

---

# AWS CodeBuild

## 用户指南

### API 版本 2016-10-06



## AWS CodeBuild: 用户指南

Copyright © 2020 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 CodeBuild ? .....	1
如何运行 CodeBuild .....	1
CodeBuild 定价 .....	2
如何开始使用 CodeBuild ? .....	2
概念 .....	2
CodeBuild 的工作原理 .....	3
后续步骤 .....	3
入门 .....	4
通过控制台开始使用 .....	4
步骤 .....	4
步骤 1：创建两个 S3 存储桶 .....	4
步骤 2：创建源代码 .....	5
步骤 3：创建构建规范文件 .....	7
步骤 4：上传源代码和构建规范文件 .....	8
步骤 5：创建构建项目 .....	9
步骤 6：运行构建 .....	10
步骤 7：查看汇总的构建信息 .....	11
步骤 8：查看详细的构建信息 .....	11
步骤 9：获取构建输出构件 .....	12
步骤 10：删除 S3 输入存储桶 .....	12
总结 .....	13
开始使用 AWS CLI .....	13
Steps .....	13
步骤 1. 创建两个S3存储桶 .....	14
步骤 2. 创建源代码 .....	14
步骤 3 创建BuildSpec文件 .....	16
步骤 4. 上传源代码和BuildSpec文件 .....	18
步骤 5. 创建构建项目 .....	18
步骤 6. 运行构建。 .....	21
步骤 7. 查看汇总的构建信息 .....	22
步骤 8。查看详细的构建信息 .....	24
步骤 9 获取构建输出伪影 .....	25
步骤 10 删除S3输入桶 .....	26
总结 .....	26
示例 .....	27
Windows 示例 .....	27
运行示例 .....	27
目录结构 .....	28
Files .....	29
基于使用案例的示例 .....	44
访问令牌示例 .....	46
Amazon ECR 示例 .....	50
Amazon EFS 示例 .....	53
AWS CodeDeploy 示例 .....	57
AWS CodePipeline 与批次构建样品集成 .....	60
AWS CodePipeline 与多输入源和输出构件集成示例 .....	64
AWS Config 示例 .....	66
AWS Elastic Beanstalk 示例 .....	68
Bitbucket 拉取请求和 Webhook 筛选条件示例 .....	75
构建徽章示例 .....	78
构建通知示例 .....	80
使用 AWS CLI 示例创建测试报告 .....	97
自定义映像示例中的 Docker .....	102
Docker 示例 .....	104

GitHub Enterprise Server 示例 .....	109
GitHub 拉取请求和 Webhook 筛选条件示例 .....	115
将构建输出托管在 S3 存储桶中 .....	118
构建规范文件示例中的运行时版本 .....	120
源版本示例 .....	127
私有注册表与 AWS Secrets Manager 示例 .....	129
多输入源和输出构件示例 .....	131
使用语义版本控制命名构建构件示例 .....	133
计划构建 .....	135
构建规范参考 .....	136
构建规范文件名称和存储位置 .....	136
构建规范语法 .....	137
构建规范示例 .....	152
构建规范版本 .....	154
批量构建参考规格 .....	155
构建环境参考 .....	159
CodeBuild 提供的 Docker 映像 .....	159
构建环境计算类型 .....	166
构建环境中的 Shell 和命令 .....	167
构建环境中的环境变量 .....	168
构建环境中的后台任务 .....	170
在本地测试和调试 .....	171
使用 CodeBuild 代理在本地计算机上测试和调试 .....	171
接收有关新的 CodeBuild 代理版本的通知 .....	171
VPC 支持 .....	173
使用案例 .....	173
在 CodeBuild 项目中允许 Amazon VPC 访问 .....	173
VPC 的最佳实践 .....	174
排查 VPC 设置的问题 .....	175
使用 VPC 终端节点 .....	175
在您创建 VPC 终端节点前 .....	175
创建 CodeBuild 的 VPC 终端节点 .....	176
为 CodeBuild 创建 VPC 终端节点策略 .....	176
AWS CloudFormation VPC 模板 .....	177
使用代理服务器 .....	180
在代理服务器中运行 CodeBuild 所需的组件 .....	181
在显式代理服务器中运行 CodeBuild .....	184
在透明代理服务器中运行 CodeBuild .....	186
在代理服务器中运行程序包管理器和其他工具 .....	187
使用构建项目和构建 .....	189
使用构建项目 .....	189
创建构建项目 () .....	189
Create a notification rule .....	209
查看构建项目名称的列表 () .....	211
查看构建项目的详细信息 .....	213
构建缓存 .....	214
创建构建触发器 .....	218
编辑构建触发器 .....	220
BitbucketWebHook 事件 .....	221
GitHub 钩型活动 .....	229
更改构建项目的设置 () .....	236
删除构建项目 .....	244
使用共享项目。 .....	245
标记项目 .....	248
批次构建 .....	251
使用构建 .....	251
运行构建 () .....	252

查看构建详细信息 .....	261
查看构建 ID 的列表 () .....	263
查看构建项目的构建 ID 列表 () .....	266
停止构建 .....	268
停止批量构建 .....	269
重试构建 .....	270
会话管理器 .....	271
删除构建 .....	274
使用测试报告 .....	276
创建测试报告 .....	276
使用报告组 .....	277
创建报告组 .....	277
更新报告组 .....	280
指定测试文件 .....	282
指定测试命令 .....	283
报告组命名 .....	283
标记报告组 .....	283
使用共享报告组 .....	287
使用报告 .....	291
使用测试报告权限 .....	291
为测试报告创建角色 .....	292
测试报告操作的权限 .....	293
测试报告权限示例 .....	293
查看测试报告 .....	294
查看构建的测试报告 .....	294
查看报告组的测试报告 .....	294
查看您的 AWS 账户中的测试报告 .....	294
使用测试框架测试报告 .....	295
使用 Jasmine 进行报告 .....	295
使用 Jest 进行报告 .....	296
使用 pytest 进行报告 .....	297
使用 RSpec 进行报告 .....	298
代码覆盖报告 .....	298
创建代码覆盖报告 .....	298
日志记录和监控 .....	299
使用 AWS CloudTrail 记录 AWS CodeBuild API 调用 .....	300
CloudTrail 中的 AWS CodeBuild 信息 .....	300
了解 AWS CodeBuild 日志文件条目 .....	301
监控 AWS CodeBuild .....	302
CloudWatch 指标 .....	303
CloudWatch 资源利用率指标 .....	304
CloudWatch 维度 .....	306
CloudWatch 警报。 .....	306
CodeBuild 指标 .....	306
CodeBuild 资源利用率指标 .....	311
CodeBuild 警报 .....	316
安全性 .....	323
数据保护 .....	323
数据加密 .....	324
密钥管理 .....	324
流量隐私 .....	324
Identity and Access Management .....	325
Authentication .....	325
访问控制 .....	326
有关管理访问的概述 .....	326
使用基于身份的策略 .....	329

AWS CodeBuild 权限参考 .....	344
使用标签控制对 AWS CodeBuild 资源的访问 .....	348
在控制台中查看资源 .....	351
合规性验证 .....	351
恢复功能 .....	352
基础设施安全性 .....	352
高级主题 .....	353
高级设置 .....	353
向 IAM 组或 IAM 用户添加 CodeBuild 访问权限 .....	353
创建 CodeBuild 服务角色 .....	357
为 CodeBuild 创建和配置 AWS KMS CMK .....	361
安装和配置 AWS CLI .....	363
命令行参考 .....	363
AWS 开发工具包和工具参考 .....	364
适用于 AWS CodeBuild 的受支持的 AWS 开发工具包和工具 .....	364
指定终端节点 .....	365
指定 AWS CodeBuild 终端节点 (AWS CLI) .....	365
指定 AWS CodeBuild 终端节点 (AWS 开发工具包) .....	365
直接运行 CodeBuild .....	367
Prerequisites .....	367
直接运行 AWS CodeBuild .....	367
将 CodePipeline 与 CodeBuild 结合使用 .....	367
Prerequisites .....	368
创建使用 CodeBuild 的管道 (CodePipeline 控制台) .....	369
创建使用 CodeBuild 的管道 (AWS CLI) .....	371
将 CodeBuild 构建操作添加到管道 (CodePipeline 控制台) .....	374
将 CodeBuild 测试操作添加到管道 (CodePipeline 控制台) .....	378
将 CodeBuild 与 Jenkins 结合使用 .....	381
设置 Jenkins .....	381
安装插件 .....	381
使用插件 .....	381
将 CodeBuild 与 codecov 结合使用 .....	382
将 codecov 集成到构建项目中 .....	383
无服务器应用 .....	385
相关资源 .....	53
故障排除 .....	387
来自错误存储库的 Apache Maven 构建参考构件 .....	387
默认情况下，以根用户身份运行构建命令 .....	388
当文件名为非美国时，内部版本可能会失败。英文字符 .....	389
当从 Amazon EC2 Parameter Store 获取参数时，构建可能失败 .....	389
无法在 CodeBuild 控制台中访问分支筛选条件 .....	390
无法查看构建是成功还是失败 .....	390
无法找到并选择 Windows Server Core 2016 平台的基本映像 .....	390
构建规范文件中的前期命令无法被后续命令识别 .....	390
错误 尝试下载缓存时“访问被拒绝” .....	391
错误 使用自定义构建图像时“BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE” .....	391
错误 “在完成构建之前发现构建容器已死。构建容器已死因为内存不足，或者不支持 Docker 镜像。 ErrorCode 500** .....	392
错误 运行内部版本时“无法连接到 Docker 守护程序” .....	392
错误：“CodeBuild 运行内部版本时遇到问题” .....	392
错误：“CodeBuild 在创建或更新构建项目时，无权执行:ss:AssumeRole” .....	393
错误 “调用GetBucketAcl时出错：bucketowner已更改，或者服务角色不再具有调用s3的权限:GetBucketAcl” .....	393
错误 “上传工作失败：运行内部版本时arn”无效 .....	393
错误 “Git克隆失败：无法访问 'your-repository-URL'：SSL证书问题：自签名证书 .....	394
错误 运行内部版本时，必须使用指定的端点解决您尝试访问的存储桶 .....	394

---

错误 "策略的默认版本不是通过增强零点击角色创建创建的,或者不是通过增强零点击角色创建创建的最新版本。"	394
错误 "此构建映像需要选择至少一个运行时版本。"	395
错误 QUEUED 当构建队列中的构建失败时,显示INSUFFICIENT_SUBNET"	395
错误 "无法下载缓存: 请求错误: 发送请求失败,原因:x509: 无法加载系统根目录,未提供根目录"	396
错误 "无法从S3下载证书。AccessDenied"	396
错误 "无法找到凭证"	396
在代理服务器中运行 CodeBuild 时出现 RequestError 超时错误	397
bourne shell (sh) 必须存在于构建映像中	398
警告 "跳过运行时的安装。运行构建时,此构建图像不支持运行时版本选择"	398
错误 BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE构建	398
错误 "无法验证JobWorker身份"	399
配额	400
构建项目	400
构建	400
报告	401
Tags	401
适用于 Windows 的 AWS CodeBuild 的第三方说明	402
1) 基本 Docker 映像—windowsservercore	402
2) 基于 Windows 的 Docker 映像—choco	403
3) 基于 Windows 的 Docker 映像—git - 版本 2.16.2	403
4) 基于 Windows 的 Docker 映像—microsoft-build-tools - 版本 15.0.26320.2	403
5) 基于 Windows 的 Docker 映像—nuget.commandline - 版本 4.5.1	405
7) 基于 Windows 的 Docker 映像—netfx-4.6.2-devpack	405
8) 基于 Windows 的 Docker 映像—visualfsharp tools , v 4.0	406
9) 基于 Windows 的 Docker 映像—netfx-pcl-reference-assemblies-4.6	407
10) 基于 Windows 的 Docker 映像—visualcppbuildtools v 14.0.25420.1	408
11) 基于 Windows 的 Docker 映像—microsoft-windows-netfx3-on-demand-package.cab	410
12) 基于 Windows 的 Docker 映像—dotnet-sdk	411
文档历史记录	412
早期更新	416
AWS 词汇表	421

# 什么是 AWS CodeBuild？

AWS CodeBuild 是一项在云中完全托管的构建服务。CodeBuild 可编译源代码，运行单元测试，并构建可供部署的项目。CodeBuild 使您无需预置、管理和扩展自己的构建服务器。它提供了适用于常用编程语言的预先打包的构建环境以及 Apache Maven 和 Gradle 等构建工具。您还可以在 CodeBuild 中自定义构建环境以使用您自己的构建工具。CodeBuild 可自动扩展以满足峰值构建请求。

CodeBuild 具有以下优势：

- 完全托管 – 利用 CodeBuild，您无需设置、修补、更新和管理自己的构建服务器。
- 按需 – CodeBuild 可以按需扩展以满足您的构建需求。您只需为使用的构建分钟数付费。
- 开箱即用 – CodeBuild 提供了适用于最热门编程语言的预配置构建环境。您只需指向您的构建脚本以开始首次构建即可。

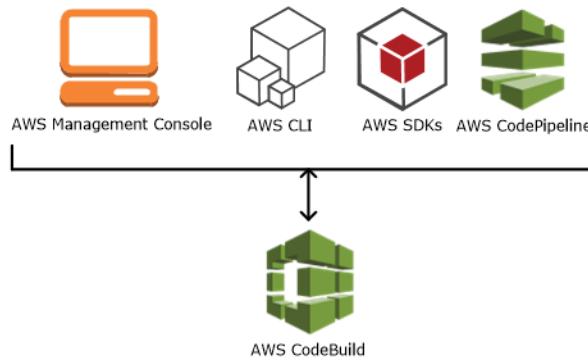
有关更多信息，请参阅 [AWS CodeBuild](#)。

## 主题

- [如何运行 CodeBuild \(p. 1\)](#)
- [CodeBuild 定价 \(p. 2\)](#)
- [如何开始使用 CodeBuild ? \(p. 2\)](#)
- [AWS CodeBuild 概念 \(p. 2\)](#)

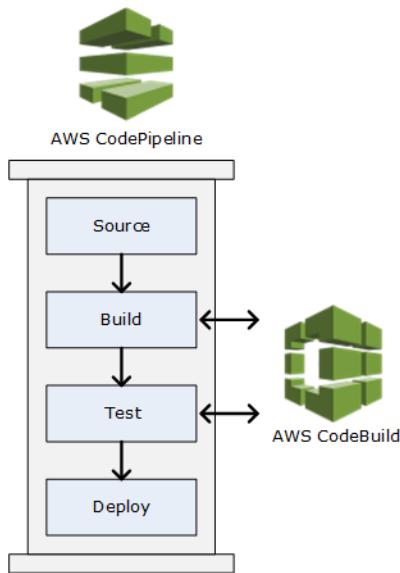
# 如何运行 CodeBuild

您可以使用 AWS CodeBuild 或 AWS CodePipeline 控制台运行 CodeBuild。您也可以使用 AWS Command Line Interface (AWS CLI) 或 AWS 开发工具包来自动运行 CodeBuild。



要使用 CodeBuild 控制台、AWS CLI 和 AWS 开发工具包运行 CodeBuild，请参阅[直接运行 AWS CodeBuild \(p. 367\)](#)。

如下图所示，您可以在 AWS CodePipeline 中将 CodeBuild 作为构建或测试操作添加到管道的构建或测试阶段。AWS CodePipeline 是一种持续交付服务，让您能够为发布您的代码所需的步骤实现模块化、可视化和自动化。其中包括构建您的代码。管道是一个描述发布流程中代码更改情况的工作流程结构。



要使用 CodePipeline 创建管道并添加 CodeBuild 构建或测试操作，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。有关 CodePipeline 的详细信息，请参阅[AWS CodePipeline 用户指南](#)。

CodeBuild 控制台还提供了一种方法，可以快速搜索诸如存储库、构建项目、部署应用程序和管道等资源。选择 Go to resource (转到资源)或按下 / 键，然后键入资源的名称。任何匹配结果都会显示在列表中。搜索不区分大小写。您只能看到您有权查看的资源。有关更多信息，请参阅[在控制台中查看资源 \(p. 351\)](#)。

## CodeBuild 定价

有关信息，请参阅[CodeBuild 定价](#)。

## 如何开始使用 CodeBuild？

我们建议您完成以下步骤：

1. 阅读[概念 \(p. 2\)](#)中的信息以详细了解 CodeBuild。
2. 按照[通过控制台开始使用 \(p. 4\)](#)中的说明在示例方案中试验 CodeBuild。
3. 按照[计划构建 \(p. 135\)](#)中的说明在自己的方案中使用 CodeBuild。

## AWS CodeBuild 概念

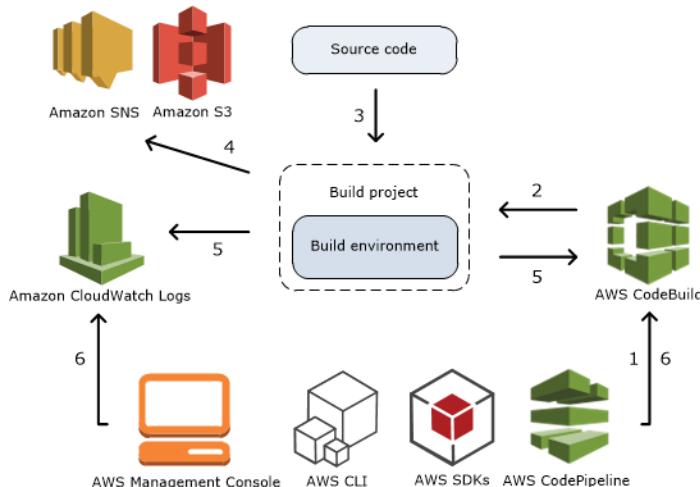
以下概念对了解 CodeBuild 的工作原理来说很重要。

### 主题

- [CodeBuild 的工作原理 \(p. 3\)](#)
- [后续步骤 \(p. 3\)](#)

## CodeBuild 的工作原理

下图显示了当您借助 CodeBuild 运行构建时会发生的情况：



1. 作为输入，您必须为 CodeBuild 提供构建项目。构建项目 包括有关如何运行构建的信息，其中包括源代码获取位置、要使用的构建环境、要运行的构建命令和构建输出的存储位置。生成环境 是由操作系统、编程语言运行时和 CodeBuild 用于运行生成任务的工具组成的。有关更多信息，请参阅：
  - [创建构建项目 \(\) \(p. 189\)](#)
  - [构建环境参考 \(p. 159\)](#)
2. CodeBuild 使用构建项目创建构建环境。
3. CodeBuild 将源代码下载到构建环境中，然后使用构建项目中定义的或源代码中直接包含的构建规范 (build spec)。构建规范 是构建命令和相关设置的集合，采用 YAML 格式，由 CodeBuild 用来运行构建任务。有关更多信息，请参阅[构建规范参考 \(p. 136\)](#)。
4. 如果存在任何构建输出，则该构建环境会将其输出上传到 S3 存储桶。构建环境也可以执行您在构建规范中指定的任务（例如，将构建通知发送到 Amazon SNS 主题）。有关示例，请参阅[构建通知示例 \(p. 80\)](#)。
5. 在构建运行时，构建环境会将信息发送到 CodeBuild 和 Amazon CloudWatch Logs。
6. 在构建运行时，您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包从 CodeBuild 中获取汇总的构建信息，并从 Amazon CloudWatch Logs 中获取详细的构建信息。如果您使用 AWS CodePipeline 运行构建，则可以从 CodePipeline 获取有限的构建信息。

## 后续步骤

现在，您已了解有关 AWS CodeBuild 的更多信息，建议您执行以下后续步骤：

1. 按照[通过控制台开始使用 \(p. 4\)](#)中的说明在示例方案中试验 CodeBuild。
2. 按照[计划构建 \(p. 135\)](#)中的说明在自己的方案中使用 CodeBuild。

# CodeBuild 入门

在以下教程中，您可以使用 AWS CodeBuild 将示例源代码输入文件的集合构建到源代码的可部署版本中。

两个教程具有相同的输入和结果，但一个教程使用的是 AWS CodeBuild 控制台，另一个教程使用的是 AWS CLI。

## Important

建议您不要使用 AWS 根账户来完成本教程。

## 通过控制台开始使用 AWS CodeBuild

在本教程中，您将使用 AWS CodeBuild 将一系列示例源代码输入文件（构建输入项目 或 构建输入）构建为一个可部署的源代码版本（构建输出项目 或 构建输出）。具体来说，您将指示 CodeBuild 使用 Apache Maven（一种常用的生成工具）将一组 Java 类文件生成为 Java 存档 (JAR) 文件。您无需熟悉 Apache Maven 或 Java 即可完成本教程。

您可以通过如下方式使用 CodeBuild：CodeBuild 控制台、AWS CodePipeline、AWS CLI 或 AWS 开发工具包。本教程演示如何使用 CodeBuild 控制台。有关使用 CodePipeline 的信息，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。有关使用 AWS 开发工具包的信息，请参阅[直接运行 CodeBuild \(p. 367\)](#)。

## Important

本教程中的步骤要求您创建可能会对您的 AWS 账户产生费用的资源（例如，S3 存储桶）。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS 和 CloudWatch Logs 相关的操作收取的费用。有关更多信息，请参阅[AWS CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)和[Amazon CloudWatch 定价](#)。

## 步骤

- 步骤 1：创建两个 S3 存储桶 (p. 4)
- 步骤 2：创建源代码 (p. 5)
- 步骤 3：创建构建规范文件 (p. 7)
- 步骤 4：上传源代码和构建规范文件 (p. 8)
- 步骤 5：创建构建项目 (p. 9)
- 步骤 6：运行构建 (p. 10)
- 步骤 7：查看汇总的构建信息 (p. 11)
- 步骤 8：查看详细的构建信息 (p. 11)
- 步骤 9：获取构建输出构件 (p. 12)
- 步骤 10：删除 S3 输入存储桶 (p. 12)
- 总结 (p. 13)

## 步骤 1：创建两个 S3 存储桶

(一部分：[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#) )

尽管您可以为本教程使用单个存储桶，但使用两个存储桶使您可以更容易地查看构建输入的来源以及构建输出的去向。

- 其中一个存储桶（输入存储桶）用于存储构建输入。在本教程中，此输入存储桶的名称为 `codebuild-region-ID-account-ID-input-bucket`，其中 `region-ID` 是存储桶的 AWS 区域，`account-ID` 是您的 AWS 账户 ID。
- 另一个存储桶（输出存储桶）用于存储构建输出。在本教程中，此输出存储桶的名称为 `codebuild-region-ID-account-ID-output-bucket`。

如果您为这些存储桶选择了不同的名称，请务必在本教程中使用它们。

这两个存储桶必须与您的生成项目处在同一个 AWS 区域中。例如，如果您指示 CodeBuild 在美国东部（俄亥俄州）区域中运行构建，这些存储桶也必须位于美国东部（俄亥俄州）区域中。

有关更多信息，请参阅 Amazon Simple Storage Service 用户指南 中的 [创建存储桶](#)。

#### Note

尽管 CodeBuild 也支持存储在 CodeCommit、GitHub 和 Bitbucket 存储库中的构建输入，但本教程不说明如何使用它们。有关更多信息，请参阅 [计划构建 \(p. 135\)](#)。

## 下一步

[步骤 2：创建源代码 \(p. 5\)](#)

## 步骤 2：创建源代码

( 上一步：[步骤 1：创建两个 S3 存储桶 \(p. 4\)](#) )

在此步骤中，您将创建需要 CodeBuild 生成到输出存储桶的源代码。此源代码包含两个 Java 类文件和一个 Apache Maven 项目对象模型 (POM) 文件。

1. 在您的本地计算机或实例上的空目录中，创建此目录结构。

```
(root directory name)
`-- src
    |-- main
    |   |-- java
    |   `-- test
    |       `-- java
```

2. 使用您选择的文本编辑器创建此文件，将其命名为 `MessageUtil.java`，然后保存在 `src/main/java` 目录中。

```
public class MessageUtil {
    private String message;

    public MessageUtil(String message) {
        this.message = message;
    }

    public String printMessage() {
        System.out.println(message);
        return message;
    }

    public String salutationMessage() {
        message = "Hi!" + message;
```

```
        System.out.println(message);
        return message;
    }
}
```

创建此类文件是用来输出传入的字符串。MessageUtil 构造函数用于设置字符串。printMessage 方法用于创建输出。salutationMessage 方法用于输出 Hi! 后跟字符串。

3. 创建此文件，将其命名为 TestMessageUtil.java，然后将它保存在 /src/test/java 目录中。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        System.out.println("Inside testPrintMessage()");
        assertEquals(message, messageUtil.printMessage());
    }

    @Test
    public void testSalutationMessage() {
        System.out.println("Inside testSalutationMessage()");
        message = "Hi!" + "Robert";
        assertEquals(message, messageUtil.salutationMessage());
    }
}
```

此类文件用于将 MessageUtil 类中的 message 变量设置为 Robert。然后，通过检查输出中是否出现字符串 Robert 和 Hi!Robert 来测试是否成功设置 message 变量。

4. 创建此文件，将其命名为 pom.xml，然后保存在根 (顶级) 目录中。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>messageUtil</artifactId>
    <version>1.0.</version>
    <packaging>jar</packaging>
    <name>Message Utility Java Sample App</name>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
            </plugin>
        </plugins>
    </build>

```

```
</build>  
</project>
```

Apache Maven 使用此文件中的指令将 `MessageUtil.java` 和 `TestMessageUtil.java` 文件转换为名为 `messageUtil-1.0.jar` 的文件，然后运行指定测试。

此时，您的目录结构应如下所示。

```
(root directory name)  
|-- pom.xml  
`-- src  
    |-- main  
    |   |-- java  
    |   |   `-- MessageUtil.java  
    |-- test  
    |   |-- java  
    |   |   `-- TestMessageUtil.java
```

## 下一步

[步骤 3：创建构建规范文件 \(p. 7\)](#)

# 步骤 3：创建构建规范文件

( 上一步：[步骤 2：创建源代码 \(p. 5\)](#)  )

在此步骤中，您将创建一个构建规范文件。构建规范是构建命令和相关设置的集合，采用 YAML 格式，由 CodeBuild 用来运行构建任务。如果没有生成规范，CodeBuild 就无法将您的生成输入成功转换为生成输出，也无法在生成环境中找到生成输出项目以便上传到输出存储桶中。

创建此文件，将其命名为 `buildspec.yml`，然后保存在根 (顶级) 目录中。

```
version: 0.2  
  
phases:  
  install:  
    runtime-versions:  
      java: corretto11  
  pre_build:  
    commands:  
      - echo Nothing to do in the pre_build phase...  
  build:  
    commands:  
      - echo Build started on `date`  
      - mvn install  
  post_build:  
    commands:  
      - echo Build completed on `date`  
artifacts:  
  files:  
    - target/messageUtil-1.0.jar
```

### Important

因为生成规范声明必须为有效的 YAML，所以生成规范声明中的间隔至关重要。如果生成规范声明中的空格数与此不匹配，则构建可能会立即失败。您可以使用 YAML 验证程序测试生成规范声明是否为有效的 YAML。

#### Note

您可以在创建构建项目时单独声明构建命令，而不是将构建规范文件包含在源代码中。如果您需要使用其他构建命令来构建源代码，而不是每次更新源代码存储库，这个方法很有用。有关更多信息，请参阅[构建规范语法 \(p. 137\)](#)。

在此生成规范声明中：

- `version` 表示正在使用的生成规范标准的版本。此生成规范声明使用最新版本 0.2。
- `phases` 表示您可以指示 CodeBuild 运行命令的生成阶段。这些构建阶段包括 `install`、`pre_build`、`build` 和 `post_build`。您无法更改这些生成阶段名称的拼写，也无法创建更多生成阶段名称。

本示例中，在 `build` 阶段，CodeBuild 运行 `mvn install` 命令。此命令指示 Apache Maven 编译和测试 Java 类文件，然后将编译完的文件打包为构建输出项目。出于完整性考虑，本示例的每个构建阶段中都放了几条 `echo` 命令。您稍后查看本教程中详细的构建信息时，这些 `echo` 命令的输出可以帮助您更好地理解 CodeBuild 运行命令的方式以及顺序。（尽管本示例中包含了所有构建阶段，但如果您的不打算在某个构建阶段运行任何命令，则无需包含该构建阶段。）对于包含的每个生成阶段，CodeBuild 将按照列出的顺序，从头到尾运行每个指定命令（一次运行一个命令）。

- `artifacts` 表示 CodeBuild 上传到输出存储桶的一组生成输出项目。`files` 表示要包含在生成输出中的文件。CodeBuild 会上传在生成环境的 `target` 相对目录中找到的单个 `messageUtil-1.0.jar`。文件 `messageUtil-1.0.jar` 和目录 `target` 只是根据本示例中 Apache Maven 创建和存储构建输出项目的方式来命名的。在您自己的生成项目中，这些文件和目录名称会有所不同。

有关更多信息，请参阅[构建规范参考 \(p. 136\)](#)。

此时，您的目录结构应如下所示。

```
(root directory name)
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |   |-- java
    |   |   `-- MessageUtil.java
    |   `-- test
    |       |-- java
    |       |   `-- TestMessageUtil.java
```

## 下一步

[步骤 4：上传源代码和构建规范文件 \(p. 8\)](#)

## 步骤 4：上传源代码和构建规范文件

( 上一步：[步骤 3：创建构建规范文件 \(p. 7\)](#) )

在此步骤中，您将源代码和构建规范文件添加到输入存储桶中。

使用操作系统的 ZIP 实用工具，创建一个名为 `MessageUtil.zip` 的文件，其中包含 `MessageUtil.java`、`TestMessageUtil.java`、`pom.xml` 和 `buildspec.yml`。

`MessageUtil.zip` 文件的目录结构必须如下所示。

```
MessageUtil.zip
|-- pom.xml
|-- buildspec.yml
`-- src
```

```
|-- main
|   '-- java
|     '-- MessageUtil.java
`-- test
    '-- java
      '-- TestMessageUtil.java
```

#### Important

请不要包含 (*root directory name*) 目录，而只包含 (*root directory name*) 目录中的目录和文件。

将 MessageUtil.zip 文件上传至名为 codebuild-*region-ID-account-ID*-input-bucket 的输入存储桶中。

#### Important

对于 CodeCommit、GitHub 和 Bitbucket 存储库，按照惯例，您必须在每个存储库的根（顶级）位置存储一个名为 buildspec.yml 的构建规范文件，或者将生成规范声明作为构建项目定义的一部分包含。请勿创建包含存储库源代码和构建规范文件的 ZIP 文件。

仅对于存储在 S3 存储桶中的构建输入，您必须创建一个包含源代码的 ZIP 文件和一个（按照惯例）位于根（顶级）位置的名为 buildspec.yml 的构建规范文件，或者将构建规范声明作为构建项目定义的一部分包含。

如果您要为构建规范文件使用其他名称，或者要在根位置之外的位置引用生成规范，则可指定生成规范覆盖作为构建项目定义的一部分。有关更多信息，请参阅 [构建规范文件名称和存储位置 \(p. 136\)](#)。

## 下一步

[步骤 5：创建构建项目 \(p. 9\)](#)

# 步骤 5：创建构建项目

( 上一步：[步骤 4：上传源代码和构建规范文件 \(p. 8\)](#) )

在此步骤中，您将创建一个构建项目，AWS CodeBuild 将使用它来运行构建项目。构建项目 包括有关如何运行构建的信息，其中包括源代码获取位置、要使用的构建环境、要运行的构建命令和构建输出的存储位置。生成环境 是由操作系统、编程语言运行时和 CodeBuild 用于运行生成任务的工具组成的。构建环境以 Docker 映像的形式表示。有关更多信息，请参阅 Docker 文档网站上的 [Docker 概述](#)。

对于此生成环境，您需要指示 CodeBuild 使用包含 Java 开发工具包 (JDK) 和 Apache Maven 的 Docker 映像。

### 创建构建项目

1. 通过以下网址登录 AWS 管理控制台并打开 AWS CodeBuild 控制台：<https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 使用 AWS 区域选择器选择支持 CodeBuild 的 AWS 区域。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [AWS CodeBuild 终端节点和配额](#)。
3. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
4. 在 Create build project (创建生成项目) 页面上的 Project Configuration (项目配置) 中，对于 Project name (项目名称)，输入此生成项目的名称（在此示例中为 codebuild-demo-project）。构建项目名称在您的各个 AWS 账户内必须是唯一的。如果您使用其他名称，请确保在本教程中通篇使用它。

### Note

在 Create build project (创建构建项目) 页面上，您可能会看到类似于以下内容的错误消息：You are not authorized to perform this operation (您没有权限执行此操作)。最可能的原

因是，您用来登录 AWS 管理控制台的 IAM 用户身份无权创建构建项目。要修复此问题，请从 AWS 管理控制台注销，然后使用属于以下任一 IAM 实体的凭证重新登录：

- AWS 账户中的 IAM 管理员用户。有关更多信息，请参阅 IAM 用户指南 中的[创建您的第一个 IAM 管理员用户和组](#)。
- 您的 IAM 账户中的 AWS 用户，该 IAM 用户或 IAM 用户所在的 IAM 组挂载了 `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess` 和 `IAMFullAccess` 托管策略。如果您的 AWS 账户中没有 IAM 用户或组具有这些权限，并且您无法将这些权限添加到您的 IAM 用户或组，请与 AWS 账户管理员联系以寻求帮助。有关更多信息，请参阅适用于 AWS CodeBuild 的 AWS 托管 (预定义) 策略 (p. 330)。

这两个选项都可让您获得创建构建项目所需的管理员权限，以便您能够完成本教程。建议您始终使用完成任务所需的最低权限。有关更多信息，请参阅[AWS CodeBuild 权限参考 \(p. 344\)](#)。

5. 在 Source (源) 中，对于 Source provider (源提供商)，选择 Amazon S3。
6. 对于 Bucket (存储桶)，选择 `codebuild-region-ID-account-ID-input-bucket`。
7. 对于 S3 object key (S3 对象键)，输入 `MessageUtil.zip`。
8. 在 Environment (环境) 中，对于 Environment image (环境映像)，让 Managed image (托管映像) 处于选中状态。
9. 对于 Operating system (操作系统)，选择 Amazon Linux 2。
10. 对于 Runtime(s) (运行时)，选择 Standard (标准)。
11. 对于 Image (映像)，选择 `aws/codebuild/amazonlinux2-x86_64-standard:3.0`。
12. 在 Service role (服务角色) 中，将 New service role (新建服务角色) 保持选中状态，并将 Role name (角色名称) 保持不变。
13. 对于 Buildspec (生成规范)，将 Use a buildspec file (使用构建规范文件) 保留为选中状态。
14. 在 Artifacts (构件) 中，对于 Type (类型)，选择 Amazon S3。
15. 对于 Bucket name (存储桶名称)，选择 `codebuild-region-ID-account-ID-output-bucket`。
16. 将 Name (名称) 和 Path (路径) 留空。
17. 选择 Create build project (创建构建项目)。

## 下一步

[步骤 6：运行构建 \(p. 10\)](#)

# 步骤 6：运行构建

( 上一步：[步骤 5：创建构建项目 \(p. 9\)](#) )

在此步骤中，您将指示 AWS CodeBuild 使用构建项目中的设置来运行构建。

## 运行构建

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 在导航窗格中，选择 Build projects。
3. 在构建项目列表中，选择 `codebuild-demo-project`，然后选择 Start build (启动构建)。
4. 在 Start build (启动构建) 页面上，选择 Start build (启动构建)。

## 下一步

[步骤 7：查看汇总的构建信息 \(p. 11\)](#)

## 步骤 7：查看汇总的构建信息

( 上一步：[步骤 6：运行构建 \(p. 10\)](#) )

在此步骤中，您将查看有关生成状态的汇总信息。

### 查看汇总的构建信息

1. 如果未显示 codebuild-demo-project:**build-ID** 页面，请在导航栏中，选择 Build history (构建历史记录)。接下来，在生成项目列表中，对于 Project (项目)，选择 codebuild-demo-project 的 Build run (生成运行) 链接。应该只有一个匹配的链接。（如果您之前已完成本教程，请在 Completed (已完成) 列中选择具有最新值的链接。）
2. 在构建详细信息页面上的 Phase details (阶段详细信息) 中，应显示以下构建阶段，并且 Status (状态) 列中的值为 Succeeded (成功)：
  - SUBMITTED
  - QUEUED
  - PROVISIONING
  - DOWNLOAD\_SOURCE
  - INSTALL
  - PRE\_BUILD
  - BUILD
  - POST\_BUILD
  - UPLOAD\_ARTIFACTS
  - FINALIZING
  - COMPLETED

在 Build Status (构建状态) 中，应显示 Succeeded (已成功)。

如果您看到的是 In Progress (正在进行)，请选择刷新按钮。

3. 在每个构建阶段的旁边，Duration (持续时间) 值表示构建阶段持续的时间。End time (结束时间) 值表示生成阶段的结束时间。

### 下一步

[步骤 8：查看详细的构建信息 \(p. 11\)](#)

## 步骤 8：查看详细的构建信息

( 上一步：[步骤 7：查看汇总的构建信息 \(p. 11\)](#) )

在此步骤中，您将查看有关 CloudWatch Logs 中生成项目的详细信息。

#### Note

为了保护敏感信息，CodeBuild 日志中隐藏以下内容：

- AWS 访问密钥 ID。有关更多信息，请参阅 AWS Identity and Access Management 用户指南 中的 [管理 IAM 用户的访问密钥](#)。
- 使用参数存储指定的字符串。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。

- 使用 AWS Secrets Manager 指定的字符串。有关更多信息，请参阅 [密钥管理 \(p. 324\)](#)。

### 查看详细的构建信息

1. 上一步完成后，构建详细页面继续显示，Build logs 中显示了构建日志的最后 10,000 行内容。要在 CloudWatch Logs 中查看完整构建日志，请选择 [View entire log \(查看完整日志\)](#) 链接。
2. 在 CloudWatch Logs 日志流中，您可以浏览日志事件。默认情况下，只显示最近的一组日志事件。要查看以前的日志事件，请滚动到列表开头。
3. 在本教程中，大多数日志事件包含的是关于 CodeBuild 下载构建相关文件并将其安装到构建环境中的详细信息，您可能并不关心这些信息。您可以使用 Filter events 框来减少显示的信息。例如，如果您在 Filter events (筛选事件) 中输入 "[INFO]"，则只显示包含 [INFO] 的事件。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的[筛选条件和模式语法](#)。

## 下一步

[步骤 9：获取构建输出构件 \(p. 12\)](#)

## 步骤 9：获取构建输出构件

( 上一步：[步骤 8：查看详细的构建信息 \(p. 11\)](#) )

在此步骤中，您会得到 CodeBuild 生成并上传到输出存储桶的 messageUtil-1.0.jar 文件。

您可以使用 CodeBuild 控制台或 Amazon S3 控制台完成此步骤。

### 获取构建输出项目（AWS CodeBuild 控制台）

1. 与 CodeBuild 控制台仍然打开，并且在上一步中仍显示了构建详细信息页面，选择 构建详情 选项卡并向下滚动 伪影 第节。

#### Note

(如果未显示构建详细信息页面，请在导航栏中选择 Build history，然后选择 Build run 链接。)

2. 链接到 Amazon S3 文件夹位于 伪造上传位置。此链接打开文件夹 Amazon S3 您在哪里找到 messageUtil-1.0.jar 构建输出工件文件。

### 获取构建输出项目（Amazon S3 控制台）

1. 通过以下网址打开 Amazon S3 控制台：[https://console.aws.amazon.com/s3/。](https://console.aws.amazon.com/s3/)
2. 打开 codebuild-*region-ID-account-ID-output-bucket*。
3. 打开 codebuild-demo-project 文件夹。
4. 打开 target 文件夹，您可以在此处找到 messageUtil-1.0.jar 构建输出项目文件。

## 下一步

[步骤 10：删除 S3 输入存储桶 \(p. 12\)](#)

## 步骤 10：删除 S3 输入存储桶

( 上一步：[步骤 9：获取构建输出构件 \(p. 12\)](#) )

为防止您的 AWS 账户持续产生费用，您可以删除本教程中使用的输入存储桶。有关说明，请参阅 Amazon Simple Storage Service 开发人员指南 中的[删除或清空存储桶](#)。

如果您使用 IAM 用户或管理员 IAM 用户删除此存储桶，则该用户必须具有更多访问权限。将标记 (### BEGIN ADDING STATEMENT HERE ### 和 ### END ADDING STATEMENTS HERE ###) 之间的下列语句添加到用户的现有访问策略中。

此语句中的省略号 (...) 旨在力求简洁。请勿删除现有访问策略中的任何语句。请勿在策略中输入这些省略号。

```
{  
    "Version": "2012-10-17",  
    "Id": "...",  
    "Statement": [  
        ### BEGIN ADDING STATEMENT HERE ###  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:DeleteBucket",  
                "s3:DeleteObject"  
            ],  
            "Resource": "*"  
        }  
        ### END ADDING STATEMENT HERE ###  
    ]  
}
```

## 下一步

[总结 \(p. 13\)](#)

## 总结

在本教程中，您使用 AWS CodeBuild 将一组 Java 类文件构建为一个 JAR 文件。然后查看了构建的结果。

您现在可以尝试在自己的场景中使用 CodeBuild。按照[计划构建 \(p. 135\)](#)中的说明进行操作。如果您觉得自己还没准备好，可以尝试生成一些示例。有关更多信息，请参阅[示例 \(p. 27\)](#)。

## 通过 AWS CLI 开始使用 AWS CodeBuild

在本教程中，您使用 AWS CodeBuild 将一系列示例源代码输入文件（称为构建输入项目 或 构建输入）构建为一个可部署的源代码版本（称为构建输出项目 或 构建输出）。具体来说，您将指示 CodeBuild 使用 Apache Maven（一种常用的生成工具）将一组 Java 类文件生成为 Java 存档 (JAR) 文件。您无需熟悉 Apache Maven 或 Java 即可完成本教程。

您可以通过如下方式使用 CodeBuild：CodeBuild 控制台、AWS CodePipeline、AWS CLI 或 AWS 开发工具包。本教程演示如何将 CodeBuild 与 AWS CLI 结合使用。有关使用 CodePipeline 的信息，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。有关使用 AWS 开发工具包的信息，请参阅[直接运行 CodeBuild \(p. 367\)](#)。

### Important

本教程中的步骤要求您创建可能会对您的 AWS 账户产生费用的资源（例如，S3 存储桶）。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS 和 CloudWatch Logs 相关的操作收取的费用。有关更多信息，请参阅[CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)和[Amazon CloudWatch 定价](#)。

## Steps

- 步骤 1. 创建两个S3存储桶 (p. 14)

- 步骤 2. 创建源代码 (p. 14)
- 步骤 3 创建BuildSpec文件 (p. 16)
- 步骤 4. 上传源代码和BuildSpec文件 (p. 18)
- 步骤 5. 创建构建项目 (p. 18)
- 步骤 6. 运行构建。 (p. 21)
- 步骤 7. 查看汇总的构建信息 (p. 22)
- 步骤 8. 查看详细的构建信息 (p. 24)
- 步骤 9 获取构建输出伪影 (p. 25)
- 步骤 10 删除S3输入桶 (p. 26)
- 总结 (p. 26)

## 步骤 1. 创建两个S3存储桶

(一部分：[通过 AWS CLI 开始使用 AWS CodeBuild \(p. 13\)](#) )

尽管您可以为本教程使用单个存储桶，但使用两个存储桶使您可以更容易地查看构建输入的来源以及构建输出的去向。

- 其中一个存储桶（输入存储桶）用于存储构建输入。在本教程中，此输入存储桶的名称为 `codebuild-region-ID-account-ID-input-bucket`，其中 `region-ID` 是存储桶的 AWS 区域，`account-ID` 是您的 AWS 账户 ID。
- 另一个存储桶（输出存储桶）用于存储构建输出。在本教程中，此输出存储桶的名称为 `codebuild-region-ID-account-ID-output-bucket`。

如果您为这些存储桶选择了不同的名称，请务必在本教程中使用它们。

这两个存储桶必须与您的生成项目处在同一个 AWS 区域中。例如，如果您指示 CodeBuild 在美国东部（俄亥俄州）区域中运行构建，这些存储桶也必须位于美国东部（俄亥俄州）区域中。

有关更多信息，请参阅 Amazon Simple Storage Service 用户指南 中的 [创建存储桶](#)。

### Note

尽管 CodeBuild 也支持存储在 CodeCommit、GitHub 和 Bitbucket 存储库中的构建输入，但本教程不说明如何使用它们。有关更多信息，请参阅 [计划构建 \(p. 135\)](#)。

## 下一步

[步骤 2. 创建源代码 \(p. 14\)](#)

## 步骤 2. 创建源代码

(上一步：[步骤 1. 创建两个S3存储桶 \(p. 14\)](#) )

在此步骤中，您将创建需要 CodeBuild 生成到输出存储桶的源代码。此源代码包含两个 Java 类文件和一个 Apache Maven 项目对象模型 (POM) 文件。

1. 在您的本地计算机或实例上的空目录中，创建此目录结构。

```
(root directory name)
  '-- src
    |-- main
```

```
|-  `-- java
`-- test
    '-- java
```

2. 使用您选择的文本编辑器创建此文件，将其命名为 `MessageUtil.java`，然后保存在 `src/main/java` 目录中。

```
public class MessageUtil {
    private String message;

    public MessageUtil(String message) {
        this.message = message;
    }

    public String printMessage() {
        System.out.println(message);
        return message;
    }

    public String salutationMessage() {
        message = "Hi!" + message;
        System.out.println(message);
        return message;
    }
}
```

创建此类文件是用来输出传入的字符串。`MessageUtil` 构造函数用于设置字符串。`printMessage` 方法用于创建输出。`salutationMessage` 方法用于输出 `Hi!` 后跟字符串。

3. 创建此文件，将其命名为 `TestMessageUtil.java`，然后将它保存在 `/src/test/java` 目录中。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        System.out.println("Inside testPrintMessage()");
        assertEquals(message, messageUtil.printMessage());
    }

    @Test
    public void testSalutationMessage() {
        System.out.println("Inside testSalutationMessage()");
        message = "Hi!" + "Robert";
        assertEquals(message, messageUtil.salutationMessage());
    }
}
```

此类文件用于将 `MessageUtil` 类中的 `message` 变量设置为 `Robert`。然后，通过检查输出中是否出现字符串 `Robert` 和 `Hi!Robert` 来测试是否成功设置 `message` 变量。

4. 创建此文件，将其命名为 `pom.xml`，然后保存在根 (顶级) 目录中。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
    maven-v4_0_0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</groupId>
<artifactId>messageUtil</artifactId>
<version>1.0</version>
<packaging>jar</packaging>
<name>Message Utility Java Sample App</name>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
        </plugin>
    </plugins>
</build>
</project>
```

Apache Maven 使用此文件中的指令将 `MessageUtil.java` 和 `TestMessageUtil.java` 文件转换为名为 `messageUtil-1.0.jar` 的文件，然后运行指定测试。

此时，您的目录结构应如下所示。

```
(root directory name)
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |   `-- MessageUtil.java
    |-- test
    |   |-- java
    |   |   `-- TestMessageUtil.java
```

## 下一步

[步骤 3 创建BuildSpec文件 \(p. 16\)](#)

## 步骤 3 创建BuildSpec文件

( 上一步：[步骤 2. 创建源代码 \(p. 14\)](#)  )

在此步骤中，您将创建一个构建规范文件。构建规范是构建命令和相关设置的集合，采用 YAML 格式，由 CodeBuild 用来运行构建任务。如果没有生成规范，CodeBuild 就无法将您的生成输入成功转换为生成输出，也无法在生成环境中找到生成输出项目以便上传到输出存储桶中。

创建此文件，将其命名为 `buildspec.yml`，然后保存在根 (顶级) 目录中。

```
version: 0.2
phases:
  install:
    runtime-versions:
```

```
java: corretto11
pre_build:
  commands:
    - echo Nothing to do in the pre_build phase...
build:
  commands:
    - echo Build started on `date`
    - mvn install
post_build:
  commands:
    - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

### Important

因为生成规范声明必须为有效的 YAML，所以生成规范声明中的间隔至关重要。如果生成规范声明中的空格数与此不匹配，则构建可能会立即失败。您可以使用 YAML 验证程序测试生成规范声明是否为有效的 YAML。

### Note

您可以在创建构建项目时单独声明构建命令，而不是将构建规范文件包含在源代码中。如果您需要使用其他构建命令来构建源代码，而不是每次更新源代码存储库，这个方法很有用。有关更多信息，请参阅[构建规范语法 \(p. 137\)](#)。

在此生成规范声明中：

- `version` 表示正在使用的生成规范标准的版本。此生成规范声明使用最新版本 0.2。
- `phases` 表示您可以指示 CodeBuild 运行命令的生成阶段。这些构建阶段包括 `install`、`pre_build`、`build` 和 `post_build`。您无法更改这些生成阶段名称的拼写，也无法创建更多生成阶段名称。

本示例中，在 `build` 阶段，CodeBuild 运行 `mvn install` 命令。此命令指示 Apache Maven 编译和测试 Java 类文件，然后将编译完的文件打包为构建输出项目。出于完整性考虑，本示例的每个构建阶段中都放了几条 `echo` 命令。您稍后查看本教程中详细的构建信息时，这些 `echo` 命令的输出可以帮助您更好地理解 CodeBuild 运行命令的方式以及顺序。（尽管本示例中包含了所有构建阶段，但如果我不打算在某个构建阶段运行任何命令，则无需包含该构建阶段。）对于包含的每个生成阶段，CodeBuild 将按照列出的顺序，从头到尾运行每个指定命令（一次运行一个命令）。

- `artifacts` 表示 CodeBuild 上传到输出存储桶的一组生成输出项目。`files` 表示要包含在生成输出中的文件。CodeBuild 会上传在生成环境的 `target` 相对目录中找到的单个 `messageUtil-1.0.jar`。文件 `messageUtil-1.0.jar` 和目录 `target` 只是根据本示例中 Apache Maven 创建和存储构建输出项目的方式来命名的。在您自己的生成项目中，这些文件和目录名称会有所不同。

有关更多信息，请参阅[构建规范参考 \(p. 136\)](#)。

此时，您的目录结构应如下所示。

```
(root directory name)
|-- pom.xml
|-- buildspec.yml
`-- src
  |-- main
  |   |-- java
  |   |   |-- MessageUtil.java
  |   |-- test
  |       |-- java
  |           |-- TestMessageUtil.java
```

## 下一步

[步骤 4. 上传源代码和BuildSpec文件 \(p. 18\)](#)

## 步骤 4. 上传源代码和BuildSpec文件

( 上一步：[步骤 3 创建BuildSpec文件 \(p. 16\)](#) )

在此步骤中，您将源代码和构建规范文件添加到输入存储桶中。

使用操作系统的 ZIP 实用工具，创建一个名为 `MessageUtil.zip` 的文件，其中包含 `MessageUtil.java`、`TestMessageUtil.java`、`pom.xml` 和 `buildspec.yml`。

`MessageUtil.zip` 文件的目录结构必须如下所示。

```
MessageUtil.zip
|-- pom.xml
|-- buildspec.yml
`-- src
    |-- main
    |   |-- java
    |   |   |-- MessageUtil.java
    |   `-- test
    |       |-- java
    |       |   |-- TestMessageUtil.java
```

### Important

请不要包含 (`root directory name`) 目录，而只包含 (`root directory name`) 目录中的目录和文件。

将 `MessageUtil.zip` 文件上传至名为 `codebuild-region-ID-account-ID-input-bucket` 的输入存储桶中。

### Important

对于 CodeCommit、GitHub 和 Bitbucket 存储库，按照惯例，您必须在每个存储库的根（顶级）位置存储一个名为 `buildspec.yml` 的构建规范文件，或者将生成规范声明作为构建项目定义的一部分包含。请勿创建包含存储库源代码和构建规范文件的 ZIP 文件。

仅对于存储在 S3 存储桶中的构建输入，您必须创建一个包含源代码的 ZIP 文件和一个（按照惯例）位于根（顶级）位置的名为 `buildspec.yml` 的构建规范文件，或者将构建规范声明作为构建项目定义的一部分包含。

如果您要为构建规范文件使用其他名称，或者要在根位置之外的位置引用生成规范，则可指定生成规范覆盖作为构建项目定义的一部分。有关更多信息，请参阅 [构建规范文件名称和存储位置 \(p. 136\)](#)。

## 下一步

[步骤 5. 创建构建项目 \(p. 18\)](#)

## 步骤 5. 创建构建项目

( 上一步：[步骤 4. 上传源代码和BuildSpec文件 \(p. 18\)](#) )

在此步骤中，您创建一个构建项目，该项目可以创建一个构建项目，AWS CodeBuild 运行构造。构建项目包括有关如何运行构建的信息，其中包括源代码获取位置、要使用的构建环境、要运行的构建命令和构建输出的存储位置。生成环境是由操作系统、编程语言运行时和 CodeBuild 用于运行生成任务的工具组成的。构建环境以靠泊器图像表示。有关更多信息，请参阅 Docker 文档网站上的 [Docker 概述](#)。

对于此生成环境，您需要指示 CodeBuild 使用包含 Java 开发工具包 (JDK) 和 Apache Maven 的 Docker 映像。

## 创建构建项目

1. 使用 AWS CLI 运行 `create-project` 命令。

```
aws codebuild create-project --generate-cli-skeleton
```

输出中将显示 JSON 格式的数据。将数据复制到已安装 AWS CLI 的本地计算机或实例上某个位置的名为 `create-project.json` 的文件中。如果您选择使用其他文件名，请务必在本教程中使用该名称。

按照以下格式修改所复制的数据，然后保存结果：

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

Replace `serviceIAMRole` 有一个亚马逊资源名称(ARN)的 CodeBuild 服务角色（例如，`arn:aws:iam::account-ID:role/role-name`）。如需创建一个角色，请参阅[创建 CodeBuild 服务角色 \(p. 357\)](#)。

在此数据中：

- `name` 表示此构建项目的必需标识符 (在本示例中为 `codebuild-demo-project`)。构建项目名称在您账户的所有构建项目中必须是唯一的。
- 对于 `source`，`type` 是一个必需值，表示源代码的存储库类型 (在本示例中，`S3` 表示 Amazon S3 存储桶)。
- 对于 `source`，`location` 表示源代码的路径 (在本示例中，为输入存储桶名称后跟 ZIP 文件名称)。
- 对于 `artifacts`，`type` 是一个必需值，表示生成输出项目的存储库类型 (在本示例中，`S3` 表示 Amazon S3 存储桶)。
- 对于 `artifacts`，`location` 表示您先前创建或识别的输出存储桶的名称 (在本示例中为 `codebuild-region-ID-account-ID-output-bucket`)。
- 对于 `environment`，`type` 是一个必需值，表示构建环境的类型 (`LINUX_CONTAINER` 是目前唯一允许的值)。
- 对于 `environment`，`image` 是一个必要值，表示该建立项目使用的靠泊装置图像名称和标签组合，如靠泊装置图像库类型所指定 (在此示例中，`aws/codebuild/standard:4.0` 对应于在 CodeBuild Docker 图像库)。`aws/codebuild/standard` 是靠泊装置图像的名称。`1.0` 是 Docker 镜像的标签。

要查找您可以在自己方案中使用的更多 Docker 映像，请参阅[构建环境参考 \(p. 159\)](#)。

- 对于 `environment`，`computeType` 是一个必需值，表示 CodeBuild 将会使用的计算资源 (在本示例中为 `BUILD_GENERAL1_SMALL`)。版本 `2016-10-06`

Note

原始JSON格式化数据中的其他可用值，例如 `description`, `buildspec`, `auth` (包括 `type` 和 `resource`)，`path`, `namespaceType`, `name` (对于 `artifacts`), `packaging`, `environmentVariables` (包括 `name` 和 `value`)，`timeoutInMinutes`, `encryptionKey`, 和 `tags` (包括 `key` 和 `value`) 为可选。本教程中未使用这些值，因此它们没有在这里显示。有关更多信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#)。)

2. 切换到您刚才保存的文件所在的目录，然后再次运行 `create-project` 命令。

```
aws codebuild create-project --cli-input-json file://create-project.json
```

如果成功，输出中将显示与此类似的数据。

```
{  
  "project": {  
    "name": "codebuild-demo-project",  
    "serviceRole": "serviceIAMRole",  
    "tags": [],  
    "artifacts": {  
      "packaging": "NONE",  
      "type": "S3",  
      "location": "codebuild-region-ID-account-ID-output-bucket",  
      "name": "message-util.zip"  
    },  
    "lastModified": 1472661575.244,  
    "timeoutInMinutes": 60,  
    "created": 1472661575.244,  
    "environment": {  
      "computeType": "BUILD_GENERAL1_SMALL",  
      "image": "aws/codebuild/standard:4.0",  
      "type": "LINUX_CONTAINER",  
      "environmentVariables": []  
    },  
    "source": {  
      "type": "S3",  
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"  
    },  
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",  
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"  
  }  
}
```

- `project` 表示有关此构建项目的信息。
  - `tags` 表示已经声明的所有标签。
  - `packaging` 表示如何将构建输出伪影存储在输出桶中。NONE 指在输出桶中创建一个文件夹。生成输出项目存储在该文件夹中。
  - `lastModified` 表示构建项目最后一次更改的时间，采用 Unix 时间格式。
  - `timeoutInMinutes` 表示生成未完成时，CodeBuild 会在多少分钟后停止生成。（默认为 60 分钟。）
  - `created` 表示构建项目的创建时间，采用 Unix 时间格式。
  - `environmentVariables` 表示已经声明且可供 CodeBuild 在生成过程中使用的所有环境变量。
  - `encryptionKey` 表示 CodeBuild 用于加密生成输出项目的 AWS KMS 客户主密钥 (CMK) 的 ARN。
  - `arn` 表示构建项目的 ARN。

#### Note

在您运行 `create-project` 命令，可能输出类似以下内容的错误消息: User `user-ARN` 未获授权执行: `codebuild:CreateProject`...这很可能是因为您配置 AWS CLI 凭借一个 IAM 没有足够的权限使用的用户 CodeBuild 创建构建项目。要修复此问题，请使用属于以下任一 IAM 实体的凭证配置 AWS CLI：

- AWS 账户中的 IAM 管理员用户。有关详细信息，请参阅 [创建您的第一个 IAM 管理员用户和组](#) 在 IAM 用户指南。
- 您的 IAM 账户中的 AWS 用户，该 IAM 用户或 IAM 用户所在的 IAM 组挂载了 `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess` 和 `IAMFullAccess` 托管策略。如果您的 AWS 账户中没有 IAM 用户或组具有这些权限，并且您无法将这些权限添加到您的 IAM 用户或组，请与 AWS 账户管理员联系以寻求帮助。有关更多信息，请参阅 [适用于 AWS CodeBuild 的 AWS 托管 \(预定义\) 策略 \(p. 330\)](#)。

## 下一步

[步骤 6. 运行构建。 \(p. 21\)](#)

## 步骤 6. 运行构建。

( 上一步：[步骤 5. 创建构建项目 \(p. 18\)](#) )

在此步骤中，您将指示 AWS CodeBuild 使用构建项目中的设置来运行构建。

### 运行构建

1. 使用 AWS CLI 运行 `start-build` 命令。

```
aws codebuild start-build --project-name project-name
```

Replace `project-name` 在上一步中使用您的构建项目名称（例如，`codebuild-demo-project`）。

2. 如果成功，输出中将显示与以下内容类似的数据：

```
{  
  "build": {  
    "buildComplete": false,  
    "initiator": "user-name",  
    "artifacts": {  
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/message-util.zip"  
    },  
    "projectName": "codebuild-demo-project",  
    "timeoutInMinutes": 60,  
    "buildStatus": "IN_PROGRESS",  
    "environment": {  
      "computeType": "BUILD_GENERAL1_SMALL",  
      "image": "aws/codebuild/standard:4.0",  
      "type": "LINUX_CONTAINER",  
      "environmentVariables": []  
    },  
    "source": {  
      "type": "S3",  
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"  
    },  
    "currentPhase": "SUBMITTED",  
    "version": 1  
  }  
}
```

```
"startTime": 1472848787.882,  
"id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",  
"arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-  
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"  
}  
}
```

- build 表示有关此构建的信息。
  - buildComplete 表示构建是否完成 (true)。否则为 false。
  - initiator 表示启动构建的实体。
  - artifacts 表示有关构建输出的信息，包括其位置。
  - projectName 表示构建项目的名称。
  - buildStatus 表示运行 start-build 命令时当前构建的状态。
  - currentPhase 表示运行 start-build 命令时的当前构建阶段。
  - startTime 表示构建过程开始的时间，采用 Unix 时间格式。
  - id 表示构建的 ID。
  - arn 表示生成的 ARN。

记下此 id 值。您在下一个步骤中需要用到它。

## 下一步

[步骤 7. 查看汇总的构建信息 \(p. 22\)](#)

# 步骤 7. 查看汇总的构建信息

( 上一步：[步骤 6. 运行构建。 \(p. 21\)](#)  )

在此步骤中，您将查看有关生成状态的汇总信息。

## 查看汇总的构建信息

使用 AWS CLI 运行 batch-get-builds 命令。

```
aws codebuild batch-get-builds --ids id
```

Replace **id** 与 id 上一步输出中出现的值。

如果成功，输出中将显示与此类似的数据。

```
{  
  "buildsNotFound": [],  
  "builds": [  
    {  
      "buildComplete": true,  
      "phases": [  
        {  
          "phaseStatus": "SUCCEEDED",  
          "endTime": 1472848788.525,  
          "phaseType": "SUBMITTED",  
          "durationInSeconds": 0,  
          "startTime": 1472848787.882  
        },  
        {  
          "phaseStatus": "PENDING",  
          "endTime": null,  
          "phaseType": "SUBMITTED",  
          "durationInSeconds": 0,  
          "startTime": 1472848787.882  
        }  
      ]  
    }  
  ]  
}
```

```
... The full list of build phases has been omitted for brevity ...
{
    "phaseType": "COMPLETED",
    "startTime": 1472848878.079
}
],
"logs": {
    "groupName": "/aws/codebuild/codebuild-demo-project",
    "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent;group=/aws/codebuild/codebuild-demo-project;stream=38calc4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
    "streamName": "38calc4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
},
"artifacts": {
    "md5sum": "MD5-hash",
    "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/message-util.zip",
    "sha256sum": "SHA-256-hash"
},
"projectName": "codebuild-demo-project",
"timeoutInMinutes": 60,
"initiator": "user-name",
"buildStatus": "SUCCEEDED",
"environment": {
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/standard:4.0",
    "type": "LINUX_CONTAINER",
    "environmentVariables": []
},
"source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
},
"currentPhase": "COMPLETED",
"startTime": 1472848787.882,
"endTime": 1472848878.079,
"id": "codebuild-demo-project:38calc4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
"arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:38calc4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
}
]
}
```

- `buildsNotFound` 表示所有不具备信息的构建的构建 ID。在本示例中，其应该为空。
- `builds` 表示有关每个具备信息的构建项目的信息。在本示例中，输出中只显示了有关一个构建项目的信息。
  - `phases` 表示 CodeBuild 在生成过程中运行的一组生成阶段。有关每个生成阶段的信息将分别列出，其中包括 `startTime`、`endTime` 和 `durationInSeconds`（采用 Unix 时间格式的生成阶段开始时间和结束时间，以及生成阶段的持续时间，以秒为单位），以及 `phaseType`（如 `SUBMITTED`、`PROVISIONING`、`DOWNLOAD_SOURCE`、`INSTALL`、`PRE_BUILD`、`BUILD`、`POST_BUILD`、`UPLOAD_FINALIZING` 或 `COMPLETED`），还有 `phaseStatus`（如 `SUCCEEDED`、`FAILED`、`FAULT`、`TIMED_OUT`、`IN_PROGRESS` 或 `STOPPED`）。首次运行 `batch-get-builds` 命令时，可能不会有太多（或没有）阶段。使用相同构建 ID 再次运行 `batch-get-builds` 命令后，输出中应当会出现更多构建阶段。
  - `logs` 表示 Amazon CloudWatch Logs 中有关构建日志的信息。
  - `md5sum` 和 `sha256sum` 表示构建输出项目的 MD5 和 SHA-256 哈希值。仅当构建项目的创建项目的出现时 `packaging` 值设置为 `ZIP...`（在本教程中未设置此值。）您可以使用这些哈希以及校验和工具确认文件完整性和真实性。

#### Note

您还可以使用 Amazon S3 控制台查看这些哈希值。选中构建输出项目旁边的框，然后依次选择 `Actions`（操作）和 `Properties`（属性）。在 `Properties` 窗格中，展开 `Metadata`，然后查看

x-amz-meta-codebuild-content-md5 和 x-amz-meta-codebuild-content-sha256 的值。（在 Amazon S3 控制台中，构建输出项目的 ETag 值不应解释为 MD5 或 SHA-256 哈希值。）如果您使用 AWS 开发工具包来获取这些哈希值，这些值会被命名为 codebuild-content-md5 和 codebuild-content-sha256。

- endTime 表示构建过程结束的时间，采用 Unix 时间格式。

## 下一步

[步骤 8。查看详细的构建信息 \(p. 24\)](#)

# 步骤 8。查看详细的构建信息

( 上一步：[步骤 7. 查看汇总的构建信息 \(p. 22\)](#) )

在此步骤中，您将查看有关 CloudWatch Logs 中生成项目的详细信息。

### Note

为了保护敏感信息，CodeBuild 日志中隐藏以下内容：

- AWS 访问密钥 ID。有关更多信息，请参阅 AWS Identity and Access Management 用户指南 中的 [管理 IAM 用户的访问密钥](#)。
- 使用参数存储指定的字符串。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。
- 使用 AWS Secrets Manager 指定的字符串。有关更多信息，请参阅 [密钥管理 \(p. 324\)](#)。

### 查看详细的构建信息

1. 使用您的 Web 浏览器，转到上一步的输出中显示的 deepLink 位置（如 <https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE>）。
2. 在 CloudWatch Logs 日志流中，您可以浏览日志事件。默认情况下，只显示最近的一组日志事件。要查看以前的日志事件，请滚动到列表开头。
3. 在本教程中，大多数日志事件包含的是关于 CodeBuild 下载构建相关文件并将其安装到构建环境中的详细信息，您可能并不关心这些信息。您可以使用 Filter events 框来减少显示的信息。例如，如果您在 Filter events (筛选事件) 中输入 “[INFO]”，则只显示包含 [INFO] 的事件。有关详细信息，请参阅 [筛选器和模式语法](#) 在 Amazon CloudWatch 用户指南。

CloudWatch Logs 日志流的这些部分与本教程有关。

```
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]
-----
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample App 1.0
```

```
[Container] 2016/04/15 17:49:44 [INFO]
-----
...
[Container] 2016/04/15 17:49:55 -----
[Container] 2016/04/15 17:49:55 T E S T S
[Container] 2016/04/15 17:49:55 -----
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
...
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INFO] Finished at: 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INFO]
-----
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

在本示例中，CodeBuild 成功完成了预构建、构建和构建后这些构建阶段。它运行单元测试并成功生成 messageUtil-1.0.jar 文件。

## 下一步

[步骤 9 获取构建输出伪影 \(p. 25\)](#)

## 步骤 9 获取构建输出伪影

( 上一步：[步骤 8。查看详细的构建信息 \(p. 24\)](#) )

在此步骤中，您会得到 CodeBuild 生成并上传到输出存储桶的 messageUtil-1.0.jar 文件。

您可以使用 CodeBuild 控制台或 Amazon S3 控制台完成此步骤。

获取构建输出项目 ( AWS CodeBuild 控制台 )

- 与 CodeBuild 控制台仍然打开，并且在上一步中仍显示了构建详细信息页面，选择 构建详情 选项卡并向下滚动 伪影 第节。

Note

(如果未显示构建详细信息页面，请在导航栏中选择 Build history，然后选择 Build run 链接。)

2. 链接到 Amazon S3 文件夹位于 伪造上传位置。此链接打开文件夹 Amazon S3 您在哪里找到 messageUtil-1.0.jar 构建输出工件文件。

获取构建输出项目 (Amazon S3 控制台)

1. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 打开 codebuild-*region-ID-account-ID-output-bucket*。
3. 打开 codebuild-demo-project 文件夹。
4. 打开 target 文件夹，您可以在此处找到 messageUtil-1.0.jar 构建输出项目文件。

## 下一步

[步骤 10 删除S3输入桶 \(p. 26\)](#)

## 步骤 10 删除S3输入桶

( 上一步：[步骤 9 获取构建输出伪影 \(p. 25\)](#) )

为防止您的 AWS 账户持续产生费用，您可以删除本教程中使用的输入存储桶。有关说明，请参阅 Amazon Simple Storage Service 开发人员指南 中的 [删除或清空存储桶](#)。

如果您使用 IAM 用户或管理员 IAM 用户删除此存储桶，则该用户必须具有更多访问权限。将标记 (**### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENTS HERE ###**) 之间的下列语句添加到用户的现有访问策略中。

此语句中的省略号 (...) 旨在力求简洁。请勿删除现有访问策略中的任何语句。请勿在策略中输入这些省略号。

```
{  
    "Version": "2012-10-17",  
    "Id": "...",  
    "Statement": [  
        ### BEGIN ADDING STATEMENT HERE ###  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:DeleteBucket",  
                "s3:DeleteObject"  
            ],  
            "Resource": "*"  
        }  
        ### END ADDING STATEMENT HERE ###  
    ]  
}
```

## 下一步

[总结 \(p. 26\)](#)

## 总结

在本教程中，您使用 AWS CodeBuild 将一组 Java 类文件构建为一个 JAR 文件。然后查看了构建的结果。

您现在可以尝试在自己的场景中使用 CodeBuild。按照[计划构建 \(p. 135\)](#)中的说明进行操作。如果您觉得自己还没准备好，可以尝试生成一些示例。有关更多信息，请参阅[示例 \(p. 27\)](#)。)

# CodeBuild 示例

这些示例组可用于试验 AWS CodeBuild：

## 主题

- 适用于 CodeBuild 的 Microsoft Windows 示例 (p. 27)
- CodeBuild 基于使用案例的示例 (p. 44)

## 适用于 CodeBuild 的 Microsoft Windows 示例

这些示例使用运行 Microsoft Windows Server 2019、.NET Framework 和 .NET Core 开发工具包的 AWS CodeBuild 构建环境，通过使用 C#、F# 和 Visual Basic 编写的代码生成可执行文件。

### Important

运行这些示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS 和 CloudWatch Logs 相关的操作收取的费用。有关更多信息，请参阅 [CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#) 和 [Amazon CloudWatch 定价](#)。

## 运行示例

要运行这些示例，请执行以下操作：

- 按照本主题的“目录结构和文件”部分中的说明操作来创建文件，然后将其上传到 S3 输入存储桶、CodeCommit 或 GitHub 存储库中。

### Important

请不要上传 (*root directory name*)，而只上传 (*root directory name*) 中的文件。  
如果您使用的是 S3 输入存储桶，请务必创建一个包含文件的 ZIP 文件，然后将其上传到输入存储桶。请不要将 (*root directory name*) 添加到 ZIP 文件中，而只添加 (*root directory name*) 中的文件。

- 创建生成项目、运行生成，并遵循[直接运行 CodeBuild \(p. 367\)](#)中的步骤。

如果您使用 AWS CLI 创建构建项目，则 `create-project` 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{  
    "name": "sample-windows-build-project",  
    "source": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-input-bucket/windows-build-input-artifact.zip"  
    },  
    "artifacts": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-output-bucket",  
        "packaging": "ZIP",  
        "name": "windows-build-output-artifact.zip"  
    },  
    "environment": {  
        "type": "WINDOWS_SERVER_2019_CONTAINER",  
        "image": "aws/codebuild/windows-base:2019-1.0",  
        "computeType": "BUILD_GENERAL1_MEDIUM"  
    }  
}
```

```
    },
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

3. 要获取构建输出构件，请在您的 S3 输出存储桶中，将 `windows-build-output-artifact.zip` 文件下载到您的本地计算机或实例。提取内容以获取可执行文件和其他文件。
  - 在 `CSharpHelloWorld\bin\Debug` 目录中可以找到使用 .NET Framework 的 C# 例程的可执行文件 `CSharpHelloWorld.exe`。
  - 在 `FSharpHelloWorld\bin\Debug` 目录中可以找到使用 .NET Framework 的 F# 例程的可执行文件 `FSharpHelloWorld.exe`。
  - 在 `VBHelloWorld\bin\Debug` 目录中可以找到使用 .NET Framework 的 Visual Basic 例程的可执行文件 `VBHelloWorld.exe`。
  - 在 `bin\Debug\netcoreapp3.1` 目录中可以找到使用 .NET 核心的 C# 例程的可执行文件 `HelloWorldSample.dll`。

## 目录结构

这些示例采用以下目录结构。

### C# 和 .NET Framework

```
(root directory name)
|-- buildspec.yml
|-- CSharpHelloWorld.sln
`-- CSharpHelloWorld
    |-- App.config
    |-- CSharpHelloWorld.csproj
    |-- Program.cs
    `-- Properties
        `-- AssemblyInfo.cs
```

### F# 和 .NET Framework

```
(root directory name)
|-- buildspec.yml
|-- FSharpHelloWorld.sln
`-- FSharpHelloWorld
    |-- App.config
    |-- AssemblyInfo.fs
    |-- FSharpHelloWorld.fsproj
    `-- Program.fs
```

### Visual Basic 和 .NET Framework

```
(root directory name)
|-- buildspec.yml
|-- VBHelloWorld.sln
`-- VBHelloWorld
    |-- App.config
    |-- HelloWorld.vb
    |-- VBHelloWorld.vbproj
    `-- My Project
        |-- Application.Designer.vb
        |-- Application.myapp
```

```
| -- AssemblyInfo.vb
| -- Resources.Designer.vb
| -- Resources.resx
| -- Settings.Designer.vb
` -- Settings.settings
```

## C# 和 .NET 核心

```
(root directory name)
|-- buildspec.yml
|-- HelloWorldSample.csproj
`-- Program.cs
```

## Files

这些示例使用以下文件。

## C# 和 .NET Framework

buildspec.yml (在 *(root directory name)*):

```
version: 0.2

env:
  variables:
    SOLUTION: .\CSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -'
      - PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -'
      p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework
      \.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
    artifacts:
      files:
        - .\CSharpHelloWorld\bin\Debug\*
```

CSharpHelloWorld.sln (在 *(root directory name)*):

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "CSharpHelloWorld", "CSharpHelloWorld
\HelloWorldSample.csproj", "{2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}.Release|Any CPU.ActiveCfg = Release|Any CPU
```

```
{2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}.Release|Any CPU.Build.0 = Release|Any CPU
EndGlobalSection
GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
EndGlobalSection
EndGlobal
```

App.config (在 *(root directory name)*\CSharpHelloWorld):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
    </startup>
</configuration>
```

CSharpHelloWorld.csproj (在 *(root directory name)*\CSharpHelloWorld):

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/
developer/msbuild/2003">
    <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props"
Condition="Exists('$(MSBuildExtensionsPath)\$(
$MSBuildToolsVersion)\Microsoft.Common.props')"/>
    <PropertyGroup>
        <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
        <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
        <ProjectGuid>{2F8752D5-E628-4A38-AA7E-BC4B4E697CBB}</ProjectGuid>
        <OutputType>Exe</OutputType>
        <AppDesignerFolder>Properties</AppDesignerFolder>
        <RootNamespace>CSharpHelloWorld</RootNamespace>
        <AssemblyName>CSharpHelloWorld</AssemblyName>
        <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
        <FileAlignment>512</FileAlignment>
        <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    </PropertyGroup>
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
        <PlatformTarget>AnyCPU</PlatformTarget>
        <DebugSymbols>true</DebugSymbols>
        <DebugType>full</DebugType>
        <Optimize>false</Optimize>
        <OutputPath>bin\Debug</OutputPath>
        <DefineConstants>DEBUG;TRACE</DefineConstants>
        <ErrorReport>prompt</ErrorReport>
        <WarningLevel>4</WarningLevel>
    </PropertyGroup>
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
        <PlatformTarget>AnyCPU</PlatformTarget>
        <DebugType>pdbonly</DebugType>
        <Optimize>true</Optimize>
        <OutputPath>bin\Release</OutputPath>
        <DefineConstants>TRACE</DefineConstants>
        <ErrorReport>prompt</ErrorReport>
        <WarningLevel>4</WarningLevel>
    </PropertyGroup>
    <ItemGroup>
        <Reference Include="System" />
        <Reference Include="System.Core" />
        <Reference Include="System.Xml.Linq" />
        <Reference Include="System.Data.DataSetExtensions" />
        <Reference Include="Microsoft.CSharp" />
        <Reference Include="System.Data" />
        <Reference Include="System.Net.Http" />
    </ItemGroup>
```

```
<Reference Include="System.Xml" />
</ItemGroup>
<ItemGroup>
    <Compile Include="Program.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
</ItemGroup>
<ItemGroup>
    <None Include="App.config" />
</ItemGroup>
<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
<!-- To modify your build process, add your task inside one of the targets below and
uncomment it.
    Other similar extension points exist, see Microsoft.Common.targets.
&lt;Target Name="BeforeBuild"&gt;
&lt;/Target&gt;
&lt;Target Name="AfterBuild"&gt;
&lt;/Target&gt;
--&gt;
&lt;/Project&gt;</pre>
```

Program.cs (在 *(root directory name)*\CSharpHelloWorld):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpHelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("Hello World");
            System.Threading.Thread.Sleep(10);
        }
    }
}
```

AssemblyInfo.cs (在 *(root directory name)*\CSharpHelloWorld\Properties):

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("CSharpHelloWorld")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("CSharpHelloWorld")]
[assembly: AssemblyCopyright("Copyright © 2017")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]
```

```
// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("2f8752d5-e628-4a38-aa7e-bc4b4e697cbb")]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

## F# 和 .NET Framework

`buildspec.yml` (在 *(root directory name)*):

```
version: 0.2

env:
  variables:
    SOLUTION: .\FSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -'
      PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -'
      p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework
      \.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
  artifacts:
    files:
      - .\FSharpHelloWorld\bin\Debug\*
```

`FSharpHelloWorld.sln` (在 *(root directory name)*):

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F2A71F9B-5D33-465A-A702-920D77279786}") = "FSharpHelloWorld", "FSharpHelloWorld
\FSharpHelloWorld.fsproj", "{D60939B6-526D-43F4-9A89-577B2980DF62}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
```

```
EndGlobal
```

App.config (在 *(root directory name)*\FSharpHelloWorld):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

AssemblyInfo.fs (在 *(root directory name)*\FSharpHelloWorld):

```
namespace FSharpHelloWorld.AssemblyInfo

open System.Reflection
open System.Runtime.CompilerServices
open System.Runtime.InteropServices

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[<assembly: AssemblyTitle("FSharpHelloWorld")>]
[<assembly: AssemblyDescription("")>]
[<assembly: AssemblyConfiguration("")>]
[<assembly: AssemblyCompany("")>]
[<assembly: AssemblyProduct("FSharpHelloWorld")>]
[<assembly: AssemblyCopyright("Copyright © 2017")>]
[<assembly: AssemblyTrademark("")>]
[<assembly: AssemblyCulture("")>]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[<assembly: ComVisible(false)>]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[<assembly: Guid("d60939b6-526d-43f4-9a89-577b2980df62")>]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [<assembly: AssemblyVersion("1.0.*")>]
[<assembly: AssemblyVersion("1.0.0.0")>]
[<assembly: AssemblyFileVersion("1.0.0.0")>]

do
()
```

FSharpHelloWorld.fsproj (在 *(root directory name)*\FSharpHelloWorld):

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/
developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props"
Condition="Exists('$(MSBuildExtensionsPath)\$(<assembly>)\Microsoft.Common.props')"/>
```

```
<PropertyGroup>
  <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
  <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
  <SchemaVersion>2.0</SchemaVersion>
  <ProjectGuid>d60939b6-526d-43f4-9a89-577b2980df62</ProjectGuid>
  <OutputType>Exe</OutputType>
  <RootNamespace>FSharpHelloWorld</RootNamespace>
  <AssemblyName>FSharpHelloWorld</AssemblyName>
  <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
  <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  <TargetFSharpCoreVersion>4.4.0.0</TargetFSharpCoreVersion>
  <Name>FSharpHelloWorld</Name>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <Optimize>false</Optimize>
  <Tailcalls>false</Tailcalls>
  <OutputPath>bin\Debug</OutputPath>
  <DefineConstants>DEBUG;TRACE</DefineConstants>
  <WarningLevel>3</WarningLevel>
  <PlatformTarget>AnyCPU</PlatformTarget>
  <DocumentationFile>bin\Debug\FSharpHelloWorld.XML</DocumentationFile>
  <Prefer32Bit>true</Prefer32Bit>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  <DebugType>pdbonly</DebugType>
  <Optimize>true</Optimize>
  <Tailcalls>true</Tailcalls>
  <OutputPath>bin\Release</OutputPath>
  <DefineConstants>TRACE</DefineConstants>
  <WarningLevel>3</WarningLevel>
  <PlatformTarget>AnyCPU</PlatformTarget>
  <DocumentationFile>bin\Release\FSharpHelloWorld.XML</DocumentationFile>
  <Prefer32Bit>true</Prefer32Bit>
</PropertyGroup>
<ItemGroup>
  <Reference Include="mscorlib" />
  <Reference Include="FSharp.Core, Version=$(TargetFSharpCoreVersion), Culture=neutral,
    PublicKeyToken=b03f5f7f11d50a3a">
    <Private>True</Private>
  </Reference>
  <Reference Include="System" />
  <Reference Include="System.Core" />
  <Reference Include="System.Numerics" />
</ItemGroup>
<ItemGroup>
  <Compile Include="AssemblyInfo.fs" />
  <Compile Include="Program.fs" />
  <None Include="App.config" />
</ItemGroup>
<PropertyGroup>
  <MinimumVisualStudioVersion Condition=" '$(MinimumVisualStudioVersion)' == '' ">11</
  MinimumVisualStudioVersion>
</PropertyGroup>
<Choose>
  <When Condition=" '$(VisualStudioVersion)' == '11.0' ">
    <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)..\Microsoft SDKs\F#
  \3.0\Framework\v4.0\Microsoft.FSharp.Targets')">
      <FSharpTargetsPath>$(MSBuildExtensionsPath32)..\Microsoft SDKs\F#\3.0\Framework
  \v4.0\Microsoft.FSharp.Targets</FSharpTargetsPath>
    </PropertyGroup>
  </When>
  <Otherwise>
    <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\
  $(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets')">
```

```
<FSharpTargetsPath>$({MSBuildExtensionsPath32})\Microsoft\VisualStudio\v$({VisualStudioVersion})\FSharp\Microsoft.FSharp.Targets</FSharpTargetsPath>
  </PropertyGroup>
  </Otherwise>
</Choose>
<Import Project="$(FSharpTargetsPath)" />
<!-- To modify your build process, add your task inside one of the targets below and uncomment it.
    Other similar extension points exist, see Microsoft.Common.targets.
&lt;Target Name="BeforeBuild"&gt;
&lt;/Target&gt;
&lt;Target Name="AfterBuild"&gt;
&lt;/Target&gt;
--&gt;
&lt;/Project&gt;</pre>
```

Program.fs (在 *(root directory name)*\FSharpHelloWorld):

```
// Learn more about F# at http://fsharp.org
// See the 'F# Tutorial' project for more help.

[<EntryPoint>]
let main argv =
    printfn "Hello World"
    0 // return an integer exit code
```

## Visual Basic 和 .NET Framework

buildspec.yml (在 *(root directory name)*):

```
version: 0.2

env:
  variables:
    SOLUTION: .\VBHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
  artifacts:
    files:
      - .\VBHelloWorld\bin\Debug\*
```

VBHelloWorld.sln (在 *(root directory name)*):

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "VBHelloWorld", "VBHelloWorld\VBHelloWorld.vbproj", "{4DCEC446-7156-4FE6-8CCC-219E34DD409D}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
```

```
Debug|Any CPU = Debug|Any CPU
Release|Any CPU = Release|Any CPU
EndGlobalSection
GlobalSection(ProjectConfigurationPlatforms) = postSolution
{4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
{4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.Build.0 = Debug|Any CPU
{4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.ActiveCfg = Release|Any CPU
{4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.Build.0 = Release|Any CPU
EndGlobalSection
GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
EndGlobalSection
EndGlobal
```

App.config (在 *(root directory name)*\VBHelloWorld):

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
    </startup>
</configuration>
```

HelloWorld.vb (在 *(root directory name)*\VBHelloWorld):

```
Module HelloWorld

    Sub Main()
        MsgBox("Hello World")
    End Sub

End Module
```

VBHelloWorld.vbproj (在 *(root directory name)*\VBHelloWorld):

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/
developer/msbuild/2003">
    <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props"
    Condition="Exists('$(MSBuildExtensionsPath)\$(<PropertyGroup>
        <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
        <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
        <ProjectGuid>{4DCEC446-7156-4FE6-8CCC-219E34DD409D}</ProjectGuid>
        <OutputType>Exe</OutputType>
        <StartupObject>VBHelloWorld.HelloWorld</StartupObject>
        <RootNamespace>VBHelloWorld</RootNamespace>
        <AssemblyName>VBHelloWorld</AssemblyName>
        <FileAlignment>512</FileAlignment>
        <MyType>Console</MyType>
        <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
        <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    </PropertyGroup>
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
        <PlatformTarget>AnyCPU</PlatformTarget>
        <DebugSymbols>true</DebugSymbols>
        <DebugType>full</DebugType>
        <DefineDebug>true</DefineDebug>
        <DefineTrace>true</DefineTrace>
        <OutputPath>bin\Debug\</OutputPath>
        <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    </PropertyGroup>
</Project>
```

```
<NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <DefineDebug>false</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
</PropertyGroup>
<PropertyGroup>
    <OptionExplicit>On</OptionExplicit>
</PropertyGroup>
<PropertyGroup>
    <OptionCompare>Binary</OptionCompare>
</PropertyGroup>
<PropertyGroup>
    <OptionStrict>Off</OptionStrict>
</PropertyGroup>
<PropertyGroup>
    <OptionInfer>On</OptionInfer>
</PropertyGroup>
<ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Deployment" />
    <Reference Include="System.Xml" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="System.Net.Http" />
</ItemGroup>
<ItemGroup>
    <Import Include="Microsoft.VisualBasic" />
    <Import Include="System" />
    <Import Include="System.Collections" />
    <Import Include="System.Collections.Generic" />
    <Import Include="System.Data" />
    <Import Include="System.Diagnostics" />
    <Import Include="System.Linq" />
    <Import Include="System.Xml.Linq" />
    <Import Include="System.Threading.Tasks" />
</ItemGroup>
<ItemGroup>
    <Compile Include="HelloWorld.vb" />
    <Compile Include="My Project\AssemblyInfo.vb" />
    <Compile Include="My Project\Application.Designer.vb">
        <AutoGen>True</AutoGen>
        <DependentUpon>Application.myapp</DependentUpon>
    </Compile>
    <Compile Include="My Project\Resources.Designer.vb">
        <AutoGen>True</AutoGen>
        <DesignTime>True</DesignTime>
        <DependentUpon>Resources.resx</DependentUpon>
    </Compile>
    <Compile Include="My Project\Settings.Designer.vb">
        <AutoGen>True</AutoGen>
        <DependentUpon>Settings.settings</DependentUpon>
        <DesignTimeSharedInput>True</DesignTimeSharedInput>
    </Compile>
</ItemGroup>
<ItemGroup>
    <EmbeddedResource Include="My Project\Resources.resx">
        <Generator>VbMyResourcesResXFileCodeGenerator</Generator>
    </EmbeddedResource>
</ItemGroup>
```

```
<LastGenOutput>Resources.Designer.vb</LastGenOutput>
<CustomToolNamespace>My.Resources</CustomToolNamespace>
<SubType>Designer</SubType>
</EmbeddedResource>
</ItemGroup>
<ItemGroup>
<None Include="My Project\Application.myapp">
<Generator>MyApplicationCodeGenerator</Generator>
<LastGenOutput>Application.Designer.vb</LastGenOutput>
</None>
<None Include="My Project\Settings.settings">
<Generator>SettingsSingleFileGenerator</Generator>
<CustomToolNamespace>My</CustomToolNamespace>
<LastGenOutput>Settings.Designer.vb</LastGenOutput>
</None>
<None Include="App.config" />
</ItemGroup>
<Import Project="$(MSBuildToolsPath)\Microsoft.VisualBasic.targets" />
<!-- To modify your build process, add your task inside one of the targets below and
uncomment it.
    Other similar extension points exist, see Microsoft.Common.targets.
&lt;Target Name="BeforeBuild"&gt;
&lt;/Target&gt;
&lt;Target Name="AfterBuild"&gt;
&lt;/Target&gt;
--&gt;
&lt;/Project&gt;</pre>
```

Application.Designer.vb (在 *(root directory name)*\VBHelloWorld\My Project):

```
'-----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'-----
```

Option Strict On  
Option Explicit On

Application.myapp (在 *(root directory name)*\VBHelloWorld\My Project):

```
<?xml version="1.0" encoding="utf-8"?>
<MyApplicationData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
<MySubMain>false</MySubMain>
<SingleInstance>false</SingleInstance>
<ShutdownMode>0</ShutdownMode>
<EnableVisualStyles>true</EnableVisualStyles>
<AuthenticationMode>0</AuthenticationMode>
<ApplicationType>2</ApplicationType>
<SaveMySettingsOnExit>true</SaveMySettingsOnExit>
</MyApplicationData>
```

AssemblyInfo.vb (在 *(root directory name)*\VBHelloWorld\My Project):

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices
```

```
' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("VBHelloWorld")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("VBHelloWorld")>
<Assembly: AssemblyCopyright("Copyright © 2017")>
<Assembly: AssemblyTrademark("")>

<Assembly: ComVisible(False)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("137c362b-36ef-4c3e-84ab-f95082487a5a")>

' Version information for an assembly consists of the following four values:
'
' Major Version
' Minor Version
' Build Number
' Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>

<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

Resources.Designer.vb (在 *(root directory name)*\VBHelloWorld\My Project):

```
'-----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'-----

Option Strict On
Option Explicit On

Namespace My.Resources

'This class was auto-generated by the StronglyTypedResourceBuilder
'class via a tool like ResGen or Visual Studio.
'To add or remove a member, edit your .ResX file then rerun ResGen
'with the /str option, or rebuild your VS project.
'''<summary>
'''   A strongly-typed resource class, for looking up localized strings, etc.
'''</summary>

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBu
"4.0.0.0"), _>
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _>
  Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _>
  Global.Microsoft.VisualBasic.HideModuleNameAttribute()> _
Friend Module Resources
```

```
Private resourceMan As Global.System.Resources.ResourceManager

Private resourceCulture As Global.System.Globalization.CultureInfo

'''<summary>
''' Returns the cached ResourceManager instance used by this class.
'''</summary>

<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Invisible)>
    Friend ReadOnly Property ResourceManager() As Global.System.Resources.ResourceManager
        Get
            If Object.ReferenceEquals(resourceMan, Nothing) Then
                Dim temp As Global.System.Resources.ResourceManager = New
                    Global.System.Resources.ResourceManager("VBHelloWorld.Resources",
                    GetType(Resources).Assembly)
                resourceMan = temp
            End If
            Return resourceMan
        End Get
    End Property

    '''<summary>
    ''' Overrides the current thread's CurrentUICulture property for all
    ''' resource lookups using this strongly typed resource class.
    '''</summary>

<Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Invisible)>
    Friend Property Culture() As Global.System.Globalization.CultureInfo
        Get
            Return resourceCulture
        End Get
        Set(ByVal value As Global.System.Globalization.CultureInfo)
            resourceCulture = value
        End Set
    End Property
End Module
End Namespace
```

Resources.resx (在 *(root directory name)\VBHelloWorld\My Project*):

```
<?xml version="1.0" encoding="utf-8"?>
<root>
    <!--
        Microsoft ResX Schema

        Version 2.0

        The primary goals of this format is to allow a simple XML format
        that is mostly human readable. The generation and parsing of the
        various data types are done through the TypeConverter classes
        associated with the data types.

        Example:

        ...
        ado.net/XML headers & schema ...
        <resheader name="resmimetype">text/microsoft-resx</resheader>
        <resheader name="version">2.0</resheader>
        <resheader name="reader">System.Resources.ResXResourceReader,
        System.Windows.Forms, ...</resheader>
        <resheader name="writer">System.Resources.ResXResourceWriter,
        System.Windows.Forms, ...</resheader>
        <data name="Name1"><value>this is my long string</value><comment>this is a comment</comment></data>
```

```
<data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
<data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
    <value>[base64 mime encoded serialized .NET Framework object]</value>
</data>
<data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-microsoft.net.object.bytearray.base64">
    <value>[base64 mime encoded string representing a byte array form of the .NET
Framework object]</value>
    <comment>This is a comment</comment>
</data>
```

There are any number of "resheader" rows that contain simple name/value pairs.

Each data row contains a name, and value. The row also contains a type or mimetype. Type corresponds to a .NET class that support text/value conversion through the TypeConverter architecture. Classes that don't support this are serialized and stored with the mimetype set.

The mimetype is used for serialized objects, and tells the ResXResourceReader how to depersist the object. This is currently not extensible. For a given mimetype the value must be set accordingly:

Note - application/x-microsoft.net.object.binary.base64 is the format that the ResXResourceWriter will generate, however the reader can read any of the formats listed below.

```
mimetype: application/x-microsoft.net.object.binary.base64
value   : The object must be serialized with
          : System.Serialization.Formatters.Binary.BinaryFormatter
          : and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.soap.base64
value   : The object must be serialized with
          : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
          : and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.bytearray.base64
value   : The object must be serialized into a byte array
          : using a System.ComponentModel.TypeConverter
          : and then encoded with base64 encoding.

