用户指南

Amazon CodeCatalyst

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

Amazon CodeCatalyst: 用户指南

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

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

Table of Contents

亚马逊 CodeCatalyst 不再向新买家开放。现有客户可以继续正常使用该服务。有关更多信息，请参阅如何从中迁移 CodeCatalyst。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

Amazon CodeCatalyst 是什么？

Amazon CodeCatalyst 是一项集成服务，面向在软件开发过程中采用持续集成和部署实践的软件开发团队。CodeCatalyst 将您所需的所有工具集中在一处。您可以利用持续集成/持续交付（CI/CD）工具规划工作、协作编写代码以及构建、测试和部署应用程序。您还可以通过将 AWS 账户连接到 CodeCatalyst 空间，将 AWS 资源与您的项目整合在一起。通过在一个工具中管理应用程序生命周期的所有阶段和方面，您可以快速、自信地交付软件。

在 CodeCatalyst 中，您可以创建一个代表公司、部门或小组的空间，然后创建包含支持开发团队和任务所需资源的项目。CodeCatalyst 资源是在项目内构建的，而项目又是在空间内运行的。为了帮助团队快速入门，CodeCatalyst 提供了基于语言或工具的项目蓝图。根据项目蓝图创建项目时，项目会附带一些资源，如包含示例代码的源存储库、构建脚本、部署操作、虚拟服务器或无服务器资源等。

我可以使用 CodeCatalyst 做什么？

您和您的开发团队可以使用 CodeCatalyst 完成软件开发的每个环节，从规划工作到部署应用程序。您可以使用 CodeCatalyst 来：

- 迭代并协作处理代码 – 使用源代码存储库中的分支、合并、拉取请求和注释与团队协作处理代码。创建开发环境，快速处理代码，无需克隆或设置与存储库的连接。
- 使用工作流构建、测试和部署应用程序 – 使用构建、测试和部署操作配置工作流，以处理应用程序的持续集成和交付。您既可以手动启动工作流，也可以根据代码推送、创建或关闭拉取请求等事件配置工作流自动启动。
- 通过事务跟踪确定团队工作的优先级 – 使用事务创建待办事项，并通过板块监控进行中任务的状态。创建和维护一个健康的待办事项供您的团队处理是软件开发的重要组成部分。
- 设置监控和通知 - 监控团队活动和资源状态，并配置通知以随时了解重要更改。

如何开始使用 CodeCatalyst？

如果您没有空间或想了解如何设置和管理空间，我们建议您开始学习 Amazon CodeCatalyst Administrator Guide。

如果您是项目或空间工作的新手，我们建议您从以下方面入手：

- 回顾 CodeCatalyst 概念
- 创建空间

- 按照教程：使用现代三层 Web 应用程序蓝图创建项目中的步骤创建您的第一个项目

了解有关 CodeCatalyst 的更多信息

您可以在本用户指南和以下资源中了解有关 CodeCatalyst 功能的更多信息：

- 有关 Amazon CodeCatalyst 的 AWS DevOps 博客文章
- Amazon CodeCatalyst API Reference Guide
- Amazon CodeCatalyst Action Development Kit Developer Guide
- CodeCatalyst 常见问题
- Testimonials

如何从中迁移 CodeCatalyst

经过深思熟虑，我们决定从 2025 年 11 月 7 日起关闭新买家访问亚马逊 CodeCatalyst的权限。现有的 Amazon CodeCatalyst 客户可以在现有空间中继续使用该服务，但无法创建新的空间。 AWS 继续投资于 Amazon 的安全性和可用性 CodeCatalyst，但我们不打算推出新功能。

客户可以手动将其数据从 Amazon 迁移 CodeCatalyst 到其他提供商。本文档介绍从 AWS 和管理控制台迁移、提取或删除数据的基本方法。 CodeCatalyst 通过 CodeCatalyst 控制台在其他 AWS 或第三方服务中创建的资源和数据需要通过这些服务删除，以停止累积费用（如果适用）。

买家可以考虑使用宣布于 2025 年 4 月 17 日正式上市的 Amazon Q 迁移到 GitLab Duo。这项新产品是一款集成产品，将 DevSecOps 平台与Amazon Q GitLab的生成人工智能功能融为一体。带有 Amazon Q 的 Gitlab Duo 将 Amazon Q 代理功能直接 GitLab嵌入到 DevSecOps 的平台中，以加快整个软件开发生命周期中复杂的多步骤任务。

迁移存储库

将您的 CodeCatalyst 存储库迁移到 GitLab 存储库

将必备的 URL 与 HTTPS Git 存储库凭据结合使用，按照文档中 GitLab关于通过 URL 从仓库导入源代码的指导进行操作。

将您的 CodeCatalyst 存储库迁移到 GitHub 存储库

将必备的 URL 与 HTTPS Git 存储库凭据结合使用，按照文档中有关导入源代码 GitHub的指导进行操作。

向其他存储库提供商的通用迁移

1. 克隆 CodeCatalyst 存储库

   使用 Git 将 Amazon CodeCatalyst 存储库克隆到您的本地计算机。如果您使用的是 HTTPS，则可以通过运行以下命令来执行此操作：

   ```
   git clone --mirror https://your-aws-repository-url your-aws-repository
   ```

   `your-aws-repository-url`替换为您的 Amazon CodeCatalyst 存储库的 URL。

   将 `your-aws-repository` 替换为该存储库的名称。

示例：

```
git clone https://git-codecatalyst.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
```

2. 设置新的远程存储库指针

   导航到您克隆的 Amazon CodeCatalyst 存储库的目录。然后，将来自新存储库提供商的存储库 URL 添加为远程存储库：

   ```
   git remote add <provider name> <provider-repository-url>
   ```

   将 <provider name> 替换为您选择的提供商名称。（示例：gitlab）

   将 < provider-repository-url > 替换为新的存储库提供商存储库的 URL。

3. 将本地存储库推送到新的远程存储库：

   这会将所有分支和标签推送到新存储库提供商的存储库。提供商名称必须与步骤 2 中的提供商名称匹配。

   ```
   git push <provider name> --mirror
   ```

   备注：

   - 远程存储库应该为空
   - 远程存储库可能有不允许强制推送的受保护分支，具体取决于提供商。如果出现这种情况，您必须导航到新存储库提供商并禁用分支保护以允许强制推送。

4. 验证迁移

   推送完成后，请验证所有文件、分支和标签是否已成功迁移到新存储库提供商。为此，您可以在线浏览存储库，也可以将其克隆到其他位置并在本地查看。

5. 更新遥控器 URLs（可选）

   如果您计划继续在本地使用迁移的存储库，则可能需要更新远程 URL，使其指向新提供商的存储库，而不是 Amazon CodeCatalyst。您可以使用下面的命令进行这项操作：

   ```
   git remote set-url origin <provider-repository-url>
   ```

将 < provider-repository-url > 替换为新的存储库提供商存储库的 URL。

从中提取数据 CodeCatalyst

下载构件

您可以下载和检查由您的 Amazon CodeCatalyst 工作流程操作生成的项目。您可以下载两种类型的构件：

- 源构件 – 包含运行开始时存在的源存储库内容的快照的构件。
- 工作流构件 – 在工作流配置文件的 Outputs 属性中定义的构件。

下载工作流输出的构件：

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 在工作流的名称下，选择运行。
6. 在运行历史记录的运行 ID 列中，选择一个运行。例如，Run-95a4d。
7. 在运行的名称下，选择构件。
8. 在构件旁边，选择下载。此时会下载存档文件。其文件名由七个随机字符组成。
9. 使用您选择的档案提取实用程序来提取存档。

下载事务附件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择要管理其附件的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
3. 要下载附件，请选择要下载的附件旁边的省略号菜单，然后选择下载。

下载操作的源代码

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 找到要查看其代码的操作：

   a. 在导航窗格中，选择 CI/CD，然后选择工作流。

   b. 选择任意工作流的名称，或创建一个工作流。有关创建工作流的信息，请参阅创建工作流。

   c. 选择编辑。

   d. 在左上角，选择 + 操作打开操作目录。

   e. 在下拉列表中，选择 CodeCatalyst要查看的 Amazon CodeCatalyst、 CodeCatalyst Labs 和第三方操作。

   f. 搜索操作，然后选择其名称。请不要选择加号（ + ）。

   有关该操作的详细信息将显示。
4. 在靠近底部的操作详细信息对话框中，选择下载。

   出现的页面中显示操作源代码所在的 Amazon S3 存储桶。有关 Amazon S3 的信息，请参阅《Amazon Simple Storage Service 用户指南》中的什么是 Amazon S3？

从中删除您的数据 CodeCatalyst

在从中删除数据之前 CodeCatalyst，请告知您的团队有关服务迁移的信息，并确认不需要任何资源。数据和资源一旦删除便无法恢复。

请求服务团队代表您删除数据

空间管理员可以通过 CodeCatalyst 控制台中的 Support Center 联系我们，要求服务团队代表他们删除空间。空间管理员必须在 CodeCatalyst 控制台中通过身份验证才能请求删除空间。提交请求后，服务团队将联系您以确认请求，然后再代表您执行操作。

删除您的 CodeCatalyst 空间

您可以删除空间以取消对该空间所有资源的访问权限。您必须拥有空间管理员角色才能删除空间。

注意：删除空间的操作无法撤销，且一旦空间被删除，其中的数据就无法恢复。

删除空间后，所有空间成员都将无法访问空间资源。空间资源的计费也将停止，第三方源存储库提示的任何工作流都将停止。

如果您属于多个空间，则可以在顶部导航栏中选择空间。

删除空间

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择删除。
4. 键入 **`delete`** 以确认删除。
5. 选择删除。

> **Note**
>
> 如果您属于多个空间，您将被重定向到空间概述页面。如果您属于一个空间，您将被重定向到空间创建页面。

如果您删除了一个空间但属于多个空间，您将被重定向到空间概述页面。如果您属于一个空间，您将被重定向到空间创建页面。

如果您通过 CodeCatalyst 控制台在其他服务 AWS 或第三方服务中创建了资源，则需要单独访问这些服务以关闭创建这些服务的结算账号中的资源。删除空间只会删除 CodeCatalyst 数据和资源。

删除项目

您可以删除项目以移除对该项目的资源的所有访问权限。您必须拥有空间管理员或项目管理员角色才能删除项目。在删除项目后，项目成员将无法访问项目资源，并且第三方源存储库所触发的任何工作流都将停止。

删除项目：

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 选择删除项目。
5. 输入 **delete** 以确认删除。
6. 选择删除项目。

如果您通过 CodeCatalyst 项目在其他服务 AWS 或第三方服务中创建了资源，则需要单独访问这些服务以关闭创建这些服务的结算账号中的资源。删除空间只会删除 CodeCatalyst 数据和资源。

删除源存储库

您可以删除 Amazon CodeCatalyst 项目的源存储库。删除源存储库也会删除存储在存储库中的所有项目信息。如果有任何工作流依赖于源存储库，则在删除该存储库后，这些工作流将从项目工作流列表中删除。引用源存储库的事务不会被删除或更改，但一旦删除源存储库，添加到事务中的任何指向源存储库的链接都将失效。

重要提示：删除源存储库的操作无法撤销。删除源存储库后，就无法再克隆它、从中提取数据或向其推送数据。删除源存储库不会删除该存储库的任何本地副本（本地存储库）。要删除本地存储库，请使用本地计算机的目录和文件管理工具。

注意：您无法在 CodeCatalyst 控制台中删除链接的存储库。要删除链接的存储库，请在存储库列表中选择链接，在托管该存储库的服务中打开该存储库，然后删除它。有关更多信息，请参阅托管链接存储库的服务的文档。

要从项目中移除链接存储库，请参阅中的取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目。 CodeCatalyst

删除源存储库

1. 导航到包含要删除的源存储库的项目。
2. 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择存储库的名称。
3. 在存储库主页上，选择更多，选择管理设置，然后选择删除存储库。
4. 查看分支、拉取请求和相关工作流信息，以确保不会删除仍在使用或有未完成工作的存储库。如果要继续，请键入 delete，然后选择删除。

删除自定义蓝图

当您从 Amazon CodeCatalyst 空间中删除蓝图时，您对该蓝图项目或蓝图版本的资源的所有访问权限都将被移除。在您删除蓝图时，项目成员将无法访问项目资源，并且第三方源存储库所触发的任何工作流都将停止。

如果删除蓝图，不会影响已应用该蓝图的项目。不会从项目中移除蓝图的资源。

重要提示：要从空间中删除已发布的自定义蓝图或自定义蓝图的目录版本，您必须使用在空间中具有空间管理员或高级用户角色的账户进行登录。

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要删除自定义蓝图的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 在设置表中，选择要删除的自定义蓝图的单选按钮，然后选择删除蓝图。
5. 输入 `delete` 以确认删除蓝图目录版本。
6. 选择删除。

删除事务附件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择要管理其附件的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
3. 要移除附件，请选择要移除的附件旁边的省略号菜单，然后选择删除。

删除通过 Amazon 访问的开发环境中的文件 CodeCatalyst

您可以在开发环境、本地计算机或集成式开发环境（IDE）中删除文件。您无法在 Amazon CodeCatalyst 控制台中删除文件。

删除空间的开发环境

有关删除开发环境的考虑因素的更多信息，请参阅删除开发环境。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理开发环境。如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

删除开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间
3. 选择设置，然后选择开发环境。
4. 选择要管理的开发环境旁边的选择器。选择删除。
5. 输入 **delete** 以确认删除开发环境。
6. 选择删除。

删除账户连接

您可以在 CodeCatalyst 控制台中删除之前添加到空间中的账户连接。删除一个账户连接后，您就无法重新建立该连接，必须建立新的连接。

必须为您的 CodeCatalyst 空间指定一个账单账户，即使空间的使用量不会超过免费套餐。您需要先为您的空间添加另一个账户，之后才能为用作指定计费账户的账户移除空间。如果您想删除该空间的计费账户，则需要删除您的空间。请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

要管理空间的账户连接，您必须具有空间管理员或高级用户角色。

稍后可以重新添加已移除的账户，但您必须在账户和空间之间创建新的连接。您需要将所有 IAM 角色重新关联到该账户。

删除账户连接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 在 Amazon CodeCatalyst 显示名称下，选择要删除的账户关联旁边的选择器。
4. 选择移除 AWS 账户。在字段中输入名称来确认删除，然后选择移除。

   这将显示成功横幅，表明已从连接列表中移除账户连接。

从 AWS 管理控制台的 CodeCatalyst 空间中移除帐户

您可以使用 CodeCatalyst 中的页面 AWS 来移除已添加到空间的帐户。在此过程中，使用您管理的特定账户的管理权限，登录 AWS 管理控制台中的 Amazon CodeCatalyst Spaces 页面，将 AWS 账户从

您的空间中移除。要移除作为您 CodeCatalyst 房源指定结算账户的账户，请务必先指定一个新的结算账号。

稍后可以重新添加已移除的账户，但您必须在账户和空间之间创建新的连接。您将需要将所有 IAM 角色重新关联到已添加的账户。

必须为您的 CodeCatalyst 空间指定一个账单账户，即使空间的使用量不会超过免费套餐。您需要先为您的空间添加另一个账户，之后才能为用作指定计费账户的账户移除空间。

