Classical 和默认 Virtual Private Cloud (VPC) 均未显示在 Network 列表中，并且您无法选择支持启动到 EC2-Classical 的其他 Amazon EC2 实例类型，则必须选择 Amazon VPC 和子网，或选择 Create new VPC 和/或 Create new subnet。有关更多信息，请[参阅您的 VPC 和子网](#)。

6. 展开 Advanced Details (高级详细信息)。
7. 在 User data 旁，在选中 As text 选项的情况下，键入以下内容以在 Amazon EC2 实例启动时安装 AWS CodeDeploy 代理。

对于 Amazon Linux 和 RHEL

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请[参阅各区域的资源工具包存储桶名称 \(p. 302\)](#)。

对于 Ubuntu Server

Important

如果您正在 Ubuntu Server 14.04 上安装 AWS CodeDeploy 代理，请将第三行更改为以下内容：

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

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请[参阅各区域的资源工具包存储桶名称 \(p. 302\)](#)。

对于 Windows Server

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle Hidden
</powershell>
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

8. 将此页上的其他项保持不变，然后选择 Next: Add Storage。
9. 将 Step 4: Add Storage 页保持不变，然后选择 Next: Add Tags。
10. 在步骤 5：添加标记页面上，选择添加标记。
11. 在 Key 框中，键入 **Name**。在值框中，键入 **CodeDeployDemo**。

Important

Key 和 Value 框的内容是区分大小写的。

12. 选择 Next: Configure Security Group。
13. 在 Step 6: Configure Security Group 页上，将 Create a new security group 选项保持选中状态。

将为运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例配置默认 SSH 角色。将为运行 Windows Server 的 Amazon EC2 实例配置默认 RDP 角色。

14. 如果您需要打开 HTTP 端口，请选择 Add Rule 按钮，然后从 Type 下拉列表中，选择 **HTTP**。接受 Anywhere 0.0.0.0/0 的默认 Source 值，然后选择 Review and Launch。

Note

在生产环境中，建议您限制对 SSH、RDP 和 HTTP 端口的访问，而不是指定 Anywhere 0.0.0.0/0。AWS CodeDeploy 不需要无限制的端口访问，也不需要 HTTP 访问。有关更多信息，请参阅[有关保护您的 Amazon EC2 实例的提示](#)。

如果 Boot from General Purpose (SSD) 对话框出现，请遵循说明，然后选择 Next。

15. 将 Step 7: Review Instance Launch 页保持不变，然后选择 Launch。
16. 在 Select an existing key pair or create a new key pair 对话框中，选择 Choose an existing key pair 或 Create a new key pair。如果您已配置 Amazon EC2 实例密钥对，则可在此处选择它。

如果您尚不拥有 Amazon EC2 实例密钥对，请选择 Create a new key pair 并为密钥对指定一个可识别的名称。选择 Download Key Pair 以将 Amazon EC2 实例密钥对下载到您的计算机。

Important

如果您需要使用 SSH 或 RDP 访问您的 Amazon EC2 实例，则必须具有密钥对。

17. 选择 Launch Instances。
18. 选择 Amazon EC2 实例的 ID。在实例启动并通过所有检查之前，请不要继续。

要验证 AWS CodeDeploy 代理是否正在实例上运行，请参阅[验证 AWS CodeDeploy 代理是否正在运行](#) (p. 119)，然后返回到此页。在执行此操作后，Amazon EC2 实例便能在 AWS CodeDeploy 部署中使用。

启动 Amazon EC2 实例 (CLI)

如果您尚未这样做，请按照[AWS CodeDeploy 入门 \(p. 16\)](#)中的说明进行操作来设置和配置 AWS CLI 并创建 IAM 实例配置文件。

1. 仅对于 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，创建允许 RDP 访问 (默认情况下不允许) 或者 HTTP 访问的安全组。例如，要创建名为 CodeDeployDemo-Windows-Security-Group 的安全组，请逐条运行以下命令：

```
aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-Group --description "For launching Windows Server images for use with AWS CodeDeploy"
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 3389
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 80
```

Note

在演示中，这些命令将创建一个安全组来允许通过端口 3389 的 RDP 无限制访问或通过端口 80 的 HTTP 无限制访问。作为最佳实践，建议您限制对 RDP 和 HTTP 端口的访问。AWS CodeDeploy 不需要无限制的端口访问，也不需要 HTTP 访问。有关更多信息，请参阅[有关保护您的 Amazon EC2 实例的提示](#)。

2. 在您的开发计算机上，创建一个包含以下内容的名为 instance-setup.sh (对于运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例) 或 instance-setup.txt (对于运行 Windows Server 的 Amazon EC2 实例) 的文件。

当 Amazon EC2 实例启动时，此脚本将从指定的 Amazon S3 位置下载 AWS CodeDeploy 代理，然后将它安装在实例上。

对于 Amazon Linux 和 RHEL

以下是 instance-setup.sh 的内容，适用于 Amazon Linux 和 RHEL：

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部 (俄亥俄) 区域，将 **bucket-name** 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称 \(p. 302\)](#)。

对于 Ubuntu Server

以下是适用于 Ubuntu Server 的 instance-setup.sh 的内容：

Important

如果您正在 Ubuntu Server 14.04 上安装 AWS CodeDeploy 代理，请将文件的第三行更改为以下内容：

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

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

对于 Windows Server

以下是适用于 Windows Server 的 instance-setup.txt 的内容：

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle Hidden
</powershell>
```

bucket-name 是包含适用于您所在区域的 AWS CodeDeploy 资源工具包文件的 Amazon S3 sds-s3-latest-bucket-name 存储桶的名称。例如，对于 美国东部（俄亥俄）区域，将 *bucket-name* 替换为 aws-codedeploy-us-east-2。有关存储桶名称的列表，请参阅[各区域的资源工具包存储桶名称](#) (p. 302)。

3. 从您创建 instance-setup.sh 或 instance-setup.txt 文件的同一目录中，调用 run-instances 命令来创建和启动 Amazon EC2 实例。

在您调用此命令之前，您需要收集以下内容：

- 将用于此实例的 Amazon 系统映像 (AMI) 的 ID (*ami-id*)。要获取此 ID，请参阅[查找合适的 AMI](#)。
- 您将创建的 Amazon EC2 实例的类型 (*instance-type*) 的名称，如 t1.micro。有关列表，请参阅[Amazon EC2 实例类型](#)。
- IAM 实例配置文件的名称，该文件有权访问 Amazon S3 存储桶，其中存储着您的区域的 AWS CodeDeploy 代理安装文件。

有关创建 IAM 实例配置文件的信息，请参阅[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件](#) (p. 22)。

- Amazon EC2 实例密钥对的名称 (*key-name*)，此密钥对支持对运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例的 SSH 访问或对运行 Windows Server 的 Amazon EC2 实例的 RDP 访问。

Important

仅键入密钥对名称而不是密钥对文件扩展名。例如，my-keypair，而不是 my-keypair.pem。

要查找密钥对名称，请在以下位置打开 Amazon EC2 控制台：<https://console.aws.amazon.com/ec2>。在导航窗格中，在 Network & Security 下，选择 Key Pairs，然后记下列表中的密钥对名称。

要生成密钥对，请参阅[使用 Amazon EC2 创建您的密钥对](#)。请确保在 AWS General Reference 的[区域和终端节点](#)中列出的某个区域中创建密钥对。否则，您无法将 Amazon EC2 实例密钥对与 AWS CodeDeploy 结合使用。

对于 Amazon Linux、RHEL 和 Ubuntu Server

要调用 run-instances 命令以启动运行 Amazon Linux、Ubuntu Server 或 RHEL 的 Amazon EC2 实例并附加您在[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)中创建的 IAM 实例配置文件，请执行以下命令。例如：

Important

Be sure to include file:// before the file name. It is required in this command.

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --user-data file://instance-setup.sh \
  --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，建议您仅限制对 SSH 和 HTTP 端口的访问。AWS 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[有关保护您的 Amazon EC2 实例的提示](#)。

对于 Windows Server

调用 run-instances 命令，启动运行 Windows Server 的 Amazon EC2 实例，附加您在[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)中创建的 IAM 实例配置文件，并指定您在步骤 1 中创建的安全组的名称。例如：

Important

Be sure to include file:// before the file name. It is required in this command.

```
aws ec2 run-instances --image-id ami-id --key-name key-name --user-data file://
instance-setup.txt --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、密钥对和实例类型的 Amazon EC2 实例，并在启动过程中运行指定脚本。

- 记下输出中的 InstanceID 的值。如果您忘记此值，可稍后通过对 Amazon EC2 实例密钥对调用 describe-instances 命令来获取它。

```
aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
"Reservations[*].Instances[*].[InstanceId]" --output text
```

使用实例 ID 调用 `create-tags` 命令，这将标记 Amazon EC2 实例，以便 AWS CodeDeploy 在部署过程中找到它。在以下示例中，标签名为 **CodeDeployDemo**，但您可指定所需的任何 Amazon EC2 实例标签。

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=CodeDeployDemo
```

您可以为一个实例同时应用多个标签。例如：

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance  
Key=Region,Value=West Key=Environment,Value=Beta
```

要验证 Amazon EC2 实例是否已启动并通过所有检查，请使用实例 ID 调用 `describe-instance-status` 命令。

```
aws ec2 describe-instance-status --instance-ids instance-id --query  
"InstanceStatuses[*].InstanceStatus.[Status]" --output text
```

如果实例已启动并通过所有检查，输出中将显示 `ok`：

要验证 AWS CodeDeploy 代理是否正在实例上运行，请参阅[验证 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)，然后返回到此页。在执行此操作后，Amazon EC2 实例便能在 AWS CodeDeploy 部署中使用。

为 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CloudFormation 模板)

您可使用 AWS CloudFormation 模板快速启动运行 Amazon Linux 或 Windows Server 的 Amazon EC2 实例。您可使用 AWS CLI、AWS CodeDeploy 控制台或 AWS API 通过模板启动实例。除了启动实例之外，模板还可用于：

- 指示 AWS CloudFormation 为实例提供参与 AWS CodeDeploy 部署的权限。
- 标记实例，以便 AWS CodeDeploy 可在部署过程中找到它。
- 安装 AWS CodeDeploy 代理并在实例上运行它。

您无需使用 AWS CloudFormation 即可设置 Amazon EC2 实例。有关替代方法，请参阅[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)。

我们未为运行 Ubuntu Server 或 Red Hat Enterprise Linux (RHEL) 的 Amazon EC2 实例提供 AWS CloudFormation 模板。

Important

如果您使用 AWS CloudFormation 模板启动 Amazon EC2 实例，则调用的 IAM 用户必须具有对 AWS CloudFormation 以及 AWS CloudFormation 所依赖的 AWS 服务和操作的访问权限。如果您尚未执行[步骤 1：预置 IAM 用户 \(p. 16\)](#)中的步骤来预置调用的 IAM 用户，则必须至少附加以下策略：

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



```
        "codedeploy:*",
        "ec2:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam>DeleteInstanceProfile",
        "iam>DeleteRole",
        "iam>DeleteRolePolicy",
        "iam:GetRole",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile"
    ],
    "Resource": "*"
}
}
```

主题

- [使用 AWS CloudFormation 模板启动 Amazon EC2 实例 \(控制台\)](#) (p. 148)
- [使用 AWS CloudFormation 模板启动 Amazon EC2 实例 \(AWS CLI\)](#) (p. 151)

使用 AWS CloudFormation 模板启动 Amazon EC2 实例 (控制台)

在开始之前，您必须具有实例密钥对才能允许针对运行 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。仅键入密钥对名称而不是密钥对文件扩展名。

要查找密钥对名称，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 下，选择 Key Pairs，然后记下列表中的密钥对名称。

要生成新的密钥对，请参阅[使用 Amazon EC2 创建您的密钥对](#)。请确保在 AWS General Reference 的[区域和终端节点](#)中列出的某个区域中创建密钥对。否则，您无法将实例密钥对与 AWS CodeDeploy 结合使用。

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台：<https://console.aws.amazon.com/cloudformation>。

Important

使用您在[AWS CodeDeploy 入门](#) (p. 16)中使用的同一账户登录 AWS 管理控制台。在导航栏上的区域选择器中，选择 AWS General Reference 的[区域和终端节点](#)中列出的某个区域。AWS CodeDeploy 仅支持这些区域。

2. 选择 Create Stack。
3. 在 Choose a template 中，选择 Specify an Amazon S3 template URL。在框中键入您所在区域的 AWS CloudFormation 模板的位置，然后选择 Next。

Region	Location of AWS CloudFormation template
美国东部 (俄亥俄) 区域	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/templates/latest/CodeDeploy_SampleCF_Template.json
美国东部 (弗吉尼亚北部) 地区	http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
美国西部 (加利福尼亚北部) 区域	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
美国西部 (俄勒冈) 区域	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
加拿大 (中部) 区域	http://s3-ca- central-1.amazonaws.com/ aws-codedeploy-ca- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (爱尔兰) 区域	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (伦敦) 区域	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu- west-3/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (法兰克福) 区域	http://s3-eu- central-1.amazonaws.com/ aws-codedeploy-eu- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (东京)	http://s3-ap- northeast-1.amazonaws.com/ aws-codedeploy-ap- northeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (首尔)	http://s3-ap- northeast-2.amazonaws.com/ aws-codedeploy-ap- northeast-2/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (新加坡)	http://s3-ap- southeast-1.amazonaws.com/ aws-codedeploy-ap- southeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
亚太区域 (悉尼)	http://s3-ap-southeast-2.amazonaws.com/ aws-codedeploy-ap-southeast-2/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太地区 (孟买) 区域	http://s3-ap-south-1.amazonaws.com/ aws-codedeploy-ap-south-1/templates/latest/ CodeDeploy_SampleCF_Template.json
南美洲 (圣保罗) 区域	http://s3-sa-east-1.amazonaws.com/ aws-codedeploy-sa-east-1/templates/latest/ CodeDeploy_SampleCF_Template.json

- 在 Stack name 框中，键入堆栈的名称 (例如，**CodeDeployDemoStack**)。
- 在 Parameters 中，键入以下内容，然后选择 Next。
 - 对于 InstanceCount，键入要启动的实例的数量。(建议您保留默认值 1。)
 - 对于 InstanceType，键入要启动的实例类型 (或保留默认值 t1.micro)。
 - 对于 KeyPairName，键入实例密钥名称。
 - 对于 OperatingSystem 框，键入 **Windows** 以启动运行 Windows Server 的实例 (或保留默认值 Linux)。
 - 对于 SSHLocation，键入要用于使用 SSH 或 RDP 连接到实例的 IP 地址范围 (或保留默认值 0.0.0.0/0)。

Important

提供默认值 **0.0.0.0/0** 仅为演示之用。AWS 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。有关更多信息，请参阅[有关保护您的 Amazon EC2 实例的提示](#)。

- 对于 TagKey，键入 AWS 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 Name)。
 - 对于 TagValue，键入 AWS 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)。
- 在 Options 页上，将选项框留空，然后选择 Next。

Important

AWS CloudFormation 标签不同于 AWS CodeDeploy 标签。AWS CloudFormation 使用标签来简化对基础设施的管理。AWS CodeDeploy 使用标签来标识 Amazon EC2 实例。您已在 Specify Parameters 页上指定 AWS CodeDeploy 标签。

- 在 Review 页上的 Capabilities 中，选中 I acknowledge that AWS CloudFormation might create IAM resources 框，然后选择 Create。

在 AWS CloudFormation 创建堆栈并启动 Amazon EC2 实例之后，在 AWS CloudFormation 控制台中，CREATE_COMPLETE 将显示在 Status 列中。此过程可能耗时数分钟。

要验证 AWS CodeDeploy 代理是否正在 Amazon EC2 实例上运行，请参阅[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)，然后继续使用 [AWS CodeDeploy 创建应用程序 \(p. 189\)](#)。

使用 AWS CloudFormation 模板启动 Amazon EC2 实例 (AWS CLI)

按照[AWS CodeDeploy 入门 \(p. 16\)](#)中的说明操作可安装和配置用于 AWS CodeDeploy 的 AWS CLI。

在调用 create-stack 命令之前，您必须具有 Amazon EC2 实例密钥对才能允许针对运行 Amazon Linux 的 Amazon EC2 实例的 SSH 访问或针对运行 Windows Server 的 Amazon EC2 实例的 RDP 访问。仅键入密钥对名称而不是密钥对文件扩展名。

要查找密钥对名称，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 下，选择 Key Pairs，然后记下列表中的密钥对名称。

要生成密钥对，请参阅[使用 Amazon EC2 创建您的密钥对](#)。请务必在 AWS General Reference 的[区域和终端节点](#)中列出的某个区域中创建密钥对。否则，您无法将实例密钥对与 AWS CodeDeploy 结合使用。

1. 在调用 create-stack 命令时使用 AWS CloudFormation 模板。此堆栈将使用安装的 AWS CodeDeploy 代理启动新的 Amazon EC2 实例。

要启动运行 Amazon Linux 的 Amazon EC2 实例，请执行以下命令：

```
aws cloudformation create-stack \  
  --stack-name CodeDeployDemoStack \  
  --template-url templateURL \  
  --parameters ParameterKey=InstanceCount,ParameterValue=1 \  
  ParameterKey=InstanceType,ParameterValue=t1.micro \  
  ParameterKey=KeyPairName,ParameterValue=keyName \  
  ParameterKey=OperatingSystem,ParameterValue=Linux \  
  ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0 \  
  ParameterKey=TagKey,ParameterValue=Name \  
  ParameterKey=TagValue,ParameterValue=CodeDeployDemo \  
  --capabilities CAPABILITY_IAM
```

要启动运行 Windows Server 的 Amazon EC2 实例，请执行以下命令：

```
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
```

template-url 是您所在区域的 AWS CloudFormation 模板的位置：

Region	Location of AWS CloudFormation template
美国东部 (俄亥俄) 区域	http://s3-us-east-2.amazonaws.com/ aws-codedeploy-us- east-2/templates/latest/ CodeDeploy_SampleCF_Template.json
美国东部 (弗吉尼亚北部) 地区	http://s3.amazonaws.com/ aws-codedeploy-us- east-1/templates/latest/ CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
美国西部 (加利福尼亚北部) 区域	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
美国西部 (俄勒冈) 区域	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
加拿大 (中部) 区域	http://s3-ca- central-1.amazonaws.com/ aws-codedeploy-ca- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (爱尔兰) 区域	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (伦敦) 区域	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu- west-3/templates/latest/ CodeDeploy_SampleCF_Template.json
欧洲 (法兰克福) 区域	http://s3-eu- central-1.amazonaws.com/ aws-codedeploy-eu- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (东京)	http://s3-ap- northeast-1.amazonaws.com/ aws-codedeploy-ap- northeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (首尔)	http://s3-ap- northeast-2.amazonaws.com/ aws-codedeploy-ap- northeast-2/templates/latest/ CodeDeploy_SampleCF_Template.json
亚太区域 (新加坡)	http://s3-ap- southeast-1.amazonaws.com/ aws-codedeploy-ap- southeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
亚太区域 (悉尼)	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
亚太地区 (孟买) 区域	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json
南美洲 (圣保罗) 区域	http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/templates/latest/CodeDeploy_SampleCF_Template.json

此命令使用 AWS CloudFormation 模板在指定的 Amazon S3 存储桶中创建一个名为 **CodeDeployDemoStack** 的 AWS CloudFormation 堆栈。虽然 Amazon EC2 实例基于 t1.micro 实例类型，但您可使用任何类型。虽然它是使用值 **CodeDeployDemo** 标记的，但您可使用任何值标记它。它已应用指定的实例密钥对。

2. 调用 `describe-stacks` 命令可验证是否已成功创建名为 **CodeDeployDemoStack** 的 AWS CloudFormation 堆栈：

```
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 AWS CodeDeploy 代理是否正在 Amazon EC2 实例上运行，请参阅[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)，然后继续使用[AWS CodeDeploy 创建应用程序 \(p. 189\)](#)。

配置用于 AWS CodeDeploy 的 Amazon EC2 实例

下面的说明显示如何配置运行 Amazon Linux、Ubuntu Server、Red Hat Enterprise Linux (RHEL) 或 Windows Server 的 Amazon EC2 实例供在 AWS CodeDeploy 部署中使用。

Note

如果您没有 Amazon EC2 实例，可以使用 AWS CloudFormation 模板启动一个运行 Amazon Linux 或 Windows Server 的实例。我们不提供适用于 Ubuntu Server 或 RHEL 的模板。

执行本主题中的步骤：

- 有权参与 AWS CodeDeploy 部署的 IAM 实例配置文件必须附加到实例。

有关如何在创建 Amazon EC2 实例时附加 IAM 实例配置文件的信息，请参阅[为 AWS CodeDeploy 创建 Amazon EC2 实例 \(AWS CLI 或 Amazon EC2 控制台\) \(p. 141\)](#)和[为 AWS CodeDeploy 创建 Amazon EC2 实例 \(AWS CloudFormation 模板\) \(p. 147\)](#)。

有关如何向现有 Amazon EC2 实例附加 IAM 实例配置文件的信息，请参阅[将 IAM 角色附加到实例](#)。

- 必须已标记 Amazon EC2 实例。
- Amazon EC2 实例上必须已安装且正在运行 AWS CodeDeploy 代理。

如果该代理未在运行，则部署将显示为停滞在挂起状态。

步骤 1：验证 IAM 实例配置文件是否已附加到 Amazon EC2 实例

1. 登录 AWS 管理控制台并通过以下网址打开 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 Amazon EC2 实例。
4. 在详细信息窗格中的 Description 选项卡上，记下 IAM role 字段中的值，然后继续下一部分。

如果该字段为空，您可以向实例附加 IAM 实例配置文件。有关信息，请参阅[将 IAM 角色附加到实例](#)。

步骤 2：验证附加的 IAM 实例配置文件是否具有正确的访问权限

1. 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。
3. 浏览并选择您在前一部分的步骤 4 中记下的 IAM 角色名称。

Note

如果您希望使用由 AWS CloudFormation 模板生成的服务角色，而不使用您通过按照[步骤 3：为 AWS CodeDeploy 创建服务角色](#) (p. 18) 中的说明创建的服务角色，则请注意以下事项：在一些版本的 AWS CloudFormation 模板中，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，您会看到二者的前缀 CodeDeploySampleStack-expnyi6-InstanceRole 是相同的。有关这些显示名称可能不同的原因的信息，请参阅[实例配置文件](#)。

4. 选择 Trust Relationships 选项卡。如果可信任的实体中没有显示 The identity provider(s) ec2.amazonaws.com 的条目，则您无法使用此 Amazon EC2 实例。使用[使用适用于 AWS CodeDeploy 的实例](#) (p. 133) 中的信息停止并创建 Amazon EC2 实例。

如果有显示 The identity provider(s) ec2.amazonaws.com 的条目，并且您只需要将应用程序存储在 GitHub 存储库中，请向前跳到[步骤 3：标记 Amazon EC2 实例](#) (p. 155)。

如果有显示 The identity provider(s) ec2.amazonaws.com 的条目，并且您会将应用程序存储在 Amazon S3 存储桶中，请选择权限选项卡。

5. 如果 Managed Policies 区域中有一个策略，则选择该策略的名称，然后选择 Edit。如果 Inline Policies 中有一个策略，则在 Actions 下选择 Edit Policy。
6. 如果您将应用程序存储在 Amazon S3 存储桶中，请在 Policy Document 框中，确保 "s3:Get*" 和 "s3:List*" 位于指定操作列表中。

它可能如下所示：

```
{
  "Statement": [
    {
      "Resource": "*",
      "Action": [
        ... Some actions may already be listed here ...
        "s3:Get*",
        "s3:List*"
        ... Some more actions may already be listed here ...
      ],
      "Effect": "Allow"
    }
  ]
}
```

也可能如下所示：

```
{
  "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，以便策略文档将进行验证。）

Note

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

```
{
  "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-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

步骤 3：标记 Amazon EC2 实例

有关如何标记 Amazon EC2 实例以便 AWS CodeDeploy 可以在部署期间找到它的说明，请参阅[在控制台中使用标签](#)，然后返回到此页。

Note

您可以使用任何所需的密钥和值标记 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。

步骤 4：在 Amazon EC2 实例上安装 AWS CodeDeploy 代理

有关如何在 Amazon EC2 实例上安装 AWS CodeDeploy 代理并验证其是否运行的说明，请参阅[管理 AWS CodeDeploy 代理操作](#) (p. 118)，然后继续使用 [AWS CodeDeploy 创建应用程序](#) (p. 189)。

使用适用于 AWS CodeDeploy 的本地实例

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

将 AWS 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 AWS 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 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \)](#) (p. 79)。有关本地实例以及如何将它们与 AWS CodeDeploy 一起使用的信息，请参阅[Working with On-Premises Instances](#) (p. 156)。

如果您不想再在部署中使用本地实例，只需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 [AWS CodeDeploy 中管理本地实例操作](#) (p. 175)。

此部分中的说明向您演示如何配置本地实例，然后使用 AWS 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 AWS CodeDeploy 来获取有关本地实例的信息，以及如何在您不再计划部署到本地实例时注销该实例。

主题

- [配置本地实例的先决条件](#) (p. 156)
- [将本地实例注册到 AWS CodeDeploy](#) (p. 157)
- [在 AWS CodeDeploy 中管理本地实例操作](#) (p. 175)

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 [register-on-premises-instance](#) 命令和 AWS Security Token Service (AWS STS) 生成的定期刷新临时凭证，则还需要满足其他一些条件。有关信息，请参阅 [IAM 会话 ARN 注册前提条件](#) (p. 170)。

设备要求

您要使用 AWS 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[AWS CodeDeploy 代理支持的操作系统](#) (p. 113)。

如果您的操作系统不受支持，则 AWS CodeDeploy 代理可以作为开源系统来适应您的需求。有关更多信息，请参阅 GitHub 中的 [AWS CodeDeploy 代理](#) 存储库。

出站通信

本地实例必须能够连接到公有 AWS 服务终端节点来与 AWS CodeDeploy 通信。

AWS 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账户或网络账户必须能够以 `sudo` 或 `root`（对于 Ubuntu Server）的身份或者以管理员的身份（对于 Windows Server）运行。

IAM 权限

您在注册本地实例时使用的 IAM 身份必须具有完成注册的权限（以及在需要时注销本地实例的权限）。

除了 [AWS CodeDeploy 入门](#) (p. 16) 中所述的策略之外，请确保发出调用的 IAM 身份还另外挂载了以下策略：

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

将本地实例注册到 AWS CodeDeploy

要注册本地实例，必须使用 IAM 身份来对您的请求进行身份验证。您可以针对您使用的 IAM 身份和注册方法从以下选项中进行选择：

- 使用 IAM 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 [使用注册命令 \(IAM 用户 ARN\) 注册本地实例](#) (p. 158)。
 - 使用 [register-on-premises-instance](#) 命令手动设置大多数注册选项。适合注册少量内部实例的情况。有关信息，请参阅 [使用 register-on-premises-instance 命令 \(IAM 用户 ARN\) 注册本地实例](#) (p. 161)。
- 使用 IAM 角色 ARN 对请求进行身份验证。
 - 使用 [register-on-premises-instance](#) 命令与通过 AWS Security Token Service (AWS STS) 生成的定期刷新临时凭证，手动设置大多数注册选项。最适合注册大量本地实例的情况。有关信息，请参阅 [使用 register-on-premises-instance 命令 \(IAM 会话 ARN\) 注册本地实例](#) (p. 169)。

主题

- [使用注册命令 \(IAM 用户 ARN\) 注册本地实例 \(p. 158\)](#)
- [使用 register-on-premises-instance 命令 \(IAM 用户 ARN\) 注册本地实例 \(p. 161\)](#)
- [使用 register-on-premises-instance 命令 \(IAM 会话 ARN\) 注册本地实例 \(p. 169\)](#)

使用注册命令 (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 AWS CodeDeploy 配置本地实例并注册和标记该实例。当您处理一个或少量的本地实例时，register 命令非常有用。仅当您使用 IAM 用户 ARN 对实例进行身份验证时，才能使用 register 命令。使用 IAM 会话 ARN 进行身份验证时，不能使用 register 命令。

在使用 register 命令时，您可以让 AWS CodeDeploy 执行以下操作：

- 如果您未随该命令指定 IAM 用户，则在 AWS Identity and Access Management 中为本地实例创建一个。
- 将该 IAM 用户的凭证保存到本地实例配置文件中。
- 向 AWS 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

[register-on-premises-instance](#) 命令是 [register](#) 命令的替代命令。如果您希望主要靠自己使用 AWS CodeDeploy 配置本地实例并注册和标记该实例，请使用 register-on-premises-instance 命令。您也可以通过 register-on-premises-instance 命令来使用 IAM 会话 ARN 注册实例，而不是使用 IAM 用户 ARN 进行注册。如果您有大量本地实例，则此方法有很大的优势。具体来说，您可以使用一个 IAM 会话 ARN 对多个实例进行身份验证，而不必为每个本地实例都单独创建一个 IAM 用户。有关更多信息，请参阅 [使用 register-on-premises-instance 命令 \(IAM 用户 ARN\) 注册本地实例 \(p. 161\)](#) 和 [使用 register-on-premises-instance 命令 \(IAM 会话 ARN\) 注册本地实例 \(p. 169\)](#)。

主题

- [步骤 1：在本地实例上安装和配置 AWS CLI \(p. 158\)](#)
- [步骤 2：调用注册命令 \(p. 159\)](#)
- [步骤 3：调用安装命令 \(p. 160\)](#)
- [步骤 4：将应用程序修订部署到本地实例 \(p. 161\)](#)
- [步骤 5：跟踪对本地实例的部署 \(p. 161\)](#)

步骤 1：在本地实例上安装和配置 AWS CLI

1. 在本地实例上安装 AWS CLI。按照 AWS 命令行界面用户指南中[使用 AWS 命令行界面进行设置](#)的说明执行操作。

Note

可用于本地实例的 AWS CodeDeploy 命令在 AWS CLI 版本 1.7.19 和更高版本中提供。如果您已安装了 AWS CLI，请调用 `aws --version` 来检查其版本。

2. 在本地实例上配置 AWS CLI。按照 AWS 命令行界面用户指南中的[配置 AWS 命令行界面](#)的说明执行操作。

Important

在您配置 AWS CLI（例如，通过调用 `aws configure` 命令）时，请确保指定 IAM 用户的私有密钥 ID 和秘密访问密钥，并且除了在[配置本地实例的先决条件 \(p. 156\)](#)中指定的权限之外，该用户还必须至少具有以下 AWS 访问权限。这使得在本地实例上下载和安装 AWS 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:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

步骤 2：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 AWS 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 AWS CLI 调用 `register` 命令，在命令中指定：

- 向 AWS 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 AWS 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[AWS CodeDeploy 限制](#) (p. 307)。

- (可选) 您希望与此本地实例关联的现有 IAM 用户的 ARN (使用 `--iam-user-arn` 选项)。要获取 IAM 用户的 ARN，请调用 `get-user` 命令，或者在 IAM 控制台的 Users 部分中选择 IAM 用户名，然后在 Summary 部分中查找 User ARN 值。如果未指定此选项，则 AWS CodeDeploy 将代表您在您的 AWS 账户中创建一个 IAM 用户，并将其与本地实例进行关联。

Important

如果您指定了 `--iam-user-arn` 选项，则还必须手动创建本地实例配置文件，如[步骤 4：将配置文件添加到本地实例](#) (p. 165)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)，AWS CodeDeploy 将使用这些标签来标识要部署到的 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 `add-tags-to-on-premises-instances` 命令。
- (可选) 将在其中向 AWS CodeDeploy 注册此本地实例的 AWS 区域 (使用 `--region` 选项)。这必须是 AWS General Reference 的[区域和终端节点](#)中列出的支持区域之一 (例如 `us-west-2`)。如果未指定此选项，则将使用与发出调用的 IAM 用户关联的默认 AWS 区域。

例如：

```
aws deploy register --instance-name AssetTag12010298EX --iam-user-arn
arn:aws:iam::80398EXAMPLE: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秘密访问密钥。本地实例将使用此 IAM 用户及其权限和凭证对 AWS CodeDeploy 进行身份验证和交互。
2. 将本地实例注册到 AWS CodeDeploy。
3. 如果指定，则在 AWS CodeDeploy 中将使用 `--tags` 选项指定的标签关联到注册的本地实例名称。
4. 如果创建了 IAM 用户，则还会在调用 `register` 命令的同一个目录中创建必需的配置文件。

如果此命令遇到错误，则将显示错误消息，说明您可以如何手动完成剩余步骤。否则将显示成功消息，说明如何调用 `install` 命令，如接下来的步骤中所列。

步骤 3：调用安装命令

从本地实例，使用 AWS 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) 将在其中向 AWS CodeDeploy 注册此本地实例的 AWS 区域 (使用 `--region` 选项)。这必须是 AWS General Reference 的[区域和终端节点](#)中列出的支持区域之一 (例如 `us-west-2`)。如果未指定此选项，则将使用与发出调用的 IAM 用户关联的默认 AWS 区域。
- (可选) 从中安装 AWS CodeDeploy 代理的自定义位置 (使用 `--agent-installer` 选项)。此选项对于安装非 AWS CodeDeploy 官方支持的自定义 AWS CodeDeploy 代理版本非常有用 (例如基于 GitHub 中[AWS CodeDeploy 代理](#)存储库的自定义版本)。该值必须是指向包含以下二者之一的 Amazon S3 存储桶的路径：

- AWS CodeDeploy 代理安装脚本 (对于基于 Linux 或 Unix 的操作系统，类似于 GitHub 中的 [AWS CodeDeploy 代理](#) 存储库中的安装文件)。
- AWS 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，AWS CodeDeploy 将尽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 AWS 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 AWS 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)，为 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。

对于 Windows Server，为 `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 AWS CodeDeploy 代理，然后启动它。

步骤 4：将应用程序修订部署到本地实例

现在，您已准备好将应用程序修订部署到已注册和标记的本地实例。

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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。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 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux\)](#) (p. 79) 的步骤 2：创建示例应用程序修订 (p. 80) 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 AWS CodeDeploy 服务角色，则必须将 `Tag:get*` 包含到服务角色策略语句的 `Action` 部分中。有关更多信息，请参阅 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)。

步骤 5：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

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 [使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)。

有关更多选项，请参阅在 [AWS CodeDeploy 中管理本地实例操作 \(p. 175\)](#)。

使用 register-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 AWS CodeDeploy 注册该实例并为其添加标签，同时使用静态 IAM 用户凭证进行身份验证。

主题

- [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#)
- [步骤 2：向 IAM 用户分配权限 \(p. 162\)](#)
- [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#)
- [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#)
- [步骤 5：安装和配置 AWS CLI \(p. 166\)](#)
- [步骤 6：设置 AWS_REGION 环境变量 \(仅针对 Ubuntu Server 和 RHEL \) \(p. 167\)](#)
- [步骤 7：安装 AWS CodeDeploy 代理 \(p. 167\)](#)
- [步骤 8：将本地实例注册到 AWS CodeDeploy \(p. 168\)](#)
- [步骤 9：标记本地实例 \(p. 168\)](#)
- [步骤 10：将应用程序修订部署到本地实例 \(p. 169\)](#)
- [步骤 11：跟踪对本地实例的部署 \(p. 169\)](#)

步骤 1：为本地实例创建 IAM 用户

创建本地实例用来与 AWS CodeDeploy 进行身份验证和交互的 IAM 用户。

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 AWS 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，例如 CodeDeployUser-OnPrem。

您可以使用 AWS CLI 或 IAM 控制台创建 IAM 用户。有关信息，请参阅 [在您的 AWS 账户中创建 IAM 用户](#)。

Important

无论您使用 AWS CLI 还是 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 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 和 [步骤 8：将本地实例注册到 AWS CodeDeploy \(p. 168\)](#) 中您将需要此信息。

步骤 2：向 IAM 用户分配权限

如果您的本地实例将从 Amazon S3 存储桶部署应用程序修订，则您必须向 IAM 用户分配权限来与这些存储桶交互。您可以使用 AWS 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 [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#)。(您仍将需要有关先前在 [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 中创建的 IAM 用户的信息。后面的步骤中将用到此信息。)

分配权限 (CLI)

1. 在您用于调用 AWS CLI 的 Amazon EC2 实例或设备上创建包含以下策略内容的文件。采用类似于 **CodeDeploy-OnPrem-Permissions.json** 的格式对文件命名，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，请确保还向包含 AWS CodeDeploy 代理的 Amazon S3 存储桶提供了访问权限。否则，只要在关联的本地实例上安装或更新 AWS 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-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}

```

2. 调用 `put-user-policy` 命令，在命令中指定 IAM 用户的名称 (使用 `--user-name` 选项)、策略的名称 (使用 `--policy-name` 选项) 和新创建策略文档的路径 (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

Be sure to include `file://` before the file name. It is required in this command.