-->
<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xsd:element name="root" msdata:IsDataSet="true">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="metadata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="value" type="xsd:string" minOccurs="0" />
            </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
            </xsd:complexType>
        </xsd:element>
      </xsd:choice>
    </xsd:element>
    <xsd:element name="assembly">
      <xsd:complexType>
        <xsd:attribute name="alias" type="xsd:string" />
        <xsd:attribute name="name" type="xsd:string" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="data">
      <xsd:complexType>
```

```
<xsd:sequence>
    <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
    <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
</xsd:sequence>
<xsd:attribute name="name" type="xsd:string" msdata:Ordinal="1" />
<xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
<xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
</xsd:complexType>
</xsd:element>
<xsd:element name="resheader">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
        </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required" />
        </xsd:complexType>
    </xsd:element>
</xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:schema>
<resheader name="resmimetype">
    <value>text/microsoft-resx</value>
</resheader>
<resheader name="version">
    <value>2.0</value>
</resheader>
<resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
<resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
</resheader>
</root>
```

Settings.Designer.vb (在 *(root directory name)*\VBHelloWorld\My Project):

```
'-----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'-----
Option Strict On
Option Explicit On

Namespace My

    <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
    Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudioEditors.SettingsDesigner
    "11.0.0.0"), _
    Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState
    - Partial Friend NotInheritable Class MySettings
```

```
Inherits Global.System.Configuration.ApplicationSettingsBase

Private Shared defaultInstance As MySettings =
CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New MySettings),
MySettings)

#Region "My.Settings Auto-Save Functionality"
#If _MyType = "WindowsForms" Then
    Private Shared addedHandler As Boolean

    Private Shared addedHandlerLockObject As New Object

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),
Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState
-
    Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal e
As Global.System.EventArgs)
        If My.Application.SaveMySettingsOnExit Then
            My.Settings.Save()
        End If
        End Sub
    #End If
#End Region

Public Shared ReadOnly Property [Default]() As MySettings
    Get

        #If _MyType = "WindowsForms" Then
            If Not addedHandler Then
                SyncLock addedHandlerLockObject
                    If Not addedHandler Then
                        AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                        addedHandler = True
                    End If
                End SyncLock
            End If
        #End If
        Return defaultInstance
    End Get
    End Property
End Class
End Namespace

Namespace My

<Global.Microsoft.VisualBasic.HideModuleNameAttribute(), _
Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute()> _
Friend Module MySettingsProperty

<Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")> _
Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
    Get
        Return Global.VBHelloWorld.My.MySettings.Default
    End Get
    End Property
End Module
End Namespace
```

Settings.settings (在 *(root directory name)*\VBHelloWorld\My Project):

```
<?xml version='1.0' encoding='utf-8'?>
<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings"
CurrentProfile="(Default)" UseMySettingsClassName="true">
<Profiles>
```

```
<Profile Name="(Default)" />
</Profiles>
<Settings />
</SettingsFile>
```

## C# 和 .NET 核心

`buildspec.yml` (在 *(root directory name)*)

```
version: 0.2

phases:
  build:
    commands:
      - dotnet restore
      - dotnet build
  artifacts:
    files:
      - .\bin\Debug\netcoreapp3.1\*
```

`HelloWorldSample.csproj` (在 *(root directory name)*)

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>
</Project>
```

`Program.cs` (在 *(root directory name)*)

```
using System;

namespace HelloWorldSample
{
    public static class Program
    {
        public static void Main()
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

## CodeBuild 基于使用案例的示例

您可以使用这些基于使用案例的示例来试验 AWS CodeBuild：

[访问令牌示例 \(p. 46\)](#)

显示如何在 CodeBuild 中使用访问令牌连接到 GitHub 和 Bitbucket。

[Amazon ECR 示例 \(p. 50\)](#)

使用 Amazon ECR 存储库中的 Docker 映像，以使用 Apache Maven 生成单个 JAR 文件。

[Amazon EFS 示例 \(p. 53\)](#)

显示如何配置构建规范文件，以便在 Amazon EFS 文件系统上挂载和构建 CodeBuild 项目。

[AWS CodeDeploy 示例 \(p. 57\)](#)

使用 Apache Maven 生成单个 JAR 文件。使用 CodeDeploy 将 JAR 文件部署到 Amazon Linux 实例。  
您也可以使用 CodePipeline 来构建和部署示例。

[AWS CodePipeline 与批次构建样品集成 \(p. 64\)](#)

演示如何使用 AWS CodePipeline 创建具有多个输入源和多个输出构件的构建。

[AWS Config 示例 \(p. 66\)](#)

说明如何设置 AWS Config。列出跟踪的 CodeBuild 资源并描述如何在 AWS Config 中查找 CodeBuild 项目。

[AWS Elastic Beanstalk 示例 \(p. 68\)](#)

使用 Apache Maven 生成单个 WAR 文件。使用 Elastic Beanstalk 将 WAR 文件部署到 Elastic Beanstalk 实例。

[Bitbucket 拉取请求和 Webhook 筛选条件示例 \(p. 75\)](#)

将 CodeBuild 和 Bitbucket 一起用作源存储库并启用 Webhook，可在每次将代码更改被推送到存储库时重建源代码。

[构建徽章示例 \(p. 78\)](#)

说明如何使用构建徽章设置 CodeBuild。

[构建通知示例 \(p. 80\)](#)

使用 Apache Maven 生成单个 JAR 文件。给 Amazon SNS 主题的订阅者发送构建通知。

[使用 AWS CLI 示例创建测试报告 \(p. 97\)](#)

使用 AWS CLI 创建、运行和查看测试报告的结果。

[自定义映像示例中的 Docker \(p. 102\)](#)

使用自定义 Docker 映像生成 Docker 映像。

[Docker 示例 \(p. 104\)](#)

使用由支持 Docker 的 CodeBuild 提供的构建映像来通过 Apache Maven 生成一个 Docker 映像。将 Docker 映像推送到 Amazon ECR 中的存储库。您还可以调整此示例，以将 Docker 映像推送到 Docker Hub。

[GitHub Enterprise Server 示例 \(p. 109\)](#)

将 CodeBuild 和 GitHub Enterprise Server 一起用作源存储库，并安装了证书、启用了 Webhook，可在每次将代码更改被推送到存储库时重建源代码。

[GitHub 拉取请求和 Webhook 筛选条件示例 \(p. 115\)](#)

将 CodeBuild 和 GitHub 一起用作源存储库并启用 Webhook，可在每次将代码更改被推送到存储库时重建源代码。

[将构建输出托管在 S3 存储桶中 \(p. 118\)](#)

说明如何使用未加密的构建构件在 S3 存储桶中创建静态网站。

[多输入源和输出构件示例 \(p. 131\)](#)

演示如何在构建项目中使用多个输入源和多个输出构件。

[私有注册表与 AWS Secrets Manager 示例 \(p. 129\)](#)

显示如何在使用存储CodeBuild在 AWS 中的专用注册表凭证Secrets Manager进行构建时将专用注册表中的 Docker 映像用作构建。

### [构建规范文件示例中的运行时版本 \(p. 120\)](#)

说明如何在 buildspec 文件中指定运行时及其版本。这是一项使用 Ubuntu 标准映像版本 2.0 的要求。  
[源版本示例 \(p. 127\)](#)

说明如何在 CodeBuild 构建项目中使用源的特定版本。

### [使用语义版本控制命名构建构件示例 \(p. 133\)](#)

演示如何使用语义版本控制在构建时创建构件名称。

## 在 CodeBuild 中将访问令牌与您的源提供商结合使用

本示例说明了如何使用访问令牌连接到 GitHub 或 Bitbucket。对于 GitHub 或 GitHub Enterprise Server，使用个人访问令牌。对于 Bitbucket，使用应用程序密码。

### 访问令牌先决条件

在开始之前，您必须向访问令牌添加适当的权限范围。

对于 GitHub，您的个人访问令牌必须具有以下权限范围。

- repo：授予私有存储库的完全控制权。
- repo:status：授予提交状态的访问权限。
- admin:repo\_hook：授予存储库挂钩的完全控制权。如果您的令牌具有 repo 范围，则不需要此权限范围。

有关更多信息，请参阅 GitHub 网站上的 [了解 OAuth 应用程序的范围](#)。

对于 Bitbucket，您的应用程序密码必须具有以下权限范围。

- repository:read：授予对授权用户有权访问的所有存储库的读取访问权限。
- pullrequest:read：授予对拉取请求的读取访问权限。如果您的项目具有 Bitbucket Webhook，则您的应用程序密码必须具有此权限范围。
- Webhook：授予对 Webhook 的访问权限。如果您的项目具有 Webhook 操作，则您的应用程序密码必须具有此权限范围。

有关更多信息，请参阅 Bitbucket 网站上的 [Bitbucket 云 REST API 的权限范围](#) 和 [Bitbucket 云上的 OAuth](#)。

### 使用访问令牌连接源提供商（控制台）

要在控制台中使用访问令牌将您的项目连接到 GitHub 或 Bitbucket，请在创建项目时执行以下操作。有关信息，请参阅[创建构建项目（控制台）\(p. 190\)](#)。

对于 GitHub：

- 对于 Source provider (源提供商)，选择 GitHub。
- 对于 Repository (存储库)，选择 Connect with a GitHub personal access token (使用 GitHub 个人访问令牌进行连接)。

**Source**

Add so

Source 1 - Primary

Source provider

GitHub ▾

Repository

AWS CodeBuild needs access to your GitHub account to display available repositories.

Connect using OAuth

Connect with a GitHub personal access token

GitHub personal access token

Save token

▶ Additional configuration

Git clone depth

The screenshot shows the AWS CodeBuild 'Source' configuration page. At the top, there's a 'Source' header and an 'Add so' button. Below it, 'Source 1 - Primary' is selected. The 'Source provider' dropdown is set to 'GitHub'. In the 'Repository' section, a note says 'AWS CodeBuild needs access to your GitHub account to display available repositories.' There are two options: 'Connect using OAuth' (radio button) and 'Connect with a GitHub personal access token' (radio button, which is selected). Below these is a text input field for the 'GitHub personal access token' and a 'Save token' button. At the bottom, there's a '▶ Additional configuration' section with a 'Git clone depth' sub-section.

3. 在 GitHub personal access token (GitHub 个人访问令牌) 中，输入您的 GitHub 个人访问令牌。
4. 选择 Save token (保存令牌)。

对于 Bitbucket :

1. 对于 Source provider (源提供商)，选择 Bitbucket。

Note

CodeBuild 不支持 Bitbucket 服务器。

2. 对于 Repository (存储库)，选择 Connect with a Bitbucket app password (使用 Bitbucket 应用程序密码进行连接)。

**Source** Add source

Source 1 - Primary

Source provider

Bitbucket ▼

Repository

AWS CodeBuild needs access to your Bitbucket account to display available repositories.

Connect using OAuth  Connect with a Bitbucket app password

Bitbucket username

Bitbucket app password

**Save Bitbucket credentials**

► Additional configuration  
Git clone depth

3. 在 Bitbucket username (Bitbucket 用户名) 中，输入您的 Bitbucket 用户名。
4. 在 Bitbucket app password (Bitbucket 应用程序密码) 中，输入您的 Bitbucket 应用程序密码。
5. 选择 Save Bitbucket credentials (保存 Bitbucket 凭证)。

## 使用访问令牌连接源提供商 (CLI)

按照以下步骤在 AWS CLI 中使用访问令牌将您的项目连接到 GitHub 或 Bitbucket。有关将 AWS CLI 与 AWS CodeBuild 结合使用的信息，请参阅[命令行参考 \(p. 363\)](#)。

1. 运行 import-source-credentials 命令：

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

输出中将显示 JSON 格式的数据。将数据复制到本地计算机上或安装 AWS CLI 的实例上某位置处的文件（如 `import-source-credentials.json`）中。按照下面所示修改复制的数据，并保存您的结果。

```
{  
  "serverType": "server-type",  
  "authType": "auth-type",  
  "shouldOverwrite": "should-overwrite",
```

```
    "token": "token",
    "username": "username"
}
```

替换以下内容：

- **server-type**：必填值。用于此凭证的源提供商。有效值为 GITHUB、GITHUB\_ENTERPRISE 和 BITBUCKET。
  - **auth-type**：必填值。用于连接到 GitHub、GitHub Enterprise Server 或 Bitbucket 存储库的身份验证类型。有效值包括 PERSONAL\_ACCESS\_TOKEN 和 BASIC\_AUTH。您不能使用 CodeBuild API 来创建 OAUTH 连接。您必须改为使用 CodeBuild 控制台。
  - **should-overwrite**：可选值。设置为 `false` 可防止覆盖存储库源凭证。设置为 `true` 可覆盖存储库源凭证。默认值为 `true`。
  - **token**：必填值。对于 GitHub 或 GitHub Enterprise Server，这是个人访问令牌。对于 Bitbucket，这是应用程序密码。
  - **username**：可选值。当 authType 为 BASIC\_AUTH 的 Bitbucket 用户名。对于其他类型的源提供商或连接，将忽略此参数。
2. 要使用访问令牌连接您的账户，请切换到包含您在步骤 1 中保存的 `import-source-credentials.json` 文件的目录，然后重新运行 `import-source-credentials` 命令。

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 格式的数据将使用 Amazon 资源名称 (ARN) 显示在输出中。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

#### Note

如果您再次使用相同的服务器类型和身份验证类型运行 `import-source-credentials` 命令，则会更新存储的访问令牌。

在使用访问令牌连接您的账户后，您可以使用 `create-project` 来创建您的 CodeBuild 项目。有关更多信息，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。

3. 要查看连接的访问令牌，请运行 `list-source-credentials` 命令。

```
aws codebuild list-source-credentials
```

JSON 格式的 `sourceCredentialsInfos` 对象将显示在输出中：

```
{
  "sourceCredentialsInfos": [
    {
      "authType": "auth-type",
      "serverType": "server-type",
      "arn": "arn"
    }
  ]
}
```

`sourceCredentialsObject` 包含连接的源凭证信息的列表：

- `authType` 是凭证使用的身份验证类型。这可以是 `OAUTH`、`BASIC_AUTH` 或 `PERSONAL_ACCESS_TOKEN`。
  - `serverType` 是源提供商类型。这可以是 `GITHUB`、`GITHUB_ENTERPRISE` 或 `BITBUCKET`。
  - `arn` 是令牌的 ARN。
4. 要断开与源提供商的连接并删除其访问令牌，请使用其 ARN 运行 `delete-source-credentials` 命令。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

将返回 JSON 格式的数据，并带有已删除证书的 ARN。

```
{  
    "arn": "arn:aws:codebuild:region:account-id:token/server-type"  
}
```

## 适用于 CodeBuild 的 Amazon ECR 示例

此示例使用 Amazon Elastic Container Registry (Amazon ECR) 映像存储库中的 Docker 映像生成示例 Go 项目。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 AWS CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS、CloudWatch Logs 和 Amazon ECR 相关的操作收取的费用。有关更多信息，请参阅 [CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)、[Amazon CloudWatch 定价](#) 和 [Amazon Elastic Container Registry 定价](#)。

## 运行示例

要运行此示例，请执行以下操作：

1. 要创建 Docker 映像并将其推送到 Amazon ECR 中的映像存储库，请完成 [Docker 示例 \(p. 104\)](#) 的“运行示例”部分中的步骤。
2. 创建 Go 项目：
  - a. 按照本主题的[Go 项目结构 \(p. 52\)](#)和[Go 项目文件 \(p. 53\)](#)部分的说明创建文件，然后将其上传到 S3 输入存储桶或者上传到 AWS CodeCommit、GitHub 或 Bitbucket 存储库。

### Important

请不要上传 `(root directory name)`，而只上传 `(root directory name)` 中的文件。

如果您使用的是 S3 输入存储桶，请务必创建一个包含文件的 ZIP 文件，然后将其上传到输入存储桶。请不要将 `(root directory name)` 添加到 ZIP 文件中，而只添加 `(root directory name)` 中的文件。

- b. 请按照 [直接运行 AWS CodeBuild \(p. 367\)](#) 中的步骤创建构建项目、运行构建并查看相关构建信息。

如果您使用 AWS CLI 创建构建项目，则 `create-project` 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{  
    "name": "sample-go-project",  
    "source": {  
        "type": "S3",  
        "location": "s3://my-bucket/my-directory"  
    },  
    "environment": {  
        "computeType": "BUILD_GENERAL1_SMALL",  
        "image": "aws/codebuild/go:1.11",  
        "privilegedMode": false,  
        "environmentVariables": [  
            {"name": "PROJECT_NAME", "value": "sample-go-project"},  
            {"name": "IMAGE_NAME", "value": "my-image"},  
            {"name": "IMAGE_TAG", "value": "latest"}  
        ]  
    },  
    "serviceRole": "arn:aws:iam::123456789012:role/service-role/CodeBuild-ExecutionRole",  
    "vpcConfig": {  
        "subnets": ["subnet-12345678"],  
        "securityGroup": "sg-12345678",  
        "enableNetworking": true  
    }  
}
```

```
        "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"  
    },  
    "artifacts": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-output-bucket",  
        "packaging": "ZIP",  
        "name": "GoOutputArtifact.zip"  
    },  
    "environment": {  
        "type": "LINUX_CONTAINER",  
        "image": "aws/codebuild/standard:4.0",  
        "computeType": "BUILD_GENERAL1_SMALL"  
    },  
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",  
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"  
}
```

- c. 要获取构建输出构件，请打开您的 S3 输出存储桶。
  - d. 将 *GoOutputArtifact.zip* 文件下载到您的本地计算机或实例，然后提取该文件的内容。在提取出来的内容中，获取 hello 文件。
3. 如果满足以下条件之一，则必须向 Amazon ECR 中的映像存储库添加权限，以便 AWS CodeBuild 可以将其 Docker 映像拉取到以下构建环境中：
- 您的项目使用 CodeBuild 凭证来拉取 Amazon ECR 映像。这是由 ProjectEnvironment 的 imagePullCredentialsType 属性中的 CODEBUILD 值指示的。
  - 您的项目使用了跨账户 Amazon ECR 映像。在这种情况下，您的项目必须使用其服务角色拉取 Amazon ECR 映像。要启用此行为，请将您的 ProjectEnvironment 的 imagePullCredentialsType 属性设置为 SERVICE\_ROLE。
1. 通过以下网址打开 Amazon ECR 控制台：<https://console.aws.amazon.com/ecr/>。
  2. 在存储库名称列表中，选择您创建或选择的存储库的名称。
  3. 在导航窗格中，依次选择 Permissions (权限)、Edit (编辑) 和 Add statement (添加语句)。
  4. 对于 Statement name (声明名称)，输入标识符（例如 **CodeBuildAccess**）。
  5. 对于 Effect (效果)，选择 Allow (允许)。这表示您希望允许访问另一个 AWS 账户。
  6. 对于 Principal (委托人)，执行以下操作之一：
    - 如果您的项目使用 CodeBuild 凭证来拉取 Amazon ECR 映像，请在 Service principal (服务委托人) 中输入 **codebuild.amazonaws.com**。
    - 如果您的项目使用跨账户 Amazon ECR 映像，对于 AWS account ID (AWS 账户 ID)，输入您要向其授予访问权限的 AWS 账户的 ID。
  7. 跳过 All IAM entities (所有 IAM 实体) 列表。
  8. 对于 Action (操作)，选择仅拉取操作：ecr:GetDownloadUrlForLayer、ecr:BatchGetImage 和 ecr:BatchCheckLayerAvailability。
  9. 选择 Save (保存)。

此策略显示在 Permissions (权限) 中。委托人是您在此过程的步骤 3 中为 Principal (委托人) 输入的内容：

- 如果您的项目使用 CodeBuild 凭证来拉取 Amazon ECR 映像，Service principals (服务委托人) 下为 "codebuild.amazonaws.com"。
- 如果您的项目使用跨账户 Amazon ECR 映像，则 AWS Account IDs (AWS 账户 ID) 下是您要向其授予访问权限的 AWS 账户的 ID。

以下示例策略同时使用 CodeBuild 凭证和跨账户 Amazon ECR 映像。

```
"Statement": [
    {
        "Sid": "CodeBuildAccessPrincipal",
        "Effect": "Allow",
        "Principal": {
            "Service": "codebuild.amazonaws.com"
        },
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ]
    },
    {
        "Sid": "CodeBuildAccessCrossAccount",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::<AWS-account-ID>:root"
        },
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ]
    }
]
```

4. 请按照 [直接运行 CodeBuild \(p. 367\)](#) 中的步骤创建构建项目、运行构建并查看相关构建信息。

如果您使用 AWS CLI 创建构建项目，则 `create-project` 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{
    "name": "amazon-ecr-sample-project",
    "source": {
        "type": "S3",
        "location": "codebuild-<region-ID>-<account-ID>-input-bucket/GoSample.zip"
    },
    "artifacts": {
        "type": "S3",
        "location": "codebuild-<region-ID>-<account-ID>-output-bucket",
        "packaging": "ZIP",
        "name": "GoOutputArtifact.zip"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "<account-ID>.dkr.ecr.<region-ID>.amazonaws.com/<your-Amazon-ECR-repo-name>:latest",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "arn:aws:iam:<account-ID>:role/<role-name>",
    "encryptionKey": "arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>"
}
```

5. 要获取构建输出构件，请打开您的 S3 输出存储桶。
6. 将 `GoOutputArtifact.zip` 文件下载到您的本地计算机或实例，然后提取 `GoOutputArtifact.zip` 文件的内容。在提取出来的内容中，获取 `hello` 文件。

## Go 项目结构

此示例假定有这样一个目录结构。

```
(root directory name)
|-- buildspec.yml
`-- hello.go
```

## Go 项目文件

此示例将使用这些文件。

`buildspec.yml` (在 `(root directory name)`)

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```

`hello.go` (在 `(root directory name)`)

```
package main
import "fmt"

func main() {
    fmt.Println("hello world")
    fmt.Println("1+1 =", 1+1)
    fmt.Println("7.0/3.0 =", 7.0/3.0)
    fmt.Println(true && false)
    fmt.Println(true || false)
    fmt.Println(!true)
}
```

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

## 适用于 AWS CodeBuild 的 Amazon Elastic File System 示例

您可能希望在 Amazon Elastic File System (一项用于 Amazon EC2 实例的可扩展的共享文件服务) 上创建自己的 AWS CodeBuild 构建。Amazon EFS 中的存储容量是弹性的，因此会随着文件的添加和删除而增长或收缩。它具有简单的 Web 服务界面，可用于创建和配置文件系统。它还为您管理所有文件存储基础设

施，因此您无需担心部署、修补或维护文件系统配置。有关详细信息，请参阅 [什么是 Amazon Elastic File System?](#) 在 AmazonElasticFileSystem 用户指南。

本示例显示了如何配置 CodeBuild 项目，使该项目可挂载 Java 应用程序，然后在 Amazon EFS 文件系统中构建 Java 应用程序。在开始之前，您必须准备好要构建的 Java 应用程序，该应用程序应上传至 S3 输入存储桶或 AWS CodeCommit、GitHub、GitHub Enterprise Server 或 Bitbucket 存储库。

系统会加密文件系统的传输中数据。要使用其他映像来加密传输中的数据，请参阅[加密传输中的数据](#)。

## 概括步骤

本示例介绍了将 Amazon EFS 与 AWS CodeBuild 一起使用所需的三个概括步骤：

1. 在您的 AWS 账户中创建 Virtual Private Cloud (VPC)。
2. 创建使用此 VPC 的文件系统。
3. 创建并构建使用此 VPC 的 CodeBuild 项目。该 CodeBuild 项目使用以下内容来标识文件系统：
  - 文件系统的唯一标识符。当您在构建项目中指定文件系统时，可以选择该标识符。
  - 文件系统 ID。当您在 Amazon EFS 控制台中查看文件系统时，系统会显示该 ID。
  - 挂载点。这是 Docker 容器中用于挂载文件系统的目录。
  - 挂载选项。这些选项包含了有关如何挂载文件系统的详细信息。

### Note

只有 Linux 平台支持在 Amazon EFS 中创建的文件系统。

## 使用 AWS CloudFormation 创建 VPC

使用 AWS CloudFormation 模板创建 VPC。

1. 按照[AWS CloudFormation VPC 模板 \(p. 177\)](#)中的说明使用 AWS CloudFormation 创建 VPC。

### Note

通过此 AWS CloudFormation 模板创建的 VPC 具有两个私有子网和两个公有子网。当您使用 AWS CodeBuild 挂载在 Amazon EFS 中创建的文件系统时，只能使用私有子网。如果您使用其中一个公有子网，则构建会失败。

2. 登录 AWS 管理控制台 并通过以下网址打开 Amazon VPC 控制台：<https://console.aws.amazon.com/vpc/>。
3. 选择您使用 AWS CloudFormation 创建的 VPC。
4. 在 Description (描述) 选项卡上，记下 VPC 的名称及其 ID。在本示例的后面部分中创建您的 AWS CodeBuild 项目时，需要这两者。

## 使用 VPC 创建 Amazon Elastic File System 文件系统

使用您之前创建的 VPC 为本示例创建简单的 Amazon EFS 文件系统。

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon EFS 控制台：<https://console.aws.amazon.com/efs/>。
2. 选择 Create file system (创建文件系统)。
3. 从 VPC，选择您之前在本示例中记录的 VPC 名称。
4. 让可用区保持与您选定的子网关联。
5. 选择 Next Step。

6. 在 Add tags (添加标签) 中，对于默认的 Name (名称) 键，在 Value (值) 中，输入 Amazon EFS 文件系统的名称。
7. 保留 Bursting (突增) 和 General Purpose (通用型) 选定为您的默认性能和吞吐量模式，然后选择 Next Step (下一步)。
8. 对于配置客户端访问，请选择下一步。
9. 选择 Create file system (创建文件系统)。

## 创建 CodeBuild 项目以便与 Amazon EFS 一起使用

创建一个 AWS CodeBuild 项目，该项目使用您之前在本示例中创建的 VPC。运行构建时，它会挂载之前创建的 Amazon EFS 文件系统。接下来，它会将 Java 应用程序创建的 .jar 文件存储在文件系统的挂载点目录中。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 从导航窗格中选择 Build projects (构建项目)，然后选择 Create build project (创建构建项目)。
3. 在 Project name (项目名称) 中，输入您的项目的名称。
4. 从 Source provider (源提供商) 中，选择包含要构建的 Java 应用程序的存储库。
5. 输入 CodeBuild 用于找到您的应用程序的信息，例如存储库 URL。每个源提供商的选项有所不同。有关更多信息，请参阅 [Choose source provider](#)。)
6. 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
7. 从 Operating system (操作系统) 中，选择 Amazon Linux 2。
8. 从 Runtime(s) (运行时) 中，选择 Standard (标准)。
9. 对于 Image (映像)，选择 aws/codebuild/amazonlinux2-x86\_64-standard:3.0。
10. 对于 Environment type (环境类型)，选择 Linux。
11. 选择 Privileged (特权)。

### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

12. 在 Service role (服务角色) 下，选择 New service role (新建服务角色)。在 Role name (角色名称) 中，输入 CodeBuild 为您创建的角色的名称。
13. 展开 Additional configuration (其他配置)。
14. 从 VPC，选择 VPC ID。
15. 对于 Subnets (子网)，选择一个或多个与您的 VPC 关联的私有子网。您必须在挂载 Amazon EFS 文件系统的构建项目中使用私有子网。如果您使用公有子网，则构建会失败。
16. 对于 Security Groups (安全组)，选择默认安全组。
17. 在 File systems (文件系统) 中，输入以下信息：
  - 对于 Identifier (标识符)，输入文件系统的唯一标识符。该标识符必须少于 129 个字符，并且只能包含字母数字字符和下划线。CodeBuild 使用此标识符来创建用于标识弹性文件系统的环境变量。该环境变量的格式为采用大写字母的 `CODEBUILD_file-system-identifier`。例如，如果输入 `efs-1`，则环境变量为 `CODEBUILD_EFS-1`。
  - 对于 ID，请选择文件系统 ID。
  - (可选) 输入文件系统中的目录。CodeBuild 会挂载此目录。如果将 Directory path (目录路径) 留为空白，则 CodeBuild 会挂载整个文件系统。该路径相对于文件系统的根目录指定。
  - 对于 Mount point (挂载点)，输入构建容器中用于挂载文件系统的目录的名称。如果此目录不存在，则 CodeBuild 会在构建过程中创建它。
  - (可选) 输入挂载选项。如果将 Mount options (挂载选项) 留为空白，则 CodeBuild 会使用其默认挂载选项 (`nfsvers=4.1,rsize=1048576,wszie=1048576,hard,timeo=600,retrans=2`)。有关更多信息，请参阅 Amazon Elastic File System 用户指南中的[建议的 NFS 挂载选项](#)。

18. 对于 Build specification (构建规范) , 选择 Insert build commands (插入构建命令) , 然后选择 Switch to editor (切换到编辑器)。
19. 在编辑器中输入以下构建规范命令。将 *file-system-identifier* 替换为您在步骤 17 中输入的标识符。使用大写字母 (例如 CODEBUILD\_EFS-1) 。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto11
  build:
    commands:
      - mvn compile -Dpgp.skip=true -Dmaven.repo.local=$CODEBUILD_<i>file-system-identifier</i>
```

20. 对所有其他设置使用默认值 , 然后选择 Create build project (创建构建项目)。构建完成后 , 系统会显示项目的控制台页面。
21. 选择 Start build。

## CodeBuild 和 Amazon EFS 示例摘要

在 AWS CodeBuild 项目构建完成后 :

- 您会拥有一个由 Java 应用程序创建的 .jar 文件 , 该文件已被构建到您的挂载点目录下的 Amazon EFS 文件系统中。
- 系统会使用您在创建项目时输入的文件系统标识符 , 创建标识文件系统的环境变量。

有关详细信息 , 请参阅 [安装文件系统](#) 在 Amazon Elastic File System 用户指南.

## Troubleshooting

以下是您在设置EFS时可能遇到的错误 CodeBuild.

### 主题

- [client\\_error:安装'127.0.0.1:/'失败。权限被拒绝 \(p. 56\)](#)
- [client\\_error:安装'127.0.0.1:/'失败。对等设置的连接重置 \(p. 56\)](#)
- [VPC\\_CLIENT\\_ERROR: 意外的EC2错误: 未授权的权限 \(p. 57\)](#)

### [client\\_error:安装'127.0.0.1:/'失败。权限被拒绝](#)

使用自定义EFS文件系统策略时 , 必须先建立与EFS之间的信任关系 CodeBuild 通过以下方式之一:

- 添加 codebuild.amazonaws.com 作为EFS文件系统策略中主体的值得信赖的服务 ,
- 添加 elasticfilesystem:ClientMount 对于对应的 CodeBuild 项目服务角色政策。

### [client\\_error:安装'127.0.0.1:/'失败。对等设置的连接重置](#)

这个错误有两个可能的原因:

- TheThe CodeBuild VPC子网位于不同的可用性区域 , 而不是EFS装载目标。您可以通过将vpc子网添加到EFS装载目标中的相同可用性区域来解决这个问题。
- 安全组没有与EFS通信的权限。您可以通过添加入站规则来解决这个问题 , 以允许来自VPC的所有流量 ( 添加用于VPC的主CIDR数据块 ) , 或者安全组本身。

## VPC\_CLIENT\_ERROR: 意外的EC2错误: 未授权的权限

当您的VPC配置中的所有子网都出现此错误 CodeBuild 项目是公共子网。您必须在vpc中至少有一个私人子网，以确保网络连接。

## 适用于 CodeBuild 的 CodeDeploy 示例

此示例指示 AWS CodeBuild 使用 Maven 来生成一个名为 my-app-1.0-SNAPSHOT.jar 的 JAR 文件作为构建输出。然后，此示例使用 CodeDeploy 将 JAR 文件部署到 Amazon Linux 实例。您也可以使用 AWS CodePipeline 来自动使用 CodeDeploy 以将 JAR 文件部署到 Amazon Linux 实例。该示例以 Apache Maven 网站上的 [5 分钟学会 Maven](#) 主题为基础。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS、CloudWatch Logs 和 Amazon EC2 相关的操作收取的费用。有关更多信息，请参阅 [CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)、[Amazon CloudWatch 定价](#)和 [Amazon EC2 定价](#)。

## 运行示例

要运行此示例，请执行以下操作：

1. 下载并安装 Maven。有关更多信息，请参阅 Apache Maven 网站上的[下载 Apache Maven](#) 和[安装 Apache Maven](#)。
2. 切换到您的本地计算机或实例上的空目录，然后运行此 Maven 命令。

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

如果成功，则会创建此目录结构和文件。

```
(root directory name)
    -- my-app
        |-- pom.xml
        '-- src
            |-- main
            |   '-- java
            |       '-- com
            |           '-- mycompany
            |               '-- app
            |                   '-- App.java
            '-- test
                '-- java
                    '-- com
                        '-- mycompany
                            '-- app
                                '-- AppTest.java
```

3. 使用此内容创建文件。将文件命名为 buildspec.yml，然后将其添加到 `(root directory name)/my-app` 目录。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto8
  build:
```

```
commands:
  - echo Build started on `date`
  - mvn test
post_build:
  commands:
    - echo Build completed on `date`
    - mvn package
artifacts:
  files:
    - target/my-app-1.0-SNAPSHOT.jar
    - appspec.yml
discard-paths: yes
```

4. 使用此内容创建文件。将文件命名为 `appspec.yml`，然后将其添加到 `(root directory name)/my-app` 目录。

```
version: 0.0
os: linux
files:
  - source: ./my-app-1.0-SNAPSHOT.jar
    destination: /tmp
```

完成后，您的目录结构和文件应该如下所示。

```
(root directory name)
  -- my-app
    |-- buildspec.yml
    |-- appspec.yml
    |-- pom.xml
    '-- src
      |-- main
      |   '-- java
      |       '-- com
      |           '-- mycompany
      |               '-- app
      |                   '-- App.java
      '-- test
          '-- java
              '-- com
                  '-- mycompany
                      '-- app
                          '-- AppTest.java
```

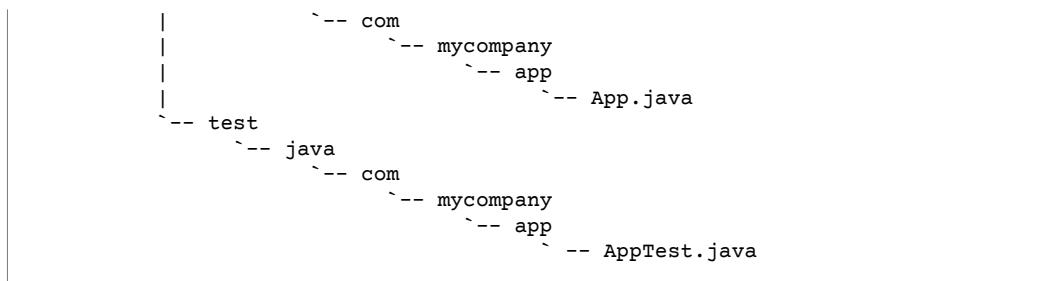
5. 创建一个在 `(root directory name)/my-app` 中包含目录结构和文件的 ZIP 文件，然后将此 ZIP 文件上传到 AWS CodeBuild 和 CodeDeploy 支持的源代码存储库类型，如 S3 输入存储桶或 GitHub 或 Bitbucket 存储库。

#### Important

如果您要使用 CodePipeline 部署构建的构建输出项目，则无法将源代码上传到 Bitbucket 存储库。

请勿将 `(root directory name)` 或 `(root directory name)/my-app` 添加到 ZIP 文件，仅添加 `(root directory name)/my-app` 内的目录和文件。ZIP 文件应包含以下目录和文件：

```
CodeDeploySample.zip
|--buildspec.yml
|-- appspec.yml
|-- pom.xml
`-- src
  |-- main
  |   '-- java
```



6. 请按照 [创建构建项目 \(\) \(p. 189\)](#) 中的步骤创建构建项目。

如果您使用 AWS CLI 创建构建项目，则 `create-project` 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{  
    "name": "sample-codedeploy-project",  
    "source": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-input-bucket/CodeDeploySample.zip"  
    },  
    "artifacts": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-output-bucket",  
        "packaging": "ZIP",  
        "name": "CodeDeployOutputArtifact.zip"  
    },  
    "environment": {  
        "type": "LINUX_CONTAINER",  
        "image": "aws/codebuild/standard:4.0",  
        "computeType": "BUILD_GENERAL1_SMALL"  
    },  
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",  
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"  
}
```

7. 如果您计划使用 CodeDeploy 部署构建输出构件，请按照[运行构建 \(\) \(p. 252\)](#)中的步骤操作。否则，请跳过此步骤。（这是因为，如果您计划使用 CodePipeline 部署构建输出构件，则 CodePipeline 将使用 CodeBuild 自动运行构建。）
8. 完成使用 CodeDeploy 的设置步骤，包括：
  - 为 IAM 用户授予访问 CodeDeploy 以及 CodeDeploy 依赖的 AWS 服务和操作的权限。有关更多信息，请参阅 AWS CodeDeploy 用户指南 中的[预置 IAM 用户](#)。
  - 创建或标识服务角色，使 CodeDeploy 能够标识将在其中部署构建输出构件的实例。有关更多信息，请参阅 AWS CodeDeploy 用户指南 中的[为 CodeDeploy 创建服务角色](#)。
  - 创建或标识 IAM 实例配置文件，使您的实例能够访问包含构建输出构件的 S3 输入存储桶或 GitHub 存储库。有关更多信息，请参阅 AWS CodeDeploy 用户指南 中的[为 Amazon EC2 实例创建 IAM 实例配置文件](#)。
9. 创建或标识在其中部署构建输出构件且与 CodeDeploy 兼容的 Amazon Linux 实例。有关更多信息，请参阅 AWS CodeDeploy 用户指南 中的[使用适用于 CodeDeploy 的实例](#)。
10. 创建或标识 CodeDeploy 应用程序和部署组。有关更多信息，请参阅 AWS CodeDeploy 用户指南 中的[使用 CodeDeploy 创建应用程序](#)。
11. 将构建输出项目部署到实例。

要使用 CodeDeploy 进行部署，请参阅 AWS CodeDeploy 用户指南 中的[使用 CodeDeploy 部署修订](#)。

要使用 CodePipeline 进行部署，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。

12. 要在完成部署后查找构建输出项目，请登录到实例，然后在 `/tmp` 目录中查找名为 `my-app-1.0-SNAPSHOT.jar` 的文件。

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

## AWS CodePipeline 与 CodeBuild 和批次构建

AWS CodeBuild 现在支持批量构建。本示例展示如何使用 AWS CodePipeline 创建使用批量构建的构建项目。

您可以使用定义管道结构的 JSON 格式文件，然后将其与 AWS CLI 配合使用来创建管道。有关详细信息，请参阅[AWS 代码管道管道结构参考](#) 在 AWS 代码管道用户指南。

### 批次构建带单个工件

使用以下 JSON 文件作为管道结构的示例，该结构创建带有独立工件的批次构建。如何启用批次 CodePipeline，设置 BatchEnabled 参数 configuration 对象到 true。

```
{  
  "pipeline": {  
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",  
    "stages": [  
      {  
        "name": "Source",  
        "actions": [  
          {  
            "inputArtifacts": [],  
            "name": "Source1",  
            "actionTypeId": {  
              "category": "Source",  
              "owner": "AWS",  
              "version": "1",  
              "provider": "S3"  
            },  
            "outputArtifacts": [  
              {  
                "name": "source1"  
              }  
            ],  
            "configuration": {  
              "S3Bucket": "my-input-bucket-name",  
              "S3ObjectKey": "my-source-code-file-name.zip"  
            },  
            "runOrder": 1  
          },  
          {  
            "inputArtifacts": [],  
            "name": "Source2",  
            "actionTypeId": {  
              "category": "Source",  
              "owner": "AWS",  
              "version": "1",  
              "provider": "S3"  
            },  
            "outputArtifacts": [  
              {  
                "name": "source2"  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  }  
}
```

```
        ],
        "configuration": {
            "S3Bucket": "my-other-input-bucket-name",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
        },
        "runOrder": 1
    }
]
},
{
    "name": "Build",
    "actions": [
        {
            "inputArtifacts": [
                {
                    "name": "source1"
                },
                {
                    "name": "source2"
                }
            ],
            "name": "Build",
            "actionTypeId": {
                "category": "Build",
                "owner": "AWS",
                "version": "1",
                "provider": "AWS CodeBuild"
            },
            "outputArtifacts": [
                {
                    "name": "build1"
                },
                {
                    "name": "build1_artifact1"
                },
                {
                    "name": "build1_artifact2"
                },
                {
                    "name": "build2_artifact1"
                },
                {
                    "name": "build2_artifact2"
                }
            ],
            "configuration": {
                "ProjectName": "my-build-project-name",
                "PrimarySource": "source1",
                "BatchEnabled": "true"
            },
            "runOrder": 1
        }
    ]
},
"artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
},
"name": "my-pipeline-name",
"version": 1
}
```

以下是一个示例 CodeBuild 将与此管道配置合作的BuildSpec文件。

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
secondary-artifacts:
  artifact1:
    files:
      - output_file
  artifact2:
    files:
      - output_file
```

管道的JSON文件中指定的输出工件的名称必须与BuildSpec文件中定义的构建和伪造的标识符匹配。语法为。*buildIdentifier* 对于主要伪影，以及 *buildIdentifier\_artifactIdentifier* 对于二次工件。

例如，对于输出工件名称 `build1`，CodeBuild 将上传主要伪影 `build1` 至地点 `build1`...对于输出名称 `build1_artifact1`，CodeBuild 将上传二次伪影 `artifact1` 的 `build1` 至地点 `build1_artifact1`，所以在。如果只指定一个输出位置，则名称应为 *buildIdentifier* 仅限。

创建 JSON 文件后，可以创建管道。使用 AWS CLI 运行 `create-pipeline` 命令并将此文件传递给 `--cli-input-json` 参数。有关详细信息，请参阅 [创建管道\(CLI\)](#) 在 AWS 代码管道用户指南。

## 带有合并工件的批次构建

使用以下JSON文件作为管道结构的示例，该结构创建带有合并工件的批次构建。如何启用批次 CodePipeline，设置 `BatchEnabled` 参数 `configuration` 对象到 `true`...要将构建伪影与同一个位置结合起来，请将 `CombineArtifacts` 参数 `configuration` 对象到 `true`.

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
              "version": "1",
              "provider": "S3"
            },
            "outputArtifacts": [
              {
                "name": "source1"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```
        },
    ],
    "configuration": {
        "S3Bucket": "my-input-bucket-name",
        "S3ObjectKey": "my-source-code-file-name.zip"
    },
    "runOrder": 1
},
{
    "inputArtifacts": [],
    "name": "Source2",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "version": "1",
        "provider": "S3"
    },
    "outputArtifacts": [
        {
            "name": "source2"
        }
    ],
    "configuration": {
        "S3Bucket": "my-other-input-bucket-name",
        "S3ObjectKey": "my-other-source-code-file-name.zip"
    },
    "runOrder": 1
}
]
},
{
    "name": "Build",
    "actions": [
        {
            "inputArtifacts": [
                {
                    "name": "source1"
                },
                {
                    "name": "source2"
                }
            ],
            "name": "Build",
            "actionTypeId": {
                "category": "Build",
                "owner": "AWS",
                "version": "1",
                "provider": "AWS CodeBuild"
            },
            "outputArtifacts": [
                {
                    "name": "output1"
                }
            ],
            "configuration": {
                "ProjectName": "my-build-project-name",
                "PrimarySource": "source1",
                "BatchEnabled": "true",
                "CombineArtifacts": "true"
            },
            "runOrder": 1
        }
    ]
},
{
    "artifactStore": {
```

```
        "type": "S3",
        "location": "AWS-CodePipeline-internal-bucket-name"
    },
    "name": "my-pipeline-name",
    "version": 1
}
```

以下是一个示例 CodeBuild 将与此管道配置合作的BuildSpec文件。

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
```

如果批次构建已启用了组合伪影，则只允许一个输出。 CodeBuild 将所有A的主要伪影合并为一个单一邮件文件。

创建 JSON 文件后，可以创建管道。使用 AWS CLI 运行 `create-pipeline` 命令并将此文件传递给 `--cli-input-json` 参数。有关详细信息，请参阅 [创建管道\(CLI\)](#) 在 AWS 代码管道用户指南。

## AWS CodePipeline 与 CodeBuild 和多输入源和输出构件集成示例

AWS CodeBuild 项目可以接受多个输入源，也可以创建多个输出构件。此示例演示如何使用 AWS CodePipeline 创建一个使用多输入源创建多输出构件的构建项目。有关更多信息，请参阅[多输入源和输出构件示例 \(p. 131\)](#)。

您可以使用定义管道结构的 JSON 格式文件，然后将其与 AWS CLI 配合使用来创建管道。使用以下 JSON 文件作为管道结构的示例，此管道结构创建一个具有多输入源和多输出项目的构建。稍后，此示例会介绍该文件如何指定多个输入和输出。有关更多信息，请参阅 AWS CodePipeline 用户指南 中的 [AWS CodePipeline 管道结构参考](#)。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
```

```
        "version": "1",
        "provider": "S3"
    },
    "outputArtifacts": [
        {
            "name": "source1"
        }
    ],
    "configuration": {
        "S3Bucket": "my-input-bucket-name",
        "S3ObjectKey": "my-source-code-file-name.zip"
    },
    "runOrder": 1
},
{
    "inputArtifacts": [],
    "name": "Source2",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "version": "1",
        "provider": "S3"
    },
    "outputArtifacts": [
        {
            "name": "source2"
        }
    ],
    "configuration": {
        "S3Bucket": "my-other-input-bucket-name",
        "S3ObjectKey": "my-other-source-code-file-name.zip"
    },
    "runOrder": 1
}
],
},
{
    "name": "Build",
    "actions": [
        {
            "inputArtifacts": [
                {
                    "name": "source1"
                },
                {
                    "name": "source2"
                }
            ],
            "name": "Build",
            "actionTypeId": {
                "category": "Build",
                "owner": "AWS",
                "version": "1",
                "provider": "AWS CodeBuild"
            },
            "outputArtifacts": [
                {
                    "name": "artifact1"
                },
                {
                    "name": "artifact2"
                }
            ],
            "configuration": {
                "ProjectName": "my-build-project-name",
                "PrimarySource": "source1"
            }
        }
    ]
}
```

```
        },
        "runOrder": 1
    }
],
"artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
},
"name": "my-pipeline-name",
"version": 1
}
}
```

在此 JSON 文件中：

- 必须将输入源之一指定为 PrimarySource。此源是 CodeBuild 查找和运行 buildspec 文件的目录。关键字 PrimarySource 用于在 JSON 文件的 CodeBuild 阶段的 configuration 部分中指定主要源。
- 每个输入源都安装在各自的目录中。此目录存储在内置环境变量 \$CODEBUILD\_SRC\_DIR (对于主源) 和 \$CODEBUILD\_SRC\_DIR\_yourInputArtifactName (对于所有其他源) 中。对于此示例中的管道，两个输入源目录为 \$CODEBUILD\_SRC\_DIR 和 \$CODEBUILD\_SRC\_DIR\_source2。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)。
- 管道的 JSON 文件中指定的输出构件的名称必须与 buildspec 文件中定义的辅助构件的名称相匹配。此管道使用以下 buildspec 文件。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#)。

```
version: 0.2

phases:
  build:
    commands:
      - touch source1_file
      - cd $CODEBUILD_SRC_DIR_source2
      - touch source2_file

artifacts:
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR
      files:
        - source1_file
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - source2_file
```

创建 JSON 文件后，可以创建管道。使用 AWS CLI 运行 create-pipeline 命令并将此文件传递给 --cli-input-json 参数。有关更多信息，请参阅 AWS CodePipeline 用户指南 中的 [创建管道 \(CLI\)](#)。

## 将 AWS Config 与 CodeBuild 结合使用的示例

AWS Config 提供了您的 AWS 资源的清单以及这些资源的配置更改历史记录。AWS Config 现在支持 AWS CodeBuild 作为 AWS 资源，这意味着该服务可以跟踪您的 CodeBuild 项目。有关 AWS Config 的更多信息，请参阅 AWS Config 开发人员指南 中的 [什么是 AWS Config ?](#)。

您可以在 AWS Config 控制台中的 Resource Inventory (资源库存) 页面上查看有关 CodeBuild 资源的以下信息：

- 您的 CodeBuild 配置更改的时间线。

- 每个 CodeBuild 项目的配置详细信息。
- 与其他 AWS 资源的关系。
- 您的 CodeBuild 项目的更改列表。

本主题中的过程展示了如何设置 AWS Config 以及如何查找和查看 CodeBuild 项目。

#### 主题

- [先决条件 \(p. 67\)](#)
- [设置 AWS Config \(p. 67\)](#)
- [查找 AWS CodeBuild 项目 \(p. 67\)](#)
- [在 AWS Config 控制台中查看 AWS CodeBuild 配置详细信息 \(p. 67\)](#)

## 先决条件

创建 AWS CodeBuild 项目。有关说明，请参阅 [创建构建项目 \(\) \(p. 189\)](#)。

## 设置 AWS Config

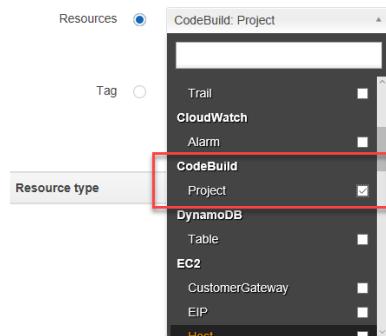
- [设置 AWS Config \( 控制台 \)](#)
- [设置 AWS Config \(AWS CLI\)](#)

#### Note

完成设置后，可能需要等待最多 10 分钟，之后才能在 AWS Config 控制台中看到 AWS CodeBuild 项目。

## 查找 AWS CodeBuild 项目

1. 登录 AWS 管理控制台，并在 <https://console.aws.amazon.com/config> 中打开 AWS Config 控制台。
2. 在 Resource inventory 页面上，选择 Resources。向下滚动并选中 CodeBuild project 复选框。



3. 选择 Look up。
4. 在添加 CodeBuild 项目的列表之后，选择 Config timeline (配置时间线) 列中的 CodeBuild 项目名称链接。

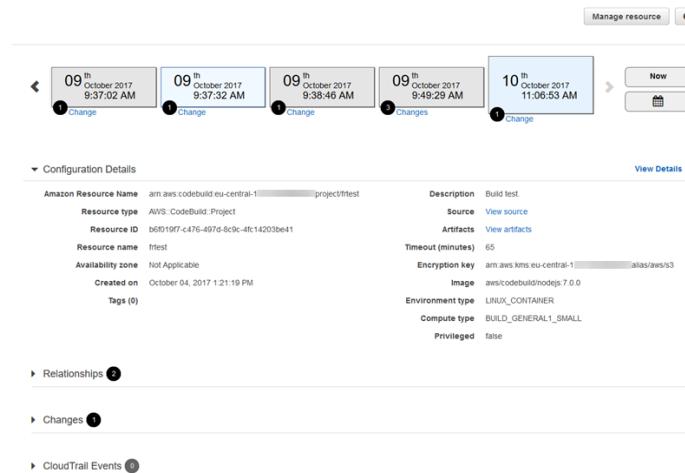
## 在 AWS Config 控制台中查看 AWS CodeBuild 配置详细信息

当您在 Resource inventory (资源库存) 页面上查找资源时，可选择 AWS Config 时间线以查看有关您的 CodeBuild 项目的详细信息。资源的详细信息页面提供了有关该资源的配置、关系和更改次数的信息。

页面顶部的块统称为时间线。时间线显示了记录的创建日期和时间。

有关更多信息，请参阅 AWS Config 开发人员指南 中的[在 AWS Config 控制台中查看配置详细信息](#)。

AWS Config 中的 CodeBuild 项目的示例：



## 适用于 CodeBuild 的 AWS Elastic Beanstalk 示例

此示例指示 AWS CodeBuild 使用 Maven 来生成一个名为 `my-web-app.war` 的 WAR 文件作为构建输出。然后该示例会将 WAR 文件部署到 AWS Elastic Beanstalk 环境中的实例中。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS、CloudWatch Logs 和 Amazon EC2 相关的操作收取的费用。有关更多信息，请参阅[CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)、[Amazon CloudWatch 定价](#)和[Amazon EC2 定价](#)。

## 创建源代码

在本节中，您将使用 Maven 生成源代码。稍后，您将使用 CodeBuild 基于该源代码生成 WAR 文件。

1. 下载并安装 Maven。有关信息，请参阅 Apache Maven 网站上的[下载 Apache Maven](#) 和[安装 Apache Maven](#)。
2. 切换到您的本地计算机或实例上的空目录，然后运行此 Maven 命令。

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-web-app -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

如果成功，将创建此目录结构和文件。

```
(root directory name)
  '-- my-web-app
    '-- pom.xml
    '-- src
      '-- main
        '-- resources
        '-- webapp
          '-- WEB-INF
            '-- web.xml
          '-- index.jsp
```

- 在 *(root directory name)*/my-web-app 目录中创建名为 .ebextensions 的子目录。在 .ebextensions 子目录中，使用此内容创建一个名为 fix-path.config 的文件。

```
container_commands:  
  fix_path:  
    command: "unzip my-web-app.war 2>&1 > /var/log/my_last_deploy.log"
```

在您运行 Maven 后，请继续执行以下方案之一：

- 方案 A：手动运行 CodeBuild 并手动部署到 Elastic Beanstalk (p. 69)
- 方案 B：使用 CodePipeline 运行 CodeBuild 并部署到 Elastic Beanstalk 中 (p. 71)
- 方案 C：使用 Elastic Beanstalk CLI 运行 AWS CodeBuild 并将其部署到 Elastic Beanstalk 环境中 (p. 73)

## 方案 A：手动运行 CodeBuild 并手动部署到 Elastic Beanstalk

在此方案中，您将创建并上传源代码。然后，您将使用 AWS CodeBuild 和 AWS Elastic Beanstalk 控制台生成源代码，创建 Elastic Beanstalk 应用程序和环境，并将生成输出部署到环境中。

### 步骤 a1：将文件添加到源代码

在此步骤中，您需要将 Elastic Beanstalk 配置文件和构建规范文件添加到[创建源代码 \(p. 68\)](#)的代码中。然后，您将源代码上传到 S3 输入存储桶或 CodeCommit、GitHub 或 Bitbucket 存储库。

- 使用以下内容创建名为 buildspec.yml 的文件。将此文件存储到 *(root directory name)*/my-web-app 目录。

```
version: 0.2  
  
phases:  
  install:  
    runtime-versions:  
      java: corretto11  
  post_build:  
    commands:  
      - mvn package  
      - mv target/my-web-app.war my-web-app.war  
  artifacts:  
    files:  
      - my-web-app.war  
      - .ebextensions/**/*
```

- 您的文件结构现在应如下所示。

```
(root directory name)  
  -- my-web-app  
    -- .ebextensions  
      -- fix-path.config  
    -- src  
      -- main  
        -- resources  
        -- webapp  
          -- WEB-INF  
            -- web.xml  
          -- index.jsp  
    -- buildspec.yml  
  -- pom.xml
```

3. 将 my-web-app 目录的内容上传到 S3 输入存储桶或上传到 CodeCommit、GitHub 或 Bitbucket 存储库。

**Important**

请不要上传 *(root directory name)* 或 *(root directory name)/my-web-app*，而只上传 *(root directory name)/my-web-app* 中的目录和文件。

如果您使用的是 S3 输入存储桶，则它必须受版本控制。请确保创建一个包含目录结构和文件的 ZIP 文件，然后将其上传至输入存储桶。请不要将 *(root directory name)* 或 *(root directory name)/my-web-app* 添加到 ZIP 文件中，而只添加 *(root directory name)/my-web-app* 中的目录和文件。有关更多信息，请参阅 Amazon S3 开发人员指南中的[如何对存储桶配置版本控制](#)。

## 步骤 a2：创建构建项目并运行构建

在此步骤中，您将使用 AWS CodeBuild 控制台创建构建项目，然后运行构建。

1. 创建或选择 S3 输出存储桶以存储构建输出。如果您将源代码存储在 S3 输入存储桶中，则输出存储桶必须与输入存储桶位于同一个 AWS 区域中。
2. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.

使用 AWS 区域选择器选择支持 CodeBuild 的 AWS 区域。这必须是用于存储 S3 输出存储桶的同一区域。

3. 创建构建项目，然后运行构建。有关更多信息，请参阅[创建构建项目（控制台）\(p. 190\)](#)和[运行构建（控制台）\(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值。

• 对于 Environment (环境)：

- 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
- 对于 Operating system (操作系统)，选择 Amazon Linux 2。
- 对于 Runtime(s) (运行时)，选择 Standard (标准)。
- 对于 Image (映像)，选择 aws/codebuild/amazonlinux2-x86\_64-standard:2.0。

• 对于 Artifacts (构件)：

- 对于 Type (类型)，选择 Amazon S3。
- 对于 Bucket name (存储桶名称)，输入 S3 存储桶的名称。
- 对于 Name (名称)，键入您容易记住的生成输出文件名称。包括 .zip 扩展名。
- 对于 Artifacts packaging，选择 Zip。

## 步骤 a3：创建应用程序和环境并部署

在此步骤中，您将使用 AWS Elastic Beanstalk 控制台创建应用程序和环境。作为创建环境的一部分，您要将之前步骤的生成输出部署到环境中。

1. 通过 <https://console.aws.amazon.com/elasticbeanstalk> 打开 AWS Elastic Beanstalk 控制台。

使用 AWS 区域选择器选择将 S3 输出存储桶存储到的 AWS 区域。

2. 创建 Elastic Beanstalk 应用程序。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南中的[管理和配置 AWS Elastic Beanstalk 应用程序](#)。
3. 为此应用程序创建 Elastic Beanstalk 环境。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南中的[创建新环境向导](#)。除这些设置以外，将所有设置保留为默认值。

• 对于 Platform，选择 Tomcat。

- 对于 Application code，选择 Upload your code，然后选择 Upload。对于 Source code origin (源代码来源)，选择 Public S3 URL (公共 S3 URL)，然后将完整的 URL 键入到输出存储桶中的生成输出 ZIP 文件中。选择 Upload (上传)。

- 在 Elastic Beanstalk 将构建输出部署至环境后，您可以在 Web 浏览器中查看结果。转到实例的环境 URL (例如，<http://my-environment-name.random-string.region-ID.elasticbeanstalk.com>)。Web 浏览器将显示文字 Hello World!。

## 方案 B：使用 CodePipeline 运行 CodeBuild 并部署到 Elastic Beanstalk 中

在此方案中，您将完成准备和上传源代码的步骤。您将使用 CodeBuild 创建一个构建项目，并使用 AWS Elastic Beanstalk 控制台创建一个 Elastic Beanstalk 应用程序和环境。然后，您使用 AWS CodePipeline 控制台创建管道。在您创建管道之后，CodePipeline 会自动生成源代码，并将生成输出部署到环境中。

### 步骤 b1：将构建规范文件添加到源代码

在此步骤中，您需要创建构建规范文件并将其添加到您在[创建源代码 \(p. 68\)](#)中创建的代码中。然后，您将源代码上传到 S3 输入存储桶或 CodeCommit、GitHub 或 Bitbucket 存储库。

- 使用以下内容创建名为 buildspec.yml 的文件。将此文件存储到 `(root directory name)/my-web-app` 目录。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app.war my-web-app.war
artifacts:
  files:
    - my-web-app.war
    - .ebextensions/**/*
base-directory: 'target/my-web-app'
```

- 您的文件结构现在应如下所示。

```
(root directory name)
  -- my-web-app
    -- .ebextensions
      -- fix-path.config
    -- src
      -- main
        -- resources
          -- webapp
            -- WEB-INF
              -- web.xml
            -- index.jsp
    -- buildspec.yml
    -- pom.xml
```

- 将 `my-web-app` 目录的内容上传到 S3 输入存储桶或上传到 CodeCommit、GitHub 或 Bitbucket 存储库。

#### Important

请不要上传 `(root directory name)` 或 `(root directory name)/my-web-app`，而只上传 `(root directory name)/my-web-app` 中的目录和文件。

如果您使用的是 S3 输入存储桶，则它必须受版本控制。请确保创建一个包含目录结构和文件的 ZIP 文件，然后将其上传至输入存储桶。请不要将 `(root directory name)` 或 `(root`

`directory name`) /my-web-app 添加到 ZIP 文件中，而只添加 (`root directory name`) /my-web-app 中的目录和文件。有关更多信息，请参阅 Amazon S3 开发人员指南 中的 [如何对存储桶配置版本控制](#)。

## 步骤 b2：创建构建项目

在本步骤中，您将创建要用于您的管道的 AWS CodeBuild 构建项目。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 创建构建项目。有关更多信息，请参阅[创建构建项目（控制台）\(p. 190\)](#)和[运行构建（控制台）\(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值。
  - 对于 Environment (环境)：
    - 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
    - 对于 Operating system (操作系统)，选择 Amazon Linux 2。
    - 对于 Runtime(s) (运行时)，选择 Standard (标准)。
    - 对于 Image (映像)，选择 aws/codebuild/amazonlinux2-x86\_64-standard:2.0。
  - 对于 Artifacts (构件)：
    - 对于 Type (类型)，选择 Amazon S3。
    - 对于 Bucket name (存储桶名称)，输入 S3 存储桶的名称。
    - 对于 Name (名称)，键入您容易记住的生成输出文件名称。包括 .zip 扩展名。
    - 对于 Artifacts packaging，选择 Zip。

## 步骤 b3：创建 Elastic Beanstalk 应用程序和环境

在此步骤中，您将创建要用于 CodePipeline 的 Elastic Beanstalk 应用程序和环境。

1. 通过以下网址打开 Elastic Beanstalk 控制台：<https://console.aws.amazon.com/elasticbeanstalk/>。
2. 使用 AWS Elastic Beanstalk 控制台创建应用程序。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南 中的[管理和配置 AWS Elastic Beanstalk 应用程序](#)。
3. 使用 AWS Elastic Beanstalk 控制台创建环境。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南 中的[新建环境向导](#)。除了 Platform (平台) 之外，保留所有设置的默认值。对于 Platform，选择 Tomcat。

## 步骤 b4：创建管道并部署

在此步骤中，您将使用 AWS CodePipeline 控制台创建管道。在您创建并运行管道之后，CodePipeline 使用 CodeBuild 生成源代码，CodePipeline 然后使用 Elastic Beanstalk 将生成输出部署到环境中。

1. 创建或标识 CodePipeline、CodeBuild 和 Elastic Beanstalk 可用来代表您访问资源的服务角色。有关更多信息，请参阅[Prerequisites \(p. 368\)](#)。
2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 中打开 CodePipeline 控制台。

使用 AWS 区域选择器选择支持 CodeBuild 的 AWS 区域。如果您将源代码存储在 S3 输入存储桶中，则输出存储桶必须与输入存储桶位于同一个 AWS 区域中。

3. 创建管道。有关信息，请参阅[创建使用 CodeBuild 的管道（CodePipeline 控制台）\(p. 369\)](#)。除这些设置以外，将所有设置保留为默认值。
  - 在 Add build stage (添加构建阶段) 上，对于 Build provider (构建提供商)，选择 AWS CodeBuild。对于项目名称，选择您刚刚创建的构建项目。

- 在 Add deploy stage (添加部署阶段) 上，对于 Deploy provider (部署提供商)，选择 AWS Elastic Beanstalk。
    - 对于 Application name (应用程序名称)，选择您刚刚创建的 Elastic Beanstalk 应用程序。
    - 对于 Environment name (环境名称)，选择您刚刚创建的环境。
4. 在管道成功运行之后，您可以在 Web 浏览器中查看结果。转到实例的环境 URL (例如，<http://my-environment-name.random-string.region-ID.elasticbeanstalk.com>)。Web 浏览器将显示文字 Hello World!。

现在，只要您更改源代码并将这些更改上传到原始 S3 输入存储桶或者上传到 CodeCommit、GitHub 或 Bitbucket 存储库，CodePipeline 就会检测更改并再次运行管道。这将导致 CodeBuild 重新生成代码，然后 Elastic Beanstalk 会将重新生成输出部署到环境中。

## 方案 C：使用 Elastic Beanstalk CLI 运行 AWS CodeBuild 并将其部署到 Elastic Beanstalk 环境中

在此方案中，您将完成准备和上传源代码的步骤。然后，您将运行 Elastic Beanstalk CLI 创建 Elastic Beanstalk 应用程序和环境，使用 CodeBuild 生成源代码，并将生成输出部署到环境中。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南 中的[将 EB CLI 与 CodeBuild 结合使用](#)。

### 步骤 c1：将文件添加到源代码

在此步骤中，您需要将 Elastic Beanstalk 配置文件和构建规范文件添加到您在[创建源代码 \(p. 68\)](#)中创建的代码中。您还要创建或标识适用于构建规范文件的服务角色。

- 创建或标识 Elastic Beanstalk 和 CLI 可以代表您使用的服务角色。有关信息，请参阅[创建 CodeBuild 服务角色 \(p. 357\)](#)。
- 使用以下内容创建名为 buildspec.yml 的文件。将此文件存储到 [`\(root directory name\)/my-web-app`](#) 目录。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app.war my-web-app.war
artifacts:
  files:
    - my-web-app.war
    - .ebextensions/**/*
eb_codebuild_settings:
  CodeBuildServiceRole: my-service-role-name
  ComputeType: BUILD_GENERAL1_SMALL
  Image: aws/codebuild/standard:4.0
  Timeout: 60
```

在前面的代码中，将 *my-service-role-name* 替换为您之前创建或标识的服务角色的名称。

- 您的文件结构现在应如下所示。

```
\(root directory name\)
  `-- my-web-app
```

```
|-- .ebextensions
|   '-- fix-path.config
|-- src
|   '-- main
|       |-- resources
|           '-- webapp
|               |-- WEB-INF
|                   '-- web.xml
|               '-- index.jsp
|-- buildpsec.yml
`-- pom.xml
```

## 步骤 c2：安装并运行 EB CLI

1. 如果您尚未完成此操作，请在您创建源代码的同一计算机或实例上安装和配置 EB CLI。有关信息，请参阅 AWS Elastic Beanstalk 开发人员指南 中的 [安装 Elastic Beanstalk 命令行界面 \(EB CLI\)](#) 和 [配置 EB CLI](#)。
2. 从命令行或终端，运行 cd 命令或类似命令以切换到您的 (*root directory name*) /my-web-app 目录。运行 eb init 命令配置 EB CLI。

```
eb init
```

出现提示时：

- 选择一个支持 AWS CodeBuild 且您要在其中创建 Elastic Beanstalk 应用程序和环境的 AWS 区域。
  - 创建 Elastic Beanstalk 应用程序，然后输入该应用程序的名称。
  - 选择 Tomcat 平台。
  - 选择 Tomcat 8 Java 8 版本。
  - 选择是否要使用 SSH 设置对您的环境实例的访问。
3. 从同一目录中，运行 eb create 命令创建 Elastic Beanstalk 环境。

```
eb create
```

出现提示时：

- 输入环境的名称，或者接受建议的名称。
  - 输入该环境的 DNS 别名记录前缀，或者接受建议值。
  - 对于此示例，接受 Classic 负载均衡器类型。
4. 在您运行 eb create 命令之后，EB CLI 将执行以下操作：
    1. 从源代码中创建一个 ZIP 文件，然后将此 ZIP 文件上传到您账户中的 S3 存储桶。
    2. 创建 Elastic Beanstalk 应用程序和应用程序版本。
    3. 创建一个 CodeBuild 项目。
    4. 基于新项目运行构建。
    5. 构建完成后删除该项目。
    6. 创建 Elastic Beanstalk 环境。
    7. 将构建输出部署到环境中。
  5. 在 EB CLI 将构建输出部署到环境中后，您可以在 Web 浏览器中查看结果。转到实例的环境 URL (例如，<http://my-environment-name.random-string.region-ID.elasticbeanstalk.com>)。Web 浏览器将显示文字 Hello World!。

如果需要，您可以更改源代码，然后从同一目录运行 eb deploy 命令。EB CLI 与 eb create 命令执行的步骤相同，但它将生成输出部署到现有环境中，而不是创建新的环境。

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

# CodeBuild 的 Bitbucket 拉取请求和 Webhook 筛选条件示例

AWS CodeBuild 当源存储库为位数据存储区时，支持Webhook。这意味着 CodeBuild 创建一个位于位置存储库中的源代码的项目，可以使用网钩来重建每次将代码更改推送到存储库的源代码。有关更多信息，请参阅[BitbucketWebHook事件 \(p. 221\)](#)。)

此示例向您演示如何使用 Bitbucket 存储库创建拉取请求。它还向您演示如何使用 Bitbucket Webhook 来触发 CodeBuild 创建一个项目的生成。

### 主题

- [Prerequisites \(p. 75\)](#)
- [创建将 Bitbucket 作为源存储库的构建项目并启用 Webhook \(p. 75\)](#)
- [使用 Bitbucket Webhook 触发构建 \(p. 77\)](#)

## Prerequisites

要运行此示例，您必须将 AWS CodeBuild 项目与您的 Bitbucket 账户相连接。

### Note

CodeBuild 通过 Bitbucket 更新了其权限。如果您以前已将项目连接到 Bitbucket 但现在收到 Bitbucket 连接错误，您必须重新连接以授予 CodeBuild 权限来管理您的 Webhook。

## 创建将 Bitbucket 作为源存储库的构建项目并启用 Webhook

以下步骤介绍如何创建一个 AWS CodeBuild 项目，使用 Bitbucket 作为源存储库并启用 Webhook。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
3. 在 Create build project (创建构建项目) 页面上的 Project configuration (项目配置) 中，输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。您也可以包含构建项目的可选描述来帮助其他用户了解此项目的用途。
4. 在 Source (源) 中，对于 Source provider (源提供商)，选择 Bitbucket。

**Source** Add source

Source 1 - Primary

Source provider Bitbucket

Repository  Repository in my Bitbucket account

Bitbucket repository iversonic/test C

Connection status  
You are connected to Bitbucket using OAuth.

Disconnect from Bitbucket

Additional configuration  
Git clone depth, Git submodules

Git clone depth - *optional* 1

Git submodules - *optional*  Enable git submodules

Build Status - *optional*  Report build statuses to source provider when your builds start and finish

按照说明进行连接或重新连接，然后选择 Grant access (授予访问权限)。

Note

CodeBuild 不支持 Bitbucket 服务器。

### Confirm access to your account

AWS CodeBuild (N. Virginia) is requesting access to the following:

- 👤 Read your account information
- ⬆️ Read your repositories and their pull requests
- ⚙️ Read and modify your repositories' webhooks

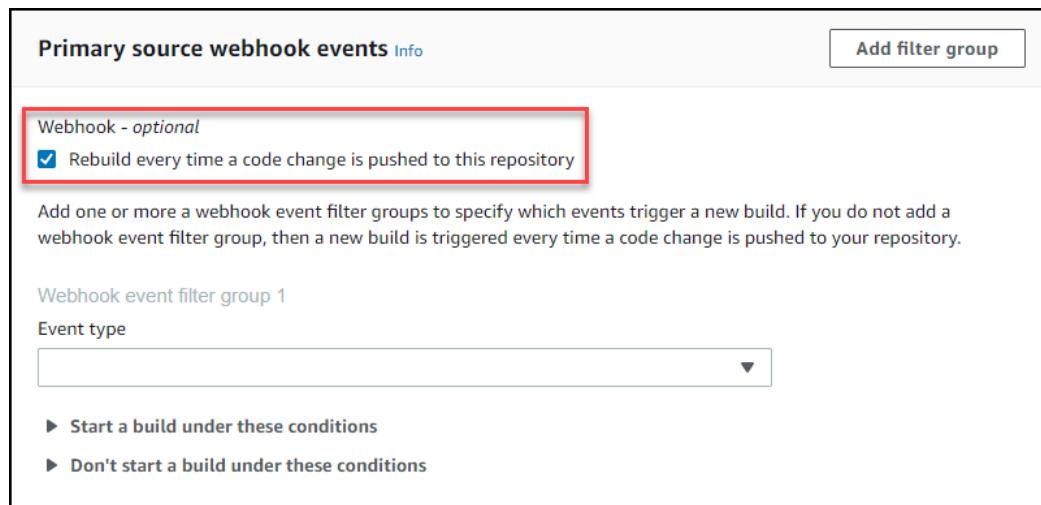
This 3rd party vendor has not provided a privacy policy or terms of use. Atlassian's Privacy Policy is not applicable to the use of this App.

Grant access Cancel

5. 选择 Use a repository in my account (在我的账户中使用存储库)。如果您使用公有 Bitbucket 存储库，则无法使用 Webhook。
6. 在 Primary source webhook events (主要源 Webhook 事件) 中，选择 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新生成)。仅当您已选择 Repository in my Bitbucket account (我的 Bitbucket 账户中的存储库) 时才选中此复选框。

Note

如果由 Bitbucket Webhook 触发了构建，则将忽略 Report build status (报告生成状态) 设置。生成状态始终发送到 Bitbucket。



The screenshot shows the 'Primary source webhook events' configuration page. At the top, there is a button labeled 'Add filter group'. Below it, a section titled 'Webhook - optional' contains a checked checkbox labeled 'Rebuild every time a code change is pushed to this repository'. A red box highlights this checkbox. Below this section, there is a note: 'Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.' Underneath, there is a section for 'Webhook event filter group 1' with a dropdown menu for 'Event type' and two options: 'Start a build under these conditions' and 'Don't start a build under these conditions'.

7. 选择项目的其他设置。有关源提供商选项和设置的更多信息，请参阅[Choose source provider](#)。
8. 选择 Create build project (创建构建项目)。在 Review (审核) 页面上，选择 Start build (启动构建) 以运行构建。

## 使用 Bitbucket Webhook 触发构建

对于使用 Bitbucket Webhook 的项目，AWS CodeBuild 在 Bitbucket 存储库检测到源代码中的更改时创建构建。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格上，选择 Build projects (生成项目)，然后选择项目以与 Bitbucket 存储库和 Webhook 关联。有关创建 BitBucketWebHook 项目的信息，请参阅 [the section called “创建将 Bitbucket 作为源存储库的构建项目并启用 Webhook” \(p. 75\)](#)。
3. 在您项目的 Bitbucket 存储库中更改一些代码。
4. 在 Bitbucket 存储库上创建拉取请求。有关更多信息，请参阅[发出拉取请求](#)。
5. 在 Bitbucket Webhook 页面上，选择 View request (查看请求) 以查看最新事件的列表。
6. 选择 View details (查看详细信息) 以查看 CodeBuild 返回的响应的详细信息。其内容如下所示：

```
"response": "Webhook received and build started: https://us-east-1.console.aws.amazon.com/codebuild/home..."  
"statusCode": 200
```

7. 导航到 Bitbucket 拉取请求页面以查看构建的状态。

## 使用 CodeBuild 构建徽章示例

AWS CodeBuild 现在支持使用构建徽章，该徽章提供一个动态生成的可嵌入映像（徽章），用以显示项目的最新构建状态。可通过为您的 CodeBuild 项目生成的公开可用的 URL 访问此映像。这将允许任何人查看 CodeBuild 项目的状态。构建徽章不包含任何安全信息，因此它们无需身份验证。

### 创建已启用构建徽章的构建项目（控制台）

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
3. 在 Create build project (创建构建项目) 页面上的 Project configuration (项目配置) 中，输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。您也可以包含构建项目的可选描述来帮助其他用户了解此项目的用途。
4. 在 Source (源) 中，对于 Source provider (源提供商)，选择源提供商类型，然后执行以下操作之一：

#### Note

CodeBuild 不支持 Amazon S3 源提供程序随附的构建徽章。由于 AWS CodePipeline 使用 Amazon S3 进行构件传输，因此对于作为在 CodePipeline 中创建的管道的一部分的构建项目，不支持构建徽章。

- 如果您选择了 CodeCommit，那么对于 Repository (存储库)，请选择存储库的名称。选择 Enable build badge (启用生成徽章)，以使您的项目的生成状态可见且可嵌入。
- 如果您选择了 GitHub，请按照说明与 GitHub 连接（或重新连接）。在 GitHub Authorize application (授权应用程序) 页面上，对于 Organization access (组织访问权限)，选择您希望 AWS CodeBuild 能够访问的每个存储库旁边的 Request access (请求访问权限)。选择 Authorize application (授权应用程序) 后，返回 AWS CodeBuild 控制台，对于 Repository (存储库)，选择包含源代码的存储库的名称。选择 Enable build badge (启用生成徽章)，以使您的项目的生成状态可见且可嵌入。
- 如果您选择了 Bitbucket，请按照说明与 Bitbucket 连接（或重新连接）。在 Bitbucket Confirm access to your account 页面上，对于 Organization access，选择 Grant access。选择 Grant access (授予访问权限) 后，返回 AWS CodeBuild 控制台，对于 Repository (存储库)，选择包含源代码的存储库的名称。选择 Enable build badge (启用生成徽章)，以使您的项目的生成状态可见且可嵌入。

#### Important

更新项目源可能会影响项目构建徽章的准确性。

5. 在 Environment (环境) 中：

对于 Environment image (环境映像)，执行下列操作之一：

- 要使用由 AWS CodeBuild 托管的 Docker 映像，请选择 Managed image (托管映像)，然后从 Operating system (操作系统)、Runtime(s) (运行时) 和 Image (映像) 以及 Image version (映像版本) 中进行相应选择。从 Environment type (环境类型) 中进行选择（如果可用）。
- 要使用其他 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。如果您针对 External registry URL (外部注册表 URL) 选择 Other registry (其他注册表)，请在 Docker Hub 中按照格式 `docker repository/docker image name` 输入 Docker 映像的名称和标签。如果您选择 Amazon ECR，请使用 Amazon ECR repository (Amazon ECR 存储库) 和 Amazon ECR image (Amazon ECR 映像) 在您的 AWS 账户中选择 Docker 映像。
- 要使用私有 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。对于 Image registry (映像注册表)，选择 Other registry (其他注册表)，然后输入您的私有 Docker 映像的凭证的 ARN。凭证必须由 Secrets Manager 创建。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的[什么是 AWS Secrets Manager](#)？

6. 在 Service role (服务角色) 中，执行下列操作之一：

- 如果您没有 CodeBuild 服务角色，请选择 New service role (新建服务角色)。在 Role name (角色名称) 中，为新角色输入一个名称。
- 如果您有 CodeBuild 服务角色，请选择 Existing service role (现有服务角色)。在 Role ARN (角色 ARN) 中，选择服务角色。

#### Note

当您使用控制台来创建或更新生成项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

7. 对于 Buildspec，执行以下操作之一：

- 选择 Use a buildspec file (使用 buildspec 文件) 以在源代码根目录中使用 buildspec.yml 文件。
- 选择 Insert build commands (插入构建命令) 以使用控制台插入构建命令。

有关更多信息，请参见 [构建规范参考 \(p. 136\)](#)。

8. 在 Artifacts (构件) 中，对于 Type (类型)，执行以下操作之一：

- 如果您不想创建构建输出项目，请选择 No artifacts (无构件)。
- 要将构建输出存储在 S3 存储桶中，请选择 Amazon S3，然后执行以下操作：
  - 如果要将项目名称用于构建输出 ZIP 文件或文件夹，请将 Name (名称) 留空。否则，请输入名称。默认情况下，构件名称是项目名称。如果您要使用其他名称，请在构件名称框中输入该名称。如果您要输出 ZIP 文件，请包含 zip 扩展名。
  - 对于 Bucket name (存储桶名称)，请选择输出存储桶的名称。
  - 如果您在此过程的前面部分选择了 Insert build commands (插入构建命令)，对于 Output files (输出文件)，请输入构建 (该构建要放到构建输出 ZIP 文件或文件夹中) 中的文件位置。对于多个位置，使用逗号分开各个位置 (appspec.yml, target/my-app.jar)。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#) 中 files 的描述。

9. 展开 Additional configuration (其他配置) 并根据需要选择选项。

10. 选择 Create build project (创建构建项目)。在 Review (审核) 页面上，选择 Start build (启动构建) 以运行构建。

## 创建已启用构建徽章的构建项目 (CLI)

有关创建构建项目的信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#)。包括在您的 AWS CodeBuild 项目，您必须指定 `badgeEnabled` 有一个值 `true`。

## 访问您的 AWS CodeBuild 构建徽章

您可以使用 AWS CodeBuild 控制台或 AWS CLI 访问构建徽章。

- 在 CodeBuild 控制台中，在生成项目列表中的 Name (名称) 列，选择与生成项目相对应的链接。在 建立项目: `project-name` 页面，配置，选择 复制徽章URL。有关更多信息，请参阅 [查看构建项目的详细信息 \(控制台\) \(p. 213\)](#)。)
- 在 AWS CLI 中，运行 `batch-get-projects` 命令。构建徽章 URL 包含在输出的项目环境详细信息部分中。有关更多信息，请参阅 [查看构建项目的详细信息 \(AWS CLI\) \(p. 213\)](#)。)

#### Important

构建徽章请求URL适用于默认分支，但您可以指定您用于运行构建的源存储库中的任何分支。

## 发布您的 CodeBuild 构建徽章

您可以将生成徽章请求 URL 包含在您的首选存储库（例如，GitHub 或 CodeCommit）中的 markdown 文件中以显示最新生成的状态。

示例 markdown 代码：

```
![Build Status](https://codebuild.us-east-1.amazonaws.com/badges?uuid=...&branch=main)
```

## CodeBuild 徽章状态

- PASSING 给定分支上的最新构建已传递。
- FAILING 给定分支上的最新构建已超时、失败、出现故障或停止。
- IN\_PROGRESS 给定分支上的最新构建正在进行中。
- UNKNOWN 项目尚未为给定分支运行构建或根本未运行。此外，构建徽章功能可能已禁用。

## 适用于 CodeBuild 的构建通知示例

Amazon CloudWatch Events 具有对 AWS CodeBuild 的内置支持。CloudWatch Events 是描述您的 AWS 资源中的变化的系统事件流。利用 CloudWatch Events，您可以写入声明性规则以将相关事件与要执行的自动操作关联。每当构建成功、失败、从一个构建阶段转到另一个构建阶段或出现这些事件的任意组合时，本示例都会使用 Amazon CloudWatch Events 和 Amazon Simple Notification Service (Amazon SNS) 向订阅者发送构建通知。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon CloudWatch 和 Amazon SNS 相关的操作收取的费用。有关更多信息，请参阅 [CodeBuild 定价](#)、[Amazon CloudWatch 定价](#) 和 [Amazon SNS 定价](#)。

## 运行示例

要运行此示例，请执行以下操作：

1. 如果您已在 Amazon SNS 中设置并订阅用于此示例的主题，请跳至第 4 步。或者，如果您通过 IAM 用户而不是 AWS 根账户或 IAM 管理员用户来使用 Amazon SNS，请向用户（或与用户关联的 IAM 组）添加以下语句（在 **### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之间）。（建议不使用 AWS 根账户。）此语句可用于查看、创建、订阅和测试向 Amazon SNS 中的主题发送通知的情况。为了简洁起见，也为了帮您查找添加语句的位置，此处使用了省略号（...）。请勿删除任何语句，也不要将这些省略号键入现有策略中。

```
{  
  "Statement": [  
    "### BEGIN ADDING STATEMENT HERE ###"  
    {  
      "Action": [  
        "sns:CreateTopic",  
        "sns:GetTopicAttributes",  
        "sns>List*",  
        "sns:Publish",  
        "sns:SetTopicAttributes",  
        "sns:Subscribe"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

```
        "Effect": "Allow"
},
### END ADDING STATEMENT HERE ####
...
],
"Version": "2012-10-17"
}
```

#### Note

修改该策略的 IAM 实体必须拥有在 IAM 中修改策略的权限。

有关更多信息，请参阅[编辑客户托管策略](#)或 IAM 用户指南 的[使用内联策略（控制台）](#)中的“编辑或删除组、用户或角色的内联策略”部分。

2. 在 Amazon SNS 中创建或标识主题。AWS CodeBuild 将使用 CloudWatch Events 通过 Amazon SNS 向该主题发送构建通知。

要创建主题，请执行以下操作：

1. 打开 Amazon SNS 控制台：<https://console.aws.amazon.com/sns>。
2. 选择 Create topic。
3. 在 Create new topic (创建新主题) 对话框中，为 Topic name (主题名称) 输入主题的名称（例如 **CodeBuildDemoTopic**）。(如果您选择了其他名称，请用该名称替换掉本示例中对应的名称。)
4. 选择 Create topic。
5. 在 Topic details: CodeBuildDemoTopic (主题详细信息: CodeBuildDemoTopic) 页面上，复制 Topic ARN (主题 ARN) 值。在下一个步骤中，您需要用到此值。

# Topic detail

Publish to topic

Topic ARN

Topic owner

Region

Display name

有关更多信息，请参阅 Amazon SNS 开发人员指南 中的[创建主题](#)。

3. 为一个或多个收件人订阅主题以接收电子邮件通知。

为收件人订阅主题：

1. 使用上一步中打开的 Amazon SNS 控制台，在导航窗格中，选择 Subscriptions (订阅)，然后选择 Create subscription (创建订阅)。
2. 在 Create subscription (创建订阅) 中，对于 Topic ARN (主题 ARN)，粘贴您在上一步中复制的主题 ARN。
3. 对于协议，选择电子邮件。
4. 对于 Endpoint (终端节点)，输入收件人的完整电子邮件地址。

Create subscription

Topic A

Protocol

Endpoint

5. 选择 Create Subscription。
  6. Amazon SNS 向收件人发送订阅确认电子邮件。要开始接收电子邮件通知，收件人必须在订阅确认电子邮件中选择 Confirm subscription 链接。在收件人单击该链接后，如果成功订阅，Amazon SNS 将在收件人的 Web 浏览器中显示一条确认消息。
- 有关更多信息，请参阅 Amazon SNS 开发人员指南 中的[订阅主题](#)。
4. 如果您通过 IAM 用户而不是 AWS 根账户或 IAM 管理员用户来使用 CloudWatch Events，请向用户（或与用户关联的 IAM 组）添加以下语句（在 **### BEGIN ADDING STATEMENT HERE ###** 和 **### END ADDING STATEMENT HERE ###** 之间）。（建议不使用 AWS 根账户。）此语句用于允许用户使用 CloudWatch Events。为了简洁起见，也为了帮您查找添加语句的位置，此处使用了省略号（...）。请勿删除任何语句，也不要将这些省略号键入现有策略中。

```
{  
    "Statement": [  
        "### BEGIN ADDING STATEMENT HERE ###"  
        {  
            "Action": [  
                "events:*",  
                "iam:PassRole"  
            ],  
            "Resource": "*",  
            "Effect": "Allow"  
        },  
        "### END ADDING STATEMENT HERE ###"  
        ...  
    ],  
    "Version": "2012-10-17"  
}
```

#### Note

修改该策略的 IAM 实体必须拥有在 IAM 中修改策略的权限。

有关更多信息，请参阅[编辑客户托管策略](#)或 IAM 用户指南 的[使用内联策略（控制台）](#)中的“编辑或删除组、用户或角色的内联策略”部分。

5. 在 CloudWatch Events 中创建规则。为此，请通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch>。
6. 在导航窗格中的 Events 下，选择 Rules，然后选择 Create rule。
7. 在 Step 1: Create rule page (步骤 1: 创建规则页面) 上，Event Pattern (事件模式) 和 Build event pattern to match events by service (构建事件模式以按服务匹配事件) 应已选中。
8. 对于 Service Name (服务名称)，选择 CodeBuild。对于 Event Type (事件类型)，All Events (所有事件) 应已选中。
9. Event Pattern Preview (事件模式预览) 中应显示以下代码：

```
{  
    "source": [  
        "aws.codebuild"  
    ]  
}
```

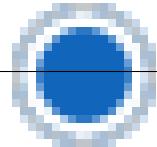
比较您的结果：

# Step 1: Create

Create rules to invoke Target

Event Source

Build or customize an Event Source  
Schedule to invoke Target



API 版本 2016-10-06  
86

Event Pattern



10. 选择 Edit (编辑) , 并将 Event Pattern Preview (事件模式预览) 中的代码替换为以下两个规则模式之一。

每当一个构建开始或完成时 , 第一个规则模式就会为 AWS CodeBuild 中的指定构建项目触发一个事件。

```
{  
    "source": [  
        "aws.codebuild"  
    ],  
    "detail-type": [  
        "CodeBuild Build State Change"  
    ],  
    "detail": {  
        "build-status": [  
            "IN_PROGRESS",  
            "SUCCEEDED",  
            "FAILED",  
            "STOPPED"  
        ],  
        "project-name": [  
            "my-demo-project-1",  
            "my-demo-project-2"  
        ]  
    }  
}
```

在前面的规则中 , 根据需要更改以下代码。

- 要在每次构建开始或完成时触发事件 , 请保留 build-status 数组中显示的所有值 , 或删除整个 build-status 数组。
- 要仅在构建完成时触发事件 , 请从 build-status 阵列中删除 IN\_PROGRESS。
- 要仅在构建开始时触发事件 , 请从 build-status 阵列中删除除 IN\_PROGRESS 以外的所有值。
- 要为所有构建项目触发事件 , 请删除整个 project-name 阵列。
- 要仅为单个构建项目触发事件 , 请在 project-name 阵列中指定每个构建项目的名称。

每当构建从一个构建阶段转到另一个构建阶段时 , 第二个规则模式将为 AWS CodeBuild 中的指定构建项目触发一个事件。

```
{  
    "source": [  
        "aws.codebuild"  
    ],  
    "detail-type": [  
        "CodeBuild Build Phase Change"  
    ],  
    "detail": {  
        "completed-phase": [  
            "SUBMITTED",  
            "PROVISIONING",  
            "DOWNLOAD_SOURCE",  
            "INSTALL",  
            "PRE_BUILD",  
            "BUILD",  
            "POST_BUILD",  
            "UPLOAD_ARTIFACTS",  
            "FINALIZING"  
        ],  
        "completed-phase-status": [  
            "TIMED_OUT",  
            "STOPPED",  
            "CANCELED"  
        ]  
    }  
}
```

```
    "FAILED",
    "SUCCEEDED",
    "FAULT",
    "CLIENT_ERROR"
],
"project-name": [
    "my-demo-project-1",
    "my-demo-project-2"
]
}
}
```

在前面的规则中，根据需要更改以下代码。

- 要为每个构建阶段更改触发一个事件（这可以为每个构建发送最多 9 条通知），请保留 completed-phase 数组中显示的所有值，或删除整个 completed-phase 数组。
- 要仅针对单个构建阶段更改触发事件，请删除 completed-phase 阵列中您不希望为其触发事件的每个构建阶段的名称。
- 要针对所有构建阶段状态更改触发事件，请保留 completed-phase-status 阵列中显示的所有值，或删除整个 completed-phase-status 阵列。
- 要仅针对单个构建阶段状态更改触发事件，请删除 completed-phase-status 阵列中您不希望对其触发事件的每个构建阶段状态的名称。
- 要为所有构建项目触发事件，请删除 project-name 阵列。
- 要为单个构建项目触发事件，请在 project-name 阵列中指定每个构建项目的名称。

Note

如果要同时为构建状态更改和构建阶段更改触发事件，则必须创建两个单独的规则：一个针对构建状态更改，另一个针对构建阶段更改。如果您尝试将两个规则合并为一个规则，则合并后的规则可能产生意外结果或停止协作。

替换完代码后，选择 Save。

- 对于 Targets，选择 Add target。
- 在目标列表中，选择 SNS 主题。
- 对于 Topic，选择您之前标识或创建的主题。
- 展开配置输入，然后选择输入转换器。
- 在 Input Path (输入路径) 框中，输入以下输入路径之一。

对于 detail-type 值为 CodeBuild Build State Change 的规则，输入以下内容。

```
{"build-id": "$.detail.build-id", "project-name": "$.detail.project-name", "build-status": "$.detail.build-status"}
```

对于 detail-type 值为 CodeBuild Build Phase Change 的规则，输入以下内容。

```
{"build-id": "$.detail.build-id", "project-name": "$.detail.project-name", "completed-phase": "$.detail.completed-phase", "completed-phase-status": "$.detail.completed-phase-status"}
```

要获取其他类型的信息，请参阅[构建通知输入格式参考 \(p. 93\)](#)。

- 在 Input Template (输入模板) 框中，输入以下输入模板之一。

对于 detail-type 值为 CodeBuild Build State Change 的规则，输入以下内容。

API 版本 2016-10-06

```
"Build '<build-id>' for build project '<project-name>' has reached the build status of '<build-status>'."
```

对于 detail-type 值为 CodeBuild Build Phase Change 的规则，输入以下内容。

```
"Build '<build-id>' for build project '<project-name>' has completed the build phase of '<completed-phase>' with a status of '<completed-phase-status>'."
```

将您迄今为止的结果与以下内容进行比较，其中显示了一条 detail-type 值为 CodeBuild Build State Change 的规则：

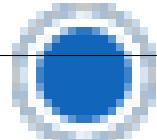
# Step 1: Create a target

Create rules to invoke Targets

## Event Source

Build or customize an Event Source

Schedule to invoke Target



API 版本 2016-10-06  
90

Event Pattern



17. 选择 Configure details。
18. 在 Step 2: Configure rule details (步骤 2: 配置规则详细信息) 页面上，输入名称和可选描述。对于 State (状态)，将 Enabled (已启用) 保持选中状态。

将您迄今为止的结果与以下屏幕截图进行比较：

# Step 2: Con

## Rule definition

Name\*

CodeB

Description

19. 选择 Create rule。
  20. 按照[直接运行 CodeBuild \(p. 367\)](#)中的步骤操作，创建构建项目、运行构建并查看构建信息。
  21. 确认 CodeBuild 立即成功发送构建通知。例如，检查您的收件箱中现在是否有构建通知电子邮件。

要更改规则的行为，请在 CloudWatch 控制台中，选择要更改的规则，然后依次选择 Actions (操作) 和 Edit (编辑)。对该规则进行更改，选择 Configure details (配置详细信息)，然后选择 Update rule (更新规则)。

要停止使用规则发送构建通知，请在 CloudWatch 控制台中，选择要停止使用的规则，然后依次选择 Actions (操作) 和 Disable (禁用)。

要删除整个规则，请在 CloudWatch 控制台中，选择要删除的规则，然后依次选择 Actions (操作) 和 Delete (删除)。

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
  - 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
  - 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

## 构建通知输入格式参考

CloudWatch 以 JSON 格式发送通知。

构建状态更改通知使用以下格式：

```
"initiator": "MyCodeBuildDemoUser",
"build-start-time": "Sep 1, 2017 4:12:29 PM",
"source": {
    "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
    "type": "S3"
},
"logs": {
    "group-name": "/aws/codebuild/my-sample-project",
    "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
    "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-edf953571bEX"
},
"phases": [
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:12:29 PM",
        "end-time": "Sep 1, 2017 4:12:29 PM",
        "duration-in-seconds": 0,
        "phase-type": "SUBMITTED",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:12:29 PM",
        "end-time": "Sep 1, 2017 4:13:05 PM",
        "duration-in-seconds": 36,
        "phase-type": "PROVISIONING",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:05 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 4,
        "phase-type": "DOWNLOAD_SOURCE",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 0,
        "phase-type": "INSTALL",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 0,
        "phase-type": "PRE_BUILD",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 70,
        "phase-type": "BUILD",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 0,
        "phase-type": "POST_BUILD",
        "phase-status": "SUCCEEDED"
    }
]
```

```
        "duration-in-seconds": 0,
        "phase-type": "POST_BUILD",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 0,
        "phase-type": "UPLOAD_ARTIFACTS",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:14:21 PM",
        "end-time": "Sep 1, 2017 4:14:26 PM",
        "duration-in-seconds": 4,
        "phase-type": "FINALIZING",
        "phase-status": "SUCCEEDED"
    },
    {
        "start-time": "Sep 1, 2017 4:14:26 PM",
        "phase-type": "COMPLETED"
    }
]
},
"current-phase": "COMPLETED",
"current-phase-context": "[ ]",
"version": "1"
}
}
```

构建阶段更改通知使用以下格式：

```
{
    "version": "0",
    "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
    "detail-type": "CodeBuild Build Phase Change",
    "source": "aws.codebuild",
    "account": "123456789012",
    "time": "2017-09-01T16:14:21Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-c340-456a-9166-edf953571bEX"
    ],
    "detail": {
        "completed-phase": "COMPLETED",
        "project-name": "my-sample-project",
        "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-c340-456a-9166-edf953571bEX",
        "completed-phase-context": "[ ]",
        "additional-information": {
            "artifact": {
                "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
                "sha256sum": "6ccc2ae1df9d15ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
                "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-artifact.zip"
            },
            "environment": {
                "image": "aws/codebuild/standard:4.0",
                "privileged-mode": false,
                "compute-type": "BUILD_GENERAL1_SMALL",
                "type": "LINUX_CONTAINER",
                "environment-variables": []
            }
        }
    }
}
```

```
},
"timeout-in-minutes": 60,
"build-complete": true,
"initiator": "MyCodeBuildDemoUser",
"build-start-time": "Sep 1, 2017 4:12:29 PM",
"source": {
    "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
    "type": "S3"
},
"logs": {
    "group-name": "/aws/codebuild/my-sample-project",
    "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
    "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-edf953571bEX"
},
"phases": [
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:12:29 PM",
        "end-time": "Sep 1, 2017 4:12:29 PM",
        "duration-in-seconds": 0,
        "phase-type": "SUBMITTED",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:12:29 PM",
        "end-time": "Sep 1, 2017 4:13:05 PM",
        "duration-in-seconds": 36,
        "phase-type": "PROVISIONING",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:05 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 4,
        "phase-type": "DOWNLOAD_SOURCE",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 0,
        "phase-type": "INSTALL",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:13:10 PM",
        "duration-in-seconds": 0,
        "phase-type": "PRE_BUILD",
        "phase-status": "SUCCEEDED"
    },
    {
        "phase-context": [],
        "start-time": "Sep 1, 2017 4:13:10 PM",
        "end-time": "Sep 1, 2017 4:14:21 PM",
        "duration-in-seconds": 70,
        "phase-type": "BUILD",
        "phase-status": "SUCCEEDED"
    }
]
```

```
"phase-context": [],
"start-time": "Sep 1, 2017 4:14:21 PM",
"end-time": "Sep 1, 2017 4:14:21 PM",
"duration-in-seconds": 0,
"phase-type": "POST_BUILD",
"phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:14:21 PM",
  "end-time": "Sep 1, 2017 4:14:21 PM",
  "duration-in-seconds": 0,
  "phase-type": "UPLOAD_ARTIFACTS",
  "phase-status": "SUCCEEDED"
},
{
  "phase-context": [],
  "start-time": "Sep 1, 2017 4:14:21 PM",
  "end-time": "Sep 1, 2017 4:14:26 PM",
  "duration-in-seconds": 4,
  "phase-type": "FINALIZING",
  "phase-status": "SUCCEEDED"
},
{
  "start-time": "Sep 1, 2017 4:14:26 PM",
  "phase-type": "COMPLETED"
}
],
"completed-phase-status": "SUCCEEDED",
"completed-phase-duration-seconds": 4,
"version": "1",
"completed-phase-start": "Sep 1, 2017 4:14:21 PM",
"completed-phase-end": "Sep 1, 2017 4:14:26 PM"
}
}
```

## 在 CodeBuild 中使用 AWS CLI 示例创建测试报告

您在 buildspec 文件中指定的测试将在构建期间运行。此示例演示如何在 CodeBuild 中使用 AWS CLI 将测试合并到构建中。您可以使用 JUnit 创建单元测试，也可以使用其他工具创建配置测试。然后，您可以评估测试结果以修复问题或优化您的应用程序。

您可以使用 CodeBuild API 或 AWS CodeBuild 控制台访问测试结果。此示例演示如何配置报告以便将其测试结果导出到 S3 存储桶。

### 主题

- [Prerequisites \(p. 97\)](#)
- [创建报告组 \(p. 98\)](#)
- [使用报告组配置项目 \(p. 99\)](#)
- [运行和查看报告结果 \(p. 100\)](#)

## Prerequisites

- 创建您的测试用例。编写此示例时假设您有要包含在示例测试报告中的测试用例。您可以在 buildspec 文件中指定测试文件的位置。

支持以下测试报告文件格式：

- Cucumber JSON

- JUnit XML
- NUnit XML
- NUnit3XML(NUnit3XML)
- TestNG XML
- Visual Studio TRX

使用任何测试框架创建测试用例，这些测试框架可以采用任何一种格式创建报告文件（例如 Surefire JUnit 插件，TestNG 或 Cucumber）。

- 创建 S3 存储桶并记下其名称。有关详细信息，请参阅 [如何创建S3桶子？](#) 在 Amazon S3 用户指南。
- 创建一个 IAM 角色并记下其 ARN。创建构建项目时，您需要 ARN。
- 如果您的角色没有以下权限，请添加它们。

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "*"  
    ],  
    "Action": [  
        "codebuild>CreateReportGroup",  
        "codebuild>CreateReport",  
        "codebuild:UpdateReport",  
        "codebuild:BatchPutTestCases"  
    ]  
}
```

有关更多信息，请参阅 [测试报告操作的权限 \(p. 293\)](#)。)

## 创建报告组

1. 创建一个名为的文件 CreateReportGroupInput.json。
2. 在 S3 存储桶中创建要将测试结果导出到的文件夹。
3. 将以下内容复制到 CreateReportGroupInput.json...对于 bucket，使用S3桶的名称。对于 path，请输入 S3 存储桶中文件夹的路径。

```
{  
    "name": "report-name",  
    "type": "TEST",  
    "exportConfig": {  
        "exportConfigType": "S3",  
        "s3Destination": {  
            "bucket": "bucket-name",  
            "path": "path-to-folder",  
            "packaging": "NONE"  
        }  
    }  
}
```

4. 在包含“目录”的目录中运行以下命令 CreateReportGroupInput.json...对于 region，指定您的 AWS区域（例如，us-east-2）。

```
aws codebuild create-report-group \  
    --cli-input-json file://CreateReportGroupInput.json \  
    --region your-region
```

输出如下所示：记下一个关于 reportGroup...创建使用此报表组的项目时，您将使用它。

```
{  
    "reportGroup": {  
        "arn": "arn:aws:codebuild:us-west-2:123456789012:report-group/report-name",  
        "name": "report-name",  
        "type": "TEST",  
        "exportConfig": {  
            "exportConfigType": "S3",  
            "s3Destination": {  
                "bucket": "s3-bucket-name",  
                "path": "folder-path",  
                "packaging": "NONE",  
                "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3"  
            }  
        },  
        "created": 1570837165.885,  
        "lastModified": 1570837165.885  
    }  
}
```

## 使用报告组配置项目

要运行报告，您首先创建配置有报告组的 CodeBuild 构建项目。为报告组指定的测试用例将在您运行构建时运行。

1. 创建一个名为 `buildspec.yml` 的 `buildspec` 文件。
2. 使用以下 YAML 作为 `buildspec.yml` 文件的模板。请务必包含运行测试的命令。在 `reports` 部分中，指定包含测试用例结果的文件。这些文件存储您可以使用 CodeBuild 访问的测试结果。它们在创建后 30 天过期。这些文件与您导出到 S3 存储桶的原始测试用例结果文件不同。

```
version: 0.2  
phases:  
  install:  
    runtime-versions:  
      java: openjdk8  
  build:  
    commands:  
      - echo Running tests  
      - enter commands to run your tests  
  
  reports:  
    report-name-or-arn: #test file information  
    files:  
      - 'test-result-files'  
    base-directory: 'optional-base-directory'  
    discard-paths: false #do not remove file paths from test result files
```

### Note

您还可以为尚未创建的报告组指定名称，而不是使用现有报告组的 ARN。如果您指定名称（而不是 ARN），CodeBuild 在运行构建时创建报告组。其名称包含您的项目名称和您在构建规范文件中指定的名称，格式如下：。`project-name-report-group-name`...有关详细信息，请参阅 [创建测试报告 \(p. 276\)](#) 和 [报告组命名 \(p. 283\)](#)。

3. 创建名为 `project.json`...此文件包含对 `create-project` 命令。
4. 将以下 JSON 复制到 `project.json`...对于 `source`，输入包含源文件的存储库的类型和位置。对于 `serviceRole`，请指定您正在使用的角色的 ARN。

```
{
```

```
"name": "test-report-project",
"description": "sample-test-report-project",
"source": {
    "type": "your-repository-type",
    "location": "https://github.com/your-repository/your-folder"
},
"artifacts": {
    "type": "NO_ARTIFACTS"
},
"cache": {
    "type": "NO_CACHE"
},
"environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:4.0",
    "computeType": "small"
},
"serviceRole": "arn:aws:iam:your-aws-account-id:role/service-role/your-role-name"
}
```

- 在包含“目录”的目录中运行以下命令 project.json... 这会创建一个名为 test-project.

```
aws codebuild create-project \
--cli-input-json file://project.json \
--region your-region
```

## 运行和查看报告结果

在此部分中，您将运行之前创建的项目的构建。在构建过程中，CodeBuild 创建包含测试用例结果的报告。该报告包含在您指定的报告组中。

- 要启动构建，请运行以下命令。记下输出中显示的构建 ID。其格式为 `test-report>:build-id`。

```
aws codebuild start-build --project-name "test-project" --region your-region
```

- 运行以下命令以获取有关您的构建的信息，包括报告的 ARN。对于 `--ids`，请指定您的构建 ID。记下输出中的报告 ARN。

```
aws codebuild batch-get-builds \
--ids "build-id" \
--region your-region
```

- 运行以下命令以获取有关报告的详细信息。对于 `--report-group-arn`，请指定您的报告 ARN。

```
aws codebuild batch-get-reports \
--report-arns report-group-arn \
--region your-region
```

输出如下所示：此示例输出显示了成功、失败、跳过、导致错误或返回未知状态的测试数量。

```
{
  "reports": [
    {
      "status": "FAILED",
      "reportGroupArn": "report-group-arn",
      "name": "report-group-name",
      "created": 1573324770.154,
      "exportConfig": {
        "exportConfigType": "S3",
```

```
        "s3Destination": {
            "bucket": "your-s3-bucket",
            "path": "path-to-your-report-results",
            "packaging": "NONE",
            "encryptionKey": "encryption-key"
        }
    },
    "expired": 1575916770.0,
    "truncated": false,
    "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/name-of-build-project:2c254862-ddf6-4831-a53f-6839a73829c1",
    "type": "TEST",
    "arn": "report-arn",
    "testSummary": {
        "durationInNanoSeconds": 6657770,
        "total": 11,
        "statusCounts": {
            "FAILED": 3,
            "SKIPPED": 7,
            "ERROR": 0,
            "SUCCEEDED": 1,
            "UNKNOWN": 0
        }
    }
},
"reportsNotFound": []
}
```

4. 运行以下命令列出有关报告的测试用例的信息。对于 **--report-arn**，请指定报告的 ARN。对于可选 **--filter** 参数，您可以指定一个状态结果 ( SUCCEEDED , FAILED , SKIPPED , ERROR 或 UNKNOWN )。

```
aws codebuild describe-test-cases \
--report-arn report-arn \
--filter status=SUCCEEDED|FAILED|SKIPPED|ERROR|UNKNOWN \
--region your-region
```

输出如下所示：

```
{
    "testCases": [
        {
            "status": "FAILED",
            "name": "Test case 1",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        },
        {
            "status": "SUCCEEDED",
            "name": "Test case 2",
            "expired": 1575916770.0,
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        }
    ]
}
```

}

## 适用于 CodeBuild 的自定义映像示例中的 Docker

此示例通过使用 AWS CodeBuild 和自定义 Docker 构建映像 (Docker Hub 中的 `docker:dind`) 来构建和运行 Docker 映像。

要了解如何改用由支持 Docker 的 CodeBuild 提供的构建映像来构建 Docker 映像，请参阅我们的[Docker 示例 \(p. 104\)](#)。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS 和 CloudWatch Logs 相关的操作收取的费用。有关更多信息，请参阅[CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)和[Amazon CloudWatch 定价](#)。

### 主题

- [运行示例 \(p. 102\)](#)
- [目录结构 \(p. 103\)](#)
- [Files \(p. 103\)](#)
- [相关资源 \(p. 53\)](#)

## 运行示例

要运行此示例，请执行以下操作：

1. 按照本主题的“目录结构”和“文件”部分中的说明创建文件，然后将其上传到 S3 输入存储桶或者上传到 AWS CodeCommit、GitHub 或 Bitbucket 存储库。

### Important

请不要上传 `(root directory name)`，而只上传 `(root directory name)` 中的文件。  
如果您使用的是 S3 输入存储桶，请务必创建一个包含文件的 ZIP 文件，然后将其上传到输入存储桶。请不要将 `(root directory name)` 添加到 ZIP 文件中，而只添加 `(root directory name)` 中的文件。

2. 请按照[直接运行 AWS CodeBuild \(p. 367\)](#)中的步骤创建构建项目、运行构建并查看相关构建信息。

如果您使用 AWS CLI 创建构建项目，则 `create-project` 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{  
    "name": "sample-docker-custom-image-project",  
    "source": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-input-  
        bucket/DockerCustomImageSample.zip"  
    },  
    "artifacts": {  
        "type": "NO_ARTIFACTS"  
    },  
    "environment": {  
        "type": "LINUX_CONTAINER",  
        "image": "docker:dind",  
        "computeType": "BUILD_GENERAL1_SMALL",  
        "privilegedMode": true  
    },  
}
```

```
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

#### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

3. 要查看构建结果，请查看字符串的构建日志 Hello, World!...有关详细信息，请参阅[查看构建详细信息 \(p. 261\)](#).

## 目录结构

此示例假定有这样一个目录结构。

```
(root directory name)
|-- buildspec.yml
`-- Dockerfile
```

## Files

在此示例中使用的操作系统的基映像是 Ubuntu。此示例将使用这些文件。有关构建规范文件中引用的 OverlayFS 存储驱动程序的更多信息，请参阅 Docker 网站上的[使用 OverlayFS 存储驱动程序](#)。

`buildspec.yml (在 (root directory name))`

```
version: 0.2

phases:
  install:
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
        host=tcp://127.0.0.1:2375 --storage-driver=overlay2
        - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"
```

#### Note

如果基本操作系统是 Alpine Linux，请在 `buildspec.yml` 中向 `timeout` 添加 `-t` 参数：

```
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

`Dockerfile (在 (root directory name))`

```
FROM maven:3.3.9-jdk-8
RUN echo "Hello World"
```

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。

- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

## 适用于 CodeBuild 的 Docker 示例

该示例会生成一个 Docker 映像作为构建输出，然后将该 Docker 映像推送到 Amazon Elastic Container Registry (Amazon ECR) 映像存储库。您可以调整该示例以将 Docker 映像推送到 Docker Hub。有关更多信息，请参阅[调整示例以将映像推送到 Docker Hub \(p. 107\)](#)。

要了解如何使用自定义 Docker 构建映像来构建 Docker 映像 (Docker Hub 中的 `docker:dind`)，请参阅我们的[自定义映像示例中的 Docker \(p. 102\)](#)。

此示例参考 `golang:1.12` 进行了测试。

此示例使用新的多阶段 Docker 构建功能，该功能将生成一个 Docker 映像作为构建输出。然后，它将 Docker 映像推送到 Amazon ECR 映像存储库。多阶段 Docker 映像构建有助于减小最终 Docker 映像的大小。有关更多信息，请参阅[将多阶段构建和 Docker 结合使用](#)。

### Important

运行该示例可能会导致您的 AWS 账户产生相关费用。这些费用包括可能针对 AWS CodeBuild 和 AWS 资源以及与 Amazon S3、AWS KMS、CloudWatch Logs 和 Amazon ECR 相关的操作收取的费用。有关更多信息，请参阅[CodeBuild 定价](#)、[Amazon S3 定价](#)、[AWS Key Management Service 定价](#)、[Amazon CloudWatch 定价](#)和[Amazon Elastic Container Registry 定价](#)。

### 主题

- [运行示例 \(p. 104\)](#)
- [目录结构 \(p. 107\)](#)
- [文件 \(p. 107\)](#)
- [调整示例以将映像推送到 Docker Hub \(p. 107\)](#)
- [相关资源 \(p. 53\)](#)

## 运行示例

要运行此示例，请执行以下操作：

- 如果 Amazon ECR 中已存在要使用的映像存储库，请跳至第 3 步。或者，如果您通过 IAM 用户而不是 AWS 根账户或 IAM 管理员用户来使用 Amazon ECR，请向用户（或与用户关联的 IAM 组）添加该语句（在 `### BEGIN ADDING STATEMENT HERE ###` 和 `### END ADDING STATEMENT HERE ###` 之间）。建议不要使用 AWS 根账户。此语句允许创建用于存储 Docker 映像的 Amazon ECR 存储库。为了简洁起见，也为了帮您查找添加语句的位置，此处使用了省略号 (...)。请勿删除任何语句，也不要将这些省略号键入策略中。有关更多信息，请参阅 IAM 用户指南 中的[通过 AWS 管理控制台 使用内联策略](#)。

```
{  
  "Statement": [  
    "### BEGIN ADDING STATEMENT HERE ###"  
    {  
      "Action": [  
        "ecr:CreateRepository"  
      ],  
      "Resource": "*",  
      "Effect": "Allow"  
    }  
  ]  
}
```

```
},
    ##### END ADDING STATEMENT HERE #####
    ...
],
"Version": "2012-10-17"
}
```

#### Note

修改该策略的 IAM 实体必须拥有在 IAM 中修改策略的权限。

2. 在 Amazon ECR 中创建映像存储库。请务必在从中创建构建环境并运行构建的同一 AWS 区域中创建存储库。有关更多信息，请参阅 Amazon ECR 用户指南 中的[创建存储库](#)。该存储库的名称必须与您将在此过程的稍后部分指定的存储库名称相匹配，并以 IMAGE\_REPO\_NAME 环境变量的形式表示。
3. 将此语句（在 **##### BEGIN ADDING STATEMENT HERE #####** 和 **##### END ADDING STATEMENT HERE #####** 之间）添加到已附加到您的 AWS CodeBuild 服务角色的策略。CodeBuild 可使用此语句将 Docker 映像上传到 Amazon ECR 存储库。为了简洁起见，也为了帮您查找添加语句的位置，此处使用了省略号 (...)。请勿删除任何语句，也不要将这些省略号键入策略中。

```
{
    "Statement": [
        ##### BEGIN ADDING STATEMENT HERE #####
        {
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetAuthorizationToken",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        ##### END ADDING STATEMENT HERE #####
        ...
    ],
    "Version": "2012-10-17"
}
```

#### Note

修改该策略的 IAM 实体必须拥有在 IAM 中修改策略的权限。

4. 按照本主题的“目录结构”和“文件”部分中的说明创建文件，然后将其上传到 S3 输入存储桶或者上传到 AWS CodeCommit、GitHub 或 Bitbucket 存储库。

#### Important

请不要上传 (*root directory name*)，而只上传 (*root directory name*) 中的文件。  
如果您使用的是 S3 输入存储桶，请务必创建一个包含文件的 ZIP 文件，然后将其上传到输入存储桶。请不要将 (*root directory name*) 添加到 ZIP 文件中，而只添加 (*root directory name*) 中的文件。

5. 请按照[直接运行 CodeBuild \(p. 367\)](#) 中的步骤创建构建项目、运行构建并查看构建信息。

如果您使用控制台创建项目：

- 对于 Operating system (操作系统)，选择 Ubuntu。
- 对于 Runtime (运行时)，选择 Standard (标准)。
- 对于 Image (映像)，选择 aws/codebuild/standard:4.0。
- 由于您使用此构建来构建 Docker 权限，因此请选择 Privileged (特权)。

### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

e. 添加以下环境变量：

- AWS\_DEFAULT\_REGION，值为 *region-ID*
- AWS\_ACCOUNT\_ID，值为 *account-ID*
- 具有最新值的 IMAGE\_TAG
- IMAGE\_REPO\_NAME，值为 *Amazon ECR-repo-name*

如果您使用 AWS CLI 创建构建项目，则 create-project 命令的 JSON 格式输入可能与此类似。(请将占位符替换为您自己的值。)

```
{  
    "name": "sample-docker-project",  
    "source": {  
        "type": "S3",  
        "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"  
    },  
    "artifacts": {  
        "type": "NO_ARTIFACTS"  
    },  
    "environment": {  
        "type": "LINUX_CONTAINER",  
        "image": "aws/codebuild/standard:4.0",  
        "computeType": "BUILD_GENERAL1_SMALL",  
        "environmentVariables": [  
            {  
                "name": "AWS_DEFAULT_REGION",  
                "value": "region-ID"  
            },  
            {  
                "name": "AWS_ACCOUNT_ID",  
                "value": "account-ID"  
            },  
            {  
                "name": "IMAGE_REPO_NAME",  
                "value": "Amazon-ECR-repo-name"  
            },  
            {  
                "name": "IMAGE_TAG",  
                "value": "latest"  
            }  
        ],  
        "privilegedMode": true  
    },  
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",  
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"  
}
```

6. 确认 CodeBuild 已成功将 Docker 映像推送到存储库：

1. 通过以下网址打开 Amazon ECR 控制台：<https://console.aws.amazon.com/ecr/>。
2. 选择存储库名称。映像应在 Image tag (映像标签) 列中列出。

## 目录结构

此示例假定有这样一个目录结构。

```
(root directory name)
|-- buildspec.yml
`-- Dockerfile
```

## 文件

此示例将使用这些文件。

`buildspec.yml` (在 `(root directory name)`)

### Note

如果您使用的是 17.06 版本之前的 Docker 版本，请删除 `--no-include-email` 选项。

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
```

`Dockerfile` (在 `(root directory name)`)

```
FROM golang:1.12-alpine AS build
#Install git
RUN apk add --no-cache git
#Get the hello world package from a GitHub repository
RUN go get github.com/golang/example/hello
WORKDIR /go/src/github.com/golang/example/hello
# Build the project and send the output to /bin/HelloWorld
RUN go build -o /bin/HelloWorld

FROM golang:1.12-alpine
#Copy the build's output binary from the previous build container
COPY --from=build /bin/HelloWorld /bin/HelloWorld
ENTRYPOINT ["/bin/HelloWorld"]
```

## 调整示例以将映像推送到 Docker Hub

要将 Docker 映像推送到 Docker Hub 而非 Amazon ECR，请编辑此示例的代码。

### Note

如果您使用的是 17.06 版本之前的 Docker 版本，请删除 --no-include-email 选项。

1. 替换 buildspec.yml 文件中的这些特定于 Amazon ECR 的代码行：

```
...
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
...