您必须具有空间管理员或高级用户角色才能管理空间的账户连接。

删除已添加的账户

1. 在中 AWS 管理控制台，请确保您使用要管理的相同帐户登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择账单。
4. 在页面上查看计费账户信息，确保要删除的账户不是该空间的指定计费账户。
5. 在中选择 “管理账单” AWS。这将在中打开 Amazon CodeCatalyst 空间 AWS 管理控制台。如果系统提示您登录，请登录 AWS，然后再次选择该按钮加载页面。
6. 在 Amazon CodeCatalyst Spaces 页面上，选择包含您要删除的账户的空间。此时将显示空间的详细信息页面。
7. 选择删除空间。
8. 在 “删除此帐户的 CodeCatalyst 空间” 中，输入空间名称进行确认。选择移除 。

删除密钥

使用以下步骤来删除密钥和密钥引用标识符。在删除密钥之前，我们建议您从所有工作流操作中移除该密钥的引用标识符。如果您删除密钥而不删除引用标识符，则该操作在下次运行时将失败。

从工作流中删除密钥的引用标识符

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。

5. 选择 YAML。
6. 在工作流中搜索以下字符串：

```
${Secrets.
```

这将找到所有密钥的所有引用标识符。

7. 删除所选密钥的参考标识符，或将其替换为纯文本值。
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

删除密钥

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择密钥。
3. 在密钥列表中，选择要删除的密钥。
4. 选择删除。
5. 输入 **delete** 以确认删除。
6. 选择删除。

删除团队

您可以删除不再需要的团队。在删除团队时，将从空间中的所有项目和资源中移除所有团队成员的关联权限。您必须拥有空间管理员角色才能管理团队。

删除团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 在操作中，选择删除团队。这将更改整个团队的角色。
4. 选择删除。

删除预置的实例集

按照以下说明操作来删除预置的实例集。

在删除某个预调配实例集之前，请从操作的 YAML 代码中删除 Fleet 属性，以从所有操作中删除该实例集。任何在删除预置实例集后继续引用该实例集的操作都将在下次运行时失败。

删除预置的实例集

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 在预置实例集列表中，选择要删除的实例集。
3. 选择删除。
4. 输入 **`delete`** 以确认删除。
5. 选择删除。

删除程序包存储库

要在中删除软件包存储库，请执行以下步骤 CodeCatalyst。

删除程序包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含要删除的程序包存储库的项目。
3. 在导航窗格中，选择程序包。
4. 在程序包存储库页面上，选择要删除的存储库。
5. 选择删除。
6. 查看提供的有关删除程序包存储库的效果的信息。
7. 在输入框中输入 `delete`，然后选择删除。

您的空间被删除后，存储在 Amazon CodeCatalyst 中的所有其他资源都将被删除。这不包括通过 CodeCatalyst 控制台在其他服务 AWS 或 3P 服务中创建的资源和数据。在 CodeCatalyst 控制台之外的服务中创建的所有资源都需要通过这些服务删除，才能停止累积费用。

如果您还有其他问题，请通过 aws-codecatalyst-service @amazon .com 联系我们，或通过亚马逊 CodeCatalyst 控制台中的支持中心联系我们。

CodeCatalyst 概念

熟悉有助于加快在 Amazon 中的协作和应用程序开发的关键概念 CodeCatalyst。这些概念涉及的术语用于源代码控制、持续集成和持续交付（CI/CD）以及建模和配置自动发布流程。

有关其他概念性信息，请参阅以下主题：

- 源存储库概念
- 工作流概念

主题

- AWS 中的建筑商 ID 空格 CodeCatalyst
- 支持身份联合的空间 CodeCatalyst
- Projects
- 蓝图
- 账户连接
- VPC 连接
- AWS 生成器 ID
- 中的用户个人资料 CodeCatalyst
- 源存储库
- 提交
- 开发环境
- 工作流
- 操作
- 事务
- 个人访问令牌 (PATs)
- 个人连接
- 角色

AWS 中的建筑商 ID 空格 CodeCatalyst

空间管理员 CodeCatalyst 通过从成员页面发送个人邀请电子邮件来邀请用户加入。受邀或注册 CodeCatalyst 创建自己的 AWS 建造者 ID 的用户。配置文件在 AWS Builder ID 中进行管理，并在中的用户设置中显示为用户名和配置文件信息 CodeCatalyst。

支持身份联合的空间 CodeCatalyst

那些已添加到 IAM Identity Center 实例的 SSO 用户和组、在身份存储中进行管理并通过 IAM Identity Center 邀请加入空间的用户。空间管理员同步 CodeCatalyst 成员页面以获取最新更新。用户使用公司 IAM Identity Center 实例中设置的 SSO 登录门户进行登录。支持身份联合验证的空间通过 Identity Center 应用程序及其与身份存储 ID 的映射，连接到身份存储实例。

Projects

项目代表一种为开发团队和任务提供支持的协作努力。 CodeCatalyst 在拥有项目后，可以添加、更新或移除用户和资源，自定义您的项目控制面板以及监控团队的工作进度。一个空间内可以有多个项目。

有关项目的更多信息，请参阅使用项目来组织工作 CodeCatalyst。

蓝图

蓝图是一种项目合成器，它可以为您生成和扩展应用程序支持文件和依赖项，并在控制台中创建 CodeCatalyst 项目。您可以从中的精选蓝图中选择一种项目类型 CodeCatalyst，查看自述文件并预览将生成的项目存储库和资源。您的项目是根据蓝图指定的基本配置生成的。您定期与项目蓝图进行合成，这将更新您的项目文件（例如软件依赖项）并重新生成资源。项目使用工具 Projen，此工具可以同步最新的项目更新并生成支持文件，从而合成项目。这些文件可能包括 `package.json`、`Makefile`、`eslint` 等，具体取决于您的应用程序类型和语言。项目蓝图可以生成支持 CDK 构造、 CloudFormation 模板和模板等 AWS 资源的文件。 AWS Serverless Application Model

有关项目蓝图的更多信息，请参阅使用 CodeCatalyst 蓝图创建综合项目。

账户连接

账户关联将 CodeCatalyst 空间与您的空间相关联 AWS 账户。在您的账户连接设置完成后， AWS 账户 该空间即可使用。然后，您可以向中添加 IAM 角色， CodeCatalyst 使其可以访问您的中的资源 AWS 账户。您也可以将这些角色用于您的 CodeCatalyst 工作流程操作。

您可以通过启用受项目限制的账户连接，来限制哪些项目和资源有权访问账户连接。已连接@@ 受项目限制的账户连接 AWS 账户 ，只能由空间中的指定项目访问。这允许空间中的团队按项目限制 AWS 账户 对整合 AWS 资源的使用。例如，在特定项目中用于部署工作流和 VPC 连接的账户只能通过受项目限制的账户连接使用。有关更多信息，请参阅 Configuring project-restricted account connections。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。

VPC 连接

VPC 连接是一种 CodeCatalyst 资源，其中包含您的工作流程访问 VPC 所需的所有配置。空间管理员可以代表空间成员在 Amazon CodeCatalyst 控制台中添加自己的 VPC 连接。通过添加 VPC 连接，空间成员可以运行工作流操作和创建符合网络规则的开发环境，并能访问已关联 VPC 中的资源。

有关 VPC 连接的更多信息，请参阅《CodeCatalyst 管理员指南》中的管理 Amazon 虚拟私有云。

AWS 生成器 ID

AWS Builder ID 是一种个人身份，可用于注册和登录 CodeCatalyst 以及其他参与的应用程序。它和...不一样 AWS 账户。您的 AWS Builder ID 管理元数据，例如用户别名和电子邮件地址。您的 AWS 建筑商 ID 是一个唯一的身份，可支持所有空间中的用户 CodeCatalyst。有关访问您的 AWS 建筑商 ID 个人资料的信息，请参阅创建个人资料。要了解有关 AWS 生成器 ID 的更多信息，请参阅中的AWS 生成器 ID AWS 一般参考。

有关注册和登录的更多信息，请参阅设置并登录 CodeCatalyst。

中的用户个人资料 CodeCatalyst

您可以通过在任何页面的登录名首字母下方的下拉列表中选择配置文件选项来访问您的 CodeCatalyst 用户个人资料。 CodeCatalyst您可以从个人资料页面创建个人访问令牌 (PATs)，但只能 PATs 使用查看或删除 AWS CLI。您的用户名是您在注册时选择的别名。您不能更改用户名。要查看其他 CodeCatalyst 用户的个人资料页面，请转到项目的 “成员” 选项卡，然后选择相应的用户。

要访问您的 AWS 建筑商 ID，请查看您的 CodeCatalyst 个人资料，然后选择前往 AWS 建筑商 ID。您将被重定向到您的 AWS Builder ID 个人资料页面。您的个人资料的全名、电子邮件地址和密码由您的 AWS 建筑商 ID 管理，您可以使用 Bu AWS ilder ID 页面编辑这些信息。您在注册时输入了这些信息。当您准备好将 MFA 设置为使用身份验证器应用程序登录时，您将使用 AWS Builder ID 页面。有关查看 AWS 建筑商 ID 个人资料的更多信息，请参阅创建个人资料。

有关注册和登录的更多信息，请参阅设置并登录 CodeCatalyst。

源存储库

源存储库用于安全地存储项目的代码和文件。它还存储文件的版本历史记录。默认情况下，源存储库与 CodeCatalyst 项目中的其他用户共享。一个项目可以有多个源存储库。您可以为中的项目创建源存储库 CodeCatalyst，也可以选择链接由其他服务托管的现有源存储库（如果已安装的扩展程序支持该服务）。例如，在安装 GitHub 存储库扩展之后，您可以将GitHub 存储库链接到项目。有关更多信息，请参阅将源代码存储在项目的存储库中 CodeCatalyst和快速入门：安装扩展、连接提供商和链接资源 CodeCatalyst。

源存储库也是存储 CodeCatalyst项目配置信息的地方，例如定义 CI/CD 工作流程属性和操作的配置文件。如果您使用蓝图创建项目，系统将创建一个源存储库，其中存储了项目配置信息。如果您创建一个空项目，则必须先创建一个源存储库，之后才能创建需要配置信息的资源（例如工作流）。

有关可帮助您使用源存储库和源代码控制的更多概念，请参阅源存储库概念。

提交

提交是对一个或一组文件所做的更改。在 Amazon CodeCatalyst 控制台中，提交会保存您的更改并将其推送到源存储库。提交包含有关更改的信息，包括执行了更改的用户的身份、更改的时间和日期、提交标题以及包含的有关更改的任何消息。有关更多信息，请参阅 通过在 Amazon 中提交来了解源代码的变化 CodeCatalyst。

在中的源存储库的上下文中 CodeCatalyst，提交是仓库内容更改的快照。用户每次提交和推送更改时，都会 CodeCatalyst 保存信息，包括谁提交了更改、提交日期和时间以及作为提交一部分所做的更改。您还可以向提交添加 Git 标签来帮助识别特定提交。

有关提交的更多信息，请参阅通过在 Amazon 中提交来了解源代码的变化 CodeCatalyst。

开发环境

开发环境是一种基于云的开发环境，您可以在中使用它 CodeCatalyst 来快速处理存储在项目源存储库中的代码。开发环境中包含的项目工具和应用程序库由项目的源存储库中的 devfile 定义。如果您的源存储库中没有 devfile，系统会自动应用默认的 devfile。默认 devfile 包括适用于最常用的编程语言和框架的工具。默认情况下，为开发环境配备了双核处理器、4 GB RAM 和 16 GiB 持久性存储。

工作流

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流

启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

有关工作流的更多信息，请参阅使用工作流进行构建、测试和部署。

操作

操作是工作流的主要构建基块，它定义了工作流运行期间要执行的逻辑工作单元（又称任务）。通常，一个工作流包括多个按顺序运行或并行运行的操作，具体取决于您配置这些操作的方式。

有关操作的更多信息，请参阅配置工作流操作。

事务

事务是跟踪与项目相关的工作的记录。您可以为功能、任务、错误或任何其他与项目相关的工作主体创建事务。如果您使用的是敏捷开发，一个事务还可以描述一个长篇故事或用户故事。

有关事务的更多信息，请参阅跟踪和组织处理问题的工作 CodeCatalyst。

个人访问令牌 (PATs)

个人访问令牌（PAT）类似于密码。它与您的用户身份相关联，可在中的所有空间和项目中使用 CodeCatalyst。您可以使用 PATs 访问包括集成开发环境 (IDEs) 和基于 Git 的源存储库在内的 CodeCatalyst 资源。 PATs 代表你 CodeCatalyst ，你可以在用户设置中对其进行管理。一个用户可以有多个 PAT。个人访问令牌仅显示一次。最佳实践是，务必将 PAT 安全地存储在本地计算机上。默认情况下，一年后 PATs 过期。

有关的更多信息 PATs，请参阅使用个人访问令牌向用户授予对存储库的访问权限。

个人连接

个人连接是您的 CodeCatalyst 身份与外部来源提供商之间的授权，例如 GitHub。您可以使用个人关系来允许 CodeCatalyst用户添加第三方来源存储库。例如，您可以将 GitHub 存储库连接到

CodeCatalyst 空间。已安装的连接器应用程序安装在 GitHub 账户中，用于账户所有者指定的存储库。您可以为特定提供商类型的所有空间中的一个用户身份（CodeCatalyst 别名）创建一个个人连接，例如 GitHub。个人关系要么与您的 AWS 生成器 ID 关联，要么与您的 SSO 用户相关联。

有关更多信息，请参阅 通过人际关系访问 GitHub 资源。

角色

角色定义了用户对项目或空间资源的访问权限以及用户可以执行的操作。当您邀请用户加入项目时，您可以为用户选择角色。中有空间级角色和项目级角色。 CodeCatalyst具有正确级别的管理角色的用户可以更改分配的角色。例如，具有项目管理员角色的用户可以完全控制该项目，并且可以更改该项目中用户的角色。有关哪些角色可用以及每个角色拥有哪些权限的信息，请参阅使用用户角色授予访问权限。

有关角色的更多信息，请参阅 使用用户角色授予访问权限。

设置并登录 CodeCatalyst

您可以在 CodeCatalyst 中设置两种类型的空间：支持 AWS 构建者 ID 用户的空间，以及支持身份联合验证的空间（通过 IAM Identity Center 管理 SSO 用户和组）。AWS 构建者 ID 空间中的用户使用其 AWS 构建者 ID 登录 CodeCatalyst，而在设置了身份联合验证的空间中，用户使用与该空间相关联的公司 SSO 门户登录 CodeCatalyst。