```
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 框中，键入此策略的名称（例如，**CodeDeploy-OnPrem-Permissions**）。
5. 在 Policy Document 框中，键入或粘贴以下权限表达式，这允许 AWS 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[步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#)中创建的 IAM 用户的名称。
9. 在 Permissions 选项卡上的 Managed Policies 中，选择 Attach Policy。
10. 选择名为 **CodeDeploy-OnPrem-Permissions** 的策略，然后选择 Attach Policy。

步骤 3：获取 IAM 用户凭证

获取 IAM 用户的私有密钥 ID 和秘密访问密钥。您在[步骤 4：将配置文件添加到本地实例 \(p. 165\)](#)中需要使用它们。您可以使用 AWS CLI 或 IAM 控制台获取私有密钥 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[步骤 4：将配置文件添加到本地实例 \(p. 165\)](#)。

获取凭证 (CLI)

1. 调用 `list-access-keys` 命令，在命令中指定 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 text` 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，则调用 `create-access-key` 命令，在命令中指定 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[步骤 4：将配置文件添加到本地实例 \(p. 165\)](#)中您将需要此信息。

Important

这是您仅有的一次查看此秘密访问密钥的机会。如果您忘记或丢失了此秘密访问密钥，则需要按照[步骤 3：获取 IAM 用户凭证 \(p. 164\)](#)中的步骤生成新的密钥。

3. 如果已经列出了两个访问密钥，则您必须删除其中之一，方法是调用 `delete-access-key` 命令，并在命令中指定 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 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 中所述使用此访问密钥，则您需要再次按照 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对该本地实例的任何部署可能会停滞在永久等待的状态或完全失败。

获取凭证（控制台）

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
 2. 如果用户的列表未显示，则在导航窗格中选择 Users。
 3. 在用户列表中，浏览并选择在 [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 中创建的 IAM 用户的名称。
2. 在 Security credentials 选项卡上，如果没有列出密钥或仅列出了一个密钥，请选择 Create access key。

如果列出了两个访问密钥，则必须删除其中之一。选择其中一个访问密钥旁边的 Delete，然后选择 Create access key。

Important

如果您选择这些访问密钥之一旁边的 Delete，并且本地实例已经按 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 中所述使用此访问密钥，则您需要再次按照 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对该本地实例的部署可能会停滞在永久等待的状态或完全失败。

3. 选择 Show 并记录访问密钥 ID 和秘密访问密钥。您在下一步中需要此信息。或者，您可以选择 Download .csv file 来保存访问密钥 ID 和秘密访问密钥的副本。

Important

除非您记录或下载了凭证，否则这是您仅有的一次查看此秘密访问密钥的机会。如果您忘记或丢失了此秘密访问密钥，则需要按照 [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#) 中的步骤生成新的密钥。

4. 选择 Close 返回 Users > **IAM User Name** 页面。

步骤 4：将配置文件添加到本地实例

使用 root 或管理员权限将配置文件添加到本地实例。此配置文件用于声明将为 AWS CodeDeploy 使用的 IAM 用户凭证和目标 AWS 区域。该文件必须添加到本地实例上的特定位置。文件必须包括 IAM 用户的 ARN、私有密钥 ID、秘密访问密钥和目标 AWS 区域。该文件必须遵循特定格式。

1. 在本地实例上的以下位置，创建名为 `codedeploy.onpremises.yml` (用于 Ubuntu Server 或 RHEL 本地实例) 或 `conf.onpremises.yml` 的文件 (用于 Windows Server 本地实例)：

- 对于 Ubuntu Server：`/etc/codedeploy-agent/conf`
- 对于 Windows Server：`C:\ProgramData\Amazon\CodeDeploy`

2. 使用文本编辑器将以下信息添加到新创建的 `codedeploy.onpremises.yml` 或 `conf.onpremises.yml` 文件：

```
---
aws_access_key_id: secret-key-id
aws_secret_access_key: secret-access-key
iam_user_arn: iam-user-arn
region: supported-region
```

其中：

- *secret-key-id* 是您在 [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 或 [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#) 中记录的对应 IAM 用户的私有密钥 ID。
- *secret-access-key* 是您在 [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 或 [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#) 中记录的对应 IAM 用户的秘密访问密钥。
- *iam-user-arn* 是您之前在 [步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 中记录的 IAM 用户的 ARN。
- *supported-region* 是 AWS CodeDeploy 支持的区域的标识符，您的 AWS 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 AWS General Reference 中的 [区域和终端节点](#)。

Important

如果您在 [步骤 3：获取 IAM 用户凭证 \(p. 164\)](#) 中选择了其中一个访问密钥旁边的 Delete，并且您的本地实例已经使用关联的访问密钥 ID 和秘密访问密钥，那么就需要按照 [步骤 4：将配置文件添加到本地实例 \(p. 165\)](#) 中的说明操作，指定与此 IAM 用户关联的其他访问密钥 ID 和秘密访问密钥。否则，对您的本地实例的任何部署可能会停滞在永久等待的状态或完全失败。

步骤 5：安装和配置 AWS CLI

在本地实例上安装和配置 AWS CLI（AWS CLI 将在 [步骤 7：安装 AWS CodeDeploy 代理 \(p. 167\)](#) 中用于在本地实例上下载和安装 AWS CodeDeploy 代理。）

1. 要在本地实例上安装 AWS CLI，请按照 AWS 命令行界面用户指南的 [使用 AWS 命令行界面进行设置](#) 中的说明执行操作。

Note

1.7.19 版 AWS CLI 中提供了用于处理本地实例的 AWS CodeDeploy 命令。如果您已经安装了 AWS CLI 版本，则可以调用 `aws --version` 来检查其版本。

2. 要在本地实例上配置 AWS CLI，请按照 AWS 命令行界面用户指南的 [配置 AWS 命令行界面](#) 中的说明执行操作。

Important

在您配置 AWS CLI（例如，通过调用 `aws configure` 命令）时，请确保指定 IAM 用户的私有密钥 ID 和秘密访问密钥，并且除了在 [配置本地实例的先决条件 \(p. 156\)](#) 中指定的访问权限之外，该用户还必须至少具有以下 AWS 访问权限。这使您能够在本地实例上下载和安装 AWS CodeDeploy 代理：

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

```
    ],
    "Resource" : "*"
  },
  {
    "Effect" : "Allow",
    "Action" : [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource" : [
      "arn:aws:s3:::aws-codedeploy-us-east-2/*",
      "arn:aws:s3:::aws-codedeploy-us-east-1/*",
      "arn:aws:s3:::aws-codedeploy-us-west-1/*",
      "arn:aws:s3:::aws-codedeploy-us-west-2/*",
      "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
      "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
      "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
      "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
      "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
    ]
  }
]
```

这些访问权限可以分配到您在 [步骤 1：为本地实例创建 IAM 用户](#) (p. 162) 中创建的 IAM 用户或者其他 IAM 用户。要将这些权限分配给 IAM 用户，请按照 [步骤 1：为本地实例创建 IAM 用户](#) (p. 162) 中的说明操作，并使用这些访问权限而不是该步骤中的权限。

步骤 6：设置 AWS_REGION 环境变量（仅针对 Ubuntu Server 和 RHEL）

如果您的本地实例上没有运行 Ubuntu Server 或 RHEL，则跳过此步骤，直接转到 [步骤 7：安装 AWS CodeDeploy 代理](#) (p. 167)。

在 Ubuntu Server 或 RHEL 本地实例上安装 AWS CodeDeploy 代理，并让实例能够在发布新版本时及时更新 AWS CodeDeploy 代理。为此，您可以将实例上的 AWS_REGION 环境变量设置为 AWS 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您的 AWS 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 AWS General Reference 中的 [区域和终端节点](#)。

要设置环境变量，请从终端调用以下命令：

```
export AWS_REGION=supported-region
```

其中 **supported-region** 是区域标识符（例如，us-west-2）。

步骤 7：安装 AWS CodeDeploy 代理

在本地实例上安装 AWS CodeDeploy 代理：

- 对于 Ubuntu Server 本地实例，请按照 [安装或重新安装适用于 Ubuntu Server 的 AWS CodeDeploy 代理](#) (p. 122) 中的说明操作，然后返回本页。
- 对于 RHEL 本地实例，请按照 [安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理](#) (p. 121) 中的说明操作，然后返回本页。

- 对于 Windows Server 本地实例，请按照[安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理 \(p. 123\)](#) 中的说明操作，然后返回本页。

步骤 8：将本地实例注册到 AWS CodeDeploy

这一步中的说明假设您从本地实例本身上注册该本地实例。您还可以从安装和配置了 AWS CLI 的其他设备或实例上注册本地实例，如[步骤 5：安装和配置 AWS CLI \(p. 166\)](#)中所述。

使用 AWS CLI 将本地实例注册到 AWS CodeDeploy，以便可以在部署中使用该实例。

1. 使用 AWS CLI 之前，您需要在[步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 中创建的 IAM 用户的用户 ARN。如果您还没有用户 ARN，请调用 `get-user` 命令，在命令中指定 IAM 用户的名称 (使用 `--user-name` 选项) 并仅查询用户 ARN (使用 `--query` 和 `--output` 选项)：

```
aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --output text
```

2. 调用 `register-on-premises-instance` 命令，在命令中指定：

- 唯一标识本地实例的名称 (使用 `--instance-name` 选项)。

Important

为了帮助标识本地实例，特别是用于调试用途，我们强烈建议您指定能够反映本地实例的某种唯一特性的名称 (例如，在适用时可以指定序列号或内部资产标识符)。如果您指定 MAC 地址作为名称，请注意 MAC 地址包含 AWS 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[AWS CodeDeploy 限制 \(p. 307\)](#)。

- 您在[步骤 1：为本地实例创建 IAM 用户 \(p. 162\)](#) 中创建的 IAM 用户的用户 ARN (使用 `--iam-user-arn` 选项)。

例如：

```
aws deploy register-on-premises-instance --instance-name AssetTag12010298EX --iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployUser-OnPrem
```

步骤 9：标记本地实例

您可以使用 AWS CLI 或 AWS CodeDeploy 控制台标记本地实例。(在部署期间，AWS 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 `add-tags-to-on-premises-instances` 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)。您必须同时指定名称和值。AWS 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. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 从 AWS CodeDeploy 菜单中，选择 On-premises instances。
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，选择或键入所需的标签密钥和标签值。当您键入标签密钥和标签值之后，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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。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 AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux\) \(p. 79\)](#) 的步骤 2：创建示例应用程序修订 (p. 80) 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 AWS CodeDeploy 服务角色，则必须将 `Tag:get*` 包含到服务角色策略语句的 Action 部分中。有关更多信息，请参阅 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)。

步骤 11：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

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 [使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)。

使用 register-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，可使用 [register-on-premises-instance](#) 命令和 AWS Security Token Service (AWS STS) 所生成的定期刷新的临时凭证。实例的一个静态 IAM 角色将担任这些刷新 AWS STS 凭证的角色来执行 AWS CodeDeploy 部署操作。

当您需注册大量实例时，此方法非常有用。您可以使用此方法来借助 AWS CodeDeploy 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用于 AWS 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 AWS 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 AWS STS 生成的临时安全凭证来配置本地实例。

主题

- [IAM 会话 ARN 注册前提条件 \(p. 170\)](#)

- [步骤 1：创建本地实例将担任的 IAM 角色 \(p. 170\)](#)
- [步骤 2：使用 AWS STS 为单独的实例生成临时凭证 \(p. 171\)](#)
- [步骤 3：将配置文件添加到本地实例 \(p. 172\)](#)
- [步骤 4：为 AWS CodeDeploy 部署准备本地实例 \(p. 172\)](#)
- [步骤 5：将本地实例注册到 AWS CodeDeploy \(p. 173\)](#)
- [步骤 6：标记本地实例 \(p. 174\)](#)
- [步骤 7：将应用程序修订部署到本地实例 \(p. 174\)](#)
- [步骤 8：跟踪对本地实例的部署 \(p. 175\)](#)

IAM 会话 ARN 注册前提条件

除了在 [配置本地实例的先决条件 \(p. 156\)](#) 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 AWS CodeDeploy 操作。确保 AWSCodeDeployFullAccess 托管策略已挂载到该 IAM 身份。有关信息，请参阅 IAM 用户指南中的 [AWS 托管策略](#)。

用于刷新临时凭证的系统

如果您使用 IAM 会话 ARN 注册本地实例，则您必须有一个系统来定期刷新该临时凭证。如果在生成凭证时指定的期限较短，临时凭证会在一小时 (或更短时间) 后过期。有两种刷新凭证的方法：

- 方法 1：使用您的企业网络中的身份和身份验证系统以及一个 CRON 脚本，该脚本定期轮询该身份和身份验证系统，并将最新的会话凭证复制到该实例。采用此方法，您可以将您的身份验证和身份结构与 AWS 集成，而无需更改 AWS CodeDeploy 代理或服务来为您在组织中使用的身验证类型提供支持。
- 方法 2：定期在实例上运行 CRON 作业来调用 AWS STS [AssumeRole](#) 操作，并将会话凭证写入 AWS CodeDeploy 代理可访问的文件。此方法仍需要使用 IAM 用户并将凭证复制到本地实例，但您可以对您的各个本地实例重复使用相同的 IAM 用户和凭证。

有关创建和使用 AWS STS 凭证的信息，请参阅 [AWS Security Token Service API Reference](#) 和 [使用临时安全凭证以请求对 AWS 资源的访问权限](#)。

步骤 1：创建本地实例将担任的 IAM 角色

您可以使用 AWS 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 AWS 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 AWS 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 AWS 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 AWS 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[管理您的 Amazon S3 资源的访问权限](#)。

创建 IAM 角色

1. 调用 `create-role` 命令，同时使用 `--role-name` 选项指定 IAM 角色的名称 (例如 `CodeDeployInstanceRole`)，并使用 `--assume-role-policy-document` 选项提供权限。
为此实例创建 IAM 角色时，您可以为其指定角色名称 `CodeDeployInstanceRole`，并在名为 `CodeDeployRolePolicy.json` 的文件中提供所需的权限：

```
aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policy-document
file://CodeDeployRolePolicy.json
```

2. 在调用 `create-role` 命令的输出中，记录 ARN 字段的值。例如：

```
arn:aws:iam::123456789012:role/CodeDeployInstanceRole
```

当您使用 AWS 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 IAM 用户指南 中的 [创建向 AWS 服务委托权限的角色](#)。

有关向现有角色分配权限的信息，请参阅 [AWS CLI Command Reference](#) 中的 [put-role-policy](#)。

步骤 2：使用 AWS STS 为单独的实例生成临时凭证

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。

有关对 IAM 身份授予 `sts:AssumeRole` 权限的信息，请参阅[创建向 AWS 服务委托权限的角色和 AssumeRole](#)。

有两种生成临时凭证的方法：

- 将 `assume-role` 命令用于 AWS 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[步骤 1：创建本地实例将担任的 IAM 角色 \(p. 170\)](#)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 CRON 脚本定期轮询身份和身份认证系统，并将最新会话凭证复制到实例 (用于刷新 [IAM 会话 ARN 注册前提条件 \(p. 170\)](#) 中所述的用于刷新临时凭证的方法 1)，则您可以改为使用任何受支持的 AWS SDK 来调用 [AssumeRole](#)。

- 使用 AWS 提供的工具。

`aws-codedeploy-session-helper` 工具生成 AWS 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 注册前提条件 \(p. 170\)](#) 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 AWS CodeDeploy 代理以访问此位置的凭证。

在继续操作之前，请确保您将用来定期刷新临时凭证的系统已经准备就绪。如果未刷新临时凭证，则部署到本地实例的操作将失败。有关更多信息，请参阅 [IAM 会话 ARN 注册前提条件](#) (p. 170) 中的“用于刷新临时凭证的系统”。

步骤 3：将配置文件添加到本地实例

使用 root 或管理员权限将配置文件添加到本地实例。此配置文件用于声明将对 AWS CodeDeploy 使用的 IAM 凭证和目标 AWS 区域。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 AWS 区域。

添加配置文件

1. 在本地实例上的以下位置，创建名为 `codedeploy.onpremises.yml` (用于 Ubuntu Server 或 RHEL 本地实例) 或 `conf.onpremises.yml` 的文件 (用于 Windows Server 本地实例)：
 - 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 [步骤 2：使用 AWS STS 为单独的实例生成临时凭证](#) (p. 171) 中记录的 IAM 会话私有密钥 ARN。
- `credentials-file` 是在 [步骤 2：使用 AWS STS 为单独的实例生成临时凭证](#) (p. 171) 中记录的临时会话 ARN 凭证文件的位置。
- `supported-region` 是 AWS CodeDeploy 所支持的区域之一，如 AWS General Reference 中的 [区域和终端节点](#) 中所列。

步骤 4：为 AWS CodeDeploy 部署准备本地实例

安装和配置 AWS CLI

在本地实例上安装和配置 AWS CLI。(AWS CLI 将用于在本地实例上下载和安装 AWS CodeDeploy 代理。)

1. 要在本地实例上安装 AWS CLI，请按照 AWS 命令行界面用户指南的 [使用 AWS 命令行界面进行设置](#) 中的说明执行操作。

Note

1.7.19 版 AWS CLI 中提供了用于处理本地实例的 AWS CodeDeploy 命令。如果您已经安装了 AWS CLI 版本，则可以调用 `aws --version` 来检查其版本。

2. 要在本地实例上配置 AWS CLI，请按照 AWS 命令行界面用户指南的 [配置 AWS 命令行界面](#) 中的说明执行操作。

Important

在配置 AWS CLI (例如通过调用 `aws configure` 命令) 时, 请确保指定私有密钥 ID 和 IAM 用户的秘密访问密钥, 该用户应至少具有[IAM 会话 ARN 注册前提条件 \(p. 170\)](#)中所述的权限。

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 Server 或 RHEL, 请跳过此步骤, 直接转到“安装 AWS CodeDeploy 代理”。

在 Ubuntu Server 或 RHEL 本地实例上安装 AWS CodeDeploy 代理, 并让实例能够在发布新版本时及时更新 AWS CodeDeploy 代理。为此, 您可以将实例上的 `AWS_REGION` 环境变量设置为 AWS 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您的 AWS 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 AWS General Reference 中的[区域和终端节点](#)。

要设置环境变量, 请从终端调用以下命令:

```
export AWS_REGION=supported-region
```

其中 *supported-region* 是区域标识符 (例如, `us-west-2`)。

安装 AWS CodeDeploy 代理

- 对于 Ubuntu Server 本地实例, 请按照[安装或重新安装适用于 Ubuntu Server 的 AWS CodeDeploy 代理 \(p. 122\)](#) 中的说明操作, 然后返回本页。
- 对于 RHEL 本地实例, 请按照[安装或重新安装适用于 Amazon Linux 或 RHEL 的 AWS CodeDeploy 代理 \(p. 121\)](#) 中的说明操作, 然后返回本页。
- 对于 Windows Server 本地实例, 请按照[安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理 \(p. 123\)](#) 中的说明操作, 然后返回本页。

步骤 5：将本地实例注册到 AWS CodeDeploy

这一步中的说明假设您从本地实例本身上注册该本地实例。您还可以从安装和配置了 AWS CLI 的其他设备或实例上注册本地实例。

使用 AWS CLI 将本地实例注册到 AWS CodeDeploy, 以便可以在部署中使用该实例。

您需要有您在[步骤 3：将配置文件添加到本地实例 \(p. 172\)](#)中创建的临时会话凭证的 ARN, 才能使用 AWS CLI。例如, 对于您指定为 `AssetTag12010298EX` 的实例:

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

调用 `register-on-premises-instance` 命令, 在命令中指定:

- 唯一标识本地实例的名称 (使用 `--instance-name` 选项)。

Important

为了帮助标识本地实例, 特别是用于调试用途, 我们强烈建议您指定能够反映本地实例的某种唯一特性的名称 (例如, 在适用时可以指定 STS 凭证的会话名称以及序列号或内部资产标识符)。如果您指定 MAC 地址作为名称, 请注意 MAC 地址包含 AWS 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[AWS CodeDeploy 限制 \(p. 307\)](#)。

- 您在[步骤 1：创建本地实例将担任的 IAM 角色 \(p. 170\)](#)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 [步骤 2：使用 AWS STS 为单独的实例生成临时凭证 \(p. 171\)](#) 中指定的会话角色的名称。

步骤 6：标记本地实例

您可以使用 AWS CLI 或 AWS CodeDeploy 控制台标记本地实例。(在部署期间，AWS 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 `add-tags-to-on-premises-instances` 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)。您必须同时指定名称和值。AWS 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. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 从 AWS CodeDeploy 菜单中，选择 On-premises instances。
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，选择或键入所需的标签密钥和标签值。当您键入标签密钥和标签值之后，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 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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。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 [AWS CodeDeploy 将应用程序部署到本地实例 \(Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux \)](#) (p. 79) 的步骤 2：创建示例应用程序修订 (p. 80) 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 AWS CodeDeploy 服务角色，则必须将 `Tag:get*` 包含到服务角色策略语句的 `Action` 部分中。有关更多信息，请参阅 [步骤 3：为 AWS CodeDeploy 创建服务角色](#) (p. 18)。

步骤 8：跟踪对本地实例的部署

将应用程序修订部署到已注册和标记的本地实例之后，您可以跟踪部署进度。

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 [使用 AWS CodeDeploy 查看部署详细信息](#) (p. 229)。

在 AWS CodeDeploy 中管理本地实例操作

将本地实例注册到 AWS 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. 175)
- [获取有关多个本地实例的信息](#) (p. 176)
- [从本地实例中手动删除本地实例标签](#) (p. 176)
- [自动从本地实例卸载 AWS CodeDeploy 代理并删除配置文件](#) (p. 177)
- [自动注销本地实例](#) (p. 178)
- [手动注销本地实例](#) (p. 178)

获取有关单个本地实例的信息

您可以按照 [使用 AWS CodeDeploy 查看部署详细信息](#) (p. 229) 中的说明获取有关单个本地实例的信息。您可以使用 AWS CLI 或 AWS CodeDeploy 控制台来获取有关单个本地实例的更多信息。

获取有关单个本地实例的信息 (CLI)

- 调用 `get-on-premises-instance` 命令，在命令中指定唯一标识本地实例的名称 (使用 `--instance-name` 选项)：

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

获取有关单个本地实例的信息 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单中，选择 On-premises instances。
3. 在本地实例列表中，选择本地实例旁边的箭头。此时将显示有关本地实例的详细信息。

获取有关多个本地实例的信息

您可以按照[使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)中的说明获取有关本地实例的信息。您可以使用 AWS CLI 或 AWS 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)

- 如需本地实例名称的列表，请调用 `list-on-premises-instances` 命令，在命令中指定：
 - 获取所有已注册本地实例还是所有已注销本地实例的相关信息 (分别使用 `--registration-status` 选项和 `Registered` 或 `Deregistered`)。如果忽略此项，则将同时返回已注册和已注销本地实例的名称。
 - 是否仅获取使用特定本地实例标签所标记的本地实例的相关信息 (使用 `--tag-filters` 选项)。对于每个本地实例标签，请指定 `Key`、`Value` 和 `Type` (始终应为 `KEY_AND_VALUE`)。在各 `Key`、`Value` 和 `Type` 三元组之间使用空格分隔多个本地实例标签。

例如：

```
aws deploy list-on-premises-instances --registration-status Registered --tag-filters
  Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE Key=Name,Value=CodeDeployDemo-
OnPrem-Beta,Type=KEY_AND_VALUE
```

- 有关更多详细信息，请调用 `batch-get-on-premises-instances` 命令，并提供本地实例的名称 (使用 `--instance-names` 选项)：

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
AssetTag09920444EX
```

获取有关多个本地实例的信息 (控制台)

- Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

- 从 AWS CodeDeploy 菜单中，选择 On-premises instances。此时将显示有关本地实例的信息。

从本地实例中手动删除本地实例标签

通常，当一个标签不再使用时，或者您希望从依赖该标签的任何部署组中删除本地实例时，您可以从本地实例中删除本地实例标签。您可以使用 AWS CLI 或 AWS 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。

从本地实例中手动删除本地实例标签不会注销实例。它不会从实例卸载 AWS CodeDeploy 代理，不会从实例中删除配置文件，也不会删除与该实例关联的 IAM 用户。

要自动注销本地实例，请参阅[自动注销本地实例 \(p. 178\)](#)。

要手动注销本地实例，请参阅[手动注销本地实例 \(p. 178\)](#)。

要自动卸载 AWS CodeDeploy 代理并从本地实例中删除配置文件，请参阅[自动从本地实例卸载 AWS CodeDeploy 代理并删除配置文件 \(p. 177\)](#)。

要仅手动从本地实例卸载 AWS CodeDeploy 代理，请参阅[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)。

要手动删除关联的 IAM 用户，请参阅[从您的 AWS 账户删除 IAM 用户](#)。

从本地实例删除本地实例标签 (CLI)

- 调用 `remove-tags-from-on-premises-instances`，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. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单中，选择 On-premises instances。
3. 在本地实例列表中，选择您要从中删除标签的本地实例旁边的箭头。
4. 在 Tags 区域中，选择要删除的标签旁边的删除图标 (✖)。
5. 删除标签之后，选择 Update tags。

自动从本地实例卸载 AWS CodeDeploy 代理并删除配置文件

通常，当您不再计划部署到某个本地实例之后，您可以从该实例卸载 AWS CodeDeploy 代理并删除配置文件。

Note

从本地实例自动卸载 AWS CodeDeploy 代理和删除配置文件不会注销本地实例。它不会取消任何本地实例标签与该本地实例的关联，也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 \(p. 178\)](#)。

要手动注销本地实例，请参阅[手动注销本地实例 \(p. 178\)](#)。

要手动取消任何已关联本地实例标签的关联，请参阅[从本地实例中手动删除本地实例标签 \(p. 176\)](#)。

要手动从本地实例卸载 AWS CodeDeploy 代理，请参阅[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)。

要手动删除关联的 IAM 用户，请参阅[从您的 AWS 账户删除 IAM 用户](#)。

从本地实例，使用 AWS CLI 调用 `uninstall` 命令。

例如：

```
aws deploy uninstall
```

`uninstall` 命令执行以下操作：

1. 停止在本地实例上运行的 AWS CodeDeploy 代理。
2. 从本地实例卸载 AWS CodeDeploy 代理。
3. 从本地实例删除配置文件。(对于 Ubuntu Server 和 RHEL，为 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`。对于 Windows Server，为 `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 AWS CLI 注销本地实例。

Note

您不能使用 AWS CodeDeploy 控制台注销本地实例。此外，注销本地实例不会取消任何本地实例标签与该本地实例的关联。它不会从本地实例卸载 AWS CodeDeploy 代理，也不会从本地实例中删除本地实例配置文件。

要使用 AWS 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 \(p. 178\)](#)的 AWS CodeDeploy 控制台部分。

要手动取消任何已关联本地实例标签的关联，请参阅[从本地实例中手动删除本地实例标签 \(p. 176\)](#)。

要自动卸载 AWS CodeDeploy 代理并从本地实例中删除配置文件，请参阅[自动从本地实例卸载 AWS CodeDeploy 代理并删除配置文件 \(p. 177\)](#)。

要仅手动从本地实例卸载 AWS CodeDeploy 代理，请参阅[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)。

使用 AWS CLI 调用 `deregister` 命令，在命令中指定：

- 对 AWS CodeDeploy 唯一标识本地实例的名称 (使用 `--instance-name` 选项)。
- (可选) 是否删除与本地实例关联的 IAM 用户 (默认情况下使用 `--delete-iam-user` 选项)。如果您不希望删除与本地实例关联的 IAM 用户，请指定 `--no-delete-iam-user` 选项。
- (可选) 将注册到 AWS CodeDeploy 的本地实例的 AWS 区域 (使用 `--region` 选项)。这必须是 AWS General Reference 的[区域和终端节点](#)中列出的支持区域之一（例如 `us-west-2`）。如果未指定此选项，则将使用与发出调用的 IAM 用户关联的默认 AWS 区域。

例如：

```
aws deploy deregister --instance-name AssetTag12010298EX --delete-iam-user --region us-west-2
```

`deregister` 命令执行以下操作：

1. 使用 AWS 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。

当您注销本地实例之后，在 AWS CodeDeploy 删除有关已注销本地实例的记录之前，您无法创建同名的替代本地实例或者相同的关联 IAM 用户名。这通常需要大约 24 个小时。

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 AWS CLI 手动注销本地实例。

手动注销本地实例不会卸载 AWS CodeDeploy 代理。它不会从实例中删除配置文件，不会删除与实例关联的 IAM 用户，也不会删除与实例关联的任何标签。

要自动卸载 AWS CodeDeploy 代理并从本地实例中删除配置文件，请参阅[自动从本地实例卸载 AWS CodeDeploy 代理并删除配置文件](#) (p. 177)。

要仅手动卸载 AWS CodeDeploy 代理，请参阅[管理 AWS CodeDeploy 代理操作](#) (p. 118)。

要手动删除关联的 IAM 用户，请参阅[从您的 AWS 账户删除 IAM 用户](#)。

要仅手动删除关联的本地实例标签，请参阅[从本地实例中手动删除本地实例标签](#) (p. 176)。

- 调用 `deregister-on-premises-instance` 命令，在命令中指定唯一标识本地实例的名称 (使用 `--instance-name` 选项)：

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

您注销本地实例之后，在 AWS CodeDeploy 删除有关已注销本地实例的记录之前，您无法创建同名的替代实例或者相同的关联 IAM 用户名。这通常需要大约 24 个小时。

使用 AWS CodeDeploy 查看实例详细信息

您可使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 查看部署中使用的实例的详细信息。

有关使用 AWS 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. 179)
- [查看实例详细信息 \(CLI\)](#) (p. 180)

查看实例详细信息 (控制台)

查看实例详细信息：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

3. 要显示部署详细信息，请选择与实例对应的部署 ID 旁边的箭头。
4. 在 Instances 中，选择 View all instances。
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[解决实例问题 \(p. 319\)](#)中找到问题排查提示。

6. 如果您需要查看有关 Amazon EC2 实例的更多信息，但 View in EC2 在实例详细信息页上不可用，请返回部署详细信息页，并在 Instance ID 列中，选择 Amazon EC2 实例的 ID。

查看实例详细信息 (CLI)

要使用 AWS CLI 查看实例详细信息，请调用 `get-deployment-instance` 命令或 `list-deployment-instances` 命令。

要查看有关单个实例的详细信息，请调用 `get-deployment-instance` 命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用 `list-deployments` 命令。
- 唯一实例 ID。要获取实例 ID，请调用 `list-deployment-instances` 命令。

要查看部署中使用的实例的 ID 列表，请调用 `list-deployment-instances` 命令，并指定：

- 唯一部署 ID。要获取部署 ID，请调用 `list-deployments` 命令。
- (可选) 是否仅按实例的部署状态包含特定的实例 ID。(如果未指定，则将列出所有匹配实例的 ID，不管其部署状态如何。)

AWS CodeDeploy 实例运行状况

AWS 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。AWS CodeDeploy 将使整个部署失败。

需要牢记的是，为了使整个部署成功，必须满足以下条件：

- AWS 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。

AWS 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%

默认部署配置名称	预定义的最小正常运行主机值
CodeDeployDefault.AllAtOnce	0

在 [在 AWS CodeDeploy 中使用部署配置 \(p. 184\)](#) 中，您将发现有关默认部署配置的更多信息。

您可以在 AWS CodeDeploy 中创建自定义部署配置，以定义您自己的最小正常运行主机值。在使用以下操作时，您可以将这些值定义为整数或百分比：

- 在 AWS CLI 中使用 [create-deployment-config](#) 命令时作为 `minimum-healthy-hosts`。
- 作为 AWS CodeDeploy API 的 [MinimumHealthyHosts](#) 数据类型中的 `Value`。
- 在 AWS CloudFormation 模板中使用 [AWS::CodeDeploy::DeploymentConfig](#) 时作为 `MinimumHealthyHosts`。

主题

- [运行状况 \(p. 181\)](#)
- [最小运行正常的实例数和部署数 \(p. 182\)](#)

运行状况

AWS 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。

AWS 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：

- “不正常”实例运行状况。
- “未知”修订运行状况。
- “旧”修订运行状况。
- “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：

- AWS 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。
- AWS 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和部署数

AWS 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，AWS CodeDeploy 将转换等量实例的百分比，并对实例数的小数部分取整。

AWS 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

AWS 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。

AWS 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，AWS 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。随后，AWS CodeDeploy 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 AWS 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 AWS CodeDeploy 会将应用程序修订部署到原始的运行正常的实例组。

如果针对运行正常的实例的部署失败，则 AWS CodeDeploy 会将实例的运行状况更改为“不正常”。当部署进行时，AWS 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 AWS 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 AWS CodeDeploy 将立即使整个部署失败。

AWS 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，AWS 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 AWS 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 AWS 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。

最小运行正常的实例数：95%

AWS CodeDeploy 会将最小运行正常的实例数取整为 10 个实例，这等于运行正常的实例数。如果未将修订部署到任何实例，则整个部署将立即失败。

最小运行正常的实例数：9

AWS CodeDeploy 一次将修订部署到一个实例。如果到任何实例的部署失败，运行正常的实例数将等于最小运行正常的实例数，因此 AWS CodeDeploy 会立即让整个部署失败。此规则的例外情况是，如果最后一个实例失败，部署仍将成功。

AWS CodeDeploy 将继续部署，一次部署到一个实例，直至任一部署失败或整个部署完成。如果 10 个部署全都成功，则部署组现在将具有 10 个运行正常的实例。

最小运行正常的实例数：8

AWS CodeDeploy 一次将修订部署到 2 个实例。如果这些部署中的 2 个部署失败，则 AWS CodeDeploy 将立即使整个部署失败。此规则的例外情况是，如果最后一个实例是第二个失败的实例，则部署仍将成功。

最小运行正常的实例数：0

AWS 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 AWS CodeDeploy 中使用部署配置

部署配置是 AWS CodeDeploy 在部署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 计算平台还是 AWS Lambda 计算平台，这些规则和条件会有所不同。

EC2/本地计算平台上的部署配置

当您部署到 EC2/本地 计算平台时，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 AWS 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。如果您未指定部署配置，AWS CodeDeploy 将使用 CodeDeployDefault.OneAtATime 部署配置。

有关部署期间 AWS CodeDeploy 如何监控和评估实例运行状况的信息，请参阅[Instance Health \(p. 180\)](#)。要查看已注册到 AWS 账户的部署配置列表，请参阅[View Deployment Configuration Details \(p. 187\)](#)。

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	说明
CodeDeployDefault.AllAtOnce	<p>就地部署：</p> <p>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 Succeeded。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 Failed。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。如果已向一个实例成功部署，则整个部署将成功；仅在向所有 9 个实例执行的部署都失败时，此部署才失败。</p> <p>蓝/绿部署：</p> <ul style="list-style-type: none">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	<p>就地部署：</p> <p>一次最多可部署到一半实例（小数向下取整）。如果将应用程序修订部署到至少一半实例（小数向下取整），则整个部署成功；否则部署失败。在包含 9 个实例的示例中，一次将部署到最多 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。</p>

部署配置	说明
	<p>蓝/绿部署：</p> <ul style="list-style-type: none"> 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；否则部署失败。
CodeDeployDefault.OneAtATime	<p>就地部署：</p> <p>一次仅将应用程序修订部署到一个实例。</p> <p>对于包含多个实例的部署组：</p> <ul style="list-style-type: none"> 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，AWS CodeDeploy 仅允许通过 CodeDeployDefault.OneAtATime 配置一次使一个实例脱机。 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。如果已成功部署到前 8 个实例，则整个部署将成功；如果无法部署到前 8 个实例中的任一实例，则整个部署将失败。 <p>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。</p> <p>蓝/绿部署：</p> <ul style="list-style-type: none"> 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。

AWS Lambda 计算平台上的部署配置

当您部署到 AWS Lambda 计算平台时，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。
- 线性的：流量使用相等的增量转移，在每次增量之间的分钟数相同。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。
- All-at-once：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金丝雀部署或线性部署配置。

AWS Lambda 计算平台的预定义部署配置

下表列出了可用于 AWS Lambda 部署的预定义配置。

部署配置	说明
CodeDeployDefault.LambdaCanary10Percent5Minutes	在第一次增量中转移 10% 的流量。其余 90% 部署在 5 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 30% 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 \(p. 186\)](#)
- [View Deployment Configuration Details \(p. 187\)](#)
- [Delete a Deployment Configuration \(p. 188\)](#)

使用 AWS CodeDeploy 创建部署配置

可以使用 AWS CLI、AWS CodeDeploy API 或 AWS CloudFormation 模板创建自定义部署配置。

Note

在 AWS CodeDeploy 控制台中，可以使用 Sample deployment wizard 进行就地部署，以创建自定义部署配置。

有关使用 AWS CloudFormation 模板创建部署配置的信息，请参阅 [适用于 AWS CodeDeploy 的 AWS CloudFormation 模板参考 \(p. 300\)](#)。

要使用 AWS CLI 创建部署配置，请调用 `create-deployment-config` 命令，并指定：

- 唯一标识部署配置的名称。此名称在您使用 AWS CodeDeploy 创建的与 AWS 账户关联的所有部署配置中必须唯一。

- 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. 180)。

以下示例创建一个名为 ThreeQuartersHealthy 的 EC2/本地 部署配置，此配置要求 75% 的目标实例在部署期间保持正常运行状态：

```
aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy --minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