```

利用这些特定于 Docker Hub 的代码行，可以：

```
...
  pre_build:
    commands:
      - echo Logging in to Docker Hub...
      # Type the command to log in to your Docker Hub account here.
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
...

```

2. 将编辑后的代码上传到 S3 输入存储桶或者上传到 AWS CodeCommit、GitHub 或 Bitbucket 存储库。

#### Important

请不要上传 (*root directory name*)，而只上传 (*root directory name*) 中的文件。  
如果您使用的是 S3 输入存储桶，请务必创建一个包含文件的 ZIP 文件，然后将其上传到输入存储桶。请不要将 (*root directory name*) 添加到 ZIP 文件中，而只添加 (*root directory name*) 中的文件。

3. 将 create-project 命令的 JSON 格式输入中的这些代码行替换为：

```
...
  "environmentVariables": [
    {
      "name": "AWS_DEFAULT_REGION",
      "value": "region-ID"
    },

```

```
{  
    "name": "AWS_ACCOUNT_ID",  
    "value": "account-ID"  
},  
{  
    "name": "IMAGE_REPO_NAME",  
    "value": "Amazon-ECR-repo-name"  
},  
{  
    "name": "IMAGE_TAG",  
    "value": "latest"  
}  
]  
...  
...
```

利用这些代码行，可以：

```
...  
    "environmentVariables": [  
        {  
            "name": "IMAGE_REPO_NAME",  
            "value": "your-Docker-Hub-repo-name"  
        },  
        {  
            "name": "IMAGE_TAG",  
            "value": "latest"  
        }  
    ]  
...
```

4. 请按照 [直接运行 CodeBuild \(p. 367\)](#) 中的步骤创建构建环境、运行构建并查看相关构建信息。
5. 确认 AWS CodeBuild 已成功将 Docker 映像推送到存储库。登录 Docker Hub，再转至存储库，然后选择 Tags 选项卡。latest 标签应包含最新的 Last Updated 值。

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

## CodeBuild 的 GitHub Enterprise Server 示例

AWS CodeBuild 支持将 GitHub Enterprise Server 作为源存储库。此示例演示在 GitHub Enterprise Server 存储库安装证书后，如何设置您的 CodeBuild 项目。它还演示了如何启用 Webhook，这样在每次代码更改推送到您的 GitHub Enterprise Server 存储库后，CodeBuild 都可以重新构建源代码。

## 先决条件

1. 为您的 CodeBuild 项目生成个人访问令牌。我们建议您创建一个 GitHub Enterprise 用户，并为该用户生成个人访问令牌。将它复制到您的剪贴板，以便在创建 CodeBuild 项目时使用。有关更多信息，请参阅 GitHub 帮助网站上的[为命令行创建个人访问令牌](#)。

在创建个人访问令牌时，请在定义中包含存储库范围。

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories

- 从 GitHub Enterprise Server 下载您的证书。CodeBuild 使用此证书与存储库建立可信 SSL 连接。

Linux/macOS 客户端：

从您的终端窗口中运行以下命令：

```
echo -n | openssl s_client -connect HOST:PORTNUMBER \
| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /folder/filename.pem
```

将命令中的占位符替换为以下值：

**HOST**。您的 GitHub Enterprise Server 存储库的 IP 地址。

**PORTNUMBER**。用于连接的端口号（例如，443）。

**folder**。下载证书的文件夹。

**filename**。证书文件的文件名。

Important

将证书另存为 .pem 文件。

Windows 客户端：

使用浏览器从 GitHub Enterprise Server 下载您的证书。要查看站点的证书详细信息，请选择挂锁图标。有关如何导出证书的信息，请参阅浏览器文档。

Important

将证书另存为 .pem 文件。

- 将您的证书文件上传到 S3 存储桶。有关如何创建 S3 存储桶的信息，请参阅[如何创建 S3 存储桶？](#)有关如何将对象上传到 S3 存储桶的信息，请参阅[如何将文件和文件夹上传至存储桶？](#)

Note

此存储桶必须与您的构建项目处在同一个 AWS 区域中。例如，如果您指示 CodeBuild 在美国东部（俄亥俄）区域运行生成任务，则存储桶也必须位于美国东部（俄亥俄）区域中。

## 创建将 GitHub Enterprise Server 作为源存储库的构建项目，并启用 Webhook（控制台）

- Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
- 如果显示 CodeBuild 信息页面，请选择 Create build project（创建构建项目）。否则，请在导航窗格上展开 Build（构建），然后依次选择 Build projects（构建项目）和 Create build project（创建构建项目）。
- 在 Create build project（创建构建项目）页面上的 Project configuration（项目配置）中，输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。您也可以包含构建项目的可选描述来帮助其他用户了解此项目的用途。 API 版本 2016-10-06

4. 在 Source (源) 的 Source provider (源提供商) 中，选择 GitHub Enterprise。

- 对于 Personal Access Token，粘贴您复制到剪贴板的令牌，然后选择 Save Token。在 Repository URL (存储库 URL) 中输入您的 GitHub Enterprise Server 存储库的 URL。

**Note**

您只需输入并保存一次个人访问令牌。未来所有 AWS CodeBuild 项目均会使用此令牌。

- 在 Repository URL (存储库 URL) 中，输入您的存储库的路径，包括存储库的名称。
- 展开 Additional configuration (其他配置)。
- 选择 Rebuild every time a code change is pushed to this repository (每次将代码推送到此存储库时都会重建) 以便每次将代码推送到此存储库时进行重建。
- 选择 Enable insecure SSL (启用不安全的 SSL)，在连接到您的 GitHub Enterprise Server 项目存储库时忽略 SSL 警告。

**Note**

建议您仅将 Enable insecure SSL (启用不安全的 SSL) 用于测试。它不应在生产环境中使用。

**Source**

Source 1 - Primary

Source provider

GitHub Enterprise

Repository URL

`https://<host-name>/<user-name>/<repository-name>`

**Disconnect GitHub Enterprise account**

▼ Additional configuration

Git clone depth, Insecure SSL

Git clone depth - *optional*

1

Webhook - *optional*

Rebuild every time a code change is pushed to this repository

Branch filter - *optional*

Enter a regular expression

Insecure SSL - *optional*

Enable this flag to ignore SSL warnings while connecting to project source.

Enable insecure SSL

5. 在 Environment (环境) 中：

对于 Environment image (环境映像)，执行下列操作之一：

- 要使用由 AWS CodeBuild 托管的 Docker 映像，请选择 Managed image (托管映像)，然后从 Operating system (操作系统)、Runtime(s) (运行时) 和 Image (映像) 以及 Image version (映像版本) 中进行相应选择。从 Environment type (环境类型) 中进行选择 (如果可用)。
- 要使用其他 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。如果您针对 External registry URL (外部注册表 URL) 选择 Other registry (其他注册表)，请在 Docker Hub 中按照格式 `docker repository/docker image name` 输入 Docker 映像的名称和标签。如果您选择 Amazon ECR，请使用 Amazon ECR repository (Amazon ECR 存储库) 和 Amazon ECR image (Amazon ECR 映像) 在您的 AWS 账户中选择 Docker 映像。
- 要使用私有 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。对于 Image registry (映像注册表)，选择 Other registry (其他注册表)，然后输入您的私有 Docker 映像的凭证的 ARN。凭证必须由 Secrets Manager 创建。有关更多信息，请参阅 AWS Secrets Manager 用户指南 中的 [什么是 AWS Secrets Manager?](#)

6. 在 Service role (服务角色) 中，执行下列操作之一：

- 如果您没有 CodeBuild 服务角色，请选择 New service role (新建服务角色)。在 Role name (角色名称) 中，为新角色输入一个名称。
- 如果您有 CodeBuild 服务角色，请选择 Existing service role (现有服务角色)。在 Role ARN (角色 ARN) 中，选择服务角色。

Note

当您使用控制台来创建或更新生成项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

7. 展开 Additional configuration (其他配置)。

如果要将 CodeBuild 与您的 VPC 结合使用：

- 对于 VPC，选择 CodeBuild 使用的 VPC ID。
- 对于 VPC Subnets (VPC 子网)，选择包含 CodeBuild 使用的资源的子网。
- 对于 VPC Security groups (VPC 安全组)，选择 CodeBuild 用来支持对 VPC 中资源的访问的安全组。

有关更多信息，请参阅 [将 AWS CodeBuild 与 Amazon Virtual Private Cloud 结合使用 \(p. 173\)](#)。

8. 对于 Buildspec，执行以下操作之一：

- 选择 Use a buildspec file (使用 buildspec 文件) 以在源代码根目录中使用 buildspec.yml 文件。
- 选择 Insert build commands (插入构建命令) 以使用控制台插入构建命令。

有关更多信息，请参见 [构建规范参考 \(p. 136\)](#)。

9. 在 Artifacts (构件) 中，对于 Type (类型)，执行以下操作之一：

- 如果您不想创建构建输出项目，请选择 No artifacts (无构件)。
- 要将构建输出存储在 S3 存储桶中，请选择 Amazon S3，然后执行以下操作：
  - 如果要将项目名称用于构建输出 ZIP 文件或文件夹，请将 Name (名称) 留空。否则，请输入名称。默认情况下，构件名称是项目名称。如果您要使用其他名称，请在构件名称框中输入该名称。如果您要输出 ZIP 文件，请包含 zip 扩展名。
  - 对于 Bucket name (存储桶名称)，请选择输出存储桶的名称。
  - 如果您在此过程的前面部分选择了 Insert build commands (插入构建命令)，对于 Output files (输出文件)，请输入构建 (该构建要放到构建输出 ZIP 文件或文件夹中) 中的文件位置。对于多个位置，

使用逗号分开各个位置 (`appspec.yml`, `target/my-app.jar`)。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#) 中 `files` 的描述。

- 对于 Cache type (缓存类型)，选择下列选项之一：

- 如果您不想使用缓存，请选择 No cache (无缓存)。
- 如果要使用 Amazon S3 缓存，请选择 Amazon S3，然后执行以下操作：
  - 对于 Bucket (存储桶)，选择存储缓存的 S3 存储桶的名称。
  - (可选) 对于 Cache path prefix (缓存路径前缀)，输入 Amazon S3 路径前缀。Cache path prefix (缓存路径前缀) 值类似于目录名称。它使您能够在存储桶的同一目录下存储缓存。

**Important**

请勿将一个尾斜杠 (/) 附加到路径前缀后面。

- 如果想要使用本地缓存，请选择 Local (本地)，然后选择一个或多个本地缓存模式。

**Note**

Docker 层缓存模式仅适用于 Linux。如果您选择该模式，您的项目必须在特权模式下运行。`ARM_CONTAINER` 和 `LINUX_GPU_CONTAINER` 环境类型以及 `BUILD_GENERAL1_2XLARGE` 计算类型不支持使用本地缓存。

使用缓存可节省大量构建时间，因为构建环境的可重用部分被存储在缓存中，并且可跨构建使用。有关在构建规范文件中指定缓存的信息，请参阅 [构建规范语法 \(p. 137\)](#)。有关缓存的更多信息，请参阅 [在 AWS CodeBuild 中构建缓存 \(p. 214\)](#)。

- 选择 Create build project (创建构建项目)。在生成项目页面上，选择 Start build (开始生成)。
- 如果您在 Source (源) 中启用了 Webhook，将出现 Create webhook (创建 Webhook) 对话框，其中显示 Payload URL (负载 URL) 和 Secret (密钥) 的值。

**Important**

Create webhook 对话框只出现一次。请复制负载 URL 和私有密钥。在 GitHub Enterprise Server 中添加 Webhook 时会用到它们。

如果您需要再次生成负载 URL 和私有密钥，必须首先删除 GitHub Enterprise Server 存储库中的 Webhook。在您的 CodeBuild 项目中，清除 Webhook 复选框，然后选择 Save (保存)。然后您可以在创建或更新 CodeBuild 项目时选中 Webhook 复选框。Create webhook (创建 webhook) 对话框将再次出现。

- 在 GitHub Enterprise Server 中，选择存储您 CodeBuild 项目的存储库。
- 选择 Settings (设置)，选择 Hooks & services (挂钩和服务)，然后选择 Add webhook (添加 webhook)。
- 输入负载 URL 和私有密钥，接受其他字段的默认值，然后选择 Add webhook。

The screenshot shows the AWS CodeBuild console interface. At the top, there are navigation links: requests 0, Projects 0, Wiki, Pulse, Graphs, and Settings (which is highlighted). Below the navigation is a header bar for 'Webhooks / Add webhook'. The main content area contains the following fields:

- Payload URL \***: A text input field containing a blurred URL.
- Content type**: A dropdown menu set to "application/json".
- Secret**: A text input field containing a long string of dots (.....).
- SSL Verification**: A note stating "By default, we verify SSL certificates when delivering payloads." with a "Disable SSL verification" link next to it.
- Event Triggers**: A section asking "Which events would you like to trigger this webhook?". It includes three radio button options:
  - Just the push event.
  - Send me everything.
  - Let me select individual events.
- Active Status**: A checked checkbox labeled "Active" with the subtext "We will deliver event details when this hook is triggered."
- Add webhook**: A green button at the bottom of the form.

16. 返回您的 CodeBuild 项目。关闭 Create webhook 对话框，然后选择 Start build。

## CodeBuild 的 GitHub 拉取请求和 Webhook 筛选条件示例

当源存储库为 GitHub 时，AWS CodeBuild 支持 Webhook。这意味着，对于将源代码存储在 GitHub 存储库中的 CodeBuild 构建项目，Webhook 可用于在每次将代码更改推送到该存储库时重新构建源代码。

### Note

我们建议您使用筛选条件组来指定哪些 GitHub 用户可以在公共存储库中触发构建。这可以防止用户触发意外的构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#)。)

## 创建将 GitHub 作为源存储库的构建项目并启用 Webhook ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。

3. 选择 Create build project (创建构建项目)。
4. 在 Project configuration (项目配置) 中：

#### 项目名称

输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。您也可以添加构建项目的可选描述,以帮助其他用户理解此项目用于的内容。

5. 在 Source (源) 中：

#### 来源提供者

选择 GitHub 店. 按照说明与 GitHub 连接 (或重新连接), 然后选择 Authorize (授权)。

#### Repository

选择 Repository in my GitHub account (我的 GitHub 账户中的存储库)。

#### GitHub 存储库

输入 GitHub 存储库的 URL。

6. 进 主要来源Webhook事件,选择以下。仅当您选择了 我的GitHub帐户中的存储库 上一步中的。

1. 创建项目时 , 选择 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新生成)。
2. 从 Event type (事件类型) 中 , 选择一个或多个事件。
3. 要在事件触发构建时进行筛选 , 请在 Start a build under these conditions (在这些条件下启动构建) 下 , 添加一个或多个可选筛选条件。
4. 要在未触发事件时进行筛选 , 请在 Start a build under these conditions (在这些条件下不启动构建) 下 , 添加一个或多个可选筛选条件。
5. 选择 添加筛选器组 如果需要,添加其他筛选器组。

7. 在 Environment (环境) 中 :

#### 环境图像

选择以下选项之一

要使用 Docker 镜像,管理方为 AWS CodeBuild:

选择 管理图像,然后从 操作系统, 运行时间, 图片、 和 图像版本. 从 Environment type (环境类型) 中进行选择 (如果可用)。

要使用另一个 Docker 镜像:

选择 自定义图像. 对于 Environment type (环境类型) , 选择 ARM, Linux, Linux GPU, or Windows。如果您选择 其他注册研究,用于 外部注册表网址,使用格式在DockerHub中输入 Docker镜像的名称和标签 `docker repository/docker image name`。如果您选择 Amazon ECR, 使用 Amazon ECR 存储库 和 Amazon ECR 图像 选择您的 AWS 帐户。

要使用专用 Docker 镜像:

选择 自定义图像. 对于 Environment type (环境类型) , 选择 ARM, Linux, Linux GPU, or Windows。对于 Image registry (映像注册表) , 选择 Other registry (其他注册表) , 然后输入您的私有 Docker 映像的凭证的 ARN。凭证必须由 Secrets Manager 创建。有关详细信息,请参阅 [什么是AWS密钥管理器?](#) 在 AWS密钥管理器用户指南.

#### 服务 角色

选择以下选项之一

- 如果您没有 CodeBuild 服务角色,选择 新服务角色. 在 Role name 中 , 为新角色输入名称。
- 如果您有 CodeBuild 服务角色,选择 现有服务角色. 进 角色RN,选择服务角色。

### Note

当您使用控制台来创建或更新构建项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

8. 对于 Buildspec，执行以下操作之一：

- 选择 Use a buildspec file (使用 buildspec 文件) 以在源代码根目录中使用 buildspec.yml 文件。
- 选择 Insert build commands (插入构建命令) 以使用控制台插入构建命令。

有关更多信息，请参见 [构建规范参考 \(p. 136\)](#)。

9. 在 Artifacts (构件) 中：

#### Type

选择以下选项之一

- 如果您不想创建构建输出项目，请选择 No artifacts (无构件)。
- 要将构建输出存储在 S3 存储桶中，请选择 Amazon S3，然后执行以下操作：
  - 如果要将项目名称用于构建输出 ZIP 文件或文件夹，请将 Name (名称) 留空。否则，请输入名称。默认情况下，构件名称是项目名称。如果您要使用其他名称，请在构件名称框中输入该名称。如果您要输出 ZIP 文件，请包含 zip 扩展名。
  - 对于 Bucket name (存储桶名称)，请选择输出存储桶的名称。
  - 如果您在此过程的前面部分选择了 Insert build commands (插入构建命令)，对于 Output files (输出文件)，请输入构建 (该构建要放到构建输出 ZIP 文件或文件夹中) 中的文件位置。对于多个位置，使用逗号分开各个位置 (例如 appspec.yml, target/my-app.jar)。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#) 中 files 的描述。

#### 其他配置

展开 Additional configuration (其他配置) 并根据需要设置选项。

10. 选择 Create build project (创建构建项目)。在 Review (审核) 页面上，选择 Start build (启动构建) 以运行构建。

## 验证检查

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。
3. 执行以下任一操作
  - 选择带有要验证的 Webhook 的构建项目的链接，然后选择 Build details (构建详细信息)。
  - 使用要验证的Webhook选择构建项目旁边的按钮,选择 查看详细信息,然后选择 构建详细信息 选项卡。
4. 进 主要来源Webhook事件,选择 网钩 URL链接。
5. 在您的 GitHub 存储库中，在 Settings (设置) 页面上的 Webhooks 下，确认已选中 Pull Requests (拉取请求) 和 Pushes (推送)。
6. 在您的 GitHub 配置文件设置中的 Personal settings (个人设置)、Applications (应用程序)、Authorized OAuth Apps (已授权 OAuth 应用程序) 下，您应该会看到已获得对所选 AWS 区域的访问权限的应用程序。

You have granted 1 application access to your account.

Sort ▾ Revoke all

 AWS CodeBuild us-east-1	Last used within the last week · Owned by <a href="#">codebuild-oauth-prod</a>	<span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 3px; color: #888;">Revoke</span>
---	--	--

## 创建将构建输出托管在 S3 存储桶中的静态网站

您可以在构建中禁用构件加密。您可能需要禁用构件加密，以便将构件发布到配置为托管网站的位置。（您不能发布加密的构件。）此示例演示如何使用 Webhook 触发构建并将其构件发布到配置为网站的 S3 存储桶。

1. 按照[设置静态网站](#)中的说明操作，以配置一个以网站方式运行的 S3 存储桶。
2. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
3. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
4. 在 Create build project (创建构建项目) 页面上的 Project configuration (项目配置) 中，输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。您也可以包含构建项目的可选描述来帮助其他用户了解此项目的用途。
5. 在 Source (源) 中，对于 Source provider (源提供商)，选择 GitHub。按照说明与 GitHub 连接（或重新连接），然后选择 Authorize (授权)。

对于 Webhook，选择 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新生成)。仅当您已选中 Use a repository in my account (在我的账户中使用存储库) 时才选中此复选框。

Source Add source

Source 1 - Primary

Source provider GitHub

Repository Public repository Repository in my GitHub account

GitHub repository [REDACTED] C

Disconnect GitHub account

▼ Additional configuration

Git clone depth 1

Build Status - optional  Report build statuses to source provider when your builds start and finish

Webhook - optional  Rebuild every time a code change is pushed to this repository

Branch filter - optional [REDACTED]  
Enter a regular expression

6. 在 Environment (环境) 中：

对于 Environment image (环境映像)，执行下列操作之一：

- 要使用由 AWS CodeBuild 托管的 Docker 映像，请选择 Managed image (托管映像)，然后从 Operating system (操作系统)、Runtime(s) (运行时) 和 Image (映像) 以及 Image version (映像版本) 中进行相应选择。从 Environment type (环境类型) 中进行选择（如果可用）。
- 要使用其他 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。如果您针对 External registry URL (外部注册表 URL) 选择 Other registry (其他注册表)，请在 Docker Hub 中按照格式 `docker repository/docker image name` 输入 Docker 映像的名称和标签。如果您选择 Amazon ECR，请使用 Amazon ECR repository (Amazon ECR 存储库) 和 Amazon ECR image (Amazon ECR 映像) 在您的 AWS 账户中选择 Docker 映像。
- 要使用私有 Docker 映像，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择 ARM, Linux, Linux GPU, or Windows。对于 Image registry (映像注册表)，选择 Other registry (其他注册表)，然后输入您的私有 Docker 映像的凭证的 ARN。凭证必须由 Secrets Manager 创建。有关更多信息，请参阅 AWS Secrets Manager 用户指南 中的[什么是 AWS Secrets Manager？](#)

7. 在 Service role (服务角色) 中，执行下列操作之一：

- 如果您没有 CodeBuild 服务角色，请选择 New service role (新建服务角色)。在 Role name (角色名称) 中，为新角色输入一个名称。

- 如果您有 CodeBuild 服务角色，请选择 Existing service role (现有服务角色)。在 Role ARN (角色 ARN) 中，选择服务角色。

Note

当您使用控制台来创建或更新生成项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

8. 对于 Buildspec，执行以下操作之一：

- 选择 Use a buildspec file (使用 buildspec 文件) 以在源代码根目录中使用 buildspec.yml 文件。
- 选择 Insert build commands (插入构建命令) 以使用控制台插入构建命令。

有关更多信息，请参见 [构建规范参考 \(p. 136\)](#)。

- 在 Artifacts (构件) 中，对于 Type (类型)，选择 Amazon S3 以将构建输出存储在 S3 存储桶中。
- 对于 Bucket name (存储桶名称)，选择您在步骤 1 中配置以用作网站的 S3 存储桶的名称。
- 如果在 Environment (环境) 中选择 Insert build commands (插入构建命令)，则对于 Output files (输出文件)，请输入要放到输出存储桶中的构建中的文件位置。如果您有多个位置，请使用逗号分隔每个位置（例如，`appspec.yml, target/my-app.jar`）。有关更多信息，请参阅[Artifacts reference-key in the buildspec file](#)。
- 选择 Disable artifacts encryption (禁用构件加密)。
- 展开 Additional configuration (其他配置) 并根据需要选择选项。
- 选择 Create build project (创建构建项目)。在生成项目页面上的 Build history (生成历史记录) 中，选择 Start build (启动生成) 以运行生成。
- (可选) 按照 Amazon S3 开发人员指南 中的[示例：使用 Amazon CloudFront 加快您的网站速度](#)中的说明操作。

## CodeBuild 的构建规范文件示例中的运行时版本

如果您使用的是 Amazon Linux 2 (AL2) 标准映像版本 1.0 或更高版本或者 Ubuntu 标准映像版本 2.0 或更高版本，则可以在构建规范文件的 runtime-versions 部分中指定一个或多个运行时。本示例向您展示了如何更改您的项目运行时、指定多个运行时以及指定依赖于另一个运行时的运行时。有关支持的运行时的信息，请参阅 [CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。

Note

如果您在构建容器中使用 Docker，则您的构建必须在特权模式下运行。有关更多信息，请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#) 和 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。

## 更新您的运行时版本

您可以通过修改 buildpec 文件的 runtime-versions 部分，将项目使用的运行时更新到新版本。以下示例说明如何指定 Java 版本 8 和 11。

- runtime-versions 部分：如果您使用的是 Amazon Linux 2 标准映像，则指定版本 8 的 Java：

```
phases:  
  install:  
    runtime-versions:  
      java: corretto8
```

- runtime-versions 部分：如果您使用的是 Amazon Linux 2 标准映像，则指定版本 11 的 Java：

```
phases:  
  install:  
    runtime-versions:  
      java: corretto11
```

- **runtime-versions** 部分：如果您使用的是 Ubuntu 标准映像 2.0，则指定版本 8 的 Java：

```
phases:  
  install:  
    runtime-versions:  
      java: openjdk8
```

- **runtime-versions** 部分：如果您使用的是 Ubuntu 标准映像 2.0，则指定版本 11 的 Java：

```
phases:  
  install:  
    runtime-versions:  
      java: openjdk11
```

以下示例说明如何使用 Ubuntu 标准映像 2.0 或 Amazon Linux 2 标准映像 2.0 指定不同版本的 Node.js：

- 一个 **runtime-versions** 部分，指定 Node.js 版本 8：

```
phases:  
  install:  
    runtime-versions:  
      nodejs: 8
```

- 一个 **runtime-versions** 部分，指定 Node.js 版本 10：

```
phases:  
  install:  
    runtime-versions:  
      nodejs: 10
```

本示例演示一个项目，该项目从 Java 版本 8 运行时开始，然后更新到 Java 版本 10 运行时。

1. 按照 [创建源代码 \(p. 68\)](#) 中的步骤 1 和 2 生成源代码。如果成功，系统会创建一个名为 my-web-app 的目录，其中包含您的源文件。
2. 使用以下内容创建名为 buildspec.yml 的文件。将此文件存储到 *(root directory name)/my-web-app* 目录。

```
version: 0.2  
  
phases:  
  install:  
    runtime-versions:  
      java: corretto8  
  build:  
    commands:  
      - java -version  
      - mvn package  
artifacts:  
  files:  
    - '**/*'
```

```
base-directory: 'target/my-web-app'
```

在 buildspec 文件中：

- `runtime-versions` 部分指定项目使用 Java 运行时版本 8。
- `- java -version` 命令显示您的项目在执行构建时所使用的 Java 版本。

您的文件结构现在应如下所示。

```
(root directory name)
-- my-web-app
  |-- src
  |   |-- main
  |   |   |-- resources
  |   |   |   |-- webapp
  |   |   |   |   |-- WEB-INF
  |   |   |   |   |   '-- web.xml
  |   |   |   '-- index.jsp
  |   '-- buildspec.yml
  '-- pom.xml
```

3. 将 `my-web-app` 目录的内容上传到 S3 输入存储桶或上传到 CodeCommit、GitHub 或 Bitbucket 存储库。

#### Important

请不要上传 `(root directory name)` 或 `(root directory name)/my-web-app`，而只上传 `(root directory name)/my-web-app` 中的目录和文件。

如果您使用的是 S3 输入存储桶，请确保创建一个包含目录结构和文件的 ZIP 文件，然后将其上传至输入存储桶。请不要将 `(root directory name)` 或 `(root directory name)/my-web-app` 添加到 ZIP 文件中，而只添加 `(root directory name)/my-web-app` 中的目录和文件。

4. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
5. 创建构建项目。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)和[运行构建 \(控制台\) \(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值。
  - 对于 Environment (环境)：
    - 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
    - 对于 Operating system (操作系统)，选择 Amazon Linux 2。
    - 对于 Runtime(s) (运行时)，选择 Standard (标准)。
    - 对于 Image (映像)，选择 `aws/codebuild/amazonlinux2-x86_64-standard:3.0`。
6. 选择 Start build。
7. 在 Build configuration (构建配置) 上，接受默认值，然后选择 Start build (开始构建)。
8. 当构建完成后，在 Build logs (构建日志) 选项卡上查看构建输出。您应该可以看到类似于如下所示的输出内容：

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto8' based on manual
selections...
[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...

[Container] Date Time Running command export JAVA_HOME="$JAVA_8_HOME"
```

```
[Container] Date Time Running command export JRE_HOME="$JRE_8_HOME"  
[Container] Date Time Running command export JDK_HOME="$JDK_8_HOME"  
[Container] Date Time Running command for tool_path in "$JAVA_8_HOME"/bin/*  
"$JRE_8_HOME"/bin/*;
```

9. 使用 Java 版本 11 更新 runtime-versions 部分：

```
install:  
  runtime-versions:  
    java: corretto11
```

10. 保存更改后，再次运行您的构建并查看构建输出。您应看到已安装的 Java 版本为 11。您应该可以看到类似于如下所示的输出内容：

```
[Container] Date Time Phase is DOWNLOAD_SOURCE  
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src  
[Container] Date Time YAML location is /codebuild/output/src460614277/src/buildspec.yml  
[Container] Date Time Processing environment variables  
[Container] Date Time Selecting 'java' runtime version 'corretto11' based on manual  
selections...  
Installing Java version 11 ...  
  
[Container] Date Time Running command export JAVA_HOME="$JAVA_11_HOME"  
[Container] Date Time Running command export JRE_HOME="$JRE_11_HOME"  
[Container] Date Time Running command export JDK_HOME="$JDK_11_HOME"  
[Container] Date Time Running command for tool_path in "$JAVA_11_HOME"/bin/*  
"$JRE_11_HOME"/bin/*;
```

## 指定运行时依赖关系

本示例说明如何指定运行时和具有依赖关系的运行时。例如，任何支持的 Android 运行时版本都依赖于 Java 运行时版本 8。例如，如果您指定 Android 版本 29 并使用 Amazon Linux 2 或 Ubuntu，则还可以指定 Java 版本 8。如果未指定依赖运行时，则 CodeBuild 会尝试为您选择它。

本例中的构建项目使用 GitHub [AWS 示例](#) 存储库中的源代码。源代码使用 Android 版本 28 运行时，构建项目使用 Amazon Linux 2，因此构建规范还必须指定 Java 版本 8。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 创建构建项目。有关更多信息，请参阅[创建构建项目（控制台）\(p. 190\)](#)和[运行构建（控制台）\(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值。
  - 对于 Environment (环境)：
    - 对于 Source provider (源提供商)，选择 GitHub。
    - 对于 Repository (存储库)，选择 Public repository (公有存储库)。
    - 对于 Repository URL (存储库 URL)，输入 <https://github.com/aws-samples/aws-mobile-android-notes-tutorial>。
    - 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
    - 对于 Operating system (操作系统)，选择 Amazon Linux 2。
    - 对于 Runtime(s) (运行时)，选择 Standard (标准)。
    - 对于 Image (映像)，选择 aws/codebuild/amazonlinux2-x86\_64-standard:3.0。
3. 对于 Build specifications (构建规范)，选择 Insert build commands (插入构建命令)，然后选择 Switch to editor (切换到编辑器)。

4. 在 Build commands (构建命令) 中，将占位符文本替换为以下内容：

```
version: 0.2

phases:
  install:
    runtime-versions:
      android: 29
      java: corretto8
  build:
    commands:
      - ./gradlew assembleDebug
artifacts:
  files:
    - app/build/outputs/apk/app-debug.apk
```

此 runtime-versions 部分同时指定 Android 版本 29 和 Java 版本 8 运行时。

5. 选择 Create build project (创建构建项目)。
6. 选择 Start build。
7. 在 Build configuration (构建配置) 上，接受默认值，然后选择 Start build (开始构建)。
8. 当构建完成后，在 Build logs (构建日志) 选项卡上查看构建输出。您应该可以看到类似于如下所示的输出内容。该内容中显示已安装 Android 版本 29 和 Java 版本 8：

```
[Container] 2019/05/14 23:21:42 Entering phase DOWNLOAD_SOURCES
[Container] Date Time Running command echo "Installing Android version 29 ..."
Installing Android version 29 ...

[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...
```

## 指定两个运行时

您可以在同一个 CodeBuild 构建项目中指定多个运行时。此示例项目使用两个源文件：一个使用 Go 运行时，另一个使用 Node.js 运行时。

1. 创建一个名为 my-source 的目录。
2. 在 my-source 目录中，创建一个名为 golang-app 的目录。
3. 使用以下内容创建名为 hello.go 的文件。将此文件存储到 golang-app 目录。

```
package main
import "fmt"

func main() {
    fmt.Println("hello world from golang")
    fmt.Println("1+1 =", 1+1)
    fmt.Println("7.0/3.0 =", 7.0/3.0)
    fmt.Println(true && false)
    fmt.Println(true || false)
    fmt.Println(!true)
    fmt.Println("good bye from golang")
}
```

4. 在 my-source 目录中，创建一个名为 nodejs-app 的目录。它应该与 golang-app 目录同级。
5. 使用以下内容创建名为 index.js 的文件。将此文件存储到 nodejs-app 目录。

```
console.log("hello world from nodejs");
console.log("1+1 =" + (1+1));
```

```
console.log("7.0/3.0 =" + 7.0/3.0);
console.log(true && false);
console.log(true || false);
console.log(!true);
console.log("good bye from nodejs");
```

6. 使用以下内容创建名为 package.json 的文件。将此文件存储到 nodejs-app 目录。

```
{
  "name": "mycompany-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"run some tests here\\\""
  },
  "author": "",
  "license": "ISC"
}
```

7. 使用以下内容创建名为 buildspec.yml 的文件。将文件存储在 my-source 目录中，该目录与 nodejs-app 以及 golang-app 目录同级。runtime-versions 部分指定 Node.js 版本 10 运行时和 Go 版本 1.13 运行时。

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
      nodejs: 10
  build:
    commands:
      - echo Building the Go code...
      - cd $CODEBUILD_SRC_DIR/golang-app
      - go build hello.go
      - echo Building the Node code...
      - cd $CODEBUILD_SRC_DIR/nodejs-app
      - npm run test
  artifacts:
    secondary-artifacts:
      golang_artifacts:
        base-directory: golang-app
        files:
          - hello
      nodejs_artifacts:
        base-directory: nodejs-app
        files:
          - index.js
          - package.json
```

8. 您的文件结构现在应如下所示。

```
-- my-source
| -- golang-app
|   -- hello.go
| -- nodejs.app
|   -- index.js
|   -- package.json
| -- buildspec.yml
```

9. 将 my-source 目录的内容上传到 S3 输入存储桶或上传到 CodeCommit、GitHub 或 Bitbucket 存储库。

### Important

如果您使用的是 S3 输入存储桶，请确保创建一个包含目录结构和文件的 ZIP 文件，然后将其上传至输入存储桶。请不要将 my-source 添加到 ZIP 文件中，而只添加 my-source 中的目录和文件。

10. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
11. 创建构建项目。有关更多信息，请参阅[创建构建项目（控制台）\(p. 190\)](#)和[运行构建（控制台）\(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值。
- 对于 Environment (环境)：
  - 对于 Environment image (环境映像)，选择 Managed image (托管映像)。
  - 对于 Operating system (操作系统)，选择 Amazon Linux 2。
  - 对于 Runtime(s) (运行时)，选择 Standard (标准)。
  - 对于 Image (映像)，选择 aws/codebuild/amazonlinux2-x86\_64-standard:2.0。
12. 选择 Create build project (创建构建项目)。
13. 选择 Start build。
14. 在 Build configuration (构建配置) 上，接受默认值，然后选择 Start build (开始构建)。
15. 当构建完成后，在 Build logs (构建日志) 选项卡上查看构建输出。您应该可以看到类似于如下所示的输出内容。它显示来自 Go 和 Node.js 运行时的输出，还显示来自 Go 和 Node.js 应用程序的输出。

```
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'golang' runtime version '1.13' based on manual
selections...
[Container] Date Time Selecting 'nodejs' runtime version '10' based on manual
selections...
[Container] Date Time Running command echo "Installing Go version 1.13 ..."
Installing Go version 1.12 ...

[Container] Date Time Running command echo "Installing Node.js version 10 ..."
Installing Node.js version 10 ...

[Container] Date Time Running command n 10.15.3

[Container] Date Time Moving to directory /codebuild/output/src819694850/src
[Container] Date Time Registering with agent
[Container] Date Time Phases found in YAML: 2
[Container] Date Time  INSTALL: 0 commands
[Container] Date Time  BUILD: 1 commands
[Container] Date Time Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase INSTALL
[Container] Date Time Phase complete: INSTALL State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase PRE_BUILD
[Container] Date Time Phase complete: PRE_BUILD State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase BUILD
[Container] Date Time Running command echo Building the Go code...
Building the Go code...

[Container] Date Time Running command cd ${CODEBUILD_SRC_DIR}/golang-app
[Container] Date Time Running command go build hello.go

[Container] Date Time Running command echo Building the Node code...
Building the Node code...

[Container] Date Time Running command cd ${CODEBUILD_SRC_DIR}/nodejs-app
```

```
[Container] Date Time Running command npm run test
> mycompany-app@1.0.0 test /codebuild/output/src924084119/src/nodejs-app
> echo "run some tests here"
run some tests here
```

## 使用 AWS CodeBuild 的源版本示例

此示例演示如何使用提交 ID 以外的格式（也称为提交 SHA）指定源的版本。您可以通过以下方式指定源的版本：

- 对于 Amazon S3 源提供程序，请使用表示构建输入 ZIP 文件的对象的版本 ID。
- 对于 CodeCommit、Bitbucket、GitHub 和 GitHub Enterprise Server，请使用下列项之一：
  - 拉取请求作为拉取请求参考（例如，`refs/pull/1/head`）。
  - 分支作为分支名称。
  - 提交 ID。
  - 标签。
  - 参考和提交 ID。参考可以是下列项之一：
    - 标签（例如，`refs/tags/mytagv1.0^{full-commit-SHA}`）。
    - 分支（例如，`refs/heads/mydevbranch^{full-commit-SHA}`）。
    - 拉取请求（例如，`refs/pull/1/head^{full-commit-SHA}`）。

### Note

仅在存储库为 GitHub 或 GitHub Enterprise Server 时可指定拉取请求源的版本。

如果使用参考和提交 ID 指定版本，则构建的 DOWNLOAD\_SOURCE 阶段比仅提供版本时更快。这是因为，在添加参考时，CodeBuild 不需要下载整个存储库来查找提交。

- 您可以指定仅具有提交ID的源版本，例如 `12345678901234567890123467890123456789...` 如果你这么做 CodeBuild 必须下载整个资料库才能找到版本。
- 您可以以此格式指定带有参考编号和提交ID的源版本：`refs/heads/branchname^{full-commit-SHA}`（例如，`refs/heads/main^{12345678901234567890123467890123456789}`）。如果您执行此操作，则 CodeBuild 仅下载指定的分支来查找版本。

### Note

要加快构建的 DOWNLOAD\_SOURCE 阶段，您也可以将 Git clone depth (Git 克隆深度) 设置为较小数字。CodeBuild 下载较少版本的存储库。

### 使用提交 ID 指定 GitHub 存储库版本

- Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
- 创建构建项目()有关信息，请参阅 [创建构建项目 \(控制台\) \(p. 190\)](#) 和 [运行构建 \(控制台\) \(p. 252\)](#)。除这些设置以外，将所有设置保留为默认值：
  - 在 Source (源) 中：
    - 对于 Source provider (源提供商)，选择 GitHub。如果未连接到 GitHub，请按照说明操作以进行连接。
    - 对于 Repository (存储库)，选择 Public repository (公有存储库)。

- 对于 Repository URL (存储库 URL) , 输入 <https://github.com/aws/aws-sdk-ruby.git>。
- 在 Environment (环境) 中 :
  - 对于 Environment image (环境映像) , 选择 Managed image (托管映像)。
  - 对于 Operating system (操作系统) , 选择 Amazon Linux 2。
  - 对于 Runtime(s) (运行时) , 选择 Standard (标准)。
  - 对于 Image (映像) , 选择 aws/codebuild/amazonlinux2-x86\_64-standard:2.0。
- 3. 对于 Build specifications (构建规范) , 选择 Insert build commands (插入构建命令) , 然后选择 Switch to editor (切换到编辑器)。
- 4. 在 Build commands (构建命令) 中 , 将占位符文本替换为以下内容 :

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  build:
    commands:
      - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
```

在使用 Ubuntu 标准映像 2.0 时需要 runtime-versions 部分。这里指定了 Ruby 版本 2.6 运行时 , 但您可以使用任何运行时。echo 命令显示存储在 CODEBUILD\_RESOLVED\_SOURCE\_VERSION 环境变量中的源代码的版本。

5. 在 Build configuration (构建配置) 上 , 接受默认值 , 然后选择 Start build (开始构建)。
6. 对于 源版本 , 输入 [046e8b67481d53bdc86c3f6affdd5d1afae6d369...这是在 https://github.com/aws/aws-sdk-ruby.git 资料库。](https://github.com/aws/aws-sdk-ruby.git)
7. 选择 Start build。
8. 在构建完成后 , 您应该看到以下内容 :

- 在 Build logs (构建日志) 选项卡上 , 使用了哪个版本的项目源。以下是 示例。

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
046e8b67481d53bdc86c3f6affdd5d1afae6d369

[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- 在 Environment variables (环境变量) 选项卡上 , Resolved source version (解析的源版本) 与用于创建构建的提交 ID 匹配。
- 在 Phase details (阶段详细信息) 选项卡上 , 显示 DOWNLOAD\_SOURCE 阶段的持续时间。

这些步骤说明如何使用相同版本的源创建构建。这一次 , 使用参考和提交 ID 指定源的版本。

### 使用提交 ID 和参考指定 GitHub 存储库版本

1. 在左侧导航窗格中 , 选择 Build projects (构建项目) , 然后选择您之前创建的项目。
2. 选择 Start build。
3. 在 源版本 , 输入 `refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369...这是相同的提交ID , 并且是一个以格式引用的分支 refs/heads/branchname^{full-commit-SHA}.`
4. 选择 Start build。
5. 在构建完成后 , 您应该看到以下内容 :

- 在 Build logs (构建日志) 选项卡上 , 使用了哪个版本的项目源。以下是 示例。

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION  
046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

```
[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- 在 Environment variables (环境变量) 选项卡上，Resolved source version (解析的源版本) 与用于创建构建的提交 ID 匹配。
- 在 Phase details (阶段详细信息) 选项卡上，DOWNLOAD\_SOURCE 阶段的持续时间应短于仅使用提交 ID 指定源版本时的持续时间。

## 适用于 CodeBuild 的私有注册表与 AWS Secrets Manager 示例

此示例向您显示如何使用在私有注册表中存储的 Docker 映像作为您的 AWS CodeBuild 运行时环境。私有注册表的凭证存储在 AWS Secrets Manager 中。任何私有注册表都适用于 CodeBuild。此示例使用 Docker Hub。

### 私有注册表示例要求

要将私有注册表与 AWS CodeBuild 结合使用，您必须具有以下各项：

- 一个 Secrets Manager 密钥，用于储存您的 Docker Hub 凭证。该凭证可用于访问您的私有存储库。
- 一个私有存储库或账户。
- 一个 CodeBuild 服务角色 IAM 策略，该策略允许访问您的 Secrets Manager 密钥。

按照以下步骤来创建这些资源，然后使用在您的私有注册表中存储的 Docker 映像来创建 CodeBuild 构建项目。

### 使用私有注册表创建 CodeBuild 项目

- 有关如何创建免费的私有存储库的更多信息，请参阅 [Docker Hub 上的存储库](#)。您还可以在终端中运行以下命令来提取映像、获取其ID，并将其推送到新的存储库。

```
docker pull amazonlinux  
docker images amazonlinux --format {{.ID}}  
docker tag image-id your-username/repository-name:latest  
docker login  
docker push your-username/repository-name
```

- 按照 AWS Secrets Manager 用户指南中的[创建基本密钥](#)中的步骤进行操作。在第 3 步的选择密钥类型中，执行以下操作：

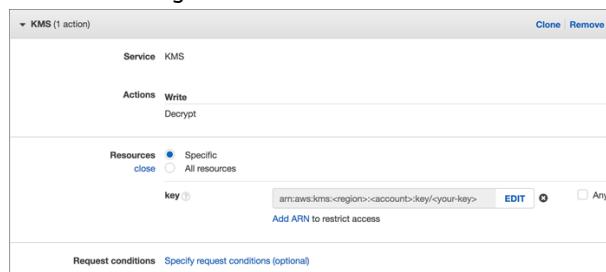
- 选择其他密钥类型。

- b. 在密钥键/值中，为您的 Docker Hub 用户名创建一个键值对，为您的 Docker Hub 密码创建一个键值对。
- c. 对于密钥名称，输入一个名称，例如 **dockerhub**。您可以输入可选说明以帮助您记住这是 Docker Hub 的密钥。
- d. 将禁用自动轮换选定，因为密钥对应于您的 Docker Hub 凭证。
- e. 选择 **Store secret (存储密钥)**。
- f. 当您检查设置时，请记下要在此示例中稍后使用的ARN。

有关更多信息，请参阅[什么是 AWS Secrets Manager ?](#)

3. 当您在控制台中创建 AWS CodeBuild 项目时，CodeBuild 会为您附加必需权限。如果您使用的是 **DefaultEncryptionKey** 之外的 AWS KMS 密钥，您必须将其添加到服务角色。有关更多信息，请参阅 [AWS Identity and Management 用户指南 中的修改角色 \( 控制台 \)](#)。

要使您的服务角色与 Secrets Manager 配合使用，它必须至少具有 **secretsmanager:GetSecretValue** 权限。



4. 要使用控制台创建一个具有在私有注册表中存储的环境的项目，请在创建项目时执行以下操作。有关信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)。

#### Note

如果您的私有注册表是在 VPC 中，它必须拥有公有 Internet 访问权限。CodeBuild 无法从 VPC 中的私有 IP 地址拉取映像。

- a. 在环境中，选择自定义映像。
- b. 对于 **Environment type (环境类型)**，选择 Linux 或 Windows。
- c. 对于自定义映像类型，选择其他位置。
- d. 在其他位置中，输入映像位置以及 Secrets Manager 凭证的 ARN 或名称。

### Note

如果您的凭证在当前区域中不存在，则必须使用 ARN。如果凭证存在于其他区域中，则无法使用凭证名称。

## 多输入源和输出构件示例

您可以创建具有多个输入源和多个输出构件集的 AWS CodeBuild 构建项目。此示例演示如何设置具有以下特点的构建项目：

- 使用多个不同类型的源和存储库。
- 将构建构件发布到单个构建中的多个 S3 存储桶。

在此示例中，将创建一个构建项目并使用它运行构建任务。示例使用构建项目的 buildspec 文件演示如何合并多个源和创建多个构件集。

1. 将源上传到一个或多个 S3 存储桶、CodeCommit、GitHub、GitHub Enterprise Server 或 Bitbucket 存储库。
2. 选择一个源作为主要源。CodeBuild 将在此源中查找和执行 buildspec 文件。
3. 创建构建项目。有关更多信息，请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。
4. 按照[直接运行 AWS CodeBuild \(p. 367\)](#) 中的说明创建构建项目、运行构建和获取有关构建的信息。
5. 如果使用 AWS CLI 创建构建项目，则 create-project 命令的 JSON 格式输入可能类似于以下内容：

```
{  
  "name": "sample-project",  
  "source": {  
    "type": "S3",  
    "location": "bucket/sample.zip"  
  },  
  "secondarySources": [  
    {  
      "type": "CODECOMMIT",  
      "location": "https://git-codecommit.us-west-2.amazonaws.com/v1/repos/repo"  
      "sourceIdentifier": "source1"  
    },  
    {  
      "type": "GITHUB",  
      "location": "https://github.com/awslabs/aws-codebuild-jenkins-plugin"  
      "sourceIdentifier": "source2"  
    }  
  ],  
  "secondaryArtifacts": [  
    {  
      "type": "S3",  
      "location": "output-bucket",  
      "artifactIdentifier": "artifact1"  
    },  
    {  
      "type": "S3",  
      "location": "other-output-bucket",  
      "artifactIdentifier": "artifact2"  
    }  
  ],  
  "environment": {  
    "type": "LINUX_CONTAINER",  
    "image": "aws/codebuild/standard:4.0",  
    "computeType": "BUILD_GENERAL1_SMALL"  
  }  
}
```

```
    },
    "serviceRole": "arn:aws:iam::account-ID:role/role-name",
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

主要源在 `source` 属性下定义。所有其他源都称为辅助源，出现在 `secondarySources` 下方。所有辅助源都安装在各自的目录中。目录存储在内置环境变量 `CODEBUILD_SRC_DIR_sourceIdentifier` 中。有关更多信息，请参阅[构建环境中的环境变量 \(p. 168\)](#)。

`secondaryArtifacts` 属性包含构件定义列表。这些构件使用 `buildspec` 文件的 `secondary-artifacts` 块（嵌套在 `artifacts` 块内）。

`buildspec` 文件中的辅助构件与构件具有相同的结构，以其构件标识符分隔。

**Note**

在 [CodeBuild API](#) 中，辅助项目的 `artifactIdentifier` 是 `CreateProject` 和 `UpdateProject` 中的必需属性。必须使用它引用辅助构件。

使用前面的 JSON 格式的输入，项目的 `buildspec` 文件可能如下所示：

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk11
  build:
    commands:
      - cd $CODEBUILD_SRC_DIR_source1
      - touch file1
      - cd $CODEBUILD_SRC_DIR_source2
      - touch file2

  artifacts:
    secondary-artifacts:
      artifact1:
        base-directory: $CODEBUILD_SRC_DIR_source1
        files:
          - file1
      artifact2:
        base-directory: $CODEBUILD_SRC_DIR_source2
        files:
          - file2
```

可以在 `sourceVersion` 中使用具有 `StartBuild` 属性的 API 覆盖主要源的版本。要覆盖一个或多个辅助源版本，请使用 `secondarySourceVersionOverride` 属性。

AWS CLI 中 `start-build` 命令的 JSON 格式输入可能类似于以下内容：

```
{
  "projectName": "sample-project",
  "secondarySourcesVersionOverride": [
    {
      "sourceIdentifier": "source1",
      "sourceVersion": "codecommit-branch"
    },
    {
      "sourceIdentifier": "source2",
      "sourceVersion": "github-branch"
    }
  ]}
```

```
    ]  
}
```

## 无源项目示例

配置 CodeBuild 项目时，可以在配置源时选择 **NO\_SOURCE** 源类型。当您的源类型为 **NO\_SOURCE** 时，您不能指定 buildspec 文件，因为您的项目没有源。相反，您必须将 JSON 格式输入的 buildspec 属性中的 YAML 格式 buildspec 字符串指定给 `create-project` CLI 命令。它可能如下所示：

```
{  
  "name": "project-name",  
  "source": {  
    "type": "NO_SOURCE",  
    "buildspec": "version: 0.2\n\nphases:\n  build:\n    commands:\n      - command"  
  },  
  "environment": {  
    "type": "LINUX_CONTAINER",  
    "image": "aws/codebuild/standard:4.0",  
    "computeType": "BUILD_GENERAL1_SMALL",  
  },  
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",  
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"  
}
```

有关更多信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#)。

要了解如何创建使用多个到 CodeBuild 的源输入创建多个输出构件的管道，请参阅 [AWS CodePipeline 与 CodeBuild 和多输入源和输出构件集成示例 \(p. 64\)](#)。

## 使用语义版本控制命名构建构件示例

此示例包含示例 buildspec 文件，演示如何指定在构建时创建的构件名称。在 buildspec 文件中指定的名称可以包含 Shell 命令和环境变量，以使其保持唯一。在 buildspec 文件中指定的名称将覆盖创建项目时在控制台中输入的名称。

如果构建多次，则使用在 buildspec 文件中指定的构件名称可以确保输出构件文件名的唯一性。例如，您可以使用在构建时插入构件名称的日期和时间戳。

要使用 buildspec 文件中的构件名称覆盖在控制台中输入的构件名称，请执行以下操作：

1. 设置构建项目以使用 buildspec 文件中的构件名称覆盖相应的构件名称。
  - 如果您使用控制台创建您的构建项目，请选择 `Enable semantic versioning` (启用语义版本控制)。有关更多信息，请参阅 [创建构建项目 \(控制台\) \(p. 190\)](#)。)
  - 如果您使用 AWS CLI，设置 `overrideArtifactName` 在 JSON 格式化文件中的“真”中，传递到 `create-project...` 有关详细信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#)。
  - 如果使用 AWS CodeBuild API，请在创建或更新项目或启动构建时在 `ProjectArtifacts` 对象上设置 `overrideArtifactName` 标记。
2. 在 buildspec 文件中指定名称。使用以下示例 buildspec 文件作为指南。

此 Linux 示例演示如何指定包含构建创建日期的构件名称：

```
version: 0.2  
phases:  
  build:  
    commands:  
      - rspec HelloWorld_spec.rb
```

```
artifacts:  
  files:  
    - '**/*'  
  name: myname-$(date +%Y-%m-%d)
```

此 Linux 示例演示如何指定使用 CodeBuild 环境变量的构件名称。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)。)

```
version: 0.2  
phases:  
  build:  
    commands:  
      - rspec HelloWorld_spec.rb  
artifacts:  
  files:  
    - '**/*'  
  name: myname-$AWS_REGION
```

此 Windows 示例演示如何指定包含构建创建日期和时间的构件名称：

```
version: 0.2  
env:  
  variables:  
    TEST_ENV_VARIABLE: myArtifactName  
phases:  
  build:  
    commands:  
      - cd samples/helloworld  
      - dotnet restore  
      - dotnet run  
artifacts:  
  files:  
    - '**/*'  
  name: $Env:TEST_ENV_VARIABLE-$(Get-Date -UFormat "%Y%m%d-%H%M%S")
```

此 Windows 示例演示如何指定使用在 buildspec 文件中声明的变量和 CodeBuild 环境变量的构件名称。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)。)

```
version: 0.2  
env:  
  variables:  
    TEST_ENV_VARIABLE: myArtifactName  
phases:  
  build:  
    commands:  
      - cd samples/helloworld  
      - dotnet restore  
      - dotnet run  
artifacts:  
  files:  
    - '**/*'  
  name: $Env:TEST_ENV_VARIABLE-$Env:AWS_REGION
```

有关更多信息，请参阅 [适用于 CodeBuild 的构建规范参考 \(p. 136\)](#)。)

# 在 AWS CodeBuild 中计划构建

在使用 AWS CodeBuild 之前，您必须回答以下问题：

- 源代码存储在什么位置？CodeBuild 目前支持通过以下源代码存储库提供商进行构建。源代码必须包含构建规范文件。构建规范是构建命令和相关设置的集合，采用 YAML 格式，由 CodeBuild 用来运行构建任务。您可以在构建项目定义中声明构建规范。

存储库提供商	必填项	文档
CodeCommit	存储库名称。  (可选) 与源代码关联的提交 ID。	请参阅 AWS CodeCommit 用户指南 中的以下主题：  <a href="#">创建 CodeCommit 存储库</a> 。  <a href="#">在 CodeCommit 中创建提交</a>
Amazon S3	输入存储桶名称。  与包含源代码的构建输入 ZIP 文件对应的对象名称。  (可选) 与构建输入 ZIP 文件相关联的版本 ID。	请参阅 Amazon S3 入门指南 中的以下主题：  <a href="#">创建存储桶</a>  <a href="#">向存储桶添加对象</a>
GitHub	存储库名称。  (可选) 与源代码关联的提交 ID。	请参阅 GitHub 帮助网站上的以下主题：  <a href="#">创建存储库</a>
Bitbucket	存储库名称。  (可选) 与源代码关联的提交 ID。	请参阅 Bitbucket 云文档网站上的以下主题：  <a href="#">创建存储库</a>

- 您需要运行哪些构建命令以及按照什么顺序运行？默认情况下，CodeBuild 会从您指定的提供商处下载构建输入，然后将构建输出上传到您指定的存储桶。您可以使用构建规范来指示如何将下载的构建输入转换为预期的构建输出。有关更多信息，请参阅[构建规范参考 \(p. 136\)](#)。
- 运行构建需要哪些运行时和工具？例如，您是否是为 Java、Ruby、Python 或 Node.js 构建的？构建是否需要 Maven 或 Ant？或者是否需要适用于 Java、Ruby 或 Python 的编译器？构建是否需要 Git、AWS CLI 或其他工具？

CodeBuild 在使用 Docker 映像的构建环境中运行构建。这些 Docker 映像必须存储在 CodeBuild 支持的存储库类型中。这些存储库包括 CodeBuild Docker 映像存储库、Docker Hub 和 Amazon Elastic Container Registry (Amazon ECR)。有关 CodeBuild Docker 映像存储库的更多信息，请参阅[CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。

## Important

如果您使用 Ubuntu 标准映像 2.0 及更高版本或 Amazon Linux 2 (AL2) 标准映像 1.0 及更高版本，则必须在构建规格文件中指定 `runtime-versions`。有关更多信息，请参阅[Specify runtime versions in the buildspec file](#)。

- 是否需要 CodeBuild 不会自动提供的 AWS 资源？如果需要，这些资源又需要哪些安全策略呢？例如，您可能需要修改 CodeBuild 服务角色来允许 CodeBuild 使用这些资源。

5. 是否要将 CodeBuild 与您的 VPC 结合使用？如果是，您需要您的 VPC 配置的 VPC ID、子网 ID 和安全组 ID。有关更多信息，请参阅[将 AWS CodeBuild 与 Amazon Virtual Private Cloud 结合使用 \(p. 173\)](#)。

回答完这些问题后，您应该已经具备了成功运行构建所需的设置和资源。要运行构建，您可以：

- 使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包。有关更多信息，请参阅[直接运行 CodeBuild \(p. 367\)](#)。
- 在 AWS CodePipeline 中创建或标识管道，然后添加构建或测试操作来指示 CodeBuild 自动测试代码、运行构建，或同时执行两者。有关更多信息，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。

## 适用于 CodeBuild 的构建规范参考

本主题提供有关构建规范(buildspec)文件的重要参考信息。构建规范是构建命令和相关设置的集合，采用 YAML 格式，由 CodeBuild 用来运行构建任务。您可以将buildspec作为源代码的一部分包括在内，或者在创建build项目时定义buildspec。有关构建规范工作原理的信息，请参阅[CodeBuild 的工作原理 \(p. 3\)](#)。

### 主题

- [构建规范文件名称和存储位置 \(p. 136\)](#)
- [构建规范语法 \(p. 137\)](#)
- [构建规范示例 \(p. 152\)](#)
- [构建规范版本 \(p. 154\)](#)
- [批量构建BuildSpec参考 \(p. 155\)](#)

## 构建规范文件名称和存储位置

如果您在源代码中包含构建规范，则默认情况下，构建规范文件必须命名为 buildspec.yml 且放置在源目录的根目录中。

可以覆盖默认构建规范文件名和位置。例如，您可以：

- 对同一存储库中的不同构建使用不同的构建规范文件，如 buildspec\_debug.yml 和 buildspec\_release.yml。
- 将构建规范文件存储在源目录的根目录之外的位置，如 config/buildspec.yml 或 S3 存储桶。S3 存储桶必须与您的构建项目位于同一 AWS 区域。使用其 ARN 指定构建规范文件（例如，arn:aws:s3:::my-codebuild-sample2/buildspec.yml）。

您可以只为构建项目指定一个构建规范，而不管构建规范文件的名称如何。

要覆盖默认构建规范文件名、默认位置或这两者，执行下列操作之一：

- 运行 AWS CLI `create-project` 或 `update-project` 命令，设置 buildspec 相对于内置环境变量的值到备用buildspec文件的路径的值 CODEBUILD\_SRC\_DIR。您也可以使用等效的 `create project` 操作 AWS SDK。有关更多信息，请参阅[创建构建项目 \(\) \(p. 189\)](#)或[更改构建项目的设置 \(\) \(p. 236\)](#)。
- 运行 AWS CLI `start-build` 命令，设置 buildspecOverride 相对于内置环境变量的值到备用 buildspec文件的路径的值 CODEBUILD\_SRC\_DIR。您也可以使用等效的 `start build` 操作 AWS SDK。有关更多信息，请参阅[运行构建 \(\) \(p. 252\)](#)。
- 在 AWS CloudFormation 模板，设置 BuildSpec 属性 Source 类型资源中 `AWS::CodeBuild::Project` 到替代buildspec文件的路径（相对于内置环境变量的值）CODEBUILD\_SRC\_DIR。有关更多信息，请参见中的BuildSpec属性 [AWS CodeBuild 项目源](#) 在 AWS CloudFormation 用户指南。

## 构建规范语法

构建规范文件必须以 [YAML](#) 格式表示。

如果命令包含 YAML 不支持的字符或字符串，则必须用引号 ("") 将命令括起来。以下命令用引号括起来，因为在 YAML 中不允许使用冒号 (:) 后跟空格。命令中的引号会被转义 ("")。

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }' | sed 's/[\", ]//g')"
```

构建规范的语法如下：

```
version (p. 138): 0.2

run-as (p. 138): Linux-user-name

env (p. 139):
  shell (p. 139): shell-tag
  variables (p. 139):
    key: "value"
    key: "value"
  parameter-store (p. 139):
    key: "value"
    key: "value"
  exported-variables (p. 140):
    - variable
    - variable
  secrets-manager (p. 140):
    key: secret-id:json-key:version-stage:version-id
  git-credential-helper (p. 141): no | yes

proxy (p. 141):
  upload-artifacts (p. 141): no | yes
  logs (p. 141): no | yes

batch (p. 155):
  fast-fail: false | true
  # build-list:
  # build-matrix:
  # build-graph:

phases (p. 141):
  install (p. 141):
    run-as (p. 141): Linux-user-name
    runtime-versions (p. 141):
      runtime: version
      runtime: version
    commands (p. 146):
      - command
      - command
  finally (p. 146):
    - command
    - command
  pre_build (p. 147):
    run-as (p. 141): Linux-user-name
    commands (p. 147):
      - command
      - command
  finally (p. 147):
    - command
    - command
  build (p. 147):
    run-as (p. 141): Linux-user-name
```

```
commands (p. 147):
  - command
  - command
finally (p. 147):
  - command
  - command
post_build (p. 147):
  run-as (p. 141): Linux-user-name
  commands (p. 147):
    - command
    - command
  finally (p. 147):
    - command
    - command
reports (p. 148):
  report-group-name-or-arn:
    files (p. 148):
      - location
      - location
    base-directory (p. 149): location
    discard-paths (p. 149): no | yes
    file-format (p. 148): report-format
artifacts (p. 149):
  files (p. 149):
    - location
    - location
  name (p. 149): artifact-name
  discard-paths (p. 150): no | yes
  base-directory (p. 150): location
secondary-artifacts (p. 151):
  artifactIdentifier:
    files (p. 149):
      - location
      - location
    name (p. 149): secondary-artifact-name
    discard-paths (p. 150): no | yes
    base-directory (p. 150): location
  artifactIdentifier:
    files (p. 149):
      - location
      - location
    discard-paths (p. 150): no | yes
    base-directory (p. 150): location
cache (p. 152):
  paths (p. 152):
    - path
    - path
```

构建规范包含以下内容：

## version

必需的映射。表示构建规范版本。建议使用 0.2。

### Note

虽然仍支持版本 0.1，但建议尽可能使用版本 0.2。有关更多信息，请参阅 [构建规范版本 \(p. 154\)](#)。)

## run-as

可选的序列。仅适用于 Linux 用户。指定在此buildspec文件中运行命令的Linux用户。run-as 授予指定用户读取和执行权限。当您在构建规范文件的顶部指定 run-as 时，它将全局应用于所有命令。如果您不希望为

所有构建规范文件命令指定一个用户，可以通过在其中一个 `phases` 语句块中使用 `run-as`，为一个阶段中的命令指定一个用户。如果未指定 `run-as`，则所有命令都将以根用户身份运行。

## env

可选的序列。表示一个或多个自定义环境变量的信息。

### Note

为了保护敏感信息，CodeBuild 日志中隐藏以下内容：

- AWS 访问密钥 ID。有关更多信息，请参阅 AWS Identity and Access Management 用户指南 中的 [管理 IAM 用户的访问密钥](#)。
- 使用参数存储指定的字符串。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。
- 使用 AWS Secrets Manager 指定的字符串。有关更多信息，请参阅 [密钥管理 \(p. 324\)](#)。

### ENV/壳牌

可选的序列。指定Linux或Windows操作系统支持的外壳。

对于Linux操作系统，支持的shell标签为：

- bash
- /bin/sh

对于Windows操作系统，支持的shell标签为：

- powershell.exe
- cmd.exe

### env/variables

在指定了 `env` 并且您希望定义纯文本格式的自定义环境变量时必需。包含一个映射 `key/value` 标度，每个映射代表纯文本中的单个自定义环境变量。`key` 是自定义环境变量的名称，以及 `value` 是变量的值。

### Important

强烈建议不要将敏感值（尤其是 AWS 访问密钥 ID 和秘密访问密钥）存储在环境变量中。使用 CodeBuild 控制台和 AWS CLI 等工具能够以纯文本格式显示环境变量。对于敏感值，建议改用 `parameter-store` 或 `secrets-manager` 映射，如本节后面所述。

您设置的任何环境变量都将替换现有的环境变量。例如，如果靠泊装置图像已经包含名称为“环境变量”的环境变量 `MY_VAR` 有一个价值 `my_value`，并且您设置了名为 `MY_VAR` 有一个价值 `other_value`，然后 `my_value` 替换为 `other_value`...同样，如果靠泊装置图像已经包含名称的环境变量 `PATH` 有一个价值 `/usr/local/sbin:/usr/local/bin`，并且您设置了名为 `PATH` 有一个价值 `$PATH:/usr/share/ant/bin`，然后 `/usr/local/sbin:/usr/local/bin` 替换为字母值 `$PATH:/usr/share/ant/bin`。

不要设置任何以下列名称的环境变量: `CODEBUILD_...` 此前缀保留为内部使用。

如果具有相同名称的环境变量在多处都有定义，则应按照如下方式确定其值：

- 构建操作调用开始时的值优先级最高。您可以在创建构建时添加或覆盖环境变量。有关更多信息，请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#)。)
- 构建项目定义中的值优先级次之。您可以在创建或编辑项目时在项目级别添加环境变量。有关更多信息，请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#) 和 [更改 AWS CodeBuild 中构建项目的设置 \(p. 236\)](#)。
- 构建规范声明中的值优先级最低。

### env/parameter-store

在指定了 `env` 并且您要检索存储在 Amazon EC2 Systems Manager Parameter Store 中的自定义环境变量时必需。包含一个映射 `key/value` 标量值，每个映射表示存储在单个自定义环境变量中的一个自

定义环境变量 Amazon EC2 Systems Manager 参数存储。**key** 您在构建命令后面使用的名称，以指定此自定义环境变量，以及 **value** 是存储在存储在中的自定义环境变量的名称 Amazon EC2 Systems Manager 参数存储。要存储敏感值，请参阅 [系统管理器参数存储](#) 和 [巡视：创建和测试字符串参数（控制台）](#) 在 Amazon EC2 Systems Manager 用户指南。

#### Important

要允许 CodeBuild 检索存储在 Amazon EC2 Systems Manager Parameter Store 中的自定义环境变量，您必须将 `ssm:GetParameters` 操作添加到您的 CodeBuild 服务角色。有关更多信息，请参阅 [创建 CodeBuild 服务角色 \(p. 357\)](#)。)

从 Amazon EC2 Systems Manager Parameter Store 检索到的任何环境变量都将替换现有环境变量。例如，如果靠泊装置图像已经包含名称为“环境变量”的环境变量 `MY_VAR` 有一个价值 `my_value`，并且您检索名为 `MY_VAR` 有一个价值 `other_value`，然后 `my_value` 替换为 `other_value`...同样，如果靠泊装置图像已经包含名称的环境变量 `PATH` 有一个价值 `/usr/local/sbin:/usr/local/bin`，并且您检索名为 `PATH` 有一个价值 `$PATH:/usr/share/ant/bin`，然后 `/usr/local/sbin:/usr/local/bin` 替换为字母值 `$PATH:/usr/share/ant/bin`。

不要将任何环境变量存储在以 `CODEBUILD_`...此前缀保留为内部使用。

如果具有相同名称的环境变量在多处都有定义，则应按照如下方式确定其值：

- 构建操作调用开始时的值优先级最高。您可以在创建构建时添加或覆盖环境变量。有关更多信息，请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#)。)
- 构建项目定义中的值优先级次之。您可以在创建或编辑项目时在项目级别添加环境变量。有关更多信息，请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\) 和更改 AWS CodeBuild 中构建项目的设置 \(p. 236\)](#)。
- 构建规范声明中的值优先级最低。

#### env/secrets-manager

在指定了 `env` 并且您要检索存储在 AWS Secrets Manager Parameter Store 中的自定义环境变量时必需。使用以下模式指定 Secrets Manager reference-key：

`secret-id:json-key:version-stage:version-id`

- `secret-id`: 用作密钥的唯一标识符的名称或 Amazon 资源名称 (ARN)。要访问您的 AWS 账户中的密钥，只需指定密钥名称。要访问其他 AWS 账户中的密钥，请指定密钥 ARN。
- `json-key`: 指定要检索其值的键值对的键名称。如果您不指定 `json-key`，CodeBuild 会索整个密钥文本。
- `version-stage`: 指定要按附加到版本的暂存标签检索的密钥版本。暂存标签用于在轮换过程中跟踪不同版本。如果您使用 `version-stage`，不要指定 `version-id`...如果您没有指定版本阶段或版本 ID，则默认情况下，将检索版本阶段值的版本 `AWSCURRENT`。
- `version-id`: 指定要使用的密钥版本的唯一标识符。如果您指定 `version-id`，不要指定 `version-stage`...如果没有指定版本阶段或版本 ID，则默认情况下，将检索 `awscurrent` 版本阶段值的版本。

有关更多信息，请参阅 [AWS Secrets Manager 用户指南](#) 中的什么是 AWS Secrets Manager。

#### env/exported-variables

可选的映射。用于列出您要导出的环境变量。指定要在单独行中导出的每个变量的名称 `exported-variables`...在构建期间，您要导出的变量必须在容器中可用。导出的变量可以是环境变量。

在构建期间，变量的值从 `install` 阶段开始可用。可以在 `install` 阶段开始和 `post_build` 阶段结束之间更新变量的值。在 `post_build` 阶段结束后，无法更改导出的变量的值。

#### Note

无法导出以下项：

- 构建项目中指定的 Amazon EC2 Systems Manager Parameter Store 密钥。
- 构建项目中指定的 Secrets Manager 密钥
- 以 `AWS_` 开头的环境变量。

### ENV/GIT凭据-助手

可选的映射。用于表明 CodeBuild 使用其 GIT凭据助手提供GIT凭据。yes 如果使用。否则为 no 或未指定。有关更多信息，请参阅 Git 网站上的 [gitcredentials](#)。

#### Note

由 Webhook 触发的公有 Git 存储库的构建不支持 `git-credential-helper`。

## proxy

可选的序列。如果是在显式代理服务器中运行构建，则用于表示设置。有关更多信息，请参阅 [在显式代理服务器中运行 CodeBuild \(p. 184\)](#)。)

### proxy/upload-artifacts

可选的映射。如果您希望显式代理服务器中的构建来上传构件，请设置为 yes。默认值为“no”。

### proxy/logs

可选的映射。要使显式代理服务器中的构建创建 CloudWatch 日志，请设置为 yes。默认值为“no”。

## phases

必需的序列。表示 CodeBuild 在构建的各个阶段将运行的命令。

#### Note

在构建规范版本 0.1 中，CodeBuild 将在构建环境中的默认 Shell 的单独实例中运行每条命令。这表示各个命令独立于其他所有命令而运行。因此，在默认情况下，您无法运行依赖所有上一个命令状态的单个命令（如更改目录或设置环境变量）。要绕开此限制，建议使用版本 0.2 来解决此问题。如果必须使用构建规范版本 0.1，建议使用 [构建环境中的 Shell 和命令 \(p. 167\)](#) 中的方法。

### phases/\*/run-as

可选的序列。在构建阶段中使用，以指定运行命令的 Linux 用户。如果还在构建规范文件的顶部为所有命令全局指定了 run-as，则阶段级别用户优先。例如，如果 run-as 全局指定用户 1，并且对于 install 阶段，只有一条 run-as 语句指定用户 2，则构建规范文件中的所有命令将以用户 1 的身份运行，但 install 阶段中的命令除外，这些命令以用户 2 的身份运行。

允许的构建阶段名称是：

### phases/install

可选的序列。表示 CodeBuild 在安装过程中将运行的命令（如果有）。建议使用仅适用于在构建环境中安装软件包的 install 阶段。例如，您可以使用此阶段来安装代码测试框架（如 Mocha 或 RSpec）。

### phases/install/runtime-versions

可选的序列。Ubuntu 标准映像 2.0 或更高版本以及 Amazon Linux 2 标准映像 1.0 或更高版本支持运行时版本。如果指定，则本节中必须包含至少一个运行时。使用特定版本指定运行时，主要版本后跟 .x 以指定 CodeBuild 使用该主要版本及其最新次要版本，或后跟 latest 以使用最新的主要版本和次要版本（例如 `java: openjdk11`、`ruby: 2.6`、`nodejs: 12.x` 或 `java: latest`）。您可以使用一个数字或一个环境变量来指定运行时。例如，如果您使用 Amazon Linux 2 标准映像 2.0，则下面指定安装版本 8 的 Java、python 版本 3 的最新次要版本以及 Ruby 的环境变量中包含的版本。有关更多信息，请参阅 [CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。)

```
phases:  
  install:  
    runtime-versions:  
      java: corretto8
```

```
python: 3.x
ruby: "$MY_RUBY_VAR"
```

您可以在构建规范文件的 `runtime-versions` 部分中指定一个或多个运行时。如果您的运行时依赖于另一个运行时，您还可以在构建规范文件中指定其依赖运行时。如果您未在构建规范文件中指定任何运行时，CodeBuild 选择在您使用的映像中提供的默认运行时。如果指定一个或多个运行时，则 CodeBuild 仅使用这些运行时。如果未指定依赖运行时，则 CodeBuild 尝试为您选择依赖运行时。

如果两个指定运行时冲突，构建将失败。例如，`android: 29` 和 `java: openjdk11` 冲突，因此，如果同时指定了这两项，构建将失败。

可以指定以下支持的运行时。

#### Ubuntu 18.04 和 Amazon Linux 2 平台运行时

运行时名称	版本	特定版本
android	28.	android: 28
	29.	android: 29
dotnet	3.0	dotnet: 3.0

运行时名称	版本	特定版本
	3.1	dotnet: 3.1
Golang	1.12	golang: 1.12
	1.13	golang: 1.13
	1.14	golang: 1.14
nodejs	8	nodejs: 8
	10*	nodejs: 10

运行时名称	版本	特定版本
	[12]	nodejs: 12
java	openjdk8	java: openjdk8
	openjdk11	java: openjdk11
	corretto8	java: corretto8

运行时名称	版本	特定版本
	corretto11	<code>java: corretto11</code>
php	7.3	<code>php: 7.3</code>
	7.4	<code>php: 7.4</code>
python	3.7	<code>python: 3.7</code>

运行时名称	版本	特定版本
	3.8	<code>python: 3.8</code>
ruby	2.6	<code>ruby: 2.6</code>
	2.7	<code>ruby: 2.7</code>

#### Note

如果您指定 `runtime-versions` 部分并使用除 Ubuntu 标准映像 2.0 及更高版本或 Amazon Linux 2 (AL2) 标准映像 1.0 及更高版本以外的其他映像，构建过程将会发出警告“`Skipping install of runtimes. Runtime version selection is not supported by this build image.`”

#### phases/install/commands

`commands`: 可选的序列。包含一系列标量，其中每个标量表示 CodeBuild 在安装期间运行的单个命令。CodeBuild 按照命令列出的顺序依次运行每个命令。

#### phases/install/finally

可选的数据块。在 `finally` 语句块中指定的命令在 `commands` 语句块命令之后执行。即便 `commands` 语句块命令失败，仍会执行 `finally` 语句块命令。例如，如果 `commands` 语句块包含三条命令，而第一条命令失败了，则 CodeBuild 会跳过剩下的两条命令来运行 `finally` 语句块中的任何命令。当 `commands` 和 `finally` 语句块中的所有命令都成功运行后，此阶段才算成功。如果某个阶段有任何命令失败，则该阶段失败。

### phases/pre\_build

可选的序列。表示 CodeBuild 在构建之前将运行的命令（如果有）。例如，可以使用此阶段登录 Amazon ECR，也可以安装 npm 依赖项。

#### phases/pre\_build/commands

如果指定 `pre_build`，则为必需的序列。包含一系列标量，其中每个标量表示 CodeBuild 在构建之前运行的单个命令。CodeBuild 按照命令列出的顺序依次运行每个命令。

#### phases/pre\_build/finally

可选的数据块。在 `finally` 语句块中指定的命令在 `commands` 语句块命令之后执行。即便 `commands` 语句块命令失败，仍会执行 `finally` 语句块命令。例如，如果 `commands` 语句块包含三条命令，而第一条命令失败了，则 CodeBuild 会跳过剩下的两条命令来运行 `finally` 语句块中的任何命令。当 `commands` 和 `finally` 语句块中的所有命令都成功运行后，此阶段才算成功。如果某个阶段有任何命令失败，则该阶段失败。

### phases/build

可选的序列。表示 CodeBuild 在生成过程中将运行的命令（如果有）。例如，您可以使用此阶段运行 Mocha、RSpec 或 sbt。

#### phases/build/commands

`commands`: 要求如果 `build` 已指定。包含一系列标量，其中每个标量表示 CodeBuild 在构建期间运行的单个命令。CodeBuild 按照命令列出的顺序依次运行每个命令。

#### phases/build/finally

可选的数据块。在 `finally` 语句块中指定的命令在 `commands` 语句块命令之后执行。即便 `commands` 语句块命令失败，仍会执行 `finally` 语句块命令。例如，如果 `commands` 语句块包含三条命令，而第一条命令失败了，则 CodeBuild 会跳过剩下的两条命令来运行 `finally` 语句块中的任何命令。当 `commands` 和 `finally` 语句块中的所有命令都成功运行后，此阶段才算成功。如果某个阶段有任何命令失败，则该阶段失败。

### phases/post\_build

可选的序列。表示 CodeBuild 在生成后将运行的命令（如果有）。例如，您可以使用 Maven 将构建项目打包成 JAR 或 WAR 文件，还可以将 Docker 映像推入到 Amazon ECR 中。然后，您可以通过 Amazon SNS 发送构建通知。

#### phases/post\_build/commands

`commands`: 要求如果 `post_build` 已指定。包含一系列标量，其中每个标量表示 CodeBuild 在构建之后运行的单个命令。CodeBuild 按照命令列出的顺序依次运行每个命令。

#### phases/post\_build/finally

可选的数据块。在 `finally` 语句块中指定的命令在 `commands` 语句块命令之后执行。即便 `commands` 语句块命令失败，仍会执行 `finally` 语句块命令。例如，如果 `commands` 语句块包含三条命令，而第一条命令失败了，则 CodeBuild 会跳过剩下的两条命令来运行 `finally` 语句块中的任何命令。当 `commands` 和 `finally` 语句块中的所有命令都成功运行后，此阶段才算成功。如果某个阶段有任何命令失败，则该阶段失败。

## Important

如果早期构建阶段中的命令失败，则一些构建阶段中的命令可能无法运行。例如，如果 `install` 阶段的命令失败，则 `pre_build`、`build` 和 `post_build` 阶段的命令都不会在构建的生命周期内运行。有关更多信息，请参阅 [构建阶段过渡 \(p. 262\)](#)。)

## reports

report-group-name-or-arn

可选的序列。指定报告将发送到的报告组。一个项目最多可具有五个报告组。指定现有报告组的 ARN 或新报告组的名称。如果您指定一个名称, CodeBuild 使用您的项目名称和以格式指定的名称创建报表组 <project-name>-<report-group-name>。有关详细信息,请参阅 [报告组命名 \(p. 283\)](#).

reports/<report-group>/files

必需的序列。表示由报告生成的测试结果的原始数据所在的位置。包含一系列标量,每个标量代表一个独立的位置, CodeBuild 可以查找相对于原始构建位置的测试文件,或者,如果已设置, base-directory。地点可以包括以下内容:

- 单个文件 (如 my-test-report-file.json)。
- 子目录中的单个文件 (例如 , *my-subdirectory*/my-test-report-file.json 或 *my-parent-subdirectory*/*my-subdirectory*/my-test-report-file.json)。
- '\*/\*' 表示所有文件均采用递归方式。
- *my-subdirectory*/\* 表示子目录中的所有文件,名称为 *my-subdirectory*。
- *my-subdirectory*/\*\*/\* 从名为的子目录递归地表示所有文件 *my-subdirectory*。

reports/<report-group>/file-format

可选的映射。表示报告文件格式。如果未指定, 则使用 JUNITXML。此值不区分大小写。可能的值为 >、=、<。

测试报告

CUCUMBERJSON

Cucumber JSON

JUNITXML

JUnit XML

NUNITXML

NUnit XML

NUNIT3XML

NUnit3XML(三元XML)

TESTNGXML

TestNG XML

VISUALSTUDIOOTRX

Visual Studio TRX

代码覆盖报告

CLOVERXML

三叶草XML

COBERTURAXML

CoberturaXML(CoberturaXML)

JACOCOXML

JaCoCoXML(JaCoCoXML)

SIMPLECOV

SimpleCovJSON(简单CovJSON)

### Note

reports/<report-group>/base-directory

可选的映射。表示一个或多个顶级目录（相对于原始构建位置），CodeBuild 使用该目录确定在何处查找原始测试文件。

reports/<report-group>/discard-paths

：可选。指定是否在输出中展开报告文件目录。如果未指定此项或包含 no，则将输出报告文件，并且其目录结构保持不变。如果此项包含 yes，则将所有测试文件放置在同一个输出目录中。例如，如果测试结果的路径为 com/myapp/mytests/TestResult.xml，指定 yes 会将此文件放置在 /TestResult.xml 中。

## artifacts

可选的序列。表示有关 CodeBuild 可在何处查找构建输出以及 CodeBuild 如何进行准备以便将其上传到 S3 输出存储桶的信息。如果出现以下情况，则不需要此序列，例如，您正在构建或将 Docker 映像推送到 Amazon ECR，或者正在运行源代码上的单元测试，但并未构建源代码。

artifacts/files

必需的序列。表示包含构建环境中的输出项目的位置。包含一系列标量，其中每个标量表示一个相对于原始构建位置或基本目录（如果已设置）的单独位置，CodeBuild 可在此处查找构建输出项目。位置可包含以下内容：

- 单个文件（如 my-file.jar）。
- 子目录中的单个文件（例如，*my-subdirectory*/my-file.jar 或 *my-parent-subdirectory*/*my-subdirectory*/my-file.jar）。
- '\*\*/\*' 表示所有文件均采用递归方式。
- *my-subdirectory*/\* 代表名为 *my-subdirectory*。
- *my-subdirectory*/\*\*/\* 代表从名为名的子目录递归开始的所有文件 *my-subdirectory*。

您在指定构建输出项目位置时，CodeBuild 可在构建环境中查找原始构建位置。您无需将包含路径的构建项目输出位置放在原始构建位置的前面，或指定 ./ 或类似。如果您需要了解此位置的路径，则可以在构建过程中运行命令 echo \$CODEBUILD\_SRC\_DIR。各个构建环境的位置可能略有不同。

artifacts/name

可选的名称。为构建构件指定名称。将在以下任一条件为 true 时使用此名称。

- 使用 CodeBuild API 创建构建，并于更新项目、创建项目或启动构建后，在 ProjectArtifacts 对象上设置 overrideArtifactName 标记。
- 使用 CodeBuild 控制台创建您的构建，在构建规范文件中指定名称，并在创建或更新项目时选择 Enable semantic versioning（启用语义版本控制）。有关更多信息，请参阅 [创建构建项目（控制台）\(p. 190\)](#)。）

可以在构建规范文件中指定在构建时计算得出的名称。构建规范文件中指定的名称使用 Shell 命令语言。例如，您可以将日期和时间附加到您的项目名称后面，以便它始终是唯一的。构件名称唯一防止覆盖构件。有关更多信息，请参阅 [Shell 命令语言](#)。

这是追加有构件创建日期的构件名称的示例。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
```

```
files:  
  - '**/*'  
name: myname-$(date +%Y-%m-%d)
```

这是使用 CodeBuild 环境变量的构件名称的示例。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)。)

```
version: 0.2  
phases:  
  build:  
    commands:  
      - rspec HelloWorld_spec.rb  
artifacts:  
  files:  
    - '**/*'  
  name: myname-$AWS_REGION
```

这是使用 CodeBuild 环境变量且追加了构件创建日期的构件名称的示例。

```
version: 0.2  
phases:  
  build:  
    commands:  
      - rspec HelloWorld_spec.rb  
artifacts:  
  files:  
    - '**/*'  
  name: $AWS_REGION-$(date +%Y-%m-%d)
```

#### artifacts/discard-paths

：可选。指定是否在输出中展平构建构件目录。如果未指定此项或包含 no，则将输出构建构件，并且其目录结构保持不变。如果此项包含 yes，则将所有构建构件放置在同一个输出目录中。例如，如果构建输出构件中某个文件的路径为 com/mycompany/app/HelloWorld.java，则指定 yes 会将此文件放置在 /HelloWorld.java 中。

#### artifacts/base-directory

可选的映射。表示相对于原始构建位置的一个或多个顶级目录，CodeBuild 使用这些目录确定在构建输出项目中包含哪些文件和子目录。有效值包括：

- 单个顶级目录（如 my-directory）。
- 'my-directory\*' 表示名称以 my-directory 为开头的所有顶级目录。

构建输出项目中不包含映射顶级目录，只包含其文件和子目录。

您可以使用 files 和 discard-paths 进一步限制包含哪些文件和子目录。例如，对于以下目录结构：

```
|-- my-build1  
|   '-- my-file1.txt  
`-- my-build2  
    |-- my-file2.txt  
    '-- my-subdirectory  
        '-- my-file3.txt
```

对于以下 artifacts 序列：

```
artifacts:  
  files:  
    - '/my-file3.txt'
```

```
base-directory: my-build2
```

以下子目录和文件应包含在构建输出项目中：

```
my-subdirectory
`-- my-file3.txt
```

对于以下 artifacts 序列：

```
artifacts:
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

以下文件应包含在构建输出项目中：

```
|-- my-file1.txt
|-- my-file2.txt
`-- my-file3.txt
```

#### artifacts/secondary-artifacts

可选的序列。将一个或多个构件定义表示为构件标识符与构件定义之间的映射。此块中的每个构件标识符都必须与项目的 secondaryArtifacts 属性中定义的构件匹配。每个单独的定义与上面的 artifacts 块具有相同的语法。

##### Note

TheThe the [artifacts/files \(p. 149\)](#) 即使只有定义的次要伪影，也需要序列。

例如，如果项目具有以下结构：

```
{
  "name": "sample-project",
  "secondaryArtifacts": [
    {
      "type": "S3",
      "location": "output-bucket1",
      "artifactIdentifier": "artifact1",
      "name": "secondary-artifact-name-1"
    },
    {
      "type": "S3",
      "location": "output-bucket2",
      "artifactIdentifier": "artifact2",
      "name": "secondary-artifact-name-2"
    }
  ]
}
```

则 Buildpec 如下所示：

```
version: 0.2

phases:
build:
  commands:
    - echo Building...
artifacts:
  files:
```

```
- '*/**'
secondary-artifacts:
  artifact1:
    files:
      - directory/file1
      name: secondary-artifact-name-1
  artifact2:
    files:
      - directory/file2
      name: secondary-artifact-name-2
```

## cache

可选的序列。表示 CodeBuild 可在何处准备用于将缓存上传到 S3 缓存存储桶的文件的相关信息。如果项目的缓存类型为 No Cache，则不需要此序列。

### cache/paths

必需的序列。表示缓存的位置。包含一系列标量，其中每个标量表示一个相对于原始构建位置或基本目录（如果已设置）的单独位置，CodeBuild 可在此处查找构建输出项目。位置可包含以下内容：

- 单个文件（如 `my-file.jar`）。
- 子目录中的单个文件（例如，`my-subdirectory/my-file.jar` 或 `my-parent-subdirectory/my-subdirectory/my-file.jar`）。
- `'*/**'` 表示所有文件均采用递归方式。
- `my-subdirectory/*` 表示子目录中的所有文件，名称为 `my-subdirectory`。
- `my-subdirectory/**/*` 从名为的子目录递归地表示所有文件 `my-subdirectory`。

### Important

因为构建规范声明必须为有效的 YAML，所以构建规范声明中的空格至关重要。如果构建规范声明中的空格数量无效，则构建可能会立即失败。您可以使用 YAML 验证程序测试构建规范声明是否为有效的 YAML。

如果您在创建或更新构建项目时使用 AWS CLI 或 AWS 开发工具包声明构建规范，则构建规范必须是以 YAML 格式表示的单个字符串，包括必需的空格和换行转义字符。将在下一部分中提供示例。如果您使用 CodeBuild 或 AWS CodePipeline 控制台而不是 `buildspec.yml` 文件，则您只能插入适合 `build` 阶段的命令。不要使用前一语法，改为单行列出要在构建阶段运行的所有命令。对于多个命令，使用 `&&` 分开各个命令（例如 `mvn test && mvn package`）。

您可以使用 CodeBuild 或 CodePipeline 控制台而不是 `buildspec.yml` 文件指定构建环境中构建输出项目的位置。您不要使用前一语法，而是要使用单行列出所有位置。对于多个位置，使用逗号分开各个位置（例如 `buildspec.yml, target/my-app.jar`）。

## 构建规范示例

以下是 `buildspec.yml` 文件的示例。

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
```

```

    - echo Entered the install phase...
    - apt-get update -y
    - apt-get install -y maven
  finally:
    - echo This always runs even if the update or install command fails
pre_build:
  commands:
    - echo Entered the pre_build phase...
    - docker login -u User -p $LOGIN_PASSWORD
  finally:
    - echo This always runs even if the login command fails
build:
  commands:
    - echo Entered the build phase...
    - echo Build started on `date`
    - mvn install
  finally:
    - echo This always runs even if the install command fails
post_build:
  commands:
    - echo Entered the post_build phase...
    - echo Build completed on `date`

reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
    files:
      - "**/*"
    base-directory: 'target/tests/reports'
    discard-paths: no
  reportGroupCucumberJson:
    files:
      - 'cucumber/target/cucumber-tests.xml'
    discard-paths: yes
    file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
    - target/messageUtil-1.0.jar
discard-paths: yes
secondary-artifacts:
  artifact1:
    files:
      - target/artifact-1.0.jar
    discard-paths: yes
  artifact2:
    files:
      - target/artifact-2.0.jar
    discard-paths: yes
cache:
  paths:
    - '/root/.m2/**/*'

```

以下是上一构建规范的示例，此规范与 AWS CLI 或 AWS 开发工具包一起使用并表示为单个字符串。

```
"version: 0.2\nnenv:\n  variables:\n    JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-amd64\\\n  parameter-store:\n    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n  phases:\n    install:\n      commands:\n        - echo Entered the install phase...\n        - apt-get update -y\n        - apt-get install -y maven\n      finally:\n        - echo This always runs even if the update or install command fails\n    pre_build:\n      commands:\n        - echo Entered the pre_build phase...\n        - docker login -u User -p $LOGIN_PASSWORD\n      finally:\n        - echo This always runs even if the login command fails\n    build:\n      commands:\n        - echo Entered the build phase...\n        - echo Build started on `date`\n        - mvn install\n      finally:\n        - echo This always runs even if the install command fails\n    post_build:\n      commands:\n        - echo Entered the post_build phase...\n        - echo Build completed on `date`\n  reports:\n    reportGroupJunitXml:\n      files:\n        - '**/*'\n    base-directory: 'target/tests/reports'\n    discard-paths: false\n"
```

```
reportGroupCucumberJson:\n    files:\n        - 'cucumber/target/cucumber-tests.xml'\n        file-format: CUCUMBERJSON\n\nartifacts:\n    files:\n        - target/messageUtil-1.0.jar\n\ndiscard-paths: yes\n    secondary-artifacts:\n        artifact1:\n            files:\n                - target/messageUtil-1.0.jar\n        discard-paths: yes\n        artifact2:\n            files:\n                - target/messageUtil-1.0.jar\n        discard-paths: yes\n    cache:\n        paths:\n            - '/root/.m2/**/*'"
```

以下是 build 阶段中命令的示例，与 CodeBuild 或 CodePipeline 控制台一起使用。

```
echo Build started on `date` && mvn install
```

在这些示例中：

- 将设置密钥为 `JAVA_HOME`，值为 `/usr/lib/jvm/java-8-openjdk-amd64` 的纯文本格式的自定义环境变量。
- 之后，将通过使用密钥 `LOGIN_PASSWORD` 在构建命令中引用您在 Amazon EC2 Systems Manager Parameter Store 中存储的一个名为 `dockerLoginPassword` 的自定义环境变量。
- 您无法更改这些构建阶段名称。将在此示例中运行的命令有 `apt-get update -y`、`apt-get install -y maven`（用于安装 Apache Maven）、`mvn install`（用于编译和测试源代码并将源代码打包到构建输出项目中，以及在构建输出构件的内部存储库中安装该项目）、`docker login`（用于使用与您在 Amazon EC2 Systems Manager Parameter Store 中设置的自定义环境变量 `dockerLoginPassword` 的值对应的密码登录 Docker）以及若干 `echo` 命令。此处包含的 `echo` 命令用于显示 CodeBuild 运行命令的方式以及运行命令的顺序。
- `files` 表示要上传到构建输出位置的文件。在这个例子中，CodeBuild 上传单个文件 `messageUtil-1.0.jar`。的 `messageUtil-1.0.jar` 文件可在名为的相对目录中找到 `target` 在构建环境中。由于指定了 `discard-paths: yes`，因此将直接上传 `messageUtil-1.0.jar`（而不上传到中间 `target` 目录）。文件名称 `messageUtil-1.0.jar` 和 `target` 的相对目录名称均基于 Apache Maven 如何创建并存储构建输出项目（仅针对此示例）。在您自己的方案中，这些文件名称和目录会有所不同。
- `reports` 表示在构建过程中生成报告的两个报告组：
- `arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1` 指定报告组的 ARN。测试框架生成的测试结果位于 `target/tests/reports` 目录中。文件格式为 `JunitXml`，路径不会从包含测试结果的文件中删除。
- `reportGroupCucumberJson` 指定一个新的报告组。如果项目名称为 `my-project`，则在运行构建时创建一个名为 `my-project-reportGroupCucumberJson` 的报告组。测试框架生成的测试结果位于 `cucumber/target/cucumber-tests.xml`。测试文件格式为 `CucumberJson` 并且路径将从包含测试结果的文件中删除。

## 构建规范版本

下表列出了构建规范版本以及版本间的变化。

版本	更改
-0.2	<ul style="list-style-type: none"> <li><code>environment_variables</code> 已重命名为 <code>env</code>。</li> <li><code>plaintext</code> 已重命名为 <code>variables</code>。</li> <li><code>artifacts</code> 的 <code>type</code> 属性已被弃用。</li> <li>在版本 0.1 中，AWS CodeBuild 将在构建环境内默认 Shell 的单独实例中运行每个构建命令。在版本 0.2 中，CodeBuild 将在构建环境中默认 Shell 的同一实例中运行所有构建命令。</li> </ul>
0.1 美元	这是构建规范格式的初始定义。

## 批量构建BuildSpec参考

This topic contains the build spec reference for batch build properties.

### batch

可选的映射。项目的批次构建设置。

批次/快速失败

: 可选。对于构建图表，此属性不会使用，并且始终是 `true`.