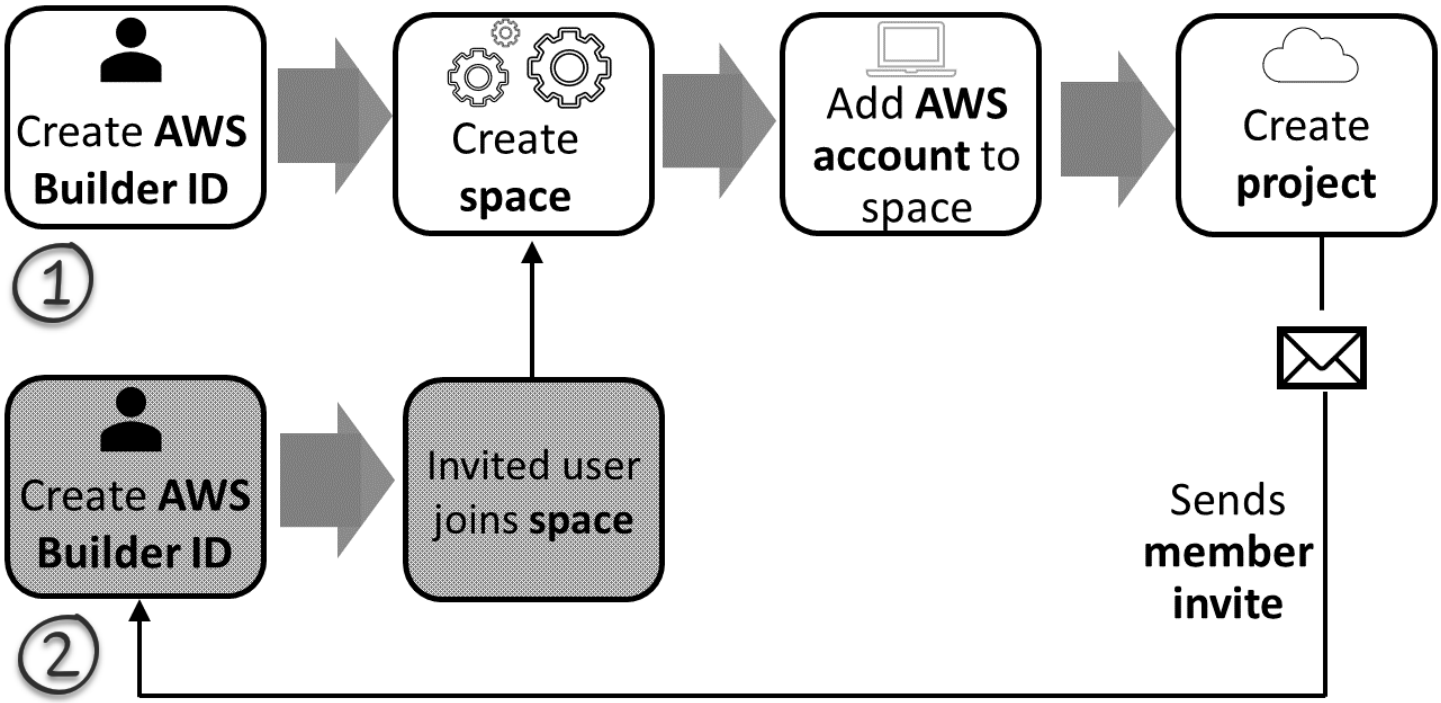
> **Note**
>
> CodeCatalyst 用户名的最小长度为 3 个字符，最大长度为 100 个字符。超过 100 个字符的用户名将被截断。这可能导致一个用户名看起来与另一个 100 个字符的用户名重复。有关更多信息，请参阅 由于用户名被截断，我无法以新用户身份访问我的 BID 空间，或无法将其添加为新的 SSO 用户。

本指南提供了设置和管理 AWS 构建者 ID 空间的步骤。要使用 CodeCatalyst AWS 构建者 ID 空间，您需要使用用户设置和登录 CodeCatalyst 时使用的 AWS 构建者 ID 来设置 CodeCatalyst。

《CodeCatalyst Administrator Guide》中提供了设置和管理支持身份联合验证的空间的步骤。要使用为身份联合验证设置的空间，请参阅《Amazon CodeCatalyst Administrator Guide》中的 Setup and administration for CodeCatalyst spaces。

此部分提供了在具有 AWS 构建者 ID 空间的 Amazon CodeCatalyst 中工作的两种常见设置路径：作为第一个用户创建空间和项目，以及接受加入现有空间或项目的邀请。这些设置工作流截然不同。下图显示了这两个注册流程：

1. 在第一种情况下，您需要为自己的公司、团队或小组创建和设置空间，然后创建一个项目，接下来才能邀请他人使用这些资源。您必须提供一个 AWS 账户用于计费目的，不过仍可在其中默认使用免费套餐。
2. 在第二种情况下，如果您通过接受项目邀请加入 CodeCatalyst，那么其他人已经为您创建了空间和项目。不过，您仍然需要配置自己的个人资料，以便开始与他人合作。



> **Tip**
>
> CodeCatalyst 使用空间对项目和资源进行分组。首次注册 CodeCatalyst 时，系统会提示您创建空间和项目。

无论您是要注册来创建空间和项目，还是在接受邀请时进行注册，您都会创建一个 AWS 构建者 ID，用于登录 CodeCatalyst。要创建 AWS 构建者 ID，您需要提供用于登录 AWS 应用程序的全名、密码和电子邮件地址。此后，您将使用电子邮件和密码登录 CodeCatalyst。您还可以使用此 AWS 构建者 ID 登录使用 AWS 构建者 ID 凭证的其他应用程序。

在 CodeCatalyst 和 AWS 构建者 ID 中，会根据您的登录信息生成个人资料。您的个人资料包含您在 CodeCatalyst 项目中对语言和通知设置的偏好。

> **Tip**
>
> 如果您在注册 Amazon CodeCatalyst 个人资料时遇到任何问题，请按照页面上提供的步骤操作。如果您需要其他帮助，请参阅注册时出现的问题。

主题

创建新的空间和开发角色（无需邀请即可开始）

您可以注册 Amazon CodeCatalyst，而无需接受加入现有空间或项目的邀请。创建 AWS 构建者 ID 后，您将创建一个空间和项目。作为创建空间的一部分，您需要添加一个 AWS 账户，以便计费。

> **Tip**
>
> 如果您在注册 Amazon CodeCatalyst 个人资料时遇到任何问题，请按照页面上提供的步骤操作。如果您需要其他帮助，请参阅注册时出现的问题。

下面介绍了在没有项目或空间邀请的情况下，用户为使用 CodeCatalyst 可以采取的流程。

Mary Major 是一名开发人员，她对 CodeCatalyst 很感兴趣，决定试用一下。她导航到 CodeCatalyst 控制台，选择注册并创建 AWS 构建者 ID。Mary 提供电子邮件地址和密码来创建 AWS 构建者 ID。她将可以使用自己的 AWS 构建者 ID 登录 CodeCatalyst 和其他应用程序。当被要求选择别名时，她指定 `MaryMajor` 作为 CodeCatalyst 用户名，该用户名将显示在 CodeCatalyst 中，其他项目成员将使用该用户名来 @ Mary。

接下来，Mary 会被自动引导创建一个空间。作为该流程的一部分，Mary 被要求将 AWS 账户与她正在创建的空间关联起来，这样她就可以在第一个项目的构建和部署中看到示例代码。她添加了这些信息并创建了自己的空间，在这里她选择创建预览开发角色的选项，该角色可用于新空间中的项目。Mary 选择创建一个项目，然后查看项目蓝图列表。在查看了可用蓝图的信息后，她决定在第一个项目中尝试现代化三层 Web 应用程序蓝图。于是她填写必填字段，然后创建项目。项目准备就绪后，她可以看到一个项目摘要页面，其中包括最近的活动、项目代码链接以及自动构建和部署代码的工作流。她探索了代码和工作流，包括查看已部署的示例 Web 应用程序。她很喜欢所看到的一切，决定邀请一些同事加入该项目，开始探索 CodeCatalyst。

Mary 花了一点时间配置自己的 AWS 构建者 ID，以便使用多重身份验证（MFA）登录 CodeCatalyst。配置 MFA 后，Mary 可以使用她的 CodeCatalyst 密码和经认可的第三方身份验证应用程序的密码或令牌组合登录 CodeCatalyst。

创建新空间和 IAM 角色

按照以下步骤注册 Amazon CodeCatalyst 个人资料、创建空间并为您的空间添加账户、支持角色和开发人员角色。

最后一个过程是创建和添加开发人员角色。开发人员角色是一个 AWS IAM 角色，使您的 CodeCatalyst 工作流能够访问 AWS 资源。开发人员角色是用于管理 AWS 服务的服务角色，将在登录的账户中创建。服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-spaceName`。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-**spaceName** 服务角色。

> **Note**
>
> 作为安全最佳实践，只为需要管理空间内 AWS 资源访问权限的管理用户和开发人员分配管理访问权限。

在开始之前，您必须准备好为您拥有管理权限的账户提供 AWS 账户 ID。准备好您的 12 位 AWS 账户 ID。有关如何查找您的 AWS 账户 ID 的信息，请参阅您的 AWS 账户 ID 及其别名。

以新用户身份注册

1. 在开始使用 CodeCatalyst 控制台之前，请打开 AWS 管理控制台，然后确保使用与要用于创建空间的相同 AWS 账户登录。
2. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
3. 在欢迎页中，选择注册。此时将显示创建您的 AWS Builder ID 页面。您的 AWS 构建者 ID 是您为登录而创建的身份。它与 AWS 账户不同。
4. 在您的电子邮件地址中，输入您要与 CodeCatalyst 关联的电子邮件地址。然后选择下一步。
5. 在您的姓名中，提供您希望在使用 AWS 构建者 ID 的应用程序中显示的名字和姓氏。允许使用空格。这将是您的 AWS 构建者 ID 个人资料名称，如 Mary Major。您以后可以更改此名称。

   选择下一步。此时将显示电子邮件验证页面。

6. 验证码将发送到您指定的电子邮件中。在验证码中输入此验证码，然后选择验证。如果您在 5 分钟后仍未收到验证码，也没能在垃圾邮件或广告邮件文件夹中找到该验证码，请选择重新发送验证码。
7. 在我们核实验证码之后，请在密码和确认密码中输入符合要求的密码。

   选中确认您同意 AWS 客户协议和 AWS 服务条款的复选框，然后选择创建 AWS 构建者 ID。
8. 在创建您的 CodeCatalyst 别名页面上，输入您要在 CodeCatalyst 中用于唯一用户标识符的别名。选择您名字的简写，不含空格，如 MaryMajor。其他 CodeCatalyst 用户会使用此名称在评论和拉取请求中 @ 您。您的 CodeCatalyst 资料将包含您的 AWS Builder ID 中的全名和您的 CodeCatalyst 别名。您不能在以后更改 CodeCatalyst 别名。

   您的全名和别名将显示在 CodeCatalyst 的不同区域。例如，您的个人资料名称会显示在活动源中列出的活动中，但项目成员会使用您的别名来 @ 您。

   选择下一步。页面更新，显示创建您的 CodeCatalyst 空间部分。
9. 在命名您的空间中，输入您的空间名称。您以后不能更改此名称。

   > **Note**
   >
   > 空间名称在 CodeCatalyst 中必须是唯一的。不能重复使用已删除空间的名称。
10. 在 AWS 区域下拉菜单中，选择要存储空间和项目数据的区域。您以后不能更改此区域。
11. 选择下一步。页面更新，显示添加 AWS 账户的页面。该账户将作为空间的计费账户。
12. 在 AWS 账户 ID 中，输入您要连接到空间的账户的十二位 ID。

    在 AWS 账户验证令牌中，复制生成的令牌 ID。系统会自动为您复制令牌，但您可能需要将其存储以便批准 AWS 连接请求。
13. 选择前往 AWS 控制台进行验证。
14. 验证 Amazon CodeCatalyst 空间页面将在 AWS 管理控制台中打开。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

    在 AWS 管理控制台中，确保为创建空间选择相同的 AWS 区域。

    要直接访问该页面，请在 AWS 管理控制台中登录到 Amazon CodeCatalyst 空间，网址为 https://console.aws.amazon.com/codecatalyst/home/。

    AWS 管理控制台 中的验证令牌字段会自动填充 CodeCatalyst 中生成的令牌。

15. （可选）在已授权的付费套餐下，选择授权付费套餐（标准版、企业版），为您的计费账户启用付费套餐。

    > **Note**
    >
    > 这不会将计费套餐升级到付费套餐。但这将配置 AWS 账户，以便您能够随时在 CodeCatalyst 中更改空间的计费套餐。您可以随时启用付费套餐。如果不进行此更改，则空间只能使用免费套餐。

16. 选择验证空间。

    这将显示账户已验证成功消息，表示已将该账户添加到空间。

17. 留在验证 Amazon CodeCatalyst 空间页面上。选择以下链接：要为该空间添加 IAM 角色，请查看空间详细信息。

    在 AWS 管理控制台中打开包含 CodeCatalyst 空间详细信息的连接页面。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

18. 返回 CodeCatalyst 页面，然后选择下一步。

19. 创建空间时会显示一条状态消息。创建空间后，CodeCatalyst 会显示以下消息：您的空间已准备就绪。最后一步是创建项目。您可以执行以下操作之一：

    - 选择暂时跳过。
    - 为您的空间选择创建第一个项目。有关向您说明如何使用蓝图创建项目的教程，请参阅教程：使用现代三层 Web 应用程序蓝图创建项目。

    > **Note**
    >
    > 如果显示权限错误或横幅，请刷新页面并尝试再次查看该页面。

创建和添加 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-*spaceName*

1. 在 CodeCatalyst 控制台中开始操作之前，请先打开 AWS 管理控制台，然后确保您使用空间的同一 AWS 账户进行登录。
2. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
4. 选择要在其中创建角色的 AWS 账户的链接。此时将显示 AWS 账户详细信息页面。

5. 选择从 AWS 管理控制台中管理角色。

   这将在 AWS 管理控制台中打开将 IAM 角色添加到 Amazon CodeCatalyst 空间页面。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

6. 选择在 IAM 中创建 CodeCatalyst 开发管理员角色。此选项将创建一个服务角色，其中包含开发角色的权限策略和信任策略。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-spaceName`。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-**spaceName** 服务角色。

   > **Note**
   >
   > 建议仅将此角色用于开发人员账户，并使用 `AdministratorAccess` AWS 托管式策略，从而向此角色授予完全访问权限以便在该 AWS 账户中创建新的策略和资源。

7. 选择创建开发角色。
8. 在连接页面上的 CodeCatalyst 可用的 IAM 角色下，查看已添加到您账户的 IAM 角色的列表中的 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色。
9. 要返回您的空间，请选择转到 Amazon CodeCatalyst。

创建和添加 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在 CodeCatalyst 控制台中开始操作之前，请先打开 AWS 管理控制台，然后确保您使用空间的同一 AWS 账户进行登录。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 选择要在其中创建角色的 AWS 账户的链接。此时将显示 AWS 账户详细信息页面。
4. 选择从 AWS 管理控制台中管理角色。

   这将在 AWS 管理控制台中打开将 IAM 角色添加到 Amazon CodeCatalyst 空间页面。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

5. 在 CodeCatalyst 空间详细信息下，选择添加 CodeCatalyst 支持角色。此选项将创建一个服务角色，其中包含预览开发角色的权限策略和信任策略。该角色的名称为 AWSRoleForCodeCatalystSupport，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 AWSRoleForCodeCatalystSupport 服务角色。
6. 在添加用于 CodeCatalyst 支持的角色页面上，保留默认选定项，然后选择创建角色。
7. 在 CodeCatalyst 可用的 IAM 角色下，查看已添加到您账户的 IAM 角色的列表中的 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色。

8. 要返回您的空间，请选择转到 Amazon CodeCatalyst。

创建 AWS 构建者 ID、创建第一个空间并添加账户后，就可以创建项目了。有关更多信息，请参阅 创建项目。如果这是您首次使用 CodeCatalyst，建议您从教程：使用现代三层 Web 应用程序蓝图创建项目开始。

接受邀请并创建 AWS 建筑商 ID

在接受项目或空间邀请 CodeCatalyst 的同时，您可以注册 Amazon。作为接受邀请的一部分，系统将提示您创建 AWS 生成器 ID。您将使用您的 AWS 生成器 ID 访问中的资源 CodeCatalyst。