以下示例创建一个名为 Canary25Percent45Minutes 的 AWS 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
```

使用 AWS CodeDeploy 查看部署配置详细信息

可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 查看有关与您的 AWS 账户关联的部署配置的详细信息。有关预定义的 AWS 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. 184)。

主题

- [查看部署配置详细信息 \(控制台\)](#) (p. 187)
- [查看部署配置 \(CLI\)](#) (p. 187)

查看部署配置详细信息 (控制台)

要使用 AWS CodeDeploy 控制台查看部署配置名称的列表，请执行以下操作：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单上选择 Deployment configurations，以查看部署配置名称的列表和每个部署配置的条件。

Note

如果未显示任何条目，请确保选择了正确的区域。在导航栏上的区域选择器中，选择 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

查看部署配置 (CLI)

要使用 AWS CLI 查看部署配置详细信息，请调用 `get-deployment-config` 命令或 `list-deployment-configs` 命令。

要查看有关单个部署配置的详细信息，请调用 `get-deployment-config` 命令，并指定唯一的部署配置名称。

要查看有关多个部署配置的详细信息，请调用 `list-deployments` 命令

。

使用 AWS CodeDeploy 删除部署配置

您可使用 AWS CLI 或 AWS CodeDeploy API 删除与您的 AWS 账户关联的自定义部署配置。您无法删除内置部署配置，例如 `CodeDeployDefault.AllAtOnce`、`CodeDeployDefault.HalfAtATime` 和 `CodeDeployDefault.OneAtATime`。

Warning

您无法删除仍在使用中的自定义部署配置。如果您删除未使用的自定义部署配置，则不再能够将它与新部署和新部署组关联。此操作无法撤销。

要使用 AWS CLI 删除部署配置，请调用 `delete-deployment-config` 命令，并指定部署配置名称。要查看部署配置名称的列表，请调用 `list-deployment-configs` 命令。

以下示例删除名为 `ThreeQuartersHealthy` 的部署配置。

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

在 AWS CodeDeploy 中使用应用程序

配置实例后，必须先在 AWS CodeDeploy 中创建应用程序，然后才能部署修订。应用程序 只是一个名称或容器，AWS CodeDeploy 使用此名称或容器来确保在部署期间引用正确的修订、部署配置和部署组。

使用下表中的信息完成接下来的步骤：

我尚未创建实例。	请参阅 使用适用于 AWS CodeDeploy 的实例 (p. 133) ，然后返回此页。
我已创建实例，但尚未完成标记。	请参阅 Tagging Instances for AWS CodeDeploy Deployments (p. 134) ，然后返回此页。
我尚未创建应用程序。	请参阅 使用 AWS CodeDeploy 创建应用程序 (p. 189)
我已创建应用程序，但尚未创建部署组。	请参阅 使用 AWS CodeDeploy 创建部署组 (p. 198) 。
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 AWS CodeDeploy 的应用程序修订 (p. 208) 。
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 AWS CodeDeploy 创建部署 (p. 220) 。

主题

- [使用 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)
- [使用 AWS CodeDeploy 查看应用程序详细信息 \(p. 195\)](#)
- [重命名 AWS CodeDeploy 应用程序 \(p. 196\)](#)
- [删除 AWS CodeDeploy 中的应用程序 \(p. 196\)](#)

使用 AWS CodeDeploy 创建应用程序

应用程序只是 AWS 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。您可使用 AWS CodeDeploy 控制台、AWS CLI、AWS CodeDeploy API 或 AWS 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。AWS CodeDeploy 支持两种部署类型：

- **就地部署**：停止部署组中每个实例上的应用程序，安装最新的应用程序修订版，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2/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 \(p. 3\)](#)。
- **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：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：
 - 系统将为替代环境配置实例。
 - 替代实例上将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来说，等待时间可选。
- 替代环境中的实例在 Elastic Load Balancing 负载均衡器中进行注册，使得流量重新路由至这些实例。系统将撤销原始环境中的实例注册，进而终止或因其他使用情形而保持运行。

Note

在使用 EC2/本地 计算平台时，蓝/绿部署只能与 Amazon EC2 实例配合使用。

- AWS Lambda 计算平台上的蓝/绿部署：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方法。所有 AWS Lambda 计算平台部署都是蓝/绿部署。因此，您无需指定部署类型。

有关蓝/绿部署的更多信息，请参阅[蓝/绿部署概述 \(p. 4\)](#)。

当您使用 AWS 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 AWS 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 AWS 账户注册的应用程序的列表，请参阅[使用 AWS CodeDeploy 查看应用程序详细信息 \(p. 195\)](#)。有关使用 AWS CloudFormation 模板创建应用程序的信息，请参阅[适用于 AWS CodeDeploy 的 AWS CloudFormation 模板参考 \(p. 300\)](#)。

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
无服务器 AWS Lambda 函数	否	是

主题

- [为就地部署创建应用程序 \(控制台\) \(p. 190\)](#)
- [为蓝/绿部署创建应用程序 \(控制台\) \(p. 192\)](#)
- [为 AWS Lambda 函数部署创建应用程序 \(控制台\) \(p. 194\)](#)
- [创建应用程序 \(CLI\) \(p. 195\)](#)

为就地部署创建应用程序 (控制台)

使用 AWS 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：

- 您没有准备好要在 AWS CodeDeploy 部署中使用的实例。要设置您的实例，请按照[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)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 \(p. 186\)](#)中的说明操作，然后执行本主题中的步骤。
- 您没有信任 AWS 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[步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)中的说明操作，然后返回到本主题中的相应步骤。
- 您希望为就地部署选择 Elastic Load Balancing 中的 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，但尚未创建它。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果出现 AWS CodeDeploy 主页，请选择 Get Started Now。
3. 选择 Create application。
4. 在 Application name 框中，键入应用程序的名称。（在 AWS 账户中，只能为每个区域使用 AWS 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。）
5. 从 计算平台 下拉列表中，选择 EC2/On-Premises。
6. 在 Deployment group name 框中，键入用于描述部署组的名称。

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 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，AWS CodeDeploy 仍认为它们是两个部署组，因为它们关联的应用程序不同。

7. 选择 In-place deployment。
8. 在 Environment configuration 中，做出以下选择：
 - 在 Auto Scaling group 选项卡上：选择要部署应用程序修订的 Auto Scaling 组的名称。当新 Amazon EC2 实例作为 Auto Scaling 组一部分启动时，AWS 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 Auto Scaling 组添加到一个部署组。
 - 在 Amazon EC2 instances 或 On-premises instance 选项卡上：在 Key 和 Value 字段中，键入您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。
 - 您可以在 Value 字段中使用通配符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，如果您在 Key 字段中选择 Name 并在 Value 字段中键入 **GRP-*a**，AWS CodeDeploy 将标识符合该模式的所有实例，例如 **GRP-1a**、**GRP-2a** 和 **GRP-XYZ-a**。
 - Value 字段区分大小写。
 - 要从列表中删除键值对，请选择删除图标。

随着 AWS CodeDeploy 查找与每个指定的键值对或 Auto Scaling 组名称匹配的实例，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 AWS CodeDeploy Deployments \(p. 134\)](#)。

9. (可选) 在 Load balancer 中，选择 Enable load balancing，然后选择现有 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，以在部署过程中管理流向实例的流量。

在部署期间，每个实例都会从负载均衡器 (Classic Load Balancer) 或目标组 (Application Load Balancer 和 Network Load Balancer) 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 AWS CodeDeploy 部署的负载均衡器的更多信息，请参阅 [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 40\)](#)。

10. 在 Deployment configuration 列表中，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 AWS CodeDeploy 中使用部署配置 \(p. 184\)](#)。
11. (可选) 在 Advanced 中，配置要包含在部署中的任何选项，例如，Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

有关更多信息，请参阅 [为部署组配置高级选项 \(p. 206\)](#)。

12. 在 Service role ARN 中，选择信任 AWS CodeDeploy 的服务角色 (至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限) 的 Amazon 资源名称 (ARN)。要获取服务角色 ARN，请参阅 [获取服务角色 ARN \(控制台\) \(p. 21\)](#)。
13. 选择 Create application。

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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

为蓝/绿部署创建应用程序 (控制台)

使用 AWS CodeDeploy 控制台为蓝/绿部署创建应用程序：

Note

对 AWS Lambda 计算平台的部署始终是蓝/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 AWS 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 [使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#) 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 \(p. 186\)](#) 中的说明操作，然后执行本主题中的步骤。
- 您没有至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限的信任 AWS CodeDeploy 的服务角色。要创建和配置服务角色，请按照 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中的说明操作，然后执行本主题中的步骤。
- 您尚未在 Elastic Load Balancing 中创建 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，以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 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果出现 AWS CodeDeploy 主页，请选择 Get Started Now。
3. 选择 Create application。
4. 在 Application name 中，键入应用程序的名称。（在 AWS 账户中，只能为每个区域使用 AWS 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。）
5. 在 计算平台 下拉列表中，选择 EC2/On-Premises。
6. 在 Deployment group name 框中，键入用于描述部署组的名称。

Note

如果您希望使用另一部署组中所用的相同设置，请在此页指定这些设置。您可能希望重复使用的设置包括部署组名称、标签、Auto Scaling 组名称或部署配置。尽管新的部署组和现有部署组具有相同的名称，但 AWS CodeDeploy 会将它们视为不同的部署组，因为它们分别与不同的应用程序关联。

7. 选择 Blue/green deployment。
8. 在 Environment configuration 中，选择为替换环境提供实例的方法：
 - Automatically copy Auto Scaling group：AWS CodeDeploy 复制您指定的 Auto Scaling 组，进行创建。

- **Manually provision instances**：在创建部署前，您不会为替换环境指定实例。您必须在启动部署前创建实例。您应于此处指定要替换的实例。
9. 根据您在步骤 7 中的选择，请执行以下操作之一：
- 如果您选择了 **Automatically copy Auto Scaling group**：在 **Auto Scaling group** 中，选择一个 **Auto Scaling** 组的名称，您要将该组用作将为替换环境中的实例创建的 **Auto Scaling** 组的模板。您选择的 **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 **Manually provision instances**：在 **Choose the EC2 instances or Auto Scaling groups where the current application revision is deployed** 中，输入 Amazon EC2 标签值或 **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)。
10. 在 **Load balancer** 中，选择部署过程中用于在替换环境中注册实例的 **传统负载均衡器**、**应用程序负载均衡器** 或 **Network Load Balancer**。

Note

您的原始环境中的实例可以注册到您选择的负载均衡器，但并非必须这样做。

有关 AWS CodeDeploy 部署的负载均衡器的更多信息，请参阅[Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 40\)](#)。

11. 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，请继续执行步骤 11。否则，请跳至步骤 12。

12. 要更改蓝/绿部署的部署设置，请选择 **Edit deployment settings**，更新以下任一设置，然后选择 **Submit**。

设置	选项
Traffic rerouting	<ul style="list-style-type: none">• Reroute traffic immediately：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。• I will choose whether to reroute traffic：替换环境中的实例不会注册到负载均衡器，除非您手动重新路由流量。如果在没有重新路由流量的情况下经过了指定的等待时间，部署状态将更改为“Stopped”。
部署配置	<p>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。</p> <p>Note</p> <p>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。</p> <p>有关更多信息，请参阅 在 AWS CodeDeploy 中使用部署配置 (p. 184)。</p>
Original instances	<ul style="list-style-type: none">• Terminate the original instances in the deployment group：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。• Keep the original instances in the deployment group running：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。

13(可选) 在 Advanced 中, 配置要包含在部署中的选项, 例如, Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

有关在部署组中指定高级选项的信息, 请参阅[为部署组配置高级选项 \(p. 206\)](#)。

14. 在 Service role ARN 框中, 选择信任 AWS CodeDeploy 的服务角色 (至少具有 [步骤 3 : 为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限) 的 Amazon 资源名称 (ARN)。要获取服务角色 ARN, 请参阅[获取服务角色 ARN \(控制台\) \(p. 21\)](#)。

15. 选择 Create application。

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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

为 AWS Lambda 函数部署创建应用程序 (控制台)

使用 AWS CodeDeploy 控制台为 Lambda 函数部署创建应用程序：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果出现 AWS CodeDeploy 主页, 请选择 Get Started Now。
3. 选择 Create application。
4. 在 Application name 中, 键入应用程序的名称。(在 AWS 账户中, 只能为每个区域使用 AWS 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。)
5. 从 计算平台 下拉列表中, 选择 AWS Lambda。
6. 在 Deployment group name 中, 键入部署组的名称。

Note

如果您希望使用另一部署组中所用的相同设置, 请在此页指定这些设置。您可能希望重复使用部署触发器、回滚部署或配置。尽管新的部署组和现有部署组具有相同的名称, 但 AWS CodeDeploy 会将它们视为不同的部署组, 因为它们分别与不同的应用程序关联。

7. 从 Deployment configuration 下拉列表中, 选择一个预定义的部署配置, 然后跳到步骤 9。

有关部署配置的更多信息, 请参阅 [AWS Lambda 计算平台上的部署配置 \(p. 185\)](#)。

8. 要创建自定义配置, 请选择 Create deployment configuration 并执行以下操作：
 - 对于 Deployment configuration name, 键入配置的名称。
 - (可选) 对于 Description, 键入配置的说明。
 - 从 Type 下拉列表中, 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。
 - 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。
 - 在 Interval 对话框中, 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, 则这是每次增量流量转移之间间隔的分钟数。

Note

AWS 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。最大时间间隔值将是 2,880 除以 4, 即 720 分钟。

- 选择 Submit。
9. (可选) 在 Advanced 中，配置要包含在部署中的任何选项，例如，Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。
- 有关更多信息，请参阅 [为部署组配置高级选项 \(p. 206\)](#)。
10. 在 Service role ARN 中，选择信任 AWS CodeDeploy 的服务角色 (至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限) 的 Amazon 资源名称 (ARN)。要获取服务角色 ARN，请参阅 [获取服务角色 ARN \(控制台\) \(p. 21\)](#)。
11. 选择 Create application。

创建应用程序 (CLI)

要使用 AWS CLI 创建应用程序，请调用 `create-application` 命令，并指定唯一表示该应用程序的名称。(在 AWS 账户中，只能为每个区域使用 AWS 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 AWS 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 AWS CodeDeploy 创建部署组 \(p. 198\)](#)。

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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

使用 AWS CodeDeploy 查看应用程序详细信息

可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 查看有关与您的 AWS 账户关联的所有应用程序的详细信息。

主题

- [查看应用程序详细信息 \(控制台\) \(p. 195\)](#)
- [查看应用程序详细信息 \(CLI\) \(p. 196\)](#)

查看应用程序详细信息 (控制台)

使用 AWS CodeDeploy 控制台查看应用程序详细信息：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

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 AWS CLI 查看应用程序详细信息，请调用 `get-application` 命令、`batch-get-application` 命令或 `list-applications` 命令。

要查看有关单个应用程序的详细信息，请调用 `get-application` 命令，并指定应用程序名称。

要查看有关多个应用程序的详细信息，请调用 `batch-get-applications` 命令，并指定多个应用程序名称。

要查看应用程序名称的列表，请调用 `list-applications` 命令。

重命名 AWS CodeDeploy 应用程序

您可以使用 AWS CLI 或 AWS CodeDeploy API 更改应用程序的名称。

要查看应用程序名称的列表，请使用 AWS CLI 调用 `list-applications` 命令。

有关使用 AWS CLI 更改应用程序名称的信息，请参阅 `update-application`。

有关使用 AWS CodeDeploy API 更改应用程序名称的信息，请参阅 `API_UpdateApplication`。

删除 AWS CodeDeploy 中的应用程序

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 操作删除应用程序。有关使用 AWS CodeDeploy API 操作的信息，请参阅 `DeleteApplication`。

Warning

删除应用程序将从 AWS 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。此操作无法撤销。

主题

- [删除应用程序 \(控制台\)](#) (p. 196)
- [删除应用程序 \(AWS CLI\)](#) (p. 197)

删除应用程序 (控制台)

要使用 AWS CodeDeploy 控制台删除应用程序，请执行以下步骤：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
3. 在应用程序列表中，选择要删除的应用程序的名称。
4. 在 Application details 页上的 Deployment groups 中，选择部署组旁边的按钮。在 Actions 菜单上，选择 Delete。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。对任何其他部署组重复此步骤。

5. 在 Application details 页的底部，选择 Delete application。
6. 在系统提示时，键入应用程序的名称以确认要删除此应用程序，然后选择 Delete。

删除应用程序 (AWS CLI)

要使用 AWS CLI 删除应用程序，请调用 `delete-application` 命令，并指定应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。

在 AWS CodeDeploy 中使用部署组

AWS Lambda 计算平台部署中的部署组

在 AWS Lambda 部署中，部署组定义了一组 AWS CodeDeploy 配置，以便在未来以无服务器方式将 Lambda 部署到组。例如，部署组可能指定警报和回滚。AWS Lambda 部署组中的单一部署会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。部署组中包含单独标记的实例和/或 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。

您可以将多个部署组与 AWS CodeDeploy 中的一个应用程序关联。这使得能够在不同的时间将一个应用程序修订部署到不同的实例组。例如，您可以使用一个部署组将一个应用程序修订部署到一组标记为 `Test` 的实例，以便在其中确保代码质量。接下来，将相同应用程序修订部署到包含标记为 `Staging` 的实例的部署组，以便进行进一步验证。最后，当您准备好向客户发布最新应用程序时，部署到包括标记为 `Production` 的实例的部署组。

您也可以使用多个标签组，进一步优化部署组中所包含实例的条件。有关信息，请参阅 [Tagging Instances for AWS CodeDeploy Deployments \(p. 134\)](#)。

当您使用 AWS 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 AWS 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 AWS 账户关联的部署组的列表，请参阅[使用 AWS CodeDeploy 查看部署组详细信息 \(p. 203\)](#)。

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 \(p. 156\)](#)。有关 Auto Scaling 的信息，请参阅[将 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。

主题

- the section called “创建部署组” (p. 198)
- the section called “查看部署组详细信息” (p. 203)
- the section called “更改部署组设置” (p. 204)
- the section called “为部署组配置高级选项” (p. 206)
- the section called “删除部署组” (p. 207)

使用 AWS CodeDeploy 创建部署组

您可使用 AWS CodeDeploy 控制台、AWS CLI、AWS CodeDeploy API 或 AWS CloudFormation 模板创建部署组。有关使用 AWS CloudFormation 模板创建部署组的信息，请参阅 [适用于 AWS CodeDeploy 的 AWS CloudFormation 模板参考 \(p. 300\)](#)。

当您使用 AWS 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 AWS 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 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)。

主题

- [为就地部署创建部署组 \(控制台\) \(p. 199\)](#)
- [为蓝/绿部署创建部署组 \(控制台\) \(p. 200\)](#)
- [在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)
- [创建部署组 \(CLI\) \(p. 203\)](#)

为就地部署创建部署组 (控制台)

使用 AWS 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：

- 您尚未为实例做好在应用程序的首次 AWS CodeDeploy 部署中使用的准备。要设置您的实例，请按照[使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)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 \(p. 186\)](#)中的说明操作，然后执行本主题中的步骤。
- 您没有至少具有[步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)中描述的信任和权限的信任 AWS CodeDeploy 的服务角色。要创建和配置服务角色，请按照[步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)中的说明操作，然后执行本主题中的步骤。
- 您希望为就地部署选择 Elastic Load Balancing 中的 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，但尚未创建它。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。
3. 在 Applications 页上，选择要为其创建部署组的应用程序的名称。
4. 选择 Create deployment group。
5. 在 Deployment group name 框中，键入用于描述部署组的名称。

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 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，AWS CodeDeploy 仍认为它们是两个部署组，因为它们关联的应用程序不同。

6. 选择 In-place deployment。
7. 在 Environment configuration 中，做出以下选择：
 - 在 Auto Scaling group 选项卡上：选择要部署应用程序修订的 Auto Scaling 组的名称。当新 Amazon EC2 实例作为 Auto Scaling 组一部分启动时，AWS 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 Auto Scaling 组添加到一个部署组。

- 在 Amazon EC2 instances 或 On-premises instance 选项卡上：在 Key 和 Value 字段中，键入您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。
- 您可以在 Value 字段中使用通配符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，如果您在 Key 字段中选择 Name 并在 Value 字段中键入 **GRP-*a**，AWS CodeDeploy 将标识符合该模式的所有实例，例如 **GRP-1a**、**GRP-2a** 和 **GRP-XYZ-a**。
- Value 字段区分大小写。
- 要从列表中删除键值对，请选择删除图标。

随着 AWS CodeDeploy 查找与每个指定的键值对或 Auto Scaling 组名称匹配的实例，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 AWS CodeDeploy Deployments \(p. 134\)](#)。

8. (可选) 在 Load balancer 中，选择 Enable load balancing，然后选择现有 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，以在部署过程中管理流向实例的流量。

在部署期间，每个实例都会从负载均衡器 (Classic Load Balancer) 或目标组 (Application Load Balancer 和 Network Load Balancer) 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 AWS CodeDeploy 部署的负载均衡器的更多信息，请参阅 [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 40\)](#)。

9. 在 Deployment configuration 列表中，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 AWS CodeDeploy 中使用部署配置 \(p. 184\)](#)。
10. (可选) 在 Advanced 中，配置要包含在部署中的任何选项，例如，Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

有关更多信息，请参阅 [为部署组配置高级选项 \(p. 206\)](#)。

11. 在 Service role ARN 中，选择信任 AWS CodeDeploy 的服务角色 (至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限) 的 Amazon 资源名称 (ARN)。要获取服务角色 ARN，请参阅 [获取服务角色 ARN \(控制台\) \(p. 21\)](#)。

12. 选择 Create deployment group。

为蓝/绿部署创建部署组 (控制台)

使用 AWS 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 AWS 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 [使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#) 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 \(p. 186\)](#) 中的说明操作，然后执行本主题中的步骤。
- 您没有至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限的信任 AWS CodeDeploy 的服务角色。要创建和配置服务角色，请按照 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中的说明操作，然后执行本主题中的步骤。
- 您尚未在 Elastic Load Balancing 中创建 传统负载均衡器 来注册替换环境中的实例。有关更多信息，请参阅 [在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer \(p. 202\)](#)。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单中选择 Applications。
3. 在 Applications 页上，选择要为其创建部署组的应用程序的名称。
4. 选择 Create deployment group。
5. 在 Deployment group name 框中，键入用于描述部署组的名称。

Note

如果您希望使用另一部署组中所用的相同设置，请在此页指定这些设置。您可能希望重复使用的设置包括部署组名称、标签、Auto Scaling 组名称或部署配置。尽管新的部署组和现有部署组具有相同的名称，但 AWS CodeDeploy 会将它们视为不同的部署组，因为它们分别与不同的应用程序关联。

6. 选择 Blue/green deployment。
7. 在 Environment configuration 中，选择为替换环境提供实例的方法：
 - Automatically copy Auto Scaling group：AWS CodeDeploy 复制您指定的 Auto Scaling 组，进行创建。
 - Manually provision instances：在创建部署前，您不会为替换环境指定实例。您必须在启动部署前创建实例。您应于此处指定要替换的实例。
8. 根据您在步骤 7 中的选择，请执行以下操作之一：
 - 如果您选择了 Automatically copy Auto Scaling group：在 Auto Scaling group 中，选择一个 Auto Scaling 组的名称，您要将该组用作将为替换环境中的实例创建的 Auto Scaling 组的模板。您选择的 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 Manually provision instances：在 Choose the EC2 instances or Auto Scaling groups where the current application revision is deployed 中，输入 Amazon EC2 标签值或 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)。
9. 在 Load balancer 中，选择部署过程中用于在替换环境中注册实例的 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer。

Note

您的原始环境中的实例可以注册到您选择的负载均衡器，但并非必须这样做。

有关 AWS CodeDeploy 部署的负载均衡器的更多信息，请参阅[Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 40\)](#)。

10. 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，请继续执行步骤 11。否则，请跳至步骤 12。

11. 要更改蓝/绿部署的部署设置，请选择 Edit deployment settings，更新以下任一设置，然后选择 Submit。

设置	选项
Traffic rerouting	<ul style="list-style-type: none">• Reroute traffic immediately：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。• I will choose whether to reroute traffic：替换环境中的实例不会注册到负载均衡器，除非您手

设置	选项
	动重新路由流量。如果在没有重新路由流量的情况下经过了指定的等待时间，部署状态将更改为“Stopped”。
部署配置	<p>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。</p> <p>Note</p> <p>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。</p> <p>有关更多信息，请参阅 在 AWS CodeDeploy 中使用部署配置 (p. 184)。</p>
Original instances	<ul style="list-style-type: none">• Terminate the original instances in the deployment group：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。• Keep the original instances in the deployment group running：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。

12(可选) 在 Advanced 中，配置要包含在部署中的选项，例如，Amazon SNS 通知触发器、Amazon CloudWatch 警报或自动回滚。

有关在部署组中指定高级选项的信息，请参阅[为部署组配置高级选项 \(p. 206\)](#)。

13.在 Service role ARN 框中，选择信任 AWS CodeDeploy 的服务角色 (至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的信任和权限) 的 Amazon 资源名称 (ARN)。要获取服务角色 ARN，请参阅[获取服务角色 ARN \(控制台\) \(p. 21\)](#)。

14.选择 Create deployment group。

在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer

在运行任何蓝/绿部署之前，或在就地部署时希望在部署组中指定可选的负载均衡器之前，您必须已在 Elastic Load Balancing 中创建 [传统负载均衡器](#) 或 [应用程序负载均衡器](#)。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 [传统负载均衡器](#)，请遵循 Classic Load Balancer 用户指南中的[教程：创建传统负载均衡器](#) 中的说明。请注意以下几点：

- 在步骤 2：定义负载均衡器中的创建内部负载均衡器中，选择创建实例时所选同一 VPC。
- 在步骤 5：向负载均衡器注册 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)。
- 在步骤 7：创建并验证您的负载均衡器中，记录负载均衡器的 DNS 地址。

例如，如果您已将负载均衡器命名为 my-load-balancer，则您的 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 [应用程序负载均衡器](#)，请遵循以下某一主题中的说明：

- [创建应用程序负载均衡器](#)

- [教程：使用 AWS CLI 创建 应用程序负载均衡器](#)

创建部署组 (CLI)

要使用 AWS CLI 创建部署组，请调用 `create-deployment-group` 命令，并指定：

- 应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
- 部署组的名称。此名称对于与部署组关联的每个应用程序来说必须是唯一的。
- 与用于标识要包含在部署组中的实例的标签、标签组或 Auto Scaling 组名称有关的信息。
- 服务角色的 Amazon 资源名称 (ARN) 标识符，在与其他 AWS 服务交互时，该服务角色允许 AWS CodeDeploy 代表您的 AWS 账户执行操作。要获取服务角色 ARN，请参阅[获取服务角色 ARN \(CLI\) \(p. 21\)](#)。有关服务角色的更多信息，请参阅 IAM 用户指南中的[角色术语和概念](#)。
- 与部署组相关联的部署类型相关信息，即就地部署或蓝/绿部署。
- (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 \(p. 187\)](#)。如果未指定，AWS CodeDeploy 将使用默认部署配置。
- (可选) 用于创建向订阅了 Amazon Simple Notification Service 主题的用户推送有关部署和实例事件的通知的触发器的命令。有关更多信息，请参阅[Monitoring Deployments with Amazon SNS Event Notifications \(p. 250\)](#)。
- (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) Elastic Load Balancing 中 传统负载均衡器 或 应用程序负载均衡器 的名称，用于在部署过程中管理实例的流量。
- 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。
 - Elastic Load Balancing 中 传统负载均衡器 或 应用程序负载均衡器 的名称，供替换环境中注册的实例使用。

使用 AWS CodeDeploy 查看部署组详细信息

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS 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 \(控制台\) \(p. 203\)](#)
- [查看部署组详细信息 \(CLI\) \(p. 204\)](#)

查看部署组详细信息 (控制台)

使用 AWS CodeDeploy 控制台查看部署组详细信息：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 如果未显示 Applications 页，请在 AWS CodeDeploy 菜单上选择 Applications。
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。在导航栏上的区域选择器中，选择 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

4. 要查看有关单个部署组的详细信息，请在 Deployment groups 中选择该部署组旁边的箭头。

查看部署组详细信息 (CLI)

要使用 AWS CLI 查看部署组详细信息，请调用 `get-deployment-group` 命令或 `list-deployment-groups` 命令。

要查看有关单个部署组的详细信息，请调用 `get-deployment-group` 命令，并指定：

- 与部署组关联的应用程序名称。要获取应用程序名称，请调用 `list-applications` 命令。
- 部署组名称。要获取部署组名称，请调用 `list-deployment-groups` 命令。

要查看部署组名称的列表，请调用 `list-deployment-groups` 命令，并指定与部署组关联的应用程序名称。要获取应用程序名称，请调用 `list-applications` 命令。

使用 AWS CodeDeploy 更改部署组设置

可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS 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 \(p. 186\)](#) 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 AWS CodeDeploy，而且至少具有 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中描述的权限。要创建和配置具有正确权限的服务角色，请按照 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#) 中的说明操作，然后返回到本主题。

主题

- [更改部署组设置 \(控制台\) \(p. 204\)](#)
- [更改部署组设置 \(CLI\) \(p. 205\)](#)

更改部署组设置 (控制台)

要使用 AWS CodeDeploy 控制台更改部署组设置，请执行以下操作：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 选择 Applications。
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。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

4. 在 Application details 页的 Deployment groups 中，选择要更改的部署组旁的按钮。
5. 在 Actions 菜单上选择 Edit。
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 AWS CodeDeploy 创建部署组](#) (p. 198)。

7. 如果要上次成功的修订部署到部署组，请选中 Deploy changes made to **deployment group name**，然后选择 Save。在系统提示时，选择 Deploy。AWS CodeDeploy 将更新部署组的信息，开始根据您指定的更改将上次成功的修订部署到部署组，并显示 Deployments 页。

Note

只有此部署组中完成过成功部署时，才会显示 Deploy changes made to **deployment group name** 复选框。

8. 如果要使用您所做的更改更新部署组的信息，但此时不希望将任何应用程序部署到部署组，请清除 Deploy changes made to **deployment group name**，然后选择 Save。AWS CodeDeploy 将更新部署组的信息，但不会将任何应用程序部署到部署组。

更改部署组设置 (CLI)

要使用 AWS CLI 更改部署组设置，请调用 `update-deployment-group` 命令，并指定：

- 对于 EC2/本地 和 AWS Lambda 部署：
 - 应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
 - 当前部署组名称。要查看部署组名称的列表，请调用 `list-deployment-groups` 命令。
 - (可选) 不同的部署组名称。
 - (可选) 与某个服务角色对应的另一 Amazon 资源名称 (ARN)，当与其他 AWS 服务交互时，该服务角色允许 AWS CodeDeploy 代表您的 AWS 账户操作。要获取服务角色 ARN，请参阅[获取服务角色 ARN \(CLI\)](#) (p. 21)。有关服务角色的更多信息，请参阅 IAM 用户指南中的[角色术语和概念](#)。
 - (可选) 部署配置的名称。要查看部署配置列表，请参阅[View Deployment Configuration Details](#) (p. 187)。(如果未指定，AWS 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 Simple Notification Service 中的某个主题发布信息，以便该主题的订阅者可以接收有关此部署组中的部署和实例事件的通知。有关信息，请参阅[Monitoring Deployments with Amazon SNS Event Notifications](#) (p. 250)。
- 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 Auto Scaling 组的名称。

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 [AWS CodeDeploy 创建应用程序](#) (p. 189)
- 使用 [AWS CodeDeploy 创建部署组](#) (p. 198)
- 使用 [AWS CodeDeploy 更改部署组设置](#) (p. 204)

Amazon SNS 通知触发器：您可以将触发器添加到一个 AWS CodeDeploy 部署组，以接收与该部署组中的部署相关的事件的通知。这些通知将会发送到订阅了您已设置触发器操作的 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 AWS CodeDeploy 必须有权从该部署组发布到此主题。如果您尚未完成这些设置步骤，可稍后向部署组添加触发器。

如果您需要立即创建触发器以接收有关此应用程序的部署组中的部署事件的通知，请选择 Create trigger。

如果您要部署到 Amazon EC2 实例，则可以为实例创建通知并接收有关实例的通知。

有关更多信息，请参阅 [Monitoring Deployments with Amazon SNS Event Notifications](#) (p. 250)。

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，您可以在 AWS CodeDeploy 操作中使用的实例或 Amazon EC2 Auto Scaling 组创建一个警报。对于 AWS Lambda 部署，您可以在 Lambda 函数中为错误创建警报。

可将部署配置为在 Amazon CloudWatch 警报检测到某个指标低于或超出定义的阈值时停止。

您必须先要在 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，请选择 Add alarm。
2. 在 Alarm name 中，键入您已经设置好监控此部署的 CloudWatch 警报的名称。

须严格按照 CloudWatch 中创建的名称输入 CloudWatch 警报。要查看警报列表，请在 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台，然后选择 ALARM。

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 AWS CodeDeploy 无法从 Amazon CloudWatch 中检索警报状态的情况下继续，请选择 Continue deployments even if alarm status is unavailable。

Note

此选项对应于 AWS CodeDeploy API 中 [AlarmConfiguration](#) 对象中的 ignorePollAlarmFailure。

有关更多信息，请参阅 [在 AWS CodeDeploy 中使用 CloudWatch 警报监控部署](#) (p. 245)。

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：

- Roll back when a deployment fails. AWS 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，AWS 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 AWS CodeDeploy 重新部署和回滚部署 \(p. 233\)](#)。

使用 AWS CodeDeploy 删除部署组

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 删除与 AWS 账户关联的部署组。

Warning

如果您删除部署组，则也将从 AWS 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）\(p. 207\)](#)
- [删除部署组 \(CLI\) \(p. 207\)](#)

删除部署组（控制台）

要使用 AWS CodeDeploy 控制台删除部署组：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。
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 页上，在 Deployment groups 中，选择要删除的部署组旁边的按钮。
5. 在 Actions 菜单上，选择 Delete。
6. 在系统提示时，键入部署组的名称以确认要删除此部署组，然后选择 Delete。

删除部署组 (CLI)

要使用 AWS CLI 删除部署组，请调用 `delete-deployment-group` 命令，并指定：

- 与部署组关联的应用程序的名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
- 与应用程序关联的部署组的名称。要查看部署组名称的列表，请调用 `list-deployment-groups` 命令。

使用 AWS CodeDeploy 的应用程序修订

在 AWS CodeDeploy 中，修订包含 AWS CodeDeploy 将部署到您实例的源文件版本或者 AWS CodeDeploy 将在您实例上运行的脚本。

您规划修订、将 AppSpec file 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

- [计划 AWS CodeDeploy 的修订 \(p. 208\)](#)
- [将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)
- [选择 AWS CodeDeploy 存储库类型 \(p. 213\)](#)
- [将 AWS CodeDeploy 的修订推送到 Amazon S3 \(p. 214\)](#)
- [使用 AWS CodeDeploy 查看应用程序修订详细信息 \(p. 216\)](#)
- [使用 AWS CodeDeploy 在 Amazon S3 中注册应用程序修订 \(p. 217\)](#)

计划 AWS CodeDeploy 的修订

良好的计划使将修订部署到实例变得轻松多了。

对于到 AWS Lambda 计算平台的部署，修订和 AppSpec file 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 AWS CodeDeploy 的修订 \(p. 209\)](#)。

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/ 根文件夹 (Linux, macOS, or Unix) 或 c:\temp 根文件夹 (Windows) 中：

```
/tmp/ or c:\temp (root folder)
|--content (subfolder)
|   |--myTextFile.txt
|   |--mySourceFile.rb
|   |--myExecutableFile.exe
|   |--myInstallerFile.msi
|   |--myPackage.rpm
|   |--myImageFile.png
|--scripts (subfolder)
|   |--myShellScript.sh
|   |--myBatchScript.bat
|   |--myPowerShellScript.ps1
--appspec.yml
```

此根文件夹还包含 application specification file (AppSpec file)，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 AWS CodeDeploy 的修订 \(p. 209\)](#)。

将应用程序规范文件添加到 AWS CodeDeploy 的修订

本主题介绍如何向部署中添加 AppSpec file。它还包括用来为 AWS Lambda 和 EC2/本地 部署创建 AppSpec file 的模板。

为 AWS Lambda 部署添加 AppSpec 文件

对于到 AWS Lambda 计算平台 的部署：

- AppSpec file 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 file 相同。
- AppSpec file 可以使用 JSON 格式或 YAML 编写。
- 在创建部署时，可以将 AppSpec file 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 AWS Lambda 计算平台 部署 \(控制台\)](#) (p. 223)。

创建 AppSpec file：

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 file。如果您使用 AppSpec 编辑器，则会在您选择 Deploy 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 AWS CLI 创建部署，请在 AppSpec file 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 file 推送到 Amazon S3。
5. (可选) 如果使用 AppSpec 编辑器，请选择 Save as text file 以将 AppSpec file 保存到您的硬盘驱动器上。

带有说明的 YAML AppSpec 文件模板

有关在 hooks 部分中使用的生命周期事件的信息，请参阅[用于 AWS Lambda 部署的 AppSpec 的“hooks”部分](#) (p. 285)。

```
# This is an appspec.yml template file for use with an AWS Lambda deployment in AWS
# 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
# "AWS CodeDeploy User Guide" at
# http://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 AWS 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 文件模板

在以下模板中，您需要将“MyFunction”替换为 AWS 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 AWS Lambda 部署的 AppSpec 的“hooks”部分 \(p. 285\)](#)。

```
{
  "version": 0.0,
  "Resources": [{
    "MyFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Name": "",
        "Alias": "",
        "CurrentVersion": "",
        "TargetVersion": ""
      }
    }
  ]},
  "Hooks": [{
    "LifecycleEvent": ""
  }]
}
```