`false`

默认值。所有运行的构建都将完成。

`true`

如果某个构建失败，则所有运行的构建都将停止。

默认情况下，所有批次构建任务都会使用构建设置运行，例如 `env` 和 `phases`，在 BuildSpec 文件中指定。您可以通过指定不同的默认构建设置 `env` 数值或其他 BuildSpec 文件中的 `batch/<batch-type>/buildspec` 参数。

的内容 `batch` 属性根据指定的批次构建类型而不同。可能的批次构建类型为：

- [batch/build-graph \(p. 155\)](#)
- [batch/build-list \(p. 156\)](#)
- [batch/build-matrix \(p. 157\)](#)

### batch/build-graph

定义 A 构建图表。构建图表定义了与批次中其他任务相关的任务集。

批次/构建图表/建立规格

: 可选。要用于此任务的 BuildSpec 文件的路径和文件名。

批次/构建图表/取决于

此任务取决于的任务标识符阵列。在完成这些任务之前，此任务将不会运行。

批次/构建图表/恩夫

: 可选。构建任务的环境覆盖。

批次/内部图表/ENV/计算类型

要用于任务的计算类型的标识符。参见 计算类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

批次/内部图表/ENV/图像

要用于任务的图像标识符。参见 图像标识符 在 [the section called “CodeBuild 提供的 Docker 映像” \(p. 159\)](#) 可能的值。

批次/内部图表/ENV/特权模式

: 可选。布尔值，表示是否在靠泊装置容器内运行 docker 守护程序。设置为 `true` 只有在构建靠泊装置图像时才能使用构建项目。否则，尝试与 Docker 守护程序进行交互的构建将失败。默认设置为 `false`。

#### 批次/内部图表/ENV/类型

要用于任务的环境类型的标识符。参见 环境类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

#### 批次/内部图表/ENV/变量

构建环境中将存在的环境变量。有关更多信息，请参阅 [env/variables \(p. 139\)](#)。

#### 批次/构建图表/标识符

必填 任务的标识符。

以下是构建图表构建规格条目的示例:

```
batch:  
  build-graph:  
    - identifier: linux_small  
      env:  
        compute-type: BUILD_GENERAL1_SMALL  
    - identifier: linux_medium  
      env:  
        compute-type: BUILD_GENERAL1_MEDIUM  
      depend-on:  
        - linux_small  
    - identifier: linux_large  
      env:  
        compute-type: BUILD_GENERAL1_LARGE  
      depend-on:  
        - linux_medium
```

## batch/build-list

定义A 建立列表. 构建列表用于定义并行运行的许多任务。

#### 批次/建立列表/建立规格

: 可选。要用于此任务的BuildSpec文件的路径和文件名。

#### 批次/建立列表/恩夫

: 可选。构建任务的环境覆盖。

#### 批次/建立列表/ENV/计算类型

要用于任务的计算类型的标识符。参见 计算类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

#### 批次/建立列表/ENV/图像

要用于任务的图像标识符。参见 图像标识符 在 [the section called “CodeBuild 提供的 Docker 映像” \(p. 159\)](#) 可能的值。

#### 批次/建立列表/ENV/特权模式

: 可选。布尔值，表示是否在靠泊装置容器内运行docker守护程序。设置为 true 只有在构建靠泊装置图像时才能使用构建项目。否则，尝试与 Docker 守护程序进行交互的构建将失败。默认设置为 false。

#### 批次/建立列表/ENV/类型

要用于任务的环境类型的标识符。参见 环境类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

## 批次/建立列表/ENV/变量

构建环境中将存在的环境变量。有关更多信息，请参阅 [env/variables \(p. 139\)](#)。

### 批次/建立列表/标识符

: 可选。任务的标识符。

### 批次/建立列表/忽略-失败

: 可选。表示批次中是否能够忽略的布尔值。

`false`

默认值。如果一个构建任务失败，批次构建将立即失败。

`true`

如果一个构建任务失败，剩余的构建任务仍将运行。

以下是构建列表构建规格条目的示例：

```
batch:  
  fast-fail: false  
  build-list:  
    - identifier: linux_small  
      env:  
        compute-type: BUILD_GENERAL1_SMALL  
    - identifier: windows_medium  
      env:  
        type: WINDOWS_CONTAINER  
        image: aws/codebuild/windows-base:2.0  
        compute-type: BUILD_GENERAL1_MEDIUM
```

## [batch/build-matrix](#)

定义A 构建矩阵。构建矩阵用于定义与不同环境并行运行的任务。CodeBuild 为每个可能的环境配置创建单独的构建。

### 批次/构建矩阵/静态

静态属性适用于所有构建任务。

### 批次/构造-矩阵/静态/忽略-失败

: 可选。表示批次中是否能够忽略的布尔值。

`false`

默认值。如果一个构建任务失败，批次构建将立即失败。

`true`

如果一个构建任务失败，剩余的构建任务仍将运行。

### 批次/构造-矩阵/静态/恩夫

: 可选。构建任务的环境覆盖。

### 批次/构造-矩阵/静态/ENV/特权模式

: 可选。布尔值，表示是否在靠泊装置容器内运行docker守护程序。设置为 `true` 只有在构建靠泊装置图像时才能使用构建项目。否则，尝试与 Docker 守护程序进行交互的构建将失败。默认设置为 `false`。

### 批次/构造-矩阵/静态/ENV/类型

: 可选。要用于任务的环境类型的标识符。参见 环境类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

### 批次/构建矩阵/动态

动态属性定义构造矩阵。

### 批次/构造-矩阵/动态/建立规格

: 可选。要用于此任务的BuildSpec文件的路径和文件名。

### 批次/构造-矩阵/动态/恩夫

: 可选。构建任务的环境覆盖。

### 批次/构造-矩阵/动态/ENV/计算类型

要用于任务的计算类型的标识符。参见 计算类型 在 [the section called “构建环境计算类型” \(p. 166\)](#) 可能的值。

### 批次/构造-矩阵/动态/ENV/图像

: 可选。要用于任务的图像标识符。参见 图像标识符 在 [the section called “CodeBuild 提供的 Docker 映像” \(p. 159\)](#) 可能的值。

### 批次/构造-矩阵/动态/ENV/变量

构建环境中将存在的环境变量。有关更多信息，请参阅 [env/variables \(p. 139\)](#)。

例如，如果构建矩阵有两个图像和环境变量的三个值，例如：

```
batch:
  build-matrix:
    static:
      ignore-failure: false
      env:
        type: LINUX_CONTAINER
        privileged-mode: true
    dynamic:
      env:
        image:
          - aws/codebuild/amazonlinux2-x86_64-standard:3.0
          - aws/codebuild/windows-base:2.0
      variables:
        MY_VAR:
          - VALUE1
          - VALUE2
          - VALUE3
```

CodeBuild 将创建六个建立内容：

- aws/codebuild/amazonlinux2-x86\_64-standard:3.0 / MY\_VAR=VALUE1
- aws/codebuild/amazonlinux2-x86\_64-standard:3.0 / MY\_VAR=VALUE2
- aws/codebuild/amazonlinux2-x86\_64-standard:3.0 / MY\_VAR=VALUE3
- aws/codebuild/windows-base:2.0 / MY\_VAR=VALUE1
- aws/codebuild/windows-base:2.0 / MY\_VAR=VALUE2
- aws/codebuild/windows-base:2.0 / MY\_VAR=VALUE3

每个构建都将具有以下设置：

- `ignore-failure` 设置为 `false`
- `env/type` 设置为 `LINUX_CONTAINER`
- `env/privileged-mode` 设置为 `true`

## AWS CodeBuild 的构建环境参考

当您致电 AWS CodeBuild 要运行内部版本，您必须提供有关内部版本环境的信息。生成环境 是由操作系统、编程语言运行时和 CodeBuild 用于运行生成任务的工具组成的。有关构建环境如何工作的信息，请参阅 [CodeBuild 的工作原理 \(p. 3\)](#)。

构建环境包含 Docker 映像。有关信息，请参阅 Docker 文档网站上的 [Docker 词汇表](#)。

当您向 CodeBuild 提供有关构建环境的信息时，您可以在支持的存储库类型中指定 Docker 映像的标识符。其中包括 CodeBuild Docker 映像存储库（Docker Hub 中公开可用的映像）和您的 AWS 账户有权访问的 Amazon Elastic Container Registry (Amazon ECR) 存储库：

- 我们建议您使用 CodeBuild Docker 映像存储库中存储的 Docker 映像，因为它们已经过优化以用于此服务。有关更多信息，请参阅 [CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。)
- 要获取 Docker Hub 中存储的公开可用的 Docker 映像的标识符，请参阅 Docker 文档网站上的 [搜索存储库](#)。
- 要了解如何在您的 AWS 账户中使用 Amazon ECR 存储库中存储的 Docker 映像，请参阅 [Amazon ECR 示例 \(p. 50\)](#)。

除了 Docker 映像标识符，您还可指定生成环境将使用的一组计算资源。有关更多信息，请参阅 [构建环境计算类型 \(p. 166\)](#)。)

### 主题

- [CodeBuild 提供的 Docker 映像 \(p. 159\)](#)
- [构建环境计算类型 \(p. 166\)](#)
- [构建环境中的 Shell 和命令 \(p. 167\)](#)
- [构建环境中的环境变量 \(p. 168\)](#)
- [构建环境中的后台任务 \(p. 170\)](#)

## CodeBuild 提供的 Docker 映像

AWS CodeBuild 管理 CodeBuild 和 AWS CodePipeline 控制台中可用的以下 Docker 镜像。

Platform	映像标识符	Definition
Amazon Linux 2	<code>aws/codebuild/amazonlinux2-x86_64-standard:3.0</code>	<a href="#">al2/standard/3.0</a>
Amazon Linux 2	<code>aws/codebuild/amazonlinux2-x86_64-standard:2.0</code>	<a href="#">al2/standard/2.0</a>
Amazon Linux 2	<code>aws/codebuild/amazonlinux2-aarch64-standard:1.0</code>	<a href="#">al2/aarch64/standard/1.0</a>

Platform	映像标识符	Definition
Ubuntu 18.04	aws/codebuild/standard:4.0	<a href="#">ubuntu/standard/4.0</a>
Ubuntu 18.04	aws/codebuild/standard:3.0	<a href="#">ubuntu/standard/3.0</a>
Ubuntu 18.04	aws/codebuild/standard:2.0 <sup>1</sup>	<a href="#">ubuntu/标准/2.0</a>
Windows Server Core 2016	aws/codebuild/windows-base:2.0	不适用*
Windows Server Core 2019 (Windows Server Core 2019)	aws/codebuild/windows-base:2019-1.0	不适用*

<sup>1</sup> 2020年6月之后不再维持。

缓存每个映像的最新版本。如果您指定了更具体的版本，则 CodeBuild 预置该版本而不是缓存版本。这可能会导致构建时间更长。例如，要受益于缓存，请指定 `aws/codebuild/amazonlinux2-x86_64-standard:3.0` 而不是更精细的版本，例如 `aws/codebuild/amazonlinux2-x86_64-standard:3.0-1.0.0`。

您可以在构建规范文件的 `runtime-versions` 部分中指定一个或多个运行时。如果您的运行时依赖于另一个运行时，您还可以在构建规范文件中指定其依赖运行时。如果您未在构建规范文件中指定任何运行时，CodeBuild 选择在您使用的映像中提供的默认运行时。如果指定一个或多个运行时，则 CodeBuild 仅使用这些运行时。如果未指定依赖运行时，则 CodeBuild 尝试为您选择依赖运行时。有关更多信息，请参阅 [Specify runtime versions in the buildspec file](#)。

在构建规范文件的 `runtime-versions` 部分中指定运行时期间，可以指定特定版本、特定主要版本和最新次要版本或最新版本。下表列出了可用的运行时及其指定方式。

#### Ubuntu 18.04 和 Amazon Linux 2 平台运行时

运行时名称	版本	特定版本
android	28.	<code>android: 28</code>
	29.	<code>android: 29</code>

运行时名称	版本	特定版本
dotnet	3.0	dotnet: 3.0
	3.1	dotnet: 3.1
Golang	1.12	golang: 1.12
	1.13	golang: 1.13
	1.14	golang: 1.14

运行时名称	版本	特定版本
nodejs	8	nodejs: 8
	10*	nodejs: 10
	[12]	nodejs: 12
java	openjdk8	java: openjdk8
	openjdk11	java: openjdk11

运行时名称	版本	特定版本
	corretto8	java: corretto8
	corretto11	java: corretto11
php	7.3	php: 7.3
	7.4	php: 7.4

运行时名称	版本	特定版本
python	3.7	python: 3.7
	3.8	python: 3.8
ruby	2.6	ruby: 2.6
	2.7	ruby: 2.7

Note

`aws/codebuild/amazonlinux2-aarch64-standard:1.0` 映像不支持 Android 运行时 (ART)。

Windows Server Core 2016 的基本映像包含以下运行时。

运行时名称	版本 <code>windows-base:2.0</code>
dotnet	2.2、3.1
golang	1.13
nodejs	10.18、12.14
java	openjdk11
php	7.3、7.4
python	3.7
ruby	2.6

WindowsServerCore2019的基础图像包含以下运行时。

运行时名称	版本 <code>windows-base:2019-1.0</code>
dotnet	3.1.3
golang	1.14
nodejs	12.18
java	corretto11
php	7.4.7
powershell/	7.0.2
python	3.8.3
ruby	2.7

Note

WindowsServerCore2016和WindowsServerCore2016平台的基础图像位于 美国东部 ( 弗吉尼亚北部 ) , 美国东部 ( 俄亥俄州 ) , 美国西部 ( 奥勒冈 ) 、 和 欧洲 ( 爱尔兰 ) 仅地区。

在 `install` 构建阶段，您可以使用构建规范来安装其他组件（例如，AWS CLI、Apache Maven、Apache Ant、Mocha、RSpec 或类似组件）。有关更多信息，请参阅 [构建规范示例 \(p. 152\)](#)。）

CodeBuild 频繁更新 Docker 映像的列表。要获取最新列表，执行下列操作之一：

- 在 CodeBuild 控制台中的 Create build project (创建构建项目) 向导或 Edit Build Project (编辑构建项目) 页面中，对于 Environment image (环境映像)，选择 Managed image (托管映像)。从 Operating system (操作系统)、Runtime (运行时) 和 Runtime version (运行时版本) 下拉列表中进行选择。有关更多信息，请参阅 [创建构建项目 \(控制台\) \(p. 190\)](#)或[更改构建项目的设置 \(控制台\) \(p. 236\)](#)。

- 对于 AWS CLI，请运行 `list-curated-environment-images` 命令：

```
aws codebuild list-curated-environment-images
```

- 对于 AWS 开发工具包，请为您的目标编程语言调用 `ListCuratedEnvironmentImages` 操作。有关更多信息，请参阅 [AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 构建环境计算类型

AWS CodeBuild 为构建环境提供了以下可用内存、vCPU 和磁盘空间：

### 操作系统 (Linux)

计算类型	computeType 值	Memory	vCPU	磁盘空间	环境类型
build.general1.SMALL	GENERAL1_SSMALL	2GB	2.	64GB	LINUX_CONTAINER
build.general1.MEDIUM	GENERAL1_SGMEDIUM	4GB	4	128GB	LINUX_CONTAINER
build.general1.LARGE	GENERAL1_SLARGE	15GB	8	128GB	LINUX_CONTAINER
build.general1.XLARGE	GENERAL1_SXLARGE	25GB	32	50 GB	LINUX_GPU_CONTAINER
build.general1.2XLARGE	GENERAL1_S2XLARGE	16GB	8	50 GB	ARM_CONTAINER
build.general1.4XLARGE	GENERAL1_S4XLARGE	145GB	72	824 GB (SSD)	LINUX_CONTAINER

为每个构建环境列出的磁盘空间仅在 `CODEBUILD_SRC_DIR` 环境变量指定的目录中可用。

#### Note

某些环境和计算类型存在一些限制：

- 环境类型 `LINUX_GPU_CONTAINER` 仅在地区中可用 美国东部 ( 弗吉尼亚北部 ) , 美国西部 ( 俄勒冈 ), 加拿大 ( 中部 ), 欧洲 ( 爱尔兰 ), 欧洲 ( 伦敦 ), 欧洲 ( 法兰克福 ), 亚太区域 ( 东京 ), 亚太区域 ( 首尔 ), 亚太区域 ( 新加坡 ), 亚太区域 ( 悉尼 ), 中国 ( 北京 ) 、 和 中国 ( 宁夏 ).
- 环境类型 `ARM_CONTAINER` 仅在地区中可用 美国东部 ( 弗吉尼亚北部 ), 美国东部 ( 俄亥俄州 ), 美国西部 ( 俄勒冈 ), 欧洲 ( 爱尔兰 ), 亚太地区 ( 孟买 ), 亚太区域 ( 东京 ), 亚太区域 ( 悉尼 ) 、 和 欧洲 ( 法兰克福 ).
- 计算类型 `build.general1.2xlarge` 仅在地区中可用 美国东部 ( 弗吉尼亚北部 ), 美国东部 ( 俄亥俄州 ), 美国西部 ( 加利福尼亚北部 ), 美国西部 ( 俄勒冈 ), 加拿大 ( 中部 ), 南美洲 ( 圣保罗 ), 欧洲 ( 斯德哥尔摩 ), 欧洲 ( 爱尔兰 ), 欧洲 ( 伦敦 ), 欧洲 ( 巴黎 ), 欧洲 ( 法兰克福 ), 中东 ( 巴林 ), 亚太地区 ( 香港 ), 亚太区域 ( 东京 ), 亚太区域 ( 首尔 ), 亚太区域 ( 新加坡 ), 亚太区域 ( 悉尼 ), 亚太地区 ( 孟买 ), 中国 ( 北京 ) 、 和 中国 ( 宁夏 ).
- 对于计算类型 `build.general1.2xlarge` , 支持高达 100 GB 的未压缩 Docker 映像。

### 操作系统 (Windows)

计算类型	computeType 值	Memory	vCPU	磁盘空间	环境类型
build.general1.MEDIUM	GENERAL1_SGMEDIUM	4GB	4	128GB	WINDOWS_CONTAINER 窗口_服务器 _2019_容器

计算类型	computeType 值	Memory	vCPU	磁盘空间	环境类型
build.general	BUILD_GENERAL1_15GB	15GB	8	128GB	WINDOWS_CONTAINER 窗口_服务器 _2019_容器

#### Note

对于自定义构建环境映像，CodeBuild 在 Linux 和 Windows 中支持高达 50 GB 的未压缩的 Docker 映像，无论计算类型如何。要检查构建映像的大小，请使用 Docker 运行 `docker images REPOSITORY:TAG` 命令。

选择计算类型：

- 在 CodeBuild 控制台中，在 Create build project (创建构建项目) 向导或 Edit Build Project (编辑构建项目) 页面的 Environment (环境) 中展开 Additional configuration (其他配置)，然后从 Compute type (计算类型) 中选择一个选项。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)或[更改构建项目的设置 \(控制台\) \(p. 236\)](#)。
- 对于 AWS CLI，请运行 `create-project` 或 `update-project` 命令，指定 `environment` 对象的 `computeType` 值。有关更多信息，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)或[更改构建项目的设置 \(AWS CLI\) \(p. 243\)](#)。
- 对于 AWS 开发工具包，请为您的目标编程语言调用等效于 `CreateProject` 或 `UpdateProject` 的操作，指定 `environment` 对象的 `computeType` 等效值。有关更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

您可以使用 Amazon EFS 在构建容器中访问更多空间。有关更多信息，请参阅[适用于 AWS CodeBuild 的 Amazon Elastic File System 示例 \(p. 53\)](#)。如果希望在构建期间操作容器磁盘空间，则构建必须运行在特权模式下。

#### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

## 构建环境中的 Shell 和命令

您为 AWS CodeBuild 提供了一组命令，用于在构建的生命周期期间（例如，安装构建依赖项、测试和编译您的源代码）在构建环境中运行。可通过多种方式指定这些命令：

- 创建构建规范文件并将其包含在您的源代码中。在此文件中，指定您要在构建生命周期的每个阶段运行的命令。有关更多信息，请参阅[适用于 CodeBuild 的构建规范参考 \(p. 136\)](#)。
- 使用 CodeBuild 控制台创建构建项目。在 Insert build commands (插入构建命令) 中，对于 Build command (构建命令)，输入您要在 build 阶段运行的命令。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)。
- 使用 CodeBuild 控制台更改构建项目的设置。在 Insert build commands (插入构建命令) 中，对于 Build command (构建命令)，输入您要在 build 阶段运行的命令。有关更多信息，请参阅[更改构建项目的设置 \(控制台\) \(p. 236\)](#)。
- 使用 AWS CLI 或 AWS 开发工具包创建生成项目或更改生成项目的设置。参考包含构建规范文件以及您的命令的源代码，或指定一个包含同等构建规范文件的内容的字符串。有关更多信息，请参阅[创建构建项目 \(\) \(p. 189\)](#)或[更改构建项目的设置 \(\) \(p. 236\)](#)。
- 使用 AWS CLI 或 AWS 开发工具包开始构建，指定构建规范文件或一个包含同等构建规范文件内容的字符串。有关更多信息，请参阅[运行构建 \(\) \(p. 252\)](#)中 `buildspecOverride` 值的描述。

您可以指定任何 Shell 命令语言 (sh) 命令。在构建规范版本 0.1 中，CodeBuild 在构建环境内单独的实例中运行每个 Shell 命令。这表示各个命令独立于其他所有命令而运行。因此，在默认情况下，您无法运行依赖所有上一个命令状态的单个命令 (如更改目录或设置环境变量)。要绕开此限制，建议使用版本 0.2 来解决此问题。如果您必须使用版本 0.1，我们建议使用以下方法：

- 在包含您要在默认 Shell 的单个实例中运行的命令的源代码中包含一个 Shell 脚本。例如，您可以添加名为 my-script.sh 包含命令的源代码中，例如 cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;。然后，在 buildspec 文件中指定命令 ./my-script.sh。
- 在您的构建规范文件或者仅在 build 阶段的 Build commands (构建命令) 设置中，输入单个命令，该命令包括您要在默认 shell 的单个实例中运行的所有命令 (例如，cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd)。

如果 CodeBuild 遇到错误，那么与在默认 Shell 的实例中运行单个命令相比，错误更难排除。

在 Windows Server Core 图像中运行的命令使用 PowerShell shell。

## 构建环境中的环境变量

AWS CodeBuild 提供了您可以在构建命令中使用的多个环境变量。

### AWS\_DEFAULT\_REGION

的 AWS 运行构建的区域(例如, us-east-1)。此环境变量主要由 AWS CLI 使用。

### AWS\_REGION

的 AWS 运行构建的区域(例如, us-east-1)。此环境变量主要由 AWS 开发工具包使用。

### CODEBUILD\_BATCH\_BUILD\_标识符

批次构建中的构建的标识符。这在批次构建规范中规定。有关更多信息，请参阅 [the section called “批量构建参考规格” \(p. 155\)](#)。)

### CODEBUILD\_BUILD\_ARN代码

构建的Amazon资源名称(ARN)(例如, arn:aws:codebuild:*region-ID:account-ID*:build/  
codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE)。

### CODEBUILD\_BUILD\_ID(代码BUILD\_BUILD\_ID)

的 CodeBuild 内部版本的ID(例如, codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE)。

### CODEBUILD\_BUILD\_图像

的 CodeBuild 构建图像标识符(例如, aws/codebuild/standard:2.0)。

### CODEBUILD\_BUILD\_NUMBER代码

：项目的当前构建编号。

### CODEBUILD\_BUILD\_SCUCEDING代码

：无论当前构建是否成功。如果构建失败，设置为 0；如果构建成功，设置为 1。

### CODEBUILD\_输入

：启动构建的实体。如果 CodePipeline 启动了生成，那么这就是管道的名称 (例如 codepipeline/my-demo-pipeline)。如果 IAM 用户启动了生成，那么这就是用户的名称 (例如 MyUserName)。如果适用于 CodeBuild 的 Jenkins 插件启动了生成，那么这就是字符串 CodeBuild-Jenkins-Plugin。

### CODEBUILD\_KMS\_KEY\_ID(代码BUILD\_KMS\_KEY\_ID)

的标识符 AWS KMS 关键是 CodeBuild 正用于加密构建输出伪影(例如, arn:aws:kms:*region-ID:account-ID*:key/*key-ID* 或 alias/*key-alias*)。

#### CODEBUILD\_LOG\_PATH代码

中的日志流名称 CloudWatch Logs 用于构建。

#### CODEBUILD\_RESOLVED\_SOURCE\_VERSION代码

: 构建的源代码的版本标识符。其格式取决于源代码存储库：

- 对于 CodeCommit、GitHub、GitHub Enterprise Server 和 Bitbucket，这是提交 ID。对于这些存储库，CODEBUILD\_RESOLVED\_SOURCE\_VERSION 仅在 DOWNLOAD\_SOURCE 阶段之后可用。
- 对于 CodePipeline，这是 CodePipeline 提供的源代码修订。对于 CodePipeline，CODEBUILD\_RESOLVED\_SOURCE\_VERSION 环境变量可能不会始终可用。
- 对于 Amazon S3，这不适用。

#### CODEBUILD\_SOURCE\_REPO\_URL

: 输入项目或源代码存储库的 URL。对于 Amazon S3，这是 s3://，后跟存储桶名称和输入项目的路径。对于 CodeCommit 和 GitHub，这是存储库的克隆 URL。如果构建源自 CodePipeline，此环境变量可能为空。

对于辅助源，辅助源存储库 URL 的环境变量为

CODEBUILD\_SOURCE\_REPO\_URL\_<sourceIdentifier>, 其中 <sourceIdentifier> 是您创建的源标识符。

#### CODEBUILD\_来源\_版本

: 值的格式取决于源代码存储库。

- 对于 Amazon S3，这是与输入构件关联的版本 ID。
- 对于 CodeCommit，这是与要生成的源代码的版本相关联的提交 ID 或分支名称。
- 对于 GitHub、GitHub Enterprise Server 和 Bitbucket，这是与要生成的源代码的版本相关联的提交 ID、分支名称或标签名称。

#### Note

对于由 Webhook 拉取请求事件触发的 GitHub 或 GitHub Enterprise Server 构建，这是 pr/pull-request-number。

对于次要来源，次要来源版本的环境变量为 CODEBUILD\_SOURCE\_VERSION\_<sourceIdentifier>，其中 <sourceIdentifier> 是您创建的源标识符。有关更多信息，请参阅 [多输入源和输出构件示例 \(p. 131\)](#)。)

#### CODEBUILD\_SRC\_DIR代码

目录路径 CodeBuild 用于构建(例如, /tmp/src123456789/src)。

对于次要来源，次要来源目录路径的环境变量为 CODEBUILD\_SRC\_DIR\_<sourceIdentifier>，其中 <sourceIdentifier> 是您创建的源标识符。有关更多信息，请参阅 [多输入源和输出构件示例 \(p. 131\)](#)。)

#### CODEBUILD\_START\_TIME代码

: 指定构建为 Unix 时间戳的开始时间 (以毫秒为单位)。

#### CODEBUILD\_WEBHOOK\_ACTOR\_ACCOUNT\_ID代码

: 触发 Webhook 事件的用户的账户 ID。

#### CODEBUILD\_WEBHOOK\_BASE\_REF(代码BUILD\_WEBHOOK\_BASE\_REF)

: 触发当前构建的 Webhook 事件的基本引用名称。对于拉取请求，这是分支引用。

#### CODEBUILD\_WEBHOOK\_EVENT代码

: 触发当前构建的 Webhook 事件。

#### CODEBUILD\_WEBHOOK\_PREV\_COMMIT 代码

: 在触发当前构建的 Webhook 推送事件之前最新提交的 ID。

#### CODEBUILD\_WEBHOOK\_HEAD\_REF 代码

: 触发当前构建的 Webhook 事件的头部引用名称。它可以是分支引用或标签引用。

#### CODEBUILD\_WEBHOOK\_TRIGGER 代码

: 显示触发构建的 Webhook 事件。此变量仅适用于 Webhook 触发的构建。该值是从通过 GitHub、GitHub Enterprise Server 或 Bitbucket 发送到 CodeBuild 的有效负载解析的。该值的格式取决于触发构建的事件类型。

- 对于拉取请求触发的构建，这是 `pr/pull-request-number`。
- 对于通过创建新分支或将提交操作推送到分支而触发的构建，这是 `branch/branch-name`。
- 对于通过将标签推送到存储库而触发的构建，这是 `tag/tag-name`。

#### 住宅

此环境变量始终设置为 `/root`。

您也可以为构建环境提供您自己的环境变量。有关更多信息，请参阅以下主题：

- [将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)
- [创建构建项目 \(\) \(p. 189\)](#)
- [更改构建项目的设置 \(\) \(p. 236\)](#)
- [运行构建 \(\) \(p. 252\)](#)
- [构建规范参考 \(p. 136\)](#)

要列出构建环境中的所有可用环境变量，在构建期间，您可以运行 `printenv` 命令（针对基于 Linux 的构建环境）或 "Get-ChildItem Env:"（针对基于 Windows 的构建环境）。除之前列出的这些变量之外，以 `CODEBUILD_` 开始的环境变量供 CodeBuild 内部使用。它们不应用于您的构建命令。

#### Important

我们强烈建议不要使用环境变量存储敏感值，尤其是 AWS 访问密钥 ID 和秘密访问密钥。使用 CodeBuild 控制台和 AWS CLI 等工具能够以纯文本格式显示环境变量。

我们建议您将敏感值存储在 Amazon EC2 Systems Manager Parameter Store 中，然后从您的构建规范中检索它们。要存储敏感值，请参阅 [系统管理器参数存储](#) 和 [演练：创建并测试字符串参数\(控制台\)](#) 在 Amazon EC2 Systems Manager 用户指南。要检索它们，请参阅 [构建规范语法 \(p. 137\)](#) 中的 `parameter-store` 映射。

## 构建环境中的后台任务

您可以在构建环境中运行后台任务。要执行此操作，请在您的 `buildspec` 中，使用 `nohup` 命令将命令作为后台中的任务运行，即使构建过程已退出 Shell 也是如此。使用 `disown` 命令强制停止正在运行的后台任务。

示例：

- 启动后台进程并等待其稍后完成：

```
nohup sleep 30 & echo $! > pidfile
...
wait $(cat pidfile)
```

- 启动后台进程，但不等待其完成：

```
nohup sleep 30 & disown $!
```

- 启动后台进程并稍后将其终止：

```
nohup sleep 30 & echo $! > pidfile
...
kill $(cat pidfile)
```

## 使用 AWS CodeBuild 代理在本地测试和调试

本主题提供了有关如何运行 AWS CodeBuild 代理和订阅代理新版本通知的信息。

### 使用 CodeBuild 代理在本地计算机上测试和调试

您可以使用 AWS CodeBuild 代理在本地计算机上测试和调试构建。

要使用此代理，请执行以下操作：

1. 下载 [codebuild.sh](#) 脚本。
2. 运行脚本并指定容器映像和输出目录：

```
codebuild_build.sh [-i image_name] [-a artifact_output_directory] [options]
```

CodeBuild 代理可从 <https://hub.docker.com/r/amazon/aws-codebuild-local/> 获得。它的安全散列算法 (SHA) 签名为 94467b3eeac4184d28a38feb27a1530691527dd49c17e30ad1b6331d791e82f5。您可以通过此签名识别代理的版本。要查看代理的 SHA 签名，运行以下命令：

```
docker inspect amazon/aws-codebuild-local
```

### 接收有关新的 CodeBuild 代理版本的通知

您可以订阅 Amazon SNS 通知，这样便能在发布 AWS CodeBuild 代理的新版本时收到通知。按照此过程中的步骤操作来订阅这些通知。

#### 订阅 CodeBuild 代理通知

1. 从 <https://console.aws.amazon.com/sns/v3/home> 打开 Amazon SNS 控制台。
2. 在导航栏中，如果尚未选中它，请将 AWS 区域更改为 美国东部（弗吉尼亚北部）。您必须选择此 AWS 区域，因为您订阅的 Amazon SNS 通知是在此区域中创建的。
3. 在导航窗格中，选择 Subscriptions。
4. 选择 Create subscription。
5. 在 Create subscription (创建订阅) 中：

对于 Topic ARN，请使用以下 Amazon 资源名称 (ARN)：

```
arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
```

对于 Protocol (协议)，选择 Email (电子邮件) 或 SMS。

对于 Endpoint (终端节点)，选择要接收通知的位置 (电子邮件或 SMS)。输入电子邮件、地址或电话号码，包括区号。

选择 Create subscription (创建订阅)。

如果选择 Email (电子邮件) , 您会收到一封要求确认订阅的电子邮件。按照电子邮件中的指示完成订阅。

如果您不再希望接收这些通知 , 请按照此过程中的步骤操作来取消订阅。

#### 取消订阅 CodeBuild 代理通知

1. 从 <https://console.aws.amazon.com/sns/v3/home> 打开 Amazon SNS 控制台。
2. 在导航窗格中 , 选择 Subscriptions。
3. 选择订阅 , 并从 Actions (操作) 中 , 选择 Delete subscriptions (删除订阅)。请在提示您进行确认时选择 Delete。

# 将 AWS CodeBuild 与 Amazon Virtual Private Cloud 结合使用

通常，AWS CodeBuild 无法访问 VPC 中的资源。要支持访问，您必须在 CodeBuild 项目配置中提供额外的 VPC 特定配置信息。这包括 VPC ID、VPC 子网 ID 和 VPC 安全组 ID。支持 VPC 的构建随后就可以访问 VPC 中的资源。有关在 Amazon VPC 中设置 VPC 的更多信息，请参阅 [Amazon VPC 用户指南](#)。

## Note

Windows 不支持来自 CodeBuild 的 VPC 连接。

### 主题

- [使用案例 \(p. 173\)](#)
- [在 CodeBuild 项目中允许 Amazon VPC 访问 \(p. 173\)](#)
- [VPC 的最佳实践 \(p. 174\)](#)
- [排查 VPC 设置的问题 \(p. 175\)](#)
- [使用 VPC 终端节点 \(p. 175\)](#)
- [AWS CloudFormation VPC 模板 \(p. 177\)](#)
- [将 AWS CodeBuild 与代理服务器结合使用 \(p. 180\)](#)

## 使用案例

来自 AWS CodeBuild 构建的 VPC 连接使以下操作成为可能：

- 通过您的构建对私有子网上隔离的 Amazon RDS 数据库中的数据运行集成测试。
- 直接通过测试查询 Amazon ElastiCache 集群中的数据。
- 与托管于 Amazon EC2、Amazon ECS 或使用内部 Elastic Load Balancing 的服务上的内部 Web 服务交互。
- 从自托管的内部项目存储库 (如适用于 Python 的 PyPI、适用于 Java 的 Maven 和适用于 Node.js 的 npm) 检索依赖项。
- 访问 S3 存储桶中配置为仅允许通过 Amazon VPC 终端节点访问的对象。
- 利用与您的子网关联的 NAT 网关或 NAT 实例的弹性 IP 地址，来查询需要固定 IP 地址的外部 Web 服务。

您的构建可以访问您的 VPC 中托管的任何资源。

## 在 CodeBuild 项目中允许 Amazon VPC 访问

在您的 VPC 配置中包含以下设置：

- 对于 VPC ID，选择 CodeBuild 使用的 VPC ID。
- 对于 Subnets (子网)，选择具有 NAT 转换的私有子网，其中包括或具有指向 CodeBuild 使用的资源的路由。
- 对于 Security Groups (安全组)，选择 CodeBuild 用来支持对 VPC 中资源的访问的安全组。

要使用控制台创建构建项目，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)。当您创建或更改 CodeBuild 项目时，请在 VPC 中选择您的 VPC ID、子网和安全组。

要使用 AWS CLI 创建构建项目，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。如果您要将 AWS CLI 与 CodeBuild 结合使用，则 CodeBuild 用来代表 IAM 用户与服务交互的服务角色必须已附加策略。有关信息，请参阅[允许 CodeBuild 访问创建 VPC 网络接口时所需的 AWS 服务 \(p. 343\)](#)。

该 `vpcConfig` 对象应包括 `vpcId`, `securityGroupIds`、 和 `subnets`。

- `vpcId` 必填 CodeBuild 使用的 VPC ID。运行此命令可获取您区域中的所有 Amazon VPC ID 的列表：

```
aws ec2 describe-vpcs
```

- `subnets` 必填 包含 CodeBuild 使用的资源的子网 ID。要获取这些 ID，请运行此命令：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

#### Note

将 `us-east-1` 替换为您的区域。

- `securityGroupIds` 必填 CodeBuild 用来支持对 VPC 中的资源的访问的安全组 ID。要获取这些 ID，请运行此命令：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

#### Note

将 `us-east-1` 替换为您的区域。

## VPC 的最佳实践

在设置 VPC 以使用 CodeBuild 时，请使用此核对清单。

- 设置具有公有和私有子网以及一个 NAT 网关的 VPC。有关详细信息，请参阅[带有公共和私有子网\(NAT\)的VPC 在 AmazonVPC 用户指南](#)。

#### Important

您需要一个 NAT 网关或 NAT 实例以便将 CodeBuild 与您的 VPC 结合使用，从而使 CodeBuild 能够访问公有终端节点（例如，在运行构建时执行 CLI 命令）。您不能使用互联网网关代替 NAT 网关或 NAT 实例，因为 CodeBuild 不支持将弹性 IP 地址分配给其创建的网络接口，并且 Amazon EC2 不支持为在 Amazon EC2 实例启动之外创建的任何网络接口自动分配公有 IP 地址。

- 将多个可用区包含在您的 VPC 中。
- 确保您的安全组不允许您构建的入站(ingress)流量。CodeBuild 对出站流量没有具体要求，但您必须允许访问构建所需的任何互联网资源，例如 GitHub 或 Amazon S3。

有关详细信息，请参阅 [安全组规则](#) 在 AmazonVPC 用户指南。

- 为您的构建设置单独的子网。
- 当您设置 CodeBuild 项目以访问 VPC 时，请仅选择私有子网。

有关在 Amazon VPC 中设置 VPC 的更多信息，请参阅 [Amazon VPC 用户指南](#)。

有关使用 AWS CloudFormation 将 VPC 配置为使用 CodeBuild VPC 功能的更多信息，请参阅 [AWS CloudFormation VPC 模板 \(p. 177\)](#)。

## 排查 VPC 设置的问题

使用错误消息中显示的信息可帮助您确定、诊断和解决问题。

以下是一些指导方针，可帮助您解决常见问题 CodeBuild VPC 错误: Build does not have internet connectivity. Please check subnet network configuration.

1. 确保您的互联网网关已连接到 VPC。
2. 确保您的公有子网的路由表指向互联网网关。
3. 确保您的网络 ACL 允许流量流动。
4. 确保您的安全组允许流量流动。
5. 排查 NAT 网关的问题。
6. 确保私有子网的路由表指向 NAT 网关。
7. 确保 CodeBuild 用来代表 IAM 用户与服务交互的服务器角色具有此策略中的权限。有关更多信息，请参阅 [创建 CodeBuild 服务角色 \(p. 357\)](#)。 )

If CodeBuild 缺少权限，您可能会收到一个错误，Unexpected EC2 error: UnauthorizedOperation...如果出现此错误，可能会发生此错误：CodeBuild 没有 Amazon EC2 使用 VPC 的权限。

## 使用 VPC 终端节点

您可以通过将 AWS CodeBuild 配置为使用接口 VPC 终端节点来提高构建的安全性。接口终端节点由 PrivateLink 提供技术支持，该技术可用于通过私有 IP 地址私下访问 Amazon EC2 和 CodeBuild。PrivateLink 将托管实例、CodeBuild 和 Amazon EC2 之间的所有网络流量限制在 Amazon 网络以内。（托管实例无法访问 Internet。）而且，您无需 Internet 网关、NAT 设备或虚拟专用网关。不要求您配置 PrivateLink，但推荐进行配置。有关 PrivateLink 和 VPC 终端节点的更多信息，请参阅 Amazon VPC 用户指南 中的 [通过 PrivateLink 访问 AWS 服务](#)。

### 在您创建 VPC 终端节点前

在配置 AWS CodeBuild 的 VPC 端点之前，请注意以下限制。

#### Note

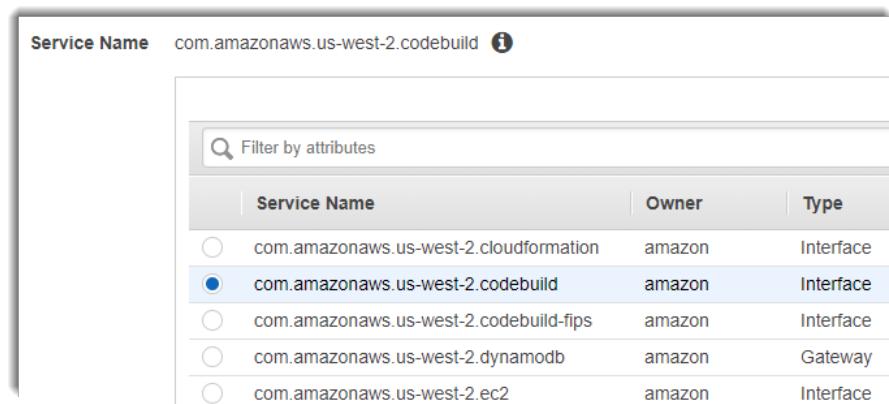
如果要将 CodeBuild 与不支持 Amazon VPC PrivateLink 连接的 AWS 服务结合使用，请使用 [NAT 网关](#)。

- VPC 终端节点仅通过 Amazon Route 53 支持 Amazon 提供的 DNS。如果您希望使用自己的 DNS，可以使用条件 DNS 转发。有关更多信息，请参阅 Amazon VPC 用户指南 中的 [DHCP 选项集](#)。

- VPC 终端节点当前不支持跨区域请求。确保您在存储构建输入和输出的任何 S3 存储桶所在的 AWS 区域内创建终端节点。您可以使用 Amazon S3 控制台或 [get-bucket-location](#) 命令来查找存储桶的位置。使用区域特定的 Amazon S3 终端节点访问存储桶（例如，`mybucket.s3-us-west-2.amazonaws.com`）。有关 Amazon S3 的区域特定终端节点的更多信息，请参阅 Amazon Web Services 一般参考 中的 [Amazon Simple Storage Service](#)。如果您使用 AWS CLI 向 Amazon S3 发起请求，请将默认区域设置为创建您的存储桶的区域，或在请求中使用 `--region` 参数。

## 创建 CodeBuild 的 VPC 终端节点

按照[创建接口终端节点](#)中的说明操作，创建终端节点 `com.amazonaws.region.codebuild`。这是用于 AWS CodeBuild 的 VPC 终端节点。



Service Name	Owner	Type
com.amazonaws.us-west-2.cloudformation	amazon	Interface
<b>com.amazonaws.us-west-2.codebuild</b>	amazon	Interface
com.amazonaws.us-west-2.codebuild-fips	amazon	Interface
com.amazonaws.us-west-2.dynamodb	amazon	Gateway
com.amazonaws.us-west-2.ec2	amazon	Interface

`region` 代表 CodeBuild 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 美国东部（俄亥俄州）区域。有关支持的 AWS 区域的列表，请参阅 AWS 一般参考 中的 [CodeBuild](#)。使用您在登录到 AWS 时指定的区域来预填充终端节点。如果更改您的区域，VPC 终端节点会相应地更新。

## 为 CodeBuild 创建 VPC 终端节点策略

您可以为 AWS CodeBuild 的 Amazon VPC 终端节点创建一个策略，在其中可以指定：

- 可执行操作的委托人。
- 可执行的操作。
- 可用于执行操作的资源。

以下示例策略指定所有委托人只能启动和查看 `project-name` 项目的构建。

```
{  
    "Statement": [  
        {  
            "Action": [  
                "codebuild>ListBuildsForProject",  
                "codebuild:StartBuild",  
                "codebuild:BatchGetBuilds"  
            ],  
            "Effect": "Allow",  
            "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",  
            "Principal": "*"  
        }  
    ]  
}
```

有关更多信息，请参阅 Amazon VPC 用户指南 中的[使用 VPC 终端节点控制对服务的访问](#)。

# AWS CloudFormation VPC 模板

AWS CloudFormation 使您能够预见性地反复创建和配置 AWS 基础设施部署，方式是使用模板文件批量创建和删除一系列资源的集合（视为一个堆栈）。有关更多信息，请参阅 [AWS CloudFormation 用户指南](#)。

下面是用于配置 VPC 以使用 AWS CodeBuild 的 AWS CloudFormation YAML 模板。此文件也可以在 [样品拉链](#)。

```
Description: This template deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone
    Type: String
    Default: 10.192.20.0/24

  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
```

```
    Value: !Ref EnvironmentName

InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet1CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PrivateSubnet2CIDR
    MapPublicIpOnLaunch: false
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc

NatGateway1:
  Type: AWS::EC2::NatGateway
```

```
Properties:  
  AllocationId: !GetAtt NatGateway1EIP.AllocationId  
  SubnetId: !Ref PublicSubnet1  
  
NatGateway2:  
  Type: AWS::EC2::NatGateway  
  Properties:  
    AllocationId: !GetAtt NatGateway2EIP.AllocationId  
    SubnetId: !Ref PublicSubnet2  
  
PublicRouteTable:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
    Tags:  
      - Key: Name  
        Value: !Sub ${EnvironmentName} Public Routes  
  
DefaultPublicRoute:  
  Type: AWS::EC2::Route  
  DependsOn: InternetGatewayAttachment  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    DestinationCidrBlock: 0.0.0.0/0  
    GatewayId: !Ref InternetGateway  
  
PublicSubnet1RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    SubnetId: !Ref PublicSubnet1  
  
PublicSubnet2RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PublicRouteTable  
    SubnetId: !Ref PublicSubnet2  
  
PrivateRouteTable1:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
    Tags:  
      - Key: Name  
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)  
  
DefaultPrivateRoute1:  
  Type: AWS::EC2::Route  
  Properties:  
    RouteTableId: !Ref PrivateRouteTable1  
    DestinationCidrBlock: 0.0.0.0/0  
    NatGatewayId: !Ref NatGateway1  
  
PrivateSubnet1RouteTableAssociation:  
  Type: AWS::EC2::SubnetRouteTableAssociation  
  Properties:  
    RouteTableId: !Ref PrivateRouteTable1  
    SubnetId: !Ref PrivateSubnet1  
  
PrivateRouteTable2:  
  Type: AWS::EC2::RouteTable  
  Properties:  
    VpcId: !Ref VPC  
    Tags:  
      - Key: Name
```

```
Value: !Sub ${EnvironmentName} Private Routes (AZ2)

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

NoIngressSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "no-ingress-sg"
    GroupDescription: "Security group with no ingress rule"
    VpcId: !Ref VPC

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  NoIngressSecurityGroup:
    Description: Security group with no ingress rule
    Value: !Ref NoIngressSecurityGroup
```

## 将 AWS CodeBuild 与代理服务器结合使用

您可以将 AWS CodeBuild 与代理服务器结合使用，以控制往来于 Internet 的 HTTP 和 HTTPS 流量。要使用代理服务器运行 CodeBuild，您需要在 VPC 的公有子网中安装代理服务器，并在私有子网中安装 CodeBuild。

在代理服务器中运行 CodeBuild 有两种主要使用案例：

- 它不再需要您的 VPC 中的 NAT 网关或 NAT 实例。
- 它允许您指定代理服务器中的实例可以访问的 URL 以及代理服务器拒绝访问的 URL。

您可以将 CodeBuild 与两种类型的代理服务器结合使用。对于这两种类型，代理服务器都在公有子网中运行，CodeBuild 在私有子网中运行。

- **显式代理**：如果使用显式代理服务器，则必须在项目级别在 CodeBuild 中配置 NO\_PROXY、HTTP\_PROXY 和 HTTPS\_PROXY 环境变量。有关更多信息，请参阅 [更改 AWS CodeBuild 中构建项目的设置 \(p. 236\)](#) 和 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。
- **透明代理**：如果使用透明代理服务器，则不需要特殊配置。

#### 主题

- [在代理服务器中运行 CodeBuild 所需的组件 \(p. 181\)](#)
- [在显式代理服务器中运行 CodeBuild \(p. 184\)](#)
- [在透明代理服务器中运行 CodeBuild \(p. 186\)](#)
- [在代理服务器中运行程序包管理器和其他工具 \(p. 187\)](#)

## 在代理服务器中运行 CodeBuild 所需的组件

您需要这些组件在透明或显式代理服务器中运行 AWS CodeBuild：

- VPC。
- 代理服务器的 VPC 中的一个公有子网。
- CodeBuild 的 VPC 中的一个私有子网。
- 一个 Internet 网关，允许 VPC 和 Internet 之间进行通信。

下图显示了组件的交互方式。



## 设置 VPC、子网和网络网关

在透明或显式代理服务器中运行 AWS CodeBuild 需要以下步骤。

1. 创建 VPC。有关信息，请参阅 Amazon VPC 用户指南 中的[创建 VPC](#)。
2. 在您的 VPC 中创建两个子网。一个是名为 Public Subnet 的公有子网，代理服务器将在其中运行。另一个是一个名为 Private Subnet 的私有子网，CodeBuild 将在其中运行。

有关信息，请参阅[在 VPC 中创建子网](#)。

3. 创建 Internet 网关，并将其连接到您的 VPC。有关更多信息，请参阅[创建并附加 Internet 网关](#)。
4. 向默认路由表添加一条规则，该规则将来自 VPC 的传出流量路由到 Internet 网关。有关信息，请参阅[在路由表中添加和删除路由](#)。

5. 向 VPC 的默认安全组添加一条规则，该规则允许来自 VPC (0.0.0.0/0) 的入站 SSH 流量 (0.0.0.0/0)。
6. 按照 Amazon EC2 用户指南 中的[使用启动实例向导启动实例](#)中的说明操作来启动 Amazon Linux 实例。当您运行该向导时，请选择以下选项：

- 在 Choose an Instance Type (选择实例类型) 中，选择一个 Amazon Linux Amazon 系统映像 (AMI)。
- 在 Subnet (子网) 中，选择您在本主题的前面步骤中创建的公有子网。如果您使用了建议的名称，则该名称是公有子网。
- 在 Auto-assign Public IP (自动分配公有 IP) 中，选择 Enable (启用)。
- 在 Configure Security Group (配置安全组) 页面上，对于 Assign a security group (分配安全组)，选择 Select an existing security group (选择现有安全组)。接下来，选择默认安全组。
- 选择 Launch (启动) 后，选择现有密钥对或创建密钥对。

选择所有其他选项的默认设置。

7. 您的 EC2 实例开始运行后，禁用源/目标检查。有关信息，请参阅 Amazon VPC 用户指南 中的[禁用源/目标检查](#)。
8. 在 VPC 中创建路由表。向路由表中添加一条规则，该规则将发往 Internet 的流量路由到您的代理服务器。将此路由表与私有子网关联。这是必需的，以便来自私有子网 (CodeBuild 在其中运行) 中的实例的出站请求始终通过代理服务器进行路由。

## 安装和配置代理服务器

有许多可供选择的代理服务器。Squid 是一个开源代理服务器，此处用于演示 AWS CodeBuild 如何在代理服务器中运行。您可以将相同的概念应用于其他代理服务器。

要安装 Squid，请通过运行以下命令使用 yum 存储库：

```
sudo yum update -y
sudo yum install -y squid
```

安装 Squid 后，请按照本主题后面的说明操作来编辑其 `squid.conf` 文件。

## 为 HTTPS 流量配置 Squid

对于 HTTPS，HTTP 流量封装在一个传输层安全性 (TLS) 连接中。Squid 使用一个名为 [SslPeekAndSplice](#) 的功能从包含请求的 Internet 主机的 TLS 启动中检索服务器名称指示 (SNI)。这是必需的，因此 Squid 不需要解密 HTTPS 流量。要启用 SslPeekAndSplice，Squid 需要一个证书。使用 OpenSSL 创建此证书：

```
sudo mkdir /etc/squid/ssl
cd /etc/squid/ssl
sudo openssl genrsa -out squid.key 2048
sudo openssl req -new -key squid.key -out squid.csr -subj "/C=XX/ST=XX/L=squid/O=squid/
CN=squid"
sudo openssl x509 -req -days 3650 -in squid.csr -signkey squid.key -out squid.crt
```

```
sudo cat squid.key squid.crt | sudo tee squid.pem
```

Note

对于 HTTP，Squid 不需要配置。它可以从所有 HTTP/1.1 请求消息中检索主机标头字段，该字段指定所请求的 Internet 主机。

## 在显式代理服务器中运行 CodeBuild

### 主题

- [将 Squid 配置为显式代理服务器 \(p. 184\)](#)
- [创建 CodeBuild 项目 \(p. 185\)](#)
- [显式代理服务器示例 squid.conf 文件 \(p. 185\)](#)

要在显式代理服务器中运行 AWS CodeBuild，您必须配置代理服务器以允许或拒绝进出外部网站的流量，然后配置 `HTTP_PROXY` 和 `HTTPS_PROXY` 环境变量。

## 将 Squid 配置为显式代理服务器

要将 Squid 代理服务器配置为显式，您必须对其 `/etc/squid/squid.conf` 文件进行以下修改：

- 删除以下默认访问控制列表 (ACL) 规则。

```
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
```

在您删除的默认 ACL 规则的位置添加以下内容。第一行允许来自您的 VPC 的请求。接下来的两行授予您的代理服务器访问 AWS CodeBuild 可能使用的目标 URL 的权限。修改最后一行中的正则表达式，以指定 AWS 区域中的 S3 存储桶或 CodeCommit 存储库。例如：

- 如果您的源是 Amazon S3，请使用命令 `acl download_src dstdom_regex .*\s3\.us-west-1\.amazonaws.com` 来授权访问 us-west-1 区域中的 S3 存储桶。
- 如果您的源是 AWS CodeCommit，请使用 `git-codecommit.<your-region>.amazonaws.com` 将 AWS 区域添加到允许列表中。

```
acl localnet src 10.1.0.0/16 #Only allow requests from within the VPC
acl allowed_sites dstdomain .github.com #Allows to download source from GitHub
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from Bitbucket
acl download_src dstdom_regex .*\.\amazonaws\.com #Allows to download source from Amazon S3 or CodeCommit
```

- 将 `http_access allow localnet` 替换为以下项：

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- 如果您希望构建上传日志和构件，请执行以下任一操作：

1. 在 `http_access deny all` 语句之前，插入以下语句。它们允许 CodeBuild 访问 CloudWatch 和 Amazon S3。必须能够访问 CloudWatch，以便 CodeBuild 可以创建 CloudWatch 日志。上传构件和 Amazon S3 缓存需要访问 Amazon S3。

```
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
```

```
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
```

- 保存 squid.conf 后，请执行以下操作：

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service squid restart
```

2. 将 proxy 添加到您的 buildspec 文件。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#)。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

#### Note

如果您收到一个 RequestError 超时错误，请参阅 [在代理服务器中运行 CodeBuild 时出现 RequestError 超时错误 \(p. 397\)](#)。

有关更多信息，请参阅本主题后面的 [显式代理服务器示例 squid.conf 文件 \(p. 185\)](#)。

## 创建 CodeBuild 项目

要使用显式代理服务器运行 AWS CodeBuild，请使用您为代理服务器创建的 EC2 实例的私有 IP 地址和项目级别的端口 3128 设置 HTTP\_PROXY 和 HTTPS\_PROXY 环境变量。私有 IP 地址看起来类似于 `http://your-ec2-private-ip-address:3128`。有关更多信息，请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#) 和 [更改 AWS CodeBuild 中构建项目的设置 \(p. 236\)](#)。

使用以下命令查看 Squid 代理服务器访问日志：

```
sudo tail -f /var/log/squid/access.log
```

## 显式代理服务器示例 squid.conf 文件

以下是为显式代理服务器配置的 squid.conf 文件的示例。

```
acl localnet src 10.0.0.0/16 #Only allow requests from within the VPC
# add all URLs to be whitelisted for download source and commands to be executed in build
environment
acl allowed_sites dstdomain .github.com      #Allows to download source from github
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from bitbucket
acl allowed_sites dstdomain ppa.launchpad.net #Allows to execute apt-get in build
environment
acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from S3 or
CodeCommit
acl SSL_ports port 443
acl Safe_ports port 80  # http
acl Safe_ports port 21  # ftp
```

```
acl Safe_ports port 443 # https
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT
#
# Recommended minimum Access Permission configuration:
#
# Deny requests to certain unsafe ports
http_access deny !Safe_ports
# Deny CONNECT to other than secure SSL ports
http_access deny CONNECT !SSL_ports
# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager
# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet allowed_sites
http_access allow localnet download_src
http_access allow localhost
# Add this for CodeBuild to access CWL end point, caching and upload artifacts S3 bucket
end point
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
# And finally deny all other access to this proxy
http_access deny all
# Squid normally listens to port 3128
http_port 3128
# Uncomment and adjust the following to add a disk cache directory.
#cache_dir ufs /var/spool/squid 100 16 256
# Leave core dumps in the first cache dir
coredump_dir /var/spool/squid
#
# Add any of your own refresh_pattern entries above these.
#
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern -i (/cgi-bin/|\.?) 0 0% 0
refresh_pattern . 0 20% 4320
```

## 在透明代理服务器中运行 CodeBuild

要在透明代理服务器中运行 AWS CodeBuild，您必须配置代理服务器，使其能够访问与其交互的网站和域。

## 将 Squid 配置为透明代理服务器

要将代理服务器配置为透明，您必须授予其访问您希望其访问的域和网站的权限。要使用透明代理服务器运行 AWS CodeBuild，您必须向它授予对 `amazonaws.com` 的访问权限。您还必须授予对 CodeBuild 使用的其他网站的访问权限。这些网站因您创建 CodeBuild 项目的方式而异。示例网站是用于存储库的网站，例如 GitHub、Bitbucket、Yum 和 Maven。要授予 Squid 访问特定域和网站的权限，请使用类似于以下内容的命令来更新 `squid.conf` 文件。此示例命令授予对 `amazonaws.com`、`github.com` 和 `bitbucket.com` 的访问权限。您可以编辑此示例以授予对其他网站的访问权限。

```
cat | sudo tee /etc/squid/squid.conf #EOF
visible_hostname squid
#Handling HTTP requests
http_port 3129 intercept
acl allowed_http_sites dstdomain .amazonaws.com
#acl allowed_http_sites dstdomain domain_name [uncomment this line to add another domain]
http_access allow allowed_http_sites
#Handling HTTPS requests
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl allowed_https_sites ssl::server_name .github.com
acl allowed_https_sites ssl::server_name .bitbucket.com
#acl allowed_https_sites ssl::server_name [uncomment this line to add another website]
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
http_access deny all
EOF
```

来自私有子网中的实例的传入请求必须重定向到 Squid 端口。Squid 在端口 3129 上侦听 HTTP 流量（而不是 80），并在端口 3130 上侦听 HTTPS 流量（而不是 443）。使用 `iptables` 命令可路由流量：

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service iptables save
sudo service squid start
```

## 创建 CodeBuild 项目

配置您的代理服务器之后，便可以将其与私有子网中的 AWS CodeBuild 结合使用，而无需进行更多配置。每个 HTTP 和 HTTPS 请求都经过公共代理服务器。使用以下命令查看 Squid 代理服务器访问日志：

```
sudo tail -f /var/log/squid/access.log
```

## 在代理服务器中运行程序包管理器和其他工具

要在代理服务器中执行一个工具，如程序包管理器，请执行以下操作：

1. 通过将语句添加到您的 `squid.conf` 文件中，将该工具添加到代理服务器的允许列表中。
2. 在 `buildspec` 文件中添加指向代理服务器的私有终端节点的命令行。

以下示例演示了如何为 apt-get、curl 和 maven 执行此操作。如果您使用其他工具，则相同的原则将适用。将它添加到 squid.conf 文件中的允许列表中，并向构建规范文件添加一个命令以使 CodeBuild 了解您的代理服务器的终端节点。

### 在代理服务器中运行 apt-get

1. 将以下语句添加到您的 squid.conf 文件中，以便将 apt-get 添加到代理服务器中的允许列表。前三行允许 apt-get 在构建环境中执行。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to execute in the build environment
acl apt_get dstdom_regex \.launchpad.net # Required for CodeBuild to execute apt-get in the build environment
acl apt_get dstdom_regex \.ubuntu.com    # Required for CodeBuild to execute apt-get in the build environment
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. 在构建规范文件中添加以下语句，以便 apt-get 命令在 /etc/apt/apt.conf.d/00proxy 中查找代理配置。

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";
Acquire::https::Proxy "http://<private-ip-of-proxy-server>:3128"; Acquire::ftp::Proxy
"http://<private-ip-of-proxy-server>:3128";' > /etc/apt/apt.conf.d/00proxy
```

### 在代理服务器中运行 curl

1. 将以下内容添加到您的 squid.conf 文件中，以便将 curl 添加到构建环境中的允许列表。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to execute apt-get in the build environment
acl allowed_sites dstdomain google.com # Required for access to a website. This example uses www.google.com.
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. 在构建规范文件中添加以下语句，以便 curl 使用专用代理服务器访问您添加到 squid.conf 的网站。在此示例中，网站为 google.com。

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

### 在代理服务器中运行 maven

1. 将以下内容添加到您的 squid.conf 文件中，以便将 maven 添加到构建环境中的允许列表。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to execute apt-get in the build environment
acl maven dstdom_regex \.maven.org # Allows access to the maven repository in the build environment
http_access allow localnet allowed_sites
http_access allow localnet maven
```

2. 在构建规范文件中添加以下语句。

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

# 使用 AWS CodeBuild 中的构建项目和构建

要开始使用，请执行[创建构建项目 \(\) \(p. 189\)](#)中的步骤，然后执行[运行构建 \(\) \(p. 252\)](#)中的步骤。有关构建项目和构建的更多信息，请参阅以下主题。

## 主题

- [使用构建项目 \(p. 189\)](#)
- [使用 AWS CodeBuild 中的构建 \(p. 251\)](#)

## 使用构建项目

构建项目 包括有关如何运行构建的信息，其中包括源代码获取位置、要使用的构建环境、要运行的构建命令和构建输出的存储位置。

在使用构建项目时，您可以执行以下任务：

## 主题

- [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)
- [Create a notification rule \(p. 209\)](#)
- [查看 AWS CodeBuild 中构建项目名称的列表 \(p. 211\)](#)
- [查看 AWS CodeBuild 中构建项目的详细信息 \(p. 213\)](#)
- [在 AWS CodeBuild 中构建缓存 \(p. 214\)](#)
- [创建 AWS CodeBuild 触发器 \(p. 218\)](#)
- [编辑 AWS CodeBuild 触发器 \(p. 220\)](#)
- [BitbucketWebHook事件 \(p. 221\)](#)
- [GitHub钩型活动 \(p. 229\)](#)
- [更改 AWS CodeBuild 中构建项目的设置 \(p. 236\)](#)
- [删除 AWS CodeBuild 中的构建项目 \(p. 244\)](#)
- [使用共享项目。 \(p. 245\)](#)
- [在 AWS CodeBuild 中标记项目 \(p. 248\)](#)
- [批次建立在 AWS CodeBuild \(p. 251\)](#)

## 在 AWS CodeBuild 中创建构建项目

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包创建构建项目。

### Prerequisites

在创建构建项目之前，请回答[计划构建 \(p. 135\)](#)。

## 主题

- [创建构建项目 \(控制台\) \(p. 190\)](#)
- [创建构建项目 \(AWS CLI\) \(p. 198\)](#)
- [创建构建项目 \( AWS 开发工具包 \) \(p. 209\)](#)
- [创建构建项目 \(AWS CloudFormation\) \(p. 209\)](#)

## 创建构建项目 (控制台)

Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.

如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。

选择 Create build project (创建构建项目)。

填写以下部分。完成后，选择 创建构建项目 在页面底部。

### 章节

- [项目配置 \(p. 190\)](#)
- [Source \(p. 190\)](#)
- [Environment \(p. 193\)](#)
- [Buildspec \(p. 196\)](#)
- [批次配置 \(p. 196\)](#)
- [Artifacts \(p. 197\)](#)
- [Logs \(p. 198\)](#)

## 项目配置

### 项目名称

为此建立项目输入名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。

### Description

输入构建项目的可选描述，以帮助其他用户了解该项目的使用情况。

### 构建徽章

选择以使项目的构建状态可见并可嵌入。有关更多信息，请参阅 [构建徽章示例 \(p. 78\)](#)。)

### Note

如果您的源提供程序是 Amazon S3，则构建徽章不适用。

### 其他信息

( 可选 ) 对于 Tags (标签)，输入您希望支持 AWS 服务使用的任何标签的名称和值。使用 Add row 添加标签。您最多可以添加 50 个标签。

## Source

### 源提供商

选择源代码提供商类型。使用以下列表为您的源提供商提供适当选择：

Note

CodeBuild 不支持 Bitbucket 服务器。

Amazon S3

Bucket

选择包含源代码的输入桶的名称。

S3对象密钥或S3文件夹

输入zip文件的名称或包含源代码的文件夹的路径。输入正斜杠 (/) 以下载 S3 存储桶中的所有内容。  
源版本

输入代表您输入文件构建的对象的版本ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。)

CodeCommit

Repository

选择要使用的存储库。

参考类型

选择 分支 , GIT标签 , 或 提交ID 要指定源代码的版本。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。)

GIT克隆深度

选择创建一个带有指定数量的历史记录的浅克隆。如果您需要完整克隆，请选择 Full (完整)。

使用GIT子模块

选择是否要在存储库中包含GIT子模块。

Bitbucket

Repository

选择 使用OAuth连接 或 与Bitbucket应用程序密码连接 并按照指示连接 ( 或重新连接 ) 到 Bitbucket。

在您的帐户中选择一个公共存储库或存储库。

源版本

输入分支、提交ID、标签或参考和提交ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)

GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

使用GIT子模块

选择是否要在存储库中包含GIT子模块。

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 name 位提交状态中的参数。有关详细信息，请参阅 [构建 在BitbucketAPI文件中](#)。

对于 目标URL，输入要用于 url 位提交状态中的参数。有关详细信息，请参阅 [构建 在BitbucketAPI文件中](#)。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#)和[BitbucketWebHook事件 \(p. 221\)](#)。

### GitHub

#### Repository

选择 使用OAuth连接 或 与Github个人访问令牌连接 并按照说明连接（或重新连接）到Github并授权访问 AWS CodeBuild.

在您的帐户中选择一个公共存储库或存储库。

#### 源版本

输入分支、提交ID、标签或参考和提交ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)

#### GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

#### 使用GIT子模块

选择是否要在存储库中包含GIT子模块。

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 context GithubCommitStatus ( GitHub提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态 在GithubDeveloper指南中](#)。

对于 目标URL，输入要用于 target\_url GithubCommitStatus ( GitHub提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态 在GithubDeveloper指南中](#)。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#)和[BitbucketWebHook事件 \(p. 221\)](#)。

## GitHub Enterprise Server

### GitHub企业个人访问令牌

参见 [GitHub Enterprise Server 示例 \(p. 109\)](#) 有关如何将个人访问令牌复制到剪贴板的信息。在文本字段中粘贴令牌，然后选择 Save Token (保存令牌)。

#### Note

您只需输入并保存一次个人访问令牌。CodeBuild 将在所有未来项目中使用此令牌。

### 源版本

输入一个拉动请求、分支、提交ID、标签或参考编号，以及提交ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。)

### GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

### 使用GIT子模块

选择是否要在存储库中包含GIT子模块。

### 生成包状态

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

#### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

### 不安全的SSL

在连接到GitHubEnterprise项目资料库时，选择忽略SSL警告。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 context GithubCommitStatus ( GitHub提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态](#) 在 GitHubDeveloper 指南中。

对于 目标URL，输入要用于 target\_url GithubCommitStatus ( GitHub提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态](#) 在 GitHubDeveloper 指南中。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#) 和 [BitbucketWebHook事件 \(p. 221\)](#)。

## Environment

### 环境图像

#### 执行以下任一操作

##### Privileged

( 可选 ) 仅当您计划使用此构建项目来构建 Docker 映像且您选择的构建环境映像不是由具有 Docker 支持的 CodeBuild 提供时，才选择 Privileged (特权)。否则，尝试与 Docker 守护程序交互的所有关联的

构建都将失败。您还必须启动 Docker 守护程序，以便您的构建与其交互。执行此操作的一种方法是，通过运行以下构建命令在您的构建规范的 `install` 阶段初始化 Docker 守护程序。如果您选择了由具有 Docker 支持的 CodeBuild 提供的构建环境映像，请不要运行这些命令。

**Note**

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

## 服务 角色

执行以下任一操作

- 如果您没有 CodeBuild 服务角色，选择 新服务角色。在 Role name 中，为新角色输入名称。
- 如果您有 CodeBuild 服务角色，选择 现有服务角色。在 角色ARN，选择服务角色。

**Note**

当您使用控制台创建构建项目时，可以创建一个创建的项目 CodeBuild 服务角色同时进行。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

## 其他配置

### Timeout

指定在5分钟和480分钟（8小时）之间的值。CodeBuild 如果没有完成，则停止构建。如果 hours 和 minutes 都留空，则将使用 60 分钟的默认值。

### VPC

如果要将 CodeBuild 与您的 VPC 结合使用：

- 对于 VPC，选择 CodeBuild 使用的 VPC ID。
- 对于 VPC Subnets (VPC 子网)，选择包含 CodeBuild 使用的资源的子网。
- 对于 VPC Security groups (VPC 安全组)，选择 CodeBuild 用来支持对 VPC 中资源的访问的安全组。

有关更多信息，请参阅[将 AWS CodeBuild 与 Amazon Virtual Private Cloud 结合使用 \(p. 173\)](#)。

### 计算

选择一个可用选项。

### 环境变量

输入名称和值，然后选择要使用的每个环境变量的类型。

**Note**

CodeBuild 会自动为您的 AWS 区域设置环境变量。如果您尚未将以下环境变量添加到 `buildspec.yml` 中，则必须设置这些变量：

- `AWS_ACCOUNT_ID`
- `IMAGE_REPO_NAME`
- `IMAGE_TAG`

控制台和 AWS CLI 用户可以查看环境变量。如果您不担心环境变量的可见性，请设置 Name 和 Value 字段，然后将 Type 设置为 Plaintext。

我们建议您将具有敏感值（例如 AWS 访问密钥 ID、AWS 秘密访问密钥或密码）的环境变量作为参数存储在 Amazon EC2 Systems Manager Parameter Store 或 AWS Secrets Manager 中。

如果您使用的是 Amazon EC2 Systems Manager Parameter Store，则对于 Type (类型)，选择 Parameter (参数)。对于 Name (名称)，输入标识符供 CodeBuild 引用。对于 Value (值)，按照 Amazon EC2 Systems Manager Parameter Store 中存储的参数名称输入参数名称。使用名为 /CodeBuild/dockerLoginPassword 的参数作为示例，对于 Type (类型)，选择 Parameter (参数)。对于 名称，输入 LOGIN\_PASSWORD...对于 值，输入 /CodeBuild/dockerLoginPassword。

#### Important

如果您使用 Amazon EC2 Systems Manager Parameter Store，我们建议您使用以 /CodeBuild/ 开头的参数名称（例如，/CodeBuild/dockerLoginPassword）来存储参数。可以使用 CodeBuild 控制台在 Amazon EC2 Systems Manager 中创建参数。选择 Create parameter (创建参数)，然后按照对话框中的说明操作。（在该对话框中，对于 KMS key (KMS 密钥)，您可以指定您账户中的 AWS KMS 密钥的 ARN。Amazon EC2 Systems Manager 使用此密钥在存储过程中加密参数的值并在检索过程中解密参数的值。）如果您使用 CodeBuild 控制台创建了一个参数，控制台将在参数名称被存储时以 /CodeBuild/ 作为它的开头。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的参数，则构建项目的服务角色必须允许 `ssm:GetParameters` 操作。如果您之前选择了 New service role (新建服务角色)，CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是，如果您选择了 Existing service role (现有服务角色)，必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的但参数名称不以 /CodeBuild/ 开头的参数，且您选择了 New service role (新建服务角色)，您必须更新该服务角色以允许访问不以 /CodeBuild/ 开头的参数名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的参数名称。

如果您选择 New service role (新建服务角色)，服务角色将包含解密 Amazon EC2 Systems Manager Parameter Store 中 /CodeBuild/ 命名空间下的所有参数的权限。您设置的环境变量将替换现有的环境变量。例如，如果 Docker 映像已经包含一个名为 MY\_VAR 的环境变量（值为 my\_value），并且您设置了一个名为 MY\_VAR 的环境变量（值为 other\_value），那么 my\_value 将被替换为 other\_value。同样，如果 Docker 映像已经包含一个名为 PATH 的环境变量（值为 /usr/local/sbin:/usr/local/bin），并且您设置了一个名为 PATH 的环境变量（值为 \$PATH:/usr/share/ant/bin），那么 /usr/local/sbin:/usr/local/bin 将被替换为文本值 \$PATH:/usr/share/ant/bin。

请勿使用以 CODEBUILD\_ 打头的名称设置任何环境变量。此前缀是专为内部使用预留的。如果具有相同名称的环境变量在多处都有定义，则应按照如下方式确定其值：

- 构建操作调用开始时的值优先级最高。
- 构建项目定义中的值优先级次之。
- 构建规范声明中的值优先级最低。

如果您使用 Secrets Manager，对于 Type (类型)，请选择 Secrets Manager。对于 Name (名称)，输入标识符供 CodeBuild 引用。对于 值，输入 reference-key 使用模式 `secret-id:json-key:version-stage:version-id`...有关信息，请参阅 [Secrets Manager reference-key in the buildspec file](#)。

#### Important

如果您使用 Secrets Manager，我们建议您以 /CodeBuild/ 开头的名称存储密钥（例如 /CodeBuild/dockerLoginPassword）。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的 [什么是 AWS Secrets Manager？](#)。

如果您的构建项目引用了 Secrets Manager 中存储的密钥，则构建项目的服务角色必须允许 secretsmanager:GetSecretValue 操作。如果您之前选择了 New service role (新建服务角色)，CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是，如果您选择了 Existing service role (现有服务角色)，必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Secrets Manager 中存储的但密钥名称不以 /codeBuild/ 开头的密钥，且您选择了 New service role (新建服务角色)，您必须更新该服务角色以允许访问不以 /CodeBuild/ 开头的密钥名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的密钥名称。

如果您选择 New service role (新建服务角色)，该服务角色将拥有解密 Secrets Manager 中 /CodeBuild/ 命名空间下的所有密钥的权限。

## Buildspec

### 构建规格

执行以下任一操作

- 如果您的源代码包括构建规范文件，请选择 Use a buildspec file (使用构建规范文件)。默认情况下，CodeBuild 在源代码根目录中查找名为 buildspec.yml 的文件。如果您的BuildSpec文件使用不同的名称或位置，请在“源根”中输入其路径 BuildSpec名称（例如，buildspec-two.yml 或 configuration/buildspec.yml...如果BuildSpec文件位于S3桶中，则必须在相同的 AWS 地区作为您的建立项目。使用其 ARN 指定构建规范文件（例如，arn:aws:s3:::my-codebuild-sample2/buildspec.yml）。
- 如果您的源代码不包括构建规范文件，或者如果您要从在源代码根目录的 buildspec.yml 文件中为 build 阶段指定的一个构建规范文件中运行构建命令，则选择 Insert build commands (插入构建命令)。对于 Build commands (构建命令)，请输入您要在 build 阶段运行的命令。对于多个命令，使用 && 分开各个命令（例如 mvn test && mvn package）。要在其他阶段运行命令，或者，如果 build 阶段对应的命令列表特别长，请将 buildspec.yml 文件添加到源代码根目录，将命令添加到该文件中，然后选择 Use the buildspec.yml in the source code root directory (在源代码根目录中使用 buildspec.yml)。

有关更多信息，请参阅 [构建规范参考 \(p. 136\)](#)。

## 批次配置

您可以运行一组构建作为单一执行。有关更多信息，请参阅 [批次建立在 AWS CodeBuild \(p. 251\)](#)。)

### 定义批次配置

选择允许在此项目中建立批次。

### 批量服务角色

提供批量建立的服务角色。

选择以下选项之一

- 如果您没有批量服务角色，请选择 新服务角色。在 服务角色，输入新角色的名称。
- 如果您有批量服务角色，请选择 现有服务角色。在 服务角色，选择服务角色。

### 批次的允许计算类型

选择批次允许的计算类型。选择所有适用项。

### 批次允许的最大构建数

输入批次允许的最大建立数。如果批次超出此限制，则批次将失败。

### 批处理超时

输入批次构建完成的最大时间量。

## 合并工件

选择 将所有图像从批次中结合到单个位置 将所有的图像从批次中的所有物料组合到一个位置。

## Artifacts

### Type

执行以下任一操作

- 如果您不想创建任何构建输出构件，请选择 No artifacts (无构件)。如果您只运行构建测试，或者您要将 Docker 映像推送到 Amazon ECR 存储库，建议执行此操作。
- 要将构建输出存储在 S3 存储桶中，请选择 Amazon S3，然后执行以下操作：
  - 如果要将项目名称用于构建输出 ZIP 文件或文件夹，请将 Name (名称) 留空。否则，请输入名称。(如果您要输出 ZIP 文件，并且要让 ZIP 文件包含文件扩展名，请务必在 ZIP 文件名之后添加扩展名。)
  - 如果希望构建规范文件中指定的名称覆盖控制台中指定的任何名称，请选择 Enable semantic versioning (启用语义版本控制)。构建规范文件中的名称是构建时计算得出的，使用 Shell 命令语言。例如，您可以将日期和时间附加到您的项目名称后面，以便它始终是唯一的。构件名称唯一防止覆盖构件。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#)。
  - 对于 Bucket name (存储桶名称)，请选择输出存储桶的名称。
  - 如果您在此过程的前面部分选择了 Insert build commands (插入构建命令)，那么对于 Output files (输出文件)，请输入构建 (该构建要放到构建输出 ZIP 文件或文件夹中) 中的文件位置。对于多个位置，使用逗号分开各个位置 (例如 appspec.yml, target/my-app.jar)。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#) 中 files 的描述。
  - 如果不想加密构建构件，请选择 Remove artifacts encryption (删除构件加密)。

对于所需的每个辅助构件集：

- 对于 Artifact identifier (构件标识符)，输入少于 128 个字符且仅包含字母数字字符和下划线的值。
- 选择 Add artifact (添加构件)。
- 按照前面步骤的说明配置辅助构件。
- 选择 Save artifact (保存构件)。

## 其他配置

### 加密密钥

( 可选 ) 执行以下操作之一：

- 要使用您的账户中适用于 Amazon S3 的 AWS 托管的客户托管密钥 (CMK) 加密构建输出构件，请将 Encryption key (加密密钥) 留空。(这是默认值。)
- 要使用客户托管 CMK 加密生成输出项目，请在 Encryption key (加密密钥) 中输入 CMK 的 ARN。采用格式 `arn:aws:kms:region-ID:account-ID:key/key-ID`。

### 缓存类型

对于 Cache type (缓存类型)，选择下列选项之一：

- 如果您不想使用缓存，请选择 No cache (无缓存)。
- 如果要使用 Amazon S3 缓存，请选择 Amazon S3，然后执行以下操作：
  - 对于 Bucket (存储桶)，选择存储缓存的 S3 存储桶的名称。
  - ( 可选 ) 对于 Cache path prefix (缓存路径前缀)，输入 Amazon S3 路径前缀。Cache path prefix (缓存路径前缀) 值类似于目录名称。它使您能够在存储桶的同一目录下存储缓存。

### Important

请勿将一个尾斜杆 (/) 附加到路径前缀后面。

- 如果想要使用本地缓存，请选择 Local (本地)，然后选择一个或多个本地缓存模式。

#### Note

Docker 层缓存模式仅适用于 Linux。如果您选择该模式，您的项目必须在特权模式下运行。ARM\_CONTAINER 和 LINUX\_GPU\_CONTAINER 环境类型以及 BUILD\_GENERAL1\_2XLARGE 计算类型不支持使用本地缓存。

使用缓存可节省大量构建时间，因为构建环境的可重用部分被存储在缓存中，并且可跨构建使用。有关在构建规范文件中指定缓存的信息，请参阅[构建规范语法 \(p. 137\)](#)。有关缓存的更多信息，请参阅[在 AWS CodeBuild 中构建缓存 \(p. 214\)](#)。

## Logs

选择要创建的日志。您可以创建 Amazon CloudWatch Logs 和/或 Amazon S3 日志。

### CloudWatch

如果要创建 Amazon CloudWatch Logs 日志：

#### CloudWatch 日志

选择 CloudWatch logs (CloudWatch 日志)。

#### Group name

输入您的 Amazon CloudWatch Logs 日志组。

#### 流名称

输入您的 Amazon CloudWatch Logs 日志流名称。

### S3

如果要创建 Amazon S3 日志：

#### S3 日志

选择 S3 logs (S3 日志)。

#### Bucket

选择日志的 S3 存储区的名称。

#### 路径前缀

输入日志的前缀。

#### 禁用 S3 日志加密

如果您不希望加密 S3 日志，请选择此项。

## 创建构建项目 (AWS CLI)

有关将 AWS CLI 与 CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

创建 CodeBuild 使用 AWS CLI，您创建了一个 JSON 格式 项目 结构，填写结构，并呼叫 `create-project` 命令来创建项目。

### 创建 JSON 文件

创建一个带 `create-project` 命令，使用 `--generate-cli-skeleton` 选项：

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

这会创建一个包含路径和文件名的JSON文件，该文件指定了 [`<json-file>`](#)。

## 填写JSON文件

按照以下方式修改JSON数据，并保存您的结果。

```
{  
    "name (p. 201)": "<project-name>",  
    "description (p. 201)": "<description>",  
    "source (p. 201)": {  
        "type (p. 201)": ,  
        "location (p. 201)": "<source-location>",  
        "gitCloneDepth (p. 202)": "<git-clone-depth>",  
        "buildspec (p. 202)": "<buildspec>",  
        "InsecureSsl (p. 203)": "<insecure-ssl>",  
        "reportBuildStatus (p. 202)": "<report-build-status>",  
        "buildStatusConfig": {  
            "context (p. 203)": "<context>",  
            "targetUrl (p. 203)": "<target-url>"  
        },  
        "gitSubmodulesConfig": {  
            "fetchSubmodules (p. 203)": "<fetch-submodules>"  
        },  
        "auth": {  
            "type (p. 202)": "<auth-type>",  
            "resource (p. 202)": "<auth-resource>"  
        },  
        "sourceIdentifier (p. 203)": "<source-identifier>"  
    },  
    "secondarySources (p. 203)": [  
        {  
            "type": ,  
            "location": "<source-location>",  
            "gitCloneDepth": "<git-clone-depth>",  
            "buildspec": "<buildspec>",  
            "InsecureSsl": "<insecure-ssl>",  
            "reportBuildStatus": "<report-build-status>",  
            "auth": {  
                "type": "<auth-type>",  
                "resource": "<auth-resource>"  
            },  
            "sourceIdentifier": "<source-identifier>"  
        }  
    ],  
    "secondarySourceVersions (p. 203)": [  
        {  
            "sourceIdentifier": "<secondary-source-identifier>",  
            "sourceVersion": "<secondary-source-version>"  
        }  
    ],  
    "sourceVersion (p. 203)": "<source-version>",  
    "artifacts (p. 204)": {  
        "type (p. 204)": ,  
        "location (p. 204)": "<artifacts-location>",  
        "path (p. 204)": "<artifacts-path>",  
        "namespaceType (p. 204)": "<artifacts-namespacetype>",  
        "name (p. 204)": "<artifacts-name>",  
        "overrideArtifactName (p. 204)": "<override-artifact-name>",  
        "packaging (p. 205)": "<artifacts-packaging>"  
    },  
    "secondaryArtifacts (p. 205)": [  
        {  
            "type": ,  
            "location": "<secondary-artifact-location>",  
            "path": "<secondary-artifact-path>"  
        }  
    ]  
}
```

```
    "namespaceType": "<secondary-artifact-namespaceType>",
    "name": "<secondary-artifact-name>",
    "packaging": "<secondary-artifact-packaging>",
    "artifactIdentifier": "<secondary-artifact-identifier>"
  }
],
"cache (p. 205)": {
  "type": "<cache-type>",
  "location": "<cache-location>",
  "mode": [
    "<cache-mode>"
  ]
},
"environment (p. 205)": {
  "type (p. 205)": ,
  "image (p. 205)": "<image>",
  "computeType (p. 205)": ,
  "certificate (p. 205)": "<certificate>",
  "environmentVariables (p. 205)": [
    {
      "name": "<environmentVariable-name>",
      "value": "<environmentVariable-value>",
      "type": "<environmentVariable-type>"
    }
  ],
  "registryCredential (p. 207)": [
    {
      "credential": "<credential-arn-or-name>",
      "credentialProvider": "<credential-provider>"
    }
  ],
  "imagePullCredentialsType (p. 207)": ,
  "privilegedMode (p. 207)": "<privileged-mode>"
},
"serviceRole (p. 207)": "<service-role>",
"timeoutInMinutes (p. 207)": <timeout>,
"queuedTimeoutInMinutes (p. 208)": <queued-timeout>,
"encryptionKey (p. 208)": "<encryption-key>",
"tags (p. 208)": [
  {
    "key": "<tag-key>",
    "value": "<tag-value>"
  }
],
"vpcConfig (p. 208)": {
  "securityGroupIds": [
    "<security-group-id>"
  ],
  "subnets": [
    "<subnet-id>"
  ],
  "vpcId": "<vpc-id>"
},
"badgeEnabled (p. 208)": "<badge-enabled>",
"logsConfig (p. 208)": {
  "cloudWatchLogs (p. 208)": {
    "status": "<cloudwatch-logs-status>",
    "groupName": "<group-name>",
    "streamName": "<stream-name>"
  },
  "s3Logs (p. 208)": {
    "status": "<s3-logs-status>",
    "location": "<s3-logs-location>",
    "encryptionDisabled": "<s3-logs-encryption-disabled>"
  }
},
```

```
"fileSystemLocations (p. 208)": [
  {
    "type": "EFS",
    "location": "<EFS-DNS-name-1>:/<directory-path>",
    "mountPoint": "<mount-point>",
    "identifier": "<efs-identifier>",
    "mountOptions": "<efs-mount-options>"
  }
],
"buildBatchConfig (p. 209)": {
  "serviceRole": "<batch-service-role>",
  "combineArtifacts": <combine-artifacts>,
  "restrictions": {
    "maximumBuildsAllowed": <max-builds>,
    "computeTypesAllowed": [
      "<compute-type>"
    ],
    "timeoutInMins": <batch-timeout>
  }
}
```

替换以下内容：

#### name

必填 此构建项目的名称。此名称在您的 AWS 账户的所有构建项目中必须是唯一的。

#### description

：可选。此构建项目的描述。

#### source

必填 A 影响源 包含此构建项目源代码设置信息的对象。添加 source 对象后，可以使用 the section called “secondarySources”添加最多 12 个源。这些设置包括：

#### 来源/类型

必填 包含要构建的源代码的存储库的类型。有效值包括：

- CODECOMMIT
- CODEPIPELINE
- GITHUB
- GITHUB\_ENTERPRISE
- BITBUCKET
- S3
- NO\_SOURCE

如果您使用 NO\_SOURCE，则构建规范不能是一个文件，因为项目没有源。相反，您必须使用 buildspec 属性为 buildspec 指定 YAML 格式的字符串。有关更多信息，请参阅 [无源项目示例 \(p. 133\)](#)。)

#### 来源/位置

必要情况，除非您设置 `<source-type>` 至 CODEPIPELINE...指定存储库类型源代码的位置。

- 对于 CodeCommit，则为指向包含源代码和构建规范文件的存储库的 HTTPS 克隆 URL（例如，`https://git-codecommit.<region-id>.amazonaws.com/v1/repos/<repo-name>`）。

- 对于 Amazon S3，构建输入存储区名称，其后跟包含源代码和BuildSpec的ZIP文件的路径和名称。例如：
  - 对于位于输入桶根的一个拉链文件: `<bucket-name>/<object-name>.zip`.
  - 对于位于输入桶子文件夹中的zip文件: `<bucket-name>/<subfolder-path>/<object-name>.zip`.
- 对于 GitHub，则为指向包含源代码和构建规范文件的存储库的 HTTPS 克隆 URL。该 URL 必须包含 `github.com`。您必须将您的 AWS 账户连接到您的 GitHub 账户。为此，请使用 CodeBuild 控制台创建构建项目。
  - 在 GitHub 授权应用程序 页面，组织访问 部分，选择 请求访问 您希望每个存储库旁边都能找到您想要的 CodeBuild 可以访问。
  - 选择 Authorize application。（连接到 GitHub 账户后，您无需完成创建项目的创建。您可以关闭 CodeBuild 控制台。）
- 对于 GitHub Enterprise Server，则为指向包含源代码和构建规范文件的存储库的 HTTP 或 HTTPS 克隆 URL。您还必须将您的 AWS 账户连接到您的 GitHub Enterprise Server 账户。为此，请使用 CodeBuild 控制台创建构建项目。
  - 在 GitHub Enterprise Server 中创建个人访问令牌。
  - 将此令牌复制到您的剪贴板，以便在创建 CodeBuild 项目时使用。有关更多信息，请参阅 GitHub 帮助网站上的[为命令行创建个人访问令牌](#)。
  - 如果您使用控制台创建 CodeBuild 项目，请在 Source (源) 中为 Source provider (源提供商) 选择 GitHub Enterprise。
  - 对于 Personal Access Token，请粘贴已复制到剪贴板中的令牌。选择 Save Token。您的 CodeBuild 账户现在已与您的 GitHub Enterprise Server 账户连接。
- 对于 Bitbucket，则为指向包含源代码和构建规范文件的存储库的 HTTPS 克隆 URL。该 URL 必须包含 `bitbucket.org`。您还必须将您的 AWS 账户连接到您的 Bitbucket 账户。为此，请使用 CodeBuild 控制台创建构建项目。
  - 当您使用控制台与 Bitbucket 连接（或重新连接）时，在 Bitbucket Confirm access to your account（确认对账户的访问）页面上，选择 Grant access（授予访问权限）。（连接到 Bitbucket 账户后，您无需完成创建项目的创建。您可以关闭 CodeBuild 控制台。）
  - 对于 AWS CodePipeline，请不要为 source 指定 location 值。CodePipeline 会忽略此值，因为当您在 CodePipeline 中创建管道时，您会在管道的源阶段指定源代码位置。

#### 来源/Gitclone深度

：可选。要下载的历史记录深度。最小值为 0。如果此值为 0、大于 25 或未提供，则会下载每个构建项目的完整历史记录。如果您的源类型是 Amazon S3，则不支持此值。

#### 来源/建立规格

：可选。要使用的生成规范定义或文件。如果此值未提供或设置为空字符串，源代码必须在其根目录中包含 `buildspec.yml` 文件。如果设置此值，则可以是内联BuildSpec定义、相对于主源根目录的备用 BuildSpec文件的路径，或者路径到S3存储区。存储桶必须与构建项目位于同一 AWS 区域中。使用其 ARN 指定构建规范文件（例如，`arn:aws:s3:::my-codebuild-sample2/buildspec.yml`）。有关更多信息，请参阅[构建规范文件名称和存储位置 \(p. 136\)](#)。）

#### 来源/授权

请勿使用：此对象用于 CodeBuild 仅控制台。

#### 来源/报表构建状态

指定是否向源提供商发送构建的开始和完成状态。如果使用源提供商而非 GitHub、GitHub Enterprise Server 或 Bitbucket 设置此项，则会引发 `invalidInputException`。

#### 来源/BuildStatusConfig

包含定义如何使用 CodeBuild 构建项目将构建状态报告给源提供商。仅当源类型为“时”选项才能使用 `GITHUB`，`GITHUB_ENTERPRISE`，或 `BITBUCKET`。

### 来源/BuildStatusConfig/上下文

对于位数源来说，此参数用于 name 位提交状态中的参数。对于Github来源，此参数用于 context GithubCommitStatus ( Github提交状态 ) 中的参数。

例如，您可以将 context 包含使用 CodeBuild 环境变量：

```
AWS CodeBuild sample-project Build #$CODEBUILD_BUILD_NUMBER -  
$CODEBUILD_WEBHOOK_TRIGGER
```

这会导致在WebHook拉索请求事件触发的构建#24上显示的上下文如此类似：

```
AWS CodeBuild sample-project Build #24 - pr/8
```

### 来源/BuildStatusConfig/目标网址

对于位数源来说，此参数用于 url 位提交状态中的参数。对于Github来源，此参数用于 target\_url GithubCommitStatus ( Github提交状态 ) 中的参数。

例如，您可以将 targetUrl 至 <https://aws.amazon.com/codebuild/> 提交状态将链接至此 URL。

### 来源/GITSUBMODULESCONFIG

：可选。有关 Git 子模块配置的信息。只能与 CodeCommit、GitHub、GitHub Enterprise Server 和 Bitbucket 一起使用。

#### 源/GITSubModulesConfig/Fetch子模块

如果您希望在存储库中包含 Git 子模块，请将 fetchSubmodules 设置为 true。包含的 Git 子模块必须配置为 HTTPS。

#### 来源/InSecuResSL

：可选。仅与 GitHub Enterprise Server 一起使用。将此值设为 true，将在连接到您的 GitHub Enterprise Server 项目存储库时忽略 TLS 警告。默认值为 false。只应将 InsecureSsl 用于测试目的。它不应在生产环境中使用。

#### 来源/SourcelIdentifier

项目源的用户定义标识符。主源可选。次要来源需要。

#### secondarySources

：可选。系列的 影响源 包含构建项目次级来源信息的对象。最多可添加12个辅助源。TheThe secondarySources 对象使用的属性与 the section called “source” 对象。在次要源对象中， sourceIdentifier 为必填项。

#### secondarySourceVersions

：可选。系列的 产品源自主题 对象。如果在构建级别指定 secondarySourceVersions，则它们优先于此对象。

#### sourceVersion

：可选。构建此项目建立输入的版本。如果未指定，则使用最新版本。如果已指定，则它必须是下列项之一：

- 对于 CodeCommit，要使用的承诺ID、分支或GIT标签。

- 对于 GitHub，指定提交 ID、拉取请求 ID、分支名称或与您要构建的源代码版本相对应的标签名称。如果指定了拉取请求 ID，则它必须使用格式 `pr/pull-request-ID`（例如，`pr/25`）。如果指定了分支名称，则将使用分支的 HEAD 提交 ID。如果未指定，则使用默认分支的 HEAD 提交 ID。
- 对于 Bitbucket，指定提交 ID、分支名称或与您要构建的源代码版本相对应的标签名称。如果指定了分支名称，则将使用分支的 HEAD 提交 ID。如果未指定，则使用默认分支的 HEAD 提交 ID。
- 对于 Amazon S3，为表示要使用的构建输入 ZIP 文件的对象的版本 ID。

如果在构建级别指定 `sourceVersion`，则该版本将优先于此 `sourceVersion`（在对象级别）。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。）

#### artifacts

必填 A [影响力](#) 包含此构建项目输出工件设置信息的对象。添加 `artifacts` 对象后，可以使用 [the section called “secondaryArtifacts”](#) 添加最多 12 个构件。这些设置包括：

##### 伪影/类型

必填 生成输出项目的类型。有效值为：, , , , .

- CODEPIPELINE
- NO\_ARTIFACTS
- S3

##### 伪影/位置

仅与 S3 工件类型。不用于其他伪影类型。

您在先决条件中创建或标识的输出磁带柜的名称。

##### 伪影/路径

仅与 S3 工件类型。不用于其他伪影类型。

输出桶中的路径放置邮件文件或文件夹。如果您未指定值 `path`，CodeBuild 使用 `namespaceType`（如果指定）和 `name` 确定构建输出 zip 文件或文件夹的路径和名称。例如，如果您指定 `MyPath` 对于 `path` 和 `MyArtifact.zip` 对于 `name`，路径和名称将是 `MyPath/MyArtifact.zip`。

##### 伪影/名称类型

仅与 S3 工件类型。不用于其他伪影类型。

构建输出 zip 文件或文件夹的 namespace。有效值包括 `BUILD_ID` 和 `NONE`... 使用 `BUILD_ID` 要将构造 ID 插入构建输出 zip 文件或文件夹的路径。否则，请使用 `NONE`... 如果您未指定值 `namespaceType`，CodeBuild 使用 `path`（如果指定）和 `name` 确定构建输出 zip 文件或文件夹的路径和名称。例如，如果您指定 `MyPath` 对于 `path`，`BUILD_ID` 对于 `namespaceType`，和 `MyArtifact.zip` 对于 `name`，路径和名称将是 `MyPath/build-ID/MyArtifact.zip`。

##### artifacts/name

仅与 S3 工件类型。不用于其他伪影类型。

内部构建输出邮件文件或文件夹的名称 `location`... 例如，如果您指定 `MyPath` 对于 `path` 和 `MyArtifact.zip` 对于 `name`，路径和名称将是 `MyPath/MyArtifact.zip`。

##### 伪影/过度核心名称

仅与 S3 工件类型一起使用。不用于其他伪影类型。

：可选。如果设置为 `true`，指定的名称 `artifacts` BuildSpec 文件的数据块覆盖 `name`... 有关详细信息，请参阅 [适用于 CodeBuild 的构建规范参考 \(p. 136\)](#)。

## 伪影/包装

仅与 S3 工件类型。不用于其他伪影类型。

: 可选。指定如何包装工件。允许的值包括：、、、。

NONE

创建包含构建工件的文件夹。: 这是默认值。

ZIP

创建包含构建工件的 zip 文件。

## secondaryArtifacts

: 可选。系列的 **影响力** 包含构建项目次级伪影设置信息的对象。最多可以添加 12 个辅助构件。`secondaryArtifacts` 使用的许多设置与 [the section called “artifacts”](#) 对象相同。

### cache

必填 A [ProjectCache](#) 包含此构建项目缓存设置信息的对象。有关更多信息，请参阅 [构建缓存 \(p. 214\)](#)。)

### environment

必填 A [投影病毒](#) 包含此项目构建环境设置信息的对象。这些设置包括：

#### 环境/类型

必填 构建环境的类型。有关详细信息，请参阅 [类型](#) 在 CodeBuild API 参考。

#### 环境/图像

必填 此构建环境使用的 Docker 映像标识符。通常，此标识符表示为 `image-name:tag`...例如，在 Docker 存储库中 CodeBuild 用于管理靠泊装置图像，这可能是 `aws/codebuild/standard:4.0...` 在靠泊站中心 `maven:3.3.9-jdk-8...` 在 Amazon ECR，`account-id.dkr.ecr.region-id.amazonaws.com/your-Amazon-ECR-repo-name:tag`...有关详细信息，请参阅 [CodeBuild 提供的 Docker 映像 \(p. 159\)](#).

#### 环境/计算类型

必填 指定此构建环境所使用的计算资源。有关详细信息，请参阅 [计算类型](#) 在 CodeBuild API 参考。

#### 环境/证书

: 可选。ARN 的 Amazon S3 包含 PEM 编码证书的桶、路径前缀和对象密钥。对象键可以仅为 .pem 文件，也可以为包含 PEM 编码的证书的 .zip 文件。例如，如果您的 Amazon S3 桶名称为 `my-bucket`，您的路径前缀是 `cert`，您的对象密钥名称为 `certificate.pem`，然后可接受的格式为 `certificate` 是 `my-bucket/cert/certificate.pem` 或 `arn:aws:s3:::my-bucket/cert/certificate.pem`.

#### 环境/环境变量

: 可选。系列的 [环境变量](#) 包含您要为此构建环境指定的环境变量的对象。每个环境变量都表示为包含 A 的对象 `name`，`value`，和 `type` 的 `name`，`value`，和 `type`。

控制台和 AWS CLI 用户可以看到所有环境变量。如果您对环境变量的可见性没有任何疑虑，请设置 `name` 和 `value`，并设置 `type` 至 PLAINTEXT.

我们建议您将环境变量存储在敏感值，例如 AWS 访问密钥 ID，AWS 密码或密码，作为参数中的参数 Amazon EC2 Systems Manager 参数存储或 AWS Secrets Manager. 对于 `name`，对于该存储的参数，设置标识符 [CodeBuild 参考](#)。

如果您使用 Amazon EC2 Systems Manager 参数存储,用于 value , 将参数名称设置为存储在参数存储区中。设置 type 至 PARAMETER\_STORE...使用名为 /CodeBuild/dockerLoginPassword 例如, 设置 name 至 LOGIN\_PASSWORD...设置 value 至 /CodeBuild/dockerLoginPassword...设置 type 至 PARAMETER\_STORE.

#### Important

如果您使用 Amazon EC2 Systems Manager Parameter Store , 我们建议您使用以 /CodeBuild/ 开头的参数名称 ( 例如 , /CodeBuild/dockerLoginPassword ) 来存储参数。可以使用 CodeBuild 控制台在 Amazon EC2 Systems Manager 中创建参数。选择 Create parameter ( 创建参数 ) , 然后按照对话框中的说明操作。( 在该对话框中 , 对于 KMS key (KMS 密钥) , 您可以指定您账户中的 AWS KMS 密钥的 ARN. Amazon EC2 Systems Manager 使用此密钥在存储过程中加密参数的值并在检索过程中解密参数的值。 ) 如果您使用 CodeBuild 控制台创建了一个参数 , 控制台将在参数名称被存储时以 /CodeBuild/ 作为它的开头。有关更多信息 , 请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的参数 , 则构建项目的服务角色必须允许 ssm:GetParameters 操作。如果您之前选择了 New service role ( 新建服务角色 ) , CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是 , 如果您选择了 Existing service role ( 现有服务角色 ) , 必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的但参数名称不以 /CodeBuild/ 开头的参数 , 且您选择了 New service role ( 新建服务角色 ) , 您必须更新该服务角色以允许访问不以 /CodeBuild/ 开头的参数名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的参数名称。

如果您选择 New service role ( 新建服务角色 ) , 服务角色将包含解密 Amazon EC2 Systems Manager Parameter Store 中 /CodeBuild/ 命名空间下的所有参数的权限。

您设置的环境变量将替换现有的环境变量。例如 , 如果 Docker 映像已经包含一个名为 MY\_VAR 的环境变量 ( 值为 my\_value ) , 并且您设置了一个名为 MY\_VAR 的环境变量 ( 值为 other\_value ) , 那么 my\_value 将被替换为 other\_value 。同样 , 如果 Docker 映像已经包含一个名为 PATH 的环境变量 ( 值为 /usr/local/sbin:/usr/local/bin ) , 并且您设置了一个名为 PATH 的环境变量 ( 值为 \$PATH:/usr/share/ant/bin ) , 那么 /usr/local/sbin:/usr/local/bin 将被替换为文本值 \$PATH:/usr/share/ant/bin 。

请勿使用以 CODEBUILD\_ 打头的名称设置任何环境变量。此前缀是专为内部使用预留的。

如果具有相同名称的环境变量在多处都有定义 , 则应按照如下方式确定其值 :

- 构建操作调用开始时的值优先级最高。
- 构建项目定义中的值优先级次之。
- 构建规范声明中的值优先级最低。

如果您使用 Secrets Manager , 对于 value , 将参数名称设置为 Secrets Manager. 设置 type 至 SECRETS\_MANAGER...使用名为 /CodeBuild/dockerLoginPassword 例如 , 设置 name 至 LOGIN\_PASSWORD...设置 value 至 /CodeBuild/dockerLoginPassword...设置 type 至 SECRETS\_MANAGER.

#### Important

如果您使用 Secrets Manager , 我们建议您以 /CodeBuild/ 开头的名称存储密钥 ( 例如 /CodeBuild/dockerLoginPassword ) 。有关更多信息 , 请参阅 AWS Secrets Manager 用户指南中的 [什么是 AWS Secrets Manager ?](#) 。

如果您的构建项目引用了 Secrets Manager 中存储的密钥 , 则构建项目的服务角色必须允许 secretsmanager:GetSecretValue 操作。如果您之前选择了 New service role ( 新建服务角色 ) , CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是 , 如果您选择了 Existing service role ( 现有服务角色 ) , 必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Secrets Manager 中存储的但密钥名称不以 /CodeBuild/ 开头的密钥 , 且您选择了 New service role ( 新建服务角色 ) , 您必须更新该服务角色以允许访问不以 /CodeBuild/ 开头的密钥名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的密钥名称。

如果您选择 New service role ( 新建服务角色 ) , 该服务角色将拥有解密 Secrets Manager 中 /CodeBuild/ 命名空间下的所有密钥的权限。

### 环境/登记凭据

: 可选。A [登记凭据](#) 指定提供访问私人泊车程序注册表访问权限的对象。

#### 环境/注册凭据/凭据

指定使用ARN或者凭据创建的凭据名称 AWS Managed Services . 仅当凭证存在于您当前的区域中时，您才能使用凭证的名称。

#### 环境/注册凭据/凭证提供商

唯一有效值为 SECRETS\_MANAGER。

当设置此属性时：

- imagePullCredentials 必须设置为 SERVICE\_ROLE。
- 图像不能是策划图像 Amazon ECR 图像。

#### 环境/ImgEpullCredentialstype

: 可选。CodeBuild 用于在您的生成中拉取映像的凭证的类型。有两个有效值：

##### 代码构建

CODEBUILD 指定 CodeBuild 使用其自己的凭证。您必须编辑 Amazon ECR 存储库策略以信任 CodeBuild 服务委托人。

##### 服务角色

指定 CodeBuild 使用您的构建项目的服务角色。

当您使用跨账户或专用镜像仓库镜像时，必须使用 SERVICE\_ROLE 凭证。当您使用 CodeBuild 精选镜像时，必须使用 CODEBUILD 凭证。

#### 环境/权限模式

设置为 true 只有您计划使用此构建项目来构建靠泊装置图像时，您指定的构建环境图像才能提供 CodeBuild 使用靠泊装置支持。否则，尝试与 Docker 守护程序交互的所有关联的构建都将失败。您还必须启动 Docker 守护程序，以便您的构建与其交互。执行此操作的一种方法是通过运行以下构建命令在您的构建规范文件的 install 阶段初始化 Docker 守护程序。如果您指定了由具有 Docker 支持的 CodeBuild 提供的构建环境映像，请不要运行这些命令。

#### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

#### serviceRole

必填 CodeBuild 代表 IAM 用户用来与服务进行交互的服务角色 ARN ( 例如，arn:aws:iam::account-id:role/*role-name* ) 。

#### timeoutInMinutes

: 可选。5 到 480 分钟 (8 个小时) 之间的一个分钟数，在此时间后，如果构建未完成，CodeBuild 会将其停止。如果未指定，则使用默认值 60。要确定 CodeBuild 是否以及何时因超时而停止生成，请运行 batch-get-builds 命令。要确定构造是否已停止，请查看输出的输出 buildStatus 价值 FAILED...要确定构建超时的时间，请查看输出的输出 endTime 与 A 相关的值 phaseStatus 价值 TIMED\_OUT.

### queuedTimeoutInMinutes

: 可选。分钟数，5到480之间（8小时），之后 CodeBuild 如果仍在排队中停止构建。如果未指定，则使用默认值 60。

### encryptionKey

: 可选。CodeBuild 用来加密构建输出的 AWS KMS 客户托管密钥 (CMK) 的别名或 ARN。如果指定别名，请使用格式 `arn:aws:kms:<region-ID>:<account-ID>/key/<key-ID>` 或者，如果别名存在，请使用格式 `alias/<key-alias>`...如果未指定，AWS-管理 CMK Amazon S3 已使用。

### tags

: 可选。系列的 [标签](#) 提供您想与此构建项目关联的标签的对象。您最多可指定 50 个标签。这些标签可由支持 CodeBuild 生成项目标签的任何 AWS 服务使用。每个标签都以一个对象表示 key 和 value.

### vpcConfig

: 可选。A [vpcconfig](#) 包含有关 VPC 配置信息的对象。这些属性包括:

#### vpcId

必填 CodeBuild 使用的 VPC ID。运行此命令获取您的区域中的所有 VPC ID 的列表：

```
aws ec2 describe-vpcs --region <region-ID>
```

#### 子网

必填 包括用于以下内容的子网ID的阵列 CodeBuild. 运行以下命令获取这些 ID：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

#### securityGroupIds

必填 一系列安全组ID，用于 CodeBuild 允许访问 vpc 中的资源。运行以下命令获取这些 ID：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

### badgeEnabled

: 可选。指定何时在 ER 中包含与您的 CodeBuild 项目。设置为 `true` 启用构建巴塞尔多项目的组织，或 `false` Otehrwise。有关更多信息，请参阅 [使用 CodeBuild 构建徽章示例 \(p. 78\)](#)。)

### logsConfig

A [日志配置](#) 包含此构建日志所在的信息的对象。

#### LogsConfig/CloudWatchLogs

A [CloudWatchLogsconfig](#) 包含将日志推送到 CloudWatch Logs.

#### LogsConfig/s3Logs

一个 [S3Logsconfig](#) ( [S3logsconfig](#) ) 包含将日志推送到 Amazon S3.

### fileSystemLocations

: 可选。系列的 [项目文件系统位置](#) 包含您的 Amazon EFS 配置。

## buildBatchConfig

: 可选。The the buildBatchConfig 对象是 ProjectBuildBatchconfig 包含项目批次构建配置信息的结构。

### BuildBatchConfig/维修服务

批量构建项目的服务角色ARN。

### BuildBatchConfig/组合概况

布尔值，指定将批次构建的图像批次是否合并到单个伪影位置。

### BuildBatchConfig/限制/最大建立已调整

允许的最大构建数。

### BuildBatchConfig/限制/计算允许的计算

一系列字符串，用于指定批次构建允许的计算类型。参见 [构建环境计算类型](#) 对于这些值。

### BuildBatchConfig/时间超时分钟

批次构建必须在中完成的最大时间量（以分钟为单位）。

## 创建项目

要创建项目，请运行 `create-project` 再次命令传递JSON文件：

```
aws codebuild create-project --cli-input-json file://<json-file>
```

如果成功，JSON代表的 [项目](#) 对象出现在控制台输出中。查看 [创建项目响应语法](#) 对于此数据。

您稍后可以更改构建项目的任何设置，但构建项目名称除外。有关更多信息，请参阅 [更改构建项目的设置 \(AWS CLI\) \(p. 243\)](#)。

要开始运行构建，请参阅 [运行构建 \(AWS CLI\) \(p. 256\)](#)。

如果您的源代码存储在 GitHub 存储库中，并且您希望 CodeBuild 在每次代码更改被推送到存储库时重建源代码，请参阅[开始自动运行构建 \(AWS CLI\) \(p. 260\)](#)。

## 创建构建项目 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 创建构建项目 (AWS CloudFormation)

有关将 AWS CodeBuild 与 AWS CloudFormation 结合使用的信息，请参阅 [AWS CloudFormation 用户指南](#) [CodeBuild 中的](#) 的 AWS CloudFormation 模板。

## Create a notification rule

You can use notification rules to notify users when important changes, such as build successes and failures, occur. Notification rules specify both the events and the Amazon SNS topic that is used to send notifications. For more information, see [What are notifications?](#)

You can use the console or the AWS CLI to create notification rules for AWS CodeBuild.

### To create a notification rule (console)

1. Sign in to the AWS Management Console and open the CodeBuild console at <https://console.aws.amazon.com/codebuild/>.
2. Choose Build, choose Build projects, and then choose a build project where you want to add notifications.
3. On the build project page, choose Notify, and then choose Create notification rule. You can also go to the Settings page for the build project and choose Create notification rule.
4. In Notification name, enter a name for the rule.
5. In Detail type, choose Basic if you want only the information provided to Amazon EventBridge included in the notification. Choose Full if you want to include information provided to Amazon EventBridge and information that might be supplied by the CodeBuild or the notification manager.

For more information, see [Understanding Notification Contents and Security](#).

6. In Events that trigger notifications, select the events for which you want to send notifications. For more information, see [Events for Notification Rules on Build Projects](#).
7. In Targets, do one of the following:
  - If you have already configured a resource to use with notifications, in Choose target type, choose either AWS Chatbot (Slack) or SNS topic. In Choose target, choose the name of the client (for a Slack client configured in AWS Chatbot) or the Amazon Resource Name (ARN) of the Amazon SNS topic (for Amazon SNS topics already configured with the policy required for notifications).
  - If you have not configured a resource to use with notifications, choose Create target, and then choose SNS topic. Provide a name for the topic after codestar-notifications-, and then choose Create.

#### Note

- If you create the Amazon SNS topic as part of creating the notification rule, the policy that allows the notifications feature to publish events to the topic is applied for you. Using a topic created for notification rules helps ensure that you subscribe only those users that you want to receive notifications about this resource.
  - You cannot create an AWS Chatbot client as part of creating a notification rule. If you choose AWS Chatbot (Slack), you will see a button directing you to configure a client in AWS Chatbot. Choosing that option opens the AWS Chatbot console. For more information, see [Configure Integrations Between Notifications and AWS Chatbot](#).
  - If you want to use an existing Amazon SNS topic as a target, you must add the required policy for AWS CodeStar Notifications in addition to any other policies that might exist for that topic. For more information, see [Configure Amazon SNS Topics for Notifications](#) and [Understanding Notification Contents and Security](#).
8. To finish creating the rule, choose Submit.
  9. You must subscribe users to the Amazon SNS topic for the rule before they can receive notifications. For more information, see [Subscribe Users to Amazon SNS Topics That Are Targets](#). You can also set up integration between notifications and AWS Chatbot to send notifications to Amazon Chime chatrooms. For more information, see [Configure Integration Between Notifications and AWS Chatbot](#).

### To create a notification rule (AWS CLI)

1. At a terminal or command prompt, run the create-notification rule command to generate the JSON skeleton:

```
aws codestarnotifications create-notification-rule --generate-cli-skeleton > rule.json
```

- You can name the file anything you want. In this example, the file is named `rule.json`.
2. Open the JSON file in a plain-text editor and edit it to include the resource, event types, and target you want for the rule. The following example shows a notification rule named `MyNotificationRule` for a build project named `MyBuildProject` in an AWS account with the ID `123456789012`. Notifications are sent with the full detail type to an Amazon SNS topic named `codestar-notifications-MyNotificationTopic` when builds are successful:

```
{  
    "Name": "MyNotificationRule",  
    "EventTypeId": [  
        "codebuild-project-build-state-succeeded"  
    ],  
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyBuildProject",  
    "Targets": [  
        {  
            "TargetType": "SNS",  
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-  
notifications-MyNotificationTopic"  
        }  
    ],  
    "Status": "ENABLED",  
    "DetailType": "FULL"  
}
```

Save the file.