> **Tip**
>
> 如果您需要其他帮助，请参阅注册时出现的问题。

对于刚开始获得项目或空间邀请 CodeCatalyst 的用户来说，这是一个可能的流程。

Saanvi Sarkar 是一名开发者，他已收到以项目管理员身份加入 CodeCatalyst 项目的邀请。Saanvi 接受邀请，这将打开登录页面。 CodeCatalyst她选择注册，并提供电子邮件地址和密码来创建自己的 AWS 构建者 ID。Saanvi 将能够使用她的 AWS Builder ID 登录 CodeCatalyst 和其他应用程序。之后，她可以编辑个人资料，更改登录电子邮件地址或密码。当被要求选择别名时，Saanvi 将指定`SaanviSarkar`为将在中 CodeCatalyst 显示的 CodeCatalyst 别名，其他项目成员也将使用该别名来 @mention Saanvi。注册后，Saanvi 还可以将她的登录凭据用于其他使用 AWS Builder ID 凭证的应用程序。

完成注册后，Saanvi 会自动加入邀请中指定的 CodeCatalyst 项目和空间。该邀请还为她在项目和空间中的角色提供了预先确定的权限。在项目设置中，Saanvi 的别名显示在成员列表中，并带有向她分配的项目角色。为了使用中的源存储库 CodeCatalyst，Saanvi 需要花点时间创建个人访问令牌 (PAT)。在 CodeCatalyst 进行源代码更改或需要身份验证令牌的操作时，PAT 将用于身份验证。

当 Saanvi 处理项目时，她的别名就会在该项目的工作活动日志中列出。Saanvi 发布的事务和评论将显示她的别名，其他项目成员可以在回复中 @ 她。对于 @mention 另一位项目成员，Saanvi 会在他们的个人资料中查找他们的别名。 CodeCatalyst

当她有时间时，Saanvi 会将她的 AWS Builder ID 配置为 CodeCatalyst 使用多重身份验证 (MFA) 登录。配置 MFA 后，Saanvi 可以使用 CodeCatalyst 密码和经 CodeCatalyst 批准的第三方身份验证应用程序中的密码或令牌组合登录。

接受邀请并创建 AWS 建筑商 ID

当你被邀请加入亚马逊的项目或空间时 CodeCatalyst，你会收到一封来自 notify@codecatalyst.aws 的电子邮件，要求你接受邀请。如果您已经拥有 AWS 建筑商 ID 并已登录 CodeCatalyst，则选择 “接受邀请” 将自动在浏览器选项卡中打开项目或空间。如果您未登录主机但拥有 AWS 生成器 ID，则系统会将您带到登录页面。有关更多信息，请参阅 使用 AWS 构建者 ID 登录。

如果您没有 AWS 生成器 ID，则选择 “接受邀请” 将带您进入登录页面，您应该在其中选择创建 AWS 生成器 ID 的选项。

> **Important**
>
> 通过接受邀请并在开发环境中打开存储库，脚本可以在访问您的 CodeCatalyst 凭据的情况下执行。在继续操作之前，请确保您信任消息来源。

接受邀请并创建 AWS 建筑商 ID

1. 在邀请电子邮件中，选择接受邀请。
2. 在登录页面上，选择未注册？ 创建您的 AWS 建筑商 ID。

   > **Tip**
   >
   > 您的 AWS 建筑商 ID 是您为登录而创建的身份。它与 AWS 账户不同。

3. 在 “创建您的 AWS 建造者 ID” 页面的 “电子邮件地址” 中，输入要用作 AWS 建造者 ID 的电子邮件地址。

   在 “您的姓名” 中，提供您希望在使用 AWS 构建器 ID 的应用程序中显示的名字和姓氏。允许使用空格。这将是你的 AWS Builder ID 个人资料名称，例如 Mary Major。您以后可以更改此名称。

   选择下一步。

   验证码将发送到您指定的电子邮件中。在验证码中输入此验证码，然后选择验证。如果您在 5 分钟后仍未收到验证码，也没能在垃圾邮件或广告邮件文件夹中找到该验证码，请选择重新发送验证码。
4. 在核实了验证码之后，请在密码和确认密码中输入符合要求的密码。
5. 选择 “创建 AWS 生成器 ID”。

6. 在创建您的别名页面上，输入您要在中用作唯一用户标识符的别名 CodeCatalyst。选择名字的缩写版本，不带空格，例如MaryMajor。其他 CodeCatalyst 用户会在评论和拉取请求中用它来 @mention 你。您的 CodeCatalyst 个人资料将包含您在 AWS 建造者 ID 中的全名和您的 CodeCatalyst 别名。您无法更改您的 CodeCatalyst 别名。

   您的全名和别名将显示在中的不同区域 CodeCatalyst。例如，您的个人资料名称会显示在活动源中列出的活动中，但项目成员会使用您的别名来 @ 您。

   选择创建别名。您将进入受邀加入的项目或空间。

使用 AWS 构建者 ID 登录

请执行以下步骤以登录您的 Amazon CodeCatalyst 个人资料。

> **Note**
>
> 您是否已注册一台设备来进行多重身份验证（MFA）？ 强烈建议您在 Amazon CodeCatalyst 中配置 MFA 以增强安全性。有关更多信息，请参阅如何注册设备以便在多重身份验证中使用。

使用您的 AWS 构建者 ID 登录

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 输入您的电子邮件地址。（可选）如果您想保存您的电子邮件地址以在将来登录时使用，请选择保存我的电子邮件地址。选择继续。
3. 输入您的密码。选择登录。如果您忘记了密码，请按照我忘记密码了中的步骤操作。

受信任设备

在从登录页面中选择选项这是受信任设备后，Amazon CodeCatalyst 会将以后从该设备进行的所有登录视为已授权。只要您使用受信任设备，Amazon CodeCatalyst 就不会显示用于输入 MFA 代码的选项。一些例外情况包括使用新浏览器登录或您的设备获得未知 IP 地址时。

使用 SSO 登录

请按照以下步骤使用 SSO 登录 Amazon CodeCatalyst。

要改用 AWS 构建者 ID 登录，请参阅使用 AWS 构建者 ID 登录。

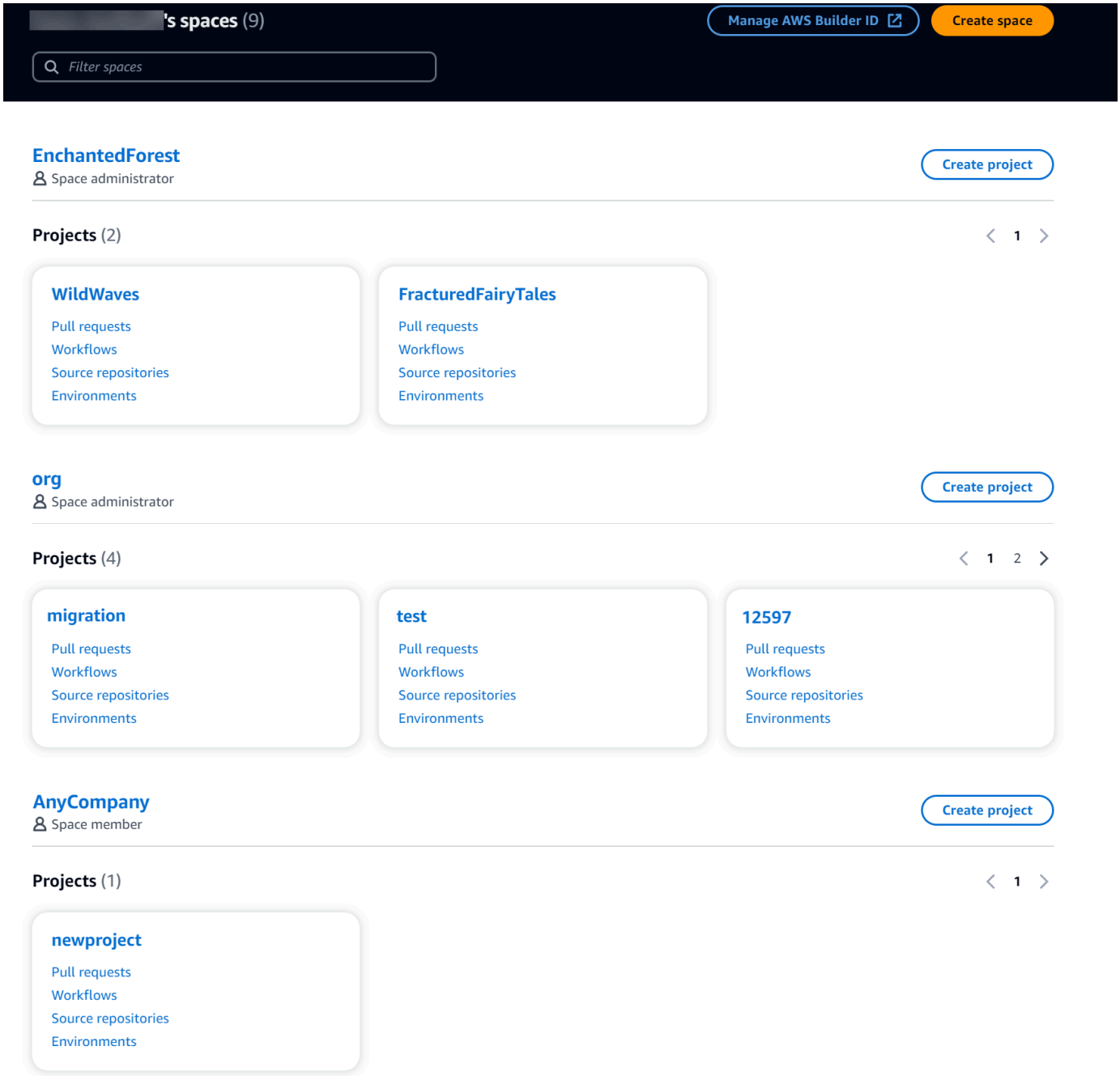
使用 SSO 登录

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在选择登录选项下，选择使用单点登录（SSO）。
3. 在 AWS Identity Center 应用程序名称中，输入身份联合验证管理员提供的应用程序名称。
4. 选择继续使用 IAM Identity Center。

查看用户的所有空间和项目

您可以在用户主页上查看您的空间和项目的列表。用户主页显示用户所属的每个空间、用户在该空间中的角色（例如空间管理员）以及用户在其中拥有成员资格的每个空间中的项目的列表。

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在浏览器中输入以下地址：https://codecatalyst.aws/home



3. 选择要打开的空间或项目。如果您没有看到预期的空间或项目，则可能需要以其他用户的身份登录。

查看和管理 CodeCatalyst 配置文件

您可以查看 Amazon CodeCatalyst 中的用户个人资料，以获取电子邮件地址和 CodeCatalyst 别名等信息。您也可以更新自己的配置文件和 AWS 构建者 ID。如果您忘记了密码，可以请求重置密码。

查看 CodeCatalyst 配置文件

您在注册时提供的信息将用作登录 Amazon CodeCatalyst 的凭证，并将在您的配置文件中进行管理。这包括您的名称、昵称和用于登录 CodeCatalyst 的电子邮件地址。

> **Note**
>
> AWS 构建者 ID 昵称不是您的 CodeCatalyst 别名。您在注册时选择了自己的 CodeCatalyst 别名。

查看您的 CodeCatalyst 配置文件

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择我的设置。这将打开 CodeCatalyst 我的设置页面。
3. 要更新您的 AWS 构建者 ID 电子邮件地址或密码，或者要设置 MFA，请选择管理 AWS 构建者 ID。这将打开 AWS 构建者 ID 页面。

查看其他用户的 CodeCatalyst 配置文件

查看其他用户的 CodeCatalyst 配置文件

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在侧边导航中，选择项目设置。选择成员选项卡。查看您的 CodeCatalyst 项目的成员列表。
3. 选择要查找或 @ 的成员名称。我的设置页面将显示用户的别名、电子邮件地址和全名。使用 CodeCatalyst 别名来 @ 项目成员。

   > **Note**
   >
   > 用户的 AWS 构建者 ID 昵称不是他们的 CodeCatalyst 别名。他们在注册时选择了自己的 CodeCatalyst 别名。

   要查看项目中其他用户的配置文件，请在列表中选择其名称。

创建个人资料

在 CodeCatalyst 中，您的个人资料包括按 AWS 构建者 ID 管理的个人信息以及在 CodeCatalyst 中管理的设置。

- 您的个人资料的全名、电子邮件地址和密码按 AWS 构建者 ID 管理。您在注册时输入了这些信息。当您将 MFA 设置为使用身份验证器应用程序来登录应用程序时，CodeCatalyst 会将您转到 AWS 构建者 ID 页面。
- 您的个人访问令牌（PAT）、CodeCatalyst 通知和语言首选项等 CodeCatalyst 设置，在 CodeCatalyst 的 我的设置页面中进行管理。有关更多信息，请参阅使用个人访问令牌向用户授予对存储库的访问权限。

> **Note**
>
> 您可以更新您的 AWS 构建者 ID 全名（CodeCatalyst 显示名称）和名字。但是，您无法更改 CodeCatalyst 别名。

更新 AWS 构建者 ID 或电子邮件地址

更新您的 AWS 构建者 ID 或电子邮件地址

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择我的设置。这将打开 CodeCatalyst 我的设置页面。
3. 在个人资料页面上，选择管理 AWS 构建者 ID。这将打开 AWS 构建者 ID 页面。
4. 在该页面的左侧，选择我的详细信息。
5. 在个人资料信息下，选择编辑以更新您的姓名或昵称。如果您未指定昵称，昵称字段将体现全名中的名字。这不是您的 CodeCatalyst 别名。

   > **Note**
   >
   > 这将更新 AWS 构建者 ID 的全名和名字。这不会更新 CodeCatalyst 别名。

   在联系人信息下，选择编辑以更新您的电子邮件地址。

> **Note**
>
> 这会更新您用于登录 CodeCatalyst 的电子邮件地址。

更改与 AWS 构建者 ID 关联的 CodeCatalyst 密码

按照以下说明操作，更改与您的 AWS 构建者 ID 关联的 Amazon CodeCatalyst 密码。

> **Note**
>
> 如果您使用 SSO 登录 CodeCatalyst，请让您的管理员更改您的密码。

更改 CodeCatalyst 密码

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择用户个人资料。这将打开 CodeCatalyst 我的设置页面。
3. 在个人资料页面上，选择管理 AWS 构建者 ID。这将打开 AWS 构建者 ID 页面。
4. 在该页面的左侧，选择安全性。
5. 选择更改密码，然后按照说明进行操作。

设置为 AWS CLI 与一起使用 CodeCatalyst

您可以在 Amazon CodeCatalyst 控制台上处理大部分日常任务。但是，当你使用开发环境、个人访问令牌或事件日志 AWS CLI 时，你可能需要设置和配置 CodeCatalyst。必须先安装 AWS CLI 并配置配置文件，然后才能将其与一起使用 CodeCatalyst。

要设置 fo AWS CLI r CodeCatalyst

1. 安装最新版本的 AWS CLI。如果您已经 AWS CLI 安装了某个版本，请确保该版本是最新版本并包含的命令 CodeCatalyst，并在需要时对其进行更新。要验证您是否安装了包含 CodeCatalyst 命令的版本，请打开命令提示符并运行以下命令：

```
aws codecatalyst help
```

如果您看到 CodeCatalyst 命令列表，则表示您的版本支持 CodeCatalyst。如果无法识别该命令，请将您的版本更新 AWS CLI 到最新版本。有关更多信息，请参阅 AWS Command Line Interface 用户指南 AWS CLI中的安装或更新最新版本的。