为 EC2/本地 部署添加 AppSpec 文件

如果没有 AppSpec file，AWS CodeDeploy 无法将应用程序修订中的源文件映射到其目标，也无法为您向 EC2/本地 计算平台 中进行的部署运行脚本。

每个修订只能包含一个 AppSpec file。

将 AppSpec file 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 验证程序检查您的 AppSpec file 是否有效。
4. 在修订的根目录中将文件另存为 appspec.yml。
5. 运行以下命令之一，验证您是否已将 AppSpec file 放置在根目录中：

- 对于 Linux, macOS, or Unix：

```
find /path/to/root/directory -name appspec.yml
```

如果 AppSpec file 不在根目录中，则不输出任何内容。

- 对于 Windows：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec file 未保存在根目录中，则会显示 File Not Found 错误。

6. 将修订推送到 Amazon S3 或 GitHub。

有关说明，请参阅[将 AWS CodeDeploy 的修订推送到 Amazon S3 \(p. 214\)](#)。

带有说明的 AppSpec 文件模板

Note

到 Windows Server 实例的部署不支持 runas 元素。如果要部署到 Windows Server 实例，请勿将其包含在您的 AppSpec file 中。

```
# This is an appspec.yml template file for use with an EC2/## deployment in AWS 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
# "AWS CodeDeploy User Guide" at
# http://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 AWS 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 AWS CodeDeploy 存储库类型

AWS CodeDeploy 需要的文件的存储位置称为存储库。对存储库的使用取决于您的部署使用哪个 计算平台。

- EC2/本地：要将您的应用程序代码部署到一个或多个实例，必须将您的代码打包到一个存档文件，并将代码放置到 AWS 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 AWS CodeDeploy 支持的某个存储库类型。
- AWS Lambda：部署需要 AppSpec file，在部署过程中可以通过下列方式之一访问它：
 - 从 Amazon S3 存储桶。
 - 从直接键入控制台中的 AppSpec 编辑器内的文本。有关更多信息，请参阅 [创建 AWS Lambda 计算平台部署 \(控制台\)](#) (p. 223)。
 - 如果您使用 AWS 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 file。有关更多信息，请参阅 [创建 AWS Lambda 计算平台部署 \(CLI\)](#) (p. 226)。

AWS CodeDeploy 当前支持以下存储库类型：

Amazon S3	<p>Amazon Simple Storage Service (Amazon S3) 是适用于安全的可扩展对象存储的 AWS 解决方案。Amazon S3 将数据作为对象存储在存储桶中。对象由文件和描述该文件的任何可选元数据组成。</p> <p>要将数据元存储到 Amazon S3 中，请将要存储的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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 在 Amazon S3 中创建存储桶• 将 AWS CodeDeploy 的修订推送到 Amazon S3 (p. 214)• 使用 AWS CodeDeploy 自动从 Amazon S3 部署
GitHub	<p>(仅 EC2/本地 部署) 您可以将应用程序修订存储在 GitHub 存储库中。只要 GitHub 存储库中的源代码发生更改，您就可以触发自该存储库的部署。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 将 AWS CodeDeploy 与 GitHub 集成 (p. 45)• 教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序 (p. 101)• 使用 AWS CodeDeploy 自动从 GitHub 部署
Bitbucket	<p>(仅 EC2/本地 部署) 您可以将针对 Amazon EC2 实例的代码直接从 Bitbucket UI 推送到任何部署组，</p>

而无需登录您的持续集成 (CI) 平台或 Amazon EC2 实例来运行手动部署过程。Bitbucket 会先将代码推送到您指定的 Amazon S3 存储桶，然后从该存储桶部署代码。不过，在支持此过程的初始设置完成后，您从 Bitbucket 推送的代码将自动部署到您的实例，而无需任何中间步骤。

了解更多：

- [Atlassian Bitbucket 对 AWS CodeDeploy 的支持](#)

Note

AWS Lambda 部署仅适用于 Amazon S3 存储库。

将 AWS CodeDeploy 的修订推送到 Amazon S3

在规划修订（如[计划 AWS CodeDeploy 的修订 \(p. 208\)](#)中所述）并将 AppSpec file 添加到修订（如[将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)中所述）后，您便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 AWS CodeDeploy 将修订从 Amazon S3 部署到实例。

Note

AWS 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[AWS CodeDeploy 入门 \(p. 16\)](#)中的说明来设置 AWS CLI。这对于调用稍后描述的 push 命令来说特别重要。

请确保您拥有 Amazon S3 存储桶。遵循[创建存储桶](#)中的说明。

如果要部署到 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，如果您需要将修订部署到美国东部（弗吉尼亚北部）地区中的一些实例和美国西部（俄勒冈）区域中的其他实例，则美国东部（弗吉尼亚北部）地区中必须有一个带修订副本的存储桶，并且美国西部（俄勒冈）区域中必须有另一个带相同修订的其他副本的存储桶。在此方案中，您随后需要分别在美国东部（弗吉尼亚北部）地区和美国西部（俄勒冈）区域中各创建一个单独的部署，即使两个区域和存储桶中的修订相同也是如此。

您必须拥有对 Amazon S3 存储桶执行上传操作的权限。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 (*) 可让 AWS 账户 111122223333 将文件上传到名为 codedeploydemobucket 的 Amazon S3 存储桶中的任何目录：

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

```
}
```

To view your AWS account ID, see [Finding Your AWS Account ID](#).

要了解如何生成和附加 Amazon S3 存储桶策略，请参阅[存储桶策略示例](#)。

调用 push 命令的 IAM 用户必须至少具有将修订上传到每个目标 Amazon S3 存储桶的权限。例如，以下策略允许 IAM 用户将修订上传到名为 codedeploydemobucket 的 Amazon S3 存储桶中的任意位置：

```
{
  "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](#)。

使用 AWS CLI 推送修订

Note

push 命令会将应用程序项目和 AppSpec file 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 AWS Lambda 部署一起使用，因为前者需要作为 JSON 格式或 YAML 格式的 AppSpec file 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 AWS 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。AWS CodeDeploy 将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 file。该位置被指定为相对于当前目录的路径。如果未指定相对路径或者对路径使用了单个句点 (".")，则使用当前目录。
- `--description`：(字符串) 可选。用于概述应用程序修订的注释。如果未指定，则使用默认字符串 "Uploaded by AWS CLI 'time' UTC (由 AWS CLI '时间'UTC 上传)"，其中，"time (时间)" 是采用协调世界时 (UTC) 的当前系统时间。

您可以使用 AWS CLI 推送 Amazon EC2 部署的修订。示例推送命令如下所示：

在 Linux, macOS, or 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 AWS CLI 或 AWS CodeDeploy 控制台从 Amazon S3 部署修订。使用 AWS CLI 部署此修订：

在 Linux, macOS, or 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 --your-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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

使用 AWS CodeDeploy 查看应用程序修订详细信息

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 查看有关注册到指定应用程序的 AWS 账户的所有应用程序修订的详细信息。

有关注册修订的信息，请参阅[使用 AWS CodeDeploy 在 Amazon S3 中注册应用程序修订 \(p. 217\)](#)。

主题

- [查看应用程序修订详细信息（控制台）\(p. 216\)](#)
- [查看应用程序修订详细信息 \(CLI\) \(p. 217\)](#)

查看应用程序修订详细信息（控制台）

要查看应用程序修订详细信息，请执行以下操作：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Applications。
3. 在 Applications 页上，选择具有要查看的修订详细信息的应用程序名称。

Note

如果未显示任何条目，请确保选择了正确的区域。在导航栏上的区域选择器中，选择 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

4. 在 Application details 页上的 Revisions 下，查看为应用程序注册的修订的列表。选择修订旁的箭头以了解更多信息。

查看应用程序修订详细信息 (CLI)

要使用 AWS 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。AWS Lambda 部署的修订不能用于 GitHub。

要查看有关单个应用程序修订的详细信息，请调用 `get-application-revision` 命令，并指定：

- 应用程序名称。要获取应用程序名称，请调用 `list-applications` 命令。
- 对于存储在 GitHub 中的修订，GitHub 存储库名称和引用已推送到存储库的应用程序修订的提交的 ID。
- 对于存储在 Amazon S3 中的修订，包含修订的 Amazon S3 存储桶名称；上传的存档文件的名称和文件类型；以及（可选）存档文件的 Amazon S3 版本标识符和 ETag。如果在调用 `register-application-revision` 的过程中指定了版本标识符和/或 ETag，则必须在此处指定它们。

要查看有关多个应用程序修订的详细信息，请调用 `list-application-revisions` 命令，并指定：

- 应用程序名称。要获取应用程序名称，请调用 `list-applications` 命令。
- （可选）要仅查看 Amazon S3 应用程序修订的详细信息，则指定包含修订的 Amazon S3 存储桶名称。
- （可选）要仅查看有关 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，AWS 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，AWS 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，AWS 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 AWS CodeDeploy 在 Amazon S3 中注册应用程序修订

如果您已调用 `push` 命令将应用程序修订推送到 Amazon S3，则无需注册修订。但是，如果您通过其他方法将修订上传到 Amazon S3，并且希望在 AWS CodeDeploy 控制台中或者通过 AWS CLI 显示修订，请先按照以下步骤注册修订。

如果您已将应用程序修订推送到 GitHub 存储库，并且希望在 AWS CodeDeploy 控制台中或者通过 AWS CLI 显示修订，则还必须遵循以下步骤。

您只能使用 AWS CLI 或 AWS CodeDeploy API 在 Amazon S3 或 GitHub 中注册应用程序修订。

主题

- [使用 AWS CodeDeploy 在 Amazon S3 中注册修订 \(CLI\) \(p. 218\)](#)
- [使用 AWS CodeDeploy 在 GitHub 中注册修订 \(CLI\) \(p. 218\)](#)

使用 AWS CodeDeploy 在 Amazon S3 中注册修订 (CLI)

1. 将修订上传到 Amazon S3。
2. 调用 `register-application-revision` 命令，在命令中指定：
 - 应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
 - 有关要注册的修订的信息：
 - 包含修订的 Amazon S3 存储桶的名称。
 - 已上传修订的名称和文件类型。对于 AWS Lambda 部署，修订是用 JSON 或 YAML 编写的 AppSpec file。对于 EC2/本地 部署，修订包含 AWS CodeDeploy 将部署到您的实例的源文件版本或 AWS CodeDeploy 将在您的实例上运行的脚本版本。

Note

The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

- (可选) 修订的 Amazon S3 版本标识符。（如果未指定版本标识符，AWS CodeDeploy 将使用最新的版本。）
- (可选) 修订的 ETag。（如果未指定 ETag，则 AWS 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，在 `register-application-revision` 调用中使用以下语法来指定有关 Amazon S3 中修订的信息。（`version` 和 `eTag` 可选。）

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 AWS Lambda 部署的修订文件：

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

使用 AWS CodeDeploy 在 GitHub 中注册修订 (CLI)

Note

AWS Lambda 部署不能用于 GitHub。

1. 将修订上传到 GitHub 存储库。
2. 调用 `register-application-revision` 命令，在命令中指定：
 - 应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。

- 有关要注册的修订的信息：
 - 分配到包含修订的存储库中的 GitHub 用户或组名，后跟正斜杠 (/) 和存储库名称。
 - 引用存储库中修订的提交的 ID。
- (可选) 您要与修订关联的任何描述。

可以在命令行中，在 register-application-revision 调用中使用以下语法来指定有关 GitHub 中修订的信息：

```
--github-location repository=string,commitId=string
```

在 AWS CodeDeploy 中使用部署

在 AWS CodeDeploy 中，部署是指在一个或多个实例上安装内容的过程以及该过程中涉及的组件。此内容可包含代码、Web 和配置文件、可执行文件、程序包、脚本等。AWS CodeDeploy 根据您的指定配置规则来部署源存储库中存储的内容。

AWS CodeDeploy 提供了两个部署类型选项，即就地部署和蓝/绿部署。

- 就地部署：停止部署组中每个实例上的应用程序，安装最新的应用程序修订版，然后启动和验证应用程序的新版本。您可以使用负载均衡器，以便在部署期间取消注册每个实例，然后在部署完成后让其重新提供服务。只有使用 EC2/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 \(p. 3\)](#)。
- 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：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：
 - 系统将为替代环境配置实例。
 - 替代实例上将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来说，等待时间可选。
 - 替代环境中的实例在 Elastic Load Balancing 负载均衡器中进行注册，使得流量重新路由至这些实例。系统将撤销原始环境中的实例注册，进而终止或因其他使用情形而保持运行。

Note

在使用 EC2/本地 计算平台时，蓝/绿部署只能与 Amazon EC2 实例配合使用。

- AWS Lambda 计算平台上的蓝/绿部署：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方法。所有 AWS Lambda 计算平台部署都是蓝/绿部署。因此，您无需指定部署类型。

有关蓝/绿部署的更多信息，请参阅[蓝/绿部署概述 \(p. 4\)](#)。

有关从 Amazon S3 自动部署的信息，请参阅[使用 AWS CodeDeploy 从 Amazon S3 自动部署](#)。

主题

- [使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)
- [使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)
- [查看 AWS CodeDeploy 部署日志数据 \(p. 230\)](#)
- [使用 AWS CodeDeploy 停止部署 \(p. 232\)](#)
- [使用 AWS CodeDeploy 重新部署和回滚部署 \(p. 233\)](#)
- [在其他 AWS 账户中部署应用程序 \(p. 235\)](#)
- [使用 AWS CodeDeploy 代理验证本地机器上的部署程序包 \(p. 238\)](#)

使用 AWS CodeDeploy 创建部署

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS 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 Amazon S3 (如果您的部署目标是 EC2/本地 计算平台，则为 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 \(p. 221\)](#)

- [创建 AWS Lambda 计算平台 部署 \(控制台\) \(p. 223\)](#)
- [创建 EC2/本地 计算平台 部署 \(控制台\) \(p. 223\)](#)
- [创建 AWS Lambda 计算平台 部署 \(CLI\) \(p. 226\)](#)
- [创建 EC2/本地 计算平台 部署 \(CLI\) \(p. 227\)](#)

部署先决条件

在您开始部署之前，请确保完成以下步骤。

AWS Lambda 计算平台上的部署先决条件

- 创建一个应用程序，其中至少包括一个部署组。有关信息，请参阅 [使用 AWS CodeDeploy 创建应用程序 \(p. 189\)](#) 和 [使用 AWS CodeDeploy 创建部署组 \(p. 198\)](#)。
- 准备应用程序修订，该修订也称为 AppSpec file，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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。
- 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它。有关信息，请参阅 [Create a Deployment Configuration \(p. 186\)](#)。

EC2/本地计算平台上的部署先决条件

- 对于就地配置，创建或配置要部署到的实例。有关信息，请参阅 [使用适用于 AWS CodeDeploy 的实例 \(p. 133\)](#)。对于蓝/绿部署，您可使用现有 Auto Scaling 组作为替换环境的模板，或指定一个或多个实例或 Auto Scaling 组作为原始环境。有关更多信息，请参阅 [教程：使用 AWS CodeDeploy 将应用程序部署到 Auto Scaling 组 \(p. 86\)](#) 和 [将 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。
- 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 AWS CodeDeploy 创建应用程序 \(p. 189\)](#) 和 [使用 AWS CodeDeploy 创建部署组 \(p. 198\)](#)。
- 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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。
- 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它。有关信息，请参阅 [Create a Deployment Configuration \(p. 186\)](#)。
- 如果您正在从某个 Amazon S3 存储桶部署应用程序修订，则该存储桶应与部署组中的实例位于相同的 AWS 区域中。
- 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 存储桶策略允许从名为 codedeploydemobucket 的 Amazon S3 存储桶中的任意位置，下载附加了 IAM 实例配置文件 (其中包含 ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo`) 的任意 Amazon EC2 实例：

```
{
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::codedeploydemobucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo"
        ]
      }
    }
  ]
}
```

```
}  
  ]  
}
```

以下 Amazon S3 存储桶策略允许从名为 `codedeploydemobucket` 的 Amazon S3 存储桶中的任意位置，下载具有关联 IAM 用户（其中包含 ARN `arn:aws:iam::80398EXAMPLE:user/CodeDeployUser`）的任意本地实例：

```
{  
  "Statement": [  
    {  
      "Action": [  
        "s3:Get*",  
        "s3:List*"  
      ],  
      "Effect": "Allow",  
      "Resource": "arn:aws:s3:::codedeploydemobucket/*",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::80398EXAMPLE:user/CodeDeployUser"  
        ]  
      }  
    }  
  ]  
}
```

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](#)。

- 如果您正在创建蓝/绿部署，或者您已为就地部署在部署组中指定可选 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer，则您已使用包含至少两个子网的 Amazon VPC 创建了 VPC。（AWS 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](#)。

- 如果您正在创建蓝/绿部署，则您已在 Elastic Load Balancing 中配置 传统负载均衡器、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 传统负载均衡器，请完成 Classic Load Balancer 用户指南中的[教程：创建 传统负载均衡器](#) 中的步骤。在操作过程中，记下以下内容：

- 在步骤 2：定义负载均衡器中的创建内部负载均衡器中，选择创建实例时所选的统一 VPC。
- 在步骤 5：向负载均衡器注册 EC2 实例中，选择原始环境中的实例。
- 在步骤 7：创建并验证您的负载均衡器中，记录负载均衡器的 DNS 地址。

例如，如果您已将负载均衡器命名为 `my-load-balancer`，则您的 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 名称粘贴到已连接 Internet 的 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 应用程序负载均衡器](#)
- [教程：使用 AWS CLI 创建 应用程序负载均衡器](#)

要配置 Network Load Balancer，请遵循以下某一主题中的说明：

- [创建 Network Load Balancer](#)
- [教程：使用 AWS CLI 创建 Network Load Balancer](#)

创建 AWS Lambda 计算平台 部署 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 执行以下任一操作：

- 在 AWS CodeDeploy 菜单上，选择 Deployments，然后选择 Create deployment。
 - 在 AWS CodeDeploy 菜单上，选择 Applications，然后选择要将修订部署到的 AWS Lambda 应用程序的名称。您可以在 计算平台 列中查看，以识别 AWS Lambda 应用程序。在 Application details 页面上，选择要将修订部署到的部署组的按钮。在 Actions 菜单上，选择 Deploy new revision。
3. 在 Application 列表中，选择要用于此部署的应用程序的名称。选择您的应用程序后，请确保 Application 列表下方的 计算平台 指明 AWS Lambda。
 4. 在 Deployment group 列表中，选择与应用程序关联的部署组的名称。
 5. 请注意 Deployment type 标签。AWS Lambda 部署都是蓝/绿部署。
 6. 在 Revision location 旁边，选择您的修订所在的位置：
 - My revision is stored in Amazon S3 - 有关信息，请参阅[指定存储在 Amazon S3 存储桶中的修订的相关信息 \(p. 225\)](#)，然后返回步骤 6。
 - I will use the AppSpec editor - 选择 JSON 或 YAML，然后将您的 AppSpec file 键入到提供的编辑器中。您可以通过选择 Save as text file 来保存键入的 AppSpec file。如果您在这些步骤结束时单击 Deploy，并且您的 JSON 或 YAML 无效，您将收到错误。有关创建 AppSpec file 的更多信息，请参阅[将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#)。
 7. (可选) 在 Deployment description 框中，键入此部署的说明。
 8. 在 Deployment configuration 列表中，选择部署配置以控制流量如何转移到 Lambda 函数版本。

有关更多信息，请参阅 [AWS Lambda 计算平台上的部署配置 \(p. 185\)](#)。

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。

Note

有关在 AWS CodeDeploy 中回滚的信息，请参阅[重新部署和部署回滚 \(p. 10\)](#)和[使用 AWS CodeDeploy 重新部署和回滚部署 \(p. 233\)](#)。

从以下选项中进行选择：

- Roll back when a deployment fails — AWS 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 — 如果为部署组添加了警报，当一个或多个指定警报激活时，AWS CodeDeploy 将重新部署上一个已知良好的版本。
 - Disable rollbacks — 不为此部署执行回滚。
10. 选择 Deploy。

要跟踪部署的状态，请参阅[使用 AWS CodeDeploy 查看部署详细信息 \(p. 229\)](#)。

创建 EC2/本地 计算平台 部署 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 执行以下任一操作：

- 在 AWS CodeDeploy 菜单上，选择 Deployments，然后选择 Create deployment。
 - 在 AWS CodeDeploy 菜单上，选择 Applications，然后选择要将修订部署到的 EC2/本地 应用程序的名称。您可以在 Compute platform 列中查看，以识别 EC2/本地 应用程序。在 Application details 页面上，选择要将修订部署到的部署组的按钮。在 Actions 菜单上，选择 Deploy new revision。
3. 在 Application 列表中，选择要用于此部署的应用程序的名称。确保已为 Compute platform 显示 EC2/本地。
4. 在 Deployment group 列表中，选择与应用程序关联的部署组的名称。
5. 在 Repository type 旁边，选择保存您的修订的存储库类型：
- My application is stored in Amazon S3 — 有关信息，请参阅[指定存储在 Amazon S3 存储桶中的修订的相关信息 \(p. 225\)](#)，然后返回步骤 6。
 - My application is stored in GitHub — 有关信息，请参阅下方的[指定有关存储在 GitHub 存储库中修订的信息 \(p. 225\)](#)，然后返回步骤 6。
6. (可选) 在 Deployment description 框中，键入此部署的说明。
7. 在 Deployment configuration 列表中，选择部署配置来控制使用新应用程序修订更新实例的速率 (就地部署) 或将流量路由到替换环境的速率 (蓝/绿部署)。

有关更多信息，请参阅 [在 AWS CodeDeploy 中使用部署配置 \(p. 184\)](#)。

8. 在 Content options 中，您可以指定 AWS CodeDeploy 如何处理部署目标位置上的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：

- Fail the deployment - 系统报告出错，并且部署状态更改为“失败”。
- Overwrite the content - 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。
- Retain the content - 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 [回滚行为与现有内容 \(p. 234\)](#)。

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。

Note

有关在 AWS CodeDeploy 中回滚的信息，请参阅[重新部署和部署回滚 \(p. 14\)](#)和[使用 AWS CodeDeploy 重新部署和回滚部署 \(p. 233\)](#)。

从以下选项中进行选择：

- Roll back when a deployment fails — AWS 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 — 如果为部署组添加了警报，当一个或多个指定警报激活时，AWS CodeDeploy 将重新部署上一个已知良好的版本。
- Disable rollbacks — 不为此部署执行回滚。

10. 选择 Deploy。

要跟踪部署的状态，请参阅[使用 AWS 版本 2014-10-06 查看部署详细信息 \(p. 229\)](#)。

主题

- 指定存储在 Amazon S3 存储桶中的修订的相关信息 (p. 225)
- 指定有关存储在 GitHub 存储库中修订的信息 (p. 225)

指定存储在 Amazon S3 存储桶中的修订的相关信息

如果您正在执行 [创建 EC2/本地 计算平台 部署 \(控制台\)](#) (p. 223) 中的步骤，请遵循这些步骤以添加有关存储在 Amazon S3 存储桶中的应用程序修订的详细信息。

1. 将您的修订的 Amazon S3 链接复制到 Revision location 框中。要查找链接值，请执行以下操作：

1. 在单独的浏览器选项卡中：

登录 AWS 管理控制台并通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

浏览并选择您的修订。

2. 如果 Properties 窗格不可见，请选择 Properties 按钮。
3. 在 Properties 窗格中，将 Link 字段的值复制到 AWS CodeDeploy 控制台中的 Revision location 框中。

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 字段值以 **?versionId=versionId** 结束，则将 **&etag=** 和 ETag 添加到 Link 字段值的结尾。
- 如果 Link 字段值未指定修订 ID，则将 **?etag=** 和 ETag 添加到 Link 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
bucket-name.s3.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 存储库中修订的信息

如果您执行 [创建 EC2/本地 计算平台 部署 \(控制台\)](#) (p. 223) 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 中，执行下列操作之一：

- 要为 AWS 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 AWS 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，请在单独的 Web 浏览器选项卡中注销 GitHub。选择 Connect to a different GitHub account，然后选择 Connect to 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 页面，则选择 Authorize application。
4. 在 Create deployment 页上的 Repository name 框中，键入包含修订的 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：
 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 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，依次后跟正斜杠 (/) 和存储库的名称。将此显示的值键入 Repository name 框中。

Note

如果目标存储库名称在 Your repositories 中不可见，请使用 Search GitHub 框查找目标存储库名称和 GitHub 用户或组织名称。

5. 在 Commit ID 框中，键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：
 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 中，选择包含目标提交的存储库名称。
 3. 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。）
 4. 将提交 ID 粘贴到 Commit ID 框中。

创建 AWS Lambda 计算平台 部署 (CLI)

在您创建应用程序和修订 (在 AWS Lambda 部署中，这是 AppSpec file) 后：

调用 `create-deployment` 命令，在命令中指定：

- 应用程序名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
- 部署组名称。要查看部署组名称的列表，请调用 `list-deployment-groups` 命令。
- 有关要部署的修订的信息：

对于存储在 Amazon S3 中的修订：

- 包含修订的 Amazon S3 存储桶名称。
- 已上传修订的名称。
- (可选) 修订的 Amazon S3 版本标识符。(如果未指定版本标识符，AWS CodeDeploy 将使用最新的版本。)
- (可选) 修订的 ETag。(如果未指定 ETag，则 AWS 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。要查看部署配置的列表，请调用 `list-deployment-configs` 命令。（如果未指定，AWS 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：

```
--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[从文件加载参数](#)。

有关 AWS Lambda 部署修订模板，请参阅为 [AWS Lambda 部署添加 AppSpec 文件](#) (p. 209)。有关示例修订，请参阅 [用于 AWS Lambda 部署的 AppSpec File 示例](#) (p. 293)。

要跟踪部署的状态，请参阅[使用 AWS CodeDeploy 查看部署详细信息](#) (p. 229)。

创建 EC2/本地 计算平台 部署 (CLI)

使用 AWS CLI 将修订部署到 EC2/本地 计算平台：

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 AWS CodeDeploy 应用程序连接到 GitHub 资料档案库](#) (p. 228)中的步骤，然后继续到步骤 2。
2. 调用 [create-deployment](#) 命令，在命令中指定：
 - 应用程序名称。要查看应用程序名称的列表，请调用 [list-applications](#) 命令。
 - Amazon EC2 部署组名称。要查看部署组名称的列表，请调用 [list-deployment-groups](#) 命令。
 - 有关要部署的修订的信息：

对于存储在 Amazon S3 中的修订：

- 包含修订的 Amazon S3 存储桶名称。
- 已上传修订的名称和文件类型。

Note

The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

- (可选) 修订的 Amazon S3 版本标识符。(如果未指定版本标识符，AWS CodeDeploy 将使用最新的版本。)
- (可选) 修订的 ETag。(如果未指定 ETag，则 AWS CodeDeploy 将跳过对象验证。)

对于存储在 GitHub 中的修订：

- 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。
- 修订的提交 ID。
- (可选) 要使用的部署配置的名称。要查看部署配置的列表，请调用 [list-deployment-configs](#) 命令。(如果未指定，AWS CodeDeploy 将使用特定的默认部署配置。)
- (可选) 您是否希望在即使 ApplicationStop 部署生命周期事件失败的情况下，实例的部署仍然继续到 BeforeInstall 部署生命周期事件。
- (可选) 部署的说明。
- 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 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

在 create-deployment 调用中使用此语法，可直接在命令行上指定有关 Amazon S3 中修订的信息。（version 和 eTag 可选。）

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|zip,version=string,eTag=string
```

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 [list-application-revisions](#) 命令。

要跟踪部署的状态，请参阅[使用 AWS CodeDeploy 查看部署详细信息](#) (p. 229)。

主题

- [将 AWS CodeDeploy 应用程序连接到 GitHub 资料档案库](#) (p. 228)

将 AWS CodeDeploy 应用程序连接到 GitHub 资料档案库

您必须先授予 AWS CodeDeploy 权限代表您的 GitHub 账户与 GitHub 交互，才能首次使用 AWS CLI 从 GitHub 存储库部署应用程序。对于使用 AWS CodeDeploy 控制台的每个应用程序，必须完成一次此步骤。

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。
3. 选择 Create deployment。

Note

您不会创建新的部署。这是当前授权 AWS CodeDeploy 代表您的 GitHub 用户账户与 GitHub 交互的唯一方式。

4. 在 Application 下拉列表中，选择要与您的 GitHub 用户账户关联的应用程序。
5. 在 Deployment group 下拉列表中，选择任何可用的部署组。
6. 在 Repository type 旁边，选择 My application revision is stored in GitHub。
7. 选择 Connect to GitHub。

Note

如果您看到 Connect to a different GitHub account 链接：
您可能已经针对应用程序授予 AWS CodeDeploy 代表其他 GitHub 账户与 GitHub 进行交互的权限。
您可能已针对 AWS CodeDeploy 中链接到的所有应用程序撤消 AWS CodeDeploy 代表已登录 GitHub 账户与 GitHub 进行交互的权限。
有关更多信息，请参阅 [GitHub 对 AWS CodeDeploy 中的应用程序进行的身份验证](#) (p. 46)。

8. 如果您尚未登录 GitHub，则按照 Sign in 页面上的说明操作。
9. 在 Authorize application 页上，选择 Authorize application。

10. 现在 AWS CodeDeploy 具有权限，选择 Cancel 并继续 [创建 EC2/本地 计算平台 部署 \(CLI\)](#) (p. 227) 中的步骤。

使用 AWS CodeDeploy 查看部署详细信息

可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 查看有关与您的 AWS 账户关联的部署的详细信息。

Note

您可以在以下位置中查看您的实例的部署日志：

- Amazon Linux、RHEL 和 Ubuntu Server：`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`
- Windows Server：`C:\ProgramData\Amazon\CodeDeploy<DEPLOYMENT-GROUP-ID>\logs\scripts.log`

有关更多信息，请参阅 [分析日志文件以调查针对实例的部署失败](#) (p. 319)。

主题

- [查看部署详细信息 \(控制台\)](#) (p. 229)
- [查看部署详细信息 \(CLI\)](#) (p. 229)

查看部署详细信息 (控制台)

使用 AWS CodeDeploy 控制台查看部署详细信息：

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门](#) (p. 16).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments 以查看部署及其详细信息的列表。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

3. 要查看单个部署的更多详细信息，请在 Deployments 中选择部署 ID。

查看部署详细信息 (CLI)

要使用 AWS CLI 查看部署详细信息，请调用 `get-deployment` 命令或 `batch-get-deployments` 命令。您可以调用 `list-deployments` 命令来获取要用作 `get-deployment` 命令和 `batch-get-deployments` 命令的输入的唯一部署 ID 的列表。

要查看有关单个部署的详细信息，请调用 `get-deployment` 命令，并指定唯一部署标识符。要获取部署 ID，请调用 `list-deployments` 命令。

要查看有关多个部署的详细信息，请调用 [batch-get-deployments](#) 命令，并指定多个唯一部署标识符。要获取部署 ID，请调用 [list-deployments](#) 命令。

要查看部署 ID 的列表，请调用 [list-deployments](#) 命令，并指定：

- 与部署关联的应用程序的名称。要查看应用程序名称的列表，请调用 [list-applications](#) 命令。
- 与部署关联的部署组的名称。要查看部署组名称的列表，请调用 [list-deployment-groups](#) 命令。
- (可选) 是否按部署状态包含有关部署的详细信息。(如果未指定，则将列出所有匹配的部署，不管其部署状态如何。)
- (可选) 是否按部署创建的开始时间和/或结束时间包含有关部署的详细信息。(如果未指定，则将列出所有匹配的部署，不管其创建时间如何。)

查看 AWS CodeDeploy 部署日志数据

您可以通过以下方式查看 AWS 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。

主题

- [在 Amazon CloudWatch 控制台中查看日志文件数据 \(p. 230\)](#)
- [查看实例上的日志文件 \(p. 230\)](#)

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 日志代理的信息，请参阅[在 CloudWatch Logs 控制台中查看 AWS CodeDeploy 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 Server 实例上的部署日志文件 \(p. 230\)](#)
- [查看 Windows Server 实例上的部署日志文件 \(p. 231\)](#)

查看 Amazon Linux、RHEL 和 Ubuntu Server 实例上的部署日志文件

在 Amazon Linux、RHEL 和 Ubuntu Server 实例上，部署日志存储在以下位置：

```
/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log
```

要查看或分析 Amazon Linux、RHEL 和 Ubuntu Server 实例上的部署日志，请登录实例，然后键入以下命令打开 AWS 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
& ##	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。
/ ##	搜索下一条错误消息。 ¹
? ##	搜索上一条错误消息。 ² 在 ERROR 一词的前后使用一个空格。
G	转到日志文件的末尾。
g	转到日志文件的开头。
q	退出日志文件。
h	了解其他命令。
¹ 在键入 / ERROR 后，为下一条错误消息键入 n 。为上一条错误消息键入 N 。	
² 在键入 ? ERROR 后，为下一条错误消息键入 n ，或为上一条错误消息键入 N 。	

您也可以键入以下命令来打开 AWS 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 在键入 /stderr 后，为下一条错误消息键入 n 。为上一条错误消息键入 N 。	
² 在键入 ?stderr 后，为下一条错误消息键入 n 。为上一条错误消息键入 N 。	

查看 Windows Server 实例上的部署日志文件

AWS CodeDeploy 代理日志文件：在 Windows Server 实例上，AWS 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 AWS 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。

AWS 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\ideployment-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 AWS CodeDeploy 脚本日志文件：

```
notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\ideployment-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 AWS CodeDeploy 停止部署

您可以使用 AWS CodeDeploy 控制台、AWS CLI 或 AWS CodeDeploy API 停止与 AWS 账户关联的部署。

Warning

停止 EC2/本地 部署可能会使您的部署组中的部分或全部实例处于不确定的部署状态。有关更多信息，请参阅 [停止和失败的部署 \(p. 14\)](#)。

主题

- [停止部署 \(控制台\) \(p. 232\)](#)
- [停止部署 \(CLI\) \(p. 233\)](#)

停止部署 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 AWS CodeDeploy 菜单上，选择 Deployments。

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 AWS General Reference 的 [区域和终端节点](#) 中列出的某个区域。AWS CodeDeploy 仅在这些区域中受支持。

3. 在要停止的部署的 Actions 列中，选择 Stop。

Note

如果 Actions 列中没有显示 Stop 按钮，则表示部署已进展到无法停止的时间点。

停止部署 (CLI)

调用 `stop-deployment` 命令，并指定部署 ID。要查看部署 ID 的列表，请调用 `list-deployments` 命令。

使用 AWS CodeDeploy 重新部署和回滚部署

AWS CodeDeploy 回滚部署的方式是将以前的应用程序部署版本重新部署为新的部署。这些回滚部署在技术上属于新部署，具有新的部署 ID，不同于之前部署的存储版本。

可以自动或手动回滚部署。

主题

- [自动回滚 \(p. 233\)](#)
- [手动回滚 \(p. 233\)](#)
- [回滚和重新部署工作流程 \(p. 233\)](#)
- [回滚行为与现有内容 \(p. 234\)](#)

自动回滚

您可以对部署组或部署进行配置，使之在部署失败或达到您指定的监控阈值时自动回滚。在这种情况下，将会部署上一个已知良好的应用程序版本。您可以在创建应用程序或是创建或更新部署组时配置自动回滚。

创建新部署时，您还可以选择覆盖已为部署组指定的自动回滚配置。

Note

可使用 Amazon Simple Notification Service 在部署自动回滚时接收通知。有关信息，请参阅 [Monitoring Deployments with Amazon SNS Event Notifications \(p. 250\)](#)。

有关配置自动回滚的更多信息，请参阅[部署组配置高级选项 \(p. 206\)](#)。

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 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

Note

如果您从部署组中删除某个实例，AWS 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，AWS CodeDeploy 首先尝试从每个参与实例中删除上次成功安装的所有文件。为此，AWS CodeDeploy 将检查下面的清除文件：

```
/opt/codedeploy-agent/deployment-root/deployment-instructions/deployment-group-ID-cleanup 文件 (对于 Amazon Linux、Ubuntu Server 和 RHEL 实例)
```

```
C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\deployment-group-ID-cleanup 文件 (对于 Windows Server 实例)
```

如果清除文件存在，则 AWS CodeDeploy 将在开始新部署之前，使用它从实例中删除列出的所有文件。

例如，前面的两个文本文件和两个脚本文件已部署到运行 Windows Server 的 Amazon EC2 实例，并且这两个脚本在部署生命周期事件期间又创建了两个文本文件：

```
c:\temp\a.txt (previously deployed by AWS CodeDeploy)
c:\temp\b.txt (previously deployed by AWS CodeDeploy)
c:\temp\c.bat (previously deployed by AWS CodeDeploy)
c:\temp\d.bat (previously deployed by AWS 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，AWS 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，AWS CodeDeploy 将不会在后续重新部署（无论是手动还是自动回滚）期间尝试还原或以其他方式协调在以前的部署中由任何脚本所执行的任何操作。例如，如果 c.bat 和 d.bat 文件包含不重新创建 e.txt 和 f.txt 文件（如果它们已存在）的逻辑，则每当 AWS CodeDeploy 在后续部署中运行 c.bat 和 d.bat 时，旧版本的 e.txt 和 f.txt 都将保持不变。您可以向 c.bat 和 d.bat 中添加逻辑，始终先检查并删除旧版本的 e.txt 和 f.txt，然后再创建新文件。