3. Using the file you just edited, at the terminal or command line, run the `create-notification-rule` command again to create the notification rule:

```
aws codestarnotifications create-notification-rule --cli-input-json file://rule.json
```

4. If successful, the command returns the ARN of the notification rule, similar to the following:

```
{  
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/  
dc82df7a-EXAMPLE"  
}
```

## 查看 AWS CodeBuild 中构建项目名称的列表

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包查看 CodeBuild 中的构建项目列表。

### 主题

- [查看构建项目名称的列表（控制台）\(p. 211\)](#)
- [查看构建项目名称的列表（AWS CLI）\(p. 212\)](#)
- [查看构建项目名称的列表（AWS 开发工具包）\(p. 213\)](#)

## 查看构建项目名称的列表（控制台）

您可以在控制台中查看 AWS 区域中的构建项目的列表。信息包括名称、源提供程序、存储库、最新构建状态以及描述（如果有）。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。

### Note

默认情况下，仅显示 10 个最新的构建项目。要查看更多生成项目，请选择齿轮图标，然后为 Projects per page (每页项目数) 选择不同值，或使用向后和向前箭头。

## 查看构建项目名称的列表 (AWS CLI)

运行 list-projects 命令。

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-token next-token
```

替换上一命令中的以下占位符：

- *sort-by*: 用于表示用于列出构建项目名称的标准的可选字符串。有效值包括：
  - CREATED\_TIME: : 根据每个构建项目的创建时间列出构建项目名称。
  - LAST\_MODIFIED\_TIME: : 根据每个构建项目信息上次更改的时间列出构建项目名称。
  - NAME: : 根据每个构建项目的名称列出构建项目名称。
- *sort-order*: 可选字符串，用于指示列表项目的列表中的顺序，基于 *sort-by* 有效值包括 ASCENDING 和 DESCENDING。
- *next-token*: 可选字符串。在上次运行时，如果列表中有 100 个以上的项目，则只能返回前 100 个项目，以及名为下一个令牌的唯一字符串。要获取列表中的下一批项目，请再次运行此命令，将下一个令牌添加到调用中。要获取列表中的所有项目，请利用每个后续的下一个令牌运行此命令，直到不再返回下一个令牌。

例如，如果您运行此命令：

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

与以下内容类似的结果可能会出现在输出中：

```
{  
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=",  
  "projects": [  
    "codebuild-demo-project",  
    "codebuild-demo-project2",  
    ... The full list of build project names has been omitted for brevity ...  
    "codebuild-demo-project99"  
  ]  
}
```

如果您再次运行此命令：

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING --next-token  
Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

与以下内容类似的结果可能会出现在输出中：

```
{  
  "projects": [  
    "codebuild-demo-project100",  
    "codebuild-demo-project101",  
    ... The full list of build project names has been omitted for brevity ...  
    "codebuild-demo-project122"
```

```
    ]  
}
```

## 查看构建项目名称的列表 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 查看 AWS CodeBuild 中构建项目的详细信息

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包查看 CodeBuild 中构建项目的详细信息。

### 主题

- [查看构建项目的详细信息 \( 控制台 \) \(p. 213\)](#)
- [查看构建项目的详细信息 \(AWS CLI\) \(p. 213\)](#)
- [查看构建项目的详细信息 \( AWS 开发工具包 \) \(p. 214\)](#)

## 查看构建项目的详细信息 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。

### Note

默认情况下，仅显示 10 个最新的构建项目。要查看更多生成项目，请选择齿轮图标，然后为 Projects per page (每页项目数) 选择不同值，或使用向后和向前箭头。

3. 在生成项目列表中的 Name (名称) 列，选择生成项目的链接。
4. 在 Build project: **project-name** (生成项目: project-name) 页面上，选择 Build details (生成详细信息)。

## 查看构建项目的详细信息 (AWS CLI)

运行 batch-get-projects 命令：

```
aws codebuild batch-get-projects --names names
```

在上述命令中，替换以下占位符：

- **names**：必需字符串，用于指示要查看其详细信息的一个或多个生成项目名称。要指定多个构建项目，请用空格分隔各个构建项目的名称。您最多可以指定 100 个构建项目名称。要获取构建项目的列表，请参阅[查看构建项目名称的列表 \(AWS CLI\) \(p. 212\)](#)。

例如，如果您运行此命令：

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2 my-other-demo-project
```

与以下内容类似的结果可能会出现在输出中。为简洁起见，使用省略号 (...) 表示省略的数据。

```
{  
  "projectsNotFound": [
```

```
        "my-other-demo-project"
    ],
    "projects": [
        {
            ...
            "name": codebuild-demo-project,
            ...
        },
        {
            ...
            "name": "codebuild-demo-project2",
            ...
        }
    ]
}
```

在前面的输出中，`projectsNotFound` 数组列出了已指定但未找到的所有生成项目名称。`projects` 数组列出了可找到相关信息的所有构建项目的详细信息。为简洁起见，前面的输出中省略了构建项目的详细信息。有关更多信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#) 的输出。

有关将 AWS CLI 与 AWS CodeBuild 结合使用的更多信息，请参阅 [命令行参考 \(p. 363\)](#)。

## 查看构建项目的详细信息 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅 [AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 在 AWS CodeBuild 中构建缓存

构建项目时，可以使用缓存来节省时间。缓存可以存储构建环境的可重用部分，并在多个构建中使用它们。您的构建项目可以使用两种缓存类型中的一种：Amazon S3 或本地。如果使用本地缓存，则必须选择三种缓存模式中的一种或多种：源缓存、Docker 层缓存和自定义缓存。

### Note

Docker 层缓存模式仅适用于 Linux 环境。如果选择此模式，则必须在特权模式下运行构建。被授予特权模式的 CodeBuild 项目会向所有设备授予其容器访问权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

### 主题

- [Amazon S3 缓存 \(p. 214\)](#)
- [本地缓存 \(p. 214\)](#)

## Amazon S3 缓存

Amazon S3 缓存将缓存存储在跨多个构建主机可用的 Amazon S3 存储桶中。对于构建成本高于下载成本的小型中间构建工件，这是一个很好的选择。这不是大型构建构件的最佳选择，因为它们可能需要很长时间才能通过网络传输，这会影响构建性能。如果您使用 Docker 图层，它也不是最好的选择。

## 本地缓存

本地缓存将缓存本地存储在构建主机上，并且仅可用于该构建主机。对于大型中间构建工件，这是一个很好的选择，因为构建主机上的缓存立即可用。如果您不经常构建，这不是最好的选择。这意味着构建性能不受网络传输时间的影响。如果您选择本地缓存，则必须选择以下一个或多个缓存模式：

- 源缓存模式用于缓存主要和辅助源的 Git 元数据。创建缓存后，后续构建仅提取两次提交之间发生的更改。对于具有干净工作目录和源为大型 Git 存储库的项目，此模式是一个不错的选择。如果选择此选项并且项目不使用 Git 存储库 (GitHub、GitHub Enterprise Server 或 Bitbucket)，则此选项将被忽略。

- Docker 层缓存模式缓存现有 Docker 层。对于构建或提取大型 Docker 映像的项目，此模式是一个不错的选择。它可以防止因从网络中提取大型 Docker 映像而导致的性能问题。

**Note**

- 您只能在 Linux 环境中使用 Docker 层缓存。
- 必须设置 `privileged` 标志以使您的项目具有所需的 Docker 权限。

**Note**

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

- 在使用 Docker 层缓存之前，您应考虑安全影响。
- 自定义缓存模式用于缓存您在构建规格文件中指定的目录。如果您的构建方案不适合另外两种本地缓存模式之一，则此模式是一个不错的选择。如果您使用自定义缓存：
  - 只能指定目录进行缓存。不能指定单独的文件。
  - 用于引用缓存目录的符号链接。
  - 缓存目录在下载项目源代码之前链接到您的构建。如果缓存项目具有相同的名称，则它们会覆盖源项目。使用 `buildspec` 文件中的缓存路径指定目录。有关更多信息，请参阅[构建规范语法 \(p. 137\)](#)。)
  - 避免在源和缓存中使用相同的目录名称。本地缓存的目录可能会覆盖或删除源存储库中具有相同名称的目录。

**Note**

`ARM_CONTAINER` 和 `LINUX_GPU_CONTAINER` 环境类型以及 `BUILD_GENERAL1_2XLARGE` 计算类型不支持使用本地缓存。有关更多信息，请参阅[构建环境计算类型 \(p. 166\)](#)。

**主题**

- [指定本地缓存 \(CLI\) \(p. 215\)](#)
- [指定本地缓存 \(控制台\) \(p. 216\)](#)
- [指定本地缓存 \(AWS CloudFormation\) \(p. 218\)](#)

您可以使用 AWS CLI、控制台、开发工具包或 AWS CloudFormation 来指定本地缓存。

## 指定本地缓存 (CLI)

您可以使用 AWS CLI 中的 `--cache` 参数指定三种本地缓存类型中的每一种。

- 指定源缓存：

```
--cache type=LOCAL,mode=[ LOCAL_SOURCE_CACHE ]
```

- 指定 Docker 层缓存：

```
--cache type=LOCAL,mode=[ LOCAL_DOCKER_LAYER_CACHE ]
```

- 指定自定义缓存：

```
--cache type=LOCAL,mode=[ LOCAL_CUSTOM_CACHE ]
```

有关更多信息，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。)

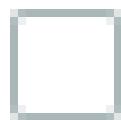
## 指定本地缓存（控制台）

您可以使用控制台的 Artifacts (构件) 部分指定缓存。对于 Cache type (缓存类型)，选择 Amazon S3 或 Local (本地)。如果您选择 Local (本地)，请选择三个本地缓存选项中的一个或多个。

## Cache type

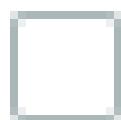
**Local**

Select one or more local cache types.



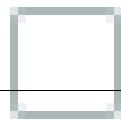
**Docker layer cache**

Caches existing Docker layers.



**Source cache**

Caches .git metadata so it can be used by the build.



**Custom cache**

API 版本 2016-10-06

217

Caches directories specified in the buildspec file.

有关更多信息，请参阅 [创建构建项目（控制台）\(p. 190\)](#)。)

## 指定本地缓存 (AWS CloudFormation)

如果您使用 AWS CloudFormation 要指定本地缓存，请在 Cache 属性，Type，指定 LOCAL...以下样本 YAML 格式化 AWS CloudFormation 代码指定所有三个本地缓存类型。您可以指定这些类型的任意组合。如果您使用 Docker 层缓存，在 Environment 下，您必须将 PrivilegedMode 设置为 true，将 Type 设置为 LINUX\_CONTAINER。

```
CodeBuildProject:  
  Type: AWS::CodeBuild::Project  
  Properties:  
    Name: MyProject  
    ServiceRole: <service-role>  
    Artifacts:  
      Type: S3  
      Location: myBucket  
      Name: myArtifact  
      EncryptionDisabled: true  
      OverrideArtifactName: true  
    Environment:  
      Type: LINUX_CONTAINER  
      ComputeType: BUILD_GENERAL1_SMALL  
      Image: aws/codebuild/standard:4.0  
      Certificate: bucket/cert.zip  
      # PrivilegedMode must be true if you specify LOCAL_DOCKER_LAYER_CACHE  
      PrivilegedMode: true  
    Source:  
      Type: GITHUB  
      Location: <github-location>  
      InsecureSsl: true  
      GitCloneDepth: 1  
      ReportBuildStatus: false  
    TimeoutInMinutes: 10  
  Cache:  
    Type: LOCAL  
    Modes: # You can specify one or more cache mode,  
    - LOCAL_CUSTOM_CACHE  
    - LOCAL_DOCKER_LAYER_CACHE  
    - LOCAL_SOURCE_CACHE
```

### Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

有关更多信息，请参阅 [创建构建项目 \(AWS CloudFormation\) \(p. 209\)](#)。)

## 创建 AWS CodeBuild 触发器

您可以在项目上创建触发器以安排每小时、每天或每周进行一次构建。您也可以将自定义规则与 Amazon CloudWatch cron 表达式结合使用来创建触发器。例如，通过使用 cron 表达式，您可以安排在每个工作日的特定时间进行构建。

创建项目后，您可以创建触发器。

### 创建触发器

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。

- 选择要将触发器添加到的构建项目的链接，然后选择 Build triggers (构建触发器) 选项卡。

**Note**

默认情况下，将显示 100 个最新的构建项目。要查看更多构建项目，请选择齿轮图标，然后为 Projects per page (每页项目数) 选择不同值，或使用向后和向前箭头。

- 选择 Create trigger。
- 在 Trigger name (触发器名称) 中输入名称。
- 从 Frequency (频率) 下拉列表中，选择触发器的频率。如果要使用 Cron 表达式创建频率，请选择 Custom (自定义)。
- 为触发器的频率指定参数。您可以在文本框中输入您的选项的前几个字符以筛选下拉菜单项。

**Note**

开始小时和分钟从零开始。开始分钟是一个介于 0 和 59 之间的数字。开始小时是一个介于 0 和 23 之间的数字。例如，每天中午 12:15 开始的每天触发的开始小时为 12，开始分钟为 15。每天午夜开始的每天触发的开始小时为 0，开始分钟为 0。每天下午 11:59 开始的每天触发的开始小时为 23，开始分钟为 59。

频率	必需参数	详细信息
每小时	开始分钟	使用 Start minute (开始分钟) 下拉菜单。
每天	开始分钟 开始小时	使用 Start minute (开始分钟) 下拉菜单。 使用 Start hour (开始小时) 下拉菜单。
每周	开始分钟 开始小时 开始日	使用 Start minute (开始分钟) 下拉菜单。 使用 Start hour (开始小时) 下拉菜单。 使用 Start day (开始日) 下拉菜单。
Custom	Cron 表达式	在 Cron expression (Cron 表达式) 中输入 Cron 表达式。Cron 表达式有六个必填字段，各字段之间以空格分隔。这些字段分别指定分钟、小时、月中日、月、周中日和年的开始值。您可以使用通配符指定范围、其他值等等。例如，Cron 表达式 <code>0 * ? * MON-FRI *</code> 安排在每个工作日上午 9:00 进行构建。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南中的 <a href="#">Cron 表达式</a> 。

- 选择 Enable this trigger (启用此触发器)。
- (可选) 展开 Advanced (高级) 部分。在源版本中，键入源的版本。
  - 对于 Amazon S3，输入与您要生成的输入构件的版本相对应的版本 ID。如果 Source version (源版本) 留空，则使用最新版本。

- 对于 AWS CodeCommit，键入一个提交 ID。如果 Source version (源版本) 留空，则使用默认分支的 HEAD 提交 ID。
  - 对于 GitHub 或 GitHub Enterprise，请键入提交 ID、拉取请求 ID、分支名称或与您要构建的源代码版本对应的标签名称。如果您要指定拉取请求 ID，则必须使用格式 pr/*pull-request-ID* (例如，pr/25)。如果您要指定分支名称，则将使用分支的 HEAD 提交 ID。如果 Source version (源版本) 留空，则将使用默认分支的 HEAD 提交 ID。
  - 对于 Bitbucket，键入提交 ID、分支名称或与您要构建的源代码版本对应的标签名称。如果您要指定分支名称，则将使用分支的 HEAD 提交 ID。如果 Source version (源版本) 留空，则将使用默认分支的 HEAD 提交 ID。
- (可选) 指定介于 5 分钟和 480 分钟 (8 小时) 之间的超时。此值指定 AWS CodeBuild 在停止前尝试构建的时间长度。如果小时和分钟保留为空，则使用项目中指定的默认超时值。
  - 选择 Create trigger。

## 编辑 AWS CodeBuild 触发器

您可以在项目上编辑触发器以安排每小时、每天或每周进行一次构建。您也可以编辑触发器以将自定义规则与 Amazon CloudWatch cron 表达式结合使用。例如，通过使用 cron 表达式，您可以安排在每个工作日的特定时间进行构建。有关创建触发器的信息，请参阅[创建 AWS CodeBuild 触发器 \(p. 218\)](#)。

要编辑触发器，请执行以下操作：

- Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
- 在导航窗格中，选择 Build projects。
- 选择要更改的生成项目的链接，然后选择 Build triggers (生成触发器) 选项卡。

### Note

默认情况下，将显示 100 个最新的构建项目。要查看更多构建项目，请选择齿轮图标，然后为 Projects per page (每页项目数) 选择不同值，或使用向后和向前箭头。

- 选择您要更改的触发器旁边的单选按钮，然后选择编辑。
- 从 Frequency (频率) 下拉列表中，选择触发器的频率。如果要使用 Cron 表达式创建频率，请选择 Custom (自定义)。
- 为触发器的频率指定参数。您可以在文本框中输入您的选项的前几个字符以筛选下拉菜单项。

### Note

开始小时和分钟从零开始。开始分钟是一个介于 0 和 59 之间的数字。开始小时是一个介于 0 和 23 之间的数字。例如，每天中午 12:15 开始的每天触发的开始小时为 12，开始分钟为 15。每天午夜开始的每天触发的开始小时为 0，开始分钟为 0。每天下午 11:59 开始的每天触发的开始小时为 23，开始分钟为 59。

频率	必需参数	详细信息
每小时	开始分钟	使用 Start minute (开始分钟) 下拉菜单。
每天	开始分钟 开始小时	使用 Start minute (开始分钟) 下拉菜单。 使用 Start hour (开始小时) 下拉菜单。
每周	开始分钟 开始小时	使用 Start minute (开始分钟) 下拉菜单。

频率	必需参数	详细信息
	开始日	使用 Start hour (开始小时) 下拉菜单。 使用 Start day (开始日) 下拉菜单。
Custom	Cron 表达式	在 Cron expression (Cron 表达式) 中输入 Cron 表达式。Cron 表达式有六个必填字段，各字段之间以空格分隔。这些字段分别指定分钟、小时、月中日、月、周中日和年的开始值。您可以使用通配符指定范围、其他值等等。例如，Cron 表达式 <b>0 9 ? * MON-FRI *</b> 安排在每个工作日上午 9:00 进行构建。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南中的 <a href="#">Cron 表达式</a> 。

- 选择 Enable this trigger (启用此触发器)。

#### Note

您可以使用 <https://console.aws.amazon.com/cloudwatch/> 上的 Amazon CloudWatch 控制台来编辑源版本、超时以及 AWS CodeBuild 中不可用的其他选项。

## BitbucketWebHook事件

您可以使用 Webhook 筛选条件组来指定哪个 Bitbucket Webhook 事件触发构建。例如，您可以指定仅为指定的分支触发构建。

您可以指定多个 Webhook 筛选条件组。如果一个或多个筛选条件组上的筛选条件评估为 True，则会触发构建。创建筛选条件组时，应指定：

事件。

对于位数据段，您可以选择以下一个或多个事件: PUSH , PULL\_REQUEST\_CREATED , PULL\_REQUEST\_UPDATED , 和 PULL\_REQUEST\_MERGED...WebHook的活动类型位于 X-Event-Key 字段。下表显示了 X-Event-Key 标头值如何映射到事件类型。

#### Note

如果您创建使用 PULL\_REQUEST\_MERGED 事件类型的 Webhook 筛选条件组，则必须在 Bitbucket Webhook 设置中启用 merged 事件。

X-Event-Key 标头值	Event type
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED

一个或多个可选筛选条件。

使用正则表达式来指定筛选条件。对于触发构建的事件，与其关联的每个筛选条件都必须评估为 True。

#### ACTOR\_ACCOUNT\_ID ( ACTOR\_ID 控制台中 )

当bitbucket帐户ID与正则表达式模式匹配时，WebHook事件会触发构建。此值显示在 Webhook 筛选条件负载中的 actor 对象的 account\_id 属性中。

#### HEAD\_REF

当头部参考匹配正则表达式模式时，WebHook事件会触发构建（例如， refs/heads/branch-name 和 refs/tags/tag-name ）。HEAD\_REF 筛选器将评估分支或标签的 Git 引用名称。分支或标签名称显示在 Webhook 负载的 push 对象中的 new 对象的 name 字段中。对于拉取请求事件，分支名称显示在 Webhook 负载中的 source 对象的 branch 中的 name 字段中。

#### BASE\_REF

：当基础引用与正则表达式模式匹配时，Webhook 事件会触发构建。BASE\_REF 筛选器仅处理拉取请求事件（例如， refs/heads/branch-name ）。BASE\_REF 筛选器评估分支的 Git 引用名称。分支名称显示在 branch 对象的 name 字段中，该对象位于 Webhook 负载的 destination 对象中。

#### FILE\_PATH

：当更改的文件的路径与正则表达式模式匹配时，Webhook 会触发构建。

#### COMMIT\_MESSAGE

：当 HEAD 提交消息与正则表达式模式匹配时，Webhook 会触发构建操作。

#### Note

您可以在 Bitbucket 存储库的 webhook 设置中找到 webhook 负载。

#### 主题

- [筛选 Bitbucket Webhook 事件 \( 控制台 \) \(p. 222\)](#)
- [筛选 Bitbucket Webhook 事件 \( 开发工具包 \) \(p. 226\)](#)
- [筛选 Bitbucket Webhook 事件 \(AWS CloudFormation\) \(p. 228\)](#)

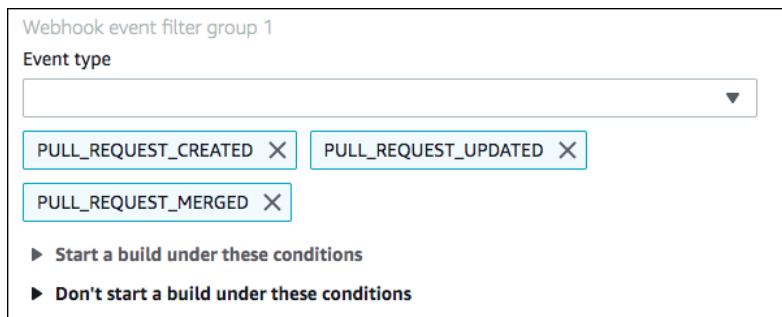
## 筛选 Bitbucket Webhook 事件 ( 控制台 )

使用 AWS 管理控制台 筛选 Webhook 事件：

1. 创建项目时，选择 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新生成)。
2. 从 Event type (事件类型) 中，选择一个或多个事件。
3. 要在事件触发构建时进行筛选，请在 Start a build under these conditions (在这些条件下启动构建) 下，添加一个或多个可选筛选条件。
4. 要在未触发事件时进行筛选，请在 Start a build under these conditions (在这些条件下不启动构建) 下，添加一个或多个可选筛选条件。
5. 选择 Add filter group (添加筛选条件组) 以添加另一个筛选条件组。

有关详细信息，请参阅 [创建构建项目 \( 控制台 \) \(p. 190\)](#) 和 [网络连接器过滤器 在 AWS CodeBuild API 参考](#)。

在此示例中，Webhook 筛选条件组仅针对拉取请求触发构建：



以两个筛选条件组为例，当一个或两个筛选条件评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称以及与 `^refs/heads/branch1!` 匹配的 HEAD 引用，指定在分支上创建或更新的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/branch1$` 匹配的 Git 引用名称，指定分支上的推送请求。

Webhook event filter group 1

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PULL\_REQUEST\_CREATED X    PULL\_REQUEST\_UPDATED X

▼ Start a build under these conditions

ACTOR\_ID - optional    HEAD\_REF - optional    BASE\_REF - optional    FILE\_PATH - optional

   ^refs/heads/branch1\$    ^refs/heads/main\$   

COMMIT\_MESSAGE - optional

► Don't start a build under these conditions

Webhook event filter group 2

Remove filter group

Event type

Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X

▼ Start a build under these conditions

ACTOR\_ID - optional    HEAD\_REF - optional    BASE\_REF - optional    FILE\_PATH - optional

   ^refs/heads/branch1\$       

COMMIT\_MESSAGE - optional

► Don't start a build under these conditions

在此示例中，Webhook 筛选条件组会针对除标记事件之外的所有请求触发构建。

Webhook event filter group 1

Event type

PUSH X    PULL\_REQUEST\_CREATED X    PULL\_REQUEST\_UPDATED X

PULL\_REQUEST\_MERGED X

► Start a build under these conditions

▼ Don't start a build under these conditions

ACTOR\_ID - optional    HEAD\_REF - optional    BASE\_REF - optional    FILE\_PATH - optional

   ^refs/tags/\*

在此示例中，仅当名称与正则表达式 `^buildspec.*` 匹配的文件发生更改时，Webhook 筛选条件组才会触发构建。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR\_ID - optional

HEAD\_REF - optional

BASE\_REF - optional

FILE\_PATH - optional

`^buildspec.*`

COMMIT\_MESSAGE -  
optional

► Don't start a build under these conditions

在此示例中，只有当其账户 ID 不与正则表达式 `actor-account-id` 匹配的 Bitbucket 用户进行更改时，Webhook 筛选条件组才会触发构建。

Note

有关如何查找位置帐户ID的信息，请参阅<https://api.bitbucket.org/2.0/users/>其中：`user-name` 是您的bitbucket用户名。

Webhook event filter group 1

Event type

PUSH X PULL\_REQUEST\_CREATED X PULL\_REQUEST\_UPDATED X  
PULL\_REQUEST\_MERGED X

▼ Start a build under these conditions

ACTOR\_ID - optional

actor-account-id

HEAD\_REF - optional

BASE\_REF - optional

FILE\_PATH - optional

► Don't start a build under these conditions

在本示例中，当 HEAD 提交消息与正则表达式 `\[CodeBuild\]` 匹配时，Webhook 筛选条件组会触发推送事件的构建。

Webhook event filter group 1

Event type

PUSH 

▼ Start a build under these conditions

ACTOR\_ID - optional

HEAD\_REF - optional

BASE\_REF - optional

FILE\_PATH - optional

COMMIT\_MESSAGE -  
optional

\[CodeBuild\]

► Don't start a build under these conditions

## 筛选 Bitbucket Webhook 事件 ( 开发工具包 )

要使用 AWS CodeBuild 开发工具包筛选 Webhook 事件，请使用 `CreateWebhook` 或 `UpdateWebhook` API 方法的请求语法中的 `filterGroups` 字段。有关详细信息，请参阅 [网络连接器过滤器](#) 在 CodeBuild API 参考。

要创建仅针对拉取请求触发构建的 Webhook 筛选条件，请在请求语法中插入以下内容：

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"  
    }  
  ]  
]
```

要创建仅针对指定分支触发构建的 Webhook 筛选条件，请使用 `pattern` 参数指定用于过滤分支名称的正则表达式。以两个筛选条件组为例，当一个或两个筛选条件评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称以及与 `^refs/heads/myBranch$` 匹配的 HEAD 引用，指定在分支上创建或更新的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/myBranch$` 匹配的 Git 引用名称，指定分支上的推送请求。

```
"filterGroups": [  
  [  
    {  
      "type": "EVENT",  
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED"  
    },  
    {  
      "type": "HEAD_REF",  
      "pattern": "^refs/heads/myBranch$"  
    }  
  ]  
]
```

```
        },
        {
          "type": "BASE_REF",
          "pattern": "^refs/heads/main$"
        }
      ],
      [
        {
          "type": "EVENT",
          "pattern": "PUSH"
        },
        {
          "type": "HEAD_REF",
          "pattern": "^refs/heads/myBranch$"
        }
      ]
    ]
```

您可以使用 `excludeMatchedPattern` 参数指定不触发构建的事件。在此示例中，将针对除标记事件之外的所有请求触发构建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

您可以创建仅在帐户 ID 为 `actor-account-id` 的 Bitbucket 用户进行更改时触发构建的筛选条件。

Note

有关如何查找位置帐户ID的信息，请参阅[https://api.bitbucket.org/2.0/users/\*user-name\*](https://api.bitbucket.org/2.0/users/user-name)其中：`user-name` 是您的bitbucket用户名。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

您可以创建只有当名称与 `pattern` 参数中的正则表达式匹配的文件发生更改时，才触发构建的筛选条件。在此示例中，筛选条件组指定仅当名称与正则表达式 `^buildspec.*` 匹配的文件更改时才触发构建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
        "pattern": "PUSH"
    },
{
    "type": "FILE_PATH",
    "pattern": "^buildspec.*"
}
]
```

您可以创建一个筛选条件，仅当 HEAD 提交消息与模式参数中的正则表达式匹配时才触发构建操作。在本示例中，筛选条件组指定仅当推送事件的 HEAD 提交消息与正则表达式 \[CodeBuild\] 匹配时，才触发构建操作。

```
"filterGroups": [
[
{
    "type": "EVENT",
    "pattern": "PUSH"
},
{
    "type": "COMMIT_MESSAGE",
    "pattern": "\[CodeBuild\]"
}
]
]
```

## 筛选 Bitbucket Webhook 事件 (AWS CloudFormation)

要使用 AWS CloudFormation 模板来筛选 Webhook 事件，请使用 AWS CodeBuild 项目的 `FilterGroups` 属性。AWS CloudFormation 模板的以下 YAML 格式的部分创建两个筛选条件组。当这两个筛选条件的其中一个或两个评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称，指定由账户 ID 不为 12345 的 Bitbucket 用户在分支上创建或更新的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/.*` 匹配的 Git 引用名称，指定在分支上创建的推送请求。
- 第三个筛选条件组指定一个推送请求，其中包含与正则表达式 `\[CodeBuild\]` 匹配的 HEAD 提交消息。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:4.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
    FilterGroups:
      - Type: EVENT
        Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
      - Type: BASE_REF
```

```
Pattern: ^refs/heads/main$  
ExcludeMatchedPattern: false  
- Type: ACTOR_ACCOUNT_ID  
  Pattern: 12345  
  ExcludeMatchedPattern: true  
  - Type: EVENT  
    Pattern: PUSH  
  - Type: HEAD_REF  
    Pattern: ^refs/heads/.*  
  - Type: EVENT  
    Pattern: PUSH  
  - Type: COMMIT_MESSAGE  
  - Pattern: \[CodeBuild\]
```

## GitHub钩型活动

您可以使用 Webhook 筛选条件组来指定哪些 GitHub Webhook 事件将触发构建。例如，您可以指定仅为指定的分支触发构建。

您可以创建一个或多个 Webhook 筛选条件组，以指定哪些 Webhook 事件触发构建。如果一个或多个筛选条件组上的所有筛选条件都评估为 True，则会触发构建。创建筛选条件组时，应指定：

事件。

对于 GitHub，您可以选择以下一个或多个事件：PUSH、PULL\_REQUEST\_CREATED、PULL\_REQUEST\_UPDATED、PULL\_REQUEST\_REOPENED 和 PULL\_REQUEST\_MERGED。webhook 事件类型位于 X-GitHub-Event 标题。在 X-GitHub-Event 标题，您可能会看到 pull\_request 或 push。对于拉动请求事件，类型位于 action Webhook 事件有效负载的字段。下表显示了 X-GitHub-Event 标头值和 webhook 拉取请求负载 action 字段值如何映射到可用的事件类型。

X-GitHub-Event 标头值	Webhook 事件负载 action 值	Event type
pull_request	opened	PULL_REQUEST_CREATED
pull_request	reopened	PULL_REQUEST_REOPENED
pull_request	synchronize	PULL_REQUEST_UPDATED
pull_request	closed 和 merged 字段为 true	PULL_REQUEST_MERGED
push	不适用	PUSH

### Note

PULL\_REQUEST\_REOPENED 事件类型可与 GitHub 和 GitHub Enterprise Server 结合使用。一个或多个可选筛选条件。

使用正则表达式来指定筛选条件。对于触发构建的事件，与其关联的每个筛选条件都必须评估为 True。ACTOR\_ACCOUNT\_ID (ACTOR\_ID 在控制台中)

当 GitHub 或 GitHub Enterprise Server 帐户 ID 与常规表达式模式匹配时，Webhook 事件会触发构建。此值在 webhook 负载中的 sender 对象的 id 属性中。

### HEAD\_REF

当头参考与正则表达式模式匹配时，Webhook 事件会触发构建（例如，refs/heads/branch-name 或 refs/tags/tag-name）。对于推送事件，引用名称在 Webhook 负载中的 ref 属性中。对于拉取请求事件，分支名称在 Webhook 负载中的 head 对象的 ref 属性中。

#### BASE\_REF

当基准参考与正则表达式模式匹配时，Webhook 事件会触发内部版本(例如, `refs/heads/branch-name`)。`BASE_REF` 筛选器只能与拉取请求事件一起使用。分支名称在 webhook 负载中的 `base` 对象的 `ref` 属性中。

#### FILE\_PATH

：当更改的文件的路径与正则表达式模式匹配时，Webhook 会触发构建。`FILE_PATH` 筛选条件可以与 GitHub 推送和拉取请求事件以及 GitHub Enterprise Server 推送事件一起使用。它不能与 GitHub Enterprise Server 拉取请求事件一起使用。

#### COMMIT\_MESSAGE

：当 HEAD 提交消息与正则表达式模式匹配时，Webhook 会触发构建操作。`COMMIT_MESSAGE` 筛选条件可以与 GitHub 推送和拉取请求事件以及 GitHub Enterprise Server 推送事件一起使用。它不能与 GitHub Enterprise Server 拉取请求事件一起使用。

#### Note

您可以在 GitHub 存储库的 webhook 设置中找到 webhook 负载。

#### 主题

- [筛选 GitHub Webhook 事件（控制台）\(p. 230\)](#)
- [筛选 GitHub Webhook 事件（开发工具包）\(p. 233\)](#)
- [筛选 GitHub Webhook 事件 \(AWS CloudFormation\) \(p. 235\)](#)

## 筛选 GitHub Webhook 事件（控制台）

进 主要来源Webhook事件,选择以下。仅当您选择了 我的GitHub帐户中的存储库 用于源存储库。

1. 创建项目时，选择 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新生成)。
2. 从 Event type (事件类型) 中，选择一个或多个事件。
3. 要在事件触发构建时进行筛选，请在 Start a build under these conditions (在这些条件下启动构建) 下，添加一个或多个可选筛选条件。
4. 要在未触发事件时进行筛选，请在 Start a build under these conditions (在这些条件下不启动构建) 下，添加一个或多个可选筛选条件。
5. 选择 添加筛选器组 如果需要,添加其他筛选器组。

有关详细信息,请参阅 [创建构建项目（控制台）\(p. 190\)](#) 和 [Webhook过滤器 在 AWS CodeBuild API参考](#).

在此示例中，Webhook 筛选条件组仅针对拉取请求触发构建：

Webhook event filter group 1

Event type

PULL\_REQUEST\_CREATED X PULL\_REQUEST\_UPDATED X

PULL\_REQUEST\_REOPENED X PULL\_REQUEST\_MERGED X

► Start a build under these conditions

► Don't start a build under these conditions

以两个 Webhook 筛选条件组为例，当一个或两个筛选条件评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称以及与 `^refs/heads/branch1$` 匹配的头部引用，指定在分支上创建、更新或重新打开的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/branch1$` 匹配的 Git 引用名称，指定分支上的推送请求。

The screenshot shows the AWS CodeBuild 'Webhook event filter' configuration interface. It displays two filter groups:

**Webhook event filter group 1**

**Event type**: Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

**PULL\_REQUEST\_CREATED** X    **PULL\_REQUEST\_UPDATED** X    **PULL\_REQUEST\_REOPENED** X

**Start a build under these conditions**

**ACTOR\_ID - optional**: [Input field]    **HEAD\_REF - optional**: `^refs/heads/branch1$`    **BASE\_REF - optional**: `^refs/heads/main$`    **FILE\_PATH - optional**: [Input field]

**Commit message - optional**: [Input field]

**Don't start a build under these conditions**

**Webhook event filter group 2**

**Remove filter group**

**Event type**: Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

**PUSH** X

**Start a build under these conditions**

**ACTOR\_ID - optional**: [Input field]    **HEAD\_REF - optional**: `^refs/heads/branch1$`    **BASE\_REF - optional**: [Input field]    **FILE\_PATH - optional**: [Input field]

**Commit message - optional**: [Input field]

**Don't start a build under these conditions**

在此示例中，Webhook 筛选条件组会针对除标记事件之外的所有请求触发构建。

Webhook event filter group 1

Event type

PUSH X PULL\_REQUEST\_CREATED X PULL\_REQUEST\_UPDATED X  
PULL\_REQUEST\_REOPENED X PULL\_REQUEST\_MERGED X

► Start a build under these conditions

▼ Don't start a build under these conditions

ACTOR\_ID - optional HEAD\_REF - optional BASE\_REF - optional FILE\_PATH - optional

^refs/tags/\*

在此示例中，仅当名称与正则表达式 `^buildspec.*` 匹配的文件发生更改时，Webhook 筛选条件组才会触发构建。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR\_ID - optional

HEAD\_REF - optional

BASE\_REF - optional

FILE\_PATH - optional

^buildspec.\*

COMMIT\_MESSAGE -  
optional

► Don't start a build under these conditions

在此示例中，只有当其账户 ID 与正则表达式 `actor-account-id` 匹配的指定 GitHub 或 GitHub Enterprise Server 用户进行更改时，Webhook 筛选条件组才会触发构建。

Note

有关如何查找GitHub帐户ID的信息,请参阅[https://api.github.com/users/\*user-name\*](https://api.github.com/users/user-name)其中：*user-name* 是您的GitHub用户名。

Webhook event filter group 1

Event type

PUSH X PULL\_REQUEST\_CREATED X PULL\_REQUEST\_UPDATED X  
PULL\_REQUEST\_REOPENED X PULL\_REQUEST\_MERGED X

▼ Start a build under these conditions

ACTOR\_ID - optional HEAD\_REF - optional BASE\_REF - optional FILE\_PATH - optional

actor-account-id

► Don't start a build under these conditions

在本示例中，当 HEAD 提交消息与正则表达式 \[CodeBuild\] 匹配时，Webhook 筛选条件组会触发推送事件的构建。

Webhook event filter group 1

Event type

PUSH X

▼ Start a build under these conditions

ACTOR\_ID - optional HEAD\_REF - optional BASE\_REF - optional FILE\_PATH - optional

COMMIT\_MESSAGE - optional

\[CodeBuild\]

► Don't start a build under these conditions

## 筛选 GitHub Webhook 事件（开发工具包）

要使用 AWS CodeBuild 开发工具包筛选 Webhook 事件，请使用 `CreateWebhook` 或 `UpdateWebhook` API 方法的请求语法中的 `filterGroups` 字段。有关详细信息,请参阅 [Webhook过滤器](#) 在 CodeBuild API 参考.

要创建仅针对拉取请求触发构建的 Webhook 筛选条件，请在请求语法中插入以下内容：

```
"filterGroups": [  
    [  
        {  
            "type": "EVENT",  
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED,  
PULL_REQUEST_MERGED"  
        }  
    ]  
]
```

要创建仅针对指定分支触发构建的 Webhook 筛选条件，请使用 `pattern` 参数指定用于过滤分支名称的正则表达式。以两个筛选条件组为例，当一个或两个筛选条件评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称以及与 `^refs/heads/myBranch$` 匹配的头部引用，指定在分支上创建、更新或重新打开的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/myBranch$` 匹配的 Git 引用名称，指定分支上的推送请求。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
]
```

您可以使用 `excludeMatchedPattern` 参数指定不触发构建的事件。例如，在此示例中，将针对除标记事件之外的所有请求触发构建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

您可以创建只有当名称与 `pattern` 参数中的正则表达式匹配的文件发生更改时，才触发构建的筛选条件。在此示例中，筛选条件组指定仅当名称与正则表达式 `^buildspec.*` 匹配的文件更改时才触发构建。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
        "pattern": "PUSH"
    },
{
    "type": "FILE_PATH",
    "pattern": "^buildspec.*"
}
]
```

您可以创建仅当账户 ID 为 `actor-account-id` 的指定 GitHub 或 GitHub Enterprise Server 用户进行更改时，才触发构建的筛选条件。

**Note**

有关如何查找 GitHub 帐户 ID 的信息，请参阅[https://api.github.com/users/\*user-name\*](https://api.github.com/users/user-name) 其中：`user-name` 是您的 GitHub 用户名。

```
"filterGroups": [
[
{
    "type": "EVENT",
    "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED"
},
{
    "type": "ACTOR_ACCOUNT_ID",
    "pattern": "actor-account-id"
}
]
]
```

您可以创建一个筛选条件，仅当 HEAD 提交消息与模式参数中的正则表达式匹配时才触发构建操作。在本示例中，筛选条件组指定仅当推送事件的 HEAD 提交消息与正则表达式 `\[CodeBuild\]` 匹配时，才触发构建操作。

```
"filterGroups": [
[
{
    "type": "EVENT",
    "pattern": "PUSH"
},
{
    "type": "COMMIT_MESSAGE",
    "pattern": "\[CodeBuild\]"
}
]
]
```

## 筛选 GitHub Webhook 事件 (AWS CloudFormation)

要使用 AWS CloudFormation 模板来筛选 Webhook 事件，请使用 AWS CodeBuild 项目的 `FilterGroups` 属性。AWS CloudFormation 模板的以下 YAML 格式的部分创建两个筛选条件组。当这两个筛选条件的其中一个或两个评估为 True 时触发构建：

- 第一个筛选条件组使用与正则表达式 `^refs/heads/main$` 匹配的 Git 引用名称，指定由账户 ID 不为 12345 的 GitHub 用户在分支上创建或更新的拉取请求。
- 第二个筛选条件组使用与正则表达式 `^refs/heads/.*` 匹配的 Git 引用名称，指定在名称与正则表达式 `READ_ME` 匹配的文件上创建的推送请求。
- 第三个筛选条件组指定一个推送请求，其中包含与正则表达式 `\[CodeBuild\]` 匹配的 HEAD 提交消息。

```
CodeBuildProject:  
  Type: AWS::CodeBuild::Project  
  Properties:  
    Name: MyProject  
    ServiceRole: service-role  
    Artifacts:  
      Type: NO_ARTIFACTS  
    Environment:  
      Type: LINUX_CONTAINER  
      ComputeType: BUILD_GENERAL1_SMALL  
      Image: aws/codebuild/standard:4.0  
    Source:  
      Type: GITHUB  
      Location: source-location  
    Triggers:  
      Webhook: true  
      FilterGroups:  
        - - Type: EVENT  
          Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED  
        - Type: BASE_REF  
          Pattern: ^refs/heads/main$  
          ExcludeMatchedPattern: false  
        - Type: ACTOR_ACCOUNT_ID  
          Pattern: 12345  
          ExcludeMatchedPattern: true  
        - - Type: EVENT  
          Pattern: PUSH  
        - Type: HEAD_REF  
          Pattern: ^refs/heads/.  
        - Type: FILE_PATH  
          Pattern: READ_ME  
          ExcludeMatchedPattern: true  
        - - Type: EVENT  
          Pattern: PUSH  
        - Type: COMMIT_MESSAGE  
          Pattern: \[CodeBuild\]
```

## 更改 AWS CodeBuild 中构建项目的设置

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包更改构建项目的设置。

如果您将测试报告添加到构建项目，请确保您的 IAM 角色具有[使用测试报告权限 \(p. 291\)](#)中介绍的权限。

### 主题

- [更改构建项目的设置（控制台）\(p. 236\)](#)
- [更改构建项目的设置（AWS CLI）\(p. 243\)](#)
- [更改构建项目的设置（AWS 开发工具包）\(p. 244\)](#)

## 更改构建项目的设置（控制台）

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。
3. 执行以下任一操作
  - 选择要更改的生成项目的链接，然后选择 Build details (生成详细信息)。
  - 选择要更改的生成项目旁边的按钮，选择 View details (查看详细信息)，然后选择 Build details (生成详细信息)。
4. 要更改项目的描述，请在 Project configuration (项目配置) 中选择 Edit (编辑)，然后输入描述。

选择 Update configuration (更新配置)。

有关此过程中引用的设置的更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)。

5. 要更改源代码位置的信息，请在 来源，选择 编辑。使用以下列表为源提供商选择适当的选项，然后选择更新来源。

Note

CodeBuild 不支持 Bitbucket 服务器。

Amazon S3

Bucket

选择包含源代码的输入桶的名称。

S3对象密钥或S3文件夹

输入zip文件的名称或包含源代码的文件夹的路径。输入正斜杠 (/) 以下载 S3 存储桶中的所有内容。

源版本

输入代表您输入文件构建的对象的版本ID。有关更多信息，请参阅[使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。)

CodeCommit

Repository

选择要使用的存储库。

参考类型

选择 分支，GIT标签，或 提交ID 要指定源代码的版本。有关更多信息，请参阅[使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。)

GIT克隆深度

选择创建一个带有指定数量的历史记录的浅克隆。如果您需要完整克隆，请选择 Full (完整)。

使用GIT子模块

选择是否要在存储库中包含GIT子模块。

Bitbucket

Repository

选择 使用OAuth连接 或 与Bitbucket应用程序密码连接 并按照指示连接 (或重新连接) 到 Bitbucket。

在您的帐户中选择一个公共存储库或存储库。

源版本

输入分支、提交ID、标签或参考和提交ID。有关更多信息，请参阅[使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)

GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

## 使用GIT子模块

选择是否要在存储库中包含GIT子模块。

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 name 位提交状态中的参数。有关详细信息，请参阅 [构建在 Bitbucket API 文件中](#)。

对于 目标URL，输入要用于 url 位提交状态中的参数。有关详细信息，请参阅 [构建在 Bitbucket API 文件中](#)。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub 钩型活动 \(p. 229\)](#) 和 [Bitbucket WebHook 事件 \(p. 221\)](#)。

## GitHub

### Repository

选择 使用 OAuth 连接 或 与 GitHub 个人访问令牌连接 并按照说明连接（或重新连接）到 GitHub 并授权访问 AWS CodeBuild.

在您的帐户中选择一个公共存储库或存储库。

### 源版本

输入分支、提交ID、标签或参考和提交ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)

### GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

## 使用GIT子模块

选择是否要在存储库中包含GIT子模块。

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 [API 版本 2016-10-06 CommitStatus](#) ( GitHub 提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态](#) 在 GitHub Developer 指南中。

对于 目标URL，输入要用于 target\_url GithubCommitStatus ( Github提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态 在GithubDeveloper指南中](#)。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#)和[BitbucketWebHook事件 \(p. 221\)](#)。

#### GitHub Enterprise Server

##### Github企业个人访问令牌

参见 [GitHub Enterprise Server 示例 \(p. 109\)](#) 有关如何将个人访问令牌复制到剪贴板的信息。  
在文本字段中粘贴令牌，然后选择 Save Token (保存令牌)。

##### Note

您只需输入并保存一次个人访问令牌。CodeBuild 将在所有未来项目中使用此令牌。

#### 源版本

输入一个拉动请求、分支、提交ID、标签或参考编号，以及提交ID。有关更多信息，请参阅 [使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。

#### GIT克隆深度

选择 Git clone depth (Git 克隆深度) 以创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

#### 使用GIT子模块

选择是否要在存储库中包含GIT子模块。

#### 生成包状态

选择 在构建开始和完成时，向源提供商报告构建状态 如果您想要将构建的开始和完成状态报告给源提供商。

##### Note

由 Webhook 触发的构建的状态将始终报告给您的源提供商。

#### 不安全的SSL

在连接到GithubEnterprise项目资料库时，选择忽略SSL警告。

选择 每次将代码更改推送到此存储库时重建 如果您想要 CodeBuild 每次将代码更改推送到此存储库时，要构建源代码。Webhook 仅允许用于您自己的 Bitbucket、GitHub 或 GitHub Enterprise 存储库。

对于 状态上下文，输入要用于 context GithubCommitStatus ( Github提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态 在GithubDeveloper指南中](#)。

对于 目标URL，输入要用于 target\_url GithubCommitStatus ( Github提交状态 ) 中的参数。有关详细信息，请参阅 [创建提交状态 在GithubDeveloper指南中](#)。

如果您选择了 Rebuild every time a code change is pushed to this repository (每次将代码更改推送到此存储库时都会重新构建)，则在 Event type (事件类型) 中，选择要触发构建的事件。使用正则表达式创建筛选条件。如果未指定筛选条件，则所有更新和创建拉取请求及所有推送事件都会触发构建。有关更多信息，请参阅 [GitHub钩型活动 \(p. 229\)](#)和[BitbucketWebHook事件 \(p. 221\)](#)。

要更改 CodeBuild 是否可以修改您用于此项目的服务角色，请选中或清除 Allow AWS CodeBuild to modify this service role so it can be used with this build project (允许 AWS CodeBuild 修改此服务角色，以便它可以用于此构建项目)。如果清除该复选框，则必须使用附加有 CodeBuild 权限的服务角色。有关更多信息，请参阅 [向 IAM 组或 IAM 用户添加 CodeBuild 访问权限 \(p. 353\)](#) 和 [创建 CodeBuild 服务角色 \(p. 357\)](#)。

6. 要更改有关构建环境的信息，请在 环境，选择 编辑。对构建环境类型（例如，Environment image (环境映像)、Operating system (操作系统)、Runtime (运行时)、Runtime version (运行时版本)、Custom image (自定义映像)、Other location (其他位置)、Amazon ECR repository (Amazon ECR 存储库) 或 Amazon ECR image (Amazon ECR 映像)）进行适当的更改。
7. 如果您计划使用此构建项目来构建 Docker 映像且指定的构建环境不是由具有 Docker 支持的 CodeBuild 提供，请选择 Privileged (特权)。否则，尝试与 Docker 守护程序交互的所有关联的构建都将失败。您还必须启动 Docker 守护程序，以便您的构建可根据需要与之交互。要这样做，您可以通过运行以下构建命令，在您构建规范文件的 install 阶段初始化 Docker 守护程序。（如果指定的构建环境映像由具有 Docker 支持的 CodeBuild 提供，请不要运行以下构建命令。）

**Note**

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

8. 要更改有关 CodeBuild 服务角色的信息，请在 Service role (服务角色) 中更改 New service role (新服务角色)、Existing service role (现有服务角色) 或 Role name (角色名称) 的值。

**Note**

当您使用控制台来创建或更新构建项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

9. 要更改有关生成超时的信息，请在 Additional configuration (其他配置) 中，对于 Timeout (超时)，更改 hours (小时) 和 minutes (分钟) 的值。如果 hours 和 minutes 都留空，则默认值为 60 分钟。
10. 要更改有关您在 Amazon VPC 中创建的 VPC 的信息，请在 Additional configuration (其他配置) 中更改 VPC、Subnets (子网) 和 Security groups (安全组) 的值。
11. 要更改有关您在 Amazon EFS 中创建的文件系统的信息，请在 Additional configuration (其他配置) 中更改其 Identifier (标识符)、ID、Directory path (目录路径)、Mount point (挂载点) 和 Mount options (挂载选项) 的值。有关更多信息，请参阅 [适用于 AWS CodeBuild 的 Amazon Elastic File System 示例 \(p. 53\)](#)。)
12. 要更改用于运行生成的内存量和 vCPU 量，请在 Additional configuration (其他配置) 中，更改 Compute (计算) 的值。
13. 要更改有关您希望构建使用的环境变量的信息，请在 Additional configuration (其他配置) 中，对于 Environment variables (环境变量)，更改 Name (名称)、Value (值) 和 Type (类型) 的值。使用 Add environment variable (添加环境变量) 添加环境变量。选择您不再想使用的环境变量旁边的 Remove (删除)。

其他人可以使用 CodeBuild 控制台和 AWS CLI 查看环境变量。如果您不担心环境变量的可见性，请设置 Name 和 Value 字段，然后将 Type 设置为 Plaintext。

我们建议您将具有敏感值（例如 AWS 访问密钥 ID、AWS 秘密访问密钥或密码）的环境变量作为参数存储在 Amazon EC2 Systems Manager Parameter Store 或 AWS Secrets Manager 中。

如果您使用的是 Amazon EC2 Systems Manager Parameter Store，则对于 Type (类型) , 选择 Parameter (参数)。对于 Name (名称) , 输入标识符供 CodeBuild 引用。对于 Value (值) , 按照 Amazon EC2 Systems Manager Parameter Store 中存储的参数名称输入参数名称。使用名为 /CodeBuild/dockerLoginPassword 的参数作为示例，对于 Type (类型) , 选择 Parameter (参数)。对于 名称 , 输入 LOGIN\_PASSWORD...对于 值 , 类型 /CodeBuild/dockerLoginPassword.

#### Important

如果您使用 Amazon EC2 Systems Manager Parameter Store , 我们建议您使用以 /CodeBuild/ 开头的参数名称 (例如 , /CodeBuild/dockerLoginPassword ) 来存储参数。可以使用 CodeBuild 控制台在 Amazon EC2 Systems Manager 中创建参数。选择 Create parameter (创建参数) , 然后按照对话框中的说明操作。(在该对话框中 , 对于 KMS key (KMS 密钥) , 您可以指定您账户中的 AWS KMS 密钥的 ARN。Amazon EC2 Systems Manager 使用此密钥在存储过程中加密参数的值并在检索过程中解密参数的值。) 如果您使用 CodeBuild 控制台创建了一个参数 , 控制台将在参数名称被存储时以 /CodeBuild/ 作为它的开头。有关更多信息 , 请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的参数，则构建项目的服务角色必须允许 ssm:GetParameters 操作。如果您之前选择了 New service role (新建服务角色) , CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是 , 如果您选择了 Existing service role (现有服务角色) , 必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的但参数名称不以 /CodeBuild/ 开头的参数 , 且您选择了 New service role (新建服务角色) , 您必须更新该服务角色以允许访问不以 /CodeBuild/ 开头的参数名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的参数名称。

如果您选择 New service role (新建服务角色) , 服务角色将包含解密 Amazon EC2 Systems Manager Parameter Store 中 /CodeBuild/ 命名空间下的所有参数的权限。

您设置的环境变量将替换现有的环境变量。例如 , 如果 Docker 映像已经包含一个名为 MY\_VAR 的环境变量 (值为 my\_value) , 并且您设置了一个名为 MY\_VAR 的环境变量 (值为 other\_value) , 那么 my\_value 将被替换为 other\_value。同样 , 如果 Docker 映像已经包含一个名为 PATH 的环境变量 (值为 /usr/local/sbin:/usr/local/bin) , 并且您设置了一个名为 PATH 的环境变量 (值为 \$PATH:/usr/share/ant/bin) , 那么 /usr/local/sbin:/usr/local/bin 将被替换为文本值 \$PATH:/usr/share/ant/bin。

请勿使用以 CODEBUILD\_ 打头的名称设置任何环境变量。此前缀是专为内部使用预留的。

如果具有相同名称的环境变量在多处都有定义 , 则应按照如下方式确定其值 :

- 构建操作调用开始时的值优先级最高。
- 构建项目定义中的值优先级次之。
- 构建规范声明中的值优先级最低。

如果您使用 Secrets Manager , 对于 Type (类型) , 请选择 Secrets Manager。对于 Name (名称) , 输入标识符供 CodeBuild 引用。对于 值 , 输入 reference-key 使用模式 `secret-id:json-key:version-stage:version-id...` 有关信息 , 请参阅 [Secrets Manager reference-key in the buildspec file](#)。

#### Important

如果您使用 Secrets Manager , 我们建议您以 /CodeBuild/ 开头的名称存储密钥 (例如 /CodeBuild/dockerLoginPassword )。有关更多信息 , 请参阅 AWS Secrets Manager 用户指南中的 [什么是 AWS Secrets Manager ?](#)。

如果您的构建项目引用了 Secrets Manager 中存储的密钥 , 则构建项目的服务角色必须允许 secretsmanager:GetSecretValue 操作。如果您之前选择了 New service role (新建服务角色) , CodeBuild 将在您的构建项目的默认服务角色中包含此操作。但是 , 如果您选择了 Existing service role (现有服务角色) , 必须单独将此操作添加到您的服务角色中。

如果您的构建项目引用了 Secrets Manager 中存储的但密钥名称不以 /CodeBuild/ 开头的密钥 , 且您选择了 New service role (新建服务角色) , 您必须更新该服务角色以允许访问不以 /

CodeBuild/ 开头的密钥名称。这是因为该服务角色仅允许访问以 /CodeBuild/ 开头的密钥名称。

如果您选择 New service role (新建服务角色) , 该服务角色将拥有解密 Secrets Manager 中 /CodeBuild/ 命名空间下的所有密钥的权限。

14. 选择 Update environment (更新环境)。
15. 要更改项目的构建规格 , 请在 建立规格 , 选择 编辑. 默认情况下 , CodeBuild 在源代码根目录中查找名为 buildspec.yml 的文件。如果您的BuildSpec文件使用不同的名称或位置 , 请在“源根”中输入其路径 BuildSpec名称 ( 例如 , **buildspec-two.yml** 或 **configuration/buildspec.yml**... 如果 BuildSpec文件位于S3桶中 , 则必须在相同的 AWS 地区作为您的建立项目。使用其 ARN 指定构建规范文件 ( 例如 , `arn:aws:s3:::my-codebuild-sample2/buildspec.yml` ) 。
  - 如果您的源代码以前不包含 buildspec.yml 文件 , 但现在却包含此文件 , 请选择 Use a buildspec file ( 使用 buildspec 文件)。
  - 如果您的源代码以前包含 buildspec.yml 文件但现在不包含此文件 , 请选择 Insert build commands ( 插入构建命令) , 然后在 Build commands (构建命令) 中 , 输入命令。
16. 选择 Update buildspec (更新构建规范)。
17. 要更改批次构建配置的信息 , 请在 批次配置 , 选择 编辑 并根据需要更新发泡值。

#### 批量服务角色

选择以下选项之一

- 如果您没有批量服务角色 , 请选择 新服务角色. 在 服务角色 , 输入新角色的名称。
- 如果您有批量服务角色 , 请选择 现有服务角色. 在 服务角色 , 选择服务角色。

#### 批次的允许计算类型

选择批次允许的计算类型。选择所有适用项。

#### 批次允许的最大构建数

输入批次允许的最大建立数。如果批次超出此限制 , 则批次将失败。

#### 批处理超时

输入批次构建完成的最大时间量。

#### 合并工件

选择 将所有图像从批次中结合到单个位置 将所有的图像从批次中的所有物料组合到一个位置。

18. 选择 更新批次配置。
19. 要更改关于构建输出工件位置和名称的信息 , 请在 伪影 , 选择 编辑 , 然后更改值 类型 , 名称 , 路径 , Namespace类型 , 或 桶名称.
20. 要更改有关 AWS KMS 客户托管密钥 (CMK) 的信息 , 请在 Additional configuration (其他配置) 中 , 更改 Encryption key (加密密钥) 的值。

#### Important

如果您将 Encryption key (加密密钥) 留空 , 则 CodeBuild 会将 AWS 托管 CMK 用于您 AWS 账户中的 Amazon S3。

21. 使用缓存可节省构建时间 , 因为构建环境的可重用部分存储在缓存中 , 并且可跨构建使用。有关在生成规范文件中指定缓存的信息 , 请参阅[构建规范语法 \(p. 137\)](#)。要更改有关缓存的信息 , 请展开 Additional configuration (其他配置)。在 Cache type (缓存类型) 中 , 执行下列操作之一 :

- 如果您之前选择了缓存但现在不想使用缓存 , 请选择 No cache (无缓存)。
- 如果您之前选择了 No cache (无缓存) 但现在想使用缓存 , 请选择 Amazon S3 , 然后执行以下操作 :
  - 对于 Cache bucket (缓存存储桶) , 选择存储缓存的 S3 存储桶的名称。
  - ( 可选 ) 对于 Cache path prefix (缓存路径前缀) , 输入 Amazon S3 路径前缀。缓存路径前缀的值类似于目录名称。您可以使用它在存储桶的同一目录下存储缓存。

### Important

请不要在 Path prefix (路径前缀) 的末尾附加正斜杠 (/)。

22. 要更改您的日志设置，请在 Logs (日志) 中选中或清除 CloudWatch logs (CloudWatch 日志) 和 S3 logs (S3 日志)。

如果选择 CloudWatch logs (CloudWatch 日志) :

- 在 Group name (组名称) 中，输入您的 Amazon CloudWatch Logs 组的名称。
- 在 Stream name (流名称) 中，输入您的 Amazon CloudWatch Logs 流名称。

如果选择 S3 logs (S3 日志) :

- 从 Bucket (存储桶) 中，选择您的日志的 S3 存储桶的名称。
- 在 Path prefix (路径前缀) 中，输入您的日志的前缀。
- 如果您不希望加密您的 S3 日志，请选择 Remove S3 log encryption (删除 S3 日志加密)。

23. 要更改有关存储构建输出构件的方式的信息，请在 Additional configuration (其他信息) 中，更改 Artifacts packaging (构件打包) 的值。

24. 要更改是否加密构建构件，请使用 Disable artifacts encryption (禁用构件加密)。

25. 选择 Update artifacts (更新项目)。

## 更改构建项目的设置 (AWS CLI)

有关将 AWS CLI 与 AWS CodeBuild 结合使用的信息，请参阅[命令行参考 \(p. 363\)](#)。

更新 CodeBuild 项目与 AWS CLI，您创建了一个带有更新属性的 JSON 文件，并将该文件传递到 `update-project` 命令。更新文件中未包含的任何属性保持不变。

在更新 JSON 文件中，只有 `name` 属性和修改的属性为必填项。The `name` 属性标识要修改的项目。对于任何修改后的结构，这些结构的必要参数也必须包括。例如，要修改项目环境，`environment/type` 和 `environment/computeType` 属性为必填项。以下是更新环境图像的示例：

```
{  
  "name": "<project-name>",  
  "environment": {  
    "type": "LINUX_CONTAINER",  
    "computeType": "BUILD_GENERAL1_SMALL",  
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:3.0"  
  }  
}
```

如果需要获取项目的当前属性值，请使用 `batch-get-projects` 命令以获取正在修改的项目的当前属性，并将输出写入文件。

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

该 `project-info.json` 文件包含一系列项目，因此无法直接使用该文件更新项目。但是，您可以复制要从 `project-info.json` 将文件粘贴到您的更新文件中，作为您要修改属性的基准。有关更多信息，请参阅[查看构建项目的详细信息 \(AWS CLI\) \(p. 213\)](#)。)

修改更新 JSON 文件，详见[创建构建项目 \(AWS CLI\) \(p. 198\)](#)，并保存您的结果。修改更新 JSON 文件后，运行 `update-project` 命令，通过 UpdateJSON 文件。

```
aws codebuild update-project --cli-input-json file://<update-project-file>
```

如果成功，则输出中会出现更新的项目 JSON。如果缺少任何必要的参数，输出中会显示错误消息，以识别缺少参数。例如，如果 environment/type 参数缺失：

```
aws codebuild update-project --cli-input-json file://update-project.json
Parameter validation failed:
Missing required parameter in environment: "type"
```

## 更改构建项目的设置 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 删除 AWS CodeBuild 中的构建项目

您可以使用 CodeBuild 控制台、AWS CLI 或 AWS 开发工具包删除 CodeBuild 中的构建项目。如果删除项目，将不会删除其构建。

### Warning

您不能删除具有构建和资源策略的项目。要删除具有资源策略和构建的项目，您必须先删除资源策略及其构建。

### 主题

- [删除构建项目 \( 控制台 \) \(p. 244\)](#)
- [删除构建项目 \(AWS CLI\) \(p. 244\)](#)
- [删除构建项目 \( AWS 开发工具包 \) \(p. 245\)](#)

## 删除构建项目 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。
3. 执行以下操作之一：
  - 选择您要删除的生成项目旁边的单选按钮，然后选择 Delete (删除)。
  - 选择您要删除的构建项目的链接，然后选择 Delete。

### Note

默认情况下，仅显示 10 个最新的构建项目。要查看更多生成项目，请为 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头查看项目。

## 删除构建项目 (AWS CLI)

1. 运行 delete-project 命令：

```
aws codebuild delete-project --name name
```

替换以下占位符：

- *name*：必需的字符串。要删除的构建项目的名称。要获取可用构建项目的列表，请运行 list-projects 命令。有关更多信息，请参阅[查看构建项目名称的列表 \(AWS CLI\) \(p. 212\)](#)。

2. 如果成功，则输出中不会出现任何数据和错误。

有关将 AWS CLI 与 AWS CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

## 删除构建项目 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

# 使用共享项目。

项目共享允许项目拥有者与其他 AWS 账户或用户共享其 AWS CodeBuild 项目。在此模型中，拥有项目的账户（拥有者）将与其他账户（使用者）共享项目。使用者无法编辑或运行项目。

内容：

- [共享项目的先决条件 \(p. 245\)](#)
- [访问与您共享的共享项目的先决条件 \(p. 245\)](#)
- [相关服务 \(p. 245\)](#)
- [共享项目 \(p. 246\)](#)
- [取消共享已共享的项目 \(p. 247\)](#)
- [标识共享的项目 \(p. 248\)](#)
- [共享项目权限 \(p. 248\)](#)

## 共享项目的先决条件

要共享项目，您的 AWS 账户必须拥有该项目。无法共享已与您共享的项目。

### 访问与您共享的共享项目的先决条件

要访问共享报告组，使用者的 IAM 角色需要 `BatchGetProjects` 权限。您可以将以下策略附加到其 IAM 角色：

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "*"  
    ],  
    "Action": [  
        "codebuild:BatchGetProjects"  
    ]  
}
```

有关更多信息，请参阅 [为 AWS CodeBuild 使用基于身份的策略 \(p. 329\)](#)。)

## 相关服务

项目共享与 AWS Resource Access Manager (AWS RAM) 集成，后者是一项服务，使您可以与任何 AWS 账户或通过 AWS Organizations 共享 AWS 资源。通过使用 AWS RAM，您可以通过创建资源共享来共享资源，该共享指定要共享的资源和要与其共享资源的使用者。使用者可以是单个 AWS 账户、AWS Organizations 中的组织部门或 AWS Organizations 中的整个组织。

有关更多信息，请参阅 [AWS RAM 用户指南](#)。

## 共享项目

消费者可以使用 AWS CLI 和 AWS CodeBuild 控制台查看您共享的项目和建立的内容。使用者无法编辑或运行项目。

您可以将项目添加到现有资源共享，也可以在 [AWS RAM 控制台](#) 中创建资源共享。

### Note

您不能删除其构建已添加到资源共享的项目。

要与组织单位或整个组织共享项目，您必须启用与 AWS Organizations 的共享。有关详细信息，请参阅 [启用与 AWS Organizations 在 AWS RAM 用户指南](#)。

您可以使用 AWS CodeBuild 控制台、AWS RAM 控制台或 AWS CLI 共享您拥有的项目。

### 共享您拥有的项目（CodeBuild 控制台）

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。

### Note

默认情况下，仅显示 10 个最新的构建项目。要查看更多生成项目，请选择齿轮图标，然后为 Projects per page (每页项目数) 选择不同值，或使用向后和向前箭头。

3. 选择要共享的项目，然后选择 Share (共享)。有关详细信息，请参阅 [创建资源共享 在 AWS RAM 用户指南](#)。

### 共享您拥有的项目（AWS RAM 控制台）

参见 [创建资源共享 在 AWS RAM 用户指南](#)。

### 共享您拥有的项目（AWS RAM 命令）

使用 [create-resource-share](#) 命令。

### 共享您拥有的项目（CodeBuild 命令）

使用 [put-resource-policy](#) 命令：

1. 创建一个名为 policy.json 的文件，并将以下内容复制到该文件中。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "<consumer-aws-account-id-or-user>"  
            },  
            "Action": [  
                "codebuild:BatchGetProjects",  
                "codebuild:BatchGetBuilds",  
                "codebuild>ListBuildsForProject"],  
            "Resource": "<arn-of-project-to-share>"  
        }  
    ]  
}
```

2. 使用项目 ARN 和标识符更新 policy.json 以便共享项目。以下示例授予对由123456789012 标识的 AWS 帐户的根用户的只读访问权限。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "123456789012"
            ]
        },
        "Action": [
            "codebuild:BatchGetProjects",
            "codebuild:BatchGetBuilds",
            "codebuild>ListBuildsForProject"
        ],
        "Resource": "arn:aws:codebuild:us-west-2:123456789012:project/my-project"
    }
]
```

3. 运行 [投资资源-政策](#) 命令。

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. 获取 AWS RAM 资源共享ARN。

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

这将返回类似于此的回复:

```
{
    "resources": [
        {
            "arn": "<project-arn>",
            "type": "<type>",
            "resourceShareArn": "<resource-share-arn>",
            "creationTime": "<creation-time>",
            "lastUpdatedTime": "<last-update-time>"
        }
    ]
}
```

从回复中复制 [\*<resource-share-arn>\*](#) 在下一步中使用的值。

5. 运行 AWS RAM [推广-资源-共享-从策略创建](#) 命令。

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-
share-arn>
```

## 取消共享已共享的项目

取消共享的项目（包括其构建）只能由其拥有者访问。如果取消共享项目，先前与其共享项目的任何 AWS 账户或用户都无法访问该项目或其构建。

要取消共享您拥有的已共享项目，必须从资源共享中将其删除。您可以使用 AWS CodeBuild 控制台、AWS RAM 控制台或 AWS CLI 执行此操作。

取消共享您拥有的共享项目（AWS RAM 控制台）

参见 [更新资源共享](#) 在 AWS RAM 用户指南。

取消共享您拥有的共享项目 (AWS CLI)

使用 [disassociate-resource-share](#) 命令。

要取消您拥有的项目 (CodeBuild 命令)

运行 [delete-resource-policy](#) 命令，并指定要取消共享的项目的 ARN：

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

## 标识共享的项目

拥有者和使用者可以使用 AWS CLI 标识共享项目。

标识与您的 AWS 账户或用户共享的项目 (AWS CLI)

使用 [list-shared-projects](#) 命令返回与您共享的项目。

## 共享项目权限

### 拥有者的权限

项目拥有者可以编辑项目并使用它来运行构建。

### 使用者的权限

项目使用者可以查看项目及其构建，但不能编辑项目或使用项目运行构建。

## 在 AWS CodeBuild 中标记项目

标签 是您或 AWS 分配给 AWS 资源的自定义属性标签。每个 AWS 标签具有两个部分：

- 标签键（例如，`CostCenter`、`Environment`、`Project` 或 `Secret`）。标签键区分大小写。
- 一个称为 `标签值` 的可选字段（例如，`111122223333`、`Production` 或团队名称）。省略标签值与使用空字符串相同。与标签键一样，标签值区分大小写。

这些被统称为键值对。有关项目可拥有的标签数量以及标签键和值的限制，请参阅[Tags \(p. 401\)](#)。

标签有助于您标识和组织 AWS 资源。许多 AWS 服务支持标记，因此，您可以将同一标签分配给来不同服务的资源，以指示这些资源是相关的。例如，您可以将相同的标签分配给为 S3 存储桶分配的 CodeBuild 项目。有关使用标签的更多信息，请参阅[标记最佳实践白皮书](#)。

在 CodeBuild 中，主要资源是项目和报告组。您可以使用 CodeBuild 控制台、AWS CLI、CodeBuild API 或 AWS 开发工具包为项目添加、管理和移除标签。除了通过标签标识、组织和跟踪项目之外，您可以在 IAM 策略中使用标签，帮助控制哪些人可以查看并与您的项目交互。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

### 主题

- [为项目添加标签 \(p. 248\)](#)
- [查看项目的标签 \(p. 249\)](#)
- [编辑项目的标签 \(p. 250\)](#)
- [从项目中移除标签 \(p. 250\)](#)

## 为项目添加标签

为项目添加标签可以帮助您标识和组织您的 AWS 资源并管理对其的访问。首先，为项目添加一个或多个标签（键值对）。请记住，项目可以拥有的标签数量有限制。键和值字段中可以使用的字符有限制。有关更多

信息，请参阅[Tags \(p. 401\)](#)。有了标签后，您可以创建 IAM 策略以根据这些标签管理对项目的访问。您可以使用 CodeBuild 控制台或 AWS CLI 为项目添加标签。

**Important**

为项目添加标签之前，请务必查看是否存在任何 IAM 策略可能使用标签来控制对资源（如构建项目）的访问。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

有关在创建项目时为其添加标签的更多信息，请参阅[为项目添加标签（控制台）\(p. 249\)](#)。

**主题**

- [为项目添加标签（控制台）\(p. 249\)](#)
- [为项目添加标签 \(AWS CLI\) \(p. 249\)](#)

## 为项目添加标签（控制台）

您可以使用 CodeBuild 控制台为 CodeBuild 项目添加一个或多个标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Build projects (构建项目) 中，选择要在其中添加标签的项目的名称。
3. 在导航窗格中，选择 Settings。选择 Build project tags (构建项目标签)。
4. 如果尚未向项目添加任何标签，请选择 Add tag (添加标签)。反之，请选择 Edit (编辑)，然后选择 Add tag (添加标签)。
5. 在 Key (键) 中，输入标签的名称。您可以在 Value (值) 中添加可选的标签值。
6. (可选) 要添加其他标签，请再次选择 Add tag (添加标签)。
7. 添加完标签后，选择 Submit (提交)。

## 为项目添加标签 (AWS CLI)

要在创建项目时为其添加标签，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。在 `create-project.json` 中，添加您的标签。

在这些步骤中，我们假设您已安装最新版本的 AWS CLI 或已更新到当前版本。有关更多信息，请参阅[安装 AWS Command Line Interface](#)。

如果成功，该命令不返回任何内容。

## 查看项目的标签

标签可以帮助您标识和组织您的 AWS 资源并管理对其的访问。有关使用标签的更多信息，请参阅[标记最佳实践白皮书](#)。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

## 查看项目的标签（控制台）

您可以使用 CodeBuild 控制台查看与 CodeBuild 项目关联的标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Build projects (构建项目) 中，选择要在其中查看标签的项目的名称。
3. 在导航窗格中，选择 Settings。选择 Build project tags (构建项目标签)。

## 查看项目的标签 (AWS CLI)

要查看构建项目的标签，请运行以下命令。使用项目名称作为 `--names` 参数。

```
aws codebuild batch-get-projects --names your-project-name
```

如果成功，此命令会返回有关构建项目的 JSON 格式信息，其中包括如下内容：

```
{  
    "tags": {  
        "Status": "Secret",  
        "Team": "JanesProject"  
    }  
}
```

如果项目没有标签，则 tags 部分为空：

```
"tags": []
```

## 编辑项目的标签

您可以更改与项目关联的标签值。您也可以更改标签键的名称，这相当于删除当前的标签并使用新名称和相同的值添加一个不同的标签。请记住，键和值字段中可以使用的字符有限制。有关更多信息，请参阅 [Tags \(p. 401\)](#)。

### Important

编辑项目的标签会影响对该项目的访问。编辑项目的标签名称（键）或值之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如构建项目）的访问。有关基于标签的访问策略示例，请参阅 [使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

### 编辑项目的标签（控制台）

您可以使用 CodeBuild 控制台编辑与 CodeBuild 项目关联的标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Build projects (构建项目) 中，选择要在其中编辑标签的项目的名称。
3. 在导航窗格中，选择 Settings。选择 Build project tags (构建项目标签)。
4. 选择 Edit。
5. 请执行下列操作之一：
  - 要更改标签，则在 Key (键) 中输入新名称。更改标签的名称相当于删除标签并使用新的键名添加新标签。
  - 要更改标签的值，则输入新值。如果您想将标签值清空，请删除当前的值并将字段保留为空白。
6. 编辑完标签后，选择 Submit (提交)。

### 编辑项目的标签 (AWS CLI)

要添加、更改或移除构建项目中的标签，请参阅 [更改构建项目的设置 \(AWS CLI\) \(p. 243\)](#)。更新用于更新项目的 JSON 格式数据中的 tags 部分。

## 从项目中移除标签

您可以移除与项目关联的一个或多个标签。删除标签不会从与该标签关联的其他 AWS 资源中删除该标签。

### Important

移除项目的标签会影响对该项目的访问。从项目中移除标签之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如构建项目）的访问。有关基于标签的访问策略示例，请参阅 [使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

## 从项目中移除标签（控制台）

您可以使用 CodeBuild 控制台移除标签和 CodeBuild 项目之间的关联。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Build projects (构建项目) 中，选择要在其中移除标签的项目的名称。
3. 在导航窗格中，选择 Settings。选择 Build project tags (构建项目标签)。
4. 选择 Edit。
5. 找到要移除的标签，然后选择 Remove tag (移除标签)。
6. 移除标签之后，选择 Submit (提交)。

## 从项目中移除标签 (AWS CLI)

要从构建项目中删除一个或多个标签，请参阅[更改构建项目的设置 \(AWS CLI\) \(p. 243\)](#)。使用不包含待删除标签的更新标签列表来更新采用 JSON 格式数据的 tags 部分。如果要删除所有标签，请将 tags 部分更新为：

```
"tags: []"
```

### Note

如果删除 CodeBuild 构建项目，则会从删除的构建项目中移除所有标签关联。您无需在删除构建项目之前移除标签。

## 批次建立在 AWS CodeBuild

AWS CodeBuild 支持执行并发和协调的项目构建 批次构建。有关更多信息，请参阅以下主题：

- [批量构建BuildSpec参考 \(p. 155\)](#)
- [批次配置 \(p. 196\)](#)
- [运行批次构建 \( AWS CLI\) \(p. 259\)](#)
- [停止批次构建 AWS CodeBuild \(p. 269\)](#)

## 使用 AWS CodeBuild 中的构建

一个构建 代表一组由 AWS CodeBuild 执行的操作，目的是基于一组输入构件（例如，一系列 Java 类文件）创建输出构件（例如，JAR 文件）。

运行多个构建时，以下规则适用：

- 如果可能，构建会同时运行。最大并发运行构建数会发生变化。有关更多信息，请参阅 [构建 \(p. 400\)](#)。)
- 如果并发运行构建数达到其限制，则构建会排队。队列中的最大构建数为并发构建限制的 5 倍。有关更多信息，请参阅 [构建 \(p. 400\)](#)。)
- 从队列中删除在超时值中指定的分钟数后，不会启动队列中的构建。默认超时值为 8 小时。运行构建时，可以使用介于 5 分钟到 8 小时之间的值覆盖构建队列超时。有关更多信息，请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#)。)
- 无法预测排队的构建的开始顺序。

### Note

您可以访问生成包一年的历史记录。

在使用构建时，您可以执行以下任务：

#### 主题

- [在 AWS CodeBuild 中运行构建 \(p. 252\)](#)
- [查看 AWS CodeBuild 中的构建详细信息 \(p. 261\)](#)
- [查看 AWS CodeBuild 中的构建 ID 的列表 \(p. 263\)](#)
- [查看 AWS CodeBuild 中构建项目的构建 ID 列表 \(p. 266\)](#)
- [在 AWS CodeBuild 停止构建 \(p. 268\)](#)
- [停止批次构建 AWS CodeBuild \(p. 269\)](#)
- [重试一个内部版本 AWS CodeBuild \(p. 270\)](#)
- [在会话管理器中查看运行的构建 \(p. 271\)](#)
- [在 AWS CodeBuild 中删除构建 \(p. 274\)](#)

## 在 AWS CodeBuild 中运行构建

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包运行 CodeBuild 中的构建项目。

#### 主题

- [运行构建 \(控制台\) \(p. 252\)](#)
- [运行构建 \(AWS CLI\) \(p. 256\)](#)
- [运行批次构建 \(AWS CLI\) \(p. 259\)](#)
- [开始自动运行构建 \(AWS CLI\) \(p. 260\)](#)
- [停止自动运行构建 \(AWS CLI\) \(p. 261\)](#)
- [运行构建 \(AWS 开发工具包\) \(p. 261\)](#)

## 运行构建 (控制台)

要使用 AWS CodePipeline 运行 CodeBuild 中的构建项目，可跳过这些步骤并按照[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)中的说明操作。

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 执行以下任一操作
  - 如果您刚刚完成创建一个构建项目，建立项目: **project-name** 页面应显示。选择 Start build。
  - 如果您之前创建了构建项目，则应在导航窗格中选择 Build projects。选择构建项目，然后选择 Start build。
3. 在 Start build (启动构建) 页面上，执行下列操作之一：
  - 对于 Amazon S3，输入要构建的输入构件版本的版本 ID 作为可选的 Source version (源版本) 值。如果 Source version (源版本) 留空，则使用最新版本。
  - 对于 CodeCommit，为 Reference type (参考类型) 选择 Branch (分支)、Git tag (Git 标签) 或 Commit ID (提交 ID)。接下来，选择分支、GIT标签，或输入一个提交ID以指定源代码的版本。有关更多信息，请参阅[使用 AWS CodeBuild 的源版本示例 \(p. 127\)](#)。更改 Git clone depth 的值。这会创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。
  - 对于 GitHub 或 GitHub Enterprise Server，请输入提交 ID、拉取请求 ID、分支名称或您要生成的源代码版本的标签名称，作为可选 Source version (源版本) 值。如果您要指定拉取请求 ID，则必须使用格式 pr/**pull-request-ID** (例如，pr/25)。如果您要指定分支名称，则将使用分支的 HEAD 提交 ID。如果 Source version (源版本) 留空，则将使用默认分支的 HEAD 提交 ID。更改 Git clone depth 的值。这会创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。

- 对于 Bitbucket，请输入提交 ID、分支名称或您要生成的源代码版本的标签名称作为可选 Source version (源版本) 值。如果您要指定分支名称，则将使用分支的 HEAD 提交 ID。如果 Source version (源版本) 留空，则将使用默认分支的 HEAD 提交 ID。更改 Git clone depth 的值。这会创建一个浅克隆，其历史记录会截断至指定数量的提交。如果您需要完整克隆，请选择 Full (完整)。
  - 要仅对此构建使用不同的源提供商，请选择 Advanced build options (高级构建选项)。有关源提供商选项和设置的更多信息，请参阅[Choose source provider](#)。
4. 选择 Advanced build overrides (高级构建覆盖)。

在这里，您可以仅为此构建更改设置。此部分中的设置是可选的。

下方 构建配置，从以下内容中选择：

单个构造

选择此选项来执行单个构造。

批次构建

选择此选项来执行批次构建。

下方 批次配置，为此构建设置批次构建配置覆盖。

Note

此部分仅在时间显示时显示 批次构建 选择在 构建配置。

服务 角色

提供批量建立的服务角色。选择以下选项之一

- 如果您没有批量服务角色，请选择 新服务角色。在 服务角色，输入新角色的名称。
- 如果您有批量服务角色，请选择 现有服务角色。在 服务角色，选择服务角色。

更改不管是否 CodeBuild 可以修改您用于此构建、选择或清除的批处理角色 允许 AWS CodeBuild 修改此服务角色，这样它就可以与此建立项目一起使用。如果清除该复选框，则必须使用附加有 CodeBuild 权限的服务角色。有关更多信息，请参阅[向 IAM 组或 IAM 用户添加 CodeBuild 访问权限 \(p. 353\)](#)和[创建 CodeBuild 服务角色 \(p. 357\)](#)。

批次的允许计算类型

选择批次允许的计算类型。选择所有适用项。

批次允许的最大构建数

输入批次允许的最大建立数。如果批次超出此限制，则批次将失败。

批处理超时

输入批次构建完成的最大时间量。

合并工件

选择 将所有图像从批次中结合到单个位置 将所有的图像从批次中的所有物料组合到一个位置。

在 Source (源) 下，您可以：

- 选择 Add source (添加源) 以添加辅助源。
- 选择 Remove source (删除源) 以删除辅助源。
- 使用 Source provider (源提供商) 和 Source version (源版本) 修改源的设置。

- 覆盖 Environment image (环境映像)、Operating system (操作系统)、Runtime (运行时) 和 Runtime version (运行时版本) 的设置。
- 选中或清除 Privileged (特权)。

Note

默认情况下，Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息，请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

- 在 Service role (服务角色) 中，您可以更改 CodeBuild 用来调用从属 AWS 服务的服务角色。选择 New service role (新建服务角色) 以让 CodeBuild 为您创建服务角色。
- 选择 Override build specification (覆盖生成规范) 来使用不同的生成规范。
- 更改 Timeout (超时) 的值。
- 更改 Compute (计算) 的值。
- 从 Certificate (证书) 中选择一种不同设置。

在 Buildspec (生成规范) 下，您可以：

- 选择 Use a buildspec file (使用构建规范文件) 以使用 buildspec.yml 文件。默认情况下，CodeBuild 在源代码根目录中查找名为 buildspec.yml 的文件。如果您的BuildSpec文件使用不同的名称或位置，请在“源根”中输入其路径 BuildSpec名称（例如，**buildspec-two.yml** 或 **configuration/buildspec.yml**...如果BuildSpec文件位于S3桶中，则必须在相同的 AWS 地区作为您的建立项目。按照构建规范文件的 ARN 指定该文件（例如，**arn:aws:s3:::my-codebuild-sample2/buildspec.yml**）。
- 选择 Insert build commands (插入生成命令) 以输入要在生成阶段运行的命令。

在 Build Artifacts (构建构件) 下，您可以：

- 从 Type (类型) 中，选择不同的构件类型。
- 在 Name (名称) 中，键入不同的输出构件名称。
- 如果希望构建规范文件中指定的名称覆盖控制台中指定的任何名称，请选择 Enable semantic versioning (启用语义版本控制)。构建规范文件中的名称使用 Shell 命令语言。例如，您可以将日期和时间附加到您的项目名称后面，以便它始终是唯一的。构件名称唯一防止覆盖构件。有关更多信息，请参阅[构建规范语法 \(p. 137\)](#)。)
- 在 Path (路径) 中，键入不同的输出构件路径。
- 在 Namespace type (命名空间类型) 中，选择不同的类型。选择 Build ID (构建 ID) 将构建 ID 插入构建输出文件的路径（例如，My-Path/Build-ID/My-Artifact.zip）。否则，选择 None (无)。
- 从 Bucket name (存储桶名称) 中，为您的输出构件选择其他 S3 存储桶。
- 如果不想加密构建构件，请选择 Disable artifacts encryption (禁用构件加密)。
- 选择 Artifacts packaging (构件打包)，然后选择 Zip 以将生成构件文件放入一个压缩文件中。要将构建构件文件分别放入指定的 S3 存储桶中（不压缩），请选择 None (无)。
- 在 Cache (缓存) 下，从 Type (类型) 中，选择不同的缓存设置。
- 仅覆盖此构建的辅助构件：
  - 要删除辅助构件，请在 Secondary artifacts (辅助构件) 中，选择辅助构件行中的 X。
  - 要添加辅助构件，请选择 Add artifact (添加构件)，然后输入辅助构件的信息。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)中的步骤 8。

在 Logs (日志) 下，您可以通过选中或清除 CloudWatch Logs 和 S3 logs (S3 日志) 来覆盖日志设置。

- 如果您启用 CloudWatch logs (CloudWatch 日志)：

- 在 Group name (组名称) 中，~~输入版本 2018-09-06~~ 输入您希望的 CloudWatch Logs 组的名称。

- 在 Stream name (流名称) 中，输入您的 Amazon CloudWatch Logs 流名称。
- 如果您启用 S3 logs (S3 日志)：
  - 从 Bucket (存储桶) 中，选择您的日志的 S3 存储桶的名称。
  - 在 Path prefix (路径前缀) 中，输入您的日志的前缀。

在 Service role (服务角色) 下，您可以更改 CodeBuild 用来调用从属 AWS 服务的服务角色。选择 Create a role (创建角色)，让 CodeBuild 为您创建一个服务角色。

5. 展开 Environment variables override (环境变量覆盖)。

环境变量列表预先填充在构建项目中设置的环境变量。如果要更改仅针对此构建的预填充环境变量的值，请更改值 值 和/或 类型。选择 Add environment variable (添加环境变量) 以仅为此构建添加新环境变量。

Note

TheTheBistro 删除 按钮不能用于删除预填充的环境变量。TheTheDelete 按钮仅用于删除此构建中添加或修改的环境变量。

其他人可以使用 CodeBuild 控制台和 AWS CLI 查看环境变量。如果您不担心环境变量的可见性，请设置 Name 和 Value 字段，然后将 Type 设置为 Plaintext。

我们建议您将具有敏感值 (例如 AWS 访问密钥 ID、AWS 秘密访问密钥或密码) 的环境变量作为参数存储在 Amazon EC2 Systems Manager Parameter Store 中。对于 Type (类型)，选择 Parameter (参数)。对于 Name (名称)，键入一个标识符以供 CodeBuild 引用。对于 Value (值)，按照 Amazon EC2 Systems Manager Parameter Store 中存储的参数名称输入参数名称。使用名为 /CodeBuild/dockerLoginPassword 的参数作为示例，对于 Type (类型)，选择 Parameter (参数)。对于 名称，输入 LOGIN\_PASSWORD...对于 值，输入 /CodeBuild/dockerLoginPassword.

我们建议您将名称以 /CodeBuild/ 开头的参数 (例如，/CodeBuild/dockerLoginPassword) 存储在 Amazon EC2 Systems Manager Parameter Store 中。可以使用 CodeBuild 控制台在 Amazon EC2 Systems Manager 中创建参数。选择 Create a parameter (创建参数)，然后按照说明操作。(在该对话框中，KMS 密钥，您可以选择指定 ARN 的 AWS KMS 关键字。Amazon EC2 Systems Manager 使用此密钥在检索期间对参数的值加密并解密。) 如果您使用 CodeBuild 控制台要创建参数，控制台启动参数 /CodeBuild/ 在存储的情况下。有关详细信息，请参阅 [系统管理器参数存储](#) 和 [巡视：创建和测试字符串参数（控制台）](#) 在 Amazon EC2 Systems Manager 用户指南。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的参数，则构建项目的 service role 必须允许 ssm:GetParameters 操作。如果您之前选择了 Create a service role in your account (在账户中创建服务角色)，则 CodeBuild 将自动在您的构建项目的默认 service role 中包含此操作。但是，如果您选择了 Choose an existing service role from your account，则必须将此操作单独包含在您的 service role 中。

如果您的构建项目是指存储在 Amazon EC2 Systems Manager 参数名称不会以 /CodeBuild/，您选择了 在您的帐户中创建服务角色，然后您必须更新该服务角色，以允许访问不会从 /CodeBuild/...这是因为服务角色仅允许访问以下的参数名称: /CodeBuild/.

您设置的任何环境变量都将替换现有的环境变量。例如，如果部署装置图像已经包含名称为“环境变量”的环境变量 MY\_VAR 有一个价值 my\_value，并且您设置了名为 MY\_VAR 有一个价值 other\_value，然后 my\_value 替换为 other\_value...同样，如果部署装置图像已经包含名称的环境变量 PATH 有一个价值 /usr/local/sbin:/usr/local/bin，并且您设置了名为 PATH 有一个价值 \$PATH:/usr/share/ant/bin，然后 /usr/local/sbin:/usr/local/bin 替换为字母值 \$PATH:/usr/share/ant/bin.

不要设置任何以开头的名称的环境变量 CODEBUILD\_...此前缀保留为内部使用。

如果具有相同名称的环境变量在多处都有定义，则应按照如下方式确定其值：

- 构建操作调用开始时的值优先级最高。
  - 构建项目定义中的值优先级次之。
  - 构建规范声明中的值优先级最低。
6. 选择 Start build。

有关此构建项目的详细信息，请参阅[查看构建详细信息（控制台）\(p. 262\)](#)。

## 运行构建 (AWS CLI)

### Note

要使用 CodePipeline 运行 AWS CodeBuild 中的构建项目，可跳过这些步骤并按照[创建使用 CodeBuild 的管道 \(AWS CLI\) \(p. 371\)](#)中的说明操作。

有关将 AWS CLI 与 CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

1. 使用以下方法之一运行 start-build 命令：

```
aws codebuild start-build --project-name <project-name>
```

如果您要运行的构建项目使用的是最新版本的构建输入项目和构建项目现有设置，请使用此方法。

```
aws codebuild start-build --generate-cli-skeleton
```

如果您要运行的构建具有早期版本的构建输入项目，或者如果您要覆盖构建输出项目、环境变量、构建规范或默认构建超时期限的设置，请使用此方法。

2. 如果您运行 start-build 命令与 --project-name 选项，更换 *<project-name>* 使用构建项目的名称，然后跳转至本程序的步骤6。要获取构建项目的列表，请参阅[查看构建项目名称的列表 \(\) \(p. 211\)](#)。
3. 如果您运行带 --idempotency-token 选项的 start-build 命令，则 start-build 请求将附带区分大小写的唯一标识符或令牌。令牌在发出请求后的 5 分钟内有效。如果您重复发出带相同令牌的 start-build 请求，但更改了参数，则 CodeBuild 会返回“参数不匹配”错误。
4. 如果您运行具有 --generate-cli-skeleton 选项的 start-build 命令，则采用 JSON 格式的数据将出现在输出中。将数据复制到本地计算机上或安装 AWS CLI 的实例上某位置处的文件（如 *start-build.json*）中。修改所复制的数据，使其符合以下格式，然后保存结果：

```
{
    "projectName": "projectName",
    "sourceVersion": "sourceVersion",
    "artifactsOverride": {
        "type": "type",
        "location": "location",
        "path": "path",
        "namespaceType": "namespaceType",
        "name": "artifactsOverride-name",
        "packaging": "packaging"
    },
    "buildspecOverride": "buildspecOverride",
    "cacheOverride": {
        "location": "cacheOverride-location",
        "type": "cacheOverride-type"
    },
    "certificateOverride": "certificateOverride",
    "computeTypeOverride": "computeTypeOverride",
    "environmentTypeOverride": "environmentTypeOverride",
    "environmentVariablesOverride": {
        "name": "environmentVariablesOverride-name",
        "value": "environmentVariablesOverride-value"
    }
}
```

```
        "value": "environmentVariablesValue",
        "type": "environmentVariablesOverride-type"
    },
    "gitCloneDepthOverride": "gitCloneDepthOverride",
    "imageOverride": "imageOverride",
    "idempotencyToken": "idempotencyToken",
    "insecureSslOverride": "insecureSslOverride",
    "privilegedModeOverride": "privilegedModeOverride",
    "queuedTimeoutInMinutesOverride": "queuedTimeoutInMinutesOverride",
    "reportBuildStatusOverride": "reportBuildStatusOverride",
    "timeoutInMinutesOverride": "timeoutInMinutesOverride",
    "sourceAuthOverride": "sourceAuthOverride",
    "sourceLocationOverride": "sourceLocationOverride",
    "serviceRoleOverride": "serviceRoleOverride",
    "sourceTypeOverride": "sourceTypeOverride"
}
```

替换以下占位符：

- *projectName* 必需的字符串。用于此构建项目的构建项目名称。
- *sourceVersion*: 可选字符串。要构建的源代码版本，如下所示：
  - 对于 Amazon S3，与您需要构建的输入 ZIP 文件的版本相对应的版本 ID。如果 *.sourceVersion* 未指定，然后使用最新版本。
  - 对于 CodeCommit，与您需要构建的源代码版本相对应的提交 ID。如果 *.sourceVersion* 未指定，使用默认分支机头的头部提交 ID。（您无法指定标签名称 *sourceVersion*，但您可以指定标签的提交 ID。）
  - 对于 GitHub，指定提交 ID、拉取请求 ID、分支名称或与您要构建的源代码版本相对应的标签名称。如果指定了拉取请求 ID，则它必须使用格式 pr/*pull-request-ID*（例如，pr/25）。如果指定了分支名称，则将使用分支的 HEAD 提交 ID。如果 *.sourceVersion* 未指定，使用默认分支机头的头部提交 ID。
  - 对于 Bitbucket，指定提交 ID、分支名称或与您要构建的源代码版本相对应的标签名称。如果指定了分支名称，则将使用分支的 HEAD 提交 ID。如果 *.sourceVersion* 未指定，使用默认分支机头的头部提交 ID。
- 以下占位符适用于 *artifactsOverride*。
  - *type*：可选。构建项目中定义覆盖此构建项目的构建输出项目类型。
  - *location*：可选。构建项目中定义覆盖此构建项目的构建输出项目位置。
  - *path*：可选。构建项目中定义覆盖此构建项目的构建输出项目路径。
  - *namespaceType*：可选。构建项目中定义覆盖此构建项目的构建输出项目路径类型。
  - *name*：可选。构建项目中定义覆盖此构建项目的构建输出项目名称。
  - *packaging*：可选。构建项目中定义覆盖此构建项目的构建输出项目打包类型。
- *buildspecOverride*：可选。构建项目中定义覆盖此构建项目的构建规范声明。如果设置了该值，则它可以是内联构建规范定义，也可以是指向相对于内置 CODEBUILD\_SRC\_DIR 环境变量的值的替代构建规范文件的路径，或者是指向 S3 存储桶的路径。S3 存储桶必须与构建项目位于同一 AWS 区域中。使用其 ARN 指定构建规范文件（例如，arn:aws:s3:::my-codebuild-sample2/buildspec.yml）。如果此值未提供或设置为空字符串，源代码必须在其根目录中包含 buildspec.yml 文件。有关更多信息，请参阅 [构建规范文件名称和存储位置 \(p. 136\)](#)。）
- 以下占位符适用于 *cacheOverride*。
  - *cacheOverride-location*：可选。A ProjectCache 这种构建对象可以覆盖 ProjectCache 构建项目中指定的对象。cacheOverride 是可选的，并且需要 ProjectCache 对象。location 对象需要 ProjectCache。
  - *cacheOverride-type*：可选。A 类的 ProjectCache 这种构建对象可以覆盖 ProjectCache 构建项目中指定的对象。cacheOverride 是可选的，并且需要 ProjectCache 对象。type 对象需要 ProjectCache。
- *certificateOverride*：可选。构建项目的证书的名称，该证书将覆盖构建项目中指定的证书。

- *environmentTypeOverride* : 可选。此构建的容器类型，该容器类型将覆盖构建项目中指定的容器类型。当前的有效字符串为 `LINUX_CONTAINER`。
- 以下占位符适用于 `environmentVariablesOverride`。
  - *environmentVariablesOverride-name* : 可选。构建项目中的环境变量名称，其值将会覆盖此构建项目中的相应值。
  - *environmentVariablesOverride-type* : 可选。构建项目中的环境变量类型，其值将会覆盖此构建项目中的相应值。
  - *environmentVariablesValue* : 可选。构建项目中定义的环境变量值，其值将会覆盖此构建项目中的相应值。
- *gitCloneDepthOverride* : 可选。构建项目中 Git clone depth 的值，您希望此构建项目会覆盖其值。如果您的源类型是 Amazon S3，则不支持此值。
- *imageOverride* : 可选。此构建的映像的名称，该映像将覆盖构建项目中指定的映像。
- *idempotencyToken* : 可选。一个字符串，该字符串用作令牌来指定构建请求是幂等的。您可以选择任何包含 64 个或更少字符的字符串。令牌在发出 start-build 请求后的 5 分钟内有效。如果您重复发出带相同令牌的 start-build 请求，但更改了参数，则 CodeBuild 会返回“参数不匹配”错误。
- *insecureSslOverride*: 可选布尔指定是否覆盖构建项目中指定的“不安全 TLS”设置。不安全的 TLS 设置确定是否忽略 TLS 警告，并连接到项目源代码。此覆盖仅在构建的源为 GitHub Enterprise Server 时适用。
- *privilegedModeOverride*: 可选布尔值。如果设置为 `true`，则构建将覆盖构建项目中的特权模式。
- *queuedTimeoutInMinutesOverride*: 可选整数，指定在超出一个构建之前可以排队的构建的分钟数。最小值为 5 分钟，最大值为 480 分钟（8 个小时）。
- *reportBuildStatusOverride*: 可选布尔值，指定是否发送源提供商的状态为构建的开始和完成。如果使用源提供商而非 GitHub、GitHub Enterprise Server 或 Bitbucket 设置此项，则会引发 `invalidInputException`。
- *sourceAuthOverride*: 可选字符串。此构建的授权类型，该授权类型将覆盖构建项目中定义的授权类型。此覆盖仅在构建项目的源为 Bitbucket 或 GitHub 时适用。
- *sourceLocationOverride*: 可选字符串。此构建的源位置，该源位置将覆盖构建项目中定义的源位置。
- *serviceRoleOverride*: 可选字符串。此构建的服务角色的名称，该角色将覆盖构建项目中指定的角色。
- *sourceTypeOverride*: 可选字符串。此构建的源输入类型，该源输入将覆盖构建项目中定义的源输入。有效字符串为 `NO_SOURCE`，`CODECOMMIT`，`CODEPIPELINE`，`GITHUB`，`S3`，`BITBUCKET`，和 `GITHUB_ENTERPRISE`。
- *timeoutInMinutesOverride*: 可选编号。构建项目中定义覆盖此构建项目的构建超时分钟数。

我们建议您将具有敏感值（例如 AWS 访问密钥 ID、AWS 秘密访问密钥或密码）的环境变量作为参数存储在 Amazon EC2 Systems Manager Parameter Store 中。如果 Amazon EC2 Systems Manager Parameter Store 中存储的参数的名称以 `/CodeBuild/` 开头（例如，`/CodeBuild/dockerLoginPassword`），则 CodeBuild 可以使用该参数。可以使用 CodeBuild 控制台在 Amazon EC2 Systems Manager 中创建参数。选择 `Create a parameter`（创建参数），然后按照说明操作。（在该对话框中，KMS 密钥，您可以选择指定 ARN 的 AWS KMS 关键字。Amazon EC2 Systems Manager 使用此密钥在检索期间对参数的值加密并解密。）如果您使用 CodeBuild 控制台要创建参数，控制台启动参数 `/CodeBuild/` 在存储的情况下。但是，如果您使用 Amazon EC2 Systems Manager Parameter Store 控制台创建参数，则必须使用以 `/CodeBuild/` 开头的参数名称，且必须将 Type（类型）设置为 Secure String（安全字符串）。有关详细信息，请参阅 [AWS Systems Manager 参数存储](#) 和 [巡视：创建和测试字符串参数（控制台）](#) 在 Amazon EC2 Systems Manager 用户指南。

如果您的构建项目引用了 Amazon EC2 Systems Manager Parameter Store 中存储的参数，则构建项目的服务角色必须允许 `ssm:GetParameters` 操作。如果您之前选择了 `Create a new service role in your account`（在账户中创建新服务角色），则 CodeBuild 将自动在您的构建项目的默认服务角色中包含此操作。但是，如果您选择了 `Choose an existing service role from your account`，则必须将此操作单独包含在您的服务角色中。

您设置的环境变量将替换现有的环境变量。例如，如果靠泊装置图像已经包含名称为“环境变量”的环境变量 MY\_VAR 有一个价值 my\_value，并且您设置了名为 MY\_VAR 有一个价值 other\_value，然后 my\_value 替换为 other\_value... 同样，如果靠泊装置图像已经包含名称的环境变量 PATH 有一个价值 /usr/local/sbin:/usr/local/bin，并且您设置了名为 PATH 有一个价值 \$PATH:/usr/share/ant/bin，然后 /usr/local/sbin:/usr/local/bin 替换为字母值 \$PATH:/usr/share/ant/bin。

不要设置任何以开头的名称的环境变量 CODEBUILD\_... 此前缀保留为内部使用。

如果具有相同名称的环境变量在多处都有定义，则将按照如下方式确定环境变量的值：

- 构建操作调用开始时的值优先级最高。
- 构建项目定义中的值优先级次之。
- 构建规范文件声明中的值优先级最低。

有关这些占位符的有效值的信息，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。有关构建项目的最新设置列表，请参阅[查看构建项目的详细信息 \(p. 213\)](#)。

5. 切换到包含您刚才保存的文件的目录，然后再次运行 start-build 命令。

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. 如果成功，与[运行构建 \(p. 21\)](#)过程中所述内容类似的数据将出现在输出中。

要了解有关此构建项目的详细信息，请记下输出中的 id 值，然后查看[查看构建详细信息 \(AWS CLI\) \(p. 262\)](#)。

## 运行批次构建 ( AWS CLI)

1. 使用以下方法之一运行 start-build-batch 命令：

```
aws codebuild start-build-batch --project-name <project-name>
```

如果您要运行的构建项目使用的是最新版本的构建输入项目和构建项目现有设置，请使用此方法。

```
aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
```

如果您要运行的构建具有早期版本的构建输入项目，或者如果您要覆盖构建输出项目、环境变量、构建规范或默认构建超时期限的设置，请使用此方法。

2. 如果您运行 start-build-batch 命令与 --project-name 选项，更换 <project-name> 使用构建项目的名称，然后跳转至本程序的步骤6。要获取构建项目的列表，请参阅[查看构建项目名称的列表 \(\) \(p. 211\)](#)。
3. 如果您运行带 --idempotency-token 选项的 start-build-batch 命令，则 start-build-batch 请求将附带区分大小写的唯一标识符或令牌。令牌在发出请求后的 5 分钟内有效。如果您重复发出带相同令牌的 start-build-batch 请求，但更改了参数，则 CodeBuild 会返回“参数不匹配”错误。
4. 如果您运行 start-build-batch 命令与 --generate-cli-skeleton 选项，JSON 格式化的数据将输出到 <json-file> --file 该文件与由 start-build 命令，添加以下对象。有关常用对象的详细信息，请参阅[运行构建 \(AWS CLI\) \(p. 256\)](#)。

修改此文件以添加任何构建覆盖，并保存您的结果。

```
"buildBatchConfigOverride": {  
    "combineArtifacts": "combineArtifacts",
```