2. 如果您没有配置文件或者想要使用专门的命名配置文件，请运行aws configure命令来创建配置文件 CodeCatalyst。我们建议创建一个专门用于的命名配置文件 CodeCatalyst，但您也可以使用默认配置文件。有关更多信息，请参阅配置基础知识。
3. 编辑配置config文件以添加用于连接的部分， CodeCatalyst 如下所示。config 文件位于 ~/.aws/config（在 Linux 或 macOS 上）或 C:\Users*USERNAME*\.aws\config（在 Windows 上）。

```
[profile codecatalyst]
region = us-west-2
sso_session = codecatalyst

[sso-session codecatalyst]
sso_region = us-east-1
sso_start_url = https://view.awsapps.com/start
sso_registration_scopes = codecatalyst:read_write
```

4. 保存该文件。
5. 在尝试运行任何 CodeCatalyst 命令之前，请打开新的终端或命令提示符并运行以下命令来请求和检索aws codecatalyst用于运行命令的凭据。如果需要，请将 codecatalyst 替换为您的个人资料名称。

```
aws sso login --profile codecatalyst
```

要查看 codecatalyst 命令示例，请参阅以下主题：

- 使用个人访问令牌向用户授予对存储库的访问权限
- 使用事件日志记录访问已记录的事件

入门教程

Amazon CodeCatalyst 提供了许多不同的模板，可帮助您启动自己的项目。您也可以选择从空项目开始，然后向其中添加资源。按照这些教程中的步骤，了解您可以在 CodeCatalyst 中使用的一些方法。

如果这是您首次使用 CodeCatalyst，建议您从教程：使用现代三层 Web 应用程序蓝图创建项目开始。

> **Note**
>
> 要按照这些教程操作，您必须先完成设置。有关更多信息，请参阅 设置并登录 CodeCatalyst。

主题

- 教程：使用现代三层 Web 应用程序蓝图创建项目
- 教程：从一个空项目开始，然后手动添加资源
- 教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作

有关侧重于 CodeCatalyst 中特定功能区域的更多教程，请参阅：

- 开始使用 Slack 通知
- 开始使用 CodeCatalyst 源存储库和单页应用程序蓝图
- 入门工作流
- 开始使用自定义蓝图
- 开始使用 Amazon CodeCatalyst 操作开发者指南

有关可供深入学习的教程，请参阅：

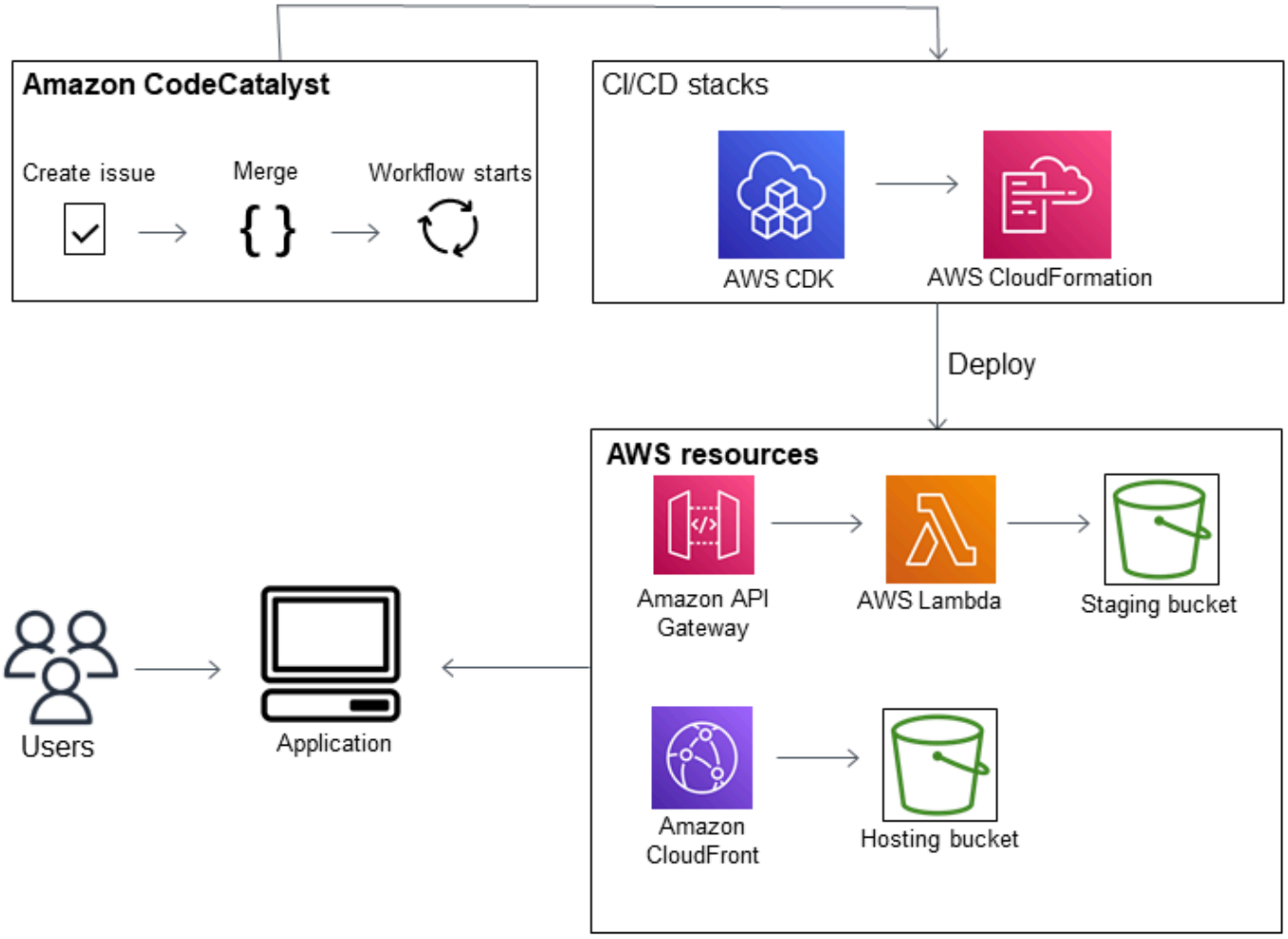
- 教程：将构件上传到 Amazon S3
- 教程：部署无服务器应用程序
- 教程：将应用程序部署到 Amazon ECS
- 教程：将应用程序部署到 Amazon EKS
- 教程：使用 GitHub Action 对代码进行 lint 检查
- 教程：创建和更新 React 应用程序

教程：使用现代三层 Web 应用程序蓝图创建项目

通过使用蓝图创建项目，您可以更快地着手开发软件。使用蓝图创建的项目中包含所需的资源，包括用于管理代码的源存储库以及用于构建和部署应用程序的工作流。在本教程中，我们将引导您使用现代化三层 Web 应用程序蓝图，在 Amazon CodeCatalyst 中创建一个项目。本教程还包括查看已部署的示例；邀请其他用户共同处理；以及使用拉取请求对代码进行更改，在拉取请求合并时，这些更改会自动构建并部署到所连接的 AWS 账户中的资源。CodeCatalyst 会使用报告、活动源和其他工具创建您的项目，而您的蓝图则在与您的项目关联的 AWS 账户中创建 AWS 资源。使用蓝图文件，您可以构建和测试示例现代化应用程序，并将应用程序部署到 AWS Cloud中的基础设施。

下图演示了如何使用 CodeCatalyst 中的工具来创建用于跟踪、合并和自动构建更改的事务，然后在 CodeCatalyst 项目中启动一个工作流，该工作流会运行操作来允许 AWS CDK 和 CloudFormation 预置您的基础设施。

这些操作在关联的 AWS 账户中生成资源，并将您的应用程序部署到带 API Gateway 端点的无服务器 AWS Lambda 函数。AWS Cloud Development Kit (AWS CDK) 操作将一个或多个 AWS CDK 堆栈转换为 CloudFormation 模板，并将堆栈部署到您的 AWS 账户。堆栈中的资源包括 Amazon CloudFront 资源（用于分发动态 Web 内容）、Amazon DynamoDB 实例（用于存储您的应用程序数据）以及支持已部署应用程序的角色和策略。



使用现代化三层 Web 应用程序蓝图创建项目时，将使用以下资源创建您的项目：

在 CodeCatalyst 项目中：

- 包含示例代码和工作流 YAML 的源存储库
- 一个工作流，每当对默认分支进行更改时都会构建并部署示例代码
- 事务面板和待办事项，可用于计划和跟踪工作
- 测试报告套件，其中带有示例代码中包含的自动化报告

在关联的 AWS 账户中：

- 三个 AWS CloudFormation 堆栈，用于创建应用程序所需的资源。

有关将作为本教程的一部分在 AWS 和 CodeCatalyst 中创建的资源的更多详细信息，请参阅参考。

> **Note**
>
> 项目中包含的具体资源和示例取决于您选择的蓝图。Amazon CodeCatalyst 提供了多个项目蓝图，这些蓝图定义了与其定义的语言或框架相关的资源。要了解有关蓝图的更多信息，请参阅使用 CodeCatalyst 蓝图创建综合项目。

主题

先决条件

要在本教程中创建现代化应用程序项目，必须按以下所示完成设置并登录 CodeCatalyst 中的任务：

- 拥有用于登录 CodeCatalyst 的 AWS 构建者 ID。
- 属于某个空间，并且在该空间中已为您分配了空间管理员或高级用户角色。有关更多信息，请参阅创建空间、向用户授予空间权限和空间管理员角色。
- 具有与您的空间关联的 AWS 账户，并具有您在注册期间创建的 IAM 角色。例如，在注册期间，您可以选择使用名为 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 的角色策略来创建服务角色。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。有关创建角色的步骤，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。

步骤 1：创建现代化三层 Web 应用程序项目

创建完您的项目后，您将在项目中开发和测试代码、协调开发任务以及查看项目指标。项目中还包含您的开发工具和资源。

在本教程中，您将使用现代化三层 Web 应用程序蓝图来创建交互式应用程序。作为项目一部分自动创建和运行的工作流将会构建和部署应用程序。只有在为您的空间配置了所有角色和账户信息后，工作流才能成功运行。成功运行工作流后，您可以访问端点 URL 来查看应用程序。

使用蓝图创建项目

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要在其中创建项目的空间。
3. 选择创建项目。
4. 选择从蓝图开始。
5. 在搜索栏中输入 **modern**。
6. 选择现代化三层 Web 应用程序蓝图，然后选择下一步。
7. 在为您的项目命名中，输入项目名称。例如：

   MyExampleProject.

   > **Note**
   >
   > 该名称在空间内必须是唯一的。

8. 在账户中，选择您在注册时添加的 AWS 账户。蓝图会将资源安装到此账户。
9. 在部署角色中，选择您在注册期间添加的角色。例如，选择 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。

   如果未列出任何角色，请添加一个。要添加角色，请选择添加 IAM 角色，然后将角色添加到您的 AWS 账户。有关更多信息，请参阅 允许在已连接的情况下访问 AWS 资源 AWS 账户。
10. 在计算平台中，选择 Lambda。
11. 在前端托管选项中，选择 Amplify Hosting。有关 AWS Amplify 的更多信息，请参阅《AWS Amplify User Guide》中的 What is AWS Amplify Hosting?
12. 在部署区域中，输入您希望蓝图用于部署 Mysfits 应用程序和支持资源的 AWS 区域的区域代码。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。
13. 在应用程序名称中，保留默认值 `mysfits`*`string`*。

14. （可选）在生成项目预览下，选择查看代码以预览蓝图将安装的源文件。选择查看工作流，预览蓝图将安装的 CI/CD 工作流定义文件。预览会根据您的选择动态地更新。
15. 选择创建项目。

在您创建项目时，会立即启动项目工作流。完成代码的构建和部署需要一些时间。在此期间您可继续并邀请其他人加入您的项目。

步骤 2：邀请他人加入您的项目

现在您已设置了项目，请邀请其他人与您合作。

邀请他人加入您的项目

1. 导航到要邀请用户的项目。
2. 在导航窗格中，选择项目设置。
3. 在成员选项卡上，选择邀请。
4. 键入要邀请的用户的电子邮件地址，该用户将作为您项目的用户。您可以键入多个电子邮件地址，用空格或逗号分隔。您也可以从空间成员而不是项目成员中进行选择。
5. 选择此用户的角色。

   添加完用户后，选择邀请。

步骤 3：创建事务以协作处理和跟踪工作

CodeCatalyst 可以协助您使用事务跟踪项目中涉及的功能、任务、错误以及其他任何工作。您可以创建事务来跟踪所需的工作和想法。默认情况下，创建事务时，事务会添加到待办事项中。您可以将事务移动到面板，在其中跟踪正在进行的工作。您也可以将事务分配给特定的项目成员。

为项目创建事务

1. 在导航窗格中，选择事务。
2. 选择创建事务。
3. 在事务标题中，提供事务的名称。可选择提供事务的描述。在此示例中，使用 **`make a change in the src/mysfit_data.json file.`**
4. 选择优先级、估算值、状态和标签。在被分派人下，选择 + 添加我，将事务分配给自己。
5. 选择创建事务。该事务现已显示在面板上。选择卡片，将事务移动到进行中列。

有关更多信息，请参阅 跟踪和组织处理问题的工作 CodeCatalyst。

步骤 4：查看您的源存储库

蓝图会安装一个源存储库，其中包含用于定义和支持您的应用程序或服务的文件。源存储库中一些需要注意的目录和文件包括：

- .cloud9 目录 – 包含 AWS Cloud9 开发环境的支持文件。
- .codecatalyst 目录 – 包含蓝图中的每个工作流的 YAML 工作流定义文件。
- .idea 目录 – 包含 JetBrains 开发环境的支持文件。
- .vscode 目录 – 包含 Visual Studio 代码开发环境的支持文件。
- cdkStacks 目录 – 包含在 AWS Cloud中定义基础设施的 AWS CDK 堆栈文件。
- src 目录 – 包含应用程序源代码。
- tests 目录 – 包含 integ 和单元测试的文件，在您构建和测试应用程序时，这些文件包含在所运行的自动化 CI/CD 工作流中一起运行。
- web 目录 – 包含前端源代码。其他文件包括项目文件，例如包含项目重要元数据的 `package.json` 文件、网站的 `index.html` 页面、用于检查代码的 `.eslintrc.cjs` 文件，以及用于指定根文件和编译器选项的 `tsconfig.json` 文件。
- `Dockerfile` 文件 – 描述应用程序的容器。
- `README.md` 文件 – 包含项目的配置信息。

导航到项目的源存储库

1. 导航到您的项目，然后执行下列操作之一：
   - 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。
   - 在导航窗格中，选择代码，然后选择源存储库。在源存储库中，从列表中选择存储库的名称。您可以通过在筛选栏中键入部分存储库名称，筛选存储库列表。
2. 在存储库主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求和工作流的数量。默认情况下，会显示默认分支的内容。您可通过选择此下拉列表中的其他分支来更改视图。