回滚行为与现有内容

作为部署过程的一部分，AWS 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 AWS CodeDeploy 在下一个部署期间对这些文件执行哪些操作：

- Fail the deployment — 系统报告出错，并且部署状态更改为“失败”。
- Overwrite the content — 来自应用程序修订的文件版本将替换实例上已有的版本。
- Retain the content — 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 AWS 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	AWS 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，AWS 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，AWS CodeDeploy 将部署上一个已知正常的应用程序修订 (应用程序修订 1)。 不过，您希望在部署 1 中保留的文件已在部署 2 失败之前被删除，且无法由 AWS CodeDeploy 检索。您可以自行将这些文件添加到实例 (如果应用程序修订 1 需要这些文件)，也可以创建新的应用程序修订。

在其他 AWS 账户中部署应用程序

组织通常会有多个 AWS 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，不过 AWS 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 AWS 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 AWS Security Token Service (AWS STS) 提供的功能，该功能可授予您对该账户的临时访问权限。

步骤 1：在任一账户中创建 S3 存储桶

在开发账户或生产账户中：

- 如果您还未创建，则创建将在其中存储生产账户的应用程序修订的 Amazon S3 存储桶。有关信息，请参阅在 [Amazon S3 中创建存储桶](#)。您甚至可以两个账户使用相同的存储桶和应用程序修订，将您在开发账户中测试和验证的相同文件部署到您的生产环境。

步骤 2：将 Amazon S3 存储桶权限授予生产账户的 IAM 实例配置文件

如果您在步骤 1 中创建的 Amazon S3 存储桶位于您的生产账户中，则不需要此步骤。稍后您将承担的角色具有对此存储桶的访问权限，因为该存储桶也位于生产账户中。

如果您在开发账户中创建了 Amazon S3 存储桶，则执行以下操作：

- 在生产账户中，创建 IAM 实例配置文件。有关信息，请参阅 [步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#)。

Note

记录此 IAM 实例配置文件的 ARN。您需要将它添加到接下来创建的跨存储桶策略中。

- 在开发账户中，将您在开发账户中创建的 Amazon S3 存储桶的访问权限授予刚刚在生产账户中创建的 IAM 实例配置文件。有关信息，请参阅 [示例 2：存储桶所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。

role-name 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。

步骤 3：在生产账户中创建资源和跨账户角色

在生产账户中：

- 按照本指南中提供的说明创建 AWS 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 AWS CodeDeploy 操作。

使用[演练：使用 IAM 角色委派跨 AWS 账户的访问权限](#)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，而是至少应将以下两个 AWS 提供的策略附加到角色：

- **AmazonS3FullAccess**：只有当 S3 存储桶位于开发账户中时才需要。提供对开发账户（修订存储在其中的）中的 Amazon S3 服务和资源具有完整访问权限的已代入生产账户角色。
- **AWSCodeDeployDeployerAccess**：允许 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 AWS CodeDeploy 任务授予权限的更多信息，请参阅[适用于 AWS CodeDeploy 的 AWS 托管 \(预定义\) 策略 \(p. 266\)](#)。

如果您希望在使用此跨账户角色时在其他 AWS 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 AWS 管理控制台 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 AWS 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。

步骤 4：将应用程序修订上传到 Amazon S3 存储桶

在您创建了 Amazon S3 存储桶的账户中：

- 将您的应用程序修订上传到 Amazon S3 存储桶。有关信息，请参阅[将 AWS CodeDeploy 的修订推送到 Amazon S3 \(p. 214\)](#)。

步骤 5：代入跨账户角色和部署应用程序

在开发账户中，您可以使用 AWS CLI 或 AWS 管理控制台 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 AWS 管理控制台 来切换角色和启动部署的说明，请参阅[切换到角色 \(AWS 管理控制台\)](#) 和 [创建 EC2/本地 计算平台 部署 \(控制台\)](#) (p. 223)。

有关如何使用 AWS CLI 来代入跨账户角色和启动部署的说明，请参阅[切换到 IAM 角色 \(AWS 命令行界面\)](#) 和 [创建 EC2/本地 计算平台 部署 \(CLI\)](#) (p. 227)。

有关通过 AWS STS 代入角色的更多信息，请参阅 [AWS Security Token Service 用户指南](#) 中的 [AssumeRole](#) 和 [AWS CLI 命令参考](#) 中的 [assume-role](#)。

相关主题：

- [AWS CodeDeploy：从开发账户部署到生产账户](#)

使用 AWS CodeDeploy 代理验证本地机器上的部署程序包

您可以使用 AWS 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 file) 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 AWS 账户。对于最简单的测试，您可以在包含 AppSpec file 和要部署内容的目录中运行 `codedeploy-local` 命令，而无需指定任何选项。工具中还有适用于其他测试用例的选项。

通过验证本地机器上的部署程序包，您可以：

- 测试应用程序修订的完整性。
- 测试 AppSpec file 的内容。
- 利用您现有的应用程序代码首次试用 AWS CodeDeploy。
- 登录实例后快速部署内容。

您要部署的内容可以存储在本地实例中，或存储在受支持的远程存储库类型 (Amazon S3 存储桶或公有 GitHub 存储库) 中。

先决条件

在开始本地部署之前，请先完成以下步骤：

- 创建或使用 AWS CodeDeploy 代理支持的实例类型。有关信息，请参阅 [AWS CodeDeploy 代理支持的操作系统](#) (p. 113)。
- 安装 AWS CodeDeploy 代理的 1.0.1.1352 版本或更高版本。有关信息，请参阅 [安装或重新安装 AWS CodeDeploy 代理](#) (p. 121)。
- 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 AWS CodeDeploy 中使用的 IAM 用户。有关信息，请参阅 [步骤 1：预置 IAM 用户](#) (p. 16)。
- 如果您要从 Amazon S3 存储桶部署应用程序修订，请在您工作的区域创建一个 Amazon S3 存储桶，并为该存储桶应用 Amazon S3 存储桶策略。此策略为您的实例授予下载应用程序修订所需的权限。

例如，以下 Amazon S3 存储桶策略允许从名为 `codedeploydemobucket` 的 Amazon S3 存储桶中的任意位置，下载附加了 IAM 实例配置文件 (其中包含 ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo`) 的任意 Amazon EC2 实例：


```
{
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::codedeploydemobucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo"
        ]
      }
    }
  ]
}
```

以下 Amazon S3 存储桶策略允许从名为 codedeploydemobucket 的 Amazon S3 存储桶中的任意位置，下载具有关联 IAM 用户（其中包含 ARN `arn:aws:iam::80398EXAMPLE:user/CodeDeployUser`）的任意本地实例：

```
{
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::codedeploydemobucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:user/CodeDeployUser"
        ]
      }
    }
  ]
}
```

有关如何生成和附加 Amazon S3 存储桶策略的信息，请参阅[存储桶策略示例](#)。

- 如果您要从 Amazon S3 存储桶或 GitHub 存储库部署您的应用程序修订，请设置 IAM 实例配置文件并附加到实例。有关信息，请参阅[步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件](#) (p. 22)、[为 AWS CodeDeploy 创建 Amazon EC2 实例](#) (AWS CLI 或 [Amazon EC2 控制台](#)) (p. 141)和[为 AWS CodeDeploy 创建 Amazon EC2 实例](#) (AWS CloudFormation 模板) (p. 147)。
- 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 AWS CodeDeploy 的应用程序修订](#) (p. 208)。
- 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（对于 Amazon Linux、RHEL 或 Ubuntu Server 实例，该文件为 `/etc/codedeploy-agent/conf/codedeployagent.yml`；对于 Windows Server 实例，该文件为 `c:\ProgramData\Amazon\CodeDeploy\conf.yml`）。有关信息，请参阅[AWS CodeDeploy 代理配置参考](#) (p. 297)。

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 AWS 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：

- 对于 Amazon Linux、RHEL 或 Ubuntu Server：`/opt/codedeploy-agent/bin`。
- 对于 Windows Server：`C:\ProgramData\Amazon\CodeDeploy\bin`。

基本的命令语法

```
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>]
```

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 [AWS CodeDeploy API Reference](#) 中的 [fileExistsBehavior](#)。

`-g, --deployment-group`

要部署内容的目标位置的文件夹路径。如果您不指定文件夹, 该工具将在部署根目录中创建一个名为 default-local-deployment-group 的文件夹。对于您创建的每个本地部署, 该工具都会在此文件夹中创建一个子目录, 名称示例为 d-98761234-local。

`-e, --events`

您要运行的一组覆盖生命周期事件挂钩 (按顺序), 而不是您在 AppSpec file 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:

- 您希望在不更新 AppSpec file 的情况下运行另一组事件。
- 您希望将单一事件挂钩作为 AppSpec file 中的例外来运行, 例如 `ApplicationStop`。

如果您没有在覆盖列表中指定 DownloadBundle 和 Install 事件, 它们会在您指定的所有事件挂钩之前运行。如果您在 `--events` 选项列表中包含 DownloadBundle 和 Install, 只有 AWS 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. 284)。

`-c, --agent-configuration-file`

要用于部署的配置文件的位置 (存储位置与默认位置不同时)。配置文件指定部署中默认值和默认行为的替代值和行为。

默认情况下, 配置文件存储在 `/etc/codedeploy-agent/conf/codedeployagent.yml` (Amazon Linux、RHEL 或 Ubuntu Server 实例) 或 `C:/ProgramData/Amazon/CodeDeploy/conf.yml` (Windows Server) 中。有关更多信息, 请参阅 [AWS CodeDeploy 代理配置参考](#) (p. 297)。

`-h, --help`

显示帮助内容的摘要。

`-v, --version`

显示工具的版本号。

示例

以下是有效命令格式的示例。

```
codedeploy-local
```

```
codedeploy-local --bundle-location /path/to/local/bundle/directory
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-group my-deployment-group
```

```
codedeploy-local --bundle-location /path/to/local/directory --type directory --deployment-group my-deployment-group
```

从 Amazon S3 部署数据包:

```
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
```

在 AWS CodeDeploy 中监控部署

监控是保持 AWS CodeDeploy 和 AWS 解决方案的可靠性、可用性和性能的重要环节。您应从 AWS 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。但是，在开始监控 AWS CodeDeploy 之前，您应创建一个可以回答以下问题的监控计划：

- 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？

接下来，通过在不同时间和不同负载条件下衡量性能，在您的环境中建立为正常的 AWS CodeDeploy 性能建立基准。监控 AWS CodeDeploy 时，应将历史监控数据存储下来，以便可以将其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 AWS 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 AWS 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

AWS 为您提供了各种可用于监控 AWS 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 AWS CodeDeploy 并在出现错误时进行报告：

- 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。该操作是向 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略发送通知。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 \(p. 244\)](#)。

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 AWS CodeDeploy 服务角色授予 CloudWatch 权限 \(p. 245\)](#)。有关向您的 AWS CodeDeploy 操作添加 CloudWatch 警报监控的信息，请参阅[使用 AWS CodeDeploy 创建应用程序 \(p. 189\)](#)、[使用 AWS CodeDeploy 创建部署组 \(p. 198\)](#)或[使用 AWS CodeDeploy 更改部署组设置 \(p. 204\)](#)。

- Amazon CloudWatch Logs – 监控、存储和访问来自 AWS CloudTrail 或其他来源的日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的[监视日志文件](#)。

有关使用 CloudWatch 控制台查看 AWS CodeDeploy 日志的信息，请参阅在 [CloudWatch Logs 控制台中查看 AWS CodeDeploy 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 Amazon CloudWatch 用户指南 中的[什么是 Amazon CloudWatch 事件](#)。

有关在 AWS CodeDeploy 操作中使用 CloudWatch Events 的信息，请参阅[使用 Amazon CloudWatch Events 监控部署 \(p. 246\)](#)。

- AWS 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，在 Java 中编写日志处理应用程序，以及验证您的日志文件在被 CloudTrail 交付后未发生更改。有关更多信息，请参阅 AWS CloudTrail User Guide 中的[使用 CloudTrail 日志文件](#)。

有关通过 AWS CodeDeploy 使用 CloudTrail 的信息，请参阅[Monitoring Deployments with AWS CloudTrail \(p. 248\)](#)。

- 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 AWS CodeDeploy 设置 Amazon SNS 通知的信息，请参阅[Monitoring Deployments with Amazon SNS Event Notifications \(p. 250\)](#)。

手动监控工具

监控 AWS CodeDeploy 时的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。AWS CodeDeploy、CloudWatch 和其他 AWS 控制台仪表板均提供 AWS 环境状态的概览视图。建议您还要查看 AWS CodeDeploy deployments 上的日志文件。

- AWS 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 AWS 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 \(p. 244\)](#)
- [Monitoring Deployments with AWS CloudTrail \(p. 248\)](#)
- [Monitoring Deployments with Amazon SNS Event Notifications \(p. 250\)](#)

使用 Amazon CloudWatch 工具监控部署

您可以使用以下 CloudWatch 工具监视 AWS CodeDeploy deployments：Amazon CloudWatch Events、CloudWatch 警报和 Amazon CloudWatch Logs。

通过查看 AWS 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 AWS CodeDeploy 日志的替代方法，您可以使用 CloudWatch Logs 在一个中心位置监控所有日志。

有关使用 CloudWatch 控制台查看 AWS CodeDeploy 日志的信息，请参阅[在 CloudWatch Logs 控制台中查看 AWS CodeDeploy 日志](#)。

有关使用 CloudWatch 警报和 CloudWatch Events 监控 AWS CodeDeploy deployments 的信息，请参阅以下主题。

主题

- [在 AWS CodeDeploy 中使用 CloudWatch 警报监控部署 \(p. 245\)](#)
- [使用 Amazon CloudWatch Events 监控部署 \(p. 246\)](#)

在 AWS CodeDeploy 中使用 CloudWatch 警报监控部署

您可以在 AWS CodeDeploy 操作中为当前使用的实例或 Amazon EC2 Auto Scaling 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每隔若干时间段执行一项或多项操作。CloudWatch 警报不会仅仅因为处于特定的状态就调用操作；该状态必须改变并在指定数量的时间段内一直保持。

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，例如发送 Amazon SNS 通知或停止、终止、重启或恢复实例。对于您的 AWS CodeDeploy 操作，您可以对部署组进行配置，使之每当与该部署组关联的任何 CloudWatch 警报激活时停止部署。

您可以为一个 AWS CodeDeploy 部署组最多关联十个 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，必须为您的 AWS 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 Amazon CloudWatch 用户指南 中的[创建 Amazon CloudWatch 警报](#)。

有关在 AWS CodeDeploy 中为部署组关联 CloudWatch 警报的信息，请参阅[使用 AWS CodeDeploy 创建部署组 \(p. 198\)](#)和[使用 AWS CodeDeploy 更改部署组设置 \(p. 204\)](#)。

主题

- [向 AWS CodeDeploy 服务角色授予 CloudWatch 权限 \(p. 245\)](#)

向 AWS CodeDeploy 服务角色授予 CloudWatch 权限

在为部署使用 CloudWatch 警报监控之前，您在 AWS CodeDeploy 操作中使用的服务角色必需获得访问 CloudWatch 资源的权限。

向服务角色授予 CloudWatch 权限

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 Roles。
3. 选择您在 AWS 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–

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 AWS 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”）并对这些更改采取相应的操作。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 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。在您的 AWS CodeDeploy 操作中使用 CloudWatch Events 时，您可以选择以下类型的目标：

- AWS Lambda 函数
- Kinesis 流
- Amazon SQS 队列
- 内置目标（CloudWatch 警报操作）
- Amazon SNS 主题

下面是一些使用案例：

- 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 AWS CodeDeploy 创建 CloudWatch Events 规则的基本步骤。然而，在创建用于 AWS CodeDeploy 操作的事件规则之前，您应当执行以下操作：

- 完成 CloudWatch Events 先决条件。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 Events – 跟踪和响应对您的 AWS 资源的更改](#)。
- 创建将在您的事件规则中使用的目标。

为 AWS CodeDeploy 创建 CloudWatch Events 规则：

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 Create rule，然后在 Event selector 下选择 AWS 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 AWS 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。

下面所示为一个事件规则的设置，每当名为 MyCodeDeployApp 的应用程序的 MyDeploymentFleet 部署组中有任何实例部署失败时，都会处理该事件规则：

Event selector

Build a pattern that selects events for processing by your targets.

AWS CodeDeploy

☐ Any detail type ☒ Specific detail type

CodeDeploy Deployment State-change Notification

☐ Any state ☒ Specific state(s)

x FAILURE

☐ Any application ☒ Specific application

MyCodeDeployApp

☐ Any deployment group ☒ Specific deployment group(s)

x MyDeploymentFleet

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 AWS CloudTrail 监视部署

AWS CodeDeploy 与 CloudTrail 集成在一起，后者是一种服务，它在 AWS 账户中捕获由 AWS CodeDeploy 自身或代表其发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 从 AWS CodeDeploy 控制台、通过 AWS CLI 从 AWS CodeDeploy 命令或直接从 AWS CodeDeploy API 捕获 API 调用。通过 CloudTrail 收集的信息，您可以确定向 AWS CodeDeploy 发出了什么请求、发出请求的源 IP 地址、何人发出的请求以及发出请求的时间等。要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和启用），请参阅 [AWS CloudTrail User Guide](#)。

CloudTrail 中的 AWS CodeDeploy 信息

在您的 AWS 账户中启用 CloudTrail 日志记录后，将在日志文件中跟踪对 AWS CodeDeploy 操作执行的 API 调用。AWS CodeDeploy 记录与其他 AWS 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建新文件并向其写入内容。

所有 AWS CodeDeploy 操作都会被记录并正式记载到 [AWS CodeDeploy Command Line Reference](#) 和 [AWS CodeDeploy API Reference](#) 中。例如，创建部署、删除应用程序和注册应用程序修订的调用会在 CloudTrail 日志文件中生成条目。

每个日志条目都包含有关生成请求的人员的信息。日志中的用户身份信息有助于确定发出的请求是否具有根或 IAM 用户证书，是否具有角色或联合用户的临时安全证书，或者是否是由其他 AWS 服务发出的。有关更多信息，请参阅 [CloudTrail 事件参考](#) 中的 `userIdentity` 字段。

日志文件可以在存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。默认情况下，将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

可选择让 CloudTrail 在传输新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅 [CloudTrail 配置 Amazon SNS 通知](#)。

您也可以将多个 AWS 区域和多个 AWS 账户中的 AWS CodeDeploy 日志文件聚合到单个 Amazon S3 存储桶中。有关更多信息，请参阅 [接收来自多个区域的 CloudTrail 日志文件](#)。

了解 AWS 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 AWS 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一个 AWS 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。这些通知将会发送到订阅了您已设置触发器操作的 Amazon SNS 主题接收人。

您可以以 SMS 消息或电子邮件形式接收有关 AWS CodeDeploy 事件的通知。您也可以按其他方式使用在指定事件发生时创建的 JSON 数据，例如发送消息到 Amazon SQS 队列或调用 AWS Lambda 中的函数。若要查看为部署和实例触发器提供的 JSON 数据的结构，请参阅 [AWS CodeDeploy 触发器的 JSON 数据格式 \(p. 258\)](#)。

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 AWS CodeDeploy 部署组创建 10 个触发器。

部署事件	实例事件
<ul style="list-style-type: none">• 成功• 失败• 已启动• 已停止• 回滚• Ready¹• 所有部署事件	<ul style="list-style-type: none">• 成功• 失败• 已启动• Ready¹• 所有实例事件
<p>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 AWS CodeDeploy 中使用部署 (p. 220)。</p>	

主题

- [向 AWS CodeDeploy 服务角色授予 Amazon SNS 权限 \(p. 251\)](#)
- [为 AWS CodeDeploy 事件创建触发器 \(p. 251\)](#)
- [在 AWS CodeDeploy 部署组中编辑触发器 \(p. 255\)](#)
- [从 AWS CodeDeploy 部署组中删除触发器 \(p. 257\)](#)
- [AWS CodeDeploy 触发器的 JSON 数据格式 \(p. 258\)](#)

向 AWS CodeDeploy 服务角色授予 Amazon SNS 权限

在您的触发器可以生成通知之前，必须向您在 AWS CodeDeploy 操作中使用的服务角色授予访问 Amazon SNS 资源的权限。

向服务角色授予 Amazon SNS 权限

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 <https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 Roles。
3. 选择您在 AWS 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–

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 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 AWS CodeDeploy 事件创建触发器

您可以创建一个针对 AWS CodeDeploy 部署或实例事件发布 Amazon Simple Notification Service (Amazon SNS) 主题的触发器。然后，当该事件发生时，关联主题的所有订阅者都将通过主题中指定的终端节点（例如 SMS 消息或电子邮件）接收通知。Amazon SNS 提供多种订阅主题的方式。

创建触发器之前，您必须设置触发器将指向的 Amazon SNS 主题。有关信息，请参阅[创建主题](#)。在创建主题时，我们建议您为主题指定一个标识其用途的名称，并采用 `Topic-group-us-west-3-deploy-fail` 或 `Topic-group-project-2-instance-stop` 这样的格式。

您也必须向 AWS CodeDeploy 服务角色授予 Amazon SNS 权限才能为您的触发器发送通知。有关信息，请参阅[向 AWS CodeDeploy 服务角色授予 Amazon SNS 权限 \(p. 251\)](#)。

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](#)。

创建触发器以发送关于 AWS CodeDeploy 事件的通知（控制台）

您可以使用 AWS CodeDeploy 控制台为 AWS 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 AWS CodeDeploy 事件创建触发器

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 Applications 页面上，选择将为其发送触发器的应用程序的名称。
3. 在 Applications details 页面上，选择将为其发送触发器的部署组旁边的箭头。
4. 在 Triggers 区域中，选择 Create trigger。
5. 在 Create trigger 窗格中，执行以下操作：
 - 在 Trigger name 中，为触发器键入一个便于标识其用途的名称。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。
 - 在 Events 中，选择将触发 Amazon SNS 主题发送通知的事件类型。
 - 在 Amazon SNS topic 中，选择您为发送此触发器的通知而创建的主题的名称。
6. 选择 Create trigger。

AWS CodeDeploy 将发送一条测试通知，以确认您已正确配置 AWS CodeDeploy 和 Amazon SNS 主题之间的访问权限。如果您已订阅该主题，您将收到通过 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。

创建触发器以发送关于 AWS 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，然后使用 --cli-input-json 选项运行 `create-deployment-group` 命令。

创建 JSON 文件最简单的方式是使用 --generate-cli-skeleton 选项获取 JSON 格式的副本，然后在纯文本编辑器中提供所需的值。

1. 运行以下命令，然后将结果复制到纯文本编辑器中。

```
aws deploy create-deployment-group --generate-cli-skeleton
```

2. 将现有 AWS 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`：在您的账户中为 AWS CodeDeploy 设置的现有服务角色的 ARN。有关信息，请参阅 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)。

在 `triggerConfigurations` 部分，为以下参数提供值：

- `triggerName`：您要为触发器给定的便于标识的名称。我们建议采用 `Trigger-group-us-west-3-deploy-fail` 或 `Trigger-group-eu-central-instance-stop` 这样的格式。
- `triggerTargetArn`：您创建的要与触发器关联的 Amazon SNS 主题的 ARN，采用以下格式：`arn:aws:sns:us-east-2:80398EXAMPLE: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，该触发器都将提示发送通知：

```
{
  "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-2:80398EXAMPLE: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

Be sure to include `file://` before the file name. It is required in this command.

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

在创建过程结束时，您将收到一条测试通知消息，指示权限和触发器详细信息均已正确设置。

创建触发器以发送关于现有部署组的通知

要使用 AWS CLI 将 AWS CodeDeploy 事件的触发器添加到某个现有部署组，请创建 JSON 文件以更新部署组，然后使用 `--cli-input-json` 选项运行 `update-deployment-group` 命令。

创建 JSON 文件最简单的方式是运行 `get-deployment-group` 命令以获取部署组配置的副本（采用 JSON 格式），然后在纯文本编辑器中更新参数值。

1. 运行以下命令，然后将结果复制到纯文本编辑器中。

```
aws deploy get-deployment-group --application-name application --deployment-group-
name deployment-group
```

2. 从输出中删除以下内容：

- 在输出的开头处，删除 `{ "deploymentGroupInfo":`。
- 在输出的结尾处，删除 `}`。
- 删除包含 `deploymentGroupId` 的行。
- 删除包含 `deploymentGroupName` 的行。

现在，您的文本文件的内容看起来应类似于以下内容：

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentConfigName": "CodeDeployDefault.OneAtATime",
  "autoScalingGroups": [],
  "ec2TagFilters": [
    {
      "Type": "KEY_AND_VALUE",
      "Value": "Project-ABC",
      "Key": "Name"
    }
  ]
}
```

```
    },
  ],
  "triggerConfigurations": [],
  "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
  "onPremisesInstanceTagFilters": []
}
```

3. 在 `triggerConfigurations` 部分,为 `triggerEvents`、`triggerTargetArn` 和 `triggerName` 参数添加数据。有关触发器配置参数的信息,请参阅 [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,此代码都将提示发送通知。

```
{
  "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-2:80398EXAMPLE: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,然后使用 `--cli-input-json` 选项运行 `update-deployment-group` 命令。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

Be sure to include `file://` before the file name. It is required in this command.

```
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,您将收到一条测试通知消息,指示权限和触发器详细信息均已正确设置。

在 AWS CodeDeploy 部署组中编辑触发器

如果您的通知要求更改,您可以修改触发器而不必创建新的触发器。

修改 AWS CodeDeploy 触发器 (控制台)

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 Applications 页面上，选择与您将在其中修改触发器的部署组关联的应用程序的名称。
3. 在 Application details 页面上，选择您将在其中编辑触发器的部署组旁边的箭头。
4. 在 Triggers 区域中，找到您要修改的触发器的名称，然后选择其所在的行结尾处的铅笔图标。
5. 更新触发器名称、选定的事件或 Amazon SNS 主题，然后选择 Save。

修改 AWS CodeDeploy 触发器 (CLI)

若要在更新部署组时使用 AWS CLI 更改 AWS CodeDeploy 事件的触发器详细信息，请创建一个 JSON 文件以定义对部署组的属性的更改，然后使用 `--cli-input-json` 选项运行 `update-deployment-group` 命令。

创建 JSON 文件最简单的方式是运行 `get-deployment-group` 命令以获取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，然后使用 `--cli-input-json` 选项运行 `update-deployment-group` 命令。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

Be sure to include `file://` before the file name. It is required in this command.

```
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，您将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 AWS CodeDeploy 部署组中删除触发器


由于每个部署组的触发器数量限制为 10 个，因此您可能需要删除不再使用的触发器。您无法撤消删除触发器的操作，但可以重新创建一个触发器。

从部署组中删除触发器（控制台）

1. Sign in to the AWS 管理控制台 and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.

Note

Sign in with the same account or IAM user information you used in [AWS CodeDeploy 入门 \(p. 16\)](#).

2. 在 Applications 页面上，选择与您要从中删除触发器的部署组关联的应用程序。
3. 在 Application details 页面上，选择该部署组旁边的箭头。
4. 在 Triggers 区域中，查找要删除的触发器的名称，选择位于其行末尾的  按钮，然后选择 Delete。

从部署组中删除触发器 (CLI)

要使用 CLI 删除触发器，请使用空的触发器配置参数调用 `update-deployment-group` 命令，并指定：

- 与部署组关联的应用程序的名称。要查看应用程序名称的列表，请调用 `list-applications` 命令。
- 与应用程序关联的部署组的名称。要查看部署组名称的列表，请调用 `list-deployment-groups` 命令。

例如：

```
aws deploy update-deployment-group --application-name application-name --current-deployment-group-name deployment-group-name --trigger-configurations
```

AWS 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，例如发送消息到 Amazon SQS 队列或调用 AWS Lambda 中的函数。

Note

本指南不解决如何使用 JSON 配置通知的问题。有关使用 Amazon SNS 将消息发送到 Amazon SQS 队列的信息，请参阅[将 Amazon SNS 消息发送到 Amazon SQS 队列](#)。有关使用 Amazon SNS 调用 Lambda 函数的信息，请参阅[使用 Amazon SNS 通知调用 Lambda 函数](#)。

以下示例将显示适用于 AWS 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"
  }
}
```

```
} }
```

AWS CodeDeploy 的身份验证和访问控制

访问 AWS CodeDeploy 需要凭证。这些凭证必须有权访问 AWS 资源，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。下面几节提供详细的信息来说明如何使用 [AWS Identity and Access Management \(IAM\)](#) 和 AWS CodeDeploy 来帮助保护对您的资源的访问：

- [身份验证](#) (p. 260)
- [访问控制](#) (p. 261)

身份验证

您可以以下面任一类型的身份访问 AWS：

- **AWS 账户根用户** – 注册 AWS 时，您需要提供与您的 AWS 账户关联的电子邮件地址和密码。这些是您的根凭证，它们提供对您所有 AWS 资源的完全访问权限。

Important

出于安全考虑，我们建议您仅使用根凭证创建管理员用户，此类用户是对您的 AWS 账户具有完全访问权限的 IAM 用户。随后，您可以使用此管理员用户来创建具有有限权限的其他 IAM 用户和角色。有关更多信息，请参阅 IAM 用户指南中的 [IAM 最佳实践](#) 和 [创建管理员用户和组](#)。

- **IAM 用户** – [IAM 用户](#) 就是您的 AWS 账户中的一种身份，它具有特定的自定义权限（例如，用于在 AWS CodeDeploy 中向目标发送事件数据的权限）。您可以使用 IAM 用户名和密码来登录以保护 AWS 网页，如 [AWS 管理控制台](#)、[AWS 开发论坛](#) 或 [AWS Support Center](#)。

除了用户名和密码之外，您还可以为每个用户生成 [访问密钥](#)。在通过 [多个软件开发工具包之一](#) 或使用 [AWS Command Line Interface \(AWS CLI\)](#) 以编程方式访问 AWS 服务时，可以使用这些密钥。SDK 和 CLI 工具使用访问密钥对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。AWS CodeDeploy supports 签名版本 4，后者是一种用于对入站 API 请求进行身份验证的协议。有关验证请求的更多信息，请参阅 AWS General Reference 中的 [签名版本 4 签名流程](#)。

- **IAM 角色** – [IAM 角色](#) 是可在账户中创建的另一种具有特定权限的 IAM 身份。它类似于 IAM 用户，但未与特定人员关联。利用 IAM 角色，您可以获得可用于访问 AWS 服务和资源的临时访问密钥。具有临时凭证的 IAM 角色在以下情况下很有用：
- **联合身份用户访问** – 您可以不创建 IAM 用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的既有用户身份。他们被称为联合身份用户。在通过 [身份提供商](#) 请求访问权限时，AWS 将为联合用户分配角色。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** – 可以使用您账户中的 IAM 角色向另一个 AWS 账户授予对您账户的资源的访问权限。有关示例，请参阅 IAM 用户指南中的 [教程：使用 IAM 角色委派跨 AWS 账户的访问权限](#)。

- **AWS 服务访问** – 可以使用您账户中的 IAM 角色向 AWS 服务授予对您账户的资源的访问权限。例如，您可以创建一个角色，此角色允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将存储在存储桶中的数据加载到 Amazon Redshift 群集中。有关更多信息，请参阅 IAM 用户指南 中的 [创建向 AWS 服务委派权限的角色](#)。
- **在 Amazon EC2 上运行的应用程序** – 您不用将访问密钥存储在 EC2 实例中以供实例上运行的应用程序使用并发出 AWS API 请求，而是可以使用 IAM 角色管理这些应用程序的临时凭证。要将 AWS 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 [对 Amazon EC2 上的应用程序使用角色](#)。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 AWS CodeDeploy 资源。例如，您必须有权创建、查看或删除应用程序、部署、部署配置和部署组；注册、取消注册、在本地实例中添加标签或删除标签；等等。

下面几节介绍如何管理 AWS CodeDeploy 的权限。我们建议您先阅读概述。

- [管理您的 AWS CodeDeploy 资源的访问权限概述 \(p. 261\)](#)
- [为 AWS CodeDeploy 使用基于身份的策略 \(IAM 策略\) \(p. 265\)](#)
- [AWS CodeDeploy 权限参考 \(p. 269\)](#)

管理您的 AWS CodeDeploy 资源的访问权限概述

每个 AWS 资源都归某个 AWS 账户所有，创建和访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份（即：用户、组和角色）挂载权限策略，某些服务（如 AWS Lambda）也支持向资源挂载权限策略。

Note

账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅 IAM 用户指南 中的 [IAM 最佳实践](#)。

在授予权限时，您要决定谁获得权限，获得对哪些资源的权限，以及您允许对这些资源执行的具体操作。

主题

- [AWS CodeDeploy 资源和操作 \(p. 261\)](#)
- [了解资源所有权 \(p. 262\)](#)
- [管理对资源的访问 \(p. 263\)](#)
- [指定策略元素：操作、效果和委托人 \(p. 264\)](#)
- [在策略中指定条件 \(p. 265\)](#)

AWS CodeDeploy 资源和操作

在 AWS CodeDeploy 中，主要资源为 deployment group。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。AWS CodeDeploy 支持可与 deployment groups 一起使用的其他资源，包括应用程序、部署配置和实例。这些资源称为子资源。这些资源和子资源具有与其关联的唯一 Amazon 资源名称 (ARN)。有关 ARN 的详细信息，请参阅 Amazon Web Services 一般参考 中的 [Amazon 资源名称 \(ARN\) 和 AWS 服务命名空间](#)。

资源类型	ARN 格式
部署组	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :deploymentgroup/ <i>deployment-group-name</i>
应用程序	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :application/ <i>application-name</i>
部署配置	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :deploymentconfig/ <i>deployment-configuration-name</i>
实例	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :instance/ <i>instance-ID</i>
所有 AWS CodeDeploy 资源	arn:aws:codedeploy:*
指定账户在指定区域拥有的所有 AWS CodeDeploy 资源	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :*

Note

AWS 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，AWS CodeDeploy 在资源模式和规则中使用精确匹配。请在创建事件模式时务必使用正确的 ARN 字符，以使其与资源中的 ARN 语法匹配。

例如，您可以使用某个特定部署组 (*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 (*) 指定属于特定账户的所有 deployment groups，如下所示：

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup/*"
```

要指定所有资源，或者如果特定 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 AWS 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"]
```

AWS CodeDeploy 提供一组操作用来处理 AWS CodeDeploy 资源。有关可用操作的列表，请参阅 [AWS CodeDeploy 权限参考](#) (p. 269)。

了解资源所有权

AWS 账户对在该账户下创建的资源具有所有权，而无论创建资源的人员是谁。具体而言，资源所有者是对资源创建请求进行身份验证的 [委托人实体](#)（即根账户、IAM 用户或 IAM 角色）的 AWS 账户。以下示例说明了它的工作原理：

- 如果您使用 AWS 账户的根账户凭证创建规则，则您的 AWS 账户即为该 AWS CodeDeploy 资源的所有者。

- 如果您在您的 AWS 账户中创建 IAM 用户并对该用户授予创建 AWS CodeDeploy 资源的权限，则该用户可以创建 AWS CodeDeploy 资源。但是，该用户所属的 AWS 账户拥有这些 AWS CodeDeploy 资源。
- 如果您在您的 AWS 账户中创建具有创建 AWS CodeDeploy 资源的权限的 IAM 角色，则能够担任该角色的任何人都可以创建 AWS CodeDeploy 资源。该角色所属的 AWS 账户拥有这些 AWS CodeDeploy 资源。

管理对资源的访问

权限策略 规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

Note

本节讨论如何在 AWS CodeDeploy 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅[什么是 IAM？](#)（在 IAM 用户指南中）。有关 IAM 策略语法和说明的信息，请参阅 IAM 用户指南中的[AWS IAM 策略参考](#)。

附加到 IAM 身份的策略称为基于身份的策略 (IAM 策略)，而附加到资源的策略称为基于资源的策略。AWS CodeDeploy 只支持基于身份的策略 (IAM 策略)。

主题

- [基于身份的策略 \(IAM 策略\)](#) (p. 263)
- [基于资源的策略](#) (p. 264)

基于身份的策略 (IAM 策略)

您可以向 IAM 身份挂载策略。例如，您可以执行以下操作：

- 将权限策略附加到您的账户中的用户或组 - 要授予查看 AWS CodeDeploy 控制台中的应用程序、部署组和其他 AWS CodeDeploy 资源的用户权限，您可以将权限策略附加到用户或用户所属的组。
- 向角色挂载权限策略（授予跨账户权限） - 您可以向 IAM 角色挂载基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，以向其他 AWS 账户（如账户 B）或某项 AWS 服务授予跨账户权限，如下所述：
 1. 账户 A 管理员创建一个 IAM 角色，向该角色挂载授权其访问账户 A 中资源的权限策略。
 2. 账户 A 管理员可以向将账户 B 标识为能够担任该角色的委托人的角色挂载信任策略。
 3. 之后，账户 B 管理员可以委派权限，指派账户 B 中的任何用户担任该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 AWS 服务权限来担任该角色，则信任策略中的委托人也可以是 AWS 服务委托人。

有关使用 IAM 委派权限的更多信息，请参阅 IAM 用户指南中的[访问权限管理](#)。

在 AWS 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

基于资源策略所控制的能力因资源类型而异，如下表所述：

资源类型	功能
全部	查看和列出有关资源的详细信息
应用程序	创建资源
部署配置	删除资源
部署组	
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	更新资源
部署组	
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