```
"restrictions": {  
    "computeTypesAllowed": [  
        allowedComputeTypes  
    ],  
    "maximumBuildsAllowed": maximumBuildsAllowed  
},  
    "serviceRole": "batchServiceRole",  
    "timeoutInMins": batchTimeout  
}
```

The `buildBatchConfigOverride` 对象是 [ProjectBuildBatchconfig](#) 包含此构建的批次构建配置的结构。

#### *combineArtifacts*

布尔值，指定批次构建的构建伪影是否应合并到单个伪影位置。

#### *allowedComputeTypes*

一系列字符串，用于指定批次构建允许的计算类型。参见 [构建环境计算类型](#) 对于这些值。

#### *maximumBuildsAllowed*

指定允许的最大构建数。

#### *batchServiceRole*

指定批量构建项目的服务角色ARN。

#### *batchTimeout*

指定批次构建必须在中完成的最大时间量（以分钟为单位）。

5. 切换到包含您刚才保存的文件的目录，然后再次运行 `start-build` 命令。

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. 如果成功，JSON代表的 [建立批次](#) 对象出现在控制台输出中。查看 [StartBuildBatch响应语法](#) 对于此数据。

## 开始自动运行构建 (AWS CLI)

如果您的源代码存储在 GitHub 或 GitHub Enterprise Server 存储库中，则您可以使用 GitHub Webhook，让 AWS CodeBuild 在代码更改每次被推送到存储库时重建源代码。

运行 `create-webhook` 命令，如下所示：

```
aws codebuild create-webhook --project-name <project-name>
```

`<project-name>` 是包含要重建源代码的构建项目的名称。

对于 GitHub，与以下内容类似的信息将显示在输出中：

```
{  
    "webhook": {  
        "url": "<url>"  
    }  
}
```

`<url>` 是 GithubWebhook 的 URL。

对于 GitHub Enterprise Server，与以下内容类似的信息将显示在输出中：

```
{ "webhook": { "secret": "YRV4JAGFsekJIirp5ytx86o2pyhUdySNSDTLNjXoxX1c7zA6XYDF37-ZFyY02r+s4JSE70mLh3v-gh-ryoVB80SSC1aAtBtUpkhvYuncCndogCVfni70ukX2_xM-nlDMA5ngIg_Bi_M465yi33zyTUNPcq0lxCpL0-BwghcVa01AurnW77-Y7i..._XCFAhuMx1f4ub0gb8BSmMt2A16apqjQ0jekSbeDxWVyzI GuyFn411AxVfv9Nn76CaCndb3FVIE78fpvgFo41xxsQ6vp06LRTKTPzbyetThbVXGdaP1vnk8lnKmJDooeRTg1m2oYr17dwjzIQLcrvoCAlgY1500_7Lkfa-nMxFc_f15fryAqem843_d0OcdzybhncE810TruEUCFrax_AjCwmlLV0kgGG677925jbpz0fr1kh5pwI193_bb_jOH01nk6101Ppf2dIDATZgqMqg7eIib_axDeTaBhnp0I8J6gfT1uykOsaqgn151zC1PERUSMgJF+In_a-2_L_kylr-4hSxsas5Nu143_XORRNwTS1xqvh-A69hV07KhvT_KcSwkxShYCEmoaFfa72QsyY6800gWfNj31yfbjthORNl1cD06-37-McDLoyrtSE0V09nxvsG5zuINs-28rKjtig_M0fhwocfUutxBz7vr+cTduhLR1dxFLRusHuVOVnUDlw9hvHMr-hukeoF_1kDKyk4e420FvZXpjYw0rFV-dwxRkP_mlfzxJ1wyfmt21FlLkp_Y2j_4weAcKGeFr_i1NaYvszpzXj78Ae1adVolF48AdDh2pilsWj1atU9ztt942gljafFmKakcvuy5yXuHxxbblyC8NHY1ESUWPFc_fnqrMsR8op394AUChp1ZCYuiwl_cac-pIUB00Xaur_lu_fyFghgOjC7cfmA36rv5XSDnFMPBP3HnBeIaf9QZ6A1jegEWTHIKJ0N3AUdwpk_z_hwTxLy0aJ8JMzPTXkbBoT6525THbHsYyR"... "payloadUrl": "https://codebuild.us-east-2.amazonaws.com/webhooks?+t=eyJlbNyeKBZnREYRhijjoUmFqMmJERGRQbgshwLNTN13r8wGfjZOThWz1ZVG1NZ1pIR1e0RUsxdzGeLwhnVFFqWTR0kEfwt2dJRNmRhc353RNc0XMEncXFtakg1cE1nSy9zPSIsIml2UGFyyW1ldGvYU381Yy16TndSQ1orc2VPOjBCZzhPeVV1LC1tXR1cmhbC1GmX0k3D&v=1" } }
```

- 从输出中复制私有密钥和负载 URL。在 GitHub Enterprise Server 中添加 Webhook 时会用到它们。
- 在 GitHub Enterprise Server 中，选择存储您 CodeBuild 项目的存储库。选择 *Settings* (设置)，选择 *Hooks & services* (挂钩和服务)，然后选择 *Add webhook* (添加 webhook)。
- 输入负载 URL 和私有密钥，接受其他字段的默认值，然后选择 *Add webhook*。

## 停止自动运行构建 (AWS CLI)

如果您的源代码存储在 GitHub 或 GitHub Enterprise Server 存储库中，则您可以设置 GitHub Webhook，让 AWS CodeBuild 在代码更改每次被推送到存储库时重建源代码。有关更多信息，请参阅 [开始自动运行构建 \(AWS CLI\) \(p. 260\)](#)。)

如果您已启用了此行为，则可以通过运行 `delete-webhook` 命令将其关闭，如下所示：

```
aws codebuild delete-webhook --project-name <project-name>
```

- 其中：`<project-name>` 是包含要重建源代码的构建项目的名称。

如果此命令成功，则输出中不会出现任何信息和错误。

### Note

仅会从您的 CodeBuild 项目中删除 Webhook。您还应该从 GitHub 或 GitHub Enterprise Server 存储库中删除 Webhook。

## 运行构建 (AWS 开发工具包)

要使用 CodePipeline 运行 AWS CodeBuild 中的构建项目，可跳过这些步骤并按照[将 AWS CodePipeline 与 AWS CodeBuild 结合使用以测试代码和运行构建 \(p. 367\)](#)中的说明操作。

有关将 CodeBuild 与 AWS 开发工具包结合使用的信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 查看 AWS CodeBuild 中的构建详细信息

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包查看由 CodeBuild 管理的构建的详细信息。

### 主题

- [查看构建详细信息 \(控制台\) \(p. 262\)](#)
- [查看构建详细信息 \(AWS CLI\) \(p. 262\)](#)
- [查看构建详细信息 \(AWS 开发工具包\) \(p. 262\)](#)
- [构建阶段过渡 \(p. 262\)](#)

## 查看构建详细信息（控制台）

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 执行以下操作之一：
  - 在导航窗格中，选择 Build history。在生成列表中的 Build run (生成运行) 列，选择生成的链接。
  - 在导航窗格中，选择 Build projects。在生成项目列表中的 Name (名称) 列，选择生成项目名称的链接。然后，在生成列表中的 Build run (生成运行) 列，选择生成的链接。

### Note

默认情况下，仅显示最新的 10 个生成或生成项目。要查看更多生成或生成项目，请选择齿轮图标，然后为 Builds per page (每页生成数) 或 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头。

## 查看构建详细信息 (AWS CLI)

有关将 AWS CLI 与 AWS CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

运行 batch-get-builds 命令：

```
aws codebuild batch-get-builds --ids ids
```

替换以下占位符：

- *ids*：必填字符串。要查看其详细信息的一个或多个构建 ID。要指定多个构建 ID，请用空格分隔各个构建 ID。您最多可以指定 100 个构建 ID。要获取构建 ID 的列表，请参阅以下主题：
  - [查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)
  - [查看构建项目的构建 ID 列表 \(AWS CLI\) \(p. 266\)](#)

例如，如果您运行此命令：

```
aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2-ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-other-project:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE
```

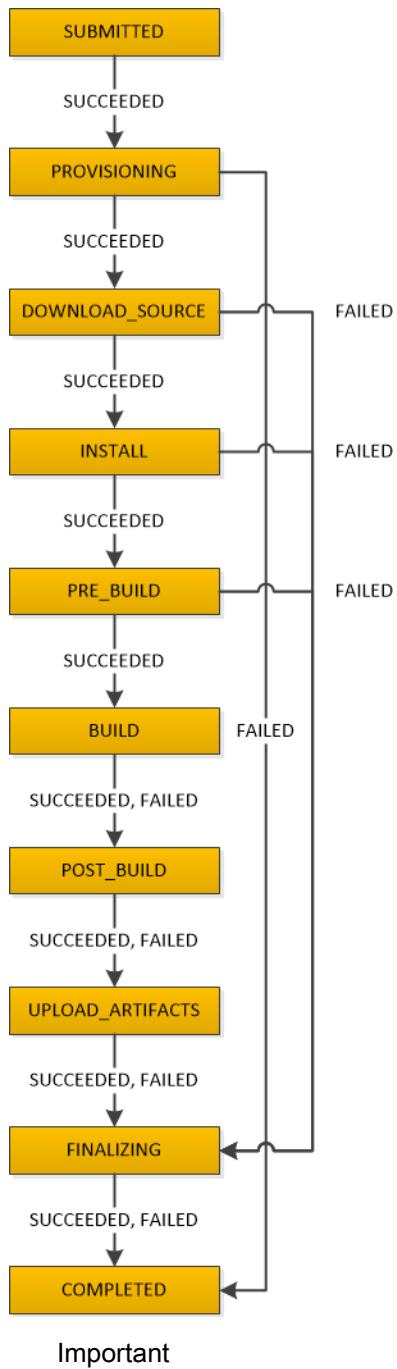
如果命令成功，与[查看汇总的构建信息 \(p. 22\)](#)中所述内容类似的数据将出现在输出中。

## 查看构建详细信息 (AWS 开发工具包)

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 构建阶段过渡

AWS CodeBuild 中的构建分阶段进行：



Important

系统始终尝试 UPLOAD\_ARTIFACTS 阶段，即使 BUILD 阶段失败。

## 查看 AWS CodeBuild 中的构建 ID 的列表

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包查看由 CodeBuild 管理的构建的构建 ID 列表。

### 主题

- [查看构建 ID 的列表（控制台）\(p. 264\)](#)

- [查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)
- [查看批次构建ID的列表 \( AWS CLI\) \(p. 265\)](#)
- [查看构建 ID 的列表 \( AWS 开发工具包 \) \(p. 266\)](#)

## 查看构建 ID 的列表 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build history。

### Note

默认情况下，仅显示 10 个最新生成。要查看更多生成，请选择齿轮图标，然后为 Builds per page (每页生成数) 选择不同值，或使用向后和向前箭头。

## 查看构建 ID 的列表 (AWS CLI)

有关将 AWS CLI 与 CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

- 运行 list-builds 命令。

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

替换上一命令中的以下占位符：

- **sort-order**: 用于指示如何列出构造ID的可选字符串。有效值包括 ASCENDING 和 DESCENDING。
- **next-token**: 可选字符串。在上次运行时，如果列表中有 100 个以上的项目，则只能返回前 100 个项目，以及名为下一个令牌的唯一字符串。要获取列表中的下一批项目，请再次运行此命令，将下一个令牌添加到调用中。要获取列表中的所有项目，请利用每个后续的下一个令牌运行此命令，直到不再返回下一个令牌。

例如，如果您运行此命令：

```
aws codebuild list-builds --sort-order ASCENDING
```

与以下内容类似的结果可能会出现在输出中：

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE",
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE",
    ...
    "... The full list of build IDs has been omitted for brevity ...",
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

如果您再次运行此命令：

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full token
has been omitted for brevity...MzY2OA==
```

与以下内容类似的结果可能会出现在输出中：

API 版本 2016-10-06

```
{  
    "ids": [  
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",  
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"  
    ]  
}
```

## 查看批次构建ID的列表 ( AWS CLI)

有关将 AWS CLI 与 CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

- 运行 list-build-batches 命令。

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

替换上一命令中的以下占位符：

- sort-order***: 用于指示如何列出批次构建ID的可选字符串。有效值包括 ASCENDING 和 DESCENDING。
- next-token***: 可选字符串。在上次运行时，如果列表中有 100 个以上的项目，则只能返回前 100 个项目，以及名为下一个令牌的唯一字符串。要获取列表中的下一批项目，请再次运行此命令，将下一个令牌添加到调用中。要获取列表中的所有项目，请利用每个后续的下一个令牌运行此命令，直到不再返回下一个令牌。

例如，如果您运行此命令：

```
aws codebuild list-build-batches --sort-order ASCENDING
```

与以下内容类似的结果可能会出现在输出中：

```
{  
    "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",  
    "ids": [  
        "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE",  
        "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE",  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"  
    ]  
}
```

如果您再次运行此命令：

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

与以下内容类似的结果可能会出现在输出中：

```
{  
    "ids": [  
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",  
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",  
        ... The full list of build IDs has been omitted for brevity ...
```

```
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"  
    ]  
}
```

## 查看构建 ID 的列表 ( AWS 开发工具包 )

有关将 CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 查看 AWS CodeBuild 中构建项目的构建 ID 列表

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包来查看 CodeBuild 中构建项目的构建 ID 列表。

### 主题

- [查看构建项目的构建 ID 列表 \( 控制台 \) \(p. 266\)](#)
- [查看构建项目的构建 ID 列表 \( AWS CLI \) \(p. 266\)](#)
- [查看构建项目的批次构建ID列表 \( AWS CLI \) \(p. 267\)](#)
- [查看构建项目的构建 ID 列表 \( AWS 开发工具包 \) \(p. 268\)](#)

## 查看构建项目的构建 ID 列表 ( 控制台 )

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在导航窗格中，选择 Build projects。在构建项目列表中的 Name (名称) 列中，选择构建项目。

### Note

默认情况下，仅显示最新的 100 个构建或构建项目。要查看更多生成或生成项目，请选择齿轮图标，然后为 Builds per page (每页生成数) 或 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头。

## 查看构建项目的构建 ID 列表 ( AWS CLI )

有关将 AWS CLI 与 AWS CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

运行 list-builds-for-project 命令，如下所示：

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order  
--next-token next-token
```

替换上一命令中的以下占位符：

- *project-name*: 用于指示为列表构建ID的构建项目名称的必填字符串。要获取构建项目的列表，请参阅[查看构建项目名称的列表 \(AWS CLI\) \(p. 212\)](#)。
- *sort-order*: 用于指示如何列出构造ID的可选字符串。有效值包括 ASCENDING 和 DESCENDING。
- *next-token*: 可选字符串。在上次运行时，如果列表中有 100 个以上的项目，则只能返回前 100 个项目，以及名为下一个令牌的唯一字符串。要获取列表中的下一批项目，请再次运行此命令，将下一个令牌添加到调用中。要获取列表中的所有项目，请借助随后返回的每个后续令牌不断运行此命令，直到不再返回后续令牌。

例如，如果您按照类似以下的方式运行此命令：

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order ASCENDING
```

输出中可能会显示类似于以下内容的结果：

```
{  
    "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",  
    "ids": [  
        "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"  
        "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"  
    ]  
}
```

如果您再次运行此命令：

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

您可能会看到类似于以下输出的结果：

```
{  
    "ids": [  
        "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"  
        "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"  
    ]  
}
```

## 查看构建项目的批次构建ID列表 ( AWS CLI)

有关将 AWS CLI 与 AWS CodeBuild 结合使用的更多信息，请参阅[命令行参考 \(p. 363\)](#)。

运行 `list-build-batches-for-project` 命令，如下所示：

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

替换上一命令中的以下占位符：

- *project-name*: 用于指示为列表构建ID的构建项目名称的必填字符串。要获取构建项目的列表，请参阅[查看构建项目名称的列表 \(AWS CLI\) \(p. 212\)](#)。
- *sort-order*: 用于指示如何列出构造ID的可选字符串。有效值包括 `ASCENDING` 和 `DESCENDING`。
- *next-token*: 可选字符串。在上次运行时，如果列表中有 100 个以上的项目，则只能返回前 100 个项目，以及名为下一个令牌的唯一字符串。要获取列表中的下一批项目，请再次运行此命令，将下一个令牌添加到调用中。要获取列表中的所有项目，请借助随后返回的每个后续令牌不断运行此命令，直到不再返回后续令牌。

例如，如果您按照类似以下的方式运行此命令：

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --sort-order ASCENDING
```

输出中可能会显示类似于以下内容的结果：

```
{  
    "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",  
    "ids": [  
        "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"  
        "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"  
    ]  
}
```

如果您再次运行此命令：

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project  
--sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for  
brevity...MzY2OA==
```

您可能会看到类似于以下输出的结果：

```
{  
    "ids": [  
        "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"  
        "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"  
        ... The full list of build IDs has been omitted for brevity ...  
        "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"  
    ]  
}
```

## 查看构建项目的构建 ID 列表 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 在 AWS CodeBuild 停止构建

您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包停止 AWS CodeBuild 中的构建。

### 主题

- [停止构建 \( 控制台 \) \(p. 268\)](#)
- [停止构建 \(AWS CLI\) \(p. 269\)](#)
- [停止构建 \( AWS 开发工具包 \) \(p. 269\)](#)

## 停止构建 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.

2. 执行以下操作之一：

- 如果显示了 **build-project-name:build-ID** 页面，请选择 Stop build (停止构建)。
- 在导航窗格中，选择 Build history。在生成列表中，选中生成的框，然后选择 Stop build (停止生成)。
- 在导航窗格中，选择 Build projects。在生成项目列表中的 Name (名称) 列，选择生成项目名称的链接。在生成列表中，选中生成的框，然后选择 Stop build (停止生成)。

### Note

默认情况下，仅显示最新的 100 个构建或构建项目。要查看更多生成或生成项目，请选择齿轮图标，然后为 Builds per page (每页生成数) 或 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头。

如果 AWS CodeBuild 无法成功停止构建（例如，构建过程已完成），则 Stop (停止) 按钮将禁用或者不显示。

## 停止构建 (AWS CLI)

- 运行 stop-build 命令：

```
aws codebuild stop-build --id id
```

在上述命令中，替换以下占位符：

- *id*：必填字符串。要停止的构建的 ID。要获取构建 ID 的列表，请参阅以下主题：

- [查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)
- [查看构建项目的构建 ID 列表 \(AWS CLI\) \(p. 266\)](#)

如果 AWS CodeBuild 成功停止构建，则输出中 build 对象的 buildStatus 值将为 STOPPED。

如果 CodeBuild 无法成功停止生成（例如，生成已完成），则输出中 build 对象的 buildStatus 值将为最终生成状态（例如，SUCCEEDED）。

## 停止构建 (AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 停止批次构建 AWS CodeBuild

您可以使用 AWS CodeBuild 控制台，AWS CLI，或 AWS 用于停止批次构建的 SDK AWS CodeBuild.

### 主题

- [停止批量构建 \( 控制台 \) \(p. 269\)](#)
- [停止批量构建 \( AWS CLI\) \(p. 270\)](#)
- [停止批量构建 \( AWS SDK \) \(p. 270\)](#)

## 停止批量构建 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 执行以下任一操作
  - 如果 ***build-project-name:build-ID*** 页面显示，选择 停止构建。
  - 在导航窗格中，选择 Build history。在生成列表中，选中生成的框，然后选择 Stop build (停止生成)。
  - 在导航窗格中，选择 Build projects。在生成项目列表中的 Name (名称) 列，选择生成项目名称的链接。在生成列表中，选中生成的框，然后选择 Stop build (停止生成)。

### Note

默认情况下，仅显示最新的 100 个构建或构建项目。要查看更多生成或生成项目，请选择齿轮图标，然后为 Builds per page (每页生成数) 或 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头。

IFIFIF AWS CodeBuild 无法成功停止批次构建（例如，如果构建过程已完成），停止构建按钮已禁用。

## 停止批量构建 ( AWS CLI)

- 运行 `stop-build-batch` 命令。

```
aws codebuild stop-build-batch --id <batch-build-id>
```

在上述命令中，替换以下占位符：

- `<batch-build-id>` 必需的字符串。要停止的批次构建的标识符。要获取批次构建标识符列表，请参阅以下主题：
  - [查看批次构建ID的列表 \( AWS CLI \) \(p. 265\)](#)
  - [查看构建项目的批次构建ID列表 \( AWS CLI \) \(p. 267\)](#)

IFIFIF AWS CodeBuild 成功停止批次构建，`buildBatchStatus` 在 `buildBatch` 输出中的对象是 `STOPPED`。

IFIFIF CodeBuild 无法成功停止批次构建（例如，如果批量构建已完成），`buildBatchStatus` 在 `buildBatch` 输出中的对象是最终构建状态（例如，`SUCCEEDED`）。

## 停止批量构建 ( AWS SDK )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 重试一个内部版本 AWS CodeBuild

您可以使用 AWS CodeBuild 控制台，AWS CLI，或 AWS SDKS 重试单个构建或批次构建 AWS CodeBuild.

### 主题

- [重试构建 \( 控制台 \) \(p. 270\)](#)
- [重试构建 \( AWS CLI \) \(p. 271\)](#)
- [重试构建 \( AWS SDK \) \(p. 271\)](#)

## 重试构建 ( 控制台 )

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 执行以下任一操作

- 如果 `build-project-name:build-ID` 页面显示，选择 重试构建。
- 在导航窗格中，选择 Build history。在构建列表中，选择构建框的框然后选择 重试构建。
- 在导航窗格中，选择 Build projects。在生成项目列表中的 Name (名称) 列，选择生成项目名称的链接。在构建列表中，选择构建框的框然后选择 重试构建。

#### Note

默认情况下，仅显示最新的 100 个构建或构建项目。要查看更多生成或生成项目，请选择齿轮图标，然后为 Builds per page (每页生成数) 或 Projects per page (每页项目数) 选择其他值，或者使用向后和向前箭头。

## 重试构建 ( AWS CLI)

- 运行 retry-build 命令。

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

在上述命令中，替换以下占位符：

- <build-id>必需的字符串。要重试的构建或批次构建的 ID。要获取构建 ID 的列表，请参阅以下主题：
  - [查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)
  - [查看批次构建ID的列表 \( AWS CLI\) \(p. 265\)](#)
  - [查看构建项目的构建 ID 列表 \(AWS CLI\) \(p. 266\)](#)
  - [查看构建项目的批次构建ID列表 \( AWS CLI\) \(p. 267\)](#)
- --idempotency-token: : 可选。如果您运行 retry-build 具有唯一大小写敏感标识符或令牌的命令包含在 retry-build 请求。令牌在发出 请求后的 5 分钟内有效。如果您重复发出带相同令牌的 retry-build 请求，但更改了参数，则 CodeBuild 会返回“参数不匹配”错误。

## 重试构建 ( AWS SDK )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的更多信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

## 在会话管理器中查看运行的构建

在 AWS CodeBuild，您可以暂停运行的构建并使用 AWS Systems Manager 会话管理器连接到构建容器并查看容器的状态。

#### 主题

- [Prerequisites \(p. 271\)](#)
- [暂停构建 \(p. 273\)](#)
- [开始构建 \(p. 273\)](#)
- [连接到构建容器 \(p. 273\)](#)
- [恢复构建 \(p. 274\)](#)

## Prerequisites

要允许会话管理器与构建会话配合使用，必须启用构建的会话连接。有两个前提条件：

- CodeBuild Linux 标准图像已经拥有 SSM 已安装代理并启用SSMAgent容器模式。

如果您正在使用自定义图像来进行构建，请执行以下操作：

1. 安装 SSM 代理 有关详细信息，请参阅 [在Linux2实例上手动安装SSMAgentforLinux](#) 在 AWS Systems Manager 用户指南。

2. 复制文件 <https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/4.0/amazon-ssm-agent.json> 到 /etc/amazon/ssm/ 图像中的目录。这可以在 SSM 代理。

- The CodeBuild 服务角色必须具有以下内容 SSM 政策:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "ssmmessages:CreateControlChannel",  
        "ssmmessages:CreateDataChannel",  
        "ssmmessages:OpenControlChannel",  
        "ssmmessages:OpenDataChannel"  
    ],  
    "Resource": "*"  
}
```

您可以将 CodeBuild 在启动构建时，控制台会自动将此策略附加到服务角色。或者，您可以手动将此策略附加到服务角色。

- 如果您有 审计和记录会话活动 已启用 Systems Manager 首选项，CodeBuild 服务角色还必须具有其他权限。权限不同，取决于存储日志的位置。

#### CloudWatch Logs

如果使用 CloudWatch Logs 要存储日志，请将以下权限添加到 CodeBuild 服务角色:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "logs:DescribeLogGroups",  
            "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:*:*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs>CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:<log-group-name>:/*"  
        }  
    ]  
}
```

#### Amazon S3

如果使用 Amazon S3 要存储日志，请将以下权限添加到 CodeBuild 服务角色:

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

```
        ]  
    }
```

有关详细信息，请参阅 [审计和记录会话活动](#) 在 AWS Systems Manager 用户指南.

## 暂停构建

要暂停构建，请插入 `codebuild-breakpoint` 在BuildSpec文件中的任何构建阶段中命令。此时将暂停构建内容，这样您便可以连接到构建容器并在当前状态下查看容器。

例如，将以下内容添加到BuildSpec文件中的构建阶段。

```
phases:  
  pre_build:  
    commands:  
      - echo Entered the pre_build phase...  
      - echo "Hello World" > /tmp/hello-world  
      - codebuild-breakpoint
```

本代码创建 `/tmp/hello-world` 然后暂停此时间点的构建。

## 开始构建

要允许会话管理器与构建会话配合使用，必须启用构建的会话连接。为此，在开始构建时，请按照以下步骤操作：

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Build projects。选择构建项目，然后选择 Start build。
3. 选择 Advanced build overrides (高级构建覆盖)。
4. 在 环境 部分，选择 启用会话连接 选项。如果未选择此选项，则所有 `codebuild-breakpoint` 和 `codebuild-resume` 命令被忽略。
5. 在 环境 部分，选择 允许AWS代码构建修改此服务角色，以便与此建立项目一起使用 允许 CodeBuild 控制台将会话管理器策略附加到服务角色。如果您已经向角色添加了会话管理器策略，则不需要选择此选项。
6. 做出任何其他所需的变更，并选择 开始构建。
7. 监控控制台中的“构造”状态。当会话可用时，AWS会话管理器 链接显示在 建立状态 第节。

## 连接到构建容器

您可以通过以下两种方式之一连接到构建容器：

CodeBuild 控制台：

在Web浏览器中，打开 AWS会话管理器 连接到构建容器的链接。将打开一个终端会话，允许您浏览和控制构建容器。

AWS CLI

Note

本地机器必须安装了此程序的SessionManager插件。有关详细信息，请参阅 [安装AWSCLI的SessionManager插件](#) 在 AWS Systems Manager 用户指南.

1. 呼叫 `batch-get-builds` API具有构建ID以获取有关构建的信息。

```
aws codebuild batch-get-builds --ids <buildID> --region <region>
```

2. 复制 sessionTarget 属性值。
3. 使用以下命令连接到构建容器。

```
aws ssm start-session --target <sessionTarget> --region <region>
```

在此示例中，请确认 /tmp/hello-world 文件存在并包含文本 Hello World.

## 恢复构建

在您完成检查构建容器后，请发出 codebuild-resume containershell 命令。

```
$ codebuild-resume
```

# 在 AWS CodeBuild 中删除构建

可以使用 AWS CLI 或 AWS 开发工具包删除 AWS CodeBuild 中的构建。

## 删除构建 (AWS CLI)

运行 batch-delete-builds 命令：

```
aws codebuild batch-delete-builds --ids <ids>
```

在上述命令中，替换以下占位符：

- **<ids>**：必填字符串。要删除的构建的 ID。要指定多个构建，请用空格将每个构建 ID 分隔开。要获取构建 ID 的列表，请参阅以下主题：
  - [查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)
  - [查看构建项目的构建 ID 列表 \(AWS CLI\) \(p. 266\)](#)

如果成功，buildsDeleted 数组将显示在输出中，其中包含已成功删除的每个构建的 Amazon 资源名称 (ARN)。有关未成功删除的构建的信息将显示在输出中的 buildsNotDeleted 数组中。

例如，如果您运行此命令：

```
aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645-b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX
```

与以下内容类似的信息将显示在输出中：

```
{  
    "buildsNotDeleted": [  
        {  
            "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-build-project:f8b888d2-5e1e-4032-8645-b115195648EX",  
            "statusCode": "BUILD_IN_PROGRESS"  
        }  
    ],  
    "buildsDeleted": [
```

```
    "arn:aws:codebuild:us-west-2:123456789012:build/my-other-demo-build-project:a18bc6ee-  
e499-4887-b36a-8c90349c7eEX"  
]  
}
```

## 删除构建 ( AWS 开发工具包 )

有关将 AWS CodeBuild 与 AWS 开发工具包结合使用的信息，请参阅[AWS 开发工具包和工具参考 \(p. 364\)](#)。

# 使用 AWS CodeBuild 中的测试报告

您可以在 CodeBuild 中创建测试报告，使其包含有关在构建期间运行的测试的详细信息。您可以创建诸如单元测试、配置测试和功能测试等测试。

支持以下测试报告文件格式：

- Cucumber JSON
- JUnit XML
- NUnit XML
- NUnit3XML(NUnit3XML)
- TestNG XML
- Visual Studio TRX

使用任何测试框架创建测试用例，这些测试框架可以采用任何一种格式创建报告文件（例如 Surefire JUnit 插件，TestNG 或 Cucumber）。

要创建测试报告，请将报告组名称添加到构建项目的 buildspec 文件中，该文件包含有关测试用例的信息。运行构建项目时，系统将运行测试用例并创建测试报告。您不需要在运行测试之前创建报告组。如果指定报告组名称，CodeBuild 会在您运行报告时为您创建报告组。如果要使用已存在的报告组，请在 buildspec 文件中指定其 ARN。

您可以使用测试报告帮助解决在构建运行期间发生的问题。如果您从构建项目的多个构建获得了许多测试报告，您可以使用测试报告查看趋势以及测试和失败率，以帮助您优化构建。

报告在创建后 30 天过期。您无法查看已过期的测试报告。如果您希望将测试报告保留 30 天以上，可以将测试结果的原始数据文件导出到 Amazon S3 存储桶。导出的测试文件不会过期。有关 S3 存储桶的信息在创建报告组时指定。

## Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

## 主题

- [创建测试报告 \(p. 276\)](#)
- [使用报告组 \(p. 277\)](#)
- [使用报告 \(p. 291\)](#)
- [使用测试报告权限 \(p. 291\)](#)
- [查看测试报告 \(p. 294\)](#)
- [使用测试框架测试报告 \(p. 295\)](#)
- [代码覆盖报告 \(p. 298\)](#)

## 创建测试报告

要创建测试报告，您运行的构建项目应在 buildspec 文件中配置有一到五个报告组。测试报告在运行期间创建。它包含为报告组指定的测试用例的结果。对于使用相同 buildspec 文件的每个后续构建，系统将生成一个新的测试报告。

## 创建测试报告

1. 创建构建项目 () 有关信息 , 请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。
2. 使用测试报告信息配置项目的 buildspec 文件 :
  - a. 添加 reports: 并指定报告组的名称。CodeBuild 使用项目名称和您以格式指定的名称为您创建报表组 project-name-report-group-name-in-buildspec。如果您已经有要使用的报告组, 请指定其RN。(如果您使用其名称而不是其ARN, CodeBuild 创建新的报告组。) 有关详细信息, 请参阅 [Reports syntax in the buildspec file](#).
  - b. 在报告组下 , 指定用于存储测试结果的文件的位置。如果您使用多个报告组 , 请为每个报告组指定测试结果文件位置。每次运行构建项目时都会创建一个新的测试报告。有关更多信息 , 请参阅 [指定测试文件 \(p. 282\)](#)。
  - c. 在 build 或 post\_build 序列的 commands 部分中 , 指定将运行您为报告组指定的测试用例的命令。有关更多信息 , 请参阅 [指定测试命令 \(p. 283\)](#)。
3. 运行构建项目中的构建。有关更多信息 , 请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#)。
4. 构建完成后 , 从项目页面上的 Build history (构建历史记录) 中选择新的构建运行。选择 Reports (报告) 以查看测试报告。有关更多信息 , 请参阅 [查看构建的测试报告 \(p. 294\)](#)。)

## 使用报告组

报告组 包含测试报告并指定共享设置。您可以使用 buildspec 文件指定要在构建时运行的测试用例以及运行它们的命令。对于在构建项目中配置的每个报告组 , 每次运行构建项目都会创建一个测试报告。多次运行配置有一个报告组的构建项目会在该报告组中创建多个测试报告 , 每个报告都包含为该报告组指定的相同测试用例的结果。

测试用例在构建项目的 buildspec 文件中针对报告组进行指定。您可以在一个构建项目中指定最多五个报告组。运行构建时 , 将运行所有测试用例。将创建一个新的测试报告 , 其中包含为报告组指定的每个测试用例的结果。每次运行新构建时 , 都会运行测试用例 , 并创建一个包含新测试结果的新测试报告。

报告组可用于多个构建项目。使用一个报告组创建的所有测试报告共享相同配置 , 如导出选项和权限 , 即使使用不同构建项目创建测试报告也是如此。在多个构建项目中使用一个报告组创建的测试报告可以包含运行不同测试用例集的结果 ( 每个构建项目对应一个测试用例组 ) 。这是因为您可以在每个项目的 buildspec 文件中为报告组指定不同的测试用例文件。您还可以通过编辑构建项目的 buildspec 文件来更改构建项目中的报告组的测试用例文件。后续运行构建会创建新的测试报告 , 其中包含更新的 buildspec 中的测试用例文件的结果。

### 主题

- [创建报告组 \(p. 277\)](#)
- [更新报告组 \(p. 280\)](#)
- [指定测试文件 \(p. 282\)](#)
- [指定测试命令 \(p. 283\)](#)
- [报告组命名 \(p. 283\)](#)
- [在 AWS CodeBuild 中标记报告组 \(p. 283\)](#)
- [使用共享报告组。 \(p. 287\)](#)

## 创建报告组

您可以使用 CodeBuild 控制台、AWS CLI 或构建规范文件创建报告组。您的 IAM 角色必须具有创建报告组所需的权限。有关更多信息 , 请参阅 [使用测试报告权限 \(p. 291\)](#)。)

### 主题

- [创建报告组（构建规范）\(p. 278\)](#)
- [创建报告组（CLI）\(p. 278\)](#)
- [创建报告组（控制台）\(p. 279\)](#)
- [创建报告组（AWS CloudFormation）\(p. 280\)](#)

## 创建报告组（构建规范）

使用 buildspec 创建的报告组不会导出原始测试结果文件。您可以查看报告组并指定导出设置。有关更多信息，请参阅 [更新报告组 \(p. 280\)](#)。)

### 使用 buildspec 文件创建报告组

1. 选择与 AWS 账户中的报告组没有关联的报告组名称。
2. 使用此名称配置 buildspec 文件的 `reports` 部分。在此示例中，报告组名称为 `new-report-group`，并使用 JUnit 框架创建使用测试用例：

```
reports:  
  new-report-group: #surefire junit reports  
    files:  
      - '**/*'  
    base-directory: 'surefire/target/surefire-reports'
```

有关更多信息，请参阅 [指定测试文件 \(p. 282\)](#) 和 [Reports syntax in the buildspec file](#)。

3. 在 `commands` 部分中，指定运行测试的命令。有关更多信息，请参阅 [指定测试命令 \(p. 283\)](#)。)
4. 运行构建。构建完成后，将创建一个新的报告组，其名称使用 `project-name-report-group-name`。有关详细信息，请参阅 [报告组命名 \(p. 283\)](#)。

## 创建报告组（CLI）

### 创建测试报告

1. 创建一个名为的文件 `CreateReportGroup.json`。
2. 根据您的要求，将以下 JSON 代码段之一复制到 `CreateReportGroup.json`：
  - 使用以下 JSON 指定测试报告组将原始测试结果文件导出到 Amazon S3 存储桶。

```
{  
  "name": "report-name",  
  "type": "TEST",  
  "exportConfig": {  
    "exportConfigType": "S3",  
    "s3Destination": {  
      "bucket": "bucket-name",  
      "path": "path",  
      "packaging": "NONE | ZIP",  
      "encryptionDisabled": "false",  
      "encryptionKey": "your-key"  
    },  
    "tags": [  
      {  
        "key": "tag-key",  
        "value": "tag-value"  
      }  
    ]  
  }  
}
```

}

将 `bucket-name` 替换为 S3 存储桶名称，并将 `path` 替换为要将文件导出到的 S3 存储桶中的路径。如果要压缩导出的文件，`packaging` 指定 `ZIP`。否则，请指定 `NONE`。使用 `encryptionDisabled` 指定是否加密导出的文件。如果要加密导出的文件，请输入客户主密钥 (CMK)。有关更多信息，请参阅 [更新报告组 \(p. 280\)](#)。)

- 使用以下 JSON 指定测试报告不会导出原始测试文件：

```
{  
    "name": "report-name",  
    "type": "TEST",  
    "exportConfig": {  
        "exportConfigType": "NO_EXPORT"  
    }  
}
```

#### Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

3. 运行以下命令：。

```
aws codebuild create-report-group \  
--cli-input-json file://CreateReportGroupInput.json \  
--region us-east-2
```

## 创建报告组（控制台）

### 创建测试报告

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 在导航窗格中，选择 Report groups (报告组)。
3. 选择 Create report group (创建报告组)。
4. 对于 Report group name (报告组名称)，输入报告组的名称。
5. (可选) 对于 Tags (标签)，输入您希望支持 AWS 服务使用的任何标签的名称和值。使用 Add row 添加标签。您最多可以添加 50 个标签。
6. 如果您想将测试报告结果的原始数据上传到 Amazon S3 存储桶：
  - a. 选择 Backup to Amazon S3 (备份到 Amazon S3)。
  - b. 对于 S3 bucket name (S3 存储桶名称)，请输入 S3 存储桶的名称。
  - c. 对于 Path prefix (路径前缀)，请输入要上传测试结果的 S3 存储桶中的路径。
  - d. 选择 Compress test result data in a zip file (将测试结果数据压缩为 zip 文件) 以便压缩原始测试结果数据文件。
  - e. 展开 Additional configuration (其他配置) 以显示加密选项。选择以下选项之一
    - Default AWS managed key (默认 AWS 托管密钥) 将使用由 AWS Key Management Service 托管的 Amazon S3 的客户主密钥 (CMK)。在 CodeBuild，默认CMK为 Amazon S3 并使用格式 `aws/s3`。有关详细信息，请参阅 [客户管理的CMK](#) 在 AWS Key Management Service 用户指南。这是默认加密选项。
    - Choose a custom key (选择自定义密钥) 将使用您创建和配置的 CMK。对于 AWS KMS encryption key (AWS KMS 加密密钥)，请输入加密密钥的 ARN。其格式为 `arn:aws:kms:region-id:aws-account-id:key/key-id`。有关详细信息，请参阅 [创建 KMS 键](#) 在 AWS Key Management Service 用户指南。

- Disable artifact encryption (禁用构件加密) 将禁用加密。如果要共享测试结果或将其发布到静态网站，则可以选择此选项。（动态网站可以运行代码来解密测试结果。）

有关静态数据加密的更多信息，请参阅[数据加密 \(p. 324\)](#)。

#### Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

7. 选择 Create report group (创建报告组)。

## 创建报告组 (AWS CloudFormation)

要使用 AWS CloudFormation 模板

您可以使用 AWS CloudFormation 模板文件创建和预配置报告组。有关更多信息，请参阅[AWS CloudFormation 用户指南](#)。

以下 AWS CloudFormation YAML 模板可创建一个不会导出原始测试结果文件的报告组。

```
Resources:  
  CodeBuildReportGroup:  
    Type: AWS::CodeBuild::ReportGroup  
    Properties:  
      Name: my-report-group-name  
      Type: TEST  
      ExportConfig:  
        ExportConfigType: NO_EXPORT
```

以下 AWS CloudFormation YAML 模板可创建一个将原始测试结果文件导出到 Amazon S3 存储桶的报告组。

```
Resources:  
  CodeBuildReportGroup:  
    Type: AWS::CodeBuild::ReportGroup  
    Properties:  
      Name: my-report-group-name  
      Type: TEST  
      ExportConfig:  
        ExportConfigType: S3  
        S3Destination:  
          Bucket: my-s3-bucket-name  
          Path: path-to-folder-for-exported-files  
          Packaging: ZIP  
          EncryptionKey: my-KMS-encryption-key  
          EncryptionDisabled: false
```

#### Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

## 更新报告组

当您更新报告组时，您可以指定是否将原始测试结果数据导出到 Amazon S3 存储桶中的文件的相关信息。如果您选择导出到 S3 存储桶，可以为报告组指定以下内容：

- 是否将原始测试结果文件压缩为 ZIP 文件。

- 是否对原始测试结果文件进行加密。您可以使用以下选项之一指定加密：
  - 由 AWS Key Management Service 托管的 Amazon S3 的客户主密钥 (CMK)。
  - 您创建和配置的 CMK。

有关更多信息，请参阅 [数据加密 \(p. 324\)](#)。)

如果您使用 AWS CLI 来更新报告组，则还可以更新或添加标签。有关更多信息，请参阅 [在 AWS CodeBuild 中标记报告组 \(p. 283\)](#)。)

#### Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

#### 主题

- [更新报告组（控制台）\(p. 281\)](#)
- [更新报告组（CLI）\(p. 281\)](#)

## 更新报告组（控制台）

#### 更新报告组的步骤

- Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
- 在导航窗格中，选择 Report groups (报告组)。
- 选择要更新的报告组。
- 选择 Edit (编辑)。
- 选择或清除 Backup to Amazon S3 (备份到 Amazon S3)。如果选择此选项，请指定您的导出设置：
  - 对于 S3 bucket name (S3 存储桶名称)，请输入 S3 存储桶的名称。
  - 对于 Path prefix (路径前缀)，请输入要上传测试结果的 S3 存储桶中的路径。
  - 选择 Compress test result data in a zip file (将测试结果数据压缩为 zip 文件) 以便压缩原始测试结果数据文件。
  - 展开 Additional configuration (其他配置) 以显示加密选项。选择以下选项之一
    - Default AWS managed key (默认 AWS 托管密钥) 将使用由 AWS Key Management Service 托管的 Amazon S3 的客户主密钥 (CMK)。进 CodeBuild，默认CMK为 Amazon S3 并使用格式 aws/S3。有关详细信息，请参阅 [客户管理的CMK 在 AWS Key Management Service 用户指南](#)。这是默认加密选项。
    - Choose a custom key (选择自定义密钥) 将使用您创建和配置的 CMK。对于 AWS KMS encryption key (AWS KMS 加密密钥)，请输入加密密钥的 ARN。其格式为 arn:aws:kms:*region-id*:*aws-account-id*:key/*key-id*。有关详细信息，请参阅 [创建 KMS 键 在 AWS Key Management Service 用户指南](#)。
    - Disable artifact encryption (禁用构件加密) 将禁用加密。如果要共享测试结果或将其实发布到静态网站，则可以选择此选项。（动态网站可以运行代码来解密测试结果。）

## 更新报告组（CLI）

#### 更新报告组的步骤

- 创建一个名为的文件 `UpdateReportGroupInput.json`。
- 将以下内容复制到 `UpdateReportGroupInput.json`。

```
{
```

```
"arn": "",  
"exportConfig": {  
    "exportConfigType": "S3",  
    "s3Destination": {  
        "bucket": "bucket-name",  
        "path": "path",  
        "packaging": "NONE | ZIP",  
        "encryptionDisabled": "false",  
        "encryptionKey": "your-key"  
    }  
},  
"tags": [  
    {  
        "key": "tag-key",  
        "value": "tag-value"  
    }  
]
```

3. 在 arn 行中输入报告组的 ARN，例如

```
"arn":"arn:aws:codebuild:region:123456789012:report-group/report-group-1")。
```

4. 使用要应用到报告组的更新来更新 UpdateReportGroupInput.json。

- 如果要更新报告组以将原始测试结果文件导出到 S3 存储桶，请更新 exportConfig 部分。将 bucket-name 替换为 S3 存储桶名称，并将 path 替换为 S3 存储桶中您要将文件导出到的路径。如果要压缩导出的文件，packaging 指定 ZIP。否则，请指定 NONE。使用 encryptionDisabled 指定是否加密导出的文件。如果要加密导出的文件，请输入客户主密钥 (CMK)。
- 如果要更新报告组，以使其不会将原始测试结果文件导出到 S3 存储桶，请使用以下 JSON 更新 exportConfig 部分：

```
{  
    "exportConfig": {  
        "exportConfigType": "NO_EXPORT"  
    }  
}
```

- 如果要更新报告组的标签，请更新 tags 部分。您可以更改、添加或删除标签。如果要删除所有标签，请使用以下 JSON 来更新：

```
"tags": []
```

5. 运行以下命令：。

```
aws codebuild update-report-group \  
--cli-input-json file://UpdateReportGroupInput.json
```

## 指定测试文件

您可以在构建项目 buildspec 文件的 reports 部分中为每个报告组指定测试结果文件及其位置。有关更多信息，请参阅 [Reports syntax in the buildspec file](#)。)

以下是 reports 部分的示例，该示例为构建项目指定了两个报告组。其中一个使用 ARN 指定，另一个使用名称指定。files 部分指定包含测试用例结果的文件。可选的 base-directory 部分指定测试用例文件所在的目录。可选的 discard-paths 部分指定是否丢弃将测试结果文件上传到 Amazon S3 存储桶的路径。

```
reports:  
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:  
  #surefire junit reports
```

```
files:
  - '**/*'
base-directory: 'surefire/target/surefire-reports'
discard-paths: false

sampleReportGroup: #Cucumber reports from json plugin
files:
  - 'cucumber-json/target/cucumber-json-report.json'
file-format: CUCUMBERJSON #Type of the report, defaults to JUNITXML
```

## 指定测试命令

您可以在 buildspec 文件的 commands 部分中指定运行测试用例的命令。这些命令运行您在 buildspec 文件的 reports 部分中为报告组指定的测试用例。以下是 commands 部分的示例，其中包含用于运行测试文件中的测试的命令：

```
commands:
  - echo Running tests for surefire junit
  - mvn test -f surefire/pom.xml -fn
  - echo
  - echo Running tests for cucumber with json plugin
  - mvn test -Dcucumber.options="--plugin json:target/cucumber-json-report.json" -f
cucumber-json/pom.xml -fn
```

## 报告组命名

使用 AWS CLI 或 AWS CodeBuild 控制台创建报告组时，您可以为报告组指定名称。如果您使用 buildspec 创建新报告组，则使用格式命名 *project-name-report-group-name-specified-in-buildspec*。通过运行该构建项目的构建创建的所有报表都属于具有新名称的新报表组。

如果您不希望 CodeBuild 创建新的报告组，请在构建项目的 buildspec 文件中指定报告组的 ARN。您可以在多个构建项目中指定一个报告组的 ARN。运行每个构建项目后，报告组包含由每个构建项目创建的测试报告。

例如，如果您创建一个名为 my-report-group 的报告组，然后在两个名为 my-project-1 和 my-project-2 的不同构建项目中使用该报告组名称，并创建两个项目的构建，则会创建两个新的报告组。结果将生成三个具有以下名称的报告组：

- my-report-group: : 没有任何测试报告。
- my-project-1-my-report-group: 包含由名为的构建项目运行的测试结果的报表 my-project-1。
- my-project-2-my-report-group: 包含由名为的构建项目运行的测试结果的报表 my-project-2.

如果您在两个项目中使用名为 my-report-group 的报告组的 ARN，然后运行每个项目的构建，则只会生成一个报告组 (my-report-group)。该报告组所含的测试报告包含由两个构建项目运行的测试的结果。

如果您选择的报告组名称不属于您的 AWS 账户中的报告组，然后在 buildspec 文件中将该名称用于报告组，并运行其构建项目的构建，则会创建一个新的报告组。新报告组的名称格式为 *project-name-new-group-name*。例如，如果您的报告中没有报告组 AWS 名为的帐户 new-report-group，并在名为的构建项目中指定它 test-project，内部版本运行会创建一个名为的新报告组 test-project-new-report-group。

## 在 AWS CodeBuild 中标记报告组

标签 是您或 AWS 分配给 AWS 资源的自定义属性标签。每个 AWS 标签具有两个部分：

- 标签键（例如，CostCenter、Environment、Project 或 Secret）。标签键区分大小写。
- 一个称为 标签值 的可选字段（例如，111122223333、Production 或团队名称）。省略标签值与使用空字符串相同。与标签键一样，标签值区分大小写。

这些被统称为键值对。有关报告组可拥有的标签数量限制以及标签键和值的限制，请参阅[Tags \(p. 401\)](#)。

标签有助于您标识和组织 AWS 资源。许多 AWS 服务支持标记，因此，您可以将同一标签分配给来不同服务的资源，以指示这些资源是相关的。例如，您可以将相同的标签分配给为 Amazon S3 存储桶分配的 CodeBuild 报告组。有关使用标签的更多信息，请参阅[标记最佳实践白皮书](#)。

在 CodeBuild 中，主要资源是报告组和项目。您可以使用 CodeBuild 控制台、AWS CLI、CodeBuild API 或 AWS 开发工具包为报告组添加、管理和移除标签。除了使用标签标识、组织和跟踪报告组以外，您还可以在 IAM 策略中使用标签以帮助控制哪些用户可以查看并与报告组交互。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

#### 主题

- [为报告组添加标签 \(p. 284\)](#)
- [查看报告组的标签 \(p. 285\)](#)
- [编辑报告组的标签 \(p. 286\)](#)
- [从报告组中移除标签 \(p. 286\)](#)

## 为报告组添加标签

为报告组添加标签可以帮助您标识和组织您的 AWS 资源并管理对其的访问。首先，为报告组添加一个或多个标签（键值对）。请记住，报告组可以拥有的标签数量有限制。键和值字段中可以使用的字符有限制。有关更多信息，请参阅[Tags \(p. 401\)](#)。有了标签后，您可以创建 IAM 策略以根据这些标签管理对报告组的访问。您可以使用 CodeBuild 控制台或 AWS CLI 为报告组添加标签。

#### Important

为报告组添加标签会影响对该报告组的访问。为报告组添加标签之前，请务必查看是否存在任何 IAM 策略可能使用标签来控制对资源（如报告组）的访问。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

有关在创建报告组时为其添加标签的更多信息，请参阅[创建报告组（控制台）\(p. 279\)](#)。

#### 主题

- [为报告组添加标签（控制台）\(p. 284\)](#)
- [为报告组添加标签（AWS CLI）\(p. 285\)](#)

## 为报告组添加标签（控制台）

您可以使用 CodeBuild 控制台为 CodeBuild 报告组添加一个或多个标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Report groups (报告组) 中，选择要在其中添加标签的报告组的名称。
3. 在导航窗格中，选择 Settings。
4. 如果此报告组中尚未添加标签，请选择 Add tag (添加标签)。您也可以选择 Edit (编辑)，然后选择 Add tag (添加标签)。
5. 在 Key (键) 中，输入标签的名称。您可以在 Value (值) 中添加可选的标签值。
6. (可选) 要添加其他标签，请再次选择 Add tag (添加标签)。

7. 添加完标签后，选择 Submit (提交)。

## 为报告组添加标签 (AWS CLI)

要在创建报告组时为其添加标签，请参阅[创建报告组 \(CLI\) \(p. 278\)](#)。在 `CreateReportGroup.json` 中，添加您的标签。

要向现有报告组添加标签，请参阅[更新报告组 \(CLI\) \(p. 281\)](#)并在 `UpdateReportGroupInput.json` 中添加标签。

在这些步骤中，我们假设您已安装最新版本的 AWS CLI 或已更新到当前版本。有关更多信息，请参阅[安装 AWS Command Line Interface](#)。

## 查看报告组的标签

标签可以帮助您标识和组织您的 AWS 资源并管理对它的访问。有关使用标签的更多信息，请参阅[标记最佳实践白皮书](#)。有关基于标签的访问策略示例，请参阅[Deny or allow actions on report groups based on resource tags](#)。

### 查看报告组的标签（控制台）

您可以使用 CodeBuild 控制台查看与 CodeBuild 报告组关联的标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Report groups (报告组) 中，选择要在其中查看标签的报告组的名称。
3. 在导航窗格中，选择 Settings。

### 查看报告组的标签 (AWS CLI)

按照以下步骤使用 AWS CLI 来查看报告组的 AWS 标签。如果尚未添加标签，则返回的标签列表为空。

1. 使用控制台或 AWS CLI 找到报告组的 ARN。记下它。

#### AWS CLI

运行以下命令。

```
aws list-report-groups
```

此命令返回类似下面的 JSON 格式信息：

```
{  
    "reportGroups": [  
        "arn:aws:codebuild:region:123456789012:report-group/report-group-1",  
        "arn:aws:codebuild:region:123456789012:report-group/report-group-2",  
        "arn:aws:codebuild:region:123456789012:report-group/report-group-3"  
    ]  
}
```

报告组 ARN 以其名称结尾，您可以使用该名称来识别您的报告组的 ARN。

#### Console

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Report groups (报告组) 中，选择带有您要查看的标签的报告组的名称。

3. 在 Configuration (配置) 中，找到您的报告组的 ARN。
2. 运行以下命令。为 --report-group-arns 参数使用您记下的 ARN。

```
aws codebuild batch-get-report-groups --report-group-arns  
arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

如果成功，此命令会返回 JSON 格式的信息，其中包含类似于下面的 tags 部分：

```
{  
  ...  
  "tags": {  
    "Status": "Secret",  
    "Project": "TestBuild"  
  }  
  ...  
}
```

## 编辑报告组的标签

您可以更改与报告组关联的标签值。您也可以更改标签键的名称，这相当于删除当前的标签并使用新名称和相同的值添加一个不同的标签。请记住，键和值字段中可以使用的字符有限制。有关更多信息，请参阅 [Tags \(p. 401\)](#)。

### Important

编辑报告组的标签会影响对该报告组的访问。编辑报告组的标签名称（键）或值之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如报告组）的访问。有关基于标签的访问策略示例，请参阅 [Deny or allow actions on report groups based on resource tags](#)。

## 编辑报告组的标签（控制台）

您可以使用 CodeBuild 控制台编辑与 CodeBuild 报告组关联的标签。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Report groups (报告组) 中，选择要在其中编辑标签的报告组的名称。
3. 在导航窗格中，选择 Settings。
4. 选择 Edit。
5. 请执行下列操作之一：
  - 要更改标签，则在 Key (键) 中输入新名称。更改标签的名称相当于删除标签并使用新的键名添加新标签。
  - 要更改标签的值，则输入新值。如果您想将标签值清空，请删除当前的值并将字段保留为空白。
6. 编辑完标签后，选择 Submit (提交)。

## 编辑报告组的标签 (AWS CLI)

要添加、更改或移除报告组中的标签，请参阅 [更新报告组 \(CLI\) \(p. 281\)](#)。更新 `UpdateReportGroupInput.json` 中的标签。

## 从报告组中移除标签

您可以移除与报告组关联的一个或多个标签。删除标签不会从与该标签关联的其他 AWS 资源中删除该标签。

### Important

删除报告组的标签会影响对该报告组的访问。从报告组中移除标签之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如报告组）的访问。有关基于标签的访问策略示例，请参阅[使用标签控制对 AWS CodeBuild 资源的访问 \(p. 348\)](#)。

## 从报告组中删除标签（控制台）

您可以使用 CodeBuild 控制台移除标签和 CodeBuild 报告组之间的关联。

1. 通过以下网址打开 CodeBuild 控制台：<https://console.aws.amazon.com/codebuild/>。
2. 在 Report groups (报告组) 中，选择要从其中移除标签的报告组的名称。
3. 在导航窗格中，选择 Settings。
4. 选择 Edit。
5. 找到要移除的标签，然后选择 Remove tag (移除标签)。
6. 移除标签之后，选择 Submit (提交)。

## 从报告组中移除标签（AWS CLI）

可以按照以下步骤使用 AWS CLI 从 CodeBuild 报告组中移除标签。移除标签并不会删除标签，而只是删除它和报告组之间的关联。

### Note

如果删除 CodeBuild 报告组，则会从删除的报告组中删除所有标签关联。您无需在删除报告组之前移除标签。

要从报告组中删除一个或多个标签，请参阅[编辑报告组的标签 \(AWS CLI\) \(p. 286\)](#)。使用不包含待删除标签的更新标签列表来更新采用 JSON 格式数据的 tags 部分。如果要删除所有标签，请将 tags 部分更新为：

```
"tags: []"
```

## 使用共享报告组。

报告组共享允许多个 AWS 账户或用户查看报告组、其未过期报告及其报告的测试结果。在此模型中，拥有报告组的账户（拥有者）将与其他账户（使用者）共享该报告组。使用者无法编辑报告组。报告在创建后 30 天过期。

内容：

- [共享报告组的先决条件 \(p. 287\)](#)
- [访问与您共享的报告组的先决条件 \(p. 288\)](#)
- [相关服务 \(p. 288\)](#)
- [共享报告组 \(p. 288\)](#)
- [取消共享已共享的报告组 \(p. 289\)](#)
- [标识共享报告组 \(p. 290\)](#)
- [共享报告组权限 \(p. 291\)](#)

## 共享报告组的先决条件

要共享报告组，您的 AWS 账户必须拥有该报告组。无法共享已与您共享的报告组。

## 访问与您共享的报告组的先决条件

要访问共享报告组，使用者的 IAM 角色需要 `BatchGetReportGroups` 权限。您可以将以下策略附加到其 IAM 角色：

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "*"  
    ],  
    "Action": [  
        "codebuild:BatchGetReportGroups"  
    ]  
}
```

有关更多信息，请参阅 [为 AWS CodeBuild 使用基于身份的策略 \(p. 329\)](#)。)

## 相关服务

报告组共享与 AWS Resource Access Manager (AWS RAM) 集成，后者是一项服务，使您可以与任何 AWS 账户或通过 AWS Organizations 共享 AWS 资源。通过使用 AWS RAM，您可以通过创建资源共享来共享您拥有的资源，该共享指定要共享的资源和要与其共享资源的使用者。使用者可以是单个 AWS 账户、AWS Organizations 中的组织部门或 AWS Organizations 中的整个组织。

有关更多信息，请参阅 [AWS RAM 用户指南](#)。

## 共享报告组

共享报告组时，将向使用者授予对报告组及其报告的只读访问权限。使用者可以使用 AWS CLI 查看报告组、报告，以及每个报告的测试用例结果。使用者不能执行以下操作：

- 在 CodeBuild 控制台中查看共享报告组或其报告。
- 编辑共享报告组。
- 使用项目中的共享报告组的 ARN 运行报告。指定共享报告组的项目构建将失败。

您可以使用 CodeBuild 控制台将报告组添加到现有资源共享。如果要将报告组添加到新的资源共享，则必须首先在 [AWS RAM 控制台](#) 中创建资源共享。

要与组织单位或整个组织共享报告组，您必须启用与 AWS Organizations 的共享。有关详细信息，请参阅 [启用与 AWS Organizations](#) 在 AWS RAM 用户指南。

您可以使用 CodeBuild 控制台、AWS RAM 控制台或 AWS CLI 共享您拥有的报告组。

### 共享您拥有的报告组（CodeBuild 控制台）

- Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
- 在导航窗格中，选择 Report groups (报告组)。
- 选择要共享的项目，然后选择 Share (共享)。有关详细信息，请参阅 [创建资源共享](#) 在 AWS RAM 用户指南。

### 共享您拥有的报告组（AWS RAM 控制台）

参见 [创建资源共享](#) 在 AWS RAM 用户指南。

### 共享您拥有的报告组 ( AWS RAM 命令 )

使用 [create-resource-share](#) 命令。

### 分享您拥有的报告组 ( CodeBuild 命令 )

使用 [put-resource-policy](#) 命令：

1. 创建一个名为 `policy.json` 的文件，并将以下内容复制到该文件中。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "consumer-aws-account-id-or-user"  
            },  
            "Action": [  
                "codebuild:BatchGetReportGroups",  
                "codebuild:BatchGetReports",  
                "codebuild>ListReportsForReportGroup",  
                "codebuild:DescribeTestCases"],  
            "Resource": "arn-of-report-group-to-share"  
        }]  
}
```

2. 使用报告组 ARN 和标识符更新 `policy.json`，以便共享该报告组。以下示例向 Alice 授予具有 ARN `arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group` 的报告组的只读访问权限，并向 123456789012 标识的 AWS 账户授予根用户身份。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": [  
                    "arn:aws:iam:123456789012:user/Alice",  
                    "123456789012"  
                ]  
            },  
            "Action": [  
                "codebuild:BatchGetReportGroups",  
                "codebuild:BatchGetReports",  
                "codebuild>ListReportsForReportGroup",  
                "codebuild:DescribeTestCases"],  
            "Resource": "arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group"  
        }]  
}
```

3. 运行以下命令：。

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://  
policy.json
```

## 取消共享已共享的报告组

取消共享的报告组（包括其报告及其测试用例结果）只能由其拥有者访问。如果取消共享报告组，先前与其共享该报告组的任何 AWS 账户或用户都无法访问该报告组、报告或报告中的测试用例结果。

要取消共享您拥有的已共享报告组，必须从资源共享中将其删除。您可以使用 AWS RAM 控制台或 AWS CLI 执行此操作。

取消共享您拥有的共享报告组 ( AWS RAM 控制台 )

参见 [更新资源共享](#) 在 AWS RAM 用户指南.

取消共享您拥有的共享报告组 ( AWS RAM 命令 )

使用 [disassociate-resource-share](#) 命令。

取消您拥有的报告组的分享报告 CodeBuild 命令 )

运行 [delete-resource-policy](#) 命令，并指定要取消共享的报告组的 ARN :

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

## 标识共享报告组

拥有者和使用者可以使用 AWS CLI 标识共享报告组。

要标识和获取有关共享报告组及其报告的信息，请使用以下命令：

- 要查看与您共享的报告组的 ARN，请运行 [list-shared-report-groups](#) :

```
aws codebuild list-shared-report-groups
```

- 要查看报告组中的报告的 ARN，请使用报告组 ARN 运行 [list-reports-for-report-group](#) :

```
aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
```

- 要查看有关报告中的测试用例的信息，请使用报告 ARN 运行 [describe-test-cases](#) :

```
aws codebuild describe-test-cases --report-arn report-arn
```

输出如下所示：

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "report-arn",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "path-to-output-report-files"
    },
    {
      "status": "SUCCEEDED",
      "name": "Test case 2",
      "expired": 1575916770.0,
      "reportArn": "report-arn",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "path-to-output-report-files"
    }
  ]
}
```

## 共享报告组权限

### 拥有者的权限

报告组拥有者可以编辑报告组，并在项目中指定该报告组以运行报告。

### 使用者的权限

报告组使用者可以查看报告组、报告以及报告的测试用例结果。使用者无法编辑报告组或其报告，也无法使用报告组创建报告。

## 使用报告

报告包含为一个报告组指定的测试用例的结果。测试报告是在构建项目运行期间创建的。您可以在 buildspec 文件中指定报告组、测试用例文件和用于运行测试用例的命令。每次运行测试用例时，都会在报告组中创建一个新的测试报告。

测试报告在创建后 30 天过期。您无法查看过期的测试报告，但您可以将测试结果导出到 S3 存储桶中的原始测试结果文件。导出的原始测试文件不会过期。有关更多信息，请参阅 [更新报告组 \(p. 280\)](#)。)

测试报告可能处于以下状态之一：

- **GENERATING:**：测试用例仍在运行中。
- **DELETING:**：正在删除测试报告。删除测试报告时，还将删除其测试用例。不会删除导出到 S3 存储桶的原始测试结果数据文件。
- **INCOMPLETE:**：测试报告未完成。由于以下原因之一，可能会返回此状态：
  - 指定此报告的测试用例的报告组配置有问题。例如，buildspec 文件中报告组下的测试用例路径可能不正确。
  - 运行构建的 IAM 用户没有运行测试的权限。有关更多信息，请参阅 [使用测试报告权限 \(p. 291\)](#)。)
  - 由于发生与测试无关的错误，构建未完成。
- **SUCCEEDED:**：所有测试用例都成功。
- **FAILED:**：部分测试用例未成功。

每个测试用例都会返回一个状态。测试用例可能处于以下状态之一：

- **SUCCEEDED:**：测试用例通过。
- **FAILED:**：测试用例失败。
- **ERROR:**：测试用例导致意外错误。
- **SKIPPED:**：测试用例未运行。
- **UNKNOWN:** 测试用例返回的状态不是 SUCCEEDED, FAILED, ERROR, 或 SKIPPED.

测试报告最多可包含 500 个测试用例结果。如果运行的测试用例超过 500 个，CodeBuild 会优先列出状态为 FAILED 的测试，并截断测试用例结果。

## 使用测试报告权限

本主题介绍与测试报告相关权限有关的重要信息。

主题

- 为测试报告创建角色 (p. 292)
- 测试报告操作的权限 (p. 293)
- 测试报告权限示例 (p. 293)

## 为测试报告创建角色

要运行测试报告，并更新项目以包含测试报告，您的 IAM 角色需要以下权限。这些权限包含在预定义的 AWS 托管策略中。如果要将测试报告添加到现有构建项目，则必须自行添加这些权限。

- `CreateReportGroup`
- `CreateReport`
- `UpdateReport`
- `BatchPutTestCases`

要运行代码覆盖报告，IAM 角色还必须包括 `BatchPutCodeCoverages` 权限。

### Note

`BatchPutTestCases`, `CreateReport`, `UpdateReport`、和 `BatchPutCodeCoverages` 不是公共权限。您不能调用相应的 AWS CLI 命令或开发工具包方法来获得这些权限。

要确保您拥有这些权限，您可以将以下策略附加到您的 IAM 角色：

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "*"  
    ],  
    "Action": [  
        "codebuild:CreateReportGroup",  
        "codebuild:CreateReport",  
        "codebuild:UpdateReport",  
        "codebuild:BatchPutTestCases",  
        "codebuild:BatchPutCodeCoverages"  
    ]  
}
```

我们建议您将此策略仅限定用于您必须使用的报告组。以下内容将权限仅限定用于策略中具有两个 ARN 的报告组：

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1",  
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-2"  
    ],  
    "Action": [  
        "codebuild:CreateReportGroup",  
        "codebuild:CreateReport",  
        "codebuild:UpdateReport",  
        "codebuild:BatchPutTestCases",  
        "codebuild:BatchPutCodeCoverages"  
    ]  
}
```

以下内容将权限仅限定用于通过运行名为 my-project 的项目构建而创建的报告组：

```
{  
    "Effect": "Allow",  
    "Resource": [  
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/my-project-*"  
    ],  
    "Action": [  
        "codebuild:CreateReportGroup",  
        "codebuild:CreateReport",  
        "codebuild:UpdateReport",  
        "codebuild:BatchPutTestCases",  
        "codebuild:BatchPutCodeCoverages"  
    ]  
}
```

#### Note

项目中指定的 CodeBuild 服务角色用于获得上传到 S3 存储桶的权限。

## 测试报告操作的权限

您可以为以下测试报告 CodeBuild API 操作指定权限：

- BatchGetReportGroups
- BatchGetReports
- CreateReportGroup
- DeleteReportGroup
- DeleteReport
- DescribeTestCases
- ListReportGroups
- ListReports
- ListReportsForReportGroup
- UpdateReportGroup

有关更多信息，请参阅 [AWS CodeBuild 权限参考 \(p. 344\)](#)。)

## 测试报告权限示例

有关测试报告相关示例策略的信息，请参阅以下内容：

- 允许用户更改报告组 ([p. 340](#))
- 允许用户创建报告组 ([p. 338](#))
- 允许用户删除报告 ([p. 339](#))
- 允许用户删除报告组 ([p. 339](#))
- 允许用户获取有关报告组的信息 ([p. 338](#))
- 允许用户获取有关报告的信息 ([p. 338](#))
- 允许用户获取报告组列表 ([p. 341](#))
- 允许用户获取报告列表 ([p. 341](#))
- 允许用户获取报告组的报告列表 ([p. 342](#))

- 允许用户获取报告的测试用例的列表 (p. 342)

## 查看测试报告

您可以查看有关测试报告的详细信息，例如，有关其测试用例、通过和失败次数及其运行时间的信息。您可以按构建运行、报告组或您的 AWS 账户分组查看测试报告。在控制台中选择测试报告以查看其详细信息和测试用例的结果。

您可以查看未过期的测试报告。测试报告在创建后 30 天过期。您无法在 CodeBuild 中查看过期报告。

### 主题

- [查看构建的测试报告 \(p. 294\)](#)
- [查看报告组的测试报告 \(p. 294\)](#)
- [查看您的 AWS 账户中的测试报告 \(p. 294\)](#)

## 查看构建的测试报告

### 查看构建的测试报告

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 找到要查看的构建。如果您知道运行构建（创建测试报告）的项目，请执行以下操作：
  1. 在导航窗格中，选择 Build projects (构建项目)，然后选择项目，该项目具有运行要查看的测试报告的构建。
  2. 选择 Build history (构建历史记录)，然后选择构建，该构建运行创建的要查看的报告。

您还可以在您的 AWS 账户的构建历史记录中查找构建：

1. 在导航窗格中，选择 Build history (构建历史记录)，然后选择构建，该构建已创建要查看的报告。
3. 在构建页面中，选择 Reports (报告)，然后选择测试报告以查看其详细信息。

## 查看报告组的测试报告

### 查看报告组中的测试报告

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Report groups (报告组)。
3. 选择包含要查看的测试报告的报告组。
4. 选择测试报告以查看其详细信息。

## 查看您的 AWS 账户中的测试报告

### 查看您的 AWS 账户中的测试报告

1. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. 在导航窗格中，选择 Report history (报告历史记录)。
3. 选择测试报告以查看其详细信息。

# 使用测试框架测试报告

本节中的主题演示如何在 AWS CodeBuild 中为各种测试框架设置测试报告。

## 主题

- [使用 Jasmine 设置测试报告 \(p. 295\)](#)
- [使用 Jest 设置测试报告 \(p. 296\)](#)
- [使用 pytest 设置测试报告 \(p. 297\)](#)
- [使用 RSpec 设置测试报告 \(p. 298\)](#)

## 使用 Jasmine 设置测试报告

以下过程演示如何在 AWS CodeBuild 中使用 [JasmineBDD 测试框架](#) 来设置测试报告。

该过程需要以下先决条件：

- 您有一个现有的 AWS CodeBuild 项目。
- 您的项目是一个 Node.js 项目，此项目设置为使用 Jasmine 测试框架。

将 [jasmine-reporters](#) 程序包添加到项目 package.json 文件的 devDependencies 部分。此程序包具有一系列可以与 Jasmine 一起使用 JavaScript 报告程序类。

```
npm install --save-dev jasmine-reporters
```

如果它尚未存在，请将 test 脚本添加到项目的 package.json 文件中。test 脚本确保在执行 npm test 时调用 Jasmine。

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

AWS CodeBuild 支持以下 Jasmine 测试报告程序：

### JUnitXmlReporter

用于以 JunitXml 格式生成报告。

### NUnitXmlReporter

用于以 NunitXml 格式生成报告。

默认情况下，具有 Jasmine 的 Node.js 项目将有一个 spec 子目录，其中包含 Jasmine 配置和测试脚本。

要将 Jasmine 配置为以 Junitxml 格式生成报告，请通过将以下代码添加到测试中来实例化 JUnitXmlReporter 报告程序。

```
var reporters = require('jasmine-reporters');

var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
```

```
    consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

要将 Jasmine 配置为以 NunitXML 格式生成报告，请通过将以下代码添加到测试中来实例化 NUnitXmlReporter 报告程序。

```
var reporters = require('jasmine-reporters');

var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

测试报告被导出至指定文件 <test report directory>/<report filename>.

在您的 buildspec.yml 文件中，添加/更新以下部分。

```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jasmine_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

如果您使用的是 NunitXml 报告格式，请将 file-format 值更改为以下值。

```
file-format: NUNITXML
```

## 使用 Jest 设置测试报告

以下过程演示如何在 AWS CodeBuild 中使用 [Jest 测试框架](#) 来设置测试报告。

该过程需要以下先决条件：

- 您有一个现有的 AWS CodeBuild 项目。
- 您的项目是一个 Node.js 项目，此项目设置为使用 Jest 测试框架。

将 [jest-junit](#) 程序包添加到项目 package.json 文件的 devDependencies 部分。AWS CodeBuild 使用此程序包生成 JunitXml 格式的报告。

```
npm install --save-dev jest-junit
```

如果它尚未存在，请将 test 脚本添加到项目的 package.json 文件中。test 脚本确保在执行 npm test 时调用 Jest。

```
{  
  "scripts": {  
    "test": "jest"  
  }  
}
```

通过将以下内容添加到 Jest 配置文件中，将 Jest 配置为使用 JunitXml 报告程序。如果您的项目没有 Jest 配置文件，请在项目的根目录中创建一个名为 jest.config.js 的文件，然后添加以下内容。测试报告被导出至指定文件 <test report directory>/<report filename>。

```
module.exports = {  
  reporters: [  
    'default',  
    [ 'jest-junit', {  
      outputDirectory: <test report directory>,  
      outputName: <report filename>,  
    } ]  
  ]  
};
```

在您的 buildspec.yml 文件中，添加/更新以下部分。

```
version: 0.2  
  
phases:  
  pre_build:  
    commands:  
      - npm install  
  build:  
    commands:  
      - npm build  
      - npm test  
  
reports:  
  jest_reports:  
    files:  
      - <report filename>  
    file-format: JUNITXML  
    base-directory: <test report directory>
```

## 使用 pytest 设置测试报告

以下过程演示如何在 AWS CodeBuild 中使用 [pytest 测试框架](#) 来设置测试报告。

该过程需要以下先决条件：

- 您有一个现有的 AWS CodeBuild 项目。
- 您的项目是一个 Python 项目，此项目设置为使用 pytest 测试框架。

将以下条目添加到 buildspec.yml 文件的 build 或 post\_build 阶段。此代码会自动发现当前目录中的测试，并将测试报告导出到 <test report directory>/<report filename> 报告使用 JunitXml 格式。

```
- python -m pytest --junitxml=<test report directory>/<report filename>
```

在您的 buildspec.yml 文件中，添加/更新以下部分。

```
version: 0.2

phases:
  install:
    runtime-versions:
      python: 3.7
    commands:
      - pip3 install pytest
  build:
    commands:
      - python -m pytest --junitxml=<test report directory>/<report filename>

reports:
  pytest_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

## 使用 RSpec 设置测试报告

以下过程演示如何在 AWS CodeBuild 中使用 [RSpec 测试框架](#) 来设置测试报告。

该过程需要以下先决条件：

- 您有一个现有的 AWS CodeBuild 项目。
- 您的项目是一个 Ruby 项目，此项目设置为使用 RSpec 测试框架。

在 buildspec.yml 文件中添加/更新以下内容。此代码在中运行测试 `<test source directory>` 目录，并将测试报告导出到指定文件，`<test report directory>/<report filename>` 报告使用 JunitXml 格式。

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  pre_build:
    commands:
      - gem install rspec
      - gem install rspec_junit_formatter
  build:
    commands:
      - rspec <test source directory>/* --format RspecJunitFormatter --out <test report directory>/<report filename>
reports:
  rspec_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

## 代码覆盖报告

CodeBuild 允许您为测试生成代码覆盖报告。提供了以下代码覆盖报告：

## 线路覆盖

线路覆盖范围衡量测试涵盖的语句数量。语句是单一指令,不包括注释或条件。

```
line coverage = (total lines covered)/(total number of lines)
```

## 分行覆盖

分支覆盖范围测量测试所涵盖的控制结构(例如 if 或 case 声明)。

```
branch coverage = (total branches covered)/(total number of branches)
```

支持以下代码覆盖报告文件格式:

- JaCoCoXML(JaCoCoXML)
- SimpleCovJSON(简单CovJSON)<sup>1</sup>
- 三叶草XML
- CoberturaXML(CoberturaXML)

<sup>1</sup>

# 创建代码覆盖报告

要创建代码覆盖报告,请运行在其buildspec文件中配置至少一个代码覆盖报告组的构建项目。 AWS CodeBuild 将解释代码覆盖结果并提供运行的代码覆盖报告。对于使用相同 buildspec 文件的每个后续构建 , 系统将生成一个新的测试报告。

## 创建测试报告

1. 创建构建项目 () 有关信息 , 请参阅[在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。
2. 使用测试报告信息配置项目的buildspec文件:
  - a. 添加 reports: 并指定报告组的名称。 AWS CodeBuild 使用项目名称和您以格式指定的名称为您创建报表组 project-name-report-group-name-in-buildspec。如果您已经有要使用的报告组,请指定其RN。如果您使用名称代替RN, AWS CodeBuild 创建新的报告组。有关更多信息 , 请参阅[Reports syntax in the buildspec file](#)。 )
  - b. 在报告组下,指定包含代码覆盖结果的文件的位置。如果您使用多个报告组,请为每个报告组指定结果文件位置。每次构建项目运行时都会创建新的代码覆盖报告。有关更多信息 , 请参阅[指定测试文件 \(p. 282\)](#)。 )

这是一个示例,为位于test-results/jacoco-coverage-report.xml.

```
reports:
  jacoco-report:
    files:
      - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

- c. 在 commands 部分 build 或 post\_build 指定运行代码覆盖分析的命令。有关更多信息 , 请参阅[指定测试命令 \(p. 283\)](#)。 )
3. 运行构建项目中的构建。有关更多信息 , 请参阅[在 AWS CodeBuild 中运行构建 \(p. 252\)](#)。 )
4. 构建完成后 , 从项目页面上的 Build history (构建历史记录) 中选择新的构建运行。选择 报告 查看代码覆盖报告。有关更多信息 , 请参阅[查看构建的测试报告 \(p. 294\)](#)。 )

# AWS CodeBuild 中的日志记录和监控

监控是保持 AWS CodeBuild 和您的 AWS 解决方案的可靠性、可用性和性能的重要方面。您应该从 AWS 解决方案的各个部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS 提供了以下工具来监控您的 CodeBuild 资源和构建并对潜在事件做出响应。

## 主题

- [使用 AWS CloudTrail 记录 AWS CodeBuild API 调用 \(p. 300\)](#)
- [监控 AWS CodeBuild \(p. 302\)](#)

## 使用 AWS CloudTrail 记录 AWS CodeBuild API 调用

AWS CodeBuild 与 AWS CloudTrail 集成，后者是在 CodeBuild 中提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将对 CodeBuild 的所有 API 调用作为事件捕获，包括来自 CodeBuild 控制台的调用和对 CodeBuild API 的代码调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 S3 存储桶（包括 CodeBuild 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台的 Event history (事件历史记录) 中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 CodeBuild 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 [AWS CloudTrail User Guide](#)。

## CloudTrail 中的 AWS CodeBuild 信息

在您创建 CloudTrail 账户时，即针对该账户启用了 AWS。CodeBuild 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 AWS CloudTrail User Guide 中的 [使用 CloudTrail 事件历史记录查看事件](#)。

要持续记录 AWS 账户中的事件（包括 CodeBuild 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 S3 存储桶。您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取措施。有关更多信息，请参阅：

- [创建跟踪概述](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [从多个区域接收 CloudTrail 日志文件](#) 和 [从多个账户中接收 CloudTrail 日志文件](#)

所有 CodeBuild 操作由 CloudTrail 记录，并且在 [CodeBuild API 参考](#) 中正式记载。例如，对 CreateProject（在 AWS CLI 中为 create-project）、StartBuild（在 AWS CLI 中为 start-project）和 UpdateProject（在 AWS CLI 中为 update-project）操作的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 AWS CloudTrail User Guide 中的 [CloudTrail userIdentity 元素](#)。

## 了解 AWS CodeBuild 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传输到您指定的 S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

### Note

为了保护敏感信息，CodeBuild 日志中隐藏以下内容：

- AWS 访问密钥 ID。有关更多信息，请参阅 AWS Identity and Access Management 用户指南中的 [管理 IAM 用户的访问密钥](#)。
- 使用参数存储指定的字符串。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store 控制台演练](#)。
- 使用 AWS Secrets Manager 指定的字符串。有关更多信息，请参阅 [密钥管理 \(p. 324\)](#)。

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了如何在 CodeBuild 中创建构建项目。

```
{  
    "eventVersion": "1.05",  
    "userIdentity": {  
        "type": "FederatedUser",  
        "principalId": "account-ID:user-name",  
        "arn": "arn:aws:sts::account-ID:federated-user/user-name",  
        "accountId": "account-ID",  
        "accessKeyId": "access-key-ID",  
        "sessionContext": {  
            "attributes": {  
                "mfaAuthenticated": "false",  
                "creationDate": "2016-09-06T17:59:10Z"  
            },  
            "sessionIssuer": {  
                "type": "IAMUser",  
                "principalId": "access-key-ID",  
                "arn": "arn:aws:iam::account-ID:user/user-name",  
                "accountId": "account-ID",  
                "userName": "user-name"  
            }  
        }  
    },  
    "eventTime": "2016-09-06T17:59:11Z",  
    "eventSource": "codebuild.amazonaws.com",  
    "eventName": "CreateProject",  
    "awsRegion": "region-ID",  
    "sourceIPAddress": "127.0.0.1",  
    "userAgent": "user-agent",  
    "requestParameters": {  
        "awsActId": "account-ID"  
    },  
    "responseElements": {  
        "project": {  
            "environment": {  
                "name": "environment-name",  
                "type": "environment-type",  
                "variables": {  
                    "variable1": "variable-value-1",  
                    "variable2": "variable-value-2",  
                    "variable3": "variable-value-3"  
                }  
            }  
        }  
    }  
}
```

```
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
    },
    "name": "codebuild-demo-project",
    "description": "This is my demo project",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project:project-ID",
    "encryptionKey": "arn:aws:kms:region-ID:key-ID",
    "timeoutInMinutes": 10,
    "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket",
        "type": "S3",
        "packaging": "ZIP",
        "outputName": "MyOutputArtifact.zip"
    },
    "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
    "lastModified": "Sep 6, 2016 10:59:11 AM",
    "source": {
        "type": "GITHUB",
        "location": "https://github.com/my-repo.git"
    },
    "created": "Sep 6, 2016 10:59:11 AM"
}
},
"requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
"eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "account-ID"
}
```

## 监控AWS CodeBuild

可以使用 Amazon CloudWatch 监控构建，在出现问题时报告以及视情况执行自动操作。可以监控两个级别的构建：

### 项目级别

这些度量标准适用于指定项目中的所有内容。要查看项目的指标，请为 CloudWatch 中的维度指定 **ProjectName**。

### AWS 账户级别

这些度量标准的内容是在帐户中构建的。要查看 AWS 账户级别的指标，请勿在 CloudWatch 中输入维度。创建资源利用率度量标准不可用 AWS 账户级别。

CloudWatch 指标显示构建随着时间推移的行为。例如，可以监控：

- 构建项目或 AWS 账户中随着时间推移尝试过多少次构建。
- 构建项目或 AWS 账户中随着时间推移成功过多少次构建。
- 构建项目或 AWS 账户中随着时间推移失败过多少次构建。
- CodeBuild 在构建项目或 AWS 账户中随着时间推移花费过多少时间执行构建。
- 构建构建或整个构建项目的资源利用率。构建资源利用率度量标准包括CPU、内存和存储利用率等度量。

有关更多信息，请参阅 [监控 CodeBuild 度量 \(p. 306\)](#)。)

## CodeBuild CloudWatch 度量

可以根据以下标准跟踪以下度量: AWS 帐户或建立项目。

### 建立持续时间

测量构建的 BUILD 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

### 构建

测量所触发构建的数量。

单位 Count

有效 CloudWatch 统计数据: 总计

### 下载源成熟

测量构建的 DOWNLOAD\_SOURCE 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

### 持续时间

测量随着时间的推移所有构建的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

### FailedBuilds

测量由于客户端错误或超时而失败的构建数。

单位 Count

有效 CloudWatch 统计数据: 总计

### 最终持续时间

测量构建的 FINALIZING 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

### 安装持续时间

测量构建的 INSTALL 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

### 建立后持续时间

测量构建的 POST\_BUILD 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值  
预构建持续时间

测量构建的 PRE\_BUILD 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值  
预备持续时间

测量构建的 PROVISIONING 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值  
QueuedDuration ( 队列持续时 )

测量构建的 QUEUED 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值  
提交持续时间

测量构建的 SUBMITTED 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值  
SucceededBuilds

测量成功构建的数量。

单位 Count

有效 CloudWatch 统计数据: 总计  
上传理论成熟

测量构建的 UPLOAD\_ARTIFACTS 阶段的持续时间。

单位 Seconds

有效 CloudWatch 统计数据: 平均值 ( 建议值 ) 、最大值、最小值

## CodeBuild CloudWatch 资源利用率指标

可追踪以下资源利用率度量。

### Note

CodeBuild 资源利用率指标仅在以下地区提供:

- 亚太区域 ( 东京 )
- 亚太区域 ( 首尔 )

- 亚太地区（孟买）区域
- 亚太区域（新加坡）
- 亚太区域（悉尼）
- 加拿大（中部）区域
- 欧洲（法兰克福）区域
- 欧洲（爱尔兰）区域
- 欧洲（伦敦）区域
- 欧洲（巴黎）区域
- 南美洲（圣保罗）区域
- 美国东部（弗吉尼亚北部）地区
- 美国东部（俄亥俄）区域
- 美国西部（加利福利亚北部）区域
- 美国西部（俄勒冈）区域

#### CPU使用

构建容器使用的已分配处理的CPU单位数。

单位 CPU单位

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

#### CPU使用率%

构建容器使用的分配处理的百分比。

单位 百分比

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

#### 已使用纪念品

构建容器使用的内存数MB。

单位 兆字节

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

#### 记忆利用率

构建容器使用的已分配内存的百分比。

单位 百分比

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

#### 存储器字节

构建容器使用的存储读取速度。

单位 字节/秒

有效 CloudWatch 统计数据: 平均值 (建议值)、最大值、最小值

#### 存储写字节

构建容器使用的存储写入速度。

单位 字节/秒

有效 CloudWatch 统计数据：平均值（建议值）、最大值、最小值

## CodeBuild CloudWatch 维度

CodeBuild 提供以下内容 CloudWatch 度量标准。如果没有指定这些标准，则指标将用于当前的度量 AWS 账户。

BuildId、BuildNumber、ProjectName

为构建标识符、构建编号和项目名称提供度量。

ProjectName

为项目名称提供度量。

## CodeBuild CloudWatch 警报

您可以使用 CloudWatch 控制台基于 CodeBuild 指标创建警报，以便可在构建出错时做出反应。两个最有用的警报指标如下：

- FailedBuild...当在预定的秒数内检测到某个故障构建数量时，可以创建触发的警报。在 CloudWatch，指定秒数以及构建失败的数量触发警报。
- Duration...您可以创建一个在构建时间比预期时间更长的警报。指定在启动构建之后、完成构建之前必须经历多少秒才会触发警报。

有关如何为 CodeBuild 指标创建警报的信息，请参阅 [使用 CloudWatch 警报监控构建 \(p. 316\)](#)。有关警报的更多信息，请参阅 [创建 Amazon CloudWatch 警报](#) 在 Amazon CloudWatch 用户指南。

## 监控 CodeBuild 度量

AWS CodeBuild 将代表您监控函数并通过 Amazon CloudWatch 报告各项指标。上述指标包括构建总数量、失败构建数量、成功构建数量和构建的持续时间。

可以使用 CodeBuild 控制台或 CloudWatch 控制台监控 CodeBuild 的指标。以下过程演示如何访问指标。

主题

- [访问构建指标 \(CodeBuild 控制台\) \(p. 306\)](#)
- [访问构建指标 \(Amazon CloudWatch 控制台\) \(p. 307\)](#)

## 访问构建指标 (CodeBuild 控制台)

Note

您无法自定义用于在该度量标准中显示的度量或图表 CodeBuild 控制台。如果您想自定义显示器，请使用 Amazon CloudWatch 控制台以查看您的构建度量。

### 客户级别度量

访问 AWS 账户级别指标

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CodeBuild 控制台：<https://console.aws.amazon.com/codesuite/codebuild/home>。

2. 在导航窗格中，选择 Account metrics (账户指标)。

## 项目级别度量

### 访问项目级别指标

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CodeBuild 控制台：<https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 在导航窗格中，选择 Build projects。
3. 在生成项目列表中的 Name (名称) 列中，选择要查看其指标的项目。
4. 选择 Metrics (指标) 选项卡。

## 访问构建指标 (Amazon CloudWatch 控制台)

您可以自定义用于显示这些度量标准的度量和图表 CloudWatch 控制台。

## 客户级别度量

### 访问账户级别指标

1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在全部指标选项卡上，选择 CodeBuild。

Metrics

Favorites

 Add a dashboard

4. 选择 Account Metrics (账户指标)。
5. 选择一个或多个项目和指标。对于每个项目，可选择 SucceededBuilds、FailedBuilds、Builds 和 Duration 指标。此页面上的图表中将显示所有选定项目和指标组合。

## 项目级别度量

### 访问项目级别指标

1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在全部指标选项卡上，选择 CodeBuild。

Metrics

Favorites

 Add a dashboard

4. 选择按项目。
5. 选择一个或多个项目和指标组合。对于每个项目，可选择 SucceededBuilds、FailedBuilds、Builds 和 Duration 指标。此页面上的图表中将显示所有选定项目和指标组合。
6. ( 可选 ) 可以自定义指标和图表。例如，从 统计数据 列，您可以选择要显示的不同统计信息。或从 Period (周期) 列的下拉菜单中，可以选择要用于监控指标的各个时间段。

有关详细信息，请参阅 [图表度量](#) 和 [查看可用度量](#) 在 Amazon CloudWatch 用户指南.

## 监控 CodeBuild 资源利用率指标

AWS CodeBuild 监控代表您构建资源利用率，并通过以下方式报告度量标准 Amazon CloudWatch. 这些度量包括CPU、内存和存储利用率等度量。

您可以使用 CodeBuild 控制台或 CloudWatch 控制台以监控资源利用率度量标准 CodeBuild. 以下步骤向您展示如何访问资源利用率度量标准。

### Note

CodeBuild 资源利用率指标仅在以下地区提供:

- 亚太区域 ( 东京 )
- 亚太区域 ( 首尔 )
- 亚太地区 ( 孟买 ) 区域
- 亚太区域 ( 新加坡 )
- 亚太区域 ( 悉尼 )
- 加拿大 ( 中部 ) 区域
- 欧洲 ( 法兰克福 ) 区域
- 欧洲 ( 爱尔兰 ) 区域
- 欧洲 ( 伦敦 ) 区域
- 欧洲 ( 巴黎 ) 区域
- 南美洲 ( 圣保罗 ) 区域
- 美国东部 ( 弗吉尼亚北部 ) 地区
- 美国东部 ( 俄亥俄 ) 区域
- 美国西部 ( 加利福利亚北部 ) 区域
- 美国西部 ( 俄勒冈 ) 区域

### 主题

- [访问资源利用率度量 \( CodeBuild 控制台 \) \(p. 311\)](#)
- [访问资源利用率度量 \( Amazon CloudWatch 控制台 \) \(p. 312\)](#)

## 访问资源利用率度量 ( CodeBuild 控制台 )

### Note

您无法自定义用于在该度量标准中显示的度量或图表 CodeBuild 控制台。如果您想自定义显示器，请使用 Amazon CloudWatch 控制台以查看您的构建度量。

## 项目级别资源利用率指标

### 如何访问项目级别资源利用率度量

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CodeBuild 控制台：<https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 在导航窗格中，选择 Build projects。
3. 在构建项目列表中，名称列，选择要查看利用率度量标准的项目。
4. 选择 Metrics (指标) 选项卡。资源利用率度量标准显示在 资源利用率指标 第节。
5. 要查看项目级别资源利用率度量标准 CloudWatch 控制台，选择 在CloudWatch中查看 在 资源利用率指标 第节。

## 构建级别资源利用率度量

### 如何访问构建级别资源利用率度量

1. 登录 AWS 管理控制台并通过以下网址打开 AWS CodeBuild 控制台：<https://console.aws.amazon.com/codesuite/codebuild/home>。
2. 在导航窗格中，选择 Build history。
3. 在构建列表中，构建运行列，选择要查看利用率度量标准的构建。
4. 选择 资源利用率 选项卡。
5. 要查看在 CloudWatch 控制台，选择 在CloudWatch中查看 在 资源利用率指标 第节。

## 访问资源利用率度量 ( Amazon CloudWatch 控制台 )

The Amazon CloudWatch 控制台可用于访问 CodeBuild 资源利用率度量。

### 项目级别资源利用率指标

#### 如何访问项目级别资源利用率度量

1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在全部指标选项卡上，选择 CodeBuild。

Metrics

Favorites

 Add a dashboard

4. 选择按项目。
5. 选择一个或多个项目和度量组合以添加到图表。此页面上的图表中将显示所有选定项目和指标组合。
6. ( 可选 ) 您可以自定义您的度量标准和图表 绘制的度量标准 选项卡。例如，从 统计数据 列，您可以选择要显示的不同统计信息。或从 Period (周期) 列的下拉菜单中，可以选择要用于监控指标的各个时间段。

有关详细信息，请参阅 [绘制度量标准](#) 和 [查看可用度量](#) 在 Amazon CloudWatch 用户指南.

## 构建级别资源利用率度量

### 如何访问构建级别资源利用率度量

1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在全部指标选项卡上，选择 CodeBuild。

Metrics

Favorites

 Add a dashboard

4. 选择 BuildId、BuildNumber、ProjectName.
5. 选择一个或多个构建和度量组合以添加到图表。所有选定的构建和度量组合都会显示在页面上的图表中。
6. ( 可选 ) 您可以自定义您的度量标准和图表 绘制的度量标准 选项卡。例如，从 统计数据 列，您可以选择要显示的不同统计信息。或从 Period (周期) 列的下拉菜单中，可以选择要用于监控指标的各个时间段。

有关详细信息，请参阅 [绘制度量标准](#) 和 [查看可用度量](#) 在 Amazon CloudWatch 用户指南.

## 使用 CloudWatch 警报监控构建

可以为构建创建 CloudWatch 警报。警报将监控某个指标在一个时间段（由您指定）的变化情况，并根据相对于指定阈值的指标值每隔若干个时间段执行一个或多个操作。通过使用本机 CloudWatch 警报功能，您可以指定在超出阈值时 CloudWatch 支持的任何操作。例如，可以指定 15 分钟内如果账户中有三个以上的构建失败，则发送 Amazon SNS 通知。

### 为 CodeBuild 指标创建 CloudWatch 警报

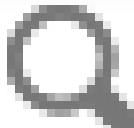
1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms。
3. 选择 Create Alarm。
4. 在按类别显示的 CloudWatch 指标下，选择 CodeBuild 指标。如果您知道您只需要项目级别指标，则选择按项目。如果您知道您只需要账户级别指标，则选择账户指标。

# Create Alarm

## 1. Select Metric

2.

Browse Metrics



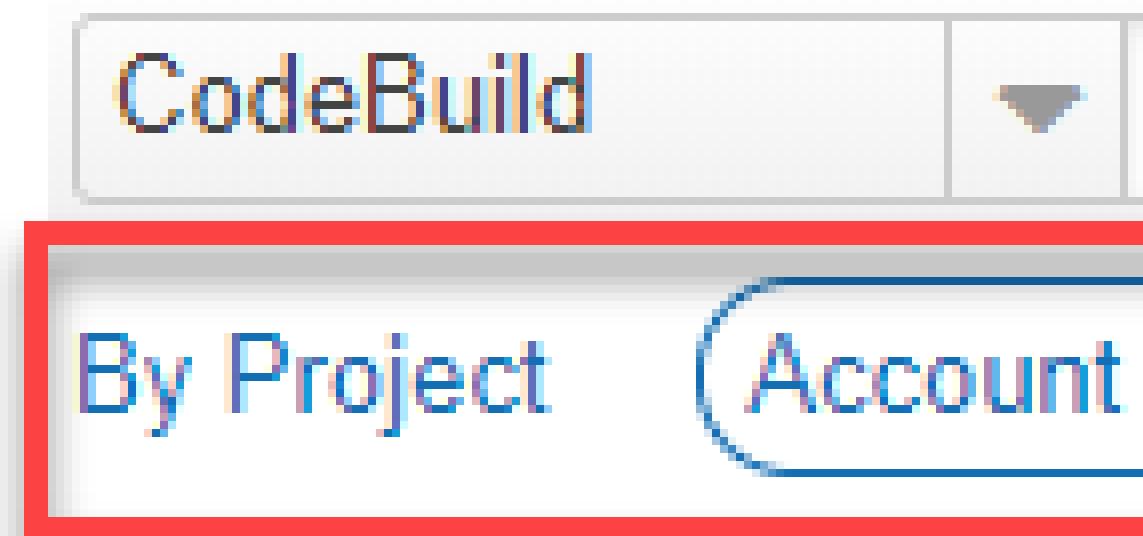
# CloudWatch Metrics

Your CloudWatch metric streams

5. 在创建警报上，请选择选择指标（如果尚未选择）。
6. 选择要为之创建警报的指标。选项为按项目或账户指标。

# Create Alarm

## 1. Select Metric



CodeBuild > Account

7. 选择下一步或定义警报，然后创建警报。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的[创建 Amazon CloudWatch 警报](#)。有关设置触发警报时的 Amazon SNS 通知的更多信息，请参阅 Amazon SNS 开发人员指南 中的[设置 Amazon SNS 通知](#)。

下面显示了一个警报，当在 15 分钟的时间内检测到一个或多个失败构建时，该警报会将 Amazon SNS 通知发送到名为 codebuild-sns-notifications 的列表。15 分钟是由 5 分钟周期乘 3 个指定数据点计算得出的。项目级别或账户级别的失败构建警报显示的信息是相同的。

# Create Alarm

## 1. Select Metric

Alarm Threshold

Provide the details and appropriate threshold.

Name:

8. 选择 Create Alarm。

# AWS CodeBuild 中的安全性

AWS 的云安全性的优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性和合规性是 AWS 与您共同承担的责任。此责任共担模式能够减轻您的操作负担：AWS 运行、管理和控制从主机操作系统和虚拟化层到服务设施的物理安全性等各种组件。您负责并管理来宾操作系统（包括更新和安全补丁程序）和其他相关应用软件。您还负责 AWS 提供的安全组防火墙的配置。您的责任因您使用的服务、此类服务与您的 IT 环境的集成以及适用的法律和法规而异。因此，您应该仔细考虑组织使用的服务。有关更多信息，请参阅[责任共担模式](#)。

要了解如何保护您的 CodeBuild 资源，请参阅以下主题。

## 主题

- [AWS CodeBuild 中的数据保护 \(p. 323\)](#)
- [AWS CodeBuild 中的 Identity and access management \(p. 325\)](#)
- [AWS CodeBuild 的合规性验证 \(p. 351\)](#)
- [AWS CodeBuild 中的弹性 \(p. 352\)](#)
- [AWS CodeBuild 中的基础设施安全性 \(p. 352\)](#)

## AWS CodeBuild 中的数据保护

AWS CodeBuild 符合 AWS [责任共担模式](#)，该模式包含适用于数据保护的法规和准则。AWS 负责保护运行所有 AWS 服务的全球基础设施。AWS 保持对该基础设施上托管的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。作为数据控制者或数据处理器，AWS 客户和 APN 合作伙伴对他们放在 AWS 云中的任何个人数据承担责任。

出于数据保护的目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置单个用户账户，以便仅向每个用户提供履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 TLS 与 AWS 资源进行通信。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如 Name（名称）字段）。这包括使用控制台、API、AWS CLI 或 AWS 开发工具包处理 CodeBuild 或其他 AWS 服务时。您输入到 CodeBuild 或其他服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供 URL 时，请勿在 URL 中包含凭证信息来验证您对该服务器的请求。

为了保护敏感信息，CodeBuild 日志中隐藏以下内容：

- AWS 访问密钥 ID。有关详细信息，请参阅[管理访问密钥 IAM 用户](#) In the AWS Identity and Access Management User Guide.

- 使用参数存储指定的字符串。有关详细信息，请参阅 [系统管理器参数存储](#) 和 [系统管理器参数存储控制台巡视](#) 在 Amazon EC2 Systems Manager 用户指南。
- 使用 AWS Secrets Manager 指定的字符串。有关更多信息，请参阅 [密钥管理 \(p. 324\)](#)。)

有关数据保护的更多信息，请参阅 [AWS 共同责任模式和GDPR](#) 博客邮寄在 AWS 安全博客。

#### 主题

- [数据加密 \(p. 324\)](#)
- [密钥管理 \(p. 324\)](#)
- [流量隐私 \(p. 324\)](#)

## 数据加密

加密是 CodeBuild 安全性的一个重要组成部分。某些加密（例如，针对传输中的数据的加密）是默认提供的，无需您执行任何操作。其他加密（例如，针对静态数据的加密）可在创建项目或构建时进行配置。

- 静态数据加密 - 默认情况下，使用 AWS Key Management Service 托管的 Amazon S3 的客户主密钥（CMK）对构建构件（例如，缓存、日志、导出的原始测试报告数据文件和构建结果）进行加密。如果您不想使用这些 CMK，则必须创建并配置一个客户管理的 CMK。了解更多信息 [创建KMS密钥](#) 和 [AWS 关键管理服务概念](#) 在 AWS Key Management Service 用户指南。
- 您可以将 CodeBuild 用来对构建输出构件进行加密的 AWS KMS 密钥的标识符存储在 `CODEBUILD_KMS_KEY_ID` 环境变量中。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)
- 可以在创建构建项目时指定客户管理的 CMK。有关详细信息，请参阅 [Set the Encryption Key Using the Console](#) 和 [使用CLI设置加密密钥](#)。

默认情况下，使用 AWS 托管的 CMK 对构建队列的 Amazon Elastic Block Store 卷进行加密。

- 传输中的数据加密 - 使用通过签名版本 4 签名流程签名的 TLS 连接来保护客户与 CodeBuild 之间以及 CodeBuild 与其下游依赖项之间的所有通信。所有 CodeBuild 终端节点都使用由 AWS Certificate Manager 私有证书颁发机构管理的 SHA-256 证书。有关更多信息，请参阅 [签名版本 4 签名流程](#) 和 [什么是 ACM PCA](#)。
- 构建构件加密 - CodeBuild 需要访问 AWS KMS CMK 以便对其构建输出构件进行加密。默认情况下，CodeBuild 使用您的 AWS 账户中适用于 Amazon S3 的 AWS Key Management Service CMK。如果您不想使用此 CMK，则必须创建并配置一个客户管理的 CMK。有关更多信息，请参阅 [创建密钥](#)。

## 密钥管理

可以通过加密保护您的内容免遭未经授权的使用。在 AWS Secrets Manager 中存储加密密钥，然后向 CodeBuild 授予从 Secrets Manager 账户获取加密密钥的权限。有关详细信息，请参阅 [为 CodeBuild 创建和配置 AWS KMS CMK \(p. 361\)](#)，[在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)，[在 AWS CodeBuild 中运行构建 \(p. 252\)](#)，和 [教程: 存储和检索密码](#)。

在 AWS KMS 密钥的构建命令中使用 `CODEBUILD_KMS_KEY` 环境变量。有关更多信息，请参阅 [构建环境中的环境变量 \(p. 168\)](#)。)

可以使用 Secrets Manager 保护存储用于运行时环境的 Docker 映像的专用注册表的凭证。有关更多信息，请参阅 [适用于 CodeBuild 的私有注册表与 AWS Secrets Manager 示例 \(p. 129\)](#)。)

## 流量隐私

您可以通过将 CodeBuild 配置为使用接口 VPC 终端节点来提高构建的安全性。为此，您无需互联网网关、NAT 设备或虚拟私有网关。也不要求您配置 PrivateLink，但建议进行配置。有关更多信息，请参阅 [使](#)

用 VPC 终端节点 (p. 175)。有关 PrivateLink 和 VPC 终端节点的更多信息，请参阅 [AWS PrivateLink 和通过 PrivateLink 访问 AWS 服务](#)。

# AWS CodeBuild 中的 Identity and access management

访问 AWS CodeBuild 需要凭证。这些凭证必须有权访问 AWS 资源，如存储和检索 S3 存储桶中的构建构件并查看构建的 Amazon CloudWatch Logs。以下几节介绍如何使用 [AWS Identity and Access Management \(IAM\)](#) 和 CodeBuild 帮助保护对您的资源的访问：

- [Authentication \(p. 325\)](#)
- [访问控制 \(p. 326\)](#)

## Authentication

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。然后，您可以使用该管理员用户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 <https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#create-iam-users> 中的 IAM 最佳实践和 IAM 用户指南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，它具有自定义权限（例如，在 CodeBuild 中创建构建项目的权限）。您可以使用 IAM 用户名和密码登录以保护 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[AWS 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。AWS 开发工具包和 AWS 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。CodeBuild 支持签名版本 4，后者是一种用于对入站 API 请求进行身份验证的协议。有关对请求进行身份验证的更多信息，请参阅 <https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html> 中的 AWS General Reference 签名版本 4 签名流程。

- IAM 角色 – [IAM 角色](#)类似于 IAM 用户，但不与特定人员关联。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。这些用户被称为联合身份用户。在通过[身份提供商](#)请求访问权限时，AWS 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction\\_access-management.html#intro-access-roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_access-management.html#intro-access-roles) 中的 IAM 用户指南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。例如，请参阅 [教程: DelegateAccessAcross AWS 账户使用 IAM 角色](#) 在 IAM 用户指南。
  - AWS 服务访问 – 您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 S3 存储桶，然后将存储在存储桶中的数据加载到 Amazon Redshift 集群中。有关更多信息，请参阅 [AWS 中的创建向 IAM 用户指南 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 Amazon 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 Amazon 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 Amazon 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 [Amazon EC2 中的对 IAM 用户指南 上的应用程序使用角色](#)。

## 访问控制

您可以使用有效的凭证对您的请求进行身份验证，但除非您具有权限，否则，无法创建或访问 AWS CodeBuild 资源。例如，您必须拥有权限才能创建、查看或删除构建项目并启动、停止或查看构建项目。

下面几节介绍如何管理 CodeBuild 的权限：我们建议您先阅读概述。

- [管理 AWS CodeBuild 资源的访问权限的概述 \(p. 326\)](#)
- [为 AWS CodeBuild 使用基于身份的策略 \(p. 329\)](#)
- [AWS CodeBuild 权限参考 \(p. 344\)](#)
- [在控制台中查看资源 \(p. 351\)](#)

## 管理 AWS CodeBuild 资源的访问权限的概述

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。

### 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](#)。

在您授予权限时，您要决定谁将获得权限、这些人可以访问的资源以及可以对这些资源执行的操作。

### 主题

- [AWS CodeBuild 资源和操作 \(p. 326\)](#)
- [了解资源所有权 \(p. 327\)](#)
- [管理对资源的访问 \(p. 327\)](#)
- [指定策略元素: 行动、效果和负责人 \(p. 328\)](#)

## AWS CodeBuild 资源和操作

在 AWS CodeBuild 中，构建项目是主要资源。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。构建项目也是资源，且具有相关联的 ARN。有关更多信息，请参阅 [AWS 中的 Amazon 资源名称 \(ARN\) 和 Amazon Web Services 一般参考 服务命名空间](#)。

资源类型	ARN 格式
构建项目	<code>arn:aws:codebuild:&lt;region-ID&gt;:&lt;account-ID&gt;:project/&lt;project-name&gt;</code>
构建	<code>arn:aws:codebuild:&lt;region-ID&gt;:&lt;account-ID&gt;:build/&lt;build-ID&gt;</code>
报告组	<code>arn:aws:codebuild:&lt;region-ID&gt;:&lt;account-ID&gt;:report-group/&lt;report-group-name&gt;</code>
报告	<code>arn:aws:codebuild:&lt;region-ID&gt;:&lt;account-ID&gt;:report/&lt;report-ID&gt;</code>
所有 CodeBuild 资源	<code>arn:aws:codebuild:*</code>

资源类型	ARN 格式
指定账户在指定 AWS 区域拥有的所有 CodeBuild 资源	arn:aws:codebuild: <i>region-ID</i> : <i>account-ID</i> :*

Note

大多数 AWS 服务将 ARN 中的冒号 (:) 或正斜杠 (/) 视为同一个字符。不过，CodeBuild 在资源模式和规则中使用精确匹配。请务必在创建事件模式时使用正确的字符，以使其与资源中的 ARN 语法匹配。

例如，您可以指定一个特定的构建项目 (*myBuildProject*) 在您的声明中使用 ARN 如下所述：

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

要指定所有资源，或者如果 API 操作不支持 ARN，请在 Resource 元素中使用通配符 (\*)，如下所示：