步骤 5：创建带测试分支的开发环境并快速更改代码

您可以通过创建开发环境，快速处理源存储库中的代码。在本教程中，我们假设您将要：

- 创建 AWS Cloud9 开发环境。

- 创建开发环境时，选择在主分支之外的新分支中工作的选项。
- 为新分支使用名称 `test`。

在后面的步骤中，您将使用开发环境来更改代码并创建拉取请求。

使用新分支创建开发环境

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到要在其中创建开发环境的项目。
3. 从项目的源存储库列表中选择存储库的名称。另外，在导航窗格中，依次选择代码、源存储库以及要为其创建开发环境的存储库。
4. 在存储库主页上，选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 选择要克隆的存储库，选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支自下拉菜单中选择要从中创建新分支的分支。
7. 可选操作，为开发环境添加别名。
8. 可选操作，选择开发环境配置编辑按钮，编辑开发环境的计算、存储或超时配置。
9. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动，开发环境创建完成后，状态列将显示正在运行。将在您选择的 IDE 中打开一个新选项卡，其中包含您的开发环境。您可以编辑代码并提交和推送更改。

在此部分中，您将在 CodeCatalyst 中使用生成的示例应用程序，具体方式是使用拉取请求对代码进行更改，在合并拉取请求时会自动构建这些更改并部署到所连接的 AWS 账户中的资源。

在 **`src/mysfit_data.json`** 文件中进行更改

1. 导航到您的项目开发环境。在 AWS Cloud9 中，展开侧导航菜单，浏览文件。展开 `mysfits`、`src`，然后打开 `src/mysfit_data.json`。
2. 在文件中，将 `"Age":` 字段的值从 6 更改为 12。此行应类似于下文：

```
   {
       "Age": 12,
       "Description": "Twilight's personality sparkles like the night sky and is
looking for a forever home with a Greek hero or God. While on the smaller side
at 14 hands, he is quite adept at accepting riders and can fly to 15,000 feet.
Twilight needs a large area to run around in and will need to be registered with
```

```
 the FAA if you plan to fly him above 500 feet. His favorite activities include
 playing with chimeras, going on epic adventures into battle, and playing with a
 large inflatable ball around the paddock. If you bring him home, he'll quickly
 become your favorite little Pegasus.",
        "GoodEvil": "Good",
        "LawChaos": "Lawful",
        "Name": "Twilight Glitter",
        "ProfileImageUri": "https://www.mythicalmysfits.com/images/
pegasus_hover.png",
        "Species": "Pegasus",
        "ThumbImageUri": "https://www.mythicalmysfits.com/images/pegasus_thumb.png"
    },
```

3. 保存该文件。
4. 使用 **cd /projects/mysfits** 命令切换到 mysfits 存储库。
5. 使用 git add、git commit 和 git push 命令添加、提交和推送您的更改。

```
git add .
git commit -m "make an example change"
git push
```

步骤 6：查看构建现代化应用程序的工作流

创建现代化应用程序项目后，CodeCatalyst 会代表您生成多个资源，包括工作流。工作流是在 .yaml 文件中定义的自动化过程，用于描述如何构建、测试和部署代码。

在本教程中，CodeCatalyst 创建了一个工作流，并且在您创建项目时自动启动该工作流。（工作流可能仍在运行，具体取决于您创建项目后经过的时间。）使用以下过程检查工作流的进度，查看生成的日志和测试报告，最后导航到已部署应用程序的 URL。

查看工作流进度

1. 在 CodeCatalyst 控制台的导航窗格中，选择 CI/CD，然后选择工作流。

   此时将显示工作流列表。下面是 CodeCatalyst 蓝图在您创建项目时生成和启动的工作流。
2. 查看工作流列表。您应该会看到四个工作流：

   - 上面的两个工作流对应于您先前在步骤 5：创建带测试分支的开发环境并快速更改代码中创建的 `test` 分支。这些工作流是 `main` 分支上的工作流的副本。ApplicationDeploymentPipeline 未处

于活动状态，因为它已配置为与分支 `main` 一起使用。OnPullRequest 工作流未运行，因为尚未发出拉取请求。

- 下面的两个工作流对应于您先前运行蓝图时创建的 `main` 分支。ApplicationDeploymentPipeline 工作流处于活动状态，并且正在运行（或已完成运行）。

> **Note**
>
> 如果 ApplicationDeploymentPipeline 运行失败并出现 Build@cdk_bootstrap 或 DeployBackend 错误，那么可能是因为您先前运行过现代化三层 Web 应用程序，留下了与当前蓝图冲突的旧资源。您需要删除这些旧资源，然后重新运行工作流。有关更多信息，请参阅 清理资源。

3. 在底部选择与 `main` 分支关联的 ApplicationDeploymentPipeline 工作流。此工作流是使用 `main` 分支上的源代码运行的。

   此时将出现工作流图表。该图表中会显示多个方块，每个方块代表一项任务或操作。大多数操作是垂直排列的，上面的操作在下面的操作之前运行。并排排列的操作会并行运行。分组在一起的操作必须全部成功运行，然后才能启动其后的操作。

   主要方块包括：

   - WorkflowSource – 此方块代表您的源存储库。在其他信息之外，它还显示源存储库名称（mysfits）和自动启动工作流运行的提交操作。CodeCatalyst 在您创建项目时生成了该提交操作。
   - Build – 此方块代表包含两个操作的分组，这两个操作都必须成功完成，才能开始下一个操作。
   - DeployBackend – 此方块代表将应用程序的后端组件部署到 AWS 云中的操作。
   - Tests – 此方块代表包含两个测试操作的分组，这两个操作都必须成功完成，才能开始下一个操作。
   - DeployFrontend – 此方块代表将应用程序的前端组件部署到 AWS 云中的操作。

4. 选择定义选项卡（靠近顶部）。工作流定义文件显示在右侧。此文件包含以下需要注意的部分：

   - 位于顶部的 `Triggers` 部分。该部分指出，每当将代码推送到源存储库的 `main` 分支时，工作流都必须启动。推送到其他分支（例如 `test`）不会启动此工作流。工作流会使用 `main` 分支上的文件运行。
   - 位于 `Triggers` 下的 `Actions` 部分。此部分定义了您在工作流图表中看到的操作。

5. 选择最新状态选项卡（靠近顶部），然后在工作流图表中选择任意操作。

6. 在右侧，选择配置选项卡，查看操作在最近一次运行期间使用的配置设置。在工作流定义文件中，每个配置设置都有一个匹配的属性。
7. 将控制台保持打开状态，然后转到下一个步骤。

查看构建日志和测试报告

1. 选择最新状态选项卡。
2. 在工作流图表中，选择 DeployFrontend 操作。
3. 等待操作完成。注意“进行中”图标
   （ ）
   会变成“成功”图标
   （ ）。
4. 选择 build_backend 操作。
5. 选择日志选项卡，然后展开几个部分，查看这些步骤的日志消息。您可以看到与后端设置相关的消息。
6. 选择报告选项卡，然后选择 `backend-coverage.xml` 报告。此时 CodeCatalyst 将显示关联的报告。该报告显示已运行的代码覆盖率测试，并指出通过测试成功验证的代码行比例，例如 80%。

   有关测试报告的更多信息，请参阅使用工作流进行测试。

   > **Tip**
   >
   > 您也可以通过在导航窗格中选择报告，查看您的测试报告。

7. 将 CodeCatalyst 控制台保持打开状态，然后转到下一个步骤。

确认现代化应用程序已成功部署

1. 返回 ApplicationDeploymentPipeline 工作流，然后选择最新运行的 Run-***string*** 链接。
2. 在工作流图表中，找到 DeployFrontend 操作并选择查看应用程序链接。此时将显示 Mysfit 网站。

> **Note**
>
> 如果您在 DeployFrontend 操作中看不到查看应用程序链接，请确保您选择了运行 ID 链接。

3. 搜索名为 Twilight Glitter 的 pegasus Mysfit。记下年龄值。该值为 6。您将更改代码来更新年龄。

步骤 7：让其他人审查您的更改

现在，您已经在名为 `test` 的分支中进行了更改，可以通过创建拉取请求来要求其他人对更改进行审查。执行以下步骤创建拉取请求，以便将更改从 `test` 分支合并到 `main` 分支中。

创建拉取请求

1. 导航到您的项目。
2. 请执行以下操作之一：
   - 在导航窗格中，依次选择代码、拉取请求和创建拉取请求。
   - 在存储库主页上，选择更多，然后选择创建拉取请求。
   - 在项目页面上，选择创建拉取请求。
3. 在源代码库中，确保指定的源存储库是包含所提交代码的存储库。只有在您未从存储库的主页创建拉取请求时，才会显示此选项。
4. 在目标分支中，在审查代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含所提交代码的分支。
6. 在拉取请求标题中，输入一个标题，帮助其他用户了解需要审查的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如事务链接或所做更改的描述之类的信息。

> **Tip**
>
> 您可以选择为我编写描述，让 CodeCatalyst 自动生成拉取请求中包含的更改的描述。您可以在将自动生成的描述添加到拉取请求后，对描述进行更改。
> 此功能要求为空间启用生成式人工智能功能，并且不适用于链接的存储库中的拉取请求。
> 有关更多信息，请参阅 Managing generative AI features。

8. （可选）在事务中，选择关联事务，然后从列表中选择事务或输入事务 ID。要取消链接事务，请选择“取消链接”图标。

9. （可选）在必需的审阅者中，选择添加必需审阅者。从项目成员列表中进行选择，添加审阅者。必需的审阅者必须批准更改，才能将拉取请求合并到目标分支。

> **Note**
>
> 您不能将审阅者同时添加为必需的审阅者和可选的审阅者。您不能将自己添加为审阅者。

10. （可选）在可选的审阅者中，选择添加可选审阅者。从项目成员列表中进行选择，添加审阅者。可选的审阅者不必批准更改，这并不是将拉取请求合并到目标分支之前的必备要求。
11. 审查分支之间的差异。拉取请求中显示的差异是源分支中的修订与合并基准之间的更改，而合并基准是创建拉取请求时目标分支的 HEAD 提交。如果未显示任何更改，则分支可能相同，或者您可能为源和目标选择了相同的分支。
12. 如果您对拉取请求中包含的希望审查的代码和更改感到满意，请选择创建。

> **Note**
>
> 创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

创建拉取请求时，OnPullRequest 工作流会开始使用 `test` 分支中的源文件。当审阅者批准您的代码更改后，您可以通过选择工作流并查看测试输出来观察结果。

审查完更改后，您就可以合并代码了。将代码合并到默认分支时，将自动启动构建和部署更改的工作流。

合并来自 CodeCatalyst 控制台的拉取请求

1. 导航到您的现代化应用程序项目。
2. 在项目页面上的打开拉取请求下，选择要合并的拉取请求。如果您没有看到拉取请求，请选择查看全部，然后从列表中选择。选择合并。
3. 从拉取请求的可用合并策略中选择。（可选）选择或取消选择在合并拉取请求后删除源分支的选项，然后选择合并。

> **Note**
>
> 如果合并按钮未激活，或者您看到不可合并标签，则说明一个或多个必需的审阅者尚未批准拉取请求，或者无法在 CodeCatalyst 控制台中合并拉取请求。在拉取请求详细信息区域的概述中，使用时钟图标来指示尚未批准拉取请求的审阅者。如果所有必需的审阅者都已批准了拉取请求，但合并按钮仍未激活，则可能存在合并冲突。您可以在 CodeCatalyst 控制台中解决目标分支的合并冲突，然后合并拉取请求；也可以解决冲突并在本地合并，然后将包含合并的提交操作推送到 CodeCatalyst。有关更多信息，请参阅合并拉取请求（Git）和您的 Git 文档。

将更改从 `test` 分支合并到 **`main`** 分支后，更改会自动启动 ApplicationDeploymentPipeline 工作流，该工作流将构建并部署您的更改。

查看通过 ApplicationDeploymentPipeline 工作流运行的合并提交

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 在工作流的 ApplicationDeploymentPipeline 中，展开最近的运行。您可以看到由合并提交启动的工作流运行。（可选）选择工作流运行来查看运行进度。
3. 运行完成后，重新加载您先前访问过的 URL。查看 pegasus 以验证年龄是否已更改。

步骤 8：关闭事务

解决事务后，可以在 CodeCatalyst 控制台上关闭事务。

关闭项目的事务

1. 导航到您的项目。
2. 在导航窗格中，选择事务。
3. 将事务拖放到完成列。

有关更多信息，请参阅 跟踪和组织处理问题的工作 CodeCatalyst。

清理资源

在 CodeCatalyst 和 AWS 中进行清理以从环境中移除本教程的跟踪。

您可以选择继续使用您用于本教程的项目，也可以删除此项目及其关联资源。

> **Note**
>
> 删除此项目时，将会删除项目中所有成员的所有存储库、事务和构件。

删除项目

1. 导航到您的项目，然后选择项目设置。
2. 选择常规选项卡。
3. 在项目名称下，选择删除项目。

删除 CloudFormation 和 Amazon S3 中的资源

1. 使用您添加到 CodeCatalyst 空间的相同账户登录到 AWS 管理控制台。
2. 转到 CloudFormation 服务。
3. 删除 mysfits***string*** 堆栈。
4. 删除 development-mysfits***string*** 堆栈。
5. 选择（但不要删除）CDKToolkit 堆栈。选择资源选项卡。选择 StagingBucket 链接，然后删除 Amazon S3 中的存储桶和存储桶内容。

> **Note**
>
> 如果您不手动删除此存储桶，则在重新运行现代化三层 Web 应用程序蓝图时可能会出现错误。

6. （可选）删除 CDKToolkit 堆栈。

参考

现代化三层 Web 应用程序蓝图会将资源部署到您的 CodeCatalyst 空间以及 AWS 云中您的 AWS 账户。这些资源包括：

- 在您的 CodeCatalyst 空间中：
  - 一个 CodeCatalyst 项目，其中包含以下资源：
    - 源代码库 – 此存储库包含“Mysfits”Web 应用程序的示例代码。
    - 工作流 – 每当对默认分支进行更改时，此工作流都会构建并部署 Mysfits 应用程序代码
    - 事务面板和待办事项 – 此面板和待办事项可用于计划和跟踪工作。
    - 测试报告套件 – 此套件包括示例代码中包含的自动化报告。
- 在关联的 AWS 账户中：
  - CDKToolkit 堆栈 – 此堆栈部署以下资源：
    - Amazon S3 暂存存储桶、存储桶策略以及用于加密存储桶的 AWS KMS 密钥。
    - 部署操作的 IAM 部署角色。
    - 用于支持堆栈中的资源的 AWS IAM 角色和策略。

    > **Note**
    >
    > CDKToolkit 不会针对每次部署彻底停止并重新创建。这是在每个账户中启动的堆栈，用于支持 AWS CDK。

  - development-mysfits*string*BackEnd 堆栈 – 此堆栈部署以下后端资源：
    - Amazon API Gateway 端点。
    - 用于支持堆栈中的资源的 AWS IAM 角色和策略。
    - AWS Lambda 函数和层为现代化应用程序提供了无服务器计算平台。
    - 用于存储桶部署和 Lambda 函数的 IAM 策略和角色。