可以创建特定的 IAM 策略来限制您账户中的用户有权访问的调用和资源，然后将这些策略与 IAM 用户关联。有关如何创建 IAM 角色和探究 AWS CodeDeploy 的 IAM 策略语句示例的更多信息，请参阅[管理您的 AWS CodeDeploy 资源的访问权限概述 \(p. 261\)](#)。

基于资源的策略

其他服务 (如 Amazon S3) 也支持基于资源的权限策略。例如，您可以将策略附加到 S3 存储桶以管理对该存储桶的访问权限。AWS CodeDeploy 不支持基于资源的策略。

指定策略元素：操作、效果和委托人

对于每个 AWS CodeDeploy 资源，该服务都定义了一组 API 操作。为授予这些 API 操作的权限，AWS CodeDeploy 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要多个操作的权限才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅[AWS CodeDeploy 资源和操作 \(p. 261\)](#) 和 [AWS CodeDeploy 权限参考 \(p. 269\)](#)。

以下是基本的策略元素：

- Resource - 您使用 Amazon 资源名称 (ARN) 来标识策略应用到的资源。有关更多信息，请参阅 [AWS CodeDeploy 资源和操作 \(p. 261\)](#)。
- Action - 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，`codedeploy:GetApplication` 权限允许执行 `GetApplication` 操作的用户权限。

- **Effect** - 用于指定当用户请求特定操作时的效果 (可以是允许或拒绝)。如果没有显式授予 (允许) 对资源的访问权限, 则隐式拒绝访问。您也可显式拒绝对资源的访问, 这样可确保用户无法访问该资源, 即使有其他策略授予了访问权限的情况下也是如此。
- **Principal** - 在基于身份的策略 (IAM 策略) 中, 附加了策略的用户是隐式委托人。对于基于资源的策略, 您可以指定要接收权限的用户、账户、服务或其他实体 (仅适用于基于资源的策略)。

要了解有关 IAM 策略语法和说明的更多信息, 请参阅 IAM 用户指南 中的 [AWS IAM 策略参考](#)。

有关显示所有 AWS CodeDeploy API 操作及其适用资源的表, 请参阅 [AWS CodeDeploy 权限参考](#) (p. 269)。

在策略中指定条件

当您授予权限时, 可使用访问策略语言来指定规定策略何时生效的条件。例如, 您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息, 请参阅 IAM 用户指南 中的 [条件](#)。

要表示条件, 您可以使用预定义的条件键。没有特定于 AWS CodeDeploy 的条件键。但有 AWS 范围内的条件密钥, 您可以根据需要使用。有关 AWS 范围内的键的完整列表, 请参阅 IAM 用户指南 中的 [条件的可用键](#)。

为 AWS CodeDeploy 使用基于身份的策略 (IAM 策略)

本主题提供了基于身份的策略的示例, 这些示例展示了账户管理员如何将权限策略附加到 IAM 身份 (即用户、组和角色), 从而授予对 AWS CodeDeploy 资源执行操作的权限。有关必须附加到 IAM 用户才能使用 AWS CodeDeploy 的策略的信息, 请参阅 [步骤 1: 预置 IAM 用户](#) (p. 16)。

Important

我们建议您首先阅读以下介绍性主题, 这些主题讲解了管理 AWS CodeDeploy 资源访问的基本概念和选项。有关更多信息, 请参阅 [管理您的 AWS CodeDeploy 资源的访问权限概述](#) (p. 261)。

主题

- [使用 AWS CodeDeploy 控制台所需的权限](#) (p. 266)
- [适用于 AWS CodeDeploy 的 AWS 托管 \(预定义\) 策略](#) (p. 266)
- [客户托管策略示例](#) (p. 267)

以下是一个权限策略示例, 允许用户删除与 **us-west-2** 区域中名为 **WordPress_App** 的应用程序关联的名为 **WordPress_DepGroup** 的部署组。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource": [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentgroup:WordPress_App/WordPress_DepGroup"
      ]
    }
  ]
}
```

```
} ]  
}
```

使用 AWS CodeDeploy 控制台所需的权限

用户若要能够使用 AWS CodeDeploy 控制台，则必须拥有一组最低的权限来允许其为自己的 AWS 账户描述其他 AWS 资源。要充分使用 AWS CodeDeploy 控制台中的 AWS CodeDeploy，您必须拥有来自以下服务的权限：

- Amazon EC2 Auto Scaling
- AWS CodeDeploy
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- AWS 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 AWS CodeDeploy 控制台，同时向用户附加 `AWSCodeDeployReadOnlyAccess` 托管策略，请参阅[适用于 AWS CodeDeploy 的 AWS 托管 \(预定义\) 策略 \(p. 266\)](#)。

对于只需要调用 AWS CLI 或 AWS CodeDeploy API 的用户，您无需为其提供最低控制台权限。

适用于 AWS CodeDeploy 的 AWS 托管 (预定义) 策略

AWS 通过提供由 AWS 创建和管理的独立 IAM 策略来解决许多常用案例。这些 AWS 托管策略可授予常用案例的必要权限，因此，您可以免去调查都需要哪些权限的工作。有关更多信息，请参阅 IAM 用户指南 中的 [AWS 托管策略](#)。

以下 AWS 托管策略（您可以将它们挂载到自己账户中的用户）是特定于 AWS CodeDeploy 的：

- `AWSCodeDeployFullAccess` – 授与对 AWS CodeDeploy 的完全访问权限。

Note

`AWSCodeDeployFullAccess` 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，仅提供对特定于 AWS CodeDeploy 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 AWS CodeDeploy 的只读访问权限。
- `AWSCodeDeployRole` – 允许 AWS CodeDeploy 用户通过 Amazon EC2 实例的 Amazon EC2 标签或 Auto Scaling 组名称来识别这些实例，通过本地实例的标签来识别本地实例，以及将应用程序修订相应地部署到这些实例。提供所需权限向 Amazon SNS 主题发布通知和从 CloudWatch 检索关于警报的信息。
- `AWSCodeDeployRoleForLambda` – 向 AWS CodeDeploy 授予 AWS Lambda 的访问权限。

部署过程某些方面的权限已授予两个代表 AWS CodeDeploy 进行操作的其他角色类型，而不是授予 IAM 用户：

- IAM instance profile: An IAM role that you attach to your Amazon EC2 instances. This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications that will be deployed by AWS CodeDeploy are stored. For more information, see [步骤 4 : 为 Amazon EC2 实例创建 IAM 实例配置文件 \(p. 22\)](#).
- Service role: An IAM role that grants permissions to an AWS service so it can access AWS resources. The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources. For AWS CodeDeploy, a service role is used for the following:
 - To read either the tags applied to the instances or the Amazon EC2 Auto Scaling group names associated with the instances. This enables AWS CodeDeploy to identify instances to which it can deploy applications.
 - To perform operations on instances, Auto Scaling groups, and Elastic Load Balancing load balancers.
 - To publish information to Amazon SNS topics so that notifications can be sent when specified deployment or instance events occur.
 - To retrieve information about CloudWatch alarms in order to set up alarm monitoring for deployments.

For more information, see [步骤 3 : 为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#).

此外，您还可以创建自己的自定义 IAM 策略，以授予 AWS CodeDeploy 操作和资源的相关权限。您可以将这些自定义策略挂载到需要这些权限的 IAM 用户或组。

客户托管策略示例

本节的用户策略示例介绍如何授予执行各个 AWS CodeDeploy 操作的权限。当您使用 AWS CodeDeploy API、AWS 开发工具包或 AWS CLI 时，可以使用这些策略。当您使用控制台时，您需要授予特定于控制台的其他权限，[使用 AWS CodeDeploy 控制台所需的权限 \(p. 266\)](#)中对此进行了讨论。

您可以使用列出的以下示例 IAM 策略来限制 IAM 用户和角色对 AWS CodeDeploy 的访问。

Note

所有示例都使用 美国西部 (俄勒冈) 区域 (us-west-2) 和虚构的账户 ID。

示例

- [示例 1 : 允许用户在单个区域中执行 AWS CodeDeploy 操作 \(p. 267\)](#)
- [示例 2 : 允许用户为单个应用程序注册修订 \(p. 268\)](#)
- [示例 3 : 允许用户为单个部署组创建部署 \(p. 268\)](#)

示例 1 : 允许用户在单个区域中执行 AWS CodeDeploy 操作

以下示例授予仅在 **us-west-2** 区域执行 AWS CodeDeploy 操作的权限：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:*"
      ]
    }
  ]
}
```



```
]
}
```

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-2:80398EXAMPLE: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** 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。所有这些资源都与 **us-west-2** 区域关联。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentgroup:WordPress_App/WordPress_DepGroup"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:application:WordPress_App"
      ]
    }
  ]
}
```

AWS CodeDeploy 权限参考

在设置 [访问控制](#) (p. 261) 和编写您可挂载到 IAM 身份的权限策略 (基于身份的策略) 时, 可以使用下表作为参考。该表列出每个 AWS CodeDeploy API 操作、您可授予执行权限的对应操作, 以及用于授予权限的资源 ARN 的格式。您可以在策略的 Action 字段中指定操作, 在策略的 Resource 字段中指定 ARN (带不带通配符 (*) 皆可) 作为资源值。

您可以在 AWS CodeDeploy 策略中使用 AWS 范围的条件键来表达条件。有关 AWS 范围内的密钥的完整列表, 请参阅 IAM 用户指南 中的 [可用密钥](#)。

要指定操作, 请在 API 操作名称之前使用 codedeploy: 前缀 (例如, codedeploy:GetApplication 和 codedeploy:CreateApplication)。要在单个语句中指定多项操作, 请使用逗号将它们隔开 (例如, "Action": ["codedeploy:action1", "codedeploy:action2"])。

使用通配符

您可以在 ARN 使用通配符 (*) 以指定多个操作或资源。例如, codedeploy:* 指定所有 AWS CodeDeploy 操作, codedeploy:Get* 指定以单词 Get 开头的的所有 AWS CodeDeploy 操作。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-2:80398EXAMPLE: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](#)。

下面列出了 IAM 策略中可指定用于 AWS CodeDeploy 的操作。

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 [AWS CodeDeploy 示例 ARN](#)。

AWS 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版本的信息所必需的。

资源 : arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

BatchGetApplications

操作 : codedeploy:BatchGetApplications

获取有关与 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 Load Balancing 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`

CreateDeployment¹

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`

获取针对与 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 用户或 AWS 账户关联的所有实例的信息所必需的。

资源 : `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

ListDeployments

操作 : `codedeploy:ListDeployments`

获取有关针对与 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 用户或 AWS 账户的所有部署所必需的。

资源 : `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

ListGitHubAccountTokenNames

操作 : `codedeploy:ListGitHubAccountTokenNames`

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一个应用程序修订的信息所必需的。

资源 : `arn:aws:codedeploy:region:account-id:application:application-name`

RegisterOnPremisesInstance

操作 : `codedeploy:RegisterOnPremisesInstance`

向 AWS 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`

在蓝/绿部署期间成功完成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`

UpdateApplication³

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`

UpdateDeploymentGroup³

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 当您指定 `CreateDeployment` 权限时，您还必须指定部署配置的 `GetDeploymentConfig` 权限和应用程序修订的 `GetApplicationRevision` 或 `RegisterApplicationRevision` 权限。

² 在提供特定部署组时对 `ListDeployments` 有效，但在列出所有与 IAM 用户关联的部署时无效。

³ 对于 `UpdateApplication`，您必须同时具有对旧应用程序名称和新应用程序名称的 `UpdateApplication` 权限。对于涉及更改部署组名称的 `UpdateDeploymentGroup` 操作，您必须同时具有对旧的和新的部署组名称的 `UpdateDeploymentGroup` 权限。

AWS CodeDeploy AppSpec File参考

本部分仅供参考。有关 AppSpec file的概念性概述，请参阅[Application Specification Files \(p. 14\)](#)。

application specification file (AppSpec file) 是 [YAML](#) 格式或 JSON 格式文件，由 AWS CodeDeploy 用来管理部署。

主题

- [AWS Lambda 计算平台上的 AppSpec 文件 \(p. 275\)](#)
- [EC2/本地计算平台上的 AppSpec 文件 \(p. 275\)](#)
- [AppSpec File结构 \(p. 276\)](#)
- [AppSpec File示例 \(p. 293\)](#)
- [AppSpec File间距 \(p. 295\)](#)
- [验证您的 AppSpec File 和文件位置 \(p. 296\)](#)

AWS Lambda 计算平台上的 AppSpec 文件

如果您的应用程序使用 AWS Lambda 计算平台，则 AWS CodeDeploy 使用 AppSpec file来确定：

- 要部署的 Lambda 函数版本。
- 要用作验证测试的 Lambda 函数。

AppSpec file可以采用 YAML 格式或 JSON 格式。在创建部署时，您还可以将 AppSpec file内容直接输入到 AWS 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 AWS CodeDeploy 使用 AppSpec file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。

AppSpec file必须是名为 `appspec.yml` 的 YAML 格式的文件，并且必须放在应用程序源代码目录结构的根目录中。否则，部署会失败。

完成 AppSpec file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 AWS CodeDeploy 的应用程序修订 \(p. 208\)](#)。

Note

The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

有了捆绑的存档文件（在 AWS CodeDeploy 中称为修订）之后，您就可以将其上传到 Amazon S3 存储桶或 Git 存储库。然后使用 AWS CodeDeploy 部署修订。有关说明，请参阅[使用 AWS CodeDeploy 创建部署 \(p. 220\)](#)。

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 部署添加 AppSpec 文件 \(p. 210\)](#) 和 [计划 AWS CodeDeploy 的修订 \(p. 208\)](#)。

AppSpec 文件结构

下面是用于部署到 AWS Lambda 和 EC2/本地 计算平台的 AppSpec file 的高级结构。

YAML 格式的 AppSpec file 中的字符串值一定不能包含在引号 (") 中，除非另行指定。

用于 AWS Lambda 部署的 AppSpec 文件结构

Note

此 AppSpec file 以 YAML 编写，不过您也可以使用相同的结构，在 JSON 中为 Lambda 部署编写 AppSpec file。JSON 格式的 AppSpec file 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：

version

此部分指定 AppSpec file 的版本。请勿更改此值。版本是必需的。目前唯一允许的值为 **0.0**。此值由 AWS 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“resources”部分 \(仅限 AWS Lambda 部署\) \(p. 280\)](#)。

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 \(p. 284\)](#)。

用于 EC2/本地 部署的 AppSpec 文件结构

```
version: 0.0
os: operating-system-name
files:
  source-destination-files-mappings
permissions:high
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

在此结构中：

version

此部分指定 AppSpec file 的版本。请勿更改此值。版本是必需的。目前唯一允许的值为 **0.0**。此值由 AWS 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 - 实例为 Amazon Linux、Ubuntu Server 或 RHEL 实例。
- windows - 实例为 Windows Server 实例。

使用字符串指定 os。

文件

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 的“files”部分 \(仅限 EC2/本地 部署\) \(p. 277\)](#)。

许可

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 Server 和 Red Hat Enterprise Linux (RHEL) 实例。

有关更多信息，请参阅 [AppSpec 的“permissions”部分 \(仅限 EC2/本地 部署\) \(p. 281\)](#)。

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 \(p. 284\)](#)。

主题

- [AppSpec 的“files”部分 \(仅限 EC2/本地 部署\) \(p. 277\)](#)
- [AppSpec 的“resources”部分 \(仅限 AWS Lambda 部署\) \(p. 280\)](#)
- [AppSpec 的“permissions”部分 \(仅限 EC2/本地 部署\) \(p. 281\)](#)
- [AppSpec 的“hooks”部分 \(p. 284\)](#)

AppSpec 的“files”部分 (仅限 EC2/本地 部署)

向 AWS CodeDeploy 提供以下相关信息：部署的 Install 事件期间应安装在实例上的应用程序修订中的文件。仅当您要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
    destination: destination-file-location
```

可以设置多个 source 和 destination 对。

source 指令标识要复制到实例的修订中的文件或目录：

- 如果 source 是指一个文件，则仅指定的文件将复制到实例。
- 如果 source 是指一个目录，则该目录中的所有文件都将复制到实例。
- 如果 source 是一个单斜杠 (/) (对于 Amazon Linux、RHEL 和 Ubuntu Server 实例，为“/”；对于 Windows Server 实例，为“\”)，则修订中的所有文件都将复制到实例。

在 source 中使用的路径是相对路径（从修订的根目录开始）。

destination 指令标识应将文件复制到的实例上位置。此路径必须是完全限定的路径。

source 和 destination 都是使用字符串指定的。

下面是 Amazon Linux、Ubuntu Server 或 RHEL 实例的示例 files 部分。

```
files:
- source: Config/config.txt
  destination: /webapps/Config
- source: source
  destination: /webapps/myApp
```

在此示例中，将在 Install 事件期间执行下面两种操作：

1. 将修订中的 Config/config.txt 文件复制到实例上的 /webapps/Config/config.txt 路径中。
2. 以递归方式将修订的 source 目录中的所有文件复制到实例上的 /webapps/myApp 目录中。

“files”部分示例

以下示例显示如何指定 files 部分。尽管这些示例描述的是 Windows Server 文件和目录（文件夹）结构，但是它们可轻松适合于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

Note

只有 EC2/本地 部署使用 files 部分。它不适用于 AWS Lambda 部署。

对于以下示例，我们假设这些文件出现在 source 的根的捆绑包中：

- appspec.yml
- my-file.txt
- my-file-2.txt
- my-file-3.txt

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
- source: ./my-file.txt
  destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#
# -----
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
- source: my-file-2.txt
  destination: c:\temp
- source: my-file-3.txt
  destination: c:\temp
#
# Result:
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# -----
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the appspec.yml file)
to the destination folder c:\temp.
#
```

```
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
```

对于以下示例，我们假设 appspec.yml 与包含三个文件的名为 my-folder 的文件夹一起出现在 source 的根的捆绑包中：

- appspec.yml
- my-folder\my-file.txt
- my-folder\my-file-2.txt
- my-folder\my-file-3.txt

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the destination
  folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# -----
#
# 5) Copy my-folder and its 3 files to my-folder within the destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# -----
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file.txt
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# -----
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destination folder
  c:\temp.
#
files:
```

```
- source: .\my-folder\my-file-2.txt
  destination: c:\temp\my-folder
- source: .\my-folder\my-file-3.txt
  destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# -----
#
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the destination
#   folder c:\temp.
#
files:
- source: .\my-folder\my-file-2.txt
  destination: c:\temp\other-folder
- source: .\my-folder\my-file-3.txt
  destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# -----
#
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the destination
#   folder c:\temp.
#
files:
- source: \
  destination: c:\temp
#
# 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“resources”部分 (仅限 AWS Lambda 部署)

“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。

AppSpec 的“permissions”部分 (仅限 EC2/本地 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 Amazon Linux、Ubuntu Server 和 RHEL 实例。

Note

“permissions”部分仅用于 EC2/本地 部署。它不用于 AWS Lambda 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 的所有者名称。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 的组名称。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。一个整数，指定要应用于 object 的权限的八进制模式。例如，**644** 表示所有者具有读取和写入权限，组具有只读权限，所有其他用户具有只读权限。**4755** 表示 **setuid** 属性已设置，所有者具有完全控制权限，组具有读取和执行权限，所有其他用户具有读取和执行权限。(有关更多示例，请参阅 Linux **chmod** 命令文档。) 如果未指定 **mode**，则在执行复制操作之后，应用于原始文件或目录/文件夹结构的所有现有模式将保持不变。

使用字符串指定 **mode**。

- **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 (-) 后跟一个空格和一个字符串的形式指定 **acls** (例如，**- u:jane:rw**)。如果您有多个 ACL，每个 ACL 通过单独的行指定。

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 file 失败。可改用 **mode** 来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。标签指定为包含 **user**、**type** 和 **range** 的关键词。(有关更多信息，请参阅 SELinux 文档。) 每个关键词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** 通过单独的行指定。

- **type** - 可选。将指定权限应用到的对象类型。**type** 是一个字符串，可设置为 **file** 或 **directory**。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 (-) 后跟一个空格和一个字符串的形式指定 **type** (例如，**- file**)。

“permissions”部分示例

以下示例说明如何使用 **object**、**pattern**、**except**、**owner**、**mode** 和 **type** 指令指定“permissions”部分。此示例仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。在此示例中，假设以下文件和文件夹按此层次结构复制到实例：

```
/tmp
|-- my-app
|   |-- my-file-1.txt
|   |-- my-file-2.txt
|   |-- my-file-3.txt
|   |-- my-folder-1
|       |-- my-file-4.txt
|       |-- my-file-5.txt
|       |-- my-file-6.txt
|   |-- my-folder-2
|       |-- my-file-7.txt
```

```
|-- my-file-8.txt
|-- my-file-9.txt
`-- my-folder-3
```

以下 AppSpec file 显示如何对这些已复制的文件和文件夹设置权限：

```
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
```

以下示例说明如何指定增加了 acls 和 context 指令的“permissions”部分。此示例仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。

```
permissions:
- object: /var/www/html/WordPress
  pattern: "*"
  except: [/var/www/html/WordPress/ReadMe.txt]
  owner: bob
  group: writers
  mode: 644
  acls:
    - u:mary:rw
    - u:sam:rw
    - m::rw
  context:
    user: unconfined_u
    type: httpd_sys_content_t
    range: s0
  type:
    - file
```

AppSpec 的“hooks”部分

AppSpec file 的“hooks”部分内容会随着部署的计算平台而改变。EC2/本地 部署的“hooks”部分包含将部署生命周期事件挂钩链接到一个或多个脚本的映射。Lambda 部署的“hooks”部分指定在部署生命周期事件期间运

行的 Lambda 验证函数。如果某个事件的挂钩不存在，则不会对该事件执行任何操作。仅当您将在部署过程中运行脚本或 Lambda 验证函数时，才需要此部分。

主题

- [用于 AWS Lambda 部署的 AppSpec 的“hooks”部分 \(p. 285\)](#)
- [用于 EC2/本地 部署的 AppSpec 的“hooks”部分 \(p. 287\)](#)

用于 AWS Lambda 部署的 AppSpec 的“hooks”部分

主题

- [用于 AWS Lambda 部署的生命周期事件挂钩的列表 \(p. 285\)](#)
- [挂钩在 Lambda 函数版本部署中的运行顺序 \(p. 285\)](#)
- [“挂钩”部分的结构 \(p. 285\)](#)
- [示例 Lambda“hooks”函数 \(p. 286\)](#)

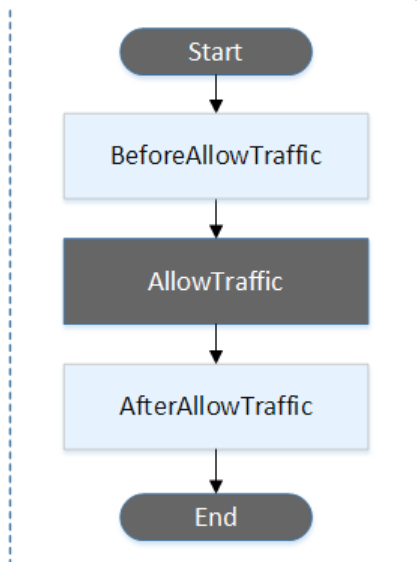
用于 AWS Lambda 部署的生命周期事件挂钩的列表

AWS Lambda 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。下面是适用于 AppSpec 文件的挂钩的描述。

- 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。

挂钩在 Lambda 函数版本部署中的运行顺序

在无服务器 Lambda 函数版本部署中，事件挂钩按以下顺序运行：



Note

部署中的 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 AWS CodeDeploy 可调用以验证 Lambda 部署的 Lambda 函数。您可以为 BeforeAllowTraffic 和 AfterAllowTraffic 部署生命周期事件使用相同或不同的函数。在完成验证测试后，Lambda 验证函数将回调 AWS CodeDeploy，并传输“成功”或“失败”结果。

Important

如果 AWS CodeDeploy 在一小时内未收到 Lambda 验证函数的通知，则假定部署失败。

在调用 Lambda 挂钩函数之前，必须向服务器通知部署 ID 和生命周期事件挂钩执行 ID：

```
aws deploy put-lifecycle-event-hook-execution-status --deployment-id <deployment-id> --
status Succeeded --lifecycle-event-hook-execution-id <execution-id> --region <region>
```

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 AWS CodeDeploy.
  var params = {
    deploymentId: deploymentId,
    lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
    status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
  };

  // Pass AWS 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 EC2/本地 部署的 AppSpec 的“hooks”部分

主题

- [生命周期事件挂钩的列表 \(p. 287\)](#)
- [生命周期事件挂钩可用性 \(p. 288\)](#)
- [挂钩在部署中的运行顺序 \(p. 289\)](#)
- [“挂钩”部分的结构 \(p. 291\)](#)
- [挂钩的环境变量可用性 \(p. 292\)](#)
- [挂钩示例 \(p. 293\)](#)

生命周期事件挂钩的列表

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。每个生命周期事件的挂钩是在单独的行中使用字符串指定的。下面是适用于 AppSpec 文件的挂钩的描述。

有关哪些生命周期事件挂钩对哪些部署和回滚类型有效的信息，请参阅[生命周期事件挂钩可用性 \(p. 288\)](#)。

- **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删除部署准备过程中当前已安装的软件包。用于此部署生命周期事件的 AppSpec file 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 file。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 钩子。

为确定上次成功部署的应用程序修订的位置，AWS CodeDeploy 代理会查看 `deployment-group-id_last_successful_install` 文件中列出的位置。此文件位于：

`/opt/codedeploy-agent/deployment-root/deployment-instructions` 文件夹中 (在 Amazon Linux、Ubuntu Server 和 RHEL Amazon EC2 实例) 上。

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` 文件夹中 (在 Windows Server Amazon EC2 实例) 上。

要对在 ApplicationStop 部署生命周期事件期间失败的部署进行故障排除，请参阅[对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 \(p. 315\)](#)。

- **DownloadBundle** - 在此部署生命周期事件期间，AWS 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` 文件夹中 (在 Amazon Linux、Ubuntu Server 和 RHEL Amazon EC2 实例) 上。

`C:\ProgramData\Amazon\CodeDeploy\deployment-group-id\deployment-id\deployment-archive` 文件夹中 (在 Windows Server Amazon EC2 实例) 上。

此事件是为 AWS 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” \(p. 316\)](#)。

- **BeforeInstall** - 您可以使用此部署生命周期事件执行预安装任务，例如解密文件和创建当前版本的备份。

- **Install** - 在此部署生命周期事件期间，AWS 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 AWS 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 ApplicationStop 期间停止的服务。
- **ValidateService** - 这是最后的部署生命周期事件。它用于验证部署已成功完成。
- **BeforeBlockTraffic** - 在从负载均衡器取消注册实例之前，您可以使用此部署生命周期事件在这些实例上运行任务。

要对在 BeforeBlockTraffic 部署生命周期事件期间失败的部署进行故障排除，请参阅[对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 \(p. 315\)](#)。

- **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 AWS 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[对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 \(p. 315\)](#)。

- **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 AWS 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性

下表列出了适用于每个部署和回滚方案的生命周期事件挂钩。

生命周期事件名称	就地部署 ¹	蓝/绿部署：原始实例	蓝/绿部署：替换实例	蓝/绿部署回滚：原始实例	蓝/绿部署回滚：替换实例
ApplicationStop	✓		✓		
DownloadBundle ²	✓		✓		
BeforeInstall	✓		✓		
安装 ²	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic ²	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic ²	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	

生命周期事件名称	就地部署 ¹	蓝/绿部署：原始实例	蓝/绿部署：替换实例	蓝/绿部署回滚：原始实例	蓝/绿部署回滚：替换实例
¹ 也适用于就地部署的回滚。					
² 为 AWS CodeDeploy 操作预留。不能用于运行脚本。					

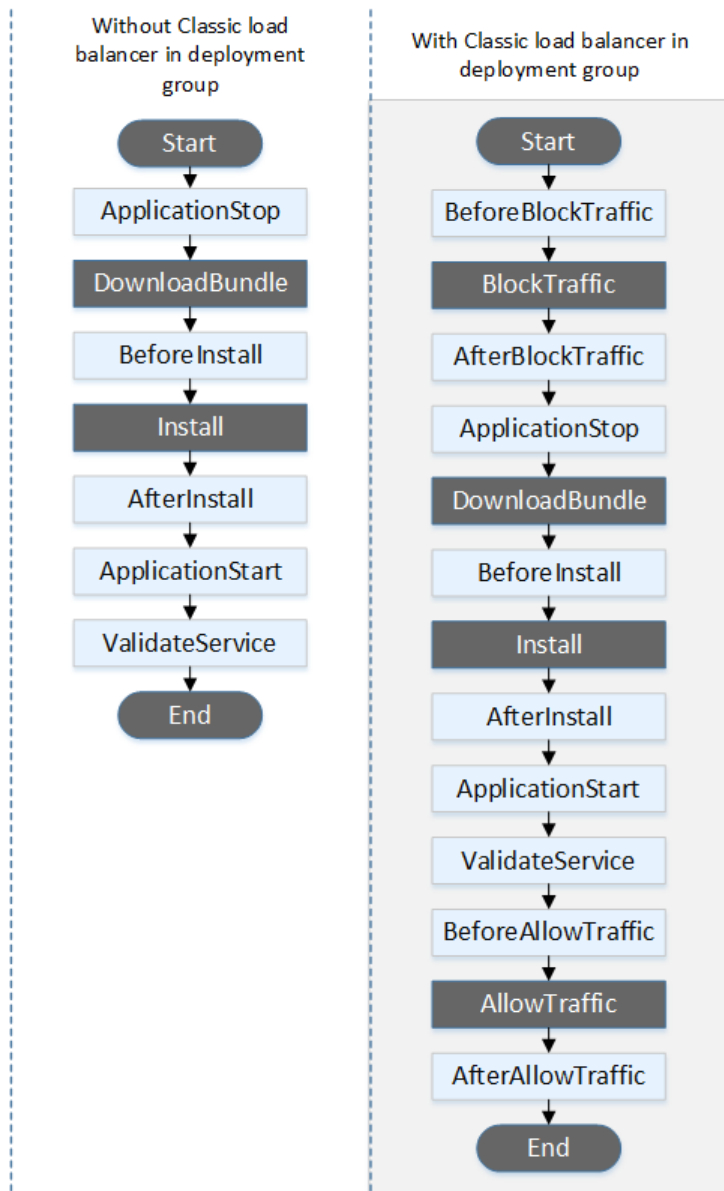
挂钩在部署中的运行顺序

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，仅当您在部署组中通过 Elastic Load Balancing 指定 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer 时，与阻止及允许流量相关的六个挂钩才适用。

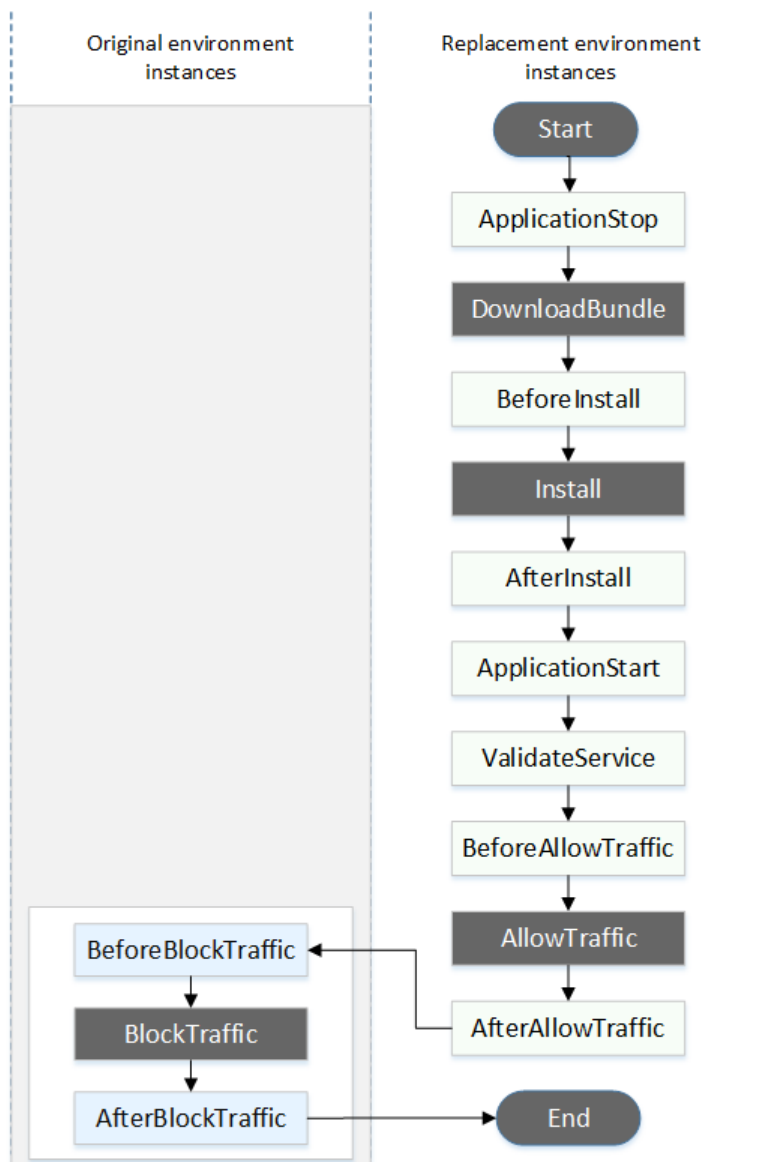


Note

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。不过，您可以编辑 AppSpec file 的“files”部分，以指定在 Install 事件期间安装的内容。

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 file 的“files”部分，以指定在 Install 事件期间安装的内容。

“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

必需。修订的脚本文件包的位置。

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 AWS CodeDeploy 代理。AWS CodeDeploy 不存储密码，因此，如果 runas 用户需要密码，则无法模拟该默认用户。此元素仅适用于 Amazon Linux 和 Ubuntu Server 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

AWS CodeDeploy 中属于当前部署的应用程序的名称 (例如 WordPress_App)。

DEPLOYMENT_ID

AWS CodeDeploy 已分配给当前部署的 ID (例如 d-AB1CDEF23)。

DEPLOYMENT_GROUP_NAME

AWS CodeDeploy 中属于当前部署的部署组的名称 (例如 WordPress_DepGroup)。

DEPLOYMENT_GROUP_ID

AWS 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 服务器上的监听端口更改为 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 AWS 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 脚本将在部署过程的 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 [计划 AWS CodeDeploy 的修订 \(p. 208\)](#)。

AppSpec File 示例

此主题提供用于 AWS Lambda 和 EC2/本地 部署的 AppSpec 文件示例。

主题

- [用于 AWS Lambda 部署的 AppSpec File 示例 \(p. 293\)](#)
- [用于 EC2/本地 部署的 AppSpec File 示例 \(p. 294\)](#)

用于 AWS Lambda 部署的 AppSpec File 示例

下面是使用 YAML 编写的、用于部署 Lambda 函数版本的 AppSpec file 示例。

```
version: 0.0
Resources:
  - myLambdaFunction:
```

```
Type: AWS::Lambda::Function
Properties:
  Name: "myLambdaFunction"
  Alias: "myLambdaFunctionAlias"
  CurrentVersion: "1"
  TargetVersion: "2"
Hooks:
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

下面是用 JSON 编写的上述示例的版本。

```
{
  "version": 0.0,
  "Resources": [{
    "myLambdaFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Name": "myLambdaFunction",
        "Alias": "myLambdaFunctionAlias",
        "CurrentVersion": "1",
        "TargetVersion": "2"
      }
    }
  ]},
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
  },
  {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
  }
]
}
```

下面是部署期间的事件序列：

1. 在将流量从名为 myLambdaFunction 的 Lambda 函数的版本 1 转移到版本 2 前，运行名为 LambdaFunctionToValidateBeforeTrafficShift 的 Lambda 函数，该函数将验证部署是否已准备好开始流量转移。
2. 如果 LambdaFunctionToValidateBeforeTrafficShift 返回了退出代码 0 (成功)，则开始将流量转移到 myLambdaFunction 的版本 2。此部署的部署配置确定流量转移的速率。
3. 在完成将流量从名为 myLambdaFunction 的 Lambda 函数的版本 1 转移到版本 2 后，运行名为 LambdaFunctionToValidateAfterTrafficShift 的 Lambda 函数，该函数将验证部署是否已成功完成。

用于 EC2/本地 部署的 AppSpec File 示例

下面是对 Amazon Linux、Ubuntu Server 或 RHEL 实例进行就地部署的 AppSpec file 的示例。

Note

到 Windows Server 实例的部署不支持 runas 元素。如果要部署到 Windows Server 实例，请勿将其包含在您的 AppSpec file 中。

```
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
```

```
- source: source
  destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

对于 Windows Server 实例，将 `os: linux` 更改为 `os: windows`。另外，您还必须完全限定 `destination` 路径（例如 `c:\temp\webapps\Config` 和 `c:\temp\webapps\myApp`）。不包括 `runas` 元素。

下面是部署期间的事件序列：

1. 运行位于 `Scripts/UnzipResourceBundle.sh` 的脚本。
2. 如果前面的脚本返回了退出代码 0（成功），则运行位于 `Scripts/UnzipDataBundle.sh` 中的脚本。
3. 将文件从 `Config/config.txt` 路径复制到 `/webapps/Config/config.txt` 路径中。
4. 以递归方式将 `source` 目录中的所有文件复制到 `/webapps/myApp` 目录中。
5. 运行位于 `Scripts/RunResourceTests.sh` 中的脚本，超时时间为 180 秒（3 分钟）。
6. 运行位于 `Scripts/RunFunctionalTests.sh` 中的脚本，超时时间为 3600 秒（1 小时）。
7. 以 `Scripts/MonitorService.sh` 用户身份运行位于 `codedeploy` 中的脚本，超时时间为 3600 秒（1 小时）。