```
"Resource": "*"
```

有些 CodeBuild API 操作接受多个资源（例如，BatchGetProjects）。要在单个语句中指定多个资源，请使用逗号分隔其 ARN，如下所示。

```
"Resource": [  
    "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",  
    "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"  
]
```

CodeBuild 提供一组操作来处理 CodeBuild 资源。有关列表，请参阅[AWS CodeBuild 权限参考 \(p. 344\)](#)。

## 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 CodeBuild 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 CodeBuild 资源的权限，则该用户可以创建 CodeBuild 资源。但是，该用户所属的 AWS 账户拥有这些 CodeBuild 资源。
- 如果您在您的 AWS 账户中创建具有创建 CodeBuild 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 CodeBuild 资源。该角色所属的 AWS 账户拥有这些 CodeBuild 资源。

## 管理对资源的访问

权限策略规定谁可以访问哪些资源。

Note

本节讨论如何在 AWS CodeBuild 中使用 IAM。它不提供有关 IAM 服务的详细信息。完成 IAM 文档，参阅[什么是 IAM?](#) 在 IAM 用户指南。有关 IAM 策略语法和描述，请参阅[AWS IAM 政策参考](#) 在 IAM 用户指南。

附加到 IAM 身份的策略称为“基于身份的策略”（IAM 策略）。附加到资源的策略称为“基于资源的策略”。CodeBuild 仅支持基于身份的策略（IAM 策略）。

## 基于身份的策略

您可以向 IAM 身份挂载策略。

- 将权限策略附加到您的账户中的用户或组 – 要向用户授予在 AWS CodeBuild 控制台中查看构建项目和其他 AWS CodeBuild 资源的权限，您可以将权限策略附加到用户或用户所属的组。
- 向角色附加权限策略（授予跨账户权限）– 您可以向 IAM 角色附加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 标识为能够代入该角色的委托人的角色附加信任策略。
  3. 之后，账户 B 管理员可以委派权限，以指派账户 B 中的任何用户代入该角色。这样，账户 B 中的用户便可以创建或访问账户 A 中的资源。如果您需要授予 AWS 服务相应的权限以代入该角色，则信任策略中的委托人还必须是 AWS 服务委托人。

有关使用 IAM 委派权限的更多信息，请参阅 <https://docs.aws.amazon.com/IAM/latest/UserGuide/access.html> 中的 IAM 用户指南访问权限管理。

在 CodeBuild 中，基于身份的策略用于管理对与部署过程相关的资源的权限。例如，您可以控制对构建项目的访问。

您可以创建 IAM 策略来限制您账户中的用户有权访问的调用和资源，然后将这些策略附加到 IAM 用户。有关如何创建 IAM 角色和探索 CodeBuild 的示例 IAM 策略语句的更多信息，请参阅 [管理 AWS CodeBuild 资源的访问权限的概述 \(p. 326\)](#)。

## 对 S3 存储桶的安全访问

我们强烈建议您在 IAM 角色中包括以下权限，确保与 CodeBuild 项目关联的 S3 存储桶由您或其他值得信任的人拥有。这些权限未包含在 AWS 托管策略和角色中。您必须自行添加。

- `s3:GetBucketACL`
- `s3:GetBucketLocation`

如果您的项目使用的 S3 存储桶的拥有者发生更改，则必须验证您是否仍拥有存储桶，如果没有，请更新您的 IAM 角色的权限。有关更多信息，请参阅 [向 IAM 组或 IAM 用户添加 CodeBuild 访问权限 \(p. 353\)](#) 和 [创建 CodeBuild 服务角色 \(p. 357\)](#)。

## 指定策略元素：行动、效果和负责人

对于每种 AWS CodeBuild 资源，该服务都定义了一组 API 操作。为了授予执行这些 API 操作的权限，CodeBuild 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。有关更多信息，请参阅 [AWS CodeBuild 资源和操作 \(p. 326\)](#) 和 [AWS CodeBuild 权限参考 \(p. 344\)](#)。

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。
- **操作** – 您可以使用操作关键字来标识要允许或拒绝的资源操作。例如，`codebuild:CreateProject` 权限授予用户执行 `CreateProject` 操作的权限。
- **效果** – 用于指定在用户请求操作时的效果（可以是允许或拒绝）。如果没有显式授予允许资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问。您可以执行此操作，以确保用户无法访问资源，即使有其他策略授予了访问权限也是如此。
- **委托人** – 在基于身份的策略（IAM 策略）中，策略所附加到的用户是隐式委托人。对于基于资源的策略，您可以指定您希望将权限授予的用户、账户、服务或其他实体。

要了解 IAM 策略语法和说明的更多信息，请参阅 [IAM 中的 AWS IAM 用户指南 策略参考](#)。

有关显示所有 CodeBuild API 操作及其适用的资源的表，请参阅[AWS CodeBuild 权限参考 \(p. 344\)](#)。

## 为 AWS CodeBuild 使用基于身份的策略

本主题提供基于身份的策略的示例，这些示例展示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 CodeBuild 资源执行操作的权限。

### Important

我们建议您首先阅读以下介绍性主题，这些主题说明了可用于管理 CodeBuild 资源访问的基本概念和选项。有关更多信息，请参阅[管理 AWS CodeBuild 资源的访问权限的概述 \(p. 326\)](#)。)

### 主题

- [使用 AWS CodeBuild 控制台所需的权限 \(p. 329\)](#)
- [AWS CodeBuild 控制台连接到源提供程序所需的权限 \(p. 330\)](#)
- [适用于 AWS CodeBuild 的 AWS 托管 \( 预定义 \) 策略 \(p. 330\)](#)
- [CodeBuild managed policies and notifications \(p. 334\)](#)
- [客户管理的策略示例 \(p. 337\)](#)

以下是一个权限策略示例，仅允许用户在 123456789012 账户的 us-east-2 区域中获取任何以 my 开头的构建项目的相关信息：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchGetProjects",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"  
        }  
    ]  
}
```

## 使用 AWS CodeBuild 控制台所需的权限

使用 AWS CodeBuild 控制台的用户必须拥有一组最低权限，这些权限允许用户描述 AWS 账户的其他 AWS 资源。您必须拥有来自以下服务的权限：

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit ( 如果您要将源代码存储在 AWS CodeCommit 存储库中 )
- Amazon Elastic Container Registry (Amazon ECR) ( 如果您要使用依赖于 Amazon ECR 存储库中 Docker 映像的构建环境 )
- Amazon Elastic Container Service (Amazon ECS) ( 如果您要使用依赖于 Amazon ECR 存储库中 Docker 映像的构建环境 )
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

如果您创建比必需的最低权限更为严格的 IAM 策略，控制台将无法按预期正常运行。

## AWS CodeBuild 控制台连接到源提供程序所需的权限

AWS CodeBuild 控制台使用以下 API 操作连接到源提供程序 (例如，GitHub 存储库)。

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild>ListRepositories`
- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

您可以使用 AWS CodeBuild 控制台将源提供程序 (如 GitHub 存储库) 与您的构建项目相关联。为此，您必须先将上述 API 操作添加到与用于访问 AWS CodeBuild 控制台的 IAM 用户关联的 IAM 访问策略。

`ListConnectedOAuthAccounts`、`ListRepositories` 和 `PersistOAuthToken` API 操作不应由您的代码调用。因此，这些 API 操作未包含在 AWS CLI 和 AWS 开发工具包中。

## 适用于 AWS CodeBuild 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，满足许多常用案例的要求。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。CodeBuild 的托管策略还提供在其他服务 (如 IAM、AWS CodeCommit、Amazon EC2、Amazon ECR、Amazon SNS 和 Amazon CloudWatch Events) 中执行操作的权限，这是授予了相关策略的用户职责所必需的。例如，`AWSCodeBuildAdminAccess` 策略是管理级用户策略，允许具有此策略的用户为项目构建创建和管理 CloudWatch 事件规则，并为项目相关事件的通知创建和管理 Amazon SNS 主题 (名称前缀为 `arn:aws:codebuild:` 的主题)，以及在 CodeBuild 中管理项目和报告组。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 ( 可附加到账户中的用户 ) 特定于 AWS CodeBuild :

- `AWSCodeBuildAdminAccess` – 提供对 CodeBuild 的完全访问权限 ( 包括管理 CodeBuild 构建项目的权限 )。
- `AWSCodeBuildDeveloperAccess` – 提供对 CodeBuild 的访问权限，但不允许对构建项目进行管理。
- `AWSCodeBuildReadOnlyAccess` – 提供对 CodeBuild 的只读访问权限。

要访问 CodeBuild 创建的构建输出构件，您还必须附加名为 `AmazonS3ReadOnlyAccess` 的 AWS 托管策略。

要创建和管理 CodeBuild 服务角色，您还必须附加名为 `IAMFullAccess` 的 AWS 托管策略。

此外，您还可以创建自己的自定义 IAM 策略，以授予对 CodeBuild 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

### 主题

- [AWSCodeBuildAdminAccess \(p. 330\)](#)
- [AWSCodeBuildDeveloperAccess \(p. 332\)](#)
- [AWSCodeBuildReadOnlyAccess \(p. 333\)](#)

### [AWSCodeBuildAdminAccess](#)

`AWSCodeBuildAdminAccess` – 提供对 CodeBuild 的完全访问权限 ( 包括管理 CodeBuild 构建项目的权限 )。仅将此策略应用于管理级别的用户，以便向其授予对 AWS 账户中的 CodeBuild 项目、报告组和相关资源的完全控制权限 ( 包括删除项目和报告组的能力 )。

`AWSCodeBuildAdminAccess` 策略包含以下策略语句：

```
{
```

```
"Version": "2012-10-17",
"Statement": [
    {
        "Action": [
            "codebuild:*",
            "codecommit:GetBranch",
            "codecommit:GetCommit",
            "codecommit:GetRepository",
            "codecommit>ListBranches",
            "codecommit>ListRepositories",
            "cloudwatch:GetMetricStatistics",
            "ec2:DescribeVpcs",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ecr:DescribeRepositories",
            "ecr>ListImages",
            "events>DeleteRule",
            "events>DescribeRule",
            "events>DisableRule",
            "events>EnableRule",
            "events>ListTargetsByRule",
            "events>ListRuleNamesByTarget",
            "events>PutRule",
            "events>PutTargets",
            "events>RemoveTargets",
            "logs:GetLogEvents",
            "s3:GetBucketLocation",
            "s3>ListAllMyBuckets"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "logs>DeleteLogGroup"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:logs:***:log-group:/aws/codebuild/*:log-stream:***"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ssm:PutParameter"
        ],
        "Resource": "arn:aws:ssm:***:parameter/CodeBuild/*"
    },
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications>CreateNotificationRule",
            "codestar-notifications>DescribeNotificationRule",
            "codestar-notifications>UpdateNotificationRule",
            "codestar-notifications>DeleteNotificationRule",
            "codestar-notifications>Subscribe",
            "codestar-notifications>Unsubscribe"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:***"
            }
        }
    }
]
```

```
"Sid": "CodeStarNotificationsListAccess",
"Effect": "Allow",
>Action": [
    "codestar-notifications>ListNotificationRules",
    "codestar-notifications>ListEventTypes",
    "codestar-notifications>ListTargets",
    "codestar-notifications>ListTagsforResource"
],
"Resource": "*"
},
{
"Sid": "CodeStarNotificationsSNSTopicCreateAccess",
"Effect": "Allow",
>Action": [
    "sns>CreateTopic",
    "sns:SetTopicAttributes"
],
"Resource": "arn:aws:sns:*::codestar-notifications"
},
{
"Sid": "SNSTopicListAccess",
"Effect": "Allow",
>Action": [
    "sns>ListTopics",
    "sns:GetTopicAttributes"
],
"Resource": "*"
}
]
```

## AWSCodeBuildDeveloperAccess

**AWSCodeBuildDeveloperAccess** – 允许访问 CodeBuild 的所有功能，以及与项目和报告组相关的资源。此策略不允许用户删除 CodeBuild 项目或报告组，或其他 AWS 服务中的相关资源（例如 CloudWatch Events）。建议对大多数用户应用此策略。

**AWSCodeBuildDeveloperAccess** 策略包含以下策略语句：

```
{
    "Statement": [
        {
            "Action": [
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "codebuild:BatchGet*",
                "codebuild:GetResourcePolicy",
                "codebuild:DescribeTestCases",
                "codebuild>List*",
                "codecommit:GetBranch",
                "codecommit:GetCommit",
                "codecommit:GetRepository",
                "codecommit>ListBranches",
                "cloudwatch:GetMetricStatistics",
                "events:DescribeRule",
                "events>ListTargetsByRule",
                "events>ListRuleNamesByTarget",
                "logs:GetLogEvents",
                "s3:GetBucketLocation",
                "s3>ListAllMyBuckets"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
    ]
}
```

```
{  
    "Effect": "Allow",  
    "Action": [  
        "ssm:PutParameter"  
    ],  
    "Resource": "arn:aws:ssm:*::parameter/CodeBuild/*"  
,  
{  
    "Sid": "CodeStarNotificationsReadWriteAccess",  
    "Effect": "Allow",  
    "Action": [  
        "codestar-notifications>CreateNotificationRule",  
        "codestar-notifications:DescribeNotificationRule",  
        "codestar-notifications:UpdateNotificationRule",  
        "codestar-notifications:Subscribe",  
        "codestar-notifications:Unsubscribe"  
    ],  
    "Resource": "*",  
    "Condition": {  
        "StringLike": {  
            "codestar-notifications:NotificationsForResource":  
"arn:aws:codebuild:*"  
        }  
    },  
},  
{  
    "Sid": "CodeStarNotificationsListAccess",  
    "Effect": "Allow",  
    "Action": [  
        "codestar-notifications>ListNotificationRules",  
        "codestar-notifications>ListEventTypes",  
        "codestar-notifications>ListTargets",  
        "codestar-notifications>ListTagsForResource"  
    ],  
    "Resource": "*"  
,  
{  
    "Sid": "SNSTopicListAccess",  
    "Effect": "Allow",  
    "Action": [  
        "sns>ListTopics",  
        "sns:GetTopicAttributes"  
    ],  
    "Resource": "*"  
}  
],  
"Version": "2012-10-17"  
}
```

## AWSCodeBuildReadOnlyAccess

**AWSCodeBuildReadOnlyAccess** – 授予对 CodeBuild 以及其他 AWS 服务中的相关资源的只读访问权限。将此策略应用于可以查看和运行构建、查看项目和查看报告组但无法对它们作出任何更改的用户。

AWSCodeBuildReadOnlyAccess 策略包含以下策略语句：

```
{  
    "Statement": [  
        {  
            "Action": [  
                "codebuild:BatchGet*",  
                "codebuild:GetResourcePolicy",  
                "codebuild>List*",  
                "codebuild:DescribeTestCases",
```

```
    "codecommit:GetBranch",
    "codecommit:GetCommit",
    "codecommit:GetRepository",
    "cloudwatch:GetMetricStatistics",
    "events:DescribeRule",
    "events>ListTargetsByRule",
    "events>ListRuleNamesByTarget",
    "logs:GetLogEvents"
],
"Effect": "Allow",
"Resource": "*"
},
{
"Sid": "CodeStarNotificationsPowerUserAccess",
"Effect": "Allow",
"Action": [
    "codestar-notifications:DescribeNotificationRule"
],
"Resource": "*",
"Condition": {
    "StringLike": {
        "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*
    }
}
},
{
"Sid": "CodeStarNotificationsListAccess",
"Effect": "Allow",
"Action": [
    "codestar-notifications>ListNotificationRules",
    "codestar-notifications>ListEventTypes"
],
"Resource": "*"
},
"Version": "2012-10-17"
}
```

## CodeBuild managed policies and notifications

CodeBuild supports notifications, which can notify users of important changes to build projects. Managed policies for CodeBuild include policy statements for notification functionality. For more information, see [What are notifications?](#).

### Permissions related to notifications in full access managed policies

The `AWSCodeBuildFullAccess` managed policy includes the following statements to allow full access to notifications. Users with this managed policy applied can also create and manage Amazon SNS topics for notifications, subscribe and unsubscribe users to topics, list topics to choose as targets for notification rules, and list AWS Chatbot clients configured for Slack.

```
{
    "Sid": "CodeStarNotificationsReadWriteAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>CreateNotificationRule",
        "codestar-notifications>DescribeNotificationRule",
        "codestar-notifications>UpdateNotificationRule",
        "codestar-notifications>DeleteNotificationRule",
        "codestar-notifications>Subscribe",
        "codestar-notifications>Unsubscribe"
```

```
],
"Resource": "*",
"Condition" : {
    "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*"}
}
},
{
    "Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>ListNotificationRules",
        "codestar-notifications>ListTargets",
        "codestar-notifications>ListTagsForResource",
        "codestar-notifications>ListEventTypes"
    ],
    "Resource": "*"
},
{
    "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
    "Effect": "Allow",
    "Action": [
        "sns>CreateTopic",
        "sns:SetTopicAttributes"
    ],
    "Resource": "arn:aws:sns:::*:codestar-notifications*"
},
{
    "Sid": "SNSTopicListAccess",
    "Effect": "Allow",
    "Action": [
        "sns>ListTopics"
    ],
    "Resource": "*"
},
{
    "Sid": "CodeStarNotificationsChatbotAccess",
    "Effect": "Allow",
    "Action": [
        "chatbot>DescribeSlackChannelConfigurations"
    ],
    "Resource": "*"
}
```

## Permissions related to notifications in read-only managed policies

The AWSCodeBuildReadOnlyAccess managed policy includes the following statements to allow read-only access to notifications. Users with this managed policy applied can view notifications for resources, but cannot create, manage, or subscribe to them.

```
{
    "Sid": "CodeStarNotificationsPowerUserAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>DescribeNotificationRule"
    ],
    "Resource": "*",
    "Condition" : {
        "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild:*"}
    }
},
{
    "Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>ListNotificationRules",
        "codestar-notifications>ListTargets",
        "codestar-notifications>ListTagsForResource",
        "codestar-notifications>ListEventTypes"
    ],
    "Resource": "*"
}
```

```
        "Effect": "Allow",
        "Action": [
            "codestar-notifications>ListNotificationRules",
            "codestar-notifications>ListEventTypes",
            "codestar-notifications>ListTargets"
        ],
        "Resource": "*"
    }
```

## Permissions related to notifications in other managed policies

The `AWSCodeBuildDeveloperAccess` managed policy includes the following statements to allow users to create, edit, and subscribe to notifications. Users cannot delete notification rules or manage tags for resources.

```
{
    "Sid": "CodeStarNotificationsReadWriteAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>CreateNotificationRule",
        "codestar-notifications>DescribeNotificationRule",
        "codestar-notifications>UpdateNotificationRule",
        "codestar-notifications>Subscribe",
        "codestar-notifications>Unsubscribe"
    ],
    "Resource": "*",
    "Condition" : {
        "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codebuild*"}
    }
},
{
    "Sid": "CodeStarNotificationsListAccess",
    "Effect": "Allow",
    "Action": [
        "codestar-notifications>ListNotificationRules",
        "codestar-notifications>ListTargets",
        "codestar-notifications>ListTagsforResource",
        "codestar-notifications>ListEventTypes"
    ],
    "Resource": "*"
},
{
    "Sid": "SNSTopicListAccess",
    "Effect": "Allow",
    "Action": [
        "sns>ListTopics"
    ],
    "Resource": "*"
},
{
    "Sid": "CodeStarNotificationsChatbotAccess",
    "Effect": "Allow",
    "Action": [
        "chatbot>DescribeSlackChannelConfigurations"
    ],
    "Resource": "*"
}
```

For more information about IAM and notifications, see [Identity and Access Management for AWS CodeStar Notifications](#).

## 客户管理的策略示例

本节的用户策略示例介绍如何授予执行 AWS CodeBuild 操作的权限。当您使用 CodeBuild 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，您必须授予特定于控制台的其他权限。有关信息，请参阅[使用 AWS CodeBuild 控制台所需的权限 \(p. 329\)](#)。

您可以使用以下示例 IAM 策略来限制 IAM 用户和角色对 CodeBuild 的访问。

### 主题

- 允许用户获取有关构建项目的信息 (p. 337)
- 允许用户获取有关报告组的信息 (p. 338)
- 允许用户获取有关报告的信息 (p. 338)
- 允许用户创建构建项目 (p. 338)
- 允许用户创建报告组 (p. 338)
- 允许用户删除报告组 (p. 339)
- 允许用户删除报告 (p. 339)
- 允许用户删除构建项目 (p. 339)
- 允许用户获取构建项目名称的列表 (p. 339)
- 允许用户更改有关构建项目的信息 (p. 340)
- 允许用户更改报告组 (p. 340)
- 允许用户获取有关构建的信息 (p. 340)
- 允许用户获取构建项目的构建 ID 的列表 (p. 341)
- 允许用户获取构建 ID 的列表 (p. 341)
- 允许用户获取报告组列表 (p. 341)
- 允许用户获取报告列表 (p. 341)
- 允许用户获取报告组的报告列表 (p. 342)
- 允许用户获取报告的测试用例的列表 (p. 342)
- 允许用户开始运行构建 (p. 342)
- 允许用户尝试停止构建 (p. 342)
- 允许用户尝试删除构建 (p. 343)
- 允许用户获取有关由 CodeBuild 管理的 Docker 映像的信息 (p. 343)
- 允许 CodeBuild 访问创建 VPC 网络接口时所需的 AWS 服务 (p. 343)
- 使用 Deny 语句可阻止 AWS CodeBuild 与源提供程序断开连接 (p. 344)

### 允许用户获取有关构建项目的信息

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取任何以名称 my 开头的构建项目的信息：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchGetProjects",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"  
        }  
    ]  
}
```

## 允许用户获取有关报告组的信息

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取有关报告组的信息：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchGetReportGroups",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"  
        }  
    ]  
}
```

## 允许用户获取有关报告的信息

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取有关报告的信息：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchGetReports",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"  
        }  
    ]  
}
```

## 允许用户创建构建项目

以下示例策略语句允许用户创建使用任何名称的构建项目，但只能在 123456789012 账户的 us-east-2 区域中创建，并且只能使用指定的 CodeBuild 服务角色：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild>CreateProject",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"  
        }  
    ]  
}
```

## 允许用户创建报告组

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中创建报告组：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:CreateReportGroup",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"  
        }  
    ]  
}
```

```
        "Action": "codebuild>CreateReportGroup",
        "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
]
```

## 允许用户删除报告组

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中删除报告组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>DeleteReportGroup",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

## 允许用户删除报告

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中删除报告：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>DeleteReport",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

## 允许用户删除构建项目

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中删除任何以名称 my 开头的构建项目：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>DeleteProject",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

## 允许用户获取构建项目名称的列表

以下示例策略语句允许用户获取同一账户的构建项目名称的列表：

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

```

```
{  
    "Effect": "Allow",  
    "Action": "codebuild>ListProjects",  
    "Resource": "*"  
}  
]  
}
```

## 允许用户更改有关构建项目的信息

以下示例策略语句仅允许用户在 123456789012 账户的 us-east-2 区域中更改有关使用任何名称的构建项目的信息，并且只能使用指定的 AWS CodeBuild 服务角色：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:UpdateProject",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam:123456789012:role/CodeBuildServiceRole"  
        }  
    ]  
}
```

## 允许用户更改报告组

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中更改报告组：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:UpdateReportGroup",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"  
        }  
    ]  
}
```

## 允许用户获取有关构建的信息

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取名为 my-build-project 和 my-other-build-project 的构建项目的信息：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchGetBuilds",  
            "Resource": [  
                "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",  
                "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"  
            ]  
        }  
    ]  
}
```

```
}
```

## 允许用户获取构建项目的构建 ID 的列表

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取名为 my-build-project 和 my-other-build-project 的构建项目的构建 ID 列表：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>ListBuildsForProject",
            "Resource": [
                "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
                "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
            ]
        }
    ]
}
```

## 允许用户获取构建 ID 的列表

以下示例策略语句允许用户获取同一账户的所有构建 ID 的列表：

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

## 允许用户获取报告组列表

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取有关报告组的列表：

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

## 允许用户获取报告列表

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取有关报告的列表：

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

```

```
        "Effect": "Allow",
        "Action": "codebuild>ListReports",
        "Resource": "*"
    }
]
}
```

## 允许用户获取报告组的报告列表

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取报告组的报告列表：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>ListReportsForReportGroup",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

## 允许用户获取报告的测试用例的列表

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中获取报告的测试用例列表：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>DescribeTestCases",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
        }
    ]
}
```

## 允许用户开始运行构建

以下示例策略语句允许用户在 123456789012 账户的 us-east-2 区域中运行任何以名称 my 开头的构建项目：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codebuild>StartBuild",
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
        }
    ]
}
```

## 允许用户尝试停止构建

以下示例策略语句仅允许用户在 123456789012 账户的 us-east-2 区域中尝试停止任何以名称 my 开头的运行中构建项目：

```
{
```

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:StopBuild",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"  
        }  
    ]  
}
```

## 允许用户尝试删除构建

以下示例策略语句仅允许用户在 123456789012 账户的 us-east-2 区域中尝试为任何以名称 my 开头的构建项目删除构建：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "codebuild:BatchDeleteBuilds",  
            "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"  
        }  
    ]  
}
```

## 允许用户获取有关由 CodeBuild 管理的 Docker 映像的信息

以下示例策略语句允许用户获取有关由 CodeBuild 管理的所有 Docker 映像的信息：

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

## 允许 CodeBuild 访问创建 VPC 网络接口时所需的 AWS 服务

以下示例策略语句向 AWS CodeBuild 授予在一个包含两个子网的 VPC 中创建网络接口的权限：

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2>CreateNetworkInterface",  
                "ec2>DescribeDhcpOptions",  
                "ec2>DescribeNetworkInterfaces",  
                "ec2>DeleteNetworkInterface",  
                "ec2>DescribeSubnets",  
                "ec2>DescribeSecurityGroups",  
                "ec2>DescribeVpcs"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:Subnet": [
                        "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
                        "arn:aws:ec2:region:account-id:subnet/subnet-id-2"
                    ],
                    "ec2:AuthorizedService": "codebuild.amazonaws.com"
                }
            }
        }
    ]
}
```

## 使用 Deny 语句可阻止 AWS CodeBuild 与源提供程序断开连接

以下示例策略语句使用 Deny 语句阻止 AWS CodeBuild 与源提供程序断开连接。它使用 codebuild:DeleteOAuthToken ( codebuild:PersistOAuthToken 和 codebuild:ImportSourceCredentials 的倒数 ) 连接到源提供程序。有关更多信息，请参阅 [AWS CodeBuild 控制台连接到源提供程序所需的权限 \(p. 330\)](#)。)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "codebuild:DeleteOAuthToken",
            "Resource": "*"
        }
    ]
}
```

## AWS CodeBuild 权限参考

在设置[访问控制 \(p. 326\)](#)和编写您可附加到 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。

您可以在 AWS CodeBuild 策略中使用 AWS 范围的条件键来表示条件。有关列表，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_elements.html#AvailableKeys](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html#AvailableKeys) 中的 IAM 用户指南可用键。

请在策略的 Action 字段中指定这些操作。要指定操作，请在 API 操作名称之前使用 codebuild: 前缀（例如，codebuild:CreateProject 和 codebuild:StartBuild）。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”: [ “codebuild:CreateProject”，“codebuild:StartBuild” ]）。

### 使用通配符

您可以在策略的 Resource 字段中指定带或不带通配符 (\*) 的 ARN 作为资源值。您可以使用通配符指定多个操作或资源。例如，codebuild:\* 指定全部 CodeBuild 行动和 codebuild:Batch\* 指定全部 CodeBuild 从词语开始的操作 Batch... 以下示例授予访问所有创建项目的权限，其中的名称以 my:

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

## CodeBuild API 操作和必需的操作权限

### BatchDeleteBuilds

操作 : codebuild:BatchDeleteBuilds

删除构建所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

### BatchGetBuilds

操作 : codebuild:BatchGetBuilds

获取有关构建项目的信息所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

### BatchGetProjects

操作 : codebuild:BatchGetProjects

获取有关构建项目的信息所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

### BatchGetReportGroups

操作 : codebuild:BatchGetReportGroups

获取有关报告组的信息所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*

### BatchGetReports

操作 : codebuild:BatchGetReports

获取有关报告的信息所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*

### BatchPutTestCases<sup>1</sup>

操作 : codebuild:BatchPutTestCases

创建或更新测试报告所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*

### CreateProject

操作: codebuild:CreateProject , iam:PassRole

是创建构建项目所必需的。

资源 :

- arn:aws:codebuild:*region-ID:account-ID:project/project-name*

- arn:aws:iam:*account-ID:role/role-name*

### CreateReport<sup>1</sup>

操作 : codebuild:CreateReport

创建测试报告所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateReportGroup

操作 : codebuild:CreateReportGroup

创建报告组所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateWebhook

操作 : codebuild:CreateWebhook

创建 Webhook 所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteReport

操作 : codebuild:DeleteReport

删除报告所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteReportGroup

操作 : codebuild:DeleteReportGroup

删除报告组所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteSourceCredentials

操作 : codebuild:DeleteSourceCredentials

要求删除一组 SourceCredentialsInfo 对象，其中包含有关 GitHub、GitHub Enterprise Server 或 Bitbucket 存储库的凭证的信息。

资源: \*

DeleteWebhook

操作 : codebuild:DeleteWebhook

创建 Webhook 所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DescribeTestCases

操作 : codebuild:DescribeTestCases

返回测试用例的分页列表所必需的。

资源: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

ImportSourceCredentials

操作 : codebuild:ImportSourceCredentials

要求导入一组 SourceCredentialsInfo 对象，其中包含有关 GitHub、GitHub Enterprise Server 或 Bitbucket 存储库的凭证的信息。

资源: \*

### InvalidateProjectCache

操作 : codebuild:InvalidateProjectCache

是重置项目缓存所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

### ListBuilds

操作 : codebuild>ListBuilds

是获取构建 ID 的列表所必需的。

资源: \*

### ListBuildsForProject

操作 : codebuild>ListBuildsForProject

是获取构建项目的构建 ID 列表所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

### ListCuratedEnvironmentImages

操作 : codebuild>ListCuratedEnvironmentImages

是获取由 AWS CodeBuild 管理的所有 Docker 映像的相关信息所必需的。

Resource \* ( 必需 , 但不引用可寻址的 AWS 资源 )

### ListProjects

操作 : codebuild>ListProjects

是获取构建项目名称的列表所必需的。

资源: \*

### ListReportGroups

操作 : codebuild>ListReportGroups

获取报告组列表所必需的。

资源: \*

### ListReports

操作 : codebuild>ListReports

获取报告列表所必需的。

资源: \*

### ListReportsForReportGroup

操作 : codebuild>ListReportsForReportGroup

获取报告组的报告列表所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*

### 回缩构建

操作 : codebuild:RetryBuild

需要重试的必要内容。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*  
StartBuild

操作 : codebuild:StartBuild

是开始运行构建项目所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*  
StopBuild

操作 : codebuild:StopBuild

是尝试停止运行中构建项目所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*  
UpdateProject

操作: codebuild:UpdateProject , iam:PassRole

是更改构建项目的相关信息所必需的。

资源 :

- arn:aws:codebuild:*region-ID:account-ID:project/project-name*
- arn:aws:iam:*account-ID:role/role-name*

UpdateReport<sup>1</sup>

操作 : codebuild:UpdateReport

创建或更新测试报告所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*  
UpdateReportGroup

操作 : codebuild:UpdateReportGroup

更新报告组所必需的。

资源: arn:aws:codebuild:*region-ID:account-ID:report-group/report-group-name*  
UpdateWebhook

操作 : codebuild:UpdateWebhook

为更新 Webhook 所必需。

资源: arn:aws:codebuild:*region-ID:account-ID:project/project-name*

<sup>1</sup> 仅供许可。此操作没有API。

## 使用标签控制对 AWS CodeBuild 资源的访问

IAM 策略语句中的条件是语法的一部分，您可以使用该语法指定对基于 CodeBuild 项目操作的权限。您可以创建一个策略（该策略基于与项目关联的标签来允许或拒绝对这些项目执行操作），然后将该策略应用于为管理 IAM 用户而配置的 IAM 组。有关使用控制台或 AWS CLI 将标签应用于项目的信息，请参阅 [在 AWS CodeBuild 中创建构建项目 \(p. 189\)](#)。有关使用 CodeBuild SDK，请参阅 [创建项目](#) 和 [标签](#) 在 CodeBuild

API 参考。有关使用标签控制访问权限的信息 AWS 资源，请参阅 [控制访问 AWS 资源标签资源 在 IAM 用户指南](#)。

### Example 示例 #1 限制 CodeBuild 基于资源标签的项目操作

以下示例拒绝所有 BatchGetProjects 标记关键项目的项目的操作 Environment 的关键价值 Production...用户的管理员必须附加此信息 IAM 除非已管理用户政策以外，还有权限政策未经授权 IAM 用户。aws:ResourceTag 条件键用于基于其标签控制对资源的访问。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": [  
                "codebuild:BatchGetProjects"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "ForAnyValue:StringEquals": {  
                    "aws:ResourceTag/Environment": "Production"  
                }  
            }  
        }  
    ]  
}
```

### Example 示例 2 限制 CodeBuild 根据请求标签的项目操作

以下策略拒绝用户授权访问 CreateProject 如果请求包含标签的密钥中有一个标签 Environment 关键价值 Production...此外，该策略还使得这些未授权用户使用 aws:TagKeys 不允许的条件键 UpdateProject 如果请求包含标签的密钥 Environment...管理员必须附加此信息 IAM 除非已授权执行这些操作的用户之外，还要对管理用户策略加以策略。aws:RequestTag 条件键用于控制可以通过 IAM 请求传递哪些标签

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": [  
                "codebuild>CreateProject"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "ForAnyValue:StringEquals": {  
                    "aws:RequestTag/Environment": "Production"  
                }  
            }  
        },  
        {  
            "Effect": "Deny",  
            "Action": [  
                "codebuild:UpdateProject"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "ForAnyValue:StringEquals": {  
                    "aws:TagKeys": ["Environment"]  
                }  
            }  
        }  
    ]  
}
```

```
    ]  
}
```

### Example 示例 3 根据资源标签拒绝或允许对报告组执行操作

您可以创建一个基于与 CodeBuild 资源关联的 AWS 标签来允许或拒绝对这些资源（项目和报告组）执行操作的策略，然后将该策略应用于为管理 IAM 用户而配置的 IAM 组。例如，您可以创建一个拒绝所有策略的策略 CodeBuild 任何报告组的操作 AWS 标签键 *Status* 以及关键值 *Secret*，然后将该策略应用到您为一般开发人员创建的 IAM 组（*Developers*）。然后您需要确保在标签的报告组上工作的开发人员不是该常规的成员 *Developers* 但是属于不同的组别 IAM 未应用限制性政策的团体（*SecretDevelopers*）。

以下示例拒绝对用键 *Status* 和键值 *Secret* 标记的报告组执行所有 CodeBuild 操作：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "codebuild:BatchGetReportGroups",
        "codebuild>CreateReportGroup",
        "codebuild>DeleteReportGroup",
        "codebuild>ListReportGroups",
        "codebuild>ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
      ]
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : "aws:ResourceTag/Status": "Secret"
      }
    }
  ]
}
```

### Example 示例 4 限制 CodeBuild AWS 的操作 CodeBuild 根据资源标签开发人员访问

您可以创建策略以允许对未使用特定标签标记的所有报告组和项目执行 CodeBuild 操作。例如，以下策略为除了使用指定标签标记的报告组和项目以外的所有其他报告组和项目提供与 [AWSCodeBuildDeveloperAccess \(p. 332\)](#) 等效的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:DescribeTestCases",
        "codebuild>List*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:getRepository",
        "codecommit>ListBranches",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events>ListTargetsByRule",
        "events>ListRuleNamesByTarget",
        "logs:GetLogEvents",
        "lambda:InvokeFunction"
      ]
    }
  ]
}
```

```
    "s3:GetBucketLocation",
    "s3>ListAllMyBuckets"
],
"Resource": "*",
"Condition": {
    "StringNotEquals": {
        "aws:ResourceTag/Status": "Secret",
        "aws:ResourceTag/Team": "Saanvi"
    }
}
]
```

## 在控制台中查看资源

AWS CodeBuild 控制台需要 `ListRepositories` 权限，以显示您的 AWS 账户在所登录的 AWS 区域中的存储库列表。该控制台还包括一个 `Go to resource` (转到资源) 功能，可对资源快速执行不区分大小写的搜索。此搜索通过您的 AWS 账户，在您登录的 AWS 区域中执行。将显示以下服务中的以下资源：

- AWS CodeBuild：构建项目
- AWS CodeCommit：存储库
- AWS CodeDeploy：应用程序
- AWS CodePipeline：管道

要在所有服务中跨资源执行此搜索，您必须具有如下权限：

- `CodeBuild:ListProjects`
- `CodeCommit:ListRepositories`
- `CodeDeploy:ListApplications`
- `CodePipeline:ListPipelines`

如果您没有针对某个服务的权限，搜索将不会针对该服务的资源返回结果。即使您有权查看资源，但如果特定资源明确 Deny 查看，也不会返回这些资源。

## AWS CodeBuild 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审核员将评估 AWS CodeBuild 的安全性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规性计划范围内的 AWS 服务的列表，请参阅[合规性计划范围内的 AWS 服务](#)。有关一般信息，请参阅[AWS 合规性计划](#)。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅[在 AWS Artifact 中下载报告](#)。

您在使用 CodeBuild 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。如果您对 CodeBuild 的使用需遵守 HIPAA、PCI 或 FedRAMP 等标准，AWS 将提供以下有用资源：

- [安全性与合规性快速入门指南](#) – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- [《设计符合 HIPAA 安全性和合规性要求的架构》白皮书](#) – 此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- [AWS 合规性资源](#) – 此业务手册和指南集合可能适用于您的行业和位置。

- [AWS Config](#) – 此 AWS 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- [AWS Security Hub](#) – 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

## AWS CodeBuild 中的弹性

AWS 全球基础设施围绕 AWS 区域和可用区构建。AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 [AWS 全球基础设施](#)。

## AWS CodeBuild 中的基础设施安全性

作为一项托管服务，AWS CodeBuild 由 [Amazon Web Services：安全流程概览](#) 白皮书中所述的 AWS 全球网络安全程序提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 CodeBuild。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

必须使用访问密钥 ID 以及与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 [AWS Security Token Service](#) (AWS STS) 生成临时安全凭证来对请求进行签名。

# 高级主题

此部分包含几个高级主题，这些主题对于经验更丰富的 AWS CodeBuild 用户很有用。

## 主题

- [高级设置 \(p. 353\)](#)
- [AWS CodeBuild 的命令行参考 \(p. 363\)](#)
- [适用于 AWS CodeBuild 的 AWS 开发工具包和工具参考 \(p. 364\)](#)
- [指定 AWS CodeBuild 终端节点 \(p. 365\)](#)
- [直接运行 AWS CodeBuild \(p. 367\)](#)
- [将 AWS CodePipeline 与 AWS CodeBuild 结合使用以测试代码和运行构建 \(p. 367\)](#)
- [将 AWS CodeBuild 与 Jenkins 结合使用 \(p. 381\)](#)
- [将 AWS CodeBuild 与 codecov 结合使用 \(p. 382\)](#)
- [使用 AWS CodeBuild 无服务器应用 \(p. 385\)](#)

# 高级设置

如果您首次按照[通过控制台开始使用 \(p. 4\)](#)中的步骤操作来访问 AWS CodeBuild，则很可能不需要本主题中的信息。但是，随着您继续使用 CodeBuild，您可能需要执行这些操作：例如，为您组织内的 IAM 组和用户提供对 CodeBuild 的访问权限、修改 IAM 中的现有服务角色或 AWS KMS 中的客户主密钥以访问 CodeBuild，或者跨您组织的工作站设置 AWS CLI 以访问 CodeBuild。本主题将介绍如何完成相关的设置步骤。

我们假定您已经有一个 AWS 账户。但是，如果您还没有账户，请转到 <http://aws.amazon.com>，选择 Sign In to the Console（登录到控制台），然后按照在线说明进行操作。

## 主题

- [向 IAM 组或 IAM 用户添加 CodeBuild 访问权限 \(p. 353\)](#)
- [创建 CodeBuild 服务角色 \(p. 357\)](#)
- [为 CodeBuild 创建和配置 AWS KMS CMK \(p. 361\)](#)
- [安装和配置 AWS CLI \(p. 363\)](#)

## 向 IAM 组或 IAM 用户添加 CodeBuild 访问权限

要以 IAM 组或 IAM 用户身份访问 AWS CodeBuild，您必须添加访问权限。本节介绍了如何使用 IAM 控制台或 AWS CLI 完成此操作。

如果以 AWS 根账户（不推荐）或 AWS 账户中的管理员 IAM 用户身份访问 CodeBuild，则无需遵循这些说明。

有关 AWS 根账户和管理员 IAM 用户的信息，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_root-user.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html) 用户指南中的IAM账户根用户和创建您的第一个 IAM 管理员用户和组。

### 为 IAM 组或 IAM 用户添加 CodeBuild 访问权限（控制台）

1. 通过以下网址打开 IAM 控制台：[https://console.aws.amazon.com/iam/。](https://console.aws.amazon.com/iam/)

您应该已使用以下任一身份登录到 AWS 管理控制台：

- 您的 AWS 根账户。我们不建议这么做。有关更多信息，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_root-user.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html) 用户指南中的IAM账户根用户。

- AWS 账户中的 IAM 管理员用户。有关更多信息，请参阅 [IAM 用户指南](#) 中的创建您的第一个 IAM 管理员用户和组。
- AWS 账户中的 IAM 用户，具有执行以下最基本操作的权限：

```
iam:AttachGroupPolicy
iam:AttachUserPolicy
iam:CreatePolicy
iam>ListAttachedGroupPolicies
iam>ListAttachedUserPolicies
iam>ListGroups
iam>ListPolicies
iam>ListUsers
```

有关更多信息，请参阅 [IAM 用户指南](#) 中的 IAM 策略概述。

2. 在导航窗格中，选择策略。
3. 要为 IAM 组或 IAM 用户添加一组自定义的 AWS CodeBuild 访问权限，请向前跳到此过程的第 4 步。

要向 IAM 组或 IAM 用户添加一组默认的 CodeBuild 访问权限，请依次选择 Policy Type (策略类型) 和 AWS Managed (AWS 托管)，然后执行以下操作：

- 要添加 CodeBuild 的完全访问权限，请选中名为 AWSCodeBuildAdminAccess 的框，选择 Policy Actions (策略操作)，然后选择 Attach (附加)。选中目标 IAM 组或 IAM 用户旁的框，然后选择 Attach Policy (附加策略)。对名为 AmazonS3ReadOnlyAccess 和 IAMFullAccess 的策略重复执行此操作。
- 要为除构建项目管理之外的所有内容添加对 CodeBuild 的访问权限，请选中名为 AWSCodeBuildDeveloperAccess 的框，然后依次选择 Policy Actions (策略操作) 和 Attach (附加)。选中目标 IAM 组或 IAM 用户旁的框，然后选择 Attach Policy (附加策略)。对名为 AmazonS3ReadOnlyAccess 的策略重复执行此操作。
- 要添加对 CodeBuild 的只读访问权限，请选中名为 AWSCodeBuildReadOnlyAccess 的框。选中目标 IAM 组或 IAM 用户旁的框，然后选择 Attach Policy (附加策略)。对名为 AmazonS3ReadOnlyAccess 的策略重复执行此操作。

现在，您已经为 IAM 组或 IAM 用户添加了一组默认的 CodeBuild 访问权限。跳过此过程中的其余步骤。

4. 选择创建策略。
5. 在 Create Policy 页面上的 Create Your Own Policy 旁，选择 Select。
6. 在 Review Policy (审查策略) 页面上，为 Policy Name (策略名称) 输入策略的名称（例如，**CodeBuildAccessPolicy**）。如果您使用其他名称，请确保在本过程中始终使用它。
7. 对于 Policy Document (策略文档)，输入以下内容，然后选择 Create Policy (创建策略)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildDefaultPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "iam:PassRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

```
        "logs:GetLogEvents"
    ],
    "Resource": "*"
},
{
    "Sid": "S3AccessPolicy",
    "Effect": "Allow",
    "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3>List*",
        "s3:PutObject"
    ],
    "Resource": "*"
},
{
    "Sid": "S3BucketIdentity",
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
    ],
    "Resource": "*"
}
]
```

#### Note

此策略允许访问所有 CodeBuild 操作以及潜在的大量 AWS 资源。要限制对特定 CodeBuild 操作的访问权限，请在 CodeBuild 策略声明中更改 `codebuild:*` 的值。有关更多信息，请参阅 [Identity and Access Management \(p. 325\)](#)。) 要限制对特定 AWS 资源的访问权限，请更改 `Resource` 对象的值。有关更多信息，请参阅 [Identity and Access Management \(p. 325\)](#)。)

8. 在导航窗格中，选择 Groups 或 Users。
9. 在组或用户列表中，选择要向其添加 CodeBuild 访问权限的 IAM 组或 IAM 用户的名称。
10. 对于组，在组设置页面上的 Permissions (权限) 选项卡上，展开 Managed Policies (托管策略)，然后选择 Attach Policy (附加策略)。

对于用户，在用户设置页面上的 Permissions 选项卡上，选择 Add permissions。

11. 对于组，在 Attach Policy (附加策略) 页面上，选择 CodeBuildAccessPolicy，然后选择 Attach Policy (附加策略)。

对于用户，在 Add permissions 页面上，选择 Attach existing policies directly。选择 CodeBuild 登录冻结，选择下一步：雷韦夫，然后选择 添加权限。

#### 为 IAM 组或 IAM 用户 (AWS CLI) 添加 CodeBuild 访问权限

1. 如前面的过程所述，确保您已为 AWS CLI 配置了与某个 IAM 实体相对应的 AWS 访问密钥和 AWS 秘密访问密钥。有关更多信息，请参阅 [AWS Command Line Interface 用户指南](#) 中的开始设置 AWS Command Line Interface。
2. 要为 IAM 组或 IAM 用户添加一组自定义的 AWS CodeBuild 访问权限，请跳至本过程中的步骤 3。

要为 IAM 组或 IAM 用户添加一组默认的 CodeBuild 访问权限，请执行以下操作：

运行以下任一命令，具体取决于您是否要为 IAM 组或 IAM 用户添加权限：

```
aws iam attach-group-policy --group-name group-name --policy-arn policy-arn
```

```
aws iam attach-user-policy --user-name user-name --policy-arn policy-arn
```

您必须运行命令三次，更换 *group-name* 或 *user-name* 与 IAM 组名称或 IAM 用户名和更换 *policy-arn* 对于以下每个策略亚马逊资源名称(ARNS)，一次：

- 要添加对 CodeBuild 的完全访问权限，请使用以下策略 ARN：
  - arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess
  - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
  - arn:aws:iam::aws:policy/IAMFullAccess
- 要添加除构建项目管理以外的所有 CodeBuild 访问权限，请使用以下策略 ARN：
  - arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess
  - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
- 要添加对 CodeBuild 的只读访问权限，请使用以下策略 ARN：
  - arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess
  - arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess

现在，您已经为 IAM 组或 IAM 用户添加了一组默认的 CodeBuild 访问权限。跳过此过程中的其余步骤。

3. 在本地工作站上的空目录中，或者在其中的实例中 AWS CLI 已安装，创建名为 put-group-policy.json 或 put-user-policy.json...如果您使用的文件名不同，请确保在整个过程中使用其中的文件名。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "iam:PassRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3>CreateBucket",
        "s3:GetObject",
        "s3>List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ]
    }
  ]
}
```

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

Note

此策略允许访问所有 CodeBuild 操作以及潜在的大量 AWS 资源。要限制对特定 CodeBuild 操作的访问权限，请在 CodeBuild 策略声明中更改 `codebuild:*` 的值。有关更多信息，请参阅 [Identity and Access Management \(p. 325\)](#)。要限制对特定 AWS 资源的访问权限，请更改相关 `Resource` 对象的值。有关更多信息，请参阅 [Identity and Access Management \(p. 325\)](#) 或特定 AWS 服务的安全文档。

4. 切换到您保存该文件的目录，然后运行以下任一命令。您可以使用不同的值 `CodeBuildGroupAccessPolicy` 和 `CodeBuildUserAccessPolicy`...如果您使用不同的值，请确保在此处使用。

对于 IAM 组：

```
aws iam put-group-policy --group-name group-name --policy-name
CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
```

对于 IAM 用户：

```
aws iam put-user-policy --user-name user-name --policy-name CodeBuildUserAccessPolicy
--policy-document file://put-user-policy.json
```

在前面的命令中，更换 `group-name` 或 `user-name` 目标名称 IAM 组或 IAM 用户。

## 创建 CodeBuild 服务角色

您需要一个 AWS CodeBuild 服务角色，以便 CodeBuild 能代表您与相关 AWS 服务进行交互。您可以使用 CodeBuild 或 AWS CodePipeline 控制台创建一个 CodeBuild 服务角色。有关信息，请参阅。

- [创建构建项目（控制台）\(p. 190\)](#)
- [创建使用 CodeBuild 的管道（CodePipeline 控制台）\(p. 369\)](#)
- [将 CodeBuild 构建操作添加到管道（CodePipeline 控制台）\(p. 374\)](#)
- [更改构建项目的设置（控制台）\(p. 236\)](#)

如果您不打算使用这些控制台，本节介绍了如何使用 IAM 控制台或 AWS CLI 创建 CodeBuild 服务角色。

Note

此页上描述的服务角色包含一项策略，可授予使用 CodeBuild 时所需的最低权限。您可能需要根据使用案例添加额外的权限。例如，如果您希望将 CodeBuild 与 Amazon Virtual Private Cloud 配合使用，则您创建的服务角色需要以下策略中的权限：[创建 CodeBuild 服务角色 \(p. 357\)](#)。

### 创建 CodeBuild 服务角色（控制台）

1. 通过以下网址打开 IAM 控制台：[https://console.aws.amazon.com/iam/。](https://console.aws.amazon.com/iam/)

您应该已使用以下任一身份登录到控制台：

- 您的 AWS 根账户。我们不建议这么做。有关更多信息，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_root-user.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html) 用户指南中的 IAM 账户根用户。
- AWS 账户中的 IAM 管理员用户。有关更多信息，请参阅 [IAM 用户指南](#) 中的创建您的第一个 IAM 管理员用户和组。
- AWS 账户中的 IAM 用户，具有执行以下最基本操作的权限：

```
iam:AddRoleToInstanceProfile
iam:AttachRolePolicy
iam>CreateInstanceProfile
iam>CreatePolicy
iam>CreateRole
iam:GetRole
iam>ListAttachedRolePolicies
iam>ListPolicies
iam>ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy
```

有关更多信息，请参阅 [IAM 用户指南](#) 中的 IAM 策略概述。

2. 在导航窗格中，选择策略。
3. 选择创建策略。
4. 在 Create Policy 页面上，选择 JSON。
5. 对于 JSON 策略，输入以下内容，然后选择 Review Policy (查看策略)：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs>CreateLogGroup",
        "logs>CreateLogStream",
        "logs>PutLogEvents"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

```
},
{
  "Sid": "S3PutObjectPolicy",
  "Effect": "Allow",
  "Action": [
    "s3:PutObject"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Sid": "ECRPullPolicy",
  "Effect": "Allow",
  "Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Sid": "ECRAuthPolicy",
  "Effect": "Allow",
  "Action": [
    "ecr:GetAuthorizationToken"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Sid": "S3BucketIdentity",
  "Effect": "Allow",
  "Action": [
    "s3:GetBucketAcl",
    "s3:GetBucketLocation"
  ],
  "Resource":
    "*"
}
]
```

#### Note

此策略包含允许访问潜在的大量 AWS 资源的语句。要限制 AWS CodeBuild 访问特定的 AWS 资源，请更改 Resource 数组的值。有关更多信息，请参阅有关 AWS 服务的安全文档。

6. 在 Review Policy (查看策略) 页面上，对于 Policy Name (策略名称)，为策略输入一个名称（例如，**CodeBuildServiceRolePolicy**），然后选择 Create policy (创建策略)。

#### Note

如果您使用其他名称，请确保在本过程中始终使用它。

7. 在导航窗格中，选择角色。
8. 选择创建角色。
9. 在 Create role (创建角色) 页面上，在已选择 AWS Service (AWS 服务) 的情况下，选择 CodeBuild，然后选择 Next:Permissions。
10. 在附加权限策略页面，选择 CodeBuild 服务政策，然后选择下一步：审核

- 在 Create role and review (创建角色并审查) 页面上，对于 Role name (角色名称)，输入角色的名称（例如，**CodeBuildServiceRole**），然后选择 Create role (创建角色)。

### 创建 CodeBuild 服务角色 (AWS CLI)

- 如前面的过程所述，确保您已为 AWS CLI 配置了与某个 IAM 实体相对应的 AWS 访问密钥和 AWS 秘密访问密钥。有关更多信息，请参阅 [AWS Command Line Interface 用户指南](#) 中的开始设置 AWS Command Line Interface。
- 在本地工作站上的空目录中，或者在其中的实例中 AWS CLI 已安装，创建两个名为 `create-role.json` 和 `put-role-policy.json`...如果您选择不同的文件名，请确保在整个过程中使用这些文件。

`create-role.json:`

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

`put-role-policy.json:`

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "CloudWatchLogsPolicy",  
            "Effect": "Allow",  
            "Action": [  
                "logs>CreateLogGroup",  
                "logs>CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": [  
                "*"  
            ]  
        },  
        {  
            "Sid": "CodeCommitPolicy",  
            "Effect": "Allow",  
            "Action": [  
                "codecommit:GitPull"  
            ],  
            "Resource": [  
                "*"  
            ]  
        },  
        {  
            "Sid": "S3GetObjectPolicy",  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObject",  
                "s3:GetObjectVersion"  
            ],  
            "Resource": "  
        }  
    ]  
}
```

```
"Resource": [
    "*"
],
},
{
    "Sid": "S3PutObjectPolicy",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Sid": "S3BucketIdentity",
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
    ],
    "Resource": [
        "*"
    ]
}
]
```

#### Note

此策略包含允许访问潜在的大量 AWS 资源的语句。要限制 AWS CodeBuild 访问特定的 AWS 资源，请更改 Resource 数组的值。有关更多信息，请参阅有关 AWS 服务的安全文档。

3. 切换到您保存上述文件的目录，然后按照这个顺序运行以下两个命令，一次运行一个。您可以为 CodeBuildServiceRole 和 CodeBuildServiceRolePolicy 使用不同的值，但请务必在此处使用它们。

```
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document file://create-role.json
```

```
aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json
```

## 为 CodeBuild 创建和配置 AWS KMS CMK

要使 AWS CodeBuild 加密其构建输出构件，需要具备对 AWS KMS 客户主密钥 (CMK) 的访问权限。默认情况下，CodeBuild 使用您 AWS 账户中适用于 Amazon S3 的 AWS 托管的 CMK。

如果您不想使用此 CMK，则必须自行创建并配置一个客户托管的 CMK。本部分介绍了如何通过 IAM 控制台执行此操作。

有关 CMK 的信息，请参阅 [AWS Key Management Service 开发人员指南](https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html) 中的 <https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html> 概念和 AWS KMS 创建密钥。

要配置由 CodeBuild 使用的 CMK，请遵循 <https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html> 开发人员指南中 AWS KMS 修改密钥策略的“如何修改密钥策略”部分中的说明。然后添加以下陈述 **### BEGIN ADDING STATEMENTS HERE ###** \$and **### END ADDING STATEMENTS HERE ###** 至关键政策。为了简洁起见，也为了帮您查找添加语句的位置，此处使用了省略号 (...)。请勿删除任何语句，也不要将这些省略号键入密钥策略中。

```
{  
    "Version": "2012-10-17",  
    "Id": "...",  
    "Statement": [  
        ### BEGIN ADDING STATEMENTS HERE ###  
        {  
            "Sid": "Allow access through Amazon S3 for all principals in the account that are authorized to use Amazon S3",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": [  
                "kms:Encrypt",  
                "kms:Decrypt",  
                "kms:ReEncrypt*",  
                "kms:GenerateDataKey*",  
                "kms:DescribeKey"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "kms:ViaService": "s3.region-ID.amazonaws.com",  
                    "kms:CallerAccount": "account-ID"  
                }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::account-ID:role/CodeBuild-service-role"  
            },  
            "Action": [  
                "kms:Encrypt",  
                "kms:Decrypt",  
                "kms:ReEncrypt*",  
                "kms:GenerateDataKey*",  
                "kms:DescribeKey"  
            ],  
            "Resource": "*"  
        },  
        ### END ADDING STATEMENTS HERE ###  
    ]  
}
```

- **region-ID** 代表该ID的ID AWS 地区 Amazon S3 与 CodeBuild 位置（例如，us-east-1）。
- **account-ID** 代表的 AWS 拥有CMK的帐户。

- *CodeBuild-service-role* 代表该名称 CodeBuild 在本主题之前创建或确定的服务角色。

#### Note

要通过 IAM 控制台创建或配置 CMK，您必须先使用以下任一身份登录该 AWS 管理控制台：

- 您的 AWS 根账户。我们不建议这么做。有关更多信息，请参阅 [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_root-user.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html) 用户指南中的 IAM 账户根用户。
- AWS 账户中的 IAM 管理员用户。有关更多信息，请参阅 [IAM 用户指南](#) 中的创建您的第一个 IAM 管理员用户和组。
- AWS 账户中具有创建或修改 CMK 权限的 IAM 用户。有关更多信息，请参阅 [AWS KMS 开发人员指南](#) 中的使用 AWS KMS 控制台所需的权限。

## 安装和配置 AWS CLI

要访问 AWS CodeBuild，您可以将 AWS CLI 与 CodeBuild 控制台、CodePipeline 控制台或 AWS 开发工具包结合使用，或者改为仅使用前者。要安装和配置 AWS CLI，请参阅 [AWS Command Line Interface 用户指南](#) 中的开始设置 AWS Command Line Interface。

1. 运行以下命令以确认您安装的 AWS CLI 是否支持 CodeBuild：

```
aws codebuild list-builds
```

如果成功，将在输出中显示与以下内容类似的信息：

```
{  
  "ids": []  
}
```

空方括号表示您尚未运行任何构建。

2. 如果输出一个错误，您必须卸载当前版本的 AWS CLI，然后安装最新版本。有关更多信息，请参阅 [AWS CLI 用户指南](#) 中的 [卸载 AWS Command Line Interface](#) 和 [安装 AWS Command Line Interface](#)。

## AWS CodeBuild 的命令行参考

AWS CLI 可提供用于自动化 AWS CodeBuild 的命令。使用本主题中的信息作为 [AWS Command Line Interface 用户指南](#) 和 [AWS CLI Reference for AWS CodeBuild](#) 的补充。

不是您要找的内容？如果要使用 AWS 开发工具包调用 CodeBuild，请参阅 [AWS 开发工具包和工具参考 \(p. 364\)](#)。

要使用本主题中的信息，您应该已经安装了 AWS CLI，并已按照 [安装和配置 AWS CLI \(p. 363\)](#) 中的说明将其配置为与 CodeBuild 配合使用。

要使用 AWS CLI 指定 CodeBuild 的终端节点，请参阅 [指定 AWS CodeBuild 终端节点 \(AWS CLI\) \(p. 365\)](#)。

运行此命令可获取 CodeBuild 命令的列表。

```
aws codebuild help
```

运行此命令可获取有关 CodeBuild 命令的信息，其中 *command-name* 是命令的名称。

```
aws codebuild command-name help
```

CodeBuild 命令包括：

- `batch-delete-builds`：删除 CodeBuild 中的一个或多个构建。有关更多信息，请参阅[删除构建 \(AWS CLI\) \(p. 274\)](#)。
- `batch-get-builds`：获取有关 CodeBuild 中的多个构建的信息。有关更多信息，请参阅[查看构建详细信息 \(AWS CLI\) \(p. 262\)](#)。
- `batch-get-projects`：获取有关一个或多个指定构建项目的信息。有关更多信息，请参阅[查看构建项目的详细信息 \(AWS CLI\) \(p. 213\)](#)。
- `create-project`：创建构建项目。有关更多信息，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。
- `delete-project`：删除构建项目。有关更多信息，请参阅[删除构建项目 \(AWS CLI\) \(p. 244\)](#)。
- `list-builds`：列出 CodeBuild 中的构建的 Amazon 资源名称 (ARN)。有关更多信息，请参阅[查看构建 ID 的列表 \(AWS CLI\) \(p. 264\)](#)。
- `list-builds-for-project`：获取与指定构建项目相关联的构建 ID 的列表。有关更多信息，请参阅[查看构建项目的构建 ID 列表 \(AWS CLI\) \(p. 266\)](#)。
- `list-curated-environment-images`：获取由 CodeBuild 管理的可用于构建的 Docker 映像的列表。有关更多信息，请参阅[CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。
- `list-projects`：获取构建项目名称的列表。有关更多信息，请参阅[查看构建项目名称的列表 \(AWS CLI\) \(p. 212\)](#)。
- `start-build`：开始运行构建。有关更多信息，请参阅[运行构建 \(AWS CLI\) \(p. 256\)](#)。
- `stop-build`：尝试停止运行指定的构建。有关更多信息，请参阅[停止构建 \(AWS CLI\) \(p. 269\)](#)。
- `update-project`：更改指定构建项目的信息。有关更多信息，请参阅[更改构建项目的设置 \(AWS CLI\) \(p. 243\)](#)。

## 适用于 AWS CodeBuild 的 AWS 开发工具包和工具参考

要使用一种 AWS 开发工具包或工具来自动化 AWS CodeBuild，请参阅以下资源。

如果要使用 AWS CLI 运行 CodeBuild，请参阅[命令行参考 \(p. 363\)](#)。

## 适用于 AWS CodeBuild 的受支持的 AWS 开发工具包和工具

以下 AWS 开发工具包和工具支持 CodeBuild：

- [适用于 C++ 的 AWS 开发工具包](#)。有关更多信息，请参阅适用于 C++ 的 AWS 开发工具包 API 参考的 `Aws::CodeBuild` 命名空间部分。
- [适用于 Go 的 AWS 开发工具包](#)。有关更多信息，请参阅适用于 Go 的 AWS 开发工具包 API 参考的 `codebuild` 部分。
- [适用于 Java 的 AWS 开发工具包](#)。有关更多信息，请参阅[适用于 Java 的 AWS 开发工具包 API 参考](#)的 `com.amazonaws.services.codebuild` 和 `com.amazonaws.services.codebuild.model` 部分。
- [适用于浏览器中 JavaScript 的 AWS 开发工具包](#)和[适用于 Node.js 中 JavaScript 的 AWS 开发工具包](#)。有关更多信息，请参阅适用于 JavaScript 的 AWS 开发工具包 API 参考的 `Class: AWS.CodeBuild` 部分。
- [适用于 .NET 的 AWS 开发工具包](#)。有关更多信息，请参阅适用于 .NET 的 AWS 开发工具包 API 参考的 `Amazon.CodeBuild` 和 `Amazon.CodeBuild.Model` 命名空间部分。

- 适用于 PHP 的 AWS 开发工具包。有关更多信息，请参阅适用于 PHP 的 AWS 开发工具包 API 参考的 Namespace Aws\CodeBuild 部分。
- 适用于 Python 的 AWS 开发工具包 (Boto3)。有关更多信息，请参阅 Boto 3 文档的 CodeBuild 部分。
- 适用于 Ruby 的 AWS 开发工具包。有关更多信息，请参阅适用于 Ruby 的 AWS 开发工具包 API 参考的 Module: Aws::CodeBuild 部分。
- 适用于 PowerShell 的 AWS 工具。有关更多信息，请参阅适用于 PowerShell 的 AWS 工具 Cmdlet 参考的 AWS CodeBuild 部分。

## 指定 AWS CodeBuild 终端节点

您可以使用 AWS Command Line Interface (AWS CLI) 或 AWS 开发工具包之一指定由 AWS CodeBuild 使用的终端节点。CodeBuild 可用的每个区域都有一个终端节点。除了一个区域终端节点之外，四个区域还有联邦信息处理标准 (FIPS) 终端节点。有关 FIPS 终端节点的更多信息，请参阅 [FIPS 140-2 概述](#)。

可以选择指定终端节点。如果您未明确告知 CodeBuild 要使用哪个终端节点，该服务将使用与您的 AWS 账户所用区域关联的终端节点。CodeBuild 从不默认使用 FIPS 终端节点。如果您希望使用 FIPS 终端节点，则必须使用以下方法之一将 CodeBuild 与其关联。

### Note

您可以使用 AWS 开发工具包，通过别名或区域名称指定终端节点。如果使用的是 AWS CLI，则您必须使用终端节点的完整名称。

有关可用于 CodeBuild 的终端节点，请参阅 [CodeBuild 区域和终端节点](#)。

### 主题

- [指定 AWS CodeBuild 终端节点 \(AWS CLI\) \(p. 365\)](#)
- [指定 AWS CodeBuild 终端节点 \(AWS 开发工具包\) \(p. 365\)](#)

## 指定 AWS CodeBuild 终端节点 (AWS CLI)

您可以在任何 CodeBuild 命令中使用 --endpoint-url 参数，通过 AWS CLI 指定访问 AWS CodeBuild 时所用的终端节点。例如，运行此命令以获取在美国东部（弗吉尼亚北部）地区中使用联邦信息处理标准 (FIPS) 终端节点的项目生成名称列表：

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-east-1.amazonaws.com
```

在终端节点的开头包括 https://。

--endpoint-url AWS CLI 参数可供所有 AWS 服务使用。有关此参数和其他 AWS CLI 参数的更多信息，请参阅 [AWS CLI Command Reference](#)。

## 指定 AWS CodeBuild 终端节点 (AWS 开发工具包)

您可以使用 AWS 开发工具包指定访问 AWS CodeBuild 时使用的终端节点。尽管此示例使用 [适用于 Java 的 AWS 开发工具包](#)，但是您可以指定具有其他 AWS 开发工具包的终端节点。

构造 AWSCodeBuild 客户端时使用 withEndpointConfiguration 方法。下面是使用的格式：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().  
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",  
    "region")).  
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
    build();
```

有关 `AWSCodeBuildClientBuilder` 的信息，请参阅类 [AWSCodeBuildClientBuilder](#)。

在 `withCredentials` 中使用的凭证的类型必须为 `AWSCredentialsProvider`。有关更多信息，请参阅使用 [AWS 凭证](#)。

不要在终端节点的开头包括 `https://`。

如果您希望指定非 FIPS 终端节点，则可以使用区域而非实际终端节点。例如，要在美国东部（弗吉尼亚北部）区域中指定终端节点，您可以使用 `us-east-1` 而不是完整的终端节点名称 `codebuild.us-east-1.amazonaws.com`。

如果您要指定 FIPS 终端节点，可以使用别名来简化代码。只有 FIPS 终端节点有别名。其他终端节点必须使用其区域或完整名称指定。

下表列出了四个可用 FIPS 终端节点的各自的别名。

区域名称	区域	终端节点	别名
美国东部 ( 弗吉尼亚 北部 )	us-east-1	<code>codebuild-fips.us-east-1.amazonaws.com</code>	<code>us-east-1- fips</code>
美国东部 ( 俄亥俄 州 )	us-east-2	<code>codebuild-fips.us-east-2.amazonaws.com</code>	<code>us-east-2- fips</code>
美国西部 ( 加利福尼 亚北部 )	us-west-1	<code>codebuild-fips.us-west-1.amazonaws.com</code>	<code>us-west-1- fips</code>
美国西部 ( 俄勒冈 )	us-west-2	<code>codebuild-fips.us-west-2.amazonaws.com</code>	<code>us-west-2- fips</code>

要指定使用美国西部（俄勒冈）区域中的 FIPS 终端节点，请使用别名：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().  
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-fips",  
    "us-west-2")).  
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
    build();
```

指定使用美国东部（弗吉尼亚北部）区域中的非 FIPS 终端节点：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().  
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1", "us-  
east-1")).  
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
    build();
```

指定使用亚太地区（孟买）区域中的非 FIPS 终端节点：

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
```

```
withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1", "ap-south-1")).  
withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).  
build();
```

## 直接运行 AWS CodeBuild

要直接借助 CodeBuild 设置、运行和监控构建，您可以使用 AWS CodeBuild 控制台、AWS CLI 或 AWS 开发工具包。

不是您要找的内容？要使用 AWS CodePipeline 运行 CodeBuild，请参阅[将 CodePipeline 与 CodeBuild 结合使用 \(p. 367\)](#)。

### 主题

- [Prerequisites \(p. 367\)](#)
- [直接运行 AWS CodeBuild \(p. 367\)](#)

## Prerequisites

回答[计划构建 \(p. 135\)](#)中的问题。

## 直接运行 AWS CodeBuild

1. 创建构建项目 要使用控制台，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)。要使用 AWS CLI，请参阅[创建构建项目 \(AWS CLI\) \(p. 198\)](#)。
2. 运行构建。要使用控制台，请参阅[运行构建 \( 控制台 \) \(p. 252\)](#)。要使用 AWS CLI，请参阅[运行构建 \(AWS CLI\) \(p. 256\)](#)。
3. 获取有关构建的信息。要使用控制台，请参阅[查看构建详细信息 \( 控制台 \) \(p. 262\)](#)。要使用 AWS CLI，请参阅[查看构建详细信息 \(AWS CLI\) \(p. 262\)](#)。

## 将 AWS CodePipeline 与 AWS CodeBuild 结合使用以测试代码和运行构建

通过使用 AWS CodePipeline 测试您的代码并借助 AWS CodeBuild 运行构建，您可以自动执行发布流程。

下表列出了可用于执行这些操作的任务和方法。本主题不介绍如何使用 AWS 开发工具包完成这些任务。

任务	可用方法	本主题中介绍的方法
借助 CodePipeline 创建可使用 CodeBuild 自动运行生成的持续交付 (CD) 管道	<ul style="list-style-type: none"><li>• CodePipeline 控制台：</li><li>• AWS CLI</li><li>• AWS 开发工具包</li></ul>	<ul style="list-style-type: none"><li>• <a href="#">使用 CodePipeline 控制台。 (p. 369)</a></li><li>• <a href="#">使用 。 AWS CLI (p. 371)</a></li><li>• 您可以调整本主题中的信息以使用 AWS 开发工具包。有关详细信息，请参阅 <code>create_pipeline</code> 您在 <a href="#">SDK 第页</a>，共页工具 Amazon Web Services 或参见 <a href="#">CreatePipeline</a> 在 AWS CodePipeline API 参考。</li></ul>
将借助 CodeBuild 实现的测试和生	<ul style="list-style-type: none"><li>• CodePipeline 控制台：</li></ul>	<ul style="list-style-type: none"><li>• <a href="#">使用 CodePipeline 添加构建自动化的控制台 (p. 374)</a></li></ul>

任务	可用方法	本主题中介绍的方法
将自动化添加到 CodePipeline 中现有的管道	<ul style="list-style-type: none"><li>AWS CLI</li><li>AWS 开发工具包</li></ul>	<ul style="list-style-type: none"><li><a href="#">使用 CodePipeline 添加测试自动化的控制台 (p. 378)</a></li><li>对于 AWS CLI，您可以调整本主题中的信息，以创建包含 CodeBuild 构建操作或测试操作的管道。有关详细信息，请参阅<a href="#">编辑管道 (AWS CLI)</a> 和 <a href="#">CodePipeline 管道结构参考</a> 在 AWS CodePipeline 用户指南。</li><li>您可以调整本主题中的信息以使用 AWS 开发工具包管道。有关更多信息，请参阅<a href="#">适用于</a> 的工具内 Amazon Web Services 软件开发工具包部分中您的编程语言对应的更新管道操作文档，或参阅<a href="#">UpdatePipeline</a> AWS CodePipeline API 参考 中的。</li></ul>

### 主题

- [Prerequisites \(p. 368\)](#)
- [创建使用 CodeBuild 的管道 \(CodePipeline 控制台\) \(p. 369\)](#)
- [创建使用 CodeBuild 的管道 \(AWS CLI\) \(p. 371\)](#)
- [将 CodeBuild 构建操作添加到管道 \(CodePipeline 控制台\) \(p. 374\)](#)
- [将 CodeBuild 测试操作添加到管道 \(CodePipeline 控制台\) \(p. 378\)](#)

## Prerequisites

- 回答[计划构建 \(p. 135\)](#) 中的问题。
- 如果您通过 IAM 用户（而不是 AWS 根账户或管理员 IAM 用户）访问 CodePipeline，请向该用户（或该用户所属的 IAM 组）附加名为 AWSCodePipelineFullAccess 的托管策略。建议不使用 AWS 根账户。此策略向用户授予在 CodePipeline 中创建管道的权限。有关详细信息，请参阅[附加管理的策略](#) 在 IAM 用户指南。

### Note

向该用户（或该用户所属的 IAM 组）附加策略的 IAM 实体在 IAM 中必须拥有附加策略的权限。有关详细信息，请参阅[授权管理权限 IAM 用户、组和凭据](#) 在 IAM 用户指南。

- 创建 CodePipeline 服务角色（如果您的 AWS 账户未提供此服务角色）。CodePipeline 使用此服务角色代表您与其他 AWS 服务（包括 AWS CodeBuild）进行交互。例如，要使用 AWS CLI 创建 CodePipeline 服务角色，请运行 IAM `create-role` 命令：

对于 Linux, macOS, or Unix：

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-role-policy-document '{ "Version": "2012-10-17", "Statement": { "Effect": "Allow", "Principal": {"Service": "codepipeline.amazonaws.com"}, "Action": "sts:AssumeRole" } }'
```

对于 Windows：

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-role-policy-document "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow\", \"Principal\": {\"Service\": \"codepipeline.amazonaws.com\"}, \"Action\": \"sts:AssumeRole\"}}"
```

### Note

创建此 CodePipeline 服务角色的 IAM 实体必须拥有在 IAM 中创建服务角色的权限。  
API 版本 2016-10-06

- 在您创建 CodePipeline 服务角色或识别现有的角色，必须添加默认值 CodePipeline 服务角色政策与服务角色的角色政策，如所述 [审核默认值 CodePipeline 服务角色政策](#) 在 AWS CodePipeline 用户指南，如果这不是职务策略的一部分。

**Note**

添加此 CodePipeline 服务角色策略的 IAM 实体必须拥有在 IAM 中将服务角色策略添加到服务角色的权限。

- 创建并上传源代码到受支持的存储库类型 CodeBuild 和 CodePipeline，例如 CodeCommit，Amazon S3 或 Github。（CodePipeline 目前不支持位数据段。）源代码应包含 BuildSpec 文件，但在本主题稍后定义一个构建项目时，您可以声明一个文件。有关更多信息，请参阅 [构建规范参考 \(p. 136\)](#)。

**Important**

如果您计划使用管道来部署已生成的源代码，则构建输出构件必须与您使用的部署系统兼容。

- 对于 CodeDeploy，请参阅 [AWS CodeDeploy 示例 \(p. 57\)](#) 在本指南和 [准备修订版 CodeDeploy](#) 在 AWS CodeDeploy 用户指南。
- 对于 AWS Elastic Beanstalk，请参阅 [AWS Elastic Beanstalk 示例 \(p. 68\)](#) 在本指南和 [创建应用程序源捆绑包](#) 在 AWS Elastic Beanstalk 开发人员指南。
- 对于 AWS OpsWorks，参见 [应用程序源](#) 和 [使用 CodePipeline 带有 AWS OpsWorks 在 AWS OpsWorks 用户指南](#)。

## 创建使用 CodeBuild 的管道 ( CodePipeline 控制台 )

执行以下步骤，创建使用 CodeBuild 来构建和部署源代码的管道。

要创建仅测试源代码的管道：

- 执行以下步骤来创建管道，然后从管道中删除构建和测试阶段。然后使用本主题中的 [将 CodeBuild 测试操作添加到管道 \( CodePipeline 控制台 \) \(p. 378\)](#) 步骤，将使用 CodeBuild 的测试操作添加到管道。
- 使用本主题中的其他步骤之一来创建管道，然后使用本主题中的 [将 CodeBuild 测试操作添加到管道 \( CodePipeline 控制台 \) \(p. 378\)](#) 步骤，将使用 CodeBuild 的测试操作添加到管道。

使用 CodePipeline 中的创建管道向导来创建使用 CodeBuild 的管道

- 使用以下项登录到 AWS 管理控制台：

- 您的 AWS 根账户。我们不建议这么做。有关详细信息，请参阅 [帐户 root 用户](#) 在 IAM 用户指南。
- AWS 账户中的 IAM 管理员用户。有关详细信息，请参阅 [创建您的第一个 IAM 管理员用户和组](#) 在 IAM 用户指南。
- 您的 AWS 账户中的 IAM 用户有权使用以下最低程度的操作：

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3>ListAllMyBuckets
s3>ListBucket
s3:PutBucketPolicy
codecommit>ListBranches
codecommit>ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy>ListApplications
```

```
codedeploy>ListDeploymentGroups  
elasticbeanstalk>DescribeApplications  
elasticbeanstalk>DescribeEnvironments  
lambda>GetFunctionConfiguration  
lambda>ListFunctions  
opsworks>DescribeStacks  
opsworks>DescribeApps  
opsworks>DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 中打开 AWS CodePipeline 控制台。
3. 在 AWS 区域选择器中，选择构建项目 AWS 资源所在的 AWS 区域。这必须是支持 CodeBuild 的 AWS 区域。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [AWS CodeBuild](#)。
4. 创建管道 如果显示 CodePipeline 信息页面，请选择 Create pipeline (创建管道)。如果显示 Pipelines (管道) 页面，请选择 Create pipeline (创建管道)。
5. 在 步骤1: 选择管道设置 页面，管道名称，输入管道的名称（例如，**CodeBuildDemoPipeline**）。如果您选择其他名称，请确保在本过程中始终使用它。
6. 对于 Role name (角色名称)，执行以下操作之一：

选择 New service role (新服务角色)，然后在 Role Name (角色名称) 中，输入新服务角色的名称。

选择 Existing service role (现有服务角色)，然后选择已创建或标识为此主题的先决条件一部分的 CodePipeline 服务角色。
7. 对于 Artifact store (项目存储)，执行下列操作之一：
  - 选择 Default location (默认位置) 以将默认构件存储（如指定为默认构件存储的 S3 构件存储桶）用于为管道选择的 AWS 区域中的管道。
  - 如果您现在已在管道所在的 AWS 区域中创建构件存储（例如，S3 构件存储桶），请选择 Custom location (自定义位置)。

#### Note

这不是管道的源代码的源存储桶。这是管道的项目存储。管道所在 AWS 区域中的每个管道都需要一个单独的构件存储（如 S3 存储桶）。

8. 选择 Next (下一步)。
9. 在 步骤2: 添加源阶段 页面，源提供商，执行以下操作之一：
  - 如果您的源代码存储在 S3 存储桶中，请选择 Amazon S3。对于 Bucket (存储桶)，选择包含源代码的 S3 存储桶。对于 S3 object key (S3 对象键)，输入包含源代码的文件的名称（例如 **file-name.zip**）。选择 Next (下一步)。
  - 如果您的源代码存储在 AWS CodeCommit 存储库中，请选择 CodeCommit。对于 Repository name，请选择包含源代码的存储库的名称。对于 Branch name (分支名称)，请选择包含要构建的源代码版本的分支名称。选择 Next (下一步)。
  - 如果您的源代码存储在 GitHub 存储库中，请选择 GitHub。选择 Connect to GitHub，然后按照说明进行操作以通过 GitHub 进行身份验证。对于 Repository，请选择包含源代码的存储库的名称。对于 Branch (分支)，请选择包含要构建的源代码版本的分支名称。

选择 Next (下一步)。

10. 在 步骤3: 添加构建阶段 页面，建立提供商，选择 CodeBuild.
11. 如果您已有要使用的构建项目的名称，则对于 Project name (项目名称)，选择构建项目的名称并跳到本过程的步骤 22。否则，使用以下步骤在 CodeBuild 中创建项目。

如果您选择一个现有的生成项目，那么它必须具有已定义的生成输出项目设置（即使 CodePipeline 覆盖它们）。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)或[更改构建项目的设置 \(控制台\) \(p. 236\)](#)。

### Important

如果您为 CodeBuild 项目启用 Webhook，并且该项目用作 CodePipeline 中的构建步骤，则将为每次提交创建两个相同的构建。一个生成通过 Webhook 触发，另一个生成通过 CodePipeline 触发。由于账单基于每个生成，因此您需要为这两个生成付费。因此，如果您使用的是 CodePipeline，建议您在 CodeBuild 中禁用 Webhook。在 AWS CodeBuild 控制台中，清除 Webhook 框。有关更多信息，请参阅 [更改构建项目的设置（控制台）\(p. 236\)](#)。

12. 在 [步骤4: 添加部署阶段](#) 页面，请执行以下操作之一：

- 如果您不想部署构建输出项目，请在系统提示时选择 Skip (跳过) 并确认此选择。
- 如果要部署构建输出项目，对于 Deploy provider (部署提供商)，选择部署提供商，然后在系统提示时指定设置。

选择 Next (下一步)。

13. 在 [Review \(查看\)](#) 页面上，查看您的选择，然后选择 Create pipeline (创建管道)。
14. 管道成功运行后，您可以获取构建输出项目。管道在 CodePipeline 控制台中显示后，在 Build (生成) 操作中，选择工具提示。记下 Output artifact (输出项目) 的值 (例如，MyAppBuild)。

### Note

您还可以通过在 CodeBuild 控制台的构建详细信息页面上选择 Build artifacts (构建构件) 链接来获取构建输出项目。要前往此页面，请跳过此过程中的剩余步骤，并参阅 [查看构建详细信息（控制台）\(p. 262\)](#)。

15. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
16. 在存储桶列表中，请打开管道使用的存储桶。桶的名称应该遵循格式 `codepipeline-region-ID-random-number`...您可以使用 AWS CLI 运行 `CodePipeline get-pipeline` 获取桶名称的命令，其中 `my-pipeline-name` 显示您的管道的显示名称：

```
aws codepipeline get-pipeline --name my-pipeline-name
```

在输出中，该 pipeline 对象包含一个 `artifactStore` 对象，其中包含带有存储桶名称的 `location` 值。

17. 打开与您的管道名称匹配的文件夹（根据管道名称的长度，文件夹名称可能被截断），然后打开与您之前记下的 Output artifact (输出构件) 的值匹配的文件夹。
18. 提取文件内容。如果该文件夹中有多个文件，请提取具有最新 Last Modified 时间戳的文件的内容。（您可能需要提供文件 .zip 扩展名以便您在系统的 ZIP 实用程序中处理此事。）构造输出伪影位于文件的提取内容中。
19. 如果您指示 CodePipeline 部署构建输出项目，请使用部署提供商的说明，获取部署目标上的构建输出项目。

## 创建使用 CodeBuild 的管道 (AWS CLI)

执行以下步骤，创建使用 CodeBuild 来构建源代码的管道。

使用 AWS CLI 要创建一个部署您构建的源代码或仅测试源代码的管道，您可以调整中的说明 [编辑管道（AWS CLI）](#) 和 [CodePipeline 管道结构参考](#) 在 AWS CodePipeline 用户指南。

1. 创建或标识 CodeBuild 中的构建项目。有关更多信息，请参阅 [创建构建项目 \(\) \(p. 189\)](#)。)

### Important

生成项目必须定义生成输出项目设置（即使 CodePipeline 覆盖它们）。有关更多信息，请参阅 [创建构建项目 \(AWS CLI\) \(p. 198\)](#) 中 `artifacts` 的描述。

2. 请确保您已使用与本主题中所述的 IAM 实体之一对应的 AWS 访问密钥和 AWS 秘密访问密钥配置 AWS CLI。有关详细信息，请参阅 [设置与 AWS Command Line Interface](#) 在 AWS Command Line Interface 用户指南。
3. 创建代表管道结构的 JSON 格式的文件。将文件命名为 `create-pipeline.json` 或类似名称。例如，此 JSON 格式的结构借助引用了 S3 输入存储桶的源操作和使用 CodeBuild 的构建操作创建了管道：

```
{  
  "pipeline": {  
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",  
    "stages": [  
      {  
        "name": "Source",  
        "actions": [  
          {  
            "inputArtifacts": [],  
            "name": "Source",  
            "actionTypeId": {  
              "category": "Source",  
              "owner": "AWS",  
              "version": "1",  
              "provider": "S3"  
            },  
            "outputArtifacts": [  
              {  
                "name": "MyApp"  
              }  
            ],  
            "configuration": {  
              "S3Bucket": "my-input-bucket-name",  
              "S3ObjectKey": "my-source-code-file-name.zip"  
            },  
            "runOrder": 1  
          }  
        ]  
      },  
      {  
        "name": "Build",  
        "actions": [  
          {  
            "inputArtifacts": [  
              {  
                "name": "MyApp"  
              }  
            ],  
            "name": "Build",  
            "actionTypeId": {  
              "category": "Build",  
              "owner": "AWS",  
              "version": "1",  
              "provider": "CodeBuild"  
            },  
            "outputArtifacts": [  
              {  
                "name": "default"  
              }  
            ],  
            "configuration": {  
              "ProjectName": "my-build-project-name"  
            },  
            "runOrder": 1  
          }  
        ]  
      }  
    ]  
  }  
}
```

```
],
"artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
},
"name": "my-pipeline-name",
"version": 1
}
}
```

在此 JSON 格式的数据中：

- `roleArn` 的值必须与您作为先决条件的一部分创建或标识的 CodePipeline 服务角色 ARN 相匹配。
- `configuration` 中 `S3Bucket` 和 `S3ObjectKey` 的值假定源代码存储在 S3 存储桶中。有关其他源代码库类型的设置，请参阅 [CodePipeline 管道结构参考](#) 在 AWS CodePipeline 用户指南。
- `ProjectName` 的值是您之前在此过程中创建的 CodeBuild 生成项目的名称。
- `location` 的值是此管道所用的 S3 存储桶的名称。有关详细信息，请参阅 [创建一个用作工艺存储的 S3 桶的策略](#) [CodePipeline](#) 在 AWS CodePipeline 用户指南。
- `name` 的值是此管道的名称。所有管道名称对您的账户都必须是唯一的。

尽管此数据仅介绍源操作和构建操作，但您可以为与测试、部署构建输出项目和调用 AWS Lambda 函数等相关的活动添加操作。有关详细信息，请参阅 [AWS CodePipeline 管道结构参考](#) 在 AWS CodePipeline 用户指南。

4. 切换到包含 JSON 文件的文件夹，然后运行 CodePipeline `create-pipeline` 命令，并指定文件名：

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

#### Note

您必须在支持 CodeBuild 的 AWS 区域中创建管道。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [AWS CodeBuild](#)。

输出中将显示 JSON 格式的数据，并且 CodePipeline 会创建管道。

5. 要获取有关管道状态的信息，请运行 CodePipeline `get-pipeline-state` 命令，指定管道名称：

```
aws codepipeline get-pipeline-state --name my-pipeline-name
```

在输出中，查找确认构建成功的信息。省略号 (...) 用于显示为简洁起见而省略的数据。

```
{
...
"stageStates": [
...
{
    "actionStates": [
        {
            "actionName": "CodeBuild",
            "latestExecution": {
                "status": "SUCCEEDED",
                ...
            },
            ...
        }
    ]
}
]
```

}

如果您过早运行此命令，您可能不会看到有关构建操作的信息。您可能需要多次运行此命令，直到管道已完成构建操作的运行。

6. 成功构建后，请按照以下说明操作，获取构建输出项目。通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。

#### Note

您还可以通过在 CodeBuild 控制台的相关构建详细信息页面上选择 Build artifacts (构建构件) 链接来获取构建输出项目。要前往此页面，请跳过此过程中的剩余步骤，并参阅 [查看构建详细信息 \( 控制台 \) \(p. 262\)](#)。

7. 在存储桶列表中，请打开管道使用的存储桶。桶的名称应该遵循格式 `codepipeline-region-ID-random-number`...您可以从 `create-pipeline.json` 或者您可以运行 `CodePipeline get-pipeline` 命令获取桶的名称。

```
aws codepipeline get-pipeline --name my-pipeline-name
```

在输出中，该 `pipeline` 对象包含一个 `artifactStore` 对象，其中包含带有存储桶名称的 `location` 值。

8. 打开与您的管道名称相匹配的文件夹 (例如，`my-pipeline-name`)。
9. 在该文件夹中，打开名为 `default` 的文件夹。
10. 提取文件内容。如果该文件夹中有多个文件，请提取具有最新 Last Modified 时间戳的文件的内容。（您可能需要提供文件A `.zip` 扩展名以便您在系统的ZIP实用程序中处理此事务。）构造输出伪影位于文件的提取内容中。

## 将 CodeBuild 构建操作添加到管道 ( CodePipeline 控制台 )

1. 使用以下项登录到 AWS 管理控制台：

- 您的 AWS 根账户。我们不建议这么做。有关详细信息，请参阅 [帐户root用户 在 IAM 用户指南](#)。
- AWS 账户中的 IAM 管理员用户。有关详细信息，请参阅 [创建您的第一个 IAM 管理员用户和组 在 IAM 用户指南](#)。
- AWS 账户中的 IAM 用户，具有执行以下最基本操作的权限：

```
codepipeline:*
iam>ListRoles
iam>PassRole
s3>CreateBucket
s3>GetBucketPolicy
s3>GetObject
s3>ListAllMyBuckets
s3>ListBucket
s3>PutBucketPolicy
codecommit>ListBranches
codecommit>ListRepositories
codedeploy>GetApplication
codedeploy>GetDeploymentGroup
codedeploy>ListApplications
codedeploy>ListDeploymentGroups
elasticbeanstalk>DescribeApplications
elasticbeanstalk>DescribeEnvironments
lambda>GetFunctionConfiguration
lambda>ListFunctions
```

```
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 中打开 CodePipeline 控制台。
3. 在 AWS 区域选择器中，选择管道所在的 AWS 区域。这必须是支持 CodeBuild 的区域。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [CodeBuild](#)。
4. 在 Pipelines (管道) 页面上，选择管道的名称。
5. 在管道详细信息页面的 Source (源) 操作中，选择工具提示。记下 Output artifact (输出项目) 的值 (例如，MyApp)：

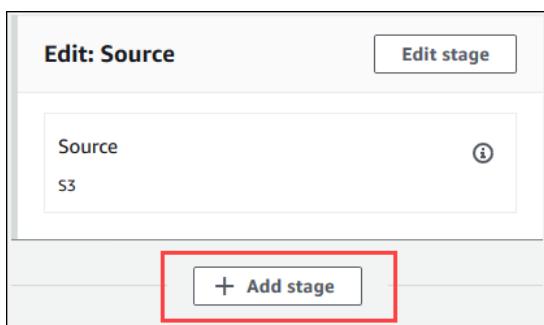
Note

此过程向您演示如何将生成操作添加到 Source 和 Beta 阶段之间的生成阶段内。如果您要在其他位置添加生成操作，在您要添加生成操作的位置之前的操作上选择工具提示，并记下 Output artifact (输出项目) 的值。

6. 选择 Edit (编辑)。
7. 在 Source (源) 和 Beta (测试版) 阶段之间，选择 Add stage (添加阶段)。

Note

此过程向您演示如何在 Source (源) 和 Beta (测试) 阶段之间添加构建阶段。要将生成操作添加到现有的阶段，请选择阶段中的 Edit stage (编辑阶段)，然后跳到此过程的步骤 8。要在其他位置添加构建阶段，请在所需位置选择 Add stage (添加阶段)。



8. 对于 Stage name (阶段名称)，输入生成阶段的名称 (例如，**Build**)。如果您选择了其他名称，请在整个过程中使用该名称。
9. 在选定阶段内，选择 Add action (添加操作)。

Note

此过程向您演示如何在构建阶段内添加构建操作。要在其他位置添加构建操作，请在所需位置选择 Add action (添加操作)。您可能需要先在您要添加构建操作的现有阶段内选择 Edit stage (编辑阶段)。

10. 在 Edit action (编辑操作) 中，对于 Action name (操作名称)，输入操作的名称 (例如，**CodeBuild**)。如果您选择了其他名称，请在整个过程中使用该名称。
11. 对于 Action provider (操作提供商)，选择 CodeBuild。
12. 如果您在 CodeBuild 中已有一个构建项目，则对于 Project name (项目名称)，选择构建项目的名称，然后跳到本过程的步骤 22。

如果您选择一个现有的生成项目，那么它必须具有已定义的生成输出项目设置 (即使 CodePipeline 覆盖它们)。有关更多信息，请参阅[创建构建项目 \(控制台\) \(p. 190\)](#)或[更改构建项目的设置 \(控制台\) \(p. 236\)](#)中的 Artifacts (构件) 的描述。

### Important

如果您为 CodeBuild 项目启用 Webhook，并且该项目用作 CodePipeline 中的构建步骤，则将为每次提交创建两个相同的构建。一个生成通过 Webhook 触发，另一个生成通过 CodePipeline 触发。由于账单基于每个生成，因此您需要为这两个生成付费。因此，如果您使用的是 CodePipeline，建议您在 CodeBuild 中禁用 Webhook。在 CodeBuild 控制台中，清除 Webhook 框。有关更多信息，请参阅[更改构建项目的设置（控制台）\(p. 236\)](#)

13. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
14. 如果显示 CodeBuild 信息页面，请选择 Create build project (创建构建项目)。否则，请在导航窗格上展开 Build (构建)，然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
15. 对于 Project name (项目名称)，输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。
16. (可选) 输入描述。
17. 对于 Environment (环境)，请执行以下操作之一：
  - 要使用以 CodeBuild 管理的 Docker 映像为基础的构建环境，请选择 Managed image (托管映像)。从 Operating system (操作系统)、Runtime (运行时) 和 Runtime version (运行时版本) 下拉列表中进行选择。有关更多信息，请参阅[CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。
  - 要使用以 AWS 账户内 Amazon ECR 存储库中的 Docker 映像为基础的构建环境，请选择 Custom image (自定义映像)。对于 Environment type (环境类型)，选择一种环境类型，然后选择 Amazon ECR。使用 Amazon ECR repository (Amazon ECR 存储库) 和 Amazon ECR image (Amazon ECR 映像) 下拉列表选择 Amazon ECR 存储库和该存储库中的 Docker 映像。
  - 要使用以 Docker Hub 中公开提供的 Docker 映像为基础的构建环境，请选择 Other location (其他位置)。在 Other location (其他位置) 中，使用格式 `docker repository/docker-image-name` 输入 Docker 映像 ID。

仅当您计划使用此构建项目来构建 Docker 映像且您选择的构建环境映像不是由具有 Docker 支持的 CodeBuild 提供时，才选择 Privileged (特权)。否则，尝试与 Docker 守护程序交互的所有关联的构建都将失败。您还必须启动 Docker 守护程序，以便您的构建可根据需要与之交互。要这样做，您可以通过运行以下构建命令，在您构建规范的 `install` 阶段初始化 Docker 守护程序。(如果选择了由支持 Docker 的 CodeBuild 提供的构建环境映像，请不要运行以下构建命令。)

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --  
host=tcp://127.0.0.1:2375 --storage-driver=overlay&  
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

18. 在 Service role (服务角色) 中，执行下列操作之一：

- 如果您没有 CodeBuild 服务角色，请选择 New service role (新建服务角色)。在 Role name (角色名称) 中，为新角色输入一个名称。
- 如果您有 CodeBuild 服务角色，请选择 Existing service role (现有服务角色)。在 Role ARN (角色 ARN) 中，选择服务角色。

### Note

当您使用控制台来创建或更新生成项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

19. 展开 Additional configuration (其他配置)。

要指定 60 分钟 (默认值) 以外的构建超时时间，请使用 hours (小时) 和 minutes (分钟) 框设置一个介于 5 和 480 分钟 (8 小时) 之间的超时时间。

对于 Compute (计算) , 请选择一个可用选项。

对于 Environment variables (环境变量) , 请使用 Name (名称) 和 Value (值) 为要使用的构建环境指定任何可选的环境变量。要添加更多环境变量 , 请选择 Add environment variable (添加环境变量)。

#### Important

强烈建议不要将敏感值 (尤其是 AWS 访问密钥 ID 和秘密访问密钥 ) 存储在环境变量中。可以使用 CodeBuild 控制台和 AWS CLI 以纯文本格式显示环境变量。

要存储和检索敏感值 , 我们建议您的构建命令使用 AWS CLI 来与 Amazon EC2 Systems Manager Parameter Store 进行交互。AWS CLI 已在由 CodeBuild 提供的所有构建环境上安装和配置。有关更多信息 , 请参阅 Amazon EC2 Systems Manager 用户指南 中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store CLI 演练](#)。

20. 对于 Buildspec , 执行以下操作之一 :

- 如果您的源代码包括生成规范文件 , 请选择 Use a buildspec file (使用 buildspec 文件)。
- 如果您的源代码不包含构建规范文件 , 请选择 Insert build commands (插入构建命令)。对于 Build commands (构建命令) , 请在构建环境中输入您要在构建阶段运行的命令。对于多条命令 , 针对基于 Linux 的构建环境 , 请使用 && 将每条命令分隔开 ; 针对基于 Windows 的构建环境 , 请使用 ; 将每条命令分隔开。对于 Output files (输出文件) , 请输入您要在构建环境中发送给 CodePipeline 的构建输出文件的路径。对于多个文件 , 请用逗号分隔各个文件路径。

21. 选择 Create build project (创建构建项目)。

22. 返回到 CodePipeline 控制台。

23. 对于 Input artifacts (输入构件) , 选择您在此过程的前面记下的输出构件。

24. 对于 Output artifacts (输出项目) , 输入输出项目的名称 (例如 , **MyAppBuild** )。

25. 选择添加操作。

26. 选择 Save (保存) , 然后选择 Save (保存) 以保存对管道的更改。

27. 选择 Release change。

28. 管道成功运行后 , 您可以获取构建输出项目。管道在 CodePipeline 控制台中显示后 , 在 Build (生成) 操作中 , 选择工具提示。记下 Output artifact (输出项目) 的值 (例如 , MyAppBuild)。

#### Note

您还可以通过在 CodeBuild 控制台的构建详细信息页面上选择 Build artifacts (构建构件) 链接来获取构建输出项目。要访问此页面 , 请参阅 [查看构建详细信息 \(控制台\) \(p. 262\)](#) , 然后跳到此过程的步骤 31。

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

30. 在存储桶列表中 , 请打开管道使用的存储桶。桶的名称应该遵循格式 `codepipeline-region-ID-random-number`...您可以使用 AWS CLI 运行 CodePipeline get-pipeline 获取桶名称的命令:

```
aws codepipeline get-pipeline --name my-pipeline-name
```

在输出中 , 该 pipeline 对象包含一个 artifactStore 对象 , 其中包含带有存储桶名称的 location 值。

31. 打开与您的管道名称匹配的文件夹 (根据管道名称的长度 , 文件夹名称可能被截断 ) , 然后打开与您在此过程的前面记下的 Output artifact (输出构件) 的值匹配的文件夹。
32. 提取文件内容。如果该文件夹中有多个文件 , 请提取具有最新 Last Modified 时间戳的文件的内容。(您可能需要提供文件 .zip 扩展名以便您在系统的ZIP实用程序中处理此事。) 构造输出伪影位于文件的提取内容中。
33. 如果您指示 CodePipeline 部署构建输出项目 , 请使用部署提供商的说明 , 获取部署目标上的构建输出项目。

## 将 CodeBuild 测试操作添加到管道 ( CodePipeline 控制台 )

### 1. 使用以下项登录到 AWS 管理控制台：

- 您的 AWS 根账户。我们不建议这么做。有关详细信息，请参阅 [帐户root用户 在 IAM 用户指南](#).
- AWS 账户中的 IAM 管理员用户。有关详细信息，请参阅 [创建您的第一个 IAM 管理员用户和组 在 IAM 用户指南](#).
- AWS 账户中的 IAM 用户，具有执行以下最基本操作的权限：

```
codepipeline:*
iam>ListRoles
iam>PassRole
s3>CreateBucket
s3>GetBucketPolicy
s3>GetObject
s3>ListAllMyBuckets
s3>ListBucket
s3>PutBucketPolicy
codecommit>ListBranches
codecommit>ListRepositories
codedeploy>GetApplication
codedeploy>GetDeploymentGroup
codedeploy>ListApplications
codedeploy>ListDeploymentGroups
elasticbeanstalk>DescribeApplications
elasticbeanstalk>DescribeEnvironments
lambda>GetFunctionConfiguration
lambda>ListFunctions
opsworks>DescribeStacks
opsworks>DescribeApps
opsworks>DescribeLayers
```

2. 在 <https://console.aws.amazon.com/codesuite/codepipeline/home> 中打开 CodePipeline 控制台。
3. 在 AWS 区域选择器中，选择管道所在的 AWS 区域。这必须是支持 CodeBuild 的 AWS 区域。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 [AWS CodeBuild](#)。
4. 在 Pipelines (管道) 页面上，选择管道的名称。
5. 在管道详细信息页面的 Source (源) 操作中，选择工具提示。记下 Output artifact (输出项目) 的值 (例如，MyApp) :

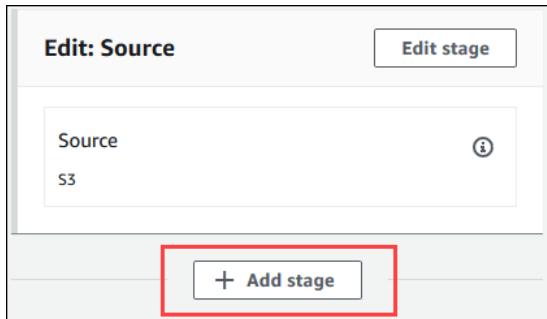
#### Note

此过程向您演示如何将测试操作添加到 Source 和 Beta 阶段之间的测试阶段内。如果您要在其他位置添加测试操作，请将鼠标指针停留在之前的操作上，然后记下 Output artifact (输出项目) 的值。

6. 选择 Edit (编辑)。
7. 紧接着 Source (源) 阶段，选择 Add stage (添加阶段)。

#### Note

此过程向您演示如何在管道中紧接着 Source (源) 阶段添加测试阶段。要将测试操作添加到现有的阶段，请选择阶段中的 Edit stage (编辑阶段)，然后跳到此过程的步骤 8。要在其他位置添加测试阶段，请在所需位置选择 Add stage (添加阶段)。



8. 对于 Stage name (阶段名称) , 输入测试阶段的名称 (例如 , **Test** ) 。如果您选择了其他名称 , 请在整个过程中使用该名称。
9. 在选定阶段中 , 选择 Add action (添加操作)。

Note

此过程向您演示如何在测试阶段内添加测试操作。要在其他位置添加测试操作 , 请在所需位置选择 Add action (添加操作)。您可能需要先在您要添加测试操作的现有阶段内选择 Edit stage (编辑阶段)。

10. 在 Edit action (编辑操作) 中 , 对于 Action name (操作名称) , 输入操作的名称 (例如 , **Test** ) 。如果您选择了其他名称 , 请在整个过程中使用该名称。
11. 对于 Action provider (操作提供商) , 选择 Test (测试) 下的 CodeBuild。
12. 如果您在 CodeBuild 中已有一个构建项目 , 则对于 Project name (项目名称) , 选择构建项目的名称 , 然后跳到本过程的步骤 22。

Important

如果您为 CodeBuild 项目启用 Webhook , 并且该项目用作 CodePipeline 中的构建步骤 , 则将为每次提交创建两个相同的构建。一个生成通过 Webhook 触发 , 另一个生成通过 CodePipeline 触发。由于账单基于每个生成 , 因此您需要为这两个生成付费。因此 , 如果您使用的是 CodePipeline , 建议您在 CodeBuild 中禁用 Webhook。在 CodeBuild 控制台中 , 清除 Webhook 框。有关更多信息 , 请参阅[更改构建项目的设置 \(控制台\) \(p. 236\)](#)

13. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>。
14. 如果显示 CodeBuild 信息页面 , 请选择 Create build project (创建构建项目)。否则 , 请在导航窗格上展开 Build (构建) , 然后依次选择 Build projects (构建项目) 和 Create build project (创建构建项目)。
15. 对于 Project name (项目名称) , 输入此构建项目的名称。构建项目名称在您的各个 AWS 账户内必须是唯一的。
16. (可选) 输入描述。
17. 对于 Environment (环境) , 请执行以下操作之一 :
  - 要使用以 CodeBuild 管理的 Docker 映像为基础的构建环境 , 请选择 Managed image (托管映像)。从 Operating system (操作系统)、Runtime (运行时) 和 Runtime version (运行时版本) 下拉列表中进行选择。有关更多信息 , 请参阅[CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。
  - 要使用以 AWS 账户内 Amazon ECR 存储库中的 Docker 映像为基础的构建环境 , 请选择 Custom image (自定义映像)。对于 Environment type (环境类型) , 选择一种环境类型 , 然后选择 Amazon ECR。使用 Amazon ECR repository (Amazon ECR 存储库) 和 Amazon ECR image (Amazon ECR 映像) 下拉列表选择 Amazon ECR 存储库和该存储库中的 Docker 映像。
  - 要使用以 Docker Hub 中公开提供的 Docker 映像为基础的构建环境 , 请选择 Other location (其他位置)。在 Other location (其他位置) 中 , 使用格式 `docker repository/docker-image-name` 输入 Docker 映像 ID。

仅当您计划使用此构建项目来构建 Docker 映像且您选择的构建环境映像不是由具有 Docker 支持的 CodeBuild 提供时 , 才选择 Privileged (特权)。否则 , 尝试与 Docker 守护程序交互的所有关联的构建

都将失败。您还必须启动 Docker 守护程序，以便您的构建可根据需要与之交互。要这样做，您可以通过运行以下构建命令，在您构建规范的 `install` 阶段初始化 Docker 守护程序。（如果选择了由支持 Docker 的 CodeBuild 提供的构建环境映像，请不要运行以下构建命令。）

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --  
host=tcp://127.0.0.1:2375 --storage-driver=overlay&  
- timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
```

18. 在 Service role (服务角色) 中，执行下列操作之一：

- 如果您没有 CodeBuild 服务角色，请选择 New service role (新建服务角色)。在 Role name (角色名称) 中，为新角色输入一个名称。
- 如果您有 CodeBuild 服务角色，请选择 Existing service role (现有服务角色)。在 Role ARN (角色 ARN) 中，选择服务角色。

Note

当您使用控制台来创建或更新生成项目时，您可以同时创建 CodeBuild 服务角色。默认情况下，这个角色仅能与该构建项目配合使用。如果您使用控制台将此服务角色与另一个构建项目关联，则此角色将更新以便与关联的构建项目结合使用。一个服务角色最多可与 10 个构建项目结合使用。

19. 展开 Additional configuration (其他配置)。

要指定 60 分钟（默认值）以外的构建超时时间，请使用 hours (小时) 和 minutes (分钟) 框设置一个介于 5 和 480 分钟（8 小时）之间的超时时间。

对于 Compute (计算)，请选择一个可用选项。

对于 Environment variables (环境变量)，请使用 Name (名称) 和 Value (值) 为要使用的构建环境指定任何可选的环境变量。要添加更多环境变量，请选择 Add environment variable (添加环境变量)。

Important

强烈建议不要将敏感值（尤其是 AWS 访问密钥 ID 和秘密访问密钥）存储在环境变量中。可以使用 CodeBuild 控制台和 AWS CLI 以纯文本格式显示环境变量。

要存储和检索敏感值，我们建议您的构建命令使用 AWS CLI 来与 Amazon EC2 Systems Manager Parameter Store 进行交互。AWS CLI 已在由 CodeBuild 提供的所有构建环境上安装和配置。有关更多信息，请参阅 Amazon EC2 Systems Manager 用户指南 中的 [Systems Manager Parameter Store](#) 和 [Systems Manager Parameter Store CLI 演练](#)。

20. 对于 Buildspec，执行以下操作之一：

- 如果您的源代码包括生成规范文件，请选择 Use a buildspec file (使用 buildspec 文件)。
- 如果您的源代码不包含构建规范文件，请选择 Insert build commands (插入构建命令)。对于 Build commands (构建命令)，请在构建环境中输入您要在构建阶段运行的命令。对于多条命令，针对基于 Linux 的构建环境，请使用 `&` 将每条命令分隔开；针对基于 Windows 的构建环境，请使用 `;` 将每条命令分隔开。对于 Output files (输出文件)，请输入您要在构建环境中发送给 CodePipeline 的构建输出文件的路径。对于多个文件，请用逗号分隔各个文件路径。

21. 选择 Create build project (创建构建项目)。

22. 返回到 CodePipeline 控制台。

23. 对于 Input artifacts (输入构件)，选择您在此过程的前面记下的 Output artifact (输出构件) 的值。

24.（可选）如果您希望测试操作来生成输出构件，并且相应地设置构建规范，那么对于 Output artifact (输出构件)，请输入您要分配给输出构件的值。

25. 选择 Save (保存)。

26. 选择 Release change。

27. 管道成功运行后，您可以获取测试结果。在管道的 Test (测试) 阶段中，选择 CodeBuild 超链接以在 CodeBuild 控制台中打开相关的生成项目页面。
28. 在生成项目页面上的 Build history (生成历史记录) 中，选择 Build run (生成运行) 超链接。
29. 在生成运行页面的 Build logs (生成日志) 中，选择 View entire log (查看完整日志) 超链接以在 Amazon CloudWatch 控制台中打开相关的生成日志。
30. 滚动浏览构建日志，查看测试结果。

## 将 AWS CodeBuild 与 Jenkins 结合使用

可以使用适用于 AWS CodeBuild 的 Jenkins 插件将 CodeBuild 与您的 Jenkins 构建作业集成。您可以使用插件将您的构建作业发送给 CodeBuild，而不是发送给 Jenkins 构建节点。这样一来，便无需预置、配置和管理 Jenkins 构建节点。

### 设置 Jenkins

有关设置 Jenkins 与 AWS CodeBuild 插件以及下载插件源代码，请参阅 <https://github.com/awslabs/aws-codebuild-jenkins-plugin>。

### 安装插件

如果您已设置 Jenkins 服务器并希望仅安装 AWS CodeBuild 插件，请在您的 Jenkins 实例上的插件管理器中搜索 **CodeBuild Plugin for Jenkins**。

### 使用插件

#### 将 AWS CodeBuild 与 VPC 外部的源结合使用

1. 在 CodeBuild 控制台中创建项目。有关更多信息，请参阅 [创建构建项目 \(控制台\) \(p. 190\)](#)。
  - 选择要在其中运行构建的 AWS 区域。
  - (可选) 将 Amazon VPC 配置设置为允许 CodeBuild 构建容器访问 VPC 中的资源。
  - 记下您的项目的名称。您在步骤 3 中需要它。
  - (可选) 如果 CodeBuild 本机不支持您的源存储库，则可以将 Amazon S3 设置为您的项目的输入源类型。
2. 在 IAM 控制台中，创建一个 IAM 用户以供 Jenkins 插件使用。
  - 当您为该用户创建凭证时，请选择编程访问。
  - 创建如下所示的策略，然后将该策略附加到您的用户。

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Resource": ["arn:aws:logs:{region}:{awsAccountId}:log-group:/aws/  
codebuild/{projectName}::*"],  
            "Action": ["logs:GetLogEvents"]  
        },  
        {  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::{inputBucket}"],  
            "Action": ["s3:GetBucketVersioning"]  
        },  
        {  
            "Effect": "Allow",  
            "Resource": ["arn:aws:s3:::{inputBucket}/*"],  
            "Action": ["s3:GetObject"]  
        }  
    ]  
}
```

```
"Resource": ["arn:aws:s3:::{inputBucket}/*{{inputObject}}"],
  "Action": ["s3:PutObject"]
},
{
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::{outputBucket}/*"],
  "Action": ["s3:GetObject"]
},
{
  "Effect": "Allow",
  "Resource": ["arn:aws:codebuild:{region}:{awsAccountId}:project/{projectName}"],
  "Action": ["codebuild:StartBuild",
    "codebuild:BatchGetBuilds",
    "codebuild:BatchGetProjects"]
}
]
```

3. 在 Jenkins 中创建一个自由式项目。

- 在 Configure (配置) 页面上，选择 Add build step (添加构建步骤)，然后选择 Run build on CodeBuild (在 AWS CodeBuild 上运行构建)。
  - 配置您的构建步骤。
    - 为 Region (区域)、Credentials (凭证) 和 Project Name (项目名称) 提供值。
    - 选择 Use Project source (使用项目源)。
    - 保存配置并从 Jenkins 运行构建任务。
4. 对于 Source Code Management (源代码管理)，选择您希望如何检索您的源。您可能需要在 Jenkins 服务器上安装 GitHub 插件 (或您的源存储库提供商的 Jenkins 插件)。
- 在 Configure (配置) 页面上，选择 Add build step (添加构建步骤)，然后选择 Run build on AWS CodeBuild (在 AWS CodeBuild 上运行构建任务)。
  - 配置您的构建步骤。
    - 为 Region (区域)、Credentials (凭证) 和 Project Name (项目名称) 提供值。
    - 选择 Use Jenkins source (使用 Jenkins 源)。
    - 保存配置并从 Jenkins 运行构建任务。

将 AWS CodeBuild 插件与 Jenkins 管道插件结合使用

- 在您的 Jenkins 管道项目页面上，使用代码段生成器来生成将 CodeBuild 作为管道中的步骤添加的管道脚本。它应生成如下所示的脚本：

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
sourceControlType: 'jenkins'
```

## 将 AWS CodeBuild 与 Codecov 结合使用

Codecov 是一种用于测量代码的测试覆盖率的工具。Codecov 可标识您代码中的哪些方法和语句未经测试。可通过结果确定在何处编写测试以提高代码质量。CODECOV 可用于支持的三个源存储库 CodeBuild、Github、GithubEnterpriseServer 和 Bitbucket。如果您的构建项目使用的是 GitHub Enterprise Server，则您必须使用 Codecov Enterprise。

当您运行与 Codecov 集成的 CodeBuild 项目的构建时，会将用于分析存储库中的代码的 Codecov 报告上传到 Codecov。构建日志包含指向报告的链接。此示例介绍如何将 Python 和 Java 构建项目与 Codecov 集成。有关 Codecov 支持的语言的列表，请参阅 Codecov 网站上的 [Codecov 支持的语言](#)。

## 将Codecov 集成到构建项目中

### 将Codecov 与构建项目集成

1. 转到 <https://codecov.io/signup> 并注册 GitHub 或 Bitbucket 源存储库。如果您使用的是 GitHub Enterprise，请参阅 Codecov 网站上的 [Codecov Enterprise](#)。
2. 在Codecov 中，添加要覆盖的存储库。
3. 在显示令牌信息时，选择 Copy (复制)。

Let's get your project covered

No repository activation required. Simply upload a report and the project will be activated automatically.