- mysfits***string*** 堆栈 – 此堆栈部署 AWS Amplify 前端应用程序。

另请参阅

有关在本教程中用于创建资源的 AWS 服务的更多信息，请参阅以下内容：

- Amazon S3 – 一项服务，用于将前端资产存储在对象存储服务上，可提供行业领先的可扩展性、数据高可用性、安全性和性能。有关更多信息，请参阅《Amazon S3 用户指南》。
- Amazon API Gateway – 一项服务，用于创建、发布、维护、监控和保护任意规模的 REST、HTTP 和 WebSocket API。有关更多信息，请参阅《API Gateway 开发人员指南》。
- Amplify – 一项用于托管前端应用程序的服务。有关更多信息，请参阅《AWS Amplify Hosting User Guide》。
- AWS Cloud Development Kit (AWS CDK) – 一个框架，用于在代码中定义云基础设施并通过 AWS CloudFormation 进行预置。AWS CDK 包括 AWS CDK Toolkit，后者是一个用于与 AWS CDK 应用程序和堆栈交互的命令行工具。有关更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) 开发人员指南》。
- Amazon DynamoDB – 一个用于存储服务的完全托管式 NoSQL 数据库服务。有关更多信息，请参阅《Amazon DynamoDB Developer Guide》。
- AWS Lambda – 一项服务，用于在高可用性计算基础设施上调用您的代码，而无需预置或管理服务器。有关更多信息，请参阅《AWS Lambda 开发人员指南》。
- AWS IAM – 一项服务，用于安全地控制对 AWS 及其资源的访问。有关更多信息，请参阅《IAM 用户指南》。

教程：从一个空项目开始，然后手动添加资源

通过在创建项目时选择空项目蓝图，可以创建一个内部不含任何预定义资源的空项目。创建空项目后，您可以根据项目需求创建和添加资源。由于在不使用蓝图的情况下创建的项目在创建时为空，因而需要有更多有关创建和配置 CodeCatalyst 资源的知识，才能开始使用此选项。

主题

- 邀请他人加入您的项目
- 创建要协作处理并跟踪工作的事务

先决条件

要创建空项目，必须为您分配空间管理员或高级用户角色。如果这是您首次登录 CodeCatalyst，请参阅设置并登录 CodeCatalyst。

创建空项目

要协同工作，首先要创建项目。如果您想创建自己的资源（例如源存储库和工作流），则可以从一个空项目开始。

创建空项目

1. 导航到要在其中创建项目的空间。
2. 在空间控制面板上，选择创建项目。
3. 选择从头开始。
4. 在为项目命名下，输入要分配给项目的名称。该名称在空间内必须是唯一的。
5. 选择创建项目。

现在您有了一个空项目，下一步是创建源存储库。

创建源存储库

创建源存储库，用于存储和协作处理项目代码。项目成员可以将此存储库克隆到其本地计算机上以处理代码。或者，您可以选择关联在受支持服务中托管的存储库，但本教程未对此进行介绍。有关更多信息，请参阅链接源存储库。

创建源存储库

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的项目。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。

5. 在存储库名称中，输入存储库的名称。在本指南中，我们使用 *codecatalyst-source-repository*，不过您可以选择不同的名称。一个项目中的存储库名称必须唯一。有关存储库名称要求的更多信息，请参阅中的源存储库配额 CodeCatalyst。
6. （可选）在描述中，添加对存储库的描述，这有助于项目中的其他用户了解存储库的用途。
7. 选择创建存储库（默认）。此选项会创建一个存储库，其中包含默认分支和 README.md 文件。与空存储库不同，您可以在创建此存储库后立即使用它。
8. 在默认分支中，除非您有理由选择其他名称，否则请将该名称保留为 main。本指南中的示例对于默认分支均使用名称 main。
9. （可选）为您计划推送的代码类型添加一个 `.gitignore` 文件。
10. 选择创建。

> **Note**
>
> 创建存储库时，CodeCatalyst 会将一个 `README.md` 文件添加到存储库中。CodeCatalyst 还在名为 main 的默认分支中为存储库创建一个初始提交。您可以编辑或删除 README.md 文件，但无法删除默认分支。

您可以通过创建开发环境在存储库中快速添加代码。在本教程中，我们建议您使用 AWS Cloud9 创建开发环境，并在创建开发环境时选择从 main 分支创建分支的选项。我们为此分支使用名称 **`test`**，但如果您愿意，可以输入不同的分支名称。

使用新分支创建开发环境

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到要在其中创建开发环境的项目。
3. 从项目的源存储库列表中选择存储库的名称。另外，在导航窗格中，依次选择代码、源存储库以及要为其创建开发环境的存储库。
4. 在存储库主页上，选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 选择要克隆的存储库，选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支自下拉菜单中选择要从中创建新分支的分支。
7. 可选操作，为开发环境添加别名。
8. 可选操作，选择开发环境配置编辑按钮，编辑开发环境的计算、存储或超时配置。

9. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动，开发环境创建完成后，状态列将显示正在运行。将在您选择的 IDE 中打开一个新选项卡，其中包含您的开发环境。您可以编辑代码并提交和推送更改。

创建用于构建、测试和部署代码更改的工作流

在 CodeCatalyst 中，您可以在工作流中组织应用程序或服务的构建、测试和部署。工作流由若干操作组成，可以配置为在发生指定的源存储库事件（例如代码推送，或者打开或更新拉取请求）后自动运行。有关工作流的更多信息，请参阅使用工作流进行构建、测试和部署。

按照入门工作流中的说明创建您的第一个工作流。

邀请他人加入您的项目

现在您已设置了自定义项目，请邀请其他人与您合作。

邀请他人加入您的项目

1. 导航到要邀请用户的项目。
2. 在导航窗格中，选择项目设置。
3. 在成员选项卡上，选择邀请。
4. 键入要邀请的用户的电子邮件地址，该用户将作为您项目的用户。您可以键入多个电子邮件地址，用空格或逗号分隔。您也可以从空间成员而不是项目成员中进行选择。
5. 选择此用户的角色。

   添加完用户后，选择邀请。

创建要协作处理并跟踪工作的事务

CodeCatalyst 可以协助您使用事务跟踪项目中涉及的功能、任务、错误以及其他任何工作。您可以创建事务来跟踪所需的工作和想法。默认情况下，创建事务时，事务会添加到待办事项中。您可以将事务移动到面板，在其中跟踪正在进行的工作。您也可以将事务分配给特定的项目成员。

为项目创建事务

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。

请确保您正在将要创建事务的项目中导航。要查看所有项目，请在导航窗格中选择 Amazon CodeCatalyst，如果需要，请选择查看所有项目。选择要在其中创建或处理事务的项目。

2. 在导航窗格中，选择跟踪，然后选择待办事项。
3. 选择创建事务。
4. 在事务标题中，提供事务的名称。可选择提供事务的描述。如果需要，可以选择事务的状态、优先级和估计值。您也可以将事务分配给项目成员列表中的项目成员。

> **Tip**
>
> 您可以选择将事务分配给 Amazon Q，让 Amazon Q 尝试解决该事务。如果尝试成功，系统将创建拉取请求，事务状态将更改为审核中，以便您可以查看和测试代码。有关更多信息，请参阅 教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作。
> 此功能要求为空间启用生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。

5. 选择保存。

创建事务后，您可以将其分配给项目成员，对其进行估算，然后在看板面板上对其进行跟踪。有关更多信息，请参阅 跟踪和组织处理问题的工作 CodeCatalyst。

教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作

如果您在启用生成人工智能功能的 Amazon CodeCatalyst 中有一个项目和源代码库，则可以使用这些功能来帮助加快软件开发。开发人员要完成的任务往往多于完成任务所需的时间。在创建拉取请求以审查这些更改时，开发人员通常不会花时间向队友解释他们的代码更改，而是期望其他用户发现这些更改一目了然。拉取请求的创建者和审阅者也没有时间彻底查找和阅读拉取请求中的所有评论，尤其是在拉取请求有多个修订版的情况下。 CodeCatalyst 与 Amazon Q Developer Agent 集成以进行软件开发，提供生成式 AI 功能，这些功能既可以帮助团队成员更快地完成任务，又可以增加他们专注于工作中最重要部分的时间。

Amazon Q Developer 是一款基于人工智能的生成式对话助手，可以帮助您理解、构建、扩展和操作 AWS 应用程序。为了加快您的构建 AWS，为Amazon Q提供支持的模型增加了高质量的 AWS 内容，以生成更完整、更具可操作性和参考性的答案。有关更多信息，请参阅《Amazon Q Developer User Guide》中的 What is Amazon Q Developer?。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

在本教程中，您将学习如何使用中的生成式 AI 功能 CodeCatalyst 来帮助您创建带有蓝图的项目，以及如何向现有项目添加蓝图。此外，您将学习如何在创建拉取请求时汇总分支之间的变化，以及如何汇总拉取请求中留下的注释。您还将学习如何根据自己的代码更改或改进想法创建事务并将其分配给 Amazon Q。在处理分配给 Amazon Q 的事务时，您将学习如何让 Amazon Q 能够提出任务建议，以及如何分配和处理它在处理事务时创建的任何任务。

主题

先决条件

要使用本教程中的 CodeCatalyst 功能，您必须先完成并有权访问以下资源：

- 您有 AWS 生成器 ID 或单点登录 (SSO) 身份可供登录。 CodeCatalyst
- 您所在的空间启用了生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。
- 在该空间的项目中，您具有贡献者或项目管理员角色。
- 除非您在使用生成式人工智能创建项目，否则您的现有项目至少有一个为其配置的源存储库。不支持链接的存储库。

- 在分配事务以使用生成式人工智能创建初始解决方案时，无法使用 Jira Software 扩展来配置该项目。该扩展不受此功能的支持。

有关更多信息，请参阅创建空间、跟踪和组织处理问题的工作 CodeCatalyst、为带有扩展程序的项目添加功能 CodeCatalyst和使用用户角色授予访问权限。

本教程基于使用带有 Python 的现代三层 Web 应用程序蓝图创建的项目。如果您使用的项目是用其他蓝图创建的，您仍然可以按照这些步骤进行操作，但有些细节会有所不同，例如示例代码和语言。

创建项目或添加功能时使用 Amazon Q 选择蓝图

作为项目开发人员，在创建新项目或者向现有项目添加组件时，您可以与生成式人工智能助手 Amazon Q 协作。您可以在类似聊天的界面中与 Amazon Q 进行交互，向其提供项目要求。根据您的要求，Amazon Q 会推荐蓝图并概述无法满足的要求。如果您的空间有自定义蓝图，Amazon Q 会学习这些蓝图并将其包含在建议中。然后，如果您对此感到满意，则可根据 Amazon Q 的建议继续执行操作，该工具将创建必要的资源，例如包含满足您要求的代码的源存储库。Amazon Q 还会为蓝图无法满足的要求创建事务。要了解有关可用 CodeCatalyst 蓝图的更多信息，请参阅使用 CodeCatalyst 蓝图创建综合项目。要了解有关将 Amazon Q 与蓝图结合使用的更多信息，请参阅使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践。

使用 Amazon Q 创建项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要创建蓝图的空间。
3. 在空间控制面板上，选择使用 Amazon Q 创建。
4. 在 Amazon Q 提示文本输入字段中，通过撰写有关您要构建的项目的简短描述来提供说明。例如，“I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data.”

   （可选）在试用示例下，您可以通过选择蓝图来使用预先撰写的提示。例如，如果您选择 React 应用程序，系统会提供以下提示：“I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data. I also want to add authentication and authorization mechanisms for security and allowable actions.”

5. 选择发送将您的说明提交给 Amazon Q。生成式人工智能助手会提供建议，并概述蓝图无法满足的要求。例如，Amazon Q 可能会根据您的标准提出以下建议：

```
I recommend using the Modern three-tier web application blueprint based on
                    your requirements. Blueprints are dynamic and can always be
 updated and edited later.

Modern three-tier web application
By Amazon Web Services

This blueprint creates a Mythical Mysfits 3-tier web application with a modular
 presentation, application, and data layers.
The application leverages containers, infrastructure as code (IaC), continuous
 integration and continuous delivery (CI/CD),
and serverless code functions.

Version: 0.1.163

View details

The following requirements could not be met so I will create issues for you.
• Add authentication and authorization mechanisms for security and allowable
 actions.
```

6. （可选）要查看建议蓝图的详细信息，请选择查看详细信息。
7. 请执行以下操作之一：
   a. 如果您对建议感到满意，请选择是，使用此蓝图。
   b. 如果要修改提示，请选择编辑提示。
   c. 如果要完全清除提示，请选择重新开始。
8. 请执行以下操作之一：
   a. 如果要配置建议的蓝图，请选择配置。您还可以稍后配置蓝图。
   b. 如果您目前不想修改蓝图配置，请选择跳过。
9. 如果您选择配置蓝图，请在修改项目资源后选择继续。
10. 在系统提示时，输入要分配给项目的名称及其关联资源名称。该名称在空间内必须是唯一的。
11. 选择创建项目以使用蓝图创建项目。Amazon Q 使用蓝图创建资源。例如，如果您使用单页应用程序蓝图创建项目，则会为其创建相关代码和工作流程的 CI/CD 源存储库。

12. （可选）默认情况下，Amazon Q 还会为蓝图无法满足的要求创建事务。您可以选择不想为哪些项目创建事务。在您选择让 Amazon Q 创建事务后，您也可以将事务分配给 Amazon Q。Amazon Q 将在给定源存储库的上下文中分析事务，提供相关源文件和代码的摘要。有关更多信息，请参阅查找和查看事务、创建事务并将其分配给 Amazon Q 和 创建和处理分配给 Amazon Q 的事务时的最佳实践。

使用 Amazon Q 创建项目后，您还可以使用 Amazon Q 添加新组件，因为它会根据您的要求建议 CodeCatalyst 蓝图。

使用 Amazon Q 添加蓝图

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要在其中添加蓝图的项目。
3. 选择使用 Amazon Q 添加。
4. 在 Amazon Q 提示文本输入字段中，通过撰写有关您要构建的项目的简短描述来提供说明。例如，“I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data.”