AppSpec File间距

下面是正确的 AppSpec file 间距格式。方括号中的数字表示各项之间必须存在的空格数。例如，`[4]` 表示在各项之间插入四个空格。如果 AppSpec file 中的位置和空格数不正确，则 AWS CodeDeploy 将引发可能难以调试的错误。

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
[4]type:
[6]-[1]object-type
```



```
hooks:  
[2]deployment-lifecycle-event-name:  
[4]-[1]location:[1]script-location  
[6]timeout:[1]timeout-in-seconds  
[6]runas:[1]user-name
```

下面是间距正确的 AppSpec file 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 File 和文件位置

文件语法

AWS CodeDeploy 不提供验证 AppSpec File 内容的工具。请考虑使用基于浏览器的工具帮助您检查 YAML 语法，如 [YAML Lint](#) 或 [Online YAML Parser](#)。

文件位置

要验证您是否已将 AppSpec file 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, or Unix 实例上：

```
ls path/to/root/directory/appspec.yml
```

如果 AppSpec file 不在该根目录中，则将显示“No such file or directory”错误。

在本地 Windows 实例上：

```
dir path\to\root\directory\appspec.yml
```

如果 AppSpec file 不在该根目录中，则将显示“File Not Found”错误。

AWS CodeDeploy 代理配置参考

安装 AWS CodeDeploy 代理后，将在实例上放置一个配置文件。此配置文件指定 AWS 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：

<code>:log_aws_wire:</code>	<p>设置为 <code>true</code> 以便 AWS CodeDeploy 代理从 Amazon S3 捕获线路日志并将它们写入 <code>codedeploy-agent.wire.log:log_dir</code> 设置指向的位置中名为 <code>aws</code> 的文件。</p> <p>Warning</p> <p>您仅应在捕获线路日志需要的时间内将 <code>:log_aws_wire:</code> 设置为 <code>true</code>。<code>codedeploy-agent.wire.log</code> 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，包括在此设置为 <code>true</code> 时传入或传出 Amazon S3 的文件的纯文本内容。此设置为 <code>true</code> 时，线路日志包含有关与 AWS 账户关联的所有 Amazon S3 活动的信息，而不仅仅包含与 AWS CodeDeploy 部署相关的活动的信息。</p> <p>默认设置为 <code>false</code>。</p> <p>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。</p>
<code>:log_dir:</code>	<p>实例上用于存储与 AWS CodeDeploy 代理操作相关的日志文件的文件夹。</p> <p>Amazon Linux、Ubuntu Server 和 RHEL 实例的默认设置为 <code>/var/log/aws/codedeploy-agent</code>，Windows Server 实例的默认设置为 <code>C:\ProgramData\Amazon\CodeDeploy\log</code>。</p>
<code>:pid_dir:</code>	<p>存储 <code>codedeploy-agent.pid</code> 的文件夹。</p> <p>此文件包含 AWS CodeDeploy 代理的进程 ID (PID)。默认设置为 <code>/opt/codedeploy-agent/state/.pid</code>。</p> <p>此设置仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。</p>
<code>:program_name:</code>	<p>AWS CodeDeploy 代理程序名称。</p> <p>默认设置为 <code>codedeploy-agent</code>。</p>

	此设置仅适用于 Amazon Linux、Ubuntu Server 和 RHEL 实例。
:root_dir:	<p>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。</p> <p>Amazon Linux、Ubuntu Server 和 RHEL 实例的默认设置为 '/opt/codedeploy-agent/deployment-root'，Windows Server 实例的默认设置为 C:\ProgramData\Amazon\CodeDeploy。</p>
:verbose:	<p>设置为 true 以便 AWS CodeDeploy 代理在实例上打印调试消息日志文件。</p> <p>Amazon Linux、Ubuntu Server 和 RHEL 实例的默认设置为 false，Windows Server 实例的默认设置为 true。</p>
:wait_between_runs:	<p>AWS CodeDeploy 代理为挂起的部署轮询 AWS CodeDeploy 的时间间隔（以秒为单位）。</p> <p>默认设置为 1。</p>
:on_premises_config_file:	<p>对于本地实例，这是名为 codedeploy.onpremise.yml（适用于 Ubuntu Server 和 RHEL）或 conf.onpremise.yml（适用于 Windows Server）的配置文件的备用位置的路径。</p> <p>默认情况下，这些文件存储在 /etc/codedeploy-agent/conf/codedeploy.onpremise.yml（适用于 Ubuntu Server 和 RHEL）以及 C:\ProgramData\Amazon\CodeDeploy\conf.onpremise.yml（适用于 Windows Server）中。</p> <p>在 AWS CodeDeploy 代理版本 1.0.1.686 及更高版本中可用。</p>
:proxy_uri:	<p>（可选）您希望 AWS CodeDeploy 代理通过其连接到 AWS 以执行您的 AWS CodeDeploy 操作的 HTTP 代理。使用类似于 https://user:password@my.proxy:443/path?query 的格式。</p> <p>在 AWS CodeDeploy 代理版本 1.0.1.824 及更高版本中可用。</p>
:max_revisions:	<p>（可选）您希望 AWS 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。</p> <p>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，AWS CodeDeploy 还将保留五个最新修订。</p> <p>在 AWS CodeDeploy 代理版本 1.0.1.966 及更高版本中受支持。</p>

相关主题

[使用 AWS CodeDeploy 代理 \(p. 113\)](#)

[管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)

适用于 AWS CodeDeploy 的 AWS CloudFormation 模板参考

除了 AWS CodeDeploy 中可供您使用的其他方法外，您还可以使用 AWS CloudFormation 模板执行下列任务：

- 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。

AWS CloudFormation 是一项服务，可帮助您使用模板来建模和设置 AWS 资源。AWS 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 AWS 资源，而 AWS CloudFormation 则可为您预配和配置这些资源。

有关更多信息，请参阅[什么是 AWS CloudFormation ?](#)和[使用 AWS CloudFormation 模板](#)（在 AWS CloudFormation 用户指南中）。

如果您计划在组织中使用与 AWS CodeDeploy 兼容的 AWS CloudFormation 模板，则您作为管理员，必须授予对 AWS CloudFormation 以及 AWS CloudFormation 依赖的 AWS 服务和操作的访问权。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 AWS CloudFormation 的 IAM 用户：

```
{
  "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 AWS CodeDeploy 创建 Amazon EC2 实例 \(AWS CloudFormation 模板\) \(p. 147\)](#)。
- 有关将策略附加到 IAM 用户的信息，请参阅 IAM 用户指南中的[使用托管策略](#)。
- 要了解如何将用户限制为使用一组有限的 AWS CodeDeploy 操作和资源，请参阅[适用于 AWS CodeDeploy 的 AWS 托管 \(预定义\) 策略 \(p. 266\)](#)。

下表显示 AWS CloudFormation 模板可代表您执行的操作，并且包括的链接指向您可添加到 AWS CloudFormation 模板的 AWS 资源类型及其属性类型的相关详细信息。

操作	AWS CloudFormation 资源类型
创建 AWS CodeDeploy 应用程序。	AWS::CodeDeploy::Application

操作	AWS CloudFormation 资源类型
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
创建 AWS CodeDeploy 将在部署期间使用的一组部署规则、部署成功条件和部署失败条件。	AWS::CodeDeploy::DeploymentConfig
创建 Amazon EC2 实例。 ²	AWS::EC2::Instance
<p>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 AWS CloudFormation 用户指南中的 AWS CodeDeploy DeploymentGroup 部署修订 S3Location 和 AWS CodeDeploy DeploymentGroup 部署修订 GitHubLocation。</p> <p>² 我们提供了您可用来在支持 AWS 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 AWS CodeDeploy 创建 Amazon EC2 实例 (AWS CloudFormation 模板) (p. 147)。</p>	

AWS CodeDeploy 资源工具包参考

AWS CodeDeploy 依赖的许多文件都存储在公共使用的 AWS 区域特定的 Amazon S3 存储桶中。这些文件包含 AWS 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 AWS CodeDeploy 资源工具包。

主题

- [各区域的资源工具包存储桶名称 \(p. 302\)](#)
- [资源工具包内容 \(p. 302\)](#)
- [显示资源工具包文件列表 \(p. 303\)](#)
- [下载资源工具包文件 \(p. 304\)](#)

各区域的资源工具包存储桶名称

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 AWS CodeDeploy 资源工具包文件的 Amazon S3 存储桶的名称。

区域名称	<code>bucket-name</code> 替换	区域标识符
美国东部 (俄亥俄州)	aws-codedeploy-us-east-2	us-east-2
美国东部 (弗吉尼亚北部)	aws-codedeploy-us-east-1	us-east-1
美国西部 (加利福尼亚北部)	aws-codedeploy-us-west-1	us-west-1
美国西部 (俄勒冈)	aws-codedeploy-us-west-2	us-west-2
加拿大 (中部)	aws-codedeploy-ca-central-1	ca-central-1
欧洲 (爱尔兰)	aws-codedeploy-eu-west-1	eu-west-1
欧洲 (伦敦)	aws-codedeploy-eu-west-2	eu-west-2
欧洲 (巴黎)	aws-codedeploy-eu-west-3	eu-west-3
欧洲 (法兰克福)	aws-codedeploy-eu-central-1	eu-central-1
亚太区域 (东京)	aws-codedeploy-ap-northeast-1	ap-northeast-1
亚太区域 (首尔)	aws-codedeploy-ap-northeast-2	ap-northeast-2
亚太区域 (新加坡)	aws-codedeploy-ap-southeast-1	ap-southeast-1
亚太区域 (悉尼)	aws-codedeploy-ap-southeast-2	ap-southeast-2
亚太地区 (孟买)	aws-codedeploy-ap-south-1	ap-south-1
南美洲 (圣保罗)	aws-codedeploy-sa-east-1	sa-east-1

资源工具包内容

下表列出了 AWS CodeDeploy 资源工具包中的文件。

文件	说明
VERSION	AWS CodeDeploy 代理运行在实例上时用来更新自身的文件。
codedeploy-agent.noarch.rpm	面向 Amazon Linux 和 Red Hat Enterprise Linux (RHEL) 的 AWS 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	面向 Ubuntu Server 的 AWS 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	面向 Windows Server 的 AWS 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 AWS CodeDeploy 代理的文件。
CodeDeploy_SampleCF_Template.json	一个 AWS CloudFormation 模板，您可用来启动运行 Amazon Linux 或 Windows Server 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 AWS 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。模板将创建三个 Amazon EC2 实例和 IAM 实例配置文件，为实例授予对 Amazon S3、Auto Scaling、AWS CloudFormation 和 Elastic Load Balancing 中的资源的访问权限。它还会创建负载均衡器和一个 AWS CodeDeploy 服务角色。
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 Server 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)。
SampleApp2_Linux.zip	一个示例应用程序修订，当您运行 Sample deployment wizard 时部署到替换实例队列。
SampleApp_Windows.zip	您可以部署到 Windows Server 实例的示例应用程序修订。可能会有多个文件具有相同的基本文件名，但这些文件具有不同的版本 (例如 -1.0)。

显示资源工具包文件列表

要查看文件列表，请对您的区域使用 `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-us-east-2`
- `aws s3 ls --recursive s3://aws-codedeploy-us-east-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ca-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-2`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-3`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-south-1`
- `aws s3 ls --recursive s3://aws-codedeploy-sa-east-1`

下载资源工具包文件

要下载文件，请对您的区域使用 `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-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2`
- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1`

- ```
aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region us-west-1
```
- ```
aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2
```
- ```
aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3
```
- ```
aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2
```
- ```
aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2
```
- ```
aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1
```
- ```
aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-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-us-east-2 . --region us-east-2
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-us-west-1 . --region us-west-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ca-central-1 . --region ca-central-1
```

- ```
aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-eu-west-2 . --region eu-west-2
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-eu-west-3 . --region eu-west-3
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-eu-central-1 . --region eu-central-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-1 . --region ap-northeast-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-2 . --region ap-northeast-2
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-1 . --region ap-southeast-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-ap-south-1 . --region ap-south-1
```
- ```
aws s3 cp --recursive s3://aws-codedeploy-sa-east-1 . --region sa-east-1
```

AWS CodeDeploy 限制

下表描述了 AWS CodeDeploy 中的限制。

Note

您可以为 Amazon Web Services 一般参考 中的 [AWS 服务限制](#) 中列出的 AWS CodeDeploy 限制请求提高限制。不能增加对部署可运行的小时数的限制。

主题

- [应用程序](#) (p. 307)
- [应用程序修订](#) (p. 307)
- [部署](#) (p. 307)
- [部署配置](#) (p. 308)
- [部署组](#) (p. 309)
- [实例](#) (p. 309)

应用程序

单个区域中与一个 AWS 账户关联的最大应用程序数量	100
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量	100
单个 AWS 账户的 GitHub 连接令牌的最大数目	25

应用程序修订

应用程序修订名称中的最大字符数	100
允许的应用程序修订文件类型	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 AWS 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 appspec.yml 的 application specification file (AppSpec file)。

部署

针对一个部署组的并发部署的最大数目 ¹	1
--------------------------------	---

与一个 AWS 账户关联的最大并发部署数量 ²	100
EC2/本地 就地部署可以运行的最大小时数	8
在 EC2/本地 蓝/绿部署期间在部署修订和将通信转移到替换环境之间的最大小时数	48
在 EC2/本地 蓝/绿部署期间在完成部署和中止原始环境之间的最大小时数	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
AWS Lambda 部署可以运行的最大小时数 ³	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： <ul style="list-style-type: none"> 使用控制台或 AWS CLI create-deployment 命令触发部署。 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
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 在 Auto Scaling 组中的扩展 Amazon EC2 实例上执行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，如果一个 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。如果这五个扩展的 Amazon EC2 实例又与两个应用程序相关联，则会生成十个额外的并发部署。	

部署配置

与一个 AWS 账户关联的自定义部署配置的最大数目	25
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 AWS Lambda canary 或线性部署期间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	2 880
在 AWS Lambda 部署期间在一次递增中可以转移的最大流量百分比	99

部署组

与单个应用程序关联的部署组的最大数量	100
部署组中的最大标签数	10
部署组中的最大 Auto Scaling 组数量	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

实例

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
所需的适用于 Ruby 的 AWS 软件开发工具包版本 (aws-sdk-core)	2.1.2 或更低版本 (适用于早于 1.0.1.880 的 AWS CodeDeploy 代理版本)。 2.2 或更低版本 (适用于 AWS CodeDeploy 代理版本 1.0.1.880 及更高版本)。

AWS CodeDeploy 问题排查

使用本部分中的主题可帮助解决您在使用 AWS CodeDeploy 时可能遇到的问题和错误。

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。有关信息，请参阅 [在 CloudWatch Logs 控制台中查看 AWS CodeDeploy 日志](#)。

主题

- [一般问题排查](#) (p. 310)
- [排查 EC2/本地 部署问题](#) (p. 313)
- [排查 AWS Lambda 部署问题](#) (p. 318)
- [解决部署组问题](#) (p. 318)
- [解决实例问题](#) (p. 319)
- [GitHub 令牌问题排查](#) (p. 320)
- [解决 Auto Scaling 问题](#) (p. 321)
- [AWS CodeDeploy 的错误代码](#) (p. 324)

一般问题排查

主题

- [一般问题排查核对清单](#) (p. 310)
- [AWS CodeDeploy 部署资源仅在特定区域中受支持](#) (p. 311)
- [所需的 IAM 角色不可用](#) (p. 312)
- [避免对同一 Amazon EC2 实例进行并行部署](#) (p. 312)
- [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. 312)
- [使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败](#) (p. 313)

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 [使用 AWS CodeDeploy 查看部署详细信息](#) (p. 229) 和 [View Instance Details](#) (p. 179) 以确定部署失败的原因。如果您无法确定原因，请继续本核查清单中的其余项目。
2. 检查您是否已正确配置实例：
 - 是否已使用指定的 Amazon EC2 密钥对启动实例？有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 [Amazon EC2 密钥对](#)。
 - 是否已将正确的 IAM 实例配置文件附加到实例？有关更多信息，请参阅 [配置用于 AWS CodeDeploy 的 Amazon EC2 实例](#) (p. 153) 和 [步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件](#) (p. 22)。

- 是否已标记实例？有关更多信息，请参阅Amazon EC2 用户指南（适用于 Linux 实例）中的[在控制台中使用标签](#)。
 - AWS CodeDeploy 代理是否已安装并在实例上运行？有关更多信息，请参阅 [管理 AWS CodeDeploy 代理操作 \(p. 118\)](#)。
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 AWS CodeDeploy 查看应用程序详细信息 \(p. 195\)](#)。
 - 要检查部署组设置，请参阅[使用 AWS CodeDeploy 查看部署组详细信息 \(p. 203\)](#)。
4. 确认已正确配置应用程序修订：
- 检查 AppSpec file 的格式。有关更多信息，请参阅 [将应用程序规范文件添加到 AWS CodeDeploy 的修订 \(p. 209\)](#) 和 [AWS CodeDeploy AppSpec File 参考 \(p. 275\)](#)。
 - 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 AWS 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 [使用 AWS CodeDeploy 查看应用程序修订详细信息 \(p. 216\)](#)。
 - 如果您正在从 Amazon S3 部署，请检查您的 Amazon S3 存储桶以验证 AWS CodeDeploy 是否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 \(p. 221\)](#)。
 - 如果您正在从 GitHub 部署，请检查您的 GitHub 存储库以验证 AWS CodeDeploy 是否获得了下载应用程序修订的权限。有关更多信息，请参阅 [使用 AWS CodeDeploy 创建部署 \(p. 220\)](#) 和 [GitHub 对 AWS CodeDeploy 中的应用程序进行的身份验证 \(p. 46\)](#)。
5. 检查是否已正确配置服务角色。有关信息，请参阅 [步骤 3：为 AWS CodeDeploy 创建服务角色 \(p. 18\)](#)。
6. 确认您已执行[AWS CodeDeploy 入门 \(p. 16\)](#)中的步骤，以便：
- 将策略附加至 IAM 用户。
 - 安装或升级并配置 AWS CLI。
 - 创建 IAM 实例配置文件和服务角色。
- 有关更多信息，请参阅 [AWS CodeDeploy 的身份验证和访问控制 \(p. 260\)](#)。
7. 确认您正在使用 AWS CLI 版本 1.6.1 或更高版本。要检查您已安装的版本，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。

AWS CodeDeploy 部署资源仅在特定区域中受支持

如果您在 AWS CLI 或 AWS CodeDeploy 控制台中看不到或无法访问应用程序、部署组、实例或其他部署资源，请确保您引用了 AWS General Reference 中的[区域和终端节点](#)中列出的某个区域。

必须在这些区域中的某个区域中启动和创建将在 AWS CodeDeploy 部署中使用的 Amazon EC2 实例和 Auto Scaling 组。

如果您使用的是 AWS CLI，请从 AWS CLI 运行 `aws configure` 命令。然后，您可以查看和设置默认区域。

如果您使用的是 AWS CodeDeploy 控制台，请从导航栏上的区域选择器中，选择一个受支持的区域。

Important

要在 中国（北京）区域 或 中国（宁夏）区域 中使用服务，您必须拥有这些区域的账户和凭据。其他 AWS 区域的账户和凭证不适用于 北京和宁夏区域，反之亦然。

有关 中国区域 的一些资源的信息（例如 AWS CodeDeploy 资源工具包存储桶名称和 AWS CodeDeploy 代理安装过程）不包含在此版本的 AWS CodeDeploy 用户指南中。

有关更多信息：

- [在中国（北京）区域开始使用 AWS 中的 AWS CodeDeploy](#)
- [中国区域的 AWS CodeDeploy 用户指南 \(英语版本 | 中文版本\)](#)

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某个 AWS CloudFormation 堆栈的一部分创建的，则当您删除该堆栈时，也将删除所有 IAM 角色。这可能是 IAM 角色不再在 IAM 控制台中显示以及 AWS CodeDeploy 不再按预期运行的原因。要纠正此问题，您必须手动重新创建已删除的 IAM 角色。

避免对同一 Amazon EC2 实例进行并行部署

作为最佳实践，您应避免导致尝试同时对一个 Amazon EC2 实例进行多次部署的情况。如果来自不同部署的命令相互竞争在一个实例上运行，则部署可能会超时并失败，原因如下：

- 如果第一个生命周期事件未在部署触发后五分钟内启动，则 AWS CodeDeploy 会使部署失败。您可以使用控制台或 AWS CLI [create-deployment](#) 命令触发部署。
- 如果某个生命周期事件未在前一个生命周期事件结束后五分钟内启动，则 AWS CodeDeploy 会使部署失败。
- AWS CodeDeploy 代理一次只能处理一个部署命令。
- 如果尝试同时运行多个部署，则无法控制部署发生的顺序。

Note

生命周期事件中脚本的默认超时时间为 30 分钟。您可以在 AppSpec file 中将超时时间更改为其他值。有关更多信息，请参阅 [为 EC2/本地 部署添加 AppSpec 文件 \(p. 210\)](#)。

如果一个部署的步骤未在 5 分钟内完成，即使部署过程按预期运行，AWS CodeDeploy 逻辑也会将该部署视为失败的部署。如果来自多个部署的命令同时发送到 AWS CodeDeploy 代理，则可能超出 5 分钟的限制。

有关您在 Auto Scaling 组中进行并行部署时可能遇到的其他难题的信息，请参阅[避免将多个部署组与一个 Auto Scaling 组关联 \(p. 322\)](#)。

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 AWS 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 file 验证失败问题和脚本执行失败的问题。

在 AWS CodeDeploy 控制台中，在部署的事件详细信息页上，选择 View logs。（或者，使用 AWS CLI 调用 [get-deployment-instance](#) 命令。）查找像“invalid character”、“command not found”或“file not found”这样的错误。

为了解决此问题，我们建议：

- 请不要使用自动将非打印字符（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自动或随机引入的字符。有关这些类型的文本编辑器的示例，请在 Internet 上搜索“文本编辑器显示回车”。
- 使用在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的文本编辑器来创建在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的 Shell 脚本文件。有关这些类型的文本编辑器的示例，请在 Internet 上搜索“Linux Shell 脚本编辑器”。
- 如果您必须使用 Windows 或 Mac OS 中的文本编辑器来创建要在 Amazon Linux、Ubuntu Server 或 RHEL 实例上运行的 Shell 脚本文件，请使用可将 Windows 或 Mac OS 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 file 和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 会在 .zip 文件中创建一个中间 `__MACOSX` 文件夹并将组件文件放入该文件夹中。AWS CodeDeploy 找不到组件文件，部署失败。

要解决此问题，建议您使用 AWS CLI 调用 `push` 命令，这会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。

排查 EC2/本地 部署问题

主题

- 部署失败，显示消息“PKCS7 签名的消息验证失败” (p. 313)
- 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. 313)
- 排查 AllowTraffic 生命周期事件失败，但部署日志中未报错的问题 (p. 315)
- 对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 (p. 315)
- 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. 316)
- 默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell (p. 316)
- 长时间运行的进程可能会导致部署失败 (p. 317)

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。有关信息，请参阅 [在 CloudWatch Logs 控制台中查看 AWS CodeDeploy 日志](#)。

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 AWS CodeDeploy 代理版本。对 SHA-2 哈希算法的支持是在 AWS CodeDeploy 代理的 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 AWS CodeDeploy 代理，部署将会失败。有关更多信息，请参阅 [AWS 切换到 SSL 证书的 SHA256 哈希算法、通知：停用低于 1.0.1.85 版本的 AWS CodeDeploy 主机代理和更新 AWS CodeDeploy 代理](#) (p. 127)。

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 AWS 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”。

AWS CodeDeploy User Guide
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”

location-name。”这是因为，在每个部署期间，AWS 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，AWS CodeDeploy 代理会将其视为错误，并且部署将失败。

Note

在 Amazon Linux、RHEL 和 Ubuntu Server 实例上，清除文件位于 `/opt/codedeploy-agent/deployment-root/deployment-instructions/` 中。在 Windows Server 实例上，此位置为 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`。

避免此错误的最简单方式是，指定默认行为之外的选项以使部署失败。对于每个部署，您可以选择是使部署失败、覆盖清除文件中未列出的文件，还是保留实例上已有的文件。

覆盖选项在以下情况下很有用：您在上一个部署后手动将文件放置在实例上，然后将一个同名文件添加到一个应用程序修订。

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 AWS 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. 223) 和 [回滚行为与现有内容](#) (p. 234)。

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 AWS 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。

当您尝试重新部署具有相同的名称和位置的文件时，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，则重新部署获得成功的可能性会更大。AWS CodeDeploy 使用基础部署组 ID 标识要在重新部署前删除的文件。

将新文件部署到实例上的相同位置或将相同的文件重新部署到实例上的相同位置可能会因以下原因而失败：

- 您为将相同修订重新部署到同一实例的操作指定不同的应用程序名称。重新部署将失败，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，AWS CodeDeploy 也将引用不同的基础部署组 ID。
- 您在 AWS 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。重新部署将失败，因为即使应用程序和部署组的名称相同，AWS 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 AWS CodeDeploy 将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，因为在第二次部署开始之前，AWS CodeDeploy 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 AWS 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，在部署开始之前，AWS CodeDeploy 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 file 中，在 `ApplicationStop` 或 `BeforeInstall` 部署生命周期事件中，指定一个自定义脚本以删除与您的修订即将安装的文件匹配的任何位置的文件。
- 将文件部署或重新部署到不属于之前的部署的位置或实例。

- 在删除应用程序或部署组之前，部署一个包含未指定要复制到实例的文件的 AppSpec file 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。（您可以使用 [get-deployment-group](#) 命令检索部署组 ID。）AWS CodeDeploy 将使用该基础部署组 ID 和 AppSpec file 来删除它在上一个成功部署中安装的所有文件。

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，但部署日志中没有指明失败原因。

发生这种失败通常是由于 Elastic Load Balancing 中对于 传统负载均衡器、应用程序负载均衡器 或 Network Load Balancer (用于管理部署组的流量) 的运行状况检查配置不正确。

要解决这一问题，请检查并更正负载均衡器的运行状况检查配置中的错误。

对于 Classic Load Balancer，请参阅 Classic Load Balancer 用户指南中的 [配置运行状况检查](#) 以及 Elastic Load Balancing API Reference version 2012-06-01 中的 [ConfigureHealthCheck](#)。

对于 Application Load Balancer，请参阅 Application Load Balancer 用户指南中的 [目标组的运行状况检查](#)。

对于 Network Load Balancer，请参阅 Network Load Balancer 用户指南中的 [目标组的运行状况检查](#)。

对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除

在部署期间，AWS CodeDeploy 代理运行上一个 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：

- AWS CodeDeploy 代理在正确位置找到了 `deployment-group-id_last_successful_install` 文件，但 `deployment-group-id_last_successful_install` 文件中列出的位置不存在。

在 Amazon Linux、Ubuntu Server 和 RHEL 实例上，此文件必须存在于 `/opt/codedeploy-agent/deployment-root/deployment-instructions` 中。

在 Windows Server 实例上，此文件必须存储在 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` 文件夹中。
- 在 `deployment-group-id_last_successful_install` 文件中列出的位置上，AppSpec file 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 AWS 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 View events。在实例的详细信息页上，在 ApplicationStop、BeforeBlockTraffic 或 AfterBlockTraffic 行中，选择 View logs。或者，使用 AWS CLI 调用 [get-deployment-instance](#) 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，请创建一个新的部署，并指定应忽略 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 失败。您可以通过两种方式执行此操作：

- 使用 AWS CodeDeploy 控制台创建部署。在 Create deployment 页上的 ApplicationStop lifecycle event failure 下，选择 Don't fail the deployment to an instance if this lifecycle event on the instance fails。
- 使用 AWS 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这些生命周期事件的修复脚本，则将来部署会成功，而无需应用此修复。

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，错误为“UnknownError: not opened for reading”：

- 发生内部 Amazon S3 服务错误。请重新部署应用程序修订。
- Amazon 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 AWS CodeDeploy 的修订推送到 Amazon S3 \(p. 214\)](#)和[部署先决条件 \(p. 221\)](#)。
- 您将部署到的实例与一个区域（例如，美国西部（俄勒冈））关联，而包含应用程序修订的 Amazon S3 存储桶与另一个区域（例如，美国东部（弗吉尼亚北部））关联。请确保 Amazon S3 存储桶中的应用程序修订与实例所在的区域关联。

在部署的事件详细信息页的 Download bundle 行中，选择 View logs。或者，使用 AWS CLI 调用 [get-deployment-instance](#) 命令。如果出现此错误，则输出中应有一个错误代码为“UnknownError”且错误消息为“not opened for reading”的错误。

要确定此错误的原因，请执行以下步骤：

1. 对至少一个实例启用线路日志记录，然后重新部署应用程序修订。
2. 查看线路日志记录文件以找到错误。此问题的常见错误消息包括短语“access denied”。
3. 在查看日志文件后，建议您禁用线路日志记录以减小日志文件的大小并减少将来可能会在实例上的输出中以纯文本格式出现的敏感信息量。

要了解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，请参阅 [使用 AWS CodeDeploy 代理 \(p. 113\)](#) 中的 :log_aws_wire:。

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，AWS 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
```

长时间运行的进程可能会导致部署失败

对于针对 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，则 AWS 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，在该进程运行的时间比该事件中的前台和后台进程预期的所需时间长的情况下，AWS 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 file 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
```

当 after-install.sh 调用 sleep.sh 时，sleep.sh 将启动并保持运行状态 3 分钟（180 秒），它比 AWS CodeDeploy 预期 sleep.sh（以及相关联的 after-install.sh）停止运行的时间晚 2 分钟（120 秒）。在超时 1 分钟（60 秒）后，AWS 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 继续按预期运行也是如此。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 AWS 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。

在 after-install.sh 中，调用 sleep.sh（如下所示），后者支持 AWS 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 AWS Lambda 部署问题

主题

- 在手动终止未配置回滚的 Lambda 部署后，AWS Lambda 部署失败 (p. 318)

在手动终止未配置回滚的 Lambda 部署后，AWS Lambda 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 AWS Lambda 控制台确保函数不配置为将在两个版本之间转移流量：

- 通过以下网址登录 AWS 管理控制台并打开 AWS Lambda 控制台：<https://console.aws.amazon.com/lambda/>。
- 在左侧窗格中，选择 Functions。
- 选择位于您的 AWS CodeDeploy 部署中的 Lambda 函数的名称。
- 从 Qualifiers 中，选择在您的 AWS CodeDeploy 部署中使用的别名。
- 从 Additional Version 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。记录显示在 Version 下拉菜单中选择的版本。
- 选择 Save and test。
- 打开 AWS 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。

解决部署组问题

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

AWS 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 AWS CodeDeploy 支持将应用程序自动部署到 Auto Scaling 组中的新 Amazon EC2 实例。有关更多信息，请参阅 [将 AWS CodeDeploy 与 Auto Scaling 集成 \(p. 38\)](#)。

解决实例问题

主题

- [必须正确设置标签 \(p. 319\)](#)
- [必须安装并在实例上运行 AWS CodeDeploy 代理 \(p. 319\)](#)
- [如果实例在部署期间终止，在最多 1 小时内部署不会失败。\(p. 319\)](#)
- [分析日志文件以调查针对实例的部署失败 \(p. 319\)](#)
- [如果 AWS CodeDeploy 日志文件被意外删除，请创建一个新的日志文件 \(p. 320\)](#)
- [排查“InvalidSignatureException – Signature expired: \[time\] is now earlier than \[time\]”部署错误 \(p. 320\)](#)

必须正确设置标签

使用 `list-deployment-instances` 命令确认已正确标记用于部署的实例。如果输出中缺少 Amazon EC2 实例，请使用 Amazon EC2 控制台确认已在实例上设置标签。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 [在控制台中使用标签](#)。

Note

如果您标记一个实例并立即使用 AWS CodeDeploy 将一个应用程序部署到该实例，则该实例可能不会包含在部署中。这是因为 AWS CodeDeploy 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。

必须安装并在实例上运行 AWS CodeDeploy 代理

要验证 AWS CodeDeploy 代理是否已安装并在实例上运行，请参阅 [验证 AWS CodeDeploy 代理是否正在运行 \(p. 119\)](#)。

要安装、卸载或重新安装 AWS CodeDeploy 代理，请参阅 [安装或重新安装 AWS CodeDeploy 代理 \(p. 121\)](#)。

如果实例在部署期间终止，在最多 1 小时内部署不会失败。

AWS CodeDeploy 为每个部署生命周期事件提供 1 小时的时段来完成运行。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发生阻止脚本完成运行的任何情况（例如，如果实例终止或 AWS CodeDeploy 代理关闭），部署状态可能需 1 小时后才显示为 Failed。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，AWS CodeDeploy 代理将关闭，并且无法处理任何其他脚本。

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 AWS CodeDeploy 部署日志数据 \(p. 230\)](#)。

如果 AWS CodeDeploy 日志文件被意外删除，请创建一个新的日志文件

如果您意外删除了实例的部署日志文件，AWS CodeDeploy 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 Amazon Linux、Ubuntu Server 或 RHEL 实例，按此顺序运行这些命令（一次运行一个）：

```
sudo service codedeploy-agent stop
```

```
sudo service codedeploy-agent
```

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

AWS CodeDeploy 需要准确的时间引用以便执行其操作。如果实例的日期和时间设置不正确，它们可能与部署请求的签名日期不匹配，从而导致 AWS CodeDeploy 拒绝请求。

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 AWS CodeDeploy 应用程序在每个 AWS 区域中均使用 GitHub OAuth 令牌。使用与特定区域关联的令牌能够让您更好地控制哪些 AWS 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 [DeleteGitHubAccountToken](#) 删除旧令牌。
2. 添加新的 OAuth 令牌。有关更多信息，请参阅 [将 AWS CodeDeploy 与 GitHub 集成 \(p. 45\)](#)。

超出了 GitHub OAuth 令牌数量上限

您在创建 AWS 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 AWS CodeDeploy API 删除旧令牌的信息，请参阅 [DeleteGitHubAccountToken](#)。

解决 Auto Scaling 问题

主题

- [一般 Auto Scaling 问题排查 \(p. 321\)](#)
- [在部署修订之前，Auto Scaling 组中的实例不断预配并终止 \(p. 321\)](#)
- [终止或重启 Auto Scaling 实例可能会导致部署失败 \(p. 322\)](#)
- [避免将多个部署组与一个 Auto Scaling 组关联 \(p. 322\)](#)
- [Auto Scaling 组中的 Amazon EC2 实例无法启动，收到错误“Heartbeat Timeout” \(p. 323\)](#)
- [不匹配的 Auto Scaling 生命周期钩子可能导致针对 Auto Scaling 组的自动部署停止或失败 \(p. 323\)](#)

一般 Auto Scaling 问题排查

针对 Auto Scaling 组中的 Amazon EC2 实例的部署可能因以下原因而失败：

- Auto Scaling 持续启动和终止 Amazon EC2 实例。如果 AWS CodeDeploy 无法自动部署应用程序修订，Auto Scaling 将持续启动和终止 Amazon EC2 实例。

解除 Auto Scaling 组与 AWS CodeDeploy 部署组的关联或更改 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 Auto Scaling 启动更多的 Amazon EC2 实例）。有关更多信息，请参阅[使用 AWS CodeDeploy 更改部署组设置 \(p. 204\)](#)或[配置您的 Auto Scaling 组](#)。

- AWS CodeDeploy 代理没有响应。如果 Amazon EC2 实例启动或开始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 AWS CodeDeploy 代理可能无法安装。AWS CodeDeploy 具有 1 个小时的超时时间，以便 AWS CodeDeploy 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 AWS CodeDeploy 应用程序修订中。
- Auto Scaling 组中的 Amazon EC2 实例在部署期间将重启。如果 Amazon EC2 实例在部署期间重启或 AWS CodeDeploy 代理在处理部署命令时关闭，您的部署可能会失败。有关更多信息，请参阅[终止或重启 Auto Scaling 实例可能会导致部署失败 \(p. 322\)](#)。
- 同时向一个 Auto Scaling 组中的相同 Amazon EC2 实例部署多个应用程序修订。同时向一个 Auto Scaling 组中的相同 Amazon EC2 实例部署多个应用程序修订可能会失败（如果其中某个具有运行几分钟以上的脚本）。请不要向一个 Auto Scaling 组中的相同 Amazon EC2 实例部署多个应用程序修订。
- 对于作为 Auto Scaling 组的一部分启动的新 Amazon EC2 实例，部署将失败。通常，在这种情况下，在部署中运行脚本可能会阻止 Auto Scaling 组中的 Amazon EC2 实例启动。（Auto Scaling 组中的其他 Amazon 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 中不包括 AWS CodeDeploy 代理：如果您在启动新实例时使用 cfn-init 命令安装 AWS CodeDeploy 代理，请将代理安装脚本置于 AWS CloudFormation 模板的 cfn-init 部分的末尾。
 - 您的 AMI 中包括 AWS CodeDeploy 代理：如果您在 AMI 中包括 AWS CodeDeploy 代理，请配置该代理以使其在创建实例时处于 Stopped 状态，然后在 cfn-init 脚本库中包括一个将启动代理作为最后一步的脚本。