STEP 1 - COPY TOKEN

Upload Token

4. 将复制的令牌作为名为 CODECOV\_TOKEN 的环境变量添加到构建项目中。有关更多信息，请参阅 [更改构建项目的设置 \(控制台\) \(p. 236\)](#)。
5. 在存储库中创建一个名为 my\_script.sh 的文本文件。在文件中输入以下内容：

```
#/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. 根据构建项目的使用情况，选择 Python 或 Java 选项卡，然后按照以下步骤操作。

#### Java

1. 将以下 JaCoCo 插件添加到存储库中的 pom.xml 中。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.2</version>
      <executions>
        <execution>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>
        <execution>
          <id>report</id>
          <phase>test</phase>
          <goals>
            <goal>report</goal>
```

```
        </goals>
    </execution>
</executions>
</plugin>
</plugins>
</build>
```

2. 在构建规范文件中输入以下命令。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#)。)

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my script.sh
```

Python

在构建规范文件中输入以下命令。有关更多信息，请参阅 [构建规范语法 \(p. 137\)](#)。

```
build:
  - pip install coverage
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

- 运行构建项目的构建。指向为项目生成的 Codecov 报告的链接将显示在构建日志中。使用链接查看 Codecov 报告。有关更多信息，请参阅 [在 AWS CodeBuild 中运行构建 \(p. 252\)](#) 和 [使用 AWS CloudTrail 记录 AWS CodeBuild API 调用 \(p. 300\)](#)。构建日志中的 Codecov 信息与以下内容类似：

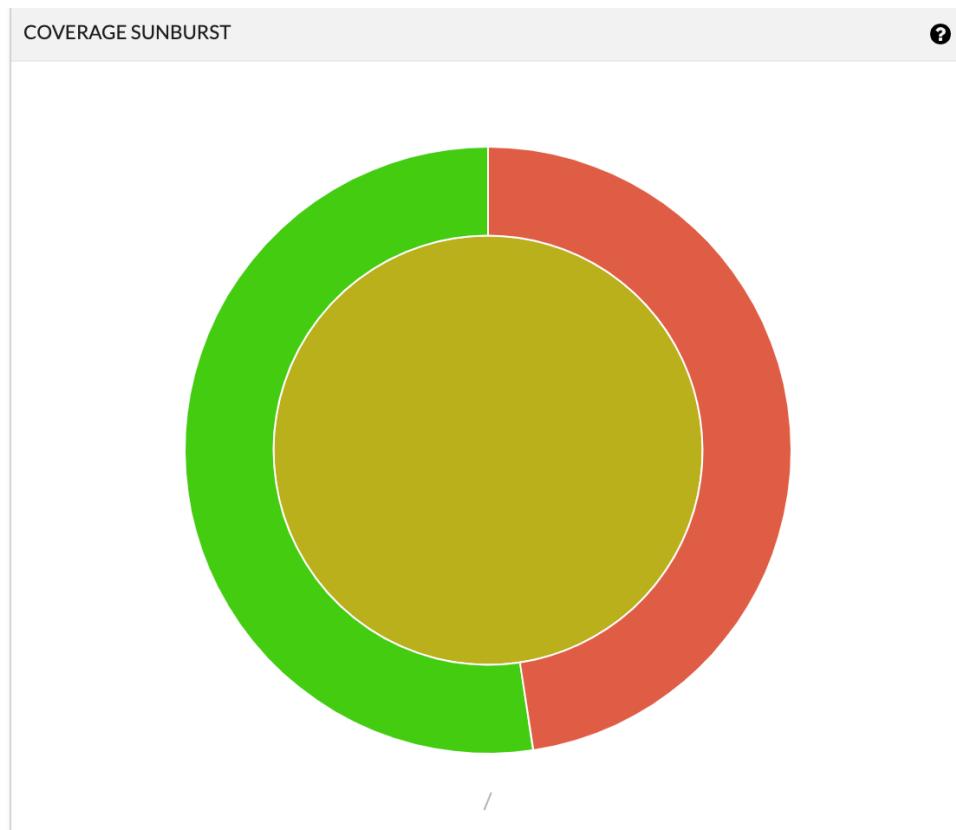
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh

      /__\|      |_
     |  | /_ \ /_ | | /_ \ /_ /_ \ \ / /
     |  |(--) | (--) | _/ (--) (--) \ v /
     \_\_\_/\_\_,|_\_\_| \_\_\_/\_\_/
                                         Bash-20200303-bc4d7e6

[0;90m=>[0m AWS Codebuild detected.
... The full list of Codecov log entries has been omitted for brevity ...
.
[0;32m->[0m View reports at [0;36mhttps://codecov.io/github/test\_py/commit/commit-id[0m

[Container] 2020/03/09 16:31:07 Phase complete: POST_BUILD State: SUCCEEDED
```

报告与以下内容类似：



File	Line Coverage
code.py	10
tests.py	11
<b>Project Totals (2 files)</b>	<b>21</b>

## 使用 AWS CodeBuild 无服务器应用

AWS 无服务器应用程序模型 (AWS SAM) 是一个用于构建无服务器应用程序的开源框架。有关更多信息，请参阅 GitHub 上的 [AWS 无服务器应用程序模型存储库](#)。

您可以使用 AWS CodeBuild 打包和部署遵循 AWS SAM 标准的无服务器应用程序。对于部署步骤，CodeBuild 可以使用 AWS CloudFormation。要通过 CodeBuild 和 AWS CloudFormation 自动构建和部署无服务器应用程序，您可以使用 AWS CodePipeline。

有关详细信息，请参阅 [部署无服务器应用程序](#) 在 AWS 无服务器应用程序模型 开发人员指南。

## 相关资源

- 有关 AWS CodeBuild 入门的信息，请参阅[通过控制台开始使用 AWS CodeBuild \(p. 4\)](#)。
- 有关解决 CodeBuild 中的问题的信息，请参阅[AWS CodeBuild 问题排查 \(p. 387\)](#)。
- 有关 CodeBuild 中的配额的信息，请参阅[AWS CodeBuild 的配额 \(p. 400\)](#)。

# AWS CodeBuild 问题排查

使用本主题中的信息来帮助您识别、诊断和解决问题。要了解如何记录和监控 CodeBuild 版本以排查问题，请参阅[日志记录和监控 \(p. 300\)](#)。

## 主题

- 来自错误存储库的 Apache Maven 构建参考构件 (p. 387)
- 默认情况下，以根用户身份运行构建命令 (p. 388)
- 当文件名为非美国时,内部版本可能会失败。英文字符 (p. 389)
- 当从 Amazon EC2 Parameter Store 获取参数时，构建可能失败 (p. 389)
- 无法在 CodeBuild 控制台中访问分支筛选条件 (p. 390)
- 无法查看构建是成功还是失败 (p. 390)
- 无法找到并选择 Windows Server Core 2016 平台的基本映像 (p. 390)
- 构建规范文件中的前期命令无法被后续命令识别 (p. 390)
- 错误 尝试下载缓存时“访问被拒绝” (p. 391)
- 错误 使用自定义构建图像时“BUILD\_CONTAINER\_UNABLE\_TO\_PULL\_IMAGE” (p. 391)
- 错误 “在完成构建之前发现构建容器已死。构建容器已死因为内存不足,或者不支持Docker镜像。ErrorCode 500\*\* (p. 392)
- 错误 运行内部版本时“无法连接到Docker守护程序” (p. 392)
- 错误：“CodeBuild 运行内部版本时遇到问题” (p. 392)
- 错误：“CodeBuild 在创建或更新构建项目时,无权执行:ss:AssumeRole” (p. 393)
- 错误 “调用GetBucketAcl时出错: bucketowner已更改,或者服务角色不再具有调用s3的权限:GetBucketAcl” (p. 393)
- 错误 “上传工件失败: 运行内部版本时arn”无效 (p. 393)
- 错误 “Git克隆失败: 无法访问 'your-repository-URL': SSL证书问题: 自签名证书 (p. 394)
- 错误 运行内部版本时,必须使用指定的端点解决您尝试访问的存储桶 (p. 394)
- 错误 “策略的默认版本不是通过增强零点击角色创建创建的,或者不是通过增强零点击角色创建创建的最新版本。” (p. 394)
- 错误 “此构建映像需要选择至少一个运行时版本。” (p. 395)
- 错误 QUEUED 当构建队列中的构建失败时,显示INSUFFICIENT\_SUBNET" (p. 395)
- 错误 “无法下载缓存: 请求错误: 发送请求失败,原因:x509: 无法加载系统根目录,未提供根目录” (p. 396)
- 错误 “无法从S3下载证书。AccessDenied" (p. 396)
- 错误 “无法找到凭证” (p. 396)
- 在代理服务器中运行 CodeBuild 时出现 RequestError 超时错误 (p. 397)
- bourne shell (sh) 必须存在于构建映像中 (p. 398)
- 警告 “跳过运行时的安装。运行构建时,此构建图像不支持运行时版本选择” (p. 398)
- 错误 BUILD\_CONTAINER\_UNABLE\_TO\_PULL\_IMAGE构建 (p. 398)
- 错误 打开 CodeBuild 控制台 (p. 399)

## 来自错误存储库的 Apache Maven 构建参考构件

问题 当您将Maven与 AWS CodeBuild提供的Java构建环境,Maven从安全的中央Maven存储库提取构建和插件依赖关系,位于 <https://repo1.maven.org/maven2>。即使您构建项目的 pom.xml 文件明确声明会改用其他位置 , 也会发生这种情况。

可能的原因：CodeBuild 提供的 Java 构建环境包含一个名为 `settings.xml` 的文件，该文件预先安装在构建环境的 `/root/.m2` 目录中。该 `settings.xml` 文件包含以下声明，这些声明将指示 Maven 始终从安全的 Maven 中央存储库（网址为 <https://repo1.maven.org/maven2>）中提取构建和插件依赖项。

```
<settings>
  <activeProfiles>
    <activeProfile>securecentral</activeProfile>
  </activeProfiles>
  <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

建议的解决方案。执行以下操作。

1. 向源代码中添加 `settings.xml` 文件。
2. 在此 `settings.xml` 文件中，使用上述 `settings.xml` 格式作为指导，声明您希望 Maven 从哪些存储库中提取构建和插件依赖项。
3. 在构建项目的 `install` 阶段，指示 CodeBuild 将您的 `settings.xml` 文件复制到构建环境的 `/root/.m2` 目录。例如，考虑说明此行为的 `buildspec.yml` 文件中的以下代码段。

```
version 0.2

phases:
  install:
    commands:
      - cp ./settings.xml /root/.m2/settings.xml
```

## 默认情况下，以根用户身份运行构建命令

问题：AWS CodeBuild 以根用户身份运行您的构建命令。即使您的相关构建映像的 `Dockerfile` 将 `USER` 指令设置为另一位用户，也会发生这种情况。

原因 默认情况下，CodeBuild 作为根用户运行所有构建命令。

建议的解决方案。无

## 当文件名为非美国时,内部版本可能会失败。英文字符

问题 当您运行使用文件名包含非美国英文字符(例如,中文字符),内部版本失败。

可能的原因 提供的构建环境 AWS CodeBuild 将其默认区域设置设置为 POSIX。POSIX 本地化设置与 CodeBuild 以及包含非美国英文字符,可能导致相关构建失败。

建议的解决方案。将以下命令添加到 pre\_build 构建规范文件的部分。这些命令使得构建环境使用美国英语UTF-8进行本地化设置,这更兼容 CodeBuild 以及包含非美国英文字符。

对于基于 Ubuntu 的构建环境 :

```
pre_build:  
  commands:  
    - export LC_ALL="en_US.UTF-8"  
    - locale-gen en_US en_US.UTF-8  
    - dpkg-reconfigure locales
```

对于基于 Amazon Linux 的构建环境 :

```
pre_build:  
  commands:  
    - export LC_ALL="en_US.utf8"
```

## 当从 Amazon EC2 Parameter Store 获取参数时 , 构建可能失败

问题 当内部版本尝试获取一个或多个参数的值时,Amazon EC2 参数存储,构建失败 DOWNLOAD\_SOURCE 阶段与错误 Parameter does not exist.

可能的原因 构建项目所依赖的服务角色无权调用 ssm:GetParameters 操作或构建项目使用由生成的服务角色 AWS CodeBuild 并允许呼叫 ssm:GetParameters 操作,但参数的名称不以 /CodeBuild/.

建议的解决方案 :

- 如果服务角色不是由 CodeBuild 生成的 , 请将其定义更新为允许 CodeBuild 调用 ssm:GetParameters 操作。例如 , 以下策略语句允许调用 ssm:GetParameters 操作以获取名称以 /CodeBuild/ 开头的参数 :

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": "ssm:GetParameters",  
      "Effect": "Allow",  
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/CodeBuild/*"  
    }  
  ]  
}
```

- 如果服务角色由 CodeBuild,更新其定义以允许 CodeBuild 访问参数 Amazon EC2 参数存储名称不是以开头 /CodeBuild/。例如,以下策略语句允许呼叫 ssm:GetParameters 获取指定名称参数的操作:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": "ssm:GetParameters",  
            "Effect": "Allow",  
            "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/PARAMETER_NAME"  
        }  
    ]  
}
```

## 无法在 CodeBuild 控制台中访问分支筛选条件

问题 创建或更新 AWS CodeBuild 项目。

可能的原因 分支过滤器选项已弃用。它已被 Webhook 筛选条件组取代，后者可以更好地控制触发新 CodeBuild 中的构建的 Webhook 事件。

建议的解决方案。要迁移在介绍Webhook筛选器之前创建的分支筛选器,请使用 HEAD\_REF 使用正则表达式筛选 ^refs/heads/**branchName\$**。例如,如果您的分支过滤了正则表达式, ^**branchName\$**,然后您放入 HEAD\_REF 过滤器是 ^refs/heads/**branchName\$**。有关详细信息,请参阅 [BitbucketWebHook事件 \(p. 221\)](#) 和 [筛选 GitHub Webhook 事件 \(控制台\) \(p. 230\)](#).

## 无法查看构建是成功还是失败

问题 您无法看到检索到的构建的成功或失败。

可能的原因 未启用报告内部版本状态的选项。

建议的解决方案：启用 报告生成状态 创建或更新 CodeBuild 项目。此选项告知 CodeBuild 在触发构建时报告状态。有关详细信息,请参阅 [报表BuildStatus](#) 在 AWS CodeBuild API参考.

## 无法找到并选择 Windows Server Core 2016 平台的基本映像

问题 无法找到或选择WindowsServerCore2016平台的基础图像。

可能的原因 您正在使用不支持此图像的AWS区域。

建议的解决方案：使用以下支持WindowsServerCore2016平台基础图像的AWS区域之一：

- 美国东部 ( 弗吉尼亚北部 )
- 美国东部 ( 俄亥俄州 )
- 美国东部 ( 俄亥俄州 )
- 美国西部 ( 加利福尼亚北部 )

## 构建规范文件中的前期命令无法被后续命令识别

问题 您的buildspec文件中的一个或多个命令的结果不被同一buildspec文件中的后续命令识别。例如，某个命令可能会设置本地环境变量，但稍后运行的命令可能无法获取该本地环境变量的值。

可能的原因 在buildspec文件版本0.1中，AWS CodeBuild 在构建环境中的默认shell的单独实例中运行每个命令。这表示各个命令独立于其他所有命令而运行。默认情况下，您无法运行依赖于任何先前命令的状态的单个命令。

建议的解决方案：我们建议你使用版本0.2内部版本解决此问题。如果您必须使用构建规范版本 0.1，建议您使用 Shell 命令链接运算符（例如，Linux 中的 `&&`）将多个命令合并为一个命令。或者，您也可以在源代码中包括一个带有多个命令的 Shell 脚本，然后从 buildspec 文件中的单个命令调用该 Shell 脚本。有关更多信息，请参阅 [构建环境中的 Shell 和命令 \(p. 167\)](#)和[构建环境中的环境变量 \(p. 168\)](#)。

## 错误 尝试下载缓存时“访问被拒绝”

问题 当尝试在已启用缓存的构建项目中下载缓存时,您将收到 `Access denied` 错误。

### 可能的原因

- 您刚刚已将缓存配置为您的构建项目的一部分。
- 最近已通过 `InvalidateProjectCache` API 使缓存失效。
- 正由 CodeBuild 使用的服务角色对包含缓存的 S3 存储桶没有 `s3:GetObject` 和 `s3:PutObject` 权限。

建议的解决方案。对于首次使用,在更新缓存配置后立即看到此信息是正常的。如果此错误持续存在，则您应该检查您的服务角色对包含缓存的 S3 存储桶是否具有 `s3:GetObject` 和 `s3:PutObject` 权限。有关详细信息,请参阅 [指定S3权限](#) 在 Amazon S3 开发者指南.

## 错误 使用自定义构建图像时“BUILD\_CONTAINER\_UNABLE\_TO\_PULL\_IMAGE”

问题 当您尝试运行使用自定义构建图像的构建时,该构建失败,出现错误 `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE`.

### 可能的原因

- 构建映像的整体未压缩大小大于构建环境计算类型的可用磁盘空间。要检查构建映像的大小，请使用 Docker 运行 `docker images REPOSITORY:TAG` 命令。有关按计算类型分类的可用磁盘空间的列表，请参阅 [构建环境计算类型 \(p. 166\)](#)。
- AWS CodeBuild 无权从您的 Amazon Elastic Container Registry (Amazon ECR) 中拉取构建映像。
- 您请求的 Amazon ECR 映像在您的 AWS 账户使用的 AWS 区域中不可用。
- 您正在使用不具有公共 Internet 访问权限的 VPC 中的私有注册表。CodeBuild 无法从 VPC 中的私有 IP 地址拉取映像。有关更多信息，请参阅 [适用于 CodeBuild 的私有注册表与 AWS Secrets Manager 示例 \(p. 129\)](#)。 )

### 建议的解决方案：

- 对较大的计算类型使用更多的可用磁盘空间，或者减小自定义构建映像的大小。
- 更新 Amazon ECR 的存储库中的权限，以便 CodeBuild 可以将自定义构建映像拉取到构建环境中。有关更多信息，请参阅 [Amazon ECR 示例 \(p. 50\)](#)。
- 使用位于您的 AWS 账户所使用的 AWS 区域中的 Amazon ECR 映像。
- 如果您在 VPC 中使用私有注册表，请确保 VPC 具有公共 Internet 访问权限。

## 错误 "在完成构建之前发现构建容器已死。构建容器已死因为内存不足,或者不支持Docker镜像。ErrorCode 500\*\*

问题 当您尝试使用Microsoft Windows或Linux容器时 AWS CodeBuild,此错误发生在配置阶段。

可能的原因

- CodeBuild 不支持容器操作系统版本。
- 在容器中指定了 `HTTP_PROXY` 和/或 `HTTPS_PROXY`。

建议的解决方案 :

- 对于 Microsoft Windows , 使用其中容器操作系统版本为 `microsoft/windowsservercore:10.0.x` ( 例如 , `microsoft/windowsservercore:10.0.14393.2125` ) 的 Windows 容器。
- 对于 Linux , 请在 Docker 映像中清除 `HTTP_PROXY` 和 `HTTPS_PROXY` 设置 , 或在构建项目中指定 VPC 配置。

## 错误 运行内部版本时“无法连接到Docker守护程序”

问题 构建失败,您收到类似于 `Cannot connect to the Docker daemon at unix:/var/run/docker.sock. Is the docker daemon running?` 在内部版本日志中。

可能的原因 您没有在特权模式下运行内部版本。

建议的解决方案。按照以下步骤在特权模式下运行构建:

1. 通过以下网址打开 CodeBuild 控制台 : <https://console.aws.amazon.com/codebuild/>。
2. 在导航窗格中选择 Build projects (构建项目) , 然后选择您的生成包。
3. 从 Edit (编辑) 中 , 选择 Environment (环境)。
4. 选择 Override images (覆盖映像) , 然后选择 Environment (环境)。
5. 指定环境映像、操作系统、运行时和映像。这些设置应与失败的构建的设置匹配。
6. 选择 Privileged (特权)。

### Note

默认情况下 , Docker 容器不允许访问任何设备。特权模式将授予构建项目的 Docker 容器访问所有设备的权限。有关更多信息 , 请参阅 Docker 文档网站上的[运行时权限和 Linux 功能](#)。

7. 选择 Update environment (更新环境)。
8. 选择 Start build (启动构建) 来重试您的生成包。

## 错误:"CodeBuild 运行内部版本时遇到问题"

问题 当您尝试运行构建项目时,在构建 PROVISIONING 阶段。

可能的原因 您的内部版本使用的环境变量太大, AWS CodeBuild。 CodeBuild 当所有环境变量(所有名称和值都添加在一起)的长度达到组合最大约5,500个字符时,会引发错误。

建议的解决方案。使用 Amazon EC2 Systems Manager 参数存储用于存储大型环境变量,然后从buildspec 文件检索这些变量。Amazon EC2 Systems Manager Parameter Store 可以存储单个环境变量(名称和值一

起添加),这些变量等于或小于4,096个字符。要存储大型环境变量 ,请参阅 <https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-paramstore.html> 用户指南 中的 Systems Manager Parameter Store 和 Amazon EC2 Systems ManagerSystems Manager Parameter Store 控制台演练。要检索它们 ,请参阅构建规范语法 (p. 137)中的 parameter-store 映射。

## 错误:"CodeBuild 在创建或更新构建项目时,无权执 行:ss:AssumeRole"

问题 当您尝试创建或更新构建项目时,您将收到错误信息 Code: InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::account-ID:role/service-role-name.

可能的原因

- AWS Security Token Service (AWS STS) 已在您尝试创建或更新构建项目的 AWS 区域停用。
- 与构建项目相关联的 AWS CodeBuild 服务角色不存在 ,或没有足够的权限来信任 CodeBuild。

建议的解决方案 :

- 确保 AWS STS 已经为您尝试创建或更新构建项目的 AWS 区域激活。有关详细信息,请参阅 [激活和停用 AWS STS 在 AWS 区域](#) 在 IAM 用户指南。
- 确保您的 AWS 账户中存在目标 CodeBuild 服务角色。如果您没有使用控制台 ,请确保在创建或更新构建项目时没有拼错服务角色的 Amazon 资源名称 (ARN)。
- 确保目标 CodeBuild 服务角色具有足够的权限来信任 CodeBuild。有关更多信息 ,请参阅 [创建 CodeBuild 服务角色 \(p. 357\)](#) 中的信任关系策略声明。

## 错误 "调用GetBucketAcl时出错: bucketowner 已更改,或者服务角色不再具有调用s3的权 限:GetBucketAcl"

问题 运行内部版本时,收到有关S3bucket所有权变更的错误, GetBucketAcl 权限。

可能的原因 您添加了 s3:GetBucketACL 和 s3:GetBucketLocation 对您的 IAM 角色。这些权限可保护您项目的 S3 存储桶 ,并确保只有您可以访问它。添加完这些权限后 ,S3 存储桶的拥有者会发生更改。

建议的解决方案。 验证您是S3bucket的所有者,然后将权限添加到 IAM 角色。有关更多信息 ,请参阅 [对 S3 存储桶的安全访问 \(p. 328\)](#)。 )

## 错误 "上传工件失败: 运行内部版本时arn"无效

问题 当您运行构建时, UPLOAD\_ARTIFACTS 构建阶段失败,出现错误 Failed to upload artifacts: Invalid arn.

可能的原因 您的S3输出存储桶(存储桶 AWS CodeBuild 存储其从内部版本的输出)位于 AWS 区域不同于 CodeBuild 构建项目。

建议的解决方案。 更新构建项目的设置以指向与构建项目位于同一AWS区域中的输出存储桶。

## 错误 “Git 克隆失败: 无法访问 'your-repository-URL': SSL 证书问题: 自签名证书”

问题 当您尝试运行构建项目时,构建失败,出现错误。

可能的原因 您的源存储库具有一个自签名证书 , 但您在构建项目的过程中未选择从您的 S3 存储桶安装此证书。

建议的解决方案 :

- 编辑您的项目。对于 Certificate , 选择 Install certificate from S3。对于 Bucket of certificate , 选择存储您的 SSL 证书的 S3 存储桶。对于 Object key of certificate (证书的对象键) , 键入您的 S3 对象键的名称。
- 编辑您的项目。选择 Insecure SSL (不安全的 SSL) , 在连接到您的 GitHub Enterprise Server 项目存储库时忽略 SSL 警告。

Note

建议您仅将 Insecure SSL 用于测试。它不应在生产环境中使用。

## 错误 运行内部版本时,必须使用指定的端点解决您尝试访问的存储桶

问题 当您运行构建时, DOWNLOAD\_SOURCE 构建阶段失败,出现错误 The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.

可能的原因 您的预构建源代码存储在S3bucket中,该bucket存储在 AWS 区域不同于 AWS CodeBuild 构建项目。

建议的解决方案。更新构建项目的设置以指向包含预构建源代码的存储桶。确保该存储桶位于与构建项目相同的 AWS 区域中。

## 错误 "策略的默认版本不是通过增强零点击角色创建创建的,或者不是通过增强零点击角色创建创建的最新版本。"

问题 当您尝试更新控制台中的项目时,更新失败,出现错误:

可能的原因

- 您已更新附加到目标 AWS CodeBuild 服务角色的策略。
- 您已选择附加到目标 CodeBuild 服务角色的策略的较早版本。

建议的解决方案 :

- 编辑 CodeBuild 项目 , 并清除 Allow CodeBuild to modify this service role so it can be used with this build project (允许 AWS CodeBuild 修改此服务角色以便它可用于此构建项目) 复选框。验证您正在使用的 CodeBuild 服务角色是否具有足够的权限。如果再次编辑 CodeBuild 项目 , 则必须再次清除此复选框。有关更多信息 , 请参阅 [创建 CodeBuild 服务角色 \(p. 357\)](#)。 )

- 请按照以下步骤操作，编辑 CodeBuild 项目以使用新的服务角色：

1. 打开 IAM 控制台并创建新的服务角色。有关更多信息，请参阅 [创建 CodeBuild 服务角色 \(p. 357\)](#)。)
2. Open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
3. 在导航窗格中，选择 Build projects。
4. 选择构建项目旁边的按钮，选择 Edit (编辑)，然后选择 Environment (环境)。
5. 对于 Service role (服务角色)，选择您创建的角色。
6. 选择 Update environment (更新环境)。

## 错误“此构建映像需要选择至少一个运行时版本。”

**问题** 当您运行构建时, DOWNLOAD\_SOURCE 构建阶段失败,出现错误 YAML\_FILE\_ERROR: This build image requires selecting at least one runtime version.

**可能的原因** 您的内部版本使用1.0或更高版本 Amazon Linux 2 (AL2)标准图像,或Ubuntu标准图像的2.0或更高版本,未在buildspec文件中指定运行时。

**建议的解决方案。** 如果您使用 `aws/codebuild/standard:2.0` CodeBuild 管理的图像,您必须在中指定运行时版本 `runtime-versions` 构建规范文件的部分。例如，您可以对使用 PHP 的项目使用以下 buildspec 文件：

```
version: 0.2

phases:
  install:
    runtime-versions:
      php: 7.3
  build:
    commands:
      - php --version
  artifacts:
    files:
      - README.md
```

### Note

如果您指定 `runtime-versions` 部分并使用除 Ubuntu 标准映像 2.0 及更高版本或 Amazon Linux 2 (AL2) 标准映像 1.0 及更高版本以外的其他映像，构建过程将会发出警告“Skipping install of runtimes. Runtime version selection is not supported by this build image。”

有关更多信息，请参阅 [Specify runtime versions in the buildspec file](#)。)

## 错误 QUEUED 当构建队列中的构建失败时,显示 INSUFFICIENT\_SUBNET"

**问题** 构建队列中的构建失败,错误类似于 `QUEUED: INSUFFICIENT_SUBNET`。

**可能的原因** 为VPC指定的IPv4CIDR块使用保留的IP地址。每个子网 CIDR 块中的前四个 IP 地址和最后一个 IP 地址无法供您使用，而且无法分配到一个实例。例如，在具有 CIDR 块 `10.0.0.0/24` 的子网中，以下五个 IP 地址是保留的：

- `10.0.0.0`: : 网络地址。
- `10.0.0.1`: : 由 AWS 预留，用于 VPC 路由器。

- 10.0.0.2: : 由 AWS 预留。DNS 服务器的 IP 地址始终为 VPC 网络范围的基址 + 2；但是，我们也保留了每个子网范围基址 + 2 的 IP 地址。对于包含多个 CIDR 块的 VPC，DNS 服务器的 IP 地址位于主要 CIDR 中。有关详细信息,请参阅 [Amazon DNS 服务器](#) 在 Amazon VPC 用户指南.
- 10.0.0.3: : 由 AWS 预留，供将来使用。
- 10.0.0.255: : 网络广播地址。我们不支持 VPC 中的广播。该地址是预留的。

建议的解决方案：检查VPC是否使用保留的IP地址。将任何预留的 IP 地址替换为未预留的 IP 地址。有关详细信息,请参阅 [VPC和子网规模](#) 在 Amazon VPC 用户指南.

## 错误“无法下载缓存: 请求错误: 发送请求失败,原因:x509: 无法加载系统根目录,未提供根目录”

问题 当您尝试运行构建项目时,构建失败,出现错误。

可能的原因 您将缓存配置为构建项目的一部分,并使用旧的Docker镜像(包括过期根证书)。

建议的解决方案。更新您的 AWS CodeBuild 项目中使用的 Docker 镜像。有关更多信息，请参阅 [CodeBuild 提供的 Docker 映像 \(p. 159\)](#)。)

## 错误“无法从S3下载证书。AccessDenied”

问题 当您尝试运行构建项目时,构建失败,出现错误。

可能的原因

- 您选择了错误的证书 S3 存储桶。
- 您输入了错误的证书对象键。

建议的解决方案：

- 编辑您的项目。对于 Bucket of certificate，选择存储您的 SSL 证书的 S3 存储桶。
- 编辑您的项目。对于 Object key of certificate (证书的对象键)，键入您的 S3 对象键的名称。

## 错误 "无法找到凭证"

问题 当您尝试运行 AWS CLI,使用 AWS SDK,或者调用其他类似组件作为构建的一部分,您就会收到与 AWS CLI, AWS SDK或组件。例如，您可能会收到构建错误，如 `Unable to locate credentials`。

可能的原因

- 构建环境中的 AWS CLI、AWS 开发工具包或组件的版本与 AWS CodeBuild 不兼容。
- 您将在使用 Docker 的构建环境内运行 Docker 容器，并且此容器在默认情况下无权访问 AWS 凭证。

建议的解决方案：

- 确保您的构建环境具有以下版本或更高版本的 AWS CLI、AWS 开发工具包或组件。
  - AWS CLI : 1.10.47
  - 适用于 C++ 的 AWS 软件开发工具包 0.2.19(2019年)

- 适用于 Go 的 AWS 开发工具包 1.2.5
  - 适用于 Java 的 AWS 开发工具包 1.11.16(11.1.16)
  - AWS JavaScript 的 SDK: 2.4.7
  - AWS PHP 的 SDK: 3.18.28
  - AWS Python 的 SDK(Boto3): 1.4.0
  - 适用于 Ruby 的 AWS 开发工具包 2.3.22
  - Botocore 1.4.37
  - CoreCLR : 3.2.6-beta
  - Node.js 2.4.7
- 如果您需要在某个构建环境中运行某个 Docker 容器，而该容器需要 AWS 凭证，则必须将此凭证从该构建环境传递到该容器。在您的构建规范文件中，包含与以下内容类似的 Docker run 命令。此示例使用 aws s3 ls 命令列出您的可用 S3 存储桶。-e 选项将为容器传递访问 AWS 凭证所需的环境变量。

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag
aws s3 ls
```

- 如果您要构建 Docker 映像并且该构建需要 AWS 凭证（例如，从 Amazon S3 下载文件），则必须将凭证从构建环境传递到 Docker 构建过程，如下所示。

1. 在您的源代码的用于 Docker 映像的 Dockerfile 中，指定以下 ARG 指令。

```
ARG AWS_DEFAULT_REGION
ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

2. 在您的构建规范文件中，包含与以下内容类似的 Docker build 命令。--build-arg 选项将为 Docker 构建过程设置访问 AWS 凭证所需的环境变量。

```
docker build --build-arg AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION --build-arg
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI -
t your-image-tag .
```

## 在代理服务器中运行 CodeBuild 时出现 RequestError 超时错误

问题 您收到 RequestError 错误类似于以下其中一项：

- RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout 从 CloudWatch Logs.
- Error uploading artifacts: RequestError: send request failed caused by: Put https://your-bucket.s3.your-aws-region.amazonaws.com/\*: dial tcp 52.219.96.208:443: connect: connection refused 从 Amazon S3.

### 可能的原因

- ssl-bump 未正确配置。
- 贵组织的安全策略不允许您使用 ssl\_bump。
- 您的 buildspec 文件没有使用 proxy 元素指定的代理设置。

### 建议的解决方案：

- 确保 `ssl-bump` 已正确配置。如果您对代理服务器使用 Squid，请参阅 [将 Squid 配置为显式代理服务器 \(p. 184\)](#)。
- 按照以下步骤操作，为 Amazon S3 和 CloudWatch Logs 使用私有终端节点：
  - 在您的私有子网路由表中，删除您添加的、将发往 Internet 的流量路由到您的代理服务器的规则。有关信息，请参阅 [在VPC中创建子网](#) 在 Amazon VPC 用户指南。
  - 创建私有 Amazon S3 终端节点和 CloudWatch Logs 终端节点，然后将其与您的 Amazon VPC 私有子网关联。有关信息，请参阅 [VPC端点服务\(AWSPrivateLink\)](#) 在 Amazon VPC 用户指南。
  - 确认 Amazon VPC 中的 Enable Private DNS Name (启用私有 DNS 名称) 处于选中状态。有关详细信息，请参阅 [创建接口端点](#) 在 Amazon VPC 用户指南。
- 如果您不将 `ssl-bump` 用于显式代理服务器，请使用 `proxy` 元素将代理配置添加到您的 `buildspec` 文件。有关更多信息，请参阅 [在显式代理服务器中运行 CodeBuild \(p. 184\)](#) 和 [构建规范语法 \(p. 137\)](#)。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

## bourne shell (sh) 必须存在于构建映像中

**问题** 您正在使用并非由 AWS CodeBuild,而您的构建失败 `Build container found dead before completing the build.`

**可能的原因** 伯恩贝壳(sh)不包含在您的构建图像中。CodeBuild 需求 sh 运行内部命令和脚本。

**建议的解决方案。** 如果 sh 在构建图像中不存在时,请确保在开始使用使用您的图像的任何其他构建之前包含它。(CodeBuild 已经包括 sh 构建图像。)

## 警告 “跳过运行时的安装。运行构建时,此构建图像不支持运行时版本选择”

**问题** 运行内部版本时,内部版本日志包含此警告。

**可能的原因** 您的内部版本不使用 1.0 或更高版本 Amazon Linux 2 (AL2) 标准图像,或 Ubuntu 标准图像的 2.0 或更高版本,运行时在 `runtime-versions` 构建规范文件中的部分。

**建议的解决方案。** 确保您的 `buildspec` 文件不包含 `runtime-versions` 第节。仅当使用 Amazon Linux 2 (AL2) 标准映像或更高版本或者 Ubuntu 标准映像版本 2.0 或更高版本时,才需要 `runtime-versions` 部分。

## 错误

## BUILD\_CONTAINER\_UNABLE\_TO\_PULL\_IMAGE 构建

**问题** 构建时,您收到类似于以下内容的错误:

BUILD\_CONTAINER\_UNABLE\_TO\_PULL\_IMAGE: Unable to pull customer's container image. CannotPullContainerError: a Windows version 10.0.17763-based image is incompatible with a 10.0.14393 host

可能的原因 您已选择WindowsServer2016环境类型(WINDOWS\_CONTAINER),但选择了WindowsServer2019图像。

建议的解决方案：将环境类型更改为 WINDOWS\_SERVER\_2019\_CONTAINER.

## 错误 打开 CodeBuild 控制台

问题 当您打开 CodeBuild 控制台,显示“无法验证JobWorker身份”错误消息。

可能的原因 的 IAM 用于控制台访问的角色具有标记 jobId 作为键。此标记密钥保留给 CodeBuild 如果存在,则会导致此错误。

建议的解决方案。更改任何自定义项 IAM 具有键的角色标签 jobId 使用不同的密钥,例如 jobIdentifier.

# AWS CodeBuild 的配额

下表列出了 AWS CodeBuild 中的当前配额。这些配额适用于每个受支持的 AWS 区域中的每个 AWS 账户，除非另有规定。

## 构建项目

资源	默认值
构建项目描述中允许使用的字符	任何
构建项目名称中允许使用的字符	字母 A-Z 和 a-z、数字 0-9，以及特殊字符 - 和 _
构建项目名称的长度	2 到 255 个字符
构建项目描述的最大长度	255 个字符
最大生成项目数	5,000
可使用 AWS CLI 或 AWS 开发工具包一次请求其相关信息的构建项目的最大数目	100
您可以添加到项目的最大报告数	5
可以与构建项目相关联的最大标签数	50
可以在构建项目中为所有相关构建的构建超时指定的分钟数	5 到 480 (8 个小时)
可以在 VPC 配置下添加的安全组的数量	1 到 5
可以在 VPC 配置下添加的子网的数量	1 到 16

## 构建

资源	默认值
可使用 AWS CLI 或 AWS 开发工具包一次请求其相关信息的构建的最大数目	100
最大并发运行构建数*	60
构建历史记录的最长保留时间	1 年
可以为单个构建的构建超时指定的分钟数	5 到 480 (8 个小时)

\* 最大并发运行构建数的配额因计算类型的不同而有所不同。对于某些平台和计算类型，默认值为 20。对于新账户，配额可以是 1—5。如需请求更高的并发构建配额，或者如果您收到“Cannot have more than X active builds for the account (账户不能有多于 x 个处于活动状态的构建)”错误，请联系 AWS Support。

## 报告

资源	默认值
测试报告创建后可用的最长持续时间	30 天
每个 AWS 账户的最大报告组数	1000
每份报告的最大测试用例数	500

## Tags

标签限制适用于 CodeBuild 构建项目和 CodeBuild 报告组资源上的标签。

资源	默认值
可以与资源相关联的最大标签数	50。标签区分大小写
资源标签键名称	UTF-8 格式的 Unicode 字母、数字、空格和允许使用的字符的任意组合，长度为 1 到 127 个字符。允许使用的字符为 + - = . _ : / @  标签键名称必须是唯一的，而且每个键只能有一个值。标签键名称不能： <ul style="list-style-type: none"><li>以 aws: 开头</li><li>只包含空格</li><li>以空格结尾</li><li>包含表情符号或以下任意字符：? ^ * [ \ ~ ! # \$ % &amp; * ( ) &gt; &lt;   " ' ` [ ] { } ;</li></ul>
资源标签值	UTF-8 格式的 Unicode 字母、数字、空格和允许使用的字符的任意组合，长度为 0 到 255 个字符。允许使用的字符为 + - = . _ : / @  一个键只能有一个值，但许多键可以具有相同的值。标签键值不能包含表情符号或以下任意字符： ? ^ * [ \ ~ ! # \$ % & * ( ) > <   " ' ` [ ] { } ;

# 适用于 Windows 的 AWS CodeBuild 的第三方说明

在您使用适用于 Windows 的 CodeBuild 构建时，可以选择使用一些第三方软件包和模块，使您可以构建运行在 Microsoft Windows 操作系统上并与一些第三方产品互操作的应用程序。以下列表包含适用的第三方法律条款，可管理您对指定的第三方程序包和模块的使用。

## 主题

- 1) 基本 Docker 映像—windowsservercore (p. 402)
- 2) 基于 Windows 的 Docker 映像—choco (p. 403)
- 3) 基于 Windows 的 Docker 映像—git - 版本 2.16.2 (p. 403)
- 4) 基于 Windows 的 Docker 映像—microsoft-build-tools - 版本 15.0.26320.2 (p. 403)
- 5) 基于 Windows 的 Docker 映像—nuget.commandline - 版本 4.5.1 (p. 405)
- 7) 基于 Windows 的 Docker 映像—netfx-4.6.2-devpack (p. 405)
- 8) 基于 Windows 的 Docker 映像—visualfsharp tools , v 4.0 (p. 406)
- 9) 基于 Windows 的 Docker 映像—netfx-pcl-reference-assemblies-4.6 (p. 407)
- 10) 基于 Windows 的 Docker 映像—visualcppbuildtools v 14.0.25420.1 (p. 408)
- 11) 基于 Windows 的 Docker 映像—microsoft-windows-netfx3-on-demand-package.cab (p. 410)
- 12) 基于 Windows 的 Docker 映像—dotnet-sdk (p. 411)

## 1) 基本 Docker 映像—windowsservercore

( 许可条款位于：<https://hub.docker.com/r/microsoft/windowsservercore/> )

许可证 通过请求和使用此容器操作系统映像处理Windows容器，您确认、理解并同意以下补充许可条款：

### MICROSOFT 软件补充许可条款

#### 容器操作系统映像

Microsoft Corporation ( 或者您所在地的其附属公司 ) ( 简称为“我们”或“Microsoft” ) 将此容器操作系统映像补充 ( 下称“补充” ) 的许可授予您。根据授予的许可，您可以将此补充与基本主机操作系统软件 ( 下称“主机软件” ) 一起使用，其目的仅用于帮助您在主机软件中运行容器功能。主机软件许可条款适用于您对补充的使用。如果您未获得主机软件的许可，就不能使用它。您可以将此补充用于主机软件的每个授予有效许可的副本。

其他许可要求和/或使用权利。

您按照前述段落中的规定使用补充许可可能会导致创建或修改容器映像 ( 下称“容器映像” )，而其中包含特定补充组件。为明确起见，容器映像是独立的，不同于虚拟机映像或虚拟设备映像。根据这些许可条款，在遵循下列条件的情况下，我们授予您重新分发此类补充组件的有限许可：

- (i) 您仅可以在自己的容器映像中，将补充组件作为映像的一部分使用，
- (ii) 只要您的容器映像中的重要主功能与补充在实质上是分离且不同的，您就可以在容器映像中使用此类补充；以及

(iii) 您同意在您的容器映像中包括这些许可条款（或者我们或托管商要求的类似条款），用于对您的最终用户可能使用补充组件正确授予许可。

我们保留未在此处明确授予的所有其他权限。

使用本补充内容即表示您接受这些条款。如果您不接受这些条款，请勿使用本补充内容。

作为适用于 Windows 容器的此容器操作系统映像补充许可条款的一部分，您还需要遵守基础 Windows Server 主机软件许可条款，该条款位于：<https://www.microsoft.com/en-us/useterms>。

## 2) 基于 Windows 的 Docker 映像—choco

（许可条款位于：<https://github.com/chocolatey/chocolatey.org/blob/master/LICENSE.txt>）

版权所有 – Present RealDimensions Software, LLC

根据 Apache 许可版本 2.0 授予许可（简称“许可”），如果不遵守许可，您不可使用这些文件。您可以在以下地址获取许可的副本

<http://www.apache.org/licenses/LICENSE-2.0>

除非适用的法律要求或以书面方式表示同意，否则，根据该许可分发的软件按“原样”分发，无任何明示或暗示的保证或条件。请参阅许可证以了解在许可证下特定语言的适用权限和限制。

## 3) 基于 Windows 的 Docker 映像—git - 版本 2.16.2

（许可条款位于：<https://chocolatey.org/packages/git/2.16.2>）

根据 GNU General Public License 版本 2 授予许可，该许可位于：<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html>。

## 4) 基于 Windows 的 Docker 映像—microsoft-build-tools - 版本 15.0.26320.2

（许可条款位于：<https://www.visualstudio.com/license-terms/mt171552/>）

MICROSOFT VISUAL STUDIO 2015 扩展、VISUAL STUDIO SHELL 和 C++ 可重新分发

-----  
这些许可条款是 Microsoft Corporation（或您所在地的其附属公司）与您达成的协议。它们适用于以上所述软件。这些条款还适用于软件的任意 Microsoft 服务或更新，除非另有附加条款。

-----  
如果您遵循这些许可条款，您将拥有以下权利。

1. 安装和使用权利。您可以安装和使用该软件任意数量的副本。
2. 特定组件的条款。

a. 实用程序。软件可能包含UtilitiesList中的一些项目,位于<https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs>。您可以将这些项目(如果包含在软件中)复制到您的或其他第三

方机器上，以调试和部署您使用软件开发的应用程序和数据库。请注意，实用程序设计用于临时使用，Microsoft 可能无法独立于软件的其余部分为实用程序打补丁或进行更新，并且一些实用程序在性质上会使得其他人可以访问安装该实用程序的计算机。因此，在您完成调试或部署应用程序及数据库之后，您应删除已安装的所有实用程序。对于任何第三方使用或访问您安装在任意计算机上的实用程序，Microsoft 不承担任何责任。

- b. Microsoft 平台。软件可能包括来自 Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office 和 Microsoft SharePoint 的组件。这些组件受独立协议及自己的产品支持政策控制，如位于组件安装目录中或随软件提供的“Licenses”文件夹中的许可条款所述。
- c. 第三方组件。软件可能会包括第三方组件，这些组件具有单独的法律声明或受其他协议控制，如随软件提供的 ThirdPartyNotices 文件中所述。即使此类组件受其他协议控制，以下免责声明以及对损害赔偿的限制或排除仍适用。软件还包含在开源许可下授予许可的组件，具有源代码可用性义务。这些许可证的副本（如果适用）包含在 ThirdPartyNotices 文件中。如果相关开源许可证要求，您可以通过向以下人员发送汇款单或支票（金额为 5.00 美元）从我们处获得此源代码：源代码合规团队，Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052。在您付款的备注栏中，请注明以下列出的一个或多个组件的源代码：
  - Remote Tools for Visual Studio 2015；
  - Standalone Profiler for Visual Studio 2015；
  - IntelliTraceCollector for Visual Studio 2015；
  - Microsoft VC++ Redistributable 2015；
  - Multibyte MFC Library for Visual Studio 2015；
  - Microsoft Build Tools 2015；
  - Feedback Client；
  - Visual Studio 2015 Integrated Shell；或
  - Visual Studio 2015 Isolated Shell。

我们还在 <http://thirdpartysource.microsoft.com> 提供了源代码的副本。

- 3. DATA。软件可能会收集有关您以及您使用软件的信息，并将信息发送给 Microsoft。Microsoft 可能会使用此信息来提供服务和改进我们的产品及服务。您可以选择退出其中的多种情况，但并非全部，如产品文档中所述。此外，软件中还有一些功能，可能使您能够从应用程序的用户收集数据。如果您使用这些功能在应用程序中启用数据收集，则必须遵循适用的法律，包括向应用程序的用户提供相应的说明。您可在帮助文档和以下网址的隐私声明中了解有关数据收集和使用的更多信息：<https://privacy.microsoft.com/en-us/privacystatement>。您使用软件操作即表明您同意这些做法。
- 4. 许可范围。该软件授予许可，而非销售。本协议仅向您提供使用软件的一些权利。Microsoft 保留所有其他权利。除非适用法律在此限制之外赋予您更多的权利，否则您只能在本协议中明确允许的情况下使用该软件。在使用时，您必须遵守软件中仅允许您以特定方式使用它的任意技术限制。您不能
  - 规避软件中的任何技术限制；
  - 逆向工程、反编译或反汇编软件，或者尝试这样做（除非且仅限于在第三方许可条款要求的范围内，该条款规定软件中可能包含的特定开源组件的使用）
  - 删除、最小化、阻止或修改软件中 Microsoft 或其供应商的通知；
  - 以违反法律的方式使用软件；或者
  - 共享、发布、租借或租用软件，或者将软件独立托管为解决方案供其他人使用。
- 5. 出口限制。您必须遵守所有适用于该软件的国内和国际出口法律及法规，这包括对目的地、最终用户和最终用途的限制。有关出口限制的更多信息，请访问 ([aka.ms/exporting](http://aka.ms/exporting))。
- 6. 支持服务。由于此软件“按原样”提供，我们可能不会为它提供支持服务。
- 7. 完整协议。本协议以及您使用的补充、更新、基于 Internet 的服务和支持服务的条款是本软件和支持服务的完整协议。
- 8. 适用的法律。如果您在美国购买本软件，华盛顿州法律管辖对本协议的解释以及违反协议的索赔，您居住州的法律适用于所有其他索赔。如果您在任何其他国家/地区购买本软件，则适用该国家/地区的法律。
- 9. 消费者权利；区域差异。本协议描述了特定法律权利。根据所在的州或国家/地区，您可能拥有其他权利，包括消费者权利。除了与 Microsoft 的关系之外，您还可能对与您购买该软件的一方的拥有相应权利。如

果您所在的州或国家/地区的法律不允许，本协议不会更改这些其他权利。例如，如果您在以下区域之一购买了本软件，或者有强制性国家/地区法律适用，则以下条款适用于您：

- a. 澳大利亚. 您在澳大利亚消费者法下获得了法定担保，本协议中的任何内容都无意影响这些权利。
- b. 加拿大. 如果您在加拿大购买此软件，您可通过关闭自动更新功能、断开设备与 Internet 的连接（但是，在您重新连接到 Internet 时，软件将恢复检查和安全更新）或者卸载软件来停止接收更新。产品文档（如果有）可能会说明如何关闭特定设备或软件的更新。
- c. 德国和奥地利。
  - i. 担保。正确授予许可的软件，将基本按照随该软件所提供的任意 Microsoft 材料中所述执行。但是，Microsoft 不提供任何与所许可软件相关的合同担保。
  - ii. 责任限制。在出现故意行为、重大过失、基于产品责任法的索赔时，以及出现死亡、人身伤害或物理伤害时，Microsoft 根据成文法律承担责任。根据前述条款 (ii)，Microsoft 仅在违背了实质性合同义务时，在承担责任有助于正当履行此协议时，违反许可会危害到此协议的目的时，以及符合当事人值得长期信任的情况下（称为“基本义务”），Microsoft 才会承担轻微过失责任。在其他轻微过失的情况下，Microsoft 不承担轻微过失的责任。

10免责声明。本软件按“原样”授予许可。您自行承担使用它的风险。MICROSOFT 不提供任何明示的担保、保证或条件。在您的当地法律允许的范围内，MICROSOFT 排除了对适销性、特定用途的适用性和非侵权的默示保证。

11损害赔偿的限制和排除。您可以直接损害 RECOVER FROM MICROSOFT 及其供应商 ONLY UP TO 美国 5.00 USD。您不能收回任何其他损害，包括后果性、利润损失、特殊、间接或附带损害。此限制适用于 (a) 第三方 Internet 站点或第三方应用程序上与软件、服务、内容（包括代码）相关的任意内容；(b) 违反合同；违反担保、保证或条件；严格责任；疏忽；或法律允许情况下其他侵权行为的索赔。

即使 Microsoft 已知或者应该知道造成损害的可能性，此条款仍适用。由于您所在的国家/地区可能不允许排除或限制事故损害、后果性损害或其他损害，上述限制或排除可能不适用于您。

EULAID: VS2015\_更新3\_壳牌编辑\_<ENU>

## 5) 基于 Windows 的 Docker 映像—nuget.commandline - 版本 4.5.1

（许可条款位于：<https://github.com/NuGet/Home/blob/dev/LICENSE.txt>）

版权所有 (c) .NET Foundation。保留所有权利。

根据 Apache 许可版本 2.0 授予许可（简称“许可”），如果不遵守许可，您不可使用这些文件。您可以在以下地址获取许可的副本

<http://www.apache.org/licenses/LICENSE-2.0>

除非适用的法律要求或以书面方式表示同意，否则，根据该许可分发的软件按“原样”分发，无任何明示或暗示的保证或条件。请参阅许可证以了解在许可证下特定语言的适用权限和限制。

## 7) 基于 Windows 的 Docker 映像—netfx-4.6.2-devpack

MICROSOFT 软件补充许可条款

用于 MICROSOFT WINDOWS 操作系统的 .NET FRAMEWORK 和相关语言包

Microsoft Corporation ( 或您所在地的其附属公司 ) 向您授予此补充的许可。如果您获得使用 Microsoft Windows 操作系统软件 ( 下称“软件” ) 的许可，则可以使用此补充。如果您未获得软件的许可，就不能使用它。您可以将此补充用于软件的每个授予有效许可的副本。

以下许可条款描述了此补充的额外使用条款。软件的这些条款和许可条款适用于您对补充的使用。如果存在冲突，这些补充许可条款适用。

使用本补充条款，即表示您接受这些条款。如果您不接受，请勿使用本补充文件。

如果您遵循这些许可条款，您将拥有以下权利。

1. 可分发代码。补充资料由可分发代码组成。“可分发代码”是在您遵守以下条款的情况下，允许您在自己开发的程序中分发的代码。
  - a. 使用和分发权利。
    - 您可以复制和分发本补充的对象代码形式。
    - 第三方分发。您可以允许程序分发者复制和分发可分发代码作为这些程序的一部分。
  - b. 分销要求。对于您分发的任何可分发代码，您必须
    - 在程序中向其添加重要主功能；
    - 对于文件扩展名为 .lib 的任意可分发代码，仅分发通过您程序的链接器运行此类可分发代码的结果；
    - 仅在可分发代码未经修改的情况下，将其作为安装程序的一部分分发；
    - 要求分发者和外部最终用户同意不低于本协议要求的保护条款；
    - 在您的程序上显示有效的版权声明；以及
    - 对于与分发或使用您的程序相关的索赔，为 Microsoft 辩护、补偿和使其免受损害，包括律师费。
  - c. 分销限制。您可能没有
    - 更改可分发代码中的任何著作权、商标或专利通知；
    - 在您的程序名称中使用 Microsoft 商标，或者以暗示程序来自 Microsoft 或者得到 Microsoft 认可的方式使用 Microsoft 商标；
    - 将可分发代码分发在 Windows 平台之外的平台上运行；
    - 在可分发代码中包括恶意、欺骗性或者非法程序；或者
    - 修改或分发任意可分发代码的源代码，以使其任何部分都受限于排除许可。排除许可是针对使用、修改或分发的条件，要求
      - 代码以源代码的形式公布或分发；或
      - 其他人有权修改它。
2. 补充的支持服务。Microsoft 根据 [www.support.microsoft.com/common/international.aspx](http://www.support.microsoft.com/common/international.aspx) 中所述向本软件提供支持服务。

## 8) 基于 Windows 的 Docker 映像— visualfsharptools , v 4.0

(许可条款可在以下网址获取: <https://github.com/dotnet/fsharp/blob/main/LICENSE.txt>)

版权所有 (c) Microsoft Corporation. 保留所有权利。

根据 Apache 许可版本 2.0 授予许可 ( 简称“许可” )，如果不遵守许可，您不可使用这些文件。您可以在以下地址获取许可的副本

<http://www.apache.org/licenses/LICENSE-2.0>

除非适用的法律要求或以书面方式表示同意，否则，根据该许可分发的软件按“原样”分发，无任何明示或暗示的保证或条件。请参阅许可证以了解在许可证下特定语言的适用权限和限制。

## 9) 基于 Windows 的 Docker 映像—netfx-pcl-reference-assemblies-4.6

Microsoft 软件许可条款

MICROSOFT .NET 可移植类库引用程序集 – 4.6

----

这些许可条款是 Microsoft Corporation ( 或您所在地的其附属公司 ) 与您达成的协议。请仔细阅读。它们适用于以上所述软件。该条款也适用于此软件的任意 Microsoft

- 更新,
- 补充、
- 基于 Internet 的服务和
- 支持服务 ,

除非随这些项目另有其他条款。如果是这样，则那些条款也适用。

使用本软件,即表示您接受这些条款。如果您不接受,请勿使用本软件。

----

如果您遵循这些许可条款，您将拥有以下永久权利。

1. 安装和使用权利。您可以安装和使用任何数量的软件副本来设计、开发和测试您的程序。
2. 其他许可要求和/或使用权利。
  - a. 可分发代码。您可以在您开发的开发人员工具程序中分发本软件，以便您程序的客户可以开发可用于任何设备或操作系统的可移植库，前提是您遵守以下条款。
    - i. 使用和分发权利。软件为“可分发代码”。
      - 可分发代码。您可以复制和分发本软件的对象代码形式。
      - 第三方分发。您可以允许程序分发者复制和分发可分发代码作为这些程序的一部分。
    - ii. 分销要求。对于您分发的任何可分发代码,您必须
      - 在程序中向其添加重要主功能；
      - 要求分发者和您的客户同意不低于本协议要求的保护条款；
      - 在您的程序上显示有效的版权声明；以及
      - 对于与分发或使用您的程序相关的索赔，为 Microsoft 辩护、补偿和使其免受损害，包括律师费。
    - iii. 分销限制。您可能没有
      - 更改可分发代码中的任何著作权、商标或专利通知；
      - 在您的程序名称中使用 Microsoft 商标，或者以暗示程序来自 Microsoft 或者得到 Microsoft 认可的方式使用 Microsoft 商标；
      - 在可分发代码中包括恶意、欺骗性或者非法程序；或者
      - 修改或分发可分发代码，以使其任何部分都受限于排除许可。排除许可是针对使用、修改或分发的条件，要求
        - 代码以源代码的形式公布或分发；或
        - 其他人有权修改它。

3. 许可范围。该软件授予许可，而非销售。本协议仅向您提供使用软件的一些权利。Microsoft 保留所有其他权利。除非适用法律在此限制之外赋予您更多的权利，否则您只能在本协议中明确允许的情况下使用该软件。在使用时，您必须遵守软件中仅允许您以特定方式使用它的任意技术限制。您不能
    - 规避软件中的任何技术限制；
    - 逆向工程、反编译或反汇编本软件，尽管有此限制，但在适用法律明确允许的范围内可以执行上述操作；
    - 发布本软件供其他人复制；或者
    - 出租、租赁或出借本软件。
  4. 反馈。您可以提供关于本软件的反馈。如果您向 Microsoft 提供关于本软件的反馈，则表示您向 Microsoft 免费提供以任何方式和任何用途使用、共享和商业化您的反馈的权利。您还可以免费向第三方提供其产品、技术和服务所需的任何专利权，以便使用包含反馈的 Microsoft 软件或服务的任何特定部分或者与之进行交互。如果反馈受下面这样的许可证的约束，则您不能提供反馈：该许可证要求 Microsoft 向第三方提供其软件或文档的许可，因为我们在相应软件或文档中包含了您的反馈。这些权利不受本协议的约束。
  5. 转让给第三方。本软件的第一个用户可以将本软件和本协议直接转让给第三方。在转让之前，该第三方必须同意本协议适用于本软件的转让和使用。第一个用户在将本软件独立于设备进行转让之前，必须先卸载本软件。第一个用户不能保留任何副本。
  6. 出口限制。本软件受美国出口法律和法规的约束。您必须遵守适用于本软件的所有国内和国际出口法律和法规。这些法律包括对目的地、最终用户和最终用途的限制。有关更多信息，请参阅 [www.microsoft.com/exporting](http://www.microsoft.com/exporting)。
  7. 支持服务。由于此软件“按原样”提供，我们可能不会为它提供支持服务。
  8. 完整协议。本协议以及您使用的补充、更新、基于 Internet 的服务和支持服务的条款是本软件和我们提供的任何支持服务的完整协议。
  9. 适用的法律。
    - a. 美国。如果您在美国购买本软件，华盛顿州法律管辖本协议的解释，并适用于违反本协议的索赔，无论法律原则是否冲突都是如此。您居住的州的法律管辖所有其他索赔，包括根据国家消费者保护法、不正当竞争法和侵权行为提起的索赔。
    - b. 在美国以外的国家或地区。如果您在任何其他国家/地区购买本软件，则适用该国家/地区的法律。
  10. 法律效力。本协议描述了特定法律权利。根据贵国法律，您可能拥有其他权利。您还可能拥有从其获得本软件的一方的相应权利。如果您所在的国家/地区的法律不允许，根据这些法律，本协议不会更改您的权利。
  11. 免责声明。本软件按“原样”授予许可。您自行承担使用它的风险。MICROSOFT 不提供任何明示的担保、保证或条件。根据所在地区的法律，您可能拥有其他本许可证无法更改的消费者权利或法定担保。在您的当地法律允许的范围内，MICROSOFT 排除了对适销性、特定用途的适用性和非侵权的默示保证。

澳洲—根据澳大利亚消费者法律，您有法定保证，并且本条款中的任何内容均不符合这些权利。
  12. 救济和损害赔偿的限制和排除。您可以直接损害 RECOVER FROM MICROSOFT 及其供应商 ONLY UP TO 美国 5.00 USD。您不能收回任何其他损害，包括后果性、利润损失、特殊、间接或附带损害。
- 此限制适用于
- 与第三方 Internet 站点或第三方程序中的软件、服务、内容（包括代码）相关的任何内容；以及
  - 违反合同；违反担保、保证或条件；严格责任；疏忽；或适用法律允许情况下其他侵权行为的索赔。
- 即使 Microsoft 已知或者应该知道造成损害的可能性，此条款仍适用。由于您所在的国家/地区可能不允许排除或限制事故损害、后果性损害或其他损害，上述限制或排除可能不适用于您。

## 10) 基于 Windows 的 Docker 映像— visualcppbuildtools v 14.0.25420.1

Microsoft Visual C++ Build Tools

Microsoft 软件许可条款

Microsoft Visual C++ Build Tools

-----

这些许可条款是 Microsoft Corporation ( 或您所在地的其附属公司 ) 与您达成的协议。它们适用于以上所述软件。这些条款还适用于软件的任意 Microsoft 服务或更新，除非另有其他条款。

-----

如果您遵循这些许可条款，您将拥有以下权利。

1. 安装和使用权利。
  - a. 可以有一位用户使用本软件的副本来自行开发和测试他们的应用程序。
2. DATA。软件可能会收集有关您以及您使用软件的信息，并将信息发送给 Microsoft。Microsoft 可能会使用此信息来提供服务和改进我们的产品及服务。您可以选择退出其中的多种情况，但并非全部，如产品文档中所述。此外，软件中还有一些功能，可能使您能够从应用程序的用户收集数据。如果您使用这些功能在应用程序中启用数据收集，则必须遵循适用的法律，包括向应用程序的用户提供相应的说明。您可在以下网址的帮助文档和隐私声明中了解有关数据收集和使用的更多信息：<http://go.microsoft.com/fwlink/?LinkId=528096>。您使用软件操作即表明您同意这些做法。
3. 特定组件的条款。
  - a. 构建服务器。本软件可能包含 BuildServer.TXT 文件中列出的一些构建服务器组件，和/或位于遵循此 Microsoft 软件许可条款的 BuildServer 列表中列出的任何文件。如果这些项目包含在本软件中，您可以复制并安装到您的构建计算机。您和您组织中的其他人可以在您的构建计算机上使用这些项目，仅用于编译、构建、验证和归档应用程序，或者作为构建过程的一部分运行质量或性能测试。
  - b. Microsoft 平台。软件可能包括来自 Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office 和 Microsoft SharePoint 的组件。这些组件受独立协议及自己的产品支持政策控制，如位于组件安装目录中或随软件提供的“Licenses”文件夹中的许可条款所述。
  - c. 第三方组件。软件可能会包括第三方组件，这些组件具有单独的法律声明或受其他协议控制，如随软件提供的 ThirdPartyNotices 文件中所述。即使此类组件受其他协议控制，以下免责声明以及对损害赔偿的限制或排除仍适用。
  - d. 程序包管理器。本软件可能包含软件包管理器（如 Nuget），它允许您下载其他 Microsoft 和第三方软件包以供您的应用程序使用。这些软件包遵循它们自己的许可证，而不是本协议。Microsoft 不会分发、许可任何第三方软件包或者为其提供任何担保。
4. 许可范围。该软件授予许可，而非销售。本协议仅向您提供使用软件的一些权利。Microsoft 保留所有其他权利。除非适用法律在此限制之外赋予您更多的权利，否则您只能在本协议中明确允许的情况下使用该软件。在使用时，您必须遵守软件中仅允许您以特定方式使用它的任意技术限制。有关更多信息，请参阅 <https://docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1-installation-and-use-rights>。您不能
  - 规避软件中的任何技术限制；
  - 逆向工程、反编译或反汇编软件，或者尝试这样做，除非且仅限于在第三方许可条款要求的范围内，该条款规定本软件中可能包含的特定开源组件的使用；
  - 删除、最小化、阻止或修改 Microsoft 或其供应商的任何通知；
  - 以违反法律的方式使用软件；或者
  - 共享、发布、租借或租用软件，或者将软件独立托管为解决方案供其他人使用。
5. 出口限制。您必须遵守所有适用于该软件的国内和国际出口法律及法规，这包括对目的地、最终用户和最终用途的限制。有关出口限制的更多信息，请访问 ([aka.ms/exporting](http://aka.ms/exporting))。
6. 支持服务。由于此软件“按原样”提供，我们可能不会为它提供支持服务。
7. 完整协议。本协议以及您使用的补充、更新、基于 Internet 的服务和支持服务的条款是本软件和支持服务的完整协议。

8. 适用的法律。如果您在美国购买本软件，华盛顿州法律管辖对本协议的解释以及违反协议的索赔，您居住州的法律适用于所有其他索赔。如果您在任何其他国家/地区购买本软件，则适用该国家/地区的法律。
9. 消费者权利；区域差异。本协议描述了特定法律权利。根据所在的州或国家/地区，您可能拥有其他权利，包括消费者权利。除了与 Microsoft 的关系之外，您还可能对与您购买该软件的一方的拥有相应权利。如果您所在的州或国家/地区的法律不允许，本协议不会更改这些其他权利。例如，如果您在以下区域之一购买了本软件，或者有强制性国家/地区法律适用，则以下条款适用于您：
  - 澳大利亚. 您在澳大利亚消费者法下获得了法定担保，本协议中的任何内容都无意影响这些权利。
  - 加拿大. 如果您在加拿大购买此软件，您可通过关闭自动更新功能、断开设备与 Internet 的连接（但是，在您重新连接到 Internet 时，软件将恢复检查和安全更新）或者卸载软件来停止接收更新。产品文档（如果有）可能会说明如何关闭特定设备或软件的更新。
  - 德国和奥地利。
  - 担保。正确授予许可的软件，将基本按照随该软件所提供的任意 Microsoft 材料中所述执行。但是，Microsoft 不提供任何与所许可软件相关的合同担保。
  - 责任限制。如果发生故意行为、重大过失、基于“产品责任法案”的索赔，以及在发生死亡或人身或身体伤害的情况下，Microsoft 将依照法定法律承担法律责任。

在遵守上述第 (ii) 条的前提下，如果 Microsoft 违反此类重大合同义务，Microsoft 将仅对轻微过失承担责任，履行这一条有助于本协议的正常履行，违反这一条会危及本协议的用途以及一方可以不断信任的合规性（所谓的“基本义务”）。在其他轻微过失的情况下，Microsoft 不承担轻微过失的责任。

10. 法律效力。本协议描述了特定法律权利。根据您所在州或国家/地区的法律，您可能拥有其他权利。如果您所在州或国家/地区的法律不允许，根据这些法律，本协议不会更改您的权利。在不限制前述条款的情况下，对于澳大利亚，您在澳大利亚消费者法下获得了法定担保，本协议中的任何条款都无意影响这些权利。

11. 免责声明。本软件按“原样”授予许可。您自行承担使用它的风险。MICROSOFT 不提供任何明示的担保、保证或条件。在您的当地法律允许的范围内，MICROSOFT 排除了对适销性、特定用途的适用性和非侵权的默示保证。

12. 损害赔偿的限制和排除。您可以直接损害 RECOVER FROM MICROSOFT 及其供应商 ONLY UP TO 美国 5.00 USD。您不能收回任何其他损害，包括后果性、利润损失、特殊、间接或附带损害。

此限制适用于 (a) 第三方 Internet 站点或第三方应用程序上与软件、服务、内容（包括代码）相关的任意内容；(b) 违反合同；违反担保、保证或条件；严格责任；疏忽；或法律允许情况下其他侵权行为的索赔。

即使 Microsoft 已知或者应该知道造成损害的可能性，此条款仍适用。由于您所在的国家/地区可能不允许排除或限制事故损害、后果性损害或其他损害，上述限制或排除可能不适用于您。

## 11) 基于 Windows 的 Docker 映像—microsoft-windows-netfx3-on-demand-package.cab

MICROSOFT 软件补充许可条款

Microsoft .NET Framework 3.5 SP1，适用于 Microsoft Windows 操作系统

-----  
Microsoft Corporation（或您所在地的其附属公司）向您授予此补充的许可。如果您获得使用 Microsoft Windows 操作系统软件（本补充内容适用）（下称“软件”）的许可，则可以使用本补充内容。如果您未获得软件的许可，就不能使用它。您可以将本补充内容的副本用于软件的每个授予有效许可的副本。

以下许可条款描述了此补充的额外使用条款。软件的这些条款和许可条款适用于您对补充的使用。如果存在冲突，这些补充许可条款适用。

使用本补充条款，即表示您接受这些条款。如果您不接受，请勿使用本补充文件。

如果您遵循这些许可条款，您将拥有以下权利。

1. 补充的支持服务。Microsoft 根据 [www.support.microsoft.com/common/international.aspx](http://www.support.microsoft.com/common/international.aspx) 中所述向本软件提供支持服务。
2. Microsoft .NET 基准测试。本软件包括 Windows 操作系统 (.NET 组件) 的 .NET Framework、Windows Communication Foundation、Windows Presentation Foundation 和 Windows Workflow Foundation 组件。您可以对 .NET 组件执行内部基准测试。您可以披露 .NET 组件的任何基准测试的结果，前提是您必须遵守在以下网址建立的条件：<http://go.microsoft.com/fwlink/?LinkId=66406>。

尽管您可能与 Microsoft 达成任何其他协议，但如果披露了此类基准测试结果，则 Microsoft 有权披露其针对与适用 .NET 组件竞争的产品进行基准测试的结果，前提是该组件符合以下网址建立的相同条件：<http://go.microsoft.com/fwlink/?LinkId=66406>。

## 12) 基于 Windows 的 Docker 映像—dotnet-sdk

( 网址：<https://github.com/dotnet/core/blob/master/LICENSE.TXT> )

MIT 许可证 (MIT)

版权所有 (c) Microsoft Corporation.

特此免费授予任何人获得本软件及相关文档文件 (“软件”) 的副本，以无限制地处理本软件，包括但不限于使用、复制、修改、合并、发布、分发、再许可和/或销售本软件的副本，并允许向其提供了本软件上述权利的人员遵守以下条件：

上述版权声明和本许可声明应包含在本软件的所有副本或主要部分中。

此软件按“原样”提供，无任何明示或暗示的保证，包括但不限于有关适销性、针对特定目的的适用性和不侵权的保证。任何情况下，作者或版权所有者都不应承担任何索赔、损害赔偿或其他责任，无论是因软件或使用或其他软件处理引起的或与其相关的合同行为、侵权行为或其他行为。

# AWS CodeBuild 用户指南文档历史记录

下表列出了自 AWS CodeBuild 上一次发布以来对文档所做的重要更改。如需获取对此文档的更新的通知，您可以订阅 RSS 源。

- 最新 API 版本：2016-10-06
- 最新文档更新：2020 年 7 月 30 日

update-history-change	update-history-description	update-history-date
<a href="#">批量构建 (p. 412)</a>	CodeBuild 现在支持执行项目的并发和协调构建。有关详细信息，请参阅 <a href="#">批量生成 CodeBuild</a> 。	July 30, 2020
<a href="#">代码覆盖报告 (p. 412)</a>	CodeBuild 现在提供代码覆盖报告。有关详细信息，请参阅 <a href="#">代码覆盖报告</a> 。	July 30, 2020
<a href="#">会话管理器 (p. 412)</a>	CodeBuild 现在允许您暂停正在运行的构建，然后使用 AWS Systems Manager SessionManager 连接到构建容器并查看容器的状态。有关详细信息，请参阅 <a href="#">会话管理器</a> 。	July 20, 2020
<a href="#">WindowsServer2019 图像 (p. 412)</a>	CodeBuild 现在提供 WindowsServerCore2019 内部版本图像。有关详细信息，请参阅 <a href="#">CodeBuild 提供的 Docker 镜像</a> 。	July 20, 2020
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持在 buildspec 文件中指定要在构建环境中使用的 shell。有关详细信息，请参阅 <a href="#">构建规范参考</a> 。	June 25, 2020
<a href="#">使用测试框架测试报告 (p. 412)</a>	添加了几个主题，介绍如何使用多个测试框架生成 CodeBuild 测试报告。有关更多信息，请参阅 <a href="#">使用测试框架测试报告</a> 。	May 29, 2020
<a href="#">支持测试报告 (p. 412)</a>	CodeBuild 对测试报告的支持现在已经普遍可用。	May 21, 2020
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持向报告组添加标签。有关更多信息，请参阅 <a href="#">报告组</a> 。	May 21, 2020
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持为 Github 和 Bitbucket 创建 Webhook 筛选条件，仅当 HEAD 提交消息匹配指定表达式时才触发构建操作。有关详细信息，请参阅 <a href="#">GitHub 拉取请求和 Webhook 筛选条件</a>	May 6, 2020

示例 以及 Bitbucket 拉取请求和  
Webhook 筛选条件示例。

新主题 (p. 412)	CodeBuild 现在支持共享构建项目和报告组资源。有关更多信息，请参阅 <a href="#">使用共享项目</a> 和 <a href="#">使用共享报告组</a> 。	December 13, 2019
新增和更新的主题 (p. 412)	CodeBuild 现在支持在构建项目运行期间测试报告。有关更多信息，请参阅 <a href="#">使用测试报告</a> 、 <a href="#">创建测试报告</a> 和 <a href="#">使用 AWS CLI 示例创建测试报告</a> 。	November 25, 2019
对主题进行了更新 (p. 412)	CodeBuild 现在支持 Linux GPU 和 Arm 环境类型以及 2xlarge 计算类型。有关更多信息，请参阅 <a href="#">构建环境计算类型</a> 。	November 19, 2019
对主题进行了更新 (p. 412)	CodeBuild 现在支持所有构建的构建编号、导出环境变量和 AWS Secrets Manager 集成。有关详细信息,请参阅 <a href="#">导出的变量</a> 和 <a href="#">密钥管理器 在 构建规范语法</a> .	November 6, 2019
新主题 : (p. 412)	CodeBuild 现在支持通知规则。您可以使用通知规则向用户通知构建项目中的重要更改。有关更多信息 , 请参阅 <a href="#">创建通知规则</a> 。	November 5, 2019
对主题进行了更新 (p. 412)	CodeBuild 现在支持安卓版本 29 和 Go 版本 1.13 运行时。有关更多信息 , 请参阅 <a href="#">CodeBuild 提供的 Docker 映像</a> 和 <a href="#">构建规范语法</a> 。	September 10, 2019
对主题进行了更新 (p. 412)	在创建项目时 , 您现在可以选择 Amazon Linux 2 (AL2) 托管映像。有关更多信息 , 请参阅 <a href="#">CodeBuild 提供的 Docker 映像</a> 和 <a href="#">CodeBuild 的构建规范文件示例</a> 中的运行时版本。	August 16, 2019
对主题进行了更新 (p. 412)	创建项目时 , 您现在可以选择禁用 S3 日志的加密 , 并且如果使用基于 Git 的源存储库 , 则还可以包括 Git 子模块。有关更多信息 , 请参阅 <a href="#">在 CodeBuild 中创建构建项目</a> 。	March 8, 2019
新主题 : (p. 412)	CodeBuild 现在支持本地缓存。创建构建时 , 可以在四种模式中的一种或多种模式中指定本地缓存。有关更多信息 , 请参阅 <a href="#">CodeBuild 中的构建缓存</a> 。	February 21, 2019

<a href="#">新主题 (p. 412)</a>	CodeBuild 现在支持 Webhook 筛选条件组以指定触发构建的事件。有关更多信息，请参阅 <a>筛选 GitHub Webhook 事件</a> 和 <a>筛选 Bitbucket Webhook 事件</a> 。	February 8, 2019
<a href="#">新主题 : (p. 412)</a>	现在，CodeBuild 用户指南显示了如何将 CodeBuild 与代理服务器结合使用。有关更多信息，请参阅 <a>将 CodeBuild 与代理服务器结合使用</a> 。	February 4, 2019
<a href="#">支持专用 Docker 镜像仓库 (p. 412)</a>	CodeBuild 现在支持使用在专用注册表中存储的 Docker 映像作为您的运行时环境。有关更多信息，请参阅 <a>私有注册表与 AWS Secrets Manager 示例</a> 。	January 24, 2019
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持使用另一 AWS 账户中的 Amazon ECR 映像。有多个主题进行了更新，以反映此更改，包括 <a>CodeBuild 的 Amazon ECR 示例</a> 、 <a>创建构建项目</a> 和 <a>创建 CodeBuild 服务角色</a> 。	January 24, 2019
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持使用访问令牌连接到 GitHub（使用个人访问令牌）和 Bitbucket（使用应用程序密码）存储库。有关更多信息，请参阅 <a>创建构建项目（控制台）</a> 和 <a>将访问令牌与源提供商结合使用</a> 。	December 6, 2018
<a href="#">对主题进行了更新 (p. 412)</a>	CodeBuild 现在支持新的构建指标，这些指标用于衡量构建中每个阶段的持续时间。有关更多信息，请参阅 <a>CodeBuild CloudWatch 指标</a> 。	November 15, 2018
<a href="#">VPC 终端节点策略主题 (p. 412)</a>	CodeBuild 的 Amazon VPC 终端节点现在支持策略。有关更多信息，请参阅 <a>为 CodeBuild 创建 VPC 终端节点策略</a> 。	November 9, 2018
<a href="#">更新的内容 (p. 412)</a>	更新了主题以反映新控制台体验。	October 30, 2018
<a href="#">Amazon EFS 示例 (p. 412)</a>	CodeBuild 可以在生成期间使用项目的构建规范文件中的命令挂载 Amazon EFS 文件系统。有关更多信息，请参阅 <a>CodeBuild 的 Amazon EFS 示例</a> 。	October 26, 2018
<a href="#">Bitbucket Webhook (p. 412)</a>	在您为存储库使用 BitBucket 时，CodeBuild 现在支持 Webhook。有关更多信息，请参阅 <a>CodeBuild 的 Bitbucket 拉取请求示例</a> 。	October 2, 2018

S3 日志 (p. 412)	CodeBuild 现在在 S3 存储桶中支持构建日志。以前，您只能使用 CloudWatch Logs 构建日志。有关更多信息，请参阅 <a href="#">创建项目</a> 。	September 17, 2018
多输入源和多输出构件 (p. 412)	CodeBuild 现在支持使用多个输入源和发布多个构件集的项目。有关更多信息，请参阅 <a href="#">多输入源和输出构件示例</a> 和 <a href="#">CodePipeline 与 CodeBuild 和多输入源和输出构件集成示例</a> 。	August 30, 2018
语义版本控制示例 (p. 412)	《CodeBuild 用户指南》现在包含一个基于使用案例的示例，演示如何使用语义版本控制在构建时创建构件名称。有关更多信息，请参阅 <a href="#">使用语义版本控制指定构建构件示例</a> 。	August 14, 2018
新的静态网站示例 (p. 412)	《CodeBuild 用户指南》现在包含一个基于使用案例的示例，演示如何在 S3 存储桶中托管构建输出。此示例利用了对未加密构建构件的最新支持。有关更多信息，请参阅 <a href="#">创建将构建输出托管在 S3 存储桶中的静态网站</a> 。	August 14, 2018
支持使用语义版本控制覆盖构件名称 (p. 412)	您现在可以使用语义版本控制指定 CodeBuild 用于命名构建构件的格式。这很有用，因为具有硬编码名称的构建构件将覆盖之前使用同一硬编码名称的构建构件。例如，如果每天触发一个构建多次，则现在可以为此构建的构件名称添加时间戳。每个构建构件名称是唯一的，不会覆盖之前构建的构件。	August 7, 2018
支持未加密的构建构件 (p. 412)	CodeBuild 现在支持包含未加密构建构件的构建。有关更多信息，请参阅 <a href="#">创建构建项目（控制台）</a> 。	July 26, 2018
支持 Amazon CloudWatch 指标和警报 (p. 412)	CodeBuild 现在提供与 CloudWatch 指标和警报的集成。可以使用 CodeBuild 或 CloudWatch 控制台监控项目级别和账户级别的构建。有关更多信息，请参阅 <a href="#">监控构建</a> 。	July 19, 2018
支持报告构建的状态 (p. 412)	CodeBuild 现在可以向您的源提供商报告构建的开始和完成状态。有关更多信息，请参阅 <a href="#">在 CodeBuild 中创建构建项目</a> 。	July 10, 2018
添加到 CodeBuild 文档的环境变量 (p. 412)	构建环境中的环境变量页面使用 CODEBUILD_BUILD_ID、CODEBUILD_LOG_PATH 和 CODEBUILD_START_TIME 环境变量进行了更新。	July 9, 2018

支持构建规范文件中的 finally 语句块 (p. 412)	CodeBuild 文档更新了有关构建规范文件中的可选 finally 语句块的详细信息。finally 语句块命令总是在其相应的命令语句块命令之后执行。有关更多信息，请参阅 <a href="#">构建规范语法</a> 。	June 20, 2018
CodeBuild 代理更新通知 (p. 412)	CodeBuild 文档中有关您使用 Amazon SNS 获得新版本 CodeBuild 代理发布通知的方式的详细信息进行了更新。有关更多信息，请参阅 <a href="#">接收有关新 AWS CodeBuild 代理版本的通知</a> 。	June 15, 2018

## 早期更新

下表描述了 2018 年 6 月之前每次发布 AWS CodeBuild 用户指南时进行的重要更改。

更改	Description	Date
支持 Windows 构建	CodeBuild 现在支持 Microsoft Windows Server 平台的构建，包括 Windows 上 .NET Core 2.0 的预打包构建环境。有关更多信息，请参阅 <a href="#">适用于 CodeBuild 的 Microsoft Windows 示例 (p. 27)</a> 。)	2018 年 5 月 25 日
支持构建幂等性	当您使用 AWS Command Line Interface (AWS CLI) 运行 start-build 命令时，可以指定构建为幂等性的。有关更多信息，请参阅 <a href="#">运行构建 (AWS CLI) (p. 256)</a> 。)	2018 年 5 月 15 日
支持覆盖多个构建项目设置	现在，在创建构建时，可以覆盖多个构建项目设置。覆盖仅针对该构建。有关更多信息，请参阅 <a href="#">在 AWS CodeBuild 中运行构建 (p. 252)</a> 。)	2018 年 5 月 15 日
VPC 终端节点支持	现在，您可以使用 VPC 终端节点来提高构建的安全性。有关更多信息，请参阅 <a href="#">使用 VPC 终端节点 (p. 175)</a> 。)	2018 年 3 月 18 日
支持触发器	现在，您可以创建触发器，按固定频率安排构建。有关更多信息，请参阅 <a href="#">创建 AWS CodeBuild 触发器 (p. 218)</a> 。)	2018 年 3 月 28 日
FIPS 终端节点文档	现在，您可以了解如何使用 AWS Command Line Interface (AWS CLI) 或 AWS 开发工具包来告知 CodeBuild 使用四种联邦信息处理标准 (FIPS) 终端节点之一。有	2018 年 3 月 28 日

更改	Description	Date
	有关更多信息，请参阅 <a href="#">指定 AWS CodeBuild 终端节点 (p. 365)</a> 。)	
AWS CodeBuild 在亚太地区（孟买）、欧洲（巴黎）和南美洲（圣保罗）中可用	AWS CodeBuild 现已在亚太地区（孟买）、欧洲（巴黎）和南美洲（圣保罗）区域中可用。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 <a href="#">AWS CodeBuild</a> 。	2018 年 3 月 28 日
GitHub Enterprise Server 支持	CodeBuild 现在可以从存储在 GitHub Enterprise Server 存储库中的源代码进行构建。有关更多信息，请参阅 <a href="#">GitHub Enterprise Server 示例 (p. 109)</a> 。)	2018 年 1 月 25 日
Git clone 深度支持	CodeBuild 现已支持创建一个浅克隆，其历史记录会截断至指定数量的提交。有关更多信息，请参阅 <a href="#">创建构建项目 () (p. 189)</a> 。)	2018 年 1 月 25 日
VPC 支持	支持 VPC 的构建现在能够访问 VPC 内的资源。有关更多信息，请参阅 <a href="#">VPC 支持 (p. 173)</a> 。)	2017 年 11 月 27 日
依赖项缓存支持	CodeBuild 现在支持依赖项缓存。这允许 CodeBuild 将构建环境的某些可重用部分保存在缓存中并在各个构建中使用它。	2017 年 11 月 27 日
构建徽章支持	CodeBuild 现在支持使用构建徽章，该徽章提供一个动态生成的可嵌入映像（徽章），用以显示项目的最新构建状态。有关更多信息，请参阅 <a href="#">构建徽章示例 (p. 78)</a> 。)	2017 年 11 月 27 日
AWS Config 集成	AWS Config 现在支持 CodeBuild 作为 AWS 资源，这表示该服务可以跟踪您的 CodeBuild 项目。有关 AWS Config 的更多信息，请参阅 <a href="#">AWS Config 示例 (p. 66)</a> 。	2017 年 10 月 20 日
在 GitHub 存储库中自动重新构建更新的源代码	如果您的源代码存储在 GitHub 存储库中，则可让 AWS CodeBuild 在代码更改被推送到存储库时重新构建源代码。有关更多信息，请参阅 <a href="#">GitHub 拉取请求和 Webhook 筛选条件示例 (p. 115)</a> 。)	2017 年 9 月 21 日

更改	Description	Date
在 Amazon EC2 Systems Manager Parameter Store 中存储和检索敏感或大型环境变量的新方法	<p>您现在可以使用 AWS CodeBuild 控制台或 AWS CLI 来检索存储在 Amazon EC2 Systems Manager Parameter Store 中的敏感或大型环境变量。现在也可以使用 AWS CodeBuild 控制台将这些类型的环境变量存储在 Amazon EC2 Systems Manager Parameter Store 中。以前，您只能通过将这些类型的环境变量包含在构建规范中或运行构建命令以自动化 AWS CLI 来检索这些变量。您只能通过使用 Amazon EC2 Systems Manager Parameter Store 控制台来存储这些类型的环境变量。有关更多信息，请参阅<a href="#">创建构建项目 () (p. 189)</a>、<a href="#">更改构建项目的设置 () (p. 236)</a>和<a href="#">运行构建 () (p. 252)</a>。</p>	2017 年 9 月 14 日
构建删除支持	<p>您现在可以在 AWS CodeBuild 中删除构建。有关更多信息，请参阅<a href="#">删除构建 (p. 274)</a>。)</p>	2017 年 8 月 31 日
更新了使用构建规范检索存储在 Amazon EC2 Systems Manager Parameter Store 中的敏感或大型环境变量的方法	<p>AWS CodeBuild 现在可让您更容易地使用构建规范来检索存储在 Amazon EC2 Systems Manager Parameter Store 中的敏感或大型环境变量。以前，您只能通过运行构建命令自动化 AWS CLI 来检索这些类型的环境变量。有关更多信息，请参阅<a href="#">构建规范语法 (p. 137)</a>中的 parameter-store 映射。</p>	2017 年 8 月 10 日
AWS CodeBuild 支持 Bitbucket	<p>CodeBuild 现在可以从存储在 Bitbucket 存储库中的源代码进行构建。有关更多信息，请参阅<a href="#">创建构建项目 () (p. 189)</a>和<a href="#">运行构建 () (p. 252)</a>。</p>	2017 年 8 月 10 日
AWS CodeBuild 在美国西部（加利福尼亚北部）、欧洲（伦敦）和加拿大（中部）中可用	<p>AWS CodeBuild 现已在<a href="#">美国西部（加利福尼亚北部）</a>、<a href="#">欧洲（伦敦）</a>和<a href="#">加拿大（中部）</a>区域中可用。有关更多信息，请参阅 Amazon Web Services 一般参考中的<a href="#">AWS CodeBuild</a>。</p>	2017 年 6 月 29 日
已支持其他构建规范文件名称和位置	<p>现在，您可以不为构建项目指定源代码根目录处的名为 <code>buildspec.yml</code> 的默认构建规范文件，而是指定使用文件名称或位置与之不同的构建规范文件。有关更多信息，请参阅<a href="#">构建规范文件名称和存储位置 (p. 136)</a>。)</p>	2017 年 6 月 27 日

更改	Description	Date
更新了构建通知示例	CodeBuild 现在通过 Amazon CloudWatch Events 和 Amazon Simple Notification Service (Amazon SNS) 提供对构建通知的内置支持。先前的 <a href="#">构建通知示例 (p. 80)</a> 已更新为演示此新行为。	2017 年 6 月 22 日
添加了自定义映像中的 Docker 示例	已添加说明如何使用 CodeBuild 和自定义 Docker 构建映像来构建和运行 Docker 映像的示例。有关更多信息，请参阅 <a href="#">自定义映像示例中的 Docker (p. 102)</a> 。	2017 年 6 月 7 日
从 GitHub 提取源代码的拉取请求	当使用依赖于存储在 GitHub 存储库中的源代码的 CodeBuild 运行构建时，您现在可以指定要构建的 GitHub 拉取请求 ID。您还可以改为指定提交 ID、分支名称或标签名称。有关更多信息，请参阅 <a href="#">运行构建 (控制台) (p. 252)</a> 中的 Source version 值或 <a href="#">运行构建 (AWS CLI) (p. 256)</a> 中的 sourceVersion 值。	2017 年 6 月 6 日
更新了构建规范版本	发布了新版本的构建规范格式。版本 0.2 可解决 CodeBuild 在默认 Shell 的单独实例中运行每条构建命令的问题。在版本 0.2 中，environment_variables 已重命名为 env、和 plaintext 已重命名为 variables。有关详细信息，请参阅 <a href="#">适用于 CodeBuild 的构建规范参考 (p. 136)</a> 。	2017 年 5 月 9 日
在 GitHub 中提供构建映像的 Dockerfile	AWS CodeBuild 提供的许多构建映像的定义在 GitHub 中作为 Dockerfile 提供。有关更多信息，请参阅 <a href="#">CodeBuild 提供的 Docker 映像 (p. 159)</a> 中表的“Definition (定义)”列。	2017 年 5 月 2 日
AWS CodeBuild 在欧洲 ( 法兰克福 ) 、亚太区域 ( 新加坡 ) 、亚太区域 ( 悉尼 ) 和亚太区域 ( 东京 ) 中可用	AWS CodeBuild 现已在欧洲 ( 法兰克福 ) 、亚太区域 ( 新加坡 ) 、亚太区域 ( 悉尼 ) 和亚太区域 ( 东京 ) 区域中可用。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 <a href="#">AWS CodeBuild</a> 。	2017 年 3 月 21 日
CodePipeline 对 CodeBuild 的测试操作支持	现在，您可以将使用 CodeBuild 的测试操作添加到 CodePipeline 中的管道。有关更多信息，请参阅 <a href="#">将 CodeBuild 测试操作添加到管道 (CodePipeline 控制台) (p. 378)</a> 。	2017 年 3 月 8 日

更改	Description	Date
构建规范文件支持从选定的顶级目录中获取构建输出	现在，您可以借助构建规范文件来指定单独的顶级目录，并指示 CodeBuild 将这些目录中的内容添加到构建输出构件中。为此，您可以使用 <code>base-directory</code> 映射。有关更多信息，请参阅 <a href="#">构建规范语法 (p. 137)</a> 。)	2017 年 2 月 8 日
内置环境变量	AWS CodeBuild 为您提供额外的内置环境变量。这些包括描述启动了构建项目的实体的环境变量、指向源代码存储库的 URL 以及源代码的版本 ID 等。有关更多信息，请参阅 <a href="#">构建环境中的环境变量 (p. 168)</a> 。)	2017 年 1 月 30 日
AWS CodeBuild 在美国东部（俄亥俄州）中可用	AWS CodeBuild 目前在美国东部（俄亥俄州）区域中可用。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 <a href="#">AWS CodeBuild</a> 。	2017 年 1 月 19 日
Shell 和命令行为信息	CodeBuild 在构建环境默认 Shell 的单独实例中运行您指定的每个命令。这种默认行为可对您的命令产生一些意想不到的副作用。我们推荐了一些方法，用于在需要时处理这种默认行为。有关更多信息，请参阅 <a href="#">构建环境中的 Shell 和命令 (p. 167)</a> 。)	2016 年 12 月 9 日
环境变量信息	CodeBuild 提供了您可以在构建命令中使用的多个环境变量。您也可以定义自己的环境变量。有关更多信息，请参阅 <a href="#">构建环境中的环境变量 (p. 168)</a> 。)	2016 年 12 月 7 日
故障排除主题	现已提供故障排除信息。有关更多信息，请参阅 <a href="#">AWS CodeBuild 问题排查 (p. 387)</a> 。)	2016 年 12 月 5 日
Jenkins 插件初始版本	这是 CodeBuild Jenkins 插件的初始版本。有关更多信息，请参阅 <a href="#">将 AWS CodeBuild 与 Jenkins 结合使用 (p. 381)</a> 。)	2016 年 12 月 5 日
用户指南初始版本	这是 CodeBuild 用户指南的初始版本。	2016 年 12 月 1 日

# AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 [AWS 词汇表](#)。