在部署修订之前，Auto Scaling 组中的实例不断预配并终止

在某些情况下，一个错误可能导致无法在 Auto Scaling 组中成功部署新预配的实例。结果是没有运行正常的实例，部署失败。由于无法运行部署或无法成功完成部署，实例在创建之后很快即被终止。然后 Auto Scaling 组配置会预配另一组实例，试图满足正常运行主机数量下限。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：

- 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. 手动创建一个 Amazon EC2 实例，该实例不在 Auto Scaling 组中。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。

这样 Auto Scaling 组就会将此应用程序修订部署到 Auto Scaling 组中将来的实例。

Note

确认部署成功后，您可以删除所创建的实例，避免继续产生相应费用。

终止或重启 Auto Scaling 实例可能会导致部署失败

如果通过 Auto Scaling 启动 Amazon EC2 实例，然后终止或重启该实例，则针对该实例的部署可能会因以下原因而失败：

- 在部署正在进行中时，缩小事件或任何其他终止事件将导致实例与 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。AWS CodeDeploy 将此情况视为超时。该服务将使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 Auto Scaling 配置指定 Windows Server 基础 Amazon 系统映像 (AMI)，并使用 EC2Config 服务设置实例的计算机名称，此行为可能导致部署失败。要禁用此行为，请在 Windows Server 基础 AMI 中，在 Ec2 Service Properties 对话框的 General 选项卡上，清除 Set Computer Name 框。清除此框后，将对使用该 Windows Server 基础 AMI 启动的所有新的 Windows Server Auto Scaling 实例禁用此行为。对于已启用此行为的 Windows Server Auto Scaling 实例，无需清除此框。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 Auto Scaling 组关联

作为最佳实践，您应仅为每个 Auto Scaling 组关联一个部署组。

这是因为，如果 Auto Scaling 向上扩展一个具有与多个部署组关联的钩子的实例，它会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 AWS 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

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 file 中将超时时间更改为其他值。有关更多信息，请参阅 [为 EC2/本地 部署添加 AppSpec 文件 \(p. 210\)](#)。

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，如果针对任一实例的部署失败，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 AWS CodeDeploy 具有 1 小时的超时以便 AWS CodeDeploy 代理响应挂起的部署，则每个实例的最大超时时间为 60 分钟。

有关尝试同时对一个实例进行多个部署时可能遇到的问题的更多信息，请参阅[避免对同一 Amazon EC2 实例进行并行部署 \(p. 312\)](#)。

有关 Auto Scaling 的更多信息，请参阅[深入剖析：AWS CodeDeploy 和 Auto Scaling 集成](#)。

Auto Scaling 组中的 Amazon EC2 实例无法启动，收到错误“Heartbeat Timeout”

Auto Scaling 组可能无法启动新的 Amazon EC2 实例，并生成与以下内容类似的消息：

```
Launching a new Amazon 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：

- 达到了与一个 AWS 账户关联的最大并发部署数量。有关部署限制的更多信息，请参阅[AWS CodeDeploy 限制 \(p. 307\)](#)。
- AWS 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，AWS CodeDeploy 会尝试清除与其关联的任何 Auto Scaling 钩子，但某些钩子可能会保留。如果您运行命令来删除部署组，则剩余的钩子将在输出中返回；但如果您运行命令来删除应用程序，则剩余的钩子将不会在输出中显示。

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钩子。

如果您收到了“Heartbeat Timeout”错误消息，则可通过执行以下操作来确定剩余的生命周期钩子是否为导致出现错误的原因并解决问题：

1. 运行 `update-deployment-group` 命令或 `delete-deployment-group` 命令。检查调用的输出。如果输出包含 `hooksNotCleanedUp` 结构和一个 Auto Scaling 生命周期钩子的列表，则剩余的生命周期钩子很有可能是导致错误的原因。
2. 调用 `describe-lifecycle-hooks` 命令，并指定与无法启动的 Amazon EC2 实例关联的 Auto Scaling 组的名称。在输出中，查找与您在步骤 1 中标识的 `hooksNotCleanedUp` 结构对应的任何 Auto Scaling 生命周期钩子名称。或者，查找包含部署组名称的 Auto Scaling 生命周期钩子名称。
3. 对每个 Auto Scaling 生命周期钩子调用 `delete-lifecycle-hook` 命令。指定 Auto Scaling 组和生命周期钩子。

如果您删除（从 Auto Scaling 组中）由 AWS CodeDeploy 创建的所有 Auto Scaling 生命周期钩子，则 AWS CodeDeploy 将不再针对作为 Auto Scaling 组的一部分向上扩展的 Amazon EC2 实例进行部署。

不匹配的 Auto Scaling 生命周期钩子可能导致针对 Auto Scaling 组的自动部署停止或失败

Auto Scaling 和 AWS CodeDeploy 使用生命周期钩子确定应将哪些应用程序修订部署到哪些已在 Auto Scaling 组中启动的 Amazon EC2 实例。如果 Auto Scaling 和 AWS CodeDeploy 中的生命周期钩子及其相关信息未准确匹配，则自动部署可能会停止或失败。

如果针对某个 Auto Scaling 组的部署失败，请检查 Auto Scaling 和 AWS CodeDeploy 中的生命周期钩子名称是否匹配。如果不匹配，请使用这些 AWS CLI 命令调用。

首先，获取 Auto Scaling 组和部署组的生命周期钩子名称的列表：

1. 调用 [describe-lifecycle-hooks](#) 命令，并在 AWS CodeDeploy 中指定与部署组关联的 Auto Scaling 组的名称。在输出中，在 LifecycleHooks 列表中，记下每个 LifecycleHookName 值。
2. 调用 [get-deployment-group](#) 命令，并指定与 Auto Scaling 组关联的部署组的名称。在输出中，在 autoScalingGroups 列表中，查找名称值与 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钩子的名称。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 Auto Scaling 问题排查步骤。

但是，如果两组生命周期钩子的名称未完全匹配（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钩子名称，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钩子名称，请调用 [delete-lifecycle-hook](#) 命令。
 - b. 调用 [update-deployment-group](#) 命令，并指定原始 Auto Scaling 组的名称。AWS CodeDeploy 将在 Auto Scaling 组中创建新的替代生命周期钩子，并将这些生命周期钩子与部署组关联。现在，自动部署应恢复，因为新的实例已添加到 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钩子名称，则执行以下操作：
 - a. 调用 [update-deployment-group](#) 命令，但不指定原始 Auto Scaling 组的名称。
 - b. 再次调用 [update-deployment-group](#) 命令，但这次指定原始 Auto Scaling 组的名称。AWS CodeDeploy 将在 Auto Scaling 组中重新创建缺失的生命周期钩子。现在，自动部署应恢复，因为新的实例已添加到 Auto Scaling 组。

在您将两组生命周期钩子名称完全匹配后（字符对字符），应用程序修订应重新部署，但仅重新部署到新的实例，因为它们已被添加到 Auto Scaling 组。将不会自动针对 Auto Scaling 组中已包含的实例进行部署。

AWS CodeDeploy 的错误代码

本主题提供有关 AWS CodeDeploy 错误的参考信息。

错误代码	说明
AGENT_ISSUE	<p>部署因 AWS CodeDeploy 代理出现问题而失败。确保此代理已安装并在该部署组中的所有实例上运行。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 验证 AWS CodeDeploy 代理是否正在运行 (p. 119)• 安装或重新安装 AWS CodeDeploy 代理 (p. 121)• 使用 AWS CodeDeploy 代理 (p. 113)
HEALTH_CONSTRAINTS	<p>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。</p> <p>了解更多：</p> <ul style="list-style-type: none">• Instance Health (p. 180)

错误代码	说明
	<ul style="list-style-type: none"> • 解决实例问题 (p. 319) • 排查 EC2/本地 部署问题 (p. 313)
HEALTH_CONSTRAINTS_INVALID	<p>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。</p> <p>了解更多：</p> <ul style="list-style-type: none"> • Instance Health (p. 180) • 使用适用于 AWS CodeDeploy 的实例 (p. 133)
IAM_ROLE_MISSING	<p>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。</p> <p>了解更多：</p> <ul style="list-style-type: none"> • 步骤 3：为 AWS CodeDeploy 创建服务角色 (p. 18) • 使用 AWS CodeDeploy 更改部署组设置 (p. 204)
IAM_ROLE_PERMISSIONS	<p>AWS CodeDeploy 没有代入角色所需的权限，或者您正在使用的 IAM 角色未为您提供在 AWS 服务中执行操作的权限。</p> <p>了解更多：</p> <ul style="list-style-type: none"> • 步骤 1：预置 IAM 用户 (p. 16) • 步骤 3：为 AWS CodeDeploy 创建服务角色 (p. 18) • 步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 (p. 22)
AUTO_SCALING_IAM_ROLE_PERMISSIONS	<p>与您的部署组关联的服务角色不具备在以下 AWS 服务中执行操作所需的权限。</p> <p>了解更多：</p> <ul style="list-style-type: none"> • 步骤 3：为 AWS CodeDeploy 创建服务角色 (p. 18) • 创建向 AWS 服务委托权限的角色
OVER_MAX_INSTANCES	<p>由于为部署定向的实例数超出您的账户允许的数量，因此部署失败。要减少为此部署定向的实例数，请更新此部署组的标签设置或删除一些定向实例。或者，您可联系 AWS Support 以请求提高限制。</p> <p>了解更多：</p> <ul style="list-style-type: none"> • 使用 AWS CodeDeploy 更改部署组设置 (p. 204) • AWS CodeDeploy 限制 (p. 307) • 请求提高限制

错误代码	说明
THROTTLED	<p>由于 IAM 角色发出的请求的数目超出 AWS CodeDeploy 允许的数目，因此部署失败。尝试减少请求数。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 查询 API 请求速率
UNABLE_TO_SEND_ASG	<p>由于部署组未与其 Auto Scaling 组正确配置，因此部署失败。在 AWS CodeDeploy 控制台中，从部署组中删除 Auto Scaling 组，然后重新添加此组。</p> <p>了解更多：</p> <ul style="list-style-type: none">• 深入剖析：AWS CodeDeploy 和 Auto Scaling 集成

相关主题

[AWS CodeDeploy 问题排查 \(p. 310\)](#)

AWS CodeDeploy 资源

下列相关资源在您使用 AWS CodeDeploy 的过程中会有所帮助。

参考指南和支持资源

- [AWS CodeDeploy API Reference](#) - 有关 AWS 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。
- [AWS CodeDeploy 技术常见问题](#) - 客户提出的有关 AWS CodeDeploy 的常见问题。
- [AWS CodeDeploy 发行说明](#) - 对当前版本和旧版本的高级概述（有关新功能、更正和已知问题的特别说明）。
- [AWS Support 中心](#) - 用于创建和管理 AWS Support 案例的中心。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 AWS Trusted Advisor。
- [AWS Support 计划](#) - 包含有关 AWS Support 计划的信息的主要网页。
- [联系我们](#) - 用于查询有关 AWS 账单、账户、事件、滥用和其他问题的中央联系点。
- [AWS 网站条款](#) - 有关我们的版权和商标、您的账户、许可和网站访问以及其他主题的详细信息。

示例

- [GitHub 上的 AWS CodeDeploy 示例](#) - AWS CodeDeploy 的示例和模板方案。
- [AWS CodeDeploy Jenkins 插件](#) - AWS CodeDeploy 的 Jenkins 插件。
- [AWS CodeDeploy 代理](#) - AWS CodeDeploy 代理的开源版本。

博客

- [AWS 开发运营博客](#) - 面向开发人员、系统管理员和架构师提供深入的见解。

AWS 软件开发工具包和工具

以下 AWS 软件开发工具包和工具支持使用 AWS CodeDeploy 开发解决方案：

- [适用于 .NET 的 AWS 开发工具包](#)
- [AWS SDK for Java](#)
- [适用于 JavaScript 的 AWS 开发工具包](#)
- [适用于 PHP 的 AWS 开发工具包](#)
- [AWS SDK for Python \(Boto\)](#)
- [适用于 Ruby 的 AWS 开发工具包](#)
- [AWS Toolkit for Eclipse](#) - 第 1 部分、第 2 部分和第 3 部分。
- [适用于 Windows PowerShell 的 AWS 工具](#) - 一组 Windows PowerShell cmdlet，用于在 PowerShell 环境中公开适用于 .NET 的 AWS 开发工具包的功能。
- [适用于 PowerShell 的 AWS 工具中的 AWS CodeDeploy Cmdlet](#) - 一组 Windows PowerShell cmdlet，用于在 PowerShell 环境中公开 AWS CodeDeploy 的功能。

- [AWS Command Line Interface](#) - 用于访问 AWS 服务的一致命令行语法。AWS CLI 使用单一设置过程来启用对所有支持的服务的访问。
- [AWS 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 AWS CodeDeploy 和 AWS 构建创新性应用程序的其他信息。

文档历史记录

下表描述了对此用户指南进行的重大更改，以阐明自上次发布 AWS CodeDeploy 用户指南以来的新增及增强功能。

- API 版本：2014-10-06
- 最近文档更新时间：2018 年 8 月 7 日

update-history-change	update-history-description	update-history-date
AWS CodeDeploy 代理最新的最低支持版本 (p. 329)	当前所支持的 AWS CodeDeploy 代理的最低版本为 。有关更多信息，请参阅 AWS CodeDeploy 代理的版本历史纪录 。	August 7, 2018

早期更新

下表描述了 2018 年 6 月之前的每个 AWS CodeDeploy 用户指南版本中的重要更改。

更改	描述	更改日期
主题更新	AWS CodeDeploy 现在可在 (eu-west-3) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。	2017 年 12 月 19 日
对主题进行了更新	<p>AWS CodeDeploy 目前在中国 (宁夏) 区域中可用。</p> <p>要在 中国 (北京) 区域 或 中国 (宁夏) 区域 中使用服务，您必须拥有这些区域的账户和凭据。其他 AWS 区域的账户和凭证不适用于 北京和宁夏区域，反之亦然。</p> <p>有关 中国区域 的一些资源的信息 (例如 AWS CodeDeploy 资源工具包存储桶名称和 AWS CodeDeploy 代理安装过程) 不包含在此版本的 AWS CodeDeploy 用户指南 中。</p> <p>有关更多信息：</p> <ul style="list-style-type: none">• 在中国 (北京) 区域 开始使用 AWS 中的 AWS CodeDeploy• 中国区域的 AWS CodeDeploy 用户指南 (英语版本 中文版本)	2017 年 12 月 11 日
对主题进行了更新	AWS CodeDeploy 现已支持部署 Lambda 函数。AWS Lambda 部署可将传入流量从现有 Lambda 函数转移到更新的 Lambda 函数版本。您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。AWS 无服务器应用程序模型 (AWS SAM) 支持 AWS Lambda 部署，以便您可以使用 AWS SAM 部署首选项来管理 AWS Lambda 部署期间流量转移的方式。为反映这一变动，新增和更新了若干个主题，其中包括 AWS CodeDeploy 计算平台概览 (p. 2) 、 AWS Lambda 计算平台 上的部署 (p. 8) 、 创建 AWS Lambda 计算平台 部署 (控制台) (p. 223) 、 为 AWS Lambda 函数部署创建应用程	2017 年 11 月 28 日

更改	描述	更改日期
	序 (控制台) (p. 194) 和为 AWS Lambda 部署添加 AppSpec 文件 (p. 209)。	
新主题	AWS CodeDeploy 现已支持直接部署到本地计算机或安装了 AWS CodeDeploy 代理的实例。您可以在本地测试部署，如果有错误，请使用 AWS 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 file 的内容以及更多内容。有关更多信息，请参阅 使用 AWS CodeDeploy 代理验证本地机器上的部署程序包 (p. 238) 。	2017 年 11 月 16 日
对主题进行了更新	AWS CodeDeploy 对部署组中 Elastic Load Balancing 负载均衡器的支持已扩展，以便蓝/绿部署和就地部署均包括 Network Load Balancer。现在，可以为您的部署组选择 应用程序负载均衡器、传统负载均衡器 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 AWS CodeDeploy with Elastic Load Balancing (p. 40) 、 为就地部署创建应用程序 (控制台) (p. 190) 、 部署先决条件 (p. 221) 、 Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40) 和 为就地部署创建应用程序 (控制台) (p. 190) 。	2017 年 9 月 12 日
对主题进行了更新	AWS CodeDeploy 对部署组中 Elastic Load Balancing 负载均衡器的支持已扩展到蓝/绿部署和就地部署均包括 Application Load Balancer。现在，可以为您的部署组选择 应用程序负载均衡器 或 传统负载均衡器。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 AWS CodeDeploy with Elastic Load Balancing (p. 40) 、 使用 AWS CodeDeploy 创建应用程序 (p. 189) 和 使用 AWS CodeDeploy 创建部署组 (p. 198) 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	AWS 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 AWS CodeDeploy Deployments (p. 134) 。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. 190) 、 为蓝/绿部署创建应用程序 (控制台) (p. 192) 、 为就地部署创建部署组 (控制台) (p. 199) 、 为蓝/绿部署创建部署组 (控制台) (p. 200) 、 Deployments (p. 8) ，和 教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序 (p. 101) 中的 步骤 5：创建应用程序和部署组 (p. 106) 。	2017 年 31 月 7 日
对主题进行了更新	用于在 Windows Server 实例上安装 AWS CodeDeploy 代理的另外两个方法已添加到 安装或重新安装适用于 Windows Server 的 AWS CodeDeploy 代理 (p. 123) 中。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

更改	描述	更改日期
对主题进行了更新	AWS CodeDeploy 改进了管理与 GitHub 账户和存储库的连接的方法。现在，您可以创建和存储最多 25 个与 GitHub 账户的连接以便将 AWS 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，AWS 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 存储库中修订的信息 (p. 225) 、 将 AWS CodeDeploy 与 GitHub 集成 (p. 45) 和 教程：使用 AWS 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
对主题进行了更新	过去，如果 AWS 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。AWS 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 AWS CodeDeploy 创建部署 (p. 220) 已更新以反映此支持，并且新的 回滚行为与现有内容 (p. 234) 部分已添加到 使用 AWS CodeDeploy 重新部署和回滚部署 (p. 233) 。	2017 年 5 月 16 日
对主题进行了更新	现在可以使用 AWS CodeDeploy 控制台或 AWS CLI 将 Elastic Load Balancing 中的 传统负载均衡器 分配给部署组。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 AWS 服务集成 (p. 35) 、 Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40) 、 为就地部署创建应用程序 (控制台) (p. 190) 、 为就地部署创建部署组 (控制台) (p. 199) 和 AppSpec 的“hooks”部分 (p. 284)。已向故障排除指南添加一个新的部分： 对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 (p. 315) 。	2017 年 4 月 27 日
对主题进行了更新	现在可以使用 AWS CodeDeploy 控制台或 AWS CLI 将 Elastic Load Balancing 中的 传统负载均衡器 分配给部署组。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 AWS 服务集成 (p. 35) 、 Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40) 、 为就地部署创建应用程序 (控制台) (p. 190) 、 为就地部署创建部署组 (控制台) (p. 199) 和 AppSpec 的“hooks”部分 (p. 284)。已向故障排除指南添加一个新的部分： 对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 (p. 315) 。	2017 年 5 月 1 日

更改	描述	更改日期
对主题进行了更新	<p>AWS CodeDeploy 目前在中国（北京）区域中可用。</p> <p>要在中国（北京）区域或中国（宁夏）区域中使用服务，您必须拥有这些区域的账户和凭据。其他 AWS 区域的账户和凭证不适用于北京和宁夏区域，反之亦然。</p> <p>有关中国区域的一些资源的信息（例如 AWS CodeDeploy 资源工具包存储桶名称和 AWS CodeDeploy 代理安装过程）不包含在此版本的 AWS CodeDeploy 用户指南中。</p> <p>有关更多信息：</p> <ul style="list-style-type: none"> 在中国（北京）区域开始使用 AWS 中的 AWS CodeDeploy 中国区域的 AWS CodeDeploy 用户指南（英语版本 中文版） 	2017 年 3 月 29 日
新增和更新的主体	<p>推出了若干新主题以反映对蓝/绿部署的新增 AWS CodeDeploy 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. 4）提供了对 AWS CodeDeploy 使用的蓝/绿方法的概要说明。在 AWS CodeDeploy 中尝试使用示例蓝/绿部署（p. 27）提供了对蓝/绿部署使用新的 Sample deployment wizard 的指南。其他新主题包括 为蓝/绿部署创建应用程序（控制台）（p. 192）。为蓝/绿部署创建部署组（控制台）（p. 200），和在 Elastic Load Balancing 中为 AWS CodeDeploy 部署设置 Load Balancer（p. 202）。</p> <p>还更新了许多主题，其中包括 使用 AWS CodeDeploy 创建部署（p. 220）、在 AWS CodeDeploy 中使用部署配置（p. 184）、使用 AWS CodeDeploy 创建应用程序（p. 189）、在 AWS CodeDeploy 中使用部署组（p. 198）、在 AWS CodeDeploy 中使用部署（p. 220）和 AppSpec 的“hooks”部分（p. 284）。</p>	2017 年 1 月 25 日
新增和更新的主体	<p>新增主题使用 register-on-premises-instance 命令（IAM 会话 ARN）注册本地实例（p. 169），介绍了如何使用通过 AWS 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. 156）的内容也已更新，以反映这一新的支持功能。</p>	2016 年 12 月 28 日
对主题进行了更新	<p>AWS CodeDeploy 现在可在 欧洲（伦敦）区域（eu-west-2）中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p>	2016 年 12 月 13 日
对主题进行了更新	<p>AWS CodeDeploy 现在可在 加拿大（中部）区域（ca-central-1）中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p>	2016 年 12 月 8 日
对主题进行了更新	<p>AWS CodeDeploy 现在可在 美国东部（俄亥俄）区域（us-east-2）中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p>	2016 年 10 月 17 日

更改	描述	更改日期
新主题	新的章节 AWS CodeDeploy 的身份验证和访问控制 (p. 260) 提供了有关使用 AWS Identity and Access Management (IAM) 和 AWS CodeDeploy 通过使用凭据帮助保护对您的资源的访问的全面地信息。这些凭据提供访问 AWS 资源所需的权限，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
对主题进行了更新	更新 Windows Server 上的 AWS CodeDeploy 代理 (p. 128) 进行了更新，以说明 Windows Server 的新 AWS CodeDeploy 代理更新程序的可用性。在 Windows Server 实例上安装后，更新程序将定期检查新版本。当检测到新版本时，更新程序将在安装最新版本之前，卸载当前版本的代理（如果已安装）。	2016 年 10 月 4 日
对主题进行了更新	<p>AWS 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。</p> <p>AWS 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。</p> <p>已更新了大量主题来反映这些更改，包括使用 AWS CodeDeploy 创建应用程序 (p. 189)、使用 AWS CodeDeploy 创建部署组 (p. 198)、使用 AWS CodeDeploy 更改部署组设置 (p. 204)、Deployments (p. 8)、使用 AWS CodeDeploy 重新部署和回滚部署 (p. 233)和AWS CodeDeploy 产品和服务集成 (p. 35)，此外新增了主题在 AWS CodeDeploy 中使用 CloudWatch 警报监控部署 (p. 245)。</p>	2016 年 9 月 15 日
新增和更新的主题	AWS CodeDeploy 现在提供与 Amazon CloudWatch Events 的集成。现在，当检测到 AWS CodeDeploy 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 Events 启动一个或多个操作。您可以在操作中调用 AWS 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. 246) 。	2016 年 9 月 9 日
主题更新	更新了 Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40) 和 与其他 AWS 服务集成 (p. 35) 主题，以反映另增的一个负载均衡选项。AWS CodeDeploy 现在支持 Elastic Load Balancing 中可用的 传统负载均衡器 和 应用程序负载均衡器。	2016 年 8 月 11 日
主题更新	AWS CodeDeploy 现在可在亚太地区（孟买）区域 (ap-south-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。	2016 年 6 月 27 日

更改	描述	更改日期
主题更新	<p>AWS CodeDeploy 现在可在 亚太区域 (首尔) (ap-northeast-2) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p> <p>已重新组织内容表，来包含针对实例、部署配置、应用程序、部署组、修订和部署的各个部分。已为 AWS CodeDeploy 教程添加新的部分。为了提高可用性，已将几个较长的主题 (包括 AWS CodeDeploy AppSpec File 参考 (p. 275) 和 AWS CodeDeploy 问题排查 (p. 310)) 划分为多个较短的主题。AWS CodeDeploy 代理的配置信息已移至新主题 AWS CodeDeploy 代理配置参考 (p. 297)。</p>	2016 年 6 月 15 日
新增和更新的主题	<p>AWS CodeDeploy 的错误代码 (p. 324) 提供了 AWS CodeDeploy 部署失败时可能显示的一些错误消息的相关信息。AWS CodeDeploy 问题排查 (p. 310) 中的下列部分已更新，现在能够更好地帮助解决部署问题：</p> <ul style="list-style-type: none"> • Auto Scaling 组中的 Amazon EC2 实例无法启动，收到错误“Heartbeat Timeout” (p. 323) • 避免对同一 Amazon EC2 实例进行并行部署 (p. 312) • 避免将多个部署组与一个 Auto Scaling 组关联 (p. 322) 	2016 年 4 月 20 日
主题更新	<p>AWS CodeDeploy 现在可在 南美洲 (圣保罗) 区域 (sa-east-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p> <p>使用 AWS CodeDeploy 代理 (p. 113) 进行了更新，用于说明新的 :max_revisions: 配置选项，您可以使用此选项来指定希望 AWS CodeDeploy 代理对一个部署组进行归档的应用程序修订数。</p>	2016 年 3 月 10 日
新增和更新的主题	<p>AWS 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。对于您加入到该触发器操作中的 Amazon Simple Notification Service 主题，通知将发送到已订阅该主题的接收人。您也可以使用当触发器在您自己的自定义通知工作流中触发时创建的 JSON 数据。有关更多信息，请参阅 Monitoring Deployments with Amazon SNS Event Notifications (p. 250)。</p> <p>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。</p> <p>AWS CodeDeploy 问题排查 (p. 310) 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. 319) 部分已更新。</p> <p>对 AWS CodeDeploy 限制 (p. 307) 进行了更新，来说明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 AWS SDK for Ruby (aws-sdk-core) 要求的版本。</p>	2016 年 2 月 17 日

更改	描述	更改日期
新增和更新的主题	<p>AWS CodeDeploy 现在可在 美国西部 (加利福尼亚北部) 区域 (us-west-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新增的区域。</p> <p>选择 AWS CodeDeploy 存储库类型 (p. 213)列出并描述了 AWS CodeDeploy 目前支持的存储库类型。在引入对其他存储库的支持时，这一新主题将会进行更新。</p> <p>管理 AWS CodeDeploy 代理操作 (p. 118) 中更新了有关新 .version 文件的信息，该文件添加到实例中以报告当前 AWS CodeDeploy 代理的版本；此外还更新了有关所支持代理版本的信息。</p> <p>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。</p> <p>将应用程序规范文件添加到 AWS CodeDeploy 的修订 (p. 209)已按照分步说明的方式重新整理。</p>	2016 年 1 月 20 日
新主题	在其他 AWS 账户中部署应用程序 (p. 235) 描述了在不需要其他账户的完整凭证集的情况下，用于启动属于您组织中其他账户的部署时的设置要求和过程。这对于将多个账户用于不同用途的组织尤为有用，例如一个账户与开发和测试环境关联，另一个账户与生产环境关联。	2015 年 12 月 30 日
主题更新	AWS CodeDeploy 产品和服务集成 (p. 35) 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 AWS 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	AWS CodeDeploy 现在可在 亚太区域 (新加坡) (ap-southeast-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS 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 AWS CodeDeploy 代理 (p. 113) 进行了更新，以说明 AWS CodeDeploy 代理配置文件中的新 :proxy_uri: 选项。 AWS CodeDeploy AppSpec File 参考 (p. 275) 中更新了有关使用新环境变量 DEPLOYMENT_GROUP_ID 的信息；在部署生命周期事件期间，挂钩脚本可以访问该变量。	2015 年 12 月 1 日
主题更新	步骤 3：为 AWS CodeDeploy 创建服务角色 (p. 18) 进行了更新，以说明为 AWS CodeDeploy 创建服务角色的新过程并介绍其他一些改进。	2015 年 11 月 13 日
主题更新	<p>AWS CodeDeploy 现在可在 欧洲 (法兰克福) 区域 (eu-central-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这一新的可用区域。</p> <p>AWS CodeDeploy 问题排查 (p. 310)主题更新了有关确保实例的时间设置准确无误的信息。</p>	2015 年 10 月 19 日
新主题	<p>发布了适用于 AWS CodeDeploy 的 AWS CloudFormation 模板参考 (p. 300)，用于说明 AWS CloudFormation 对 AWS CodeDeploy 操作的新增支持。</p> <p>创建了Primary Components (p. 6)主题，并引入了目标修订的定义。</p>	2015 年 10 月 1 日

更改	描述	更改日期
主题更新	<p>使用 AWS CodeDeploy 创建部署组 (p. 198)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。</p> <p>Instance Health (p. 180)进行了更新，以澄清正常运行的最少实例数概念。</p>	2015 年 8 月 31 日
主题更新	AWS CodeDeploy 现在可在 亚太区域 (东京) (ap-northeast-1) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS 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	<p>AWS CodeDeploy 现在支持部署到 Red Hat Enterprise Linux (RHEL) 本地实例和 Amazon EC2 实例。有关更多信息，请参阅以下主题：</p> <ul style="list-style-type: none"> • AWS CodeDeploy 代理支持的操作系统 (p. 113) • 使用适用于 AWS CodeDeploy 的实例 (p. 133) • 教程：将 WordPress 部署到 Amazon EC2 实例 (Amazon Linux 或 Red Hat Enterprise Linux 和 Linux, macOS, or Unix) (p. 49) • 教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 (Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux) (p. 79) 	2015 年 6 月 23 日
主题更新	<p>AWS 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，例如当前 AWS CodeDeploy 应用程序、部署组和部署生命周期事件的名称，以及当前 AWS CodeDeploy 部署标识符。有关更多信息，请参阅AWS CodeDeploy AppSpec File 参考 (p. 275)中 AppSpec 的“hooks”部分 (p. 284)部分的结尾。</p>	2015 年 5 月 29 日
主题更新	<p>AWS CodeDeploy 现在提供了 IAM 中的一组 AWS 托管策略，您可以使用这些策略而无需自行手动创建等同策略。包括：</p> <ul style="list-style-type: none"> • 一个用于允许 IAM 用户仅将修订注册到 AWS CodeDeploy 并随后通过 AWS CodeDeploy 进行部署的策略。 • 一个用于向 IAM 用户提供对 AWS CodeDeploy 资源的完整访问权限的策略。 • 一个用于向 IAM 用户提供对 AWS CodeDeploy 资源的只读访问权限的策略。 • 一个用于附加服务角色的策略，这样 AWS CodeDeploy 可以通过实例的 Amazon EC2 标签、本地实例标签或 Auto Scaling 组名来识别 Amazon EC2 实例，并相应对其部署应用程序修订。 <p>有关更多信息，请参阅AWS CodeDeploy 的身份验证和访问控制 (p. 260)中的客户托管策略示例 (p. 267)部分。</p>	2015 年 5 月 29 日
主题更新	AWS CodeDeploy 现在可在 欧洲 (爱尔兰) 区域 (eu-west-1) 和 亚太区域 (悉尼) (ap-southeast-2) 中使用。有多个主题进行了更新，包括介绍如何设置 AWS CodeDeploy 代理的主题，来说明这些新的可用区域。	2015 年 5 月 7 日

更改	描述	更改日期
新主题	<p>AWS CodeDeploy 现在支持部署到本地实例和 Amazon EC2 实例。增加了以下主题来描述这一新的支持：</p> <ul style="list-style-type: none"> • Working with On-Premises Instances (p. 156) • 教程：使用 AWS CodeDeploy 将应用程序部署到本地实例 (Windows Server、Ubuntu Server 或 Red Hat Enterprise Linux) (p. 79) • Working with On-Premises Instances (p. 156) 	2015 年 4 月 2 日
新主题	增加了 AWS CodeDeploy 资源 (p. 327)。	2015 年 4 月 2 日
主题更新	<p>更新了AWS CodeDeploy 问题排查 (p. 310)：</p> <ul style="list-style-type: none"> • 新的长时间运行的进程可能会导致部署失败 (p. 317)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。 • 一般 Auto Scaling 问题排查 (p. 321)部分进行了更新，以说明 AWS CodeDeploy 将其针对 AWS CodeDeploy 代理的 Auto Scaling 超时逻辑从 5 分钟增加到 1 小时。 • 新的不匹配的 Auto Scaling 生命周期钩子可能导致针对 Auto Scaling 组的自动部署停止或失败 (p. 323)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 Auto Scaling 组时的故障。 	2015 年 4 月 2 日
主题更新	<p>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：</p> <ul style="list-style-type: none"> • 配置用于 AWS CodeDeploy 的 Amazon EC2 实例 (p. 153) • 步骤 4：为 Amazon EC2 实例创建 IAM 实例配置文件 (p. 22) • 步骤 3：为 AWS CodeDeploy 创建服务角色 (p. 18) • AWS CodeDeploy 的身份验证和访问控制 (p. 260) <p>AWS CodeDeploy 问题排查 (p. 310)中增加了两个部分：</p> <ul style="list-style-type: none"> • 一般问题排查核对清单 (p. 310) • 默认情况下，Windows PowerShell 脚本无法使用 64 位版本的 Windows PowerShell (p. 316) <p>AWS CodeDeploy AppSpec File 参考 (p. 275)中的AppSpec 的“hooks”部分 (p. 284)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。</p>	2015 年 2 月 12 日
主题更新	<p>AWS CodeDeploy 问题排查 (p. 310)：Auto Scaling 组中的 Amazon EC2 实例无法启动，收到错误“Heartbeat Timeout” (p. 323)中增加了新的部分。</p> <p>AWS CodeDeploy 产品和服务集成 (p. 35)中增加了 CloudBees 部分。</p>	2015 年 1 月 28 日

更改	描述	更改日期
主题更新	<p>AWS CodeDeploy 问题排查 (p. 310)中增加了以下部分：</p> <ul style="list-style-type: none"> 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. 312) 使用 macOS 中的 Finder 捆绑应用程序修订可能会导致部署失败 (p. 313) 对失败的 ApplicationStop、BeforeBlockTraffic 和 AfterBlockTraffic 部署生命周期事件进行故障排除 (p. 315) 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. 316) 一般 Auto Scaling 问题排查 (p. 321) <p>步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 (p. 25)中增加了相关信息，来澄清发出调用的 IAM 用户所需的特定权限，具体而言：</p> <ul style="list-style-type: none"> 步骤 3：配置实例 (p. 32)说明了使用演练 AWS CloudFormation 模板所需的特定权限。 步骤 7：选择服务角色 (p. 33)说明了在演练过程中创建服务角色所需的特定权限。 步骤 9：审核部署详细信息 (p. 33)说明了创建应用程序和部署组以及部署应用程序所需的特定权限。 <p>有关所需权限的信息，请参阅先决条件 (p. 26)。</p>	2015 年 1 月 20 日
新主题	<p>AWS CodeDeploy 产品和服务集成 (p. 35)部分进行了更新，现在包括下列主题：</p> <ul style="list-style-type: none"> 将 AWS CodeDeploy 与 Auto Scaling 集成 (p. 38) 教程：使用 AWS CodeDeploy 将应用程序部署到 Auto Scaling 组 (p. 86) Monitoring Deployments with AWS CloudTrail (p. 248) Integrating AWS CodeDeploy with Elastic Load Balancing (p. 40) 将 AWS CodeDeploy 与 GitHub 集成 (p. 45) 教程：使用 AWS CodeDeploy 从 GitHub 部署应用程序 (p. 101) 	2015 年 1 月 9 日
主题更新	<ul style="list-style-type: none"> 将 AWS CodeDeploy 与 GitHub 集成 (p. 45)中增加了使用 AWS CodeDeploy 自动从 GitHub 部署 (p. 47)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。 AWS CodeDeploy 问题排查 (p. 310)中增加了解决 Auto Scaling 问题 (p. 321)部分。这一新部分描述了如何排除在部署到 Auto Scaling 组时遇到的常见问题。 AWS CodeDeploy AppSpec File 参考 (p. 275)的 AppSpec 的“files”部分 (仅限 EC2/本地 部署) (p. 277)部分中增加了新的子部分“文件示例”。这一新的子部分中包含多个示例，介绍如何使用 AppSpec file 的 files 部分来指示 AWS CodeDeploy 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

更改	描述	更改日期
新主题	增加了 Monitoring Deployments with AWS CloudTrail (p. 248) 。AWS CodeDeploy 与 AWS CloudTrail 相集成，后者是一种服务，可在 AWS 账户中捕获由 AWS CodeDeploy 自身或代表其发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。	2014 年 12 月 17 日
主题更新	更新了 步骤 5：尝试使用 AWS CodeDeploy 示例部署向导 (p. 25) 中的 步骤 3：配置实例 (p. 32) 部分。	2014 年 12 月 3 日
第一个公开发行版	这是 AWS CodeDeploy 用户指南的第一个公开发行版。	2014 年 11 月 12 日

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 [AWS 词汇表](#)。