   （可选）在试用示例下，您可以通过选择蓝图来使用预先撰写的提示。例如，如果您选择 React 应用程序，系统会提供以下提示：“I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data. I also want to add authentication and authorization mechanisms for security and allowable actions.”
5. 选择发送将您的说明提交给 Amazon Q。生成式人工智能助手会提供建议，并概述蓝图无法满足的要求。例如，Amazon Q 可能会根据您的标准提出以下建议：

```
I recommend using the Single-page application blueprint based on your requirements.
 Blueprints are dynamic and can always be updated and edited later.

Single-page application
By Amazon Web Services

This blueprint creates a SPA (single-page application) using React, Vue, or Angular
 frameworks and deploys to AWS Amplify Hosting.
```

```
Version: 0.2.15
View details

The following requirements could not be met so I will create issues for you.
• The application should have reusable UI components
• The application should support for client-side routing
• The application may require server-side rendering for improved performance and
 SEO
```

6. （可选）要查看建议蓝图的详细信息，请选择查看详细信息。
7. 请执行以下操作之一：
   a. 如果您对建议感到满意，请选择是，使用此蓝图。
   b. 如果要修改提示，请选择编辑提示。
   c. 如果要完全清除提示，请选择重新开始。
8. 请执行以下操作之一：
   a. 如果要配置建议的蓝图，请选择配置。您还可以稍后配置蓝图。
   b. 如果您目前不想修改蓝图配置，请选择跳过。
9. 如果您选择配置蓝图，请在修改项目资源后选择继续。
10. 选择添加到项目，以便使用蓝图向项目添加资源。Amazon Q 使用蓝图创建资源。例如，如果您使用单页应用程序蓝图添加到项目中，则会为其创建相关代码和工作流程的 CI/CD 源存储库。
11. （可选）默认情况下，Amazon Q 还会为蓝图无法满足的要求创建事务。您可以选择不想为哪些项目创建事务。在您选择让 Amazon Q 创建事务后，您也可以将事务分配给 Amazon Q。Amazon Q 将在给定源存储库的上下文中分析事务，提供相关源文件和代码的摘要。有关更多信息，请参阅创建事务并将其分配给 Amazon Q和创建和处理分配给 Amazon Q 的事务时的最佳实践。

在创建拉取请求时创建分支之间的代码更改摘要

拉取请求是您和其他项目成员用于审查、注释和合并分支间的代码更改的主要方式。您可以使用拉取请求来协同审查代码更改，包括次要的更改或修复、主要功能添加或已发布软件的新版本。作为拉取请求描述的一部分，总结代码更改和更改背后的意图对其他要查看代码的人很有帮助，也有助于了解代码随时间推移所发生变化的历史记录。但是，开发人员通常依靠自己的代码，来解释代码自身或者提供模棱两可的细节，而不是用足够的细节来描述他们的更改，让审阅者了解他们正在审查的内容或代码更改背后的意图。

在创建拉取请求时，您可以使用为我编写描述功能，让 Amazon Q 为拉取请求中包含的更改创建描述。当您选择此选项时，Amazon Q 会分析包含代码更改的源分支与要合并这些更改的目标分支之间的差异。然后，它会总结了这些更改的内容，以及对这些更改的意图和效果的最佳解释。

> **Note**
>
> 此功能不适用于 Git 子模块。作为拉取请求一部分的 Git 子模块中的任何更改不会进行汇总。此功能不适用于链接存储库中的拉取请求。

您可以用您创建的任何拉取请求试用这个功能，但在本教程中，我们将通过对在基于 Python 的现代三层 Web 应用程序蓝图中创建的项目所包含的代码进行一些简单的更改，对此功能进行测试。

> **Tip**
>
> 如果您使用的项目是用不同蓝图或自己代码创建的，您仍然可以按照本教程进行操作，但是本教程中的示例与您项目中的代码会不匹配。与其使用下面建议的示例，不如在一个分支中对项目代码进行简单更改，然后创建一个拉取请求来测试该功能，如以下步骤所示。

首先，需要在源存储库中创建一个分支。然后，使用控制台中的文本编辑器对该分支中的一个文件进行简单的代码更改。然后，创建一个拉取请求，并使用为我编写描述功能来总结您所做的更改。

创建分支（控制台）

1. 在 CodeCatalyst 控制台中，导航到源存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。
3. 选择要在其中创建分支的存储库。
4. 在存储库的概述页面上，选择更多，然后选择创建分支。
5. 输入分支的名称。
6. 选择要从中创建分支的分支，然后选择创建。

有了分支后，只需对该分支中的文件进行简单的更改。在本例中，您将编辑 `test_endpoint.py` 文件，将测试的重试次数从 **3** 更改为 5。

> **Tip**
>
> 您还可以选择创建或使用一个开发环境来进行这次的代码更改。有关更多信息，请参阅 创建开发环境。

在控制台中编辑 **test_endpoint.py** 文件

1. 在 **mysfits** 源存储库的概述页面上，选择分支下拉列表，并选择您在上一个过程中创建的分支。
2. 在文件中，导航到要编辑的文件。例如，要编辑 test_endpoint.py 文件，请展开测试，展开 integ，然后选择 test_endpoint.py。
3. 选择编辑。
4. 在第 7 行，将重试所有测试的次数从：

   ```
   def test_list_all(retry=3):
   ```

   更改为：

   ```
   def test_list_all(retry=5):
   ```
5. 选择提交，然后将更改提交到您的分支。

现在，您已经有一个分支含有更改，可以创建拉取请求。

创建包含更改摘要的拉取请求

1. 在存储库的概述页面上，选择更多，然后选择创建拉取请求。
2. 在目标分支中，在审查代码后，选择要将代码合并到的分支。

   > **Tip**
   >
   > 选择您在上一个过程中用来创建您的分支的分支，以最方便地演示此功能。例如，如果您利用存储库的默认分支创建了分支，请选择该分支作为拉取请求的目标分支。
3. 在源分支中，选择包含您刚刚提交给 test_endpoint.py 文件的更改的分支。
4. 在拉取请求标题中，输入一个标题，帮助其他用户了解需要审查的内容及其原因。
5. 在拉取请求描述中，选择为我编写描述，让 Amazon Q 为拉取请求中包含的更改创建描述。

6. 此时将显示更改摘要。查看建议的文本，然后选择接受并添加到描述。
7. （可选）修改摘要以更好地反映您对代码所做的更改。您还可以选择添加审阅者，或者将事务链接到此拉取请求。完成所需的其他任何更改后，选择创建。

创建拉取请求中对代码更改所留下注释的摘要

当用户查看拉取请求时，他们通常会对该拉取请求中的更改留下多条注释。如果有很多审阅者提供了很多注释，则可能很难在反馈中挑出共同的主题，甚至很难确保您已经审核了所有修订中的所有注释。您可以使用创建注释摘要功能，让 Amazon Q 分析拉取请求中更改代码时留下的所有注释，并创建这些注释的摘要。

> **Note**
>
> 注释摘要是临时的。如果您刷新拉取请求，摘要将消失。内容摘要不包括对整个拉取请求的注释，只包括对拉取请求的修订中代码差异留下的注释。
> 此功能不适用于在 Git 子模块中对代码更改留下的任何注释。
> 此功能不适用于链接存储库中的拉取请求。

创建拉取请求中的注释的摘要

1. 导航到您在前一个过程中创建的拉取请求。

   > **Tip**
   >
   > 如果您愿意，可以在项目中使用任何已打开的拉取请求。在导航窗格中，依次选择代码、拉取请求和任何打开的拉取请求。

2. 如果拉取请求还没有注释，请在更改中向拉取请求添加一些注释。
3. 在概述中，选择创建注释摘要。完成后，注释摘要部分将展开。
4. 查看拉取请求修订中对代码更改留下的注释摘要，并将其与拉取请求中的注释进行比较。

创建事务并将其分配给 Amazon Q

开发团队会创建问题来跟踪和管理他们的工作，但有时问题会持续存在，因为要么不清楚谁应该处理这个问题，要么问题需要对代码库的特定部分进行研究，要么必须先处理其他紧急工作。 CodeCatalyst 包括与 Amazon Q 开发者代理集成，用于软件开发。您可以将事务分配给名为 Amazon Q 的生成式人

工智能助手，该助手可以根据事务的标题和描述来分析事务。如果您将事务分配给 Amazon Q，它将尝试创建解决方案草稿以供您评估。这有助于您和您的团队专注于并优化处理需要您关注的事务，而 Amazon Q 则专注于解决您没有资源来立即解决的问题。

> **Tip**
>
> Amazon Q 能够出色地处理简单的事务和问题。为了获得最佳结果，请使用简明易懂的语言来清楚地说明您要执行的操作。

当您将问题分配给 A mazon Q 时， CodeCatalyst 会将该问题标记为已阻止，直到您确认希望 Amazon Q 如何处理该问题。它需要您回答三个问题才能继续：

- 无论您是想确认它所采取的每个步骤，还是希望它在没有反馈的情况下继续进行。如果您选择确认每个步骤，则可以回复 Amazon Q，并提供有关其创建方法的反馈，以便它可以在需要时对其方法进行迭代。如果您选择此选项，Amazon Q 还可以查看用户在该软件创建的任何拉取请求中留下的反馈。如果您选择不确认每个步骤，Amazon Q 可能会更快地完成工作，但它不会审查您在事务中或它创建的任何拉取请求中给出的任何反馈。
- 您是否要让它能够更新工作流文件作为其工作的一部分。您的项目可能已将工作流配置为在拉取请求事件上开始运行。如果是这样的话，Amazon Q 创建的任何拉取请求（包括创建或更新工作流 YAML）都可能启动拉取请求中包含的那些工作流的运行。作为最佳实践，请不要选择让 Amazon Q 能够使用工作流文件，除非您确定项目中没有将自动运行这些工作流文件的工作流，然后再审核和批准它创建的拉取请求。
- 您是否让它能够建议创建任务，将事务中的工作分解为可以单独分配给用户（包括 Amazon Q 本身）的较小增量。让 Amazon Q 能够建议和创建任务可以让多人处理事务的不同部分，从而有助于加快复杂事务的处理。它还可以帮助降低整个工作的理解复杂性，因为理想情况下，完成每项任务所需的工作比它所属的事务更简单。
- 您想让它在哪个源存储库中工作。即使您的项目有多个源存储库，Amazon Q 也只能处理一个源存储库中的代码。不支持链接的存储库。

在您做出并确认选择后，Amazon Q 会将事务移至进行中状态，同时它会尝试根据事务标题及其描述以及指定存储库中的代码，来确定请求的内容。它将创建一个固定注释，在其中提供有关其工作状态的最新信息。在审查数据后，Amazon Q 将制定一种潜在的解决方案方法。Amazon Q 通过更新其固定注释，并在每个阶段注释该事务的进展，来记录自己的操作。与固定的注释和回复不同，它没有严格按时间顺序记录其工作。相反，它将有关其工作的最相关信息放在固定注释的顶部。它将尝试根据其方法和对存储库中已有代码的分析来创建代码。如果它成功生成了潜在的解决方案，它将创建一个分支，并将

代码提交给该分支。然后，它会创建一个拉取请求，用于将该分支与默认分支合并。当 Amazon Q 完成其工作后，它会将事务移至审核中，以便您和您的团队知道有可供您评估的代码。

> **Note**
>
> 此功能仅通过美国西部（俄勒冈州）区域中的事务提供。如果您已将项目配置为使用带有 Jira Software 扩展的 Jira，则该功能不可用。此外，如果您自定义了面板的布局，则事务可能不会改变状态。为获得最佳效果，请仅在采用标准面板布局的项目中使用此功能。
> 此功能不适用于 Git 子模块。它无法对存储库中包含的任何 Git 子模块进行更改。
> 您将事务分配给 Amazon Q 后，就无法更改事务的标题或描述，也不能将其分配给其他任何人。如果您从事务中取消分配 Amazon Q，它将完成当前步骤，然后停止工作。一旦取消分配它，它就无法恢复工作或重新分配给事务。
> 如果用户选择让 Amazon Q 能够创建任务，则如果将事务分配给 Amazon Q，则该事务可以自动移至审核中列。但是，审阅中状态的事务可能仍有处于不同状态的任务，例如处于进行中状态。

在本教程的这一部分中，您将根据使用现代三层 Web 应用程序蓝图所创建项目中包含的代码的潜在功能，创建三个事务：一个用于创建新的 mysfit 生物，一个用于添加排序功能，另一个用于更新工作流以包含名为 **`test`** 的分支。

> **Note**
>
> 如果您在使用的项目含有不同代码，请使用与该代码库相关的标题和描述来创建事务。

创建事务并生成解决方案供您评估

1. 在导航窗格中，选择事务，并确保您处于面板视图。
2. 选择创建事务。
3. 给事务起一个标题，用通俗易懂的语言解释您想做什么。例如，对于事务，输入标题 **`Create another mysfit named Quokkapus`**。对于描述，请提供以下详细信息：

```
Expand the table of mysfits to 13, and give the new mysfit the following
 characteristics:

Name: Quokkapus
```

```
Species: Quokka-Octopus hybrid

Good/Evil: Good

Lawful/Chaotic: Chaotic

Age: 216

Description: Australia is full of amazing marsupials, but there's nothing there
 quite like the Quokkapus.
She's always got a friendly smile on her face, especially when she's using her
 eight limbs to wrap you up
in a great big hug. She exists on a diet of code bugs and caffeine. If you've got
 some gnarly code that needsa
assistance, adopt Quokkapus and put her to work - she'll love it! Just make sure
 you leave enough room for
her to grow, and keep that coffee coming.
```

4. （可选）在事务上附加一张图像，用作 mysfit 缩略图和个人资料图片。如果您这样做，请更新描述，以包含您要使用哪些图像及其原因的详细信息。例如，您可以在描述中添加以下内容：“mysfit 要求将图像文件部署到网站。作为工作的一部分，将本事务所附的这些图像添加到源存储库中，然后将这些图像部署到网站。”

> **Note**
>
> 在本教程的互动过程中，附加的图像可能会部署到网站，也可能不会部署到网站。您可以自己将图像添加到网站，然后在创建拉取请求后留下注释，让 Amazon Q 更新其代码以指向您希望它使用的图片。

在继续执行下一步之前，请查看描述，并确保其中包含可能需要的所有详细信息。

5. 在被分派人中，选择分配给 Amazon Q。
6. 对于源存储库，请选择包含项目代码的源存储库。
7. 如有必要，请将要求 Amazon Q 在每个步骤后停止并等待审核其工作选择器滑动到活动状态。

> **Note**
>
> 选择在每个步骤后让 Amazon Q 停止的选项后，您就可以对事务或任何已创建的任务进行注释，从而可以选择让 Amazon Q 根据您的注释最多三次更改其方法。如果您选择不让 Amazon Q 在每个步骤后停止的选项，以便您可以查看其工作，则工作可能会更快地进

行，因为 Amazon Q 不等待您的反馈，但您将无法通过进行注释来影响 Amazon Q 执行的方向。如果您选择该选项，Amazon Q 也不会回复拉取请求中留下的注释。

8. 使允许 Amazon Q 修改工作流文件选择器保持非活动状态。
9. 将允许 Amazon Q 建议创建任务选择器滑动到活动状态。
10. 选择创建事务。您的视图将更改为“事务”面板。
11. 选择创建事务以创建另一个事务，这次是具有以下标题的事务：**`Change the get_all_mysfits() API to return mysfits sorted by the Age attribute`**。将此事务分配给 Amazon Q 并创建事务。
12. 选择创建事务以创建另一个事务，这次是具有以下标题的事务：**`Update the OnPullRequest workflow to include a branch named test in its triggers`**。可以选择链接到描述中的工作流。将此事务分配给 Amazon Q，但这次请确保将允许 Amazon Q 修改工作流文件选择器设置为活动状态。创建事务以返回“事务”面板。

> **Tip**
>
> 您可以通过输入 @ 符号和文件名来搜索文件，包括工作流文件。

创建并分配事务后，事务将移至处理中。Amazon Q 将添加注释，用于跟踪固定注释中的事务的进展。如果它能够定义解决方案的方法，它将使用包含其对代码库的分析的背景部分，以及详细介绍其创建解决方案的建议方法的方法部分，来更新事务的描述。对于事务中描述的问题，如果 Amazon Q 能够成功提出解决方案，它将创建一个分支，并在该分支中更改代码，实现其建议的解决方案。如果建议的代码与 Amazon Q 知道的开源代码有相似之处，该软件将提供一个包含该代码的链接的文件，以便您可以进行查看。代码一旦准备就绪，它就会创建一个拉取请求，以便您可以查看建议的代码更改，添加指向该事务的拉取请求的链接，然后将事务移至审核中状态中。

> **Important**
>
> 在合并拉取请求之前，您应始终查看其中的任何代码更改。与其他任何代码更改一样，如果合并后的代码未经过适当审查并且在合并时包含错误，则合并 Amazon Q 所做的代码更改时，可能会对您的代码库和基础设施代码产生负面影响。

查看事务和包含 Amazon Q 所做更改的关联拉取请求

1. 在事务中，选择分配给 Amazon Q 且处于进行中的事务。查看注释以监控 Amazon Q 的进展情况。如果有注释，请查看背景以及该软件在事务描述中记录的方法。如果您选择让 Amazon Q 建议任务，请查看所有建议的任务并采取任何必要的操作。例如，如果 Amazon Q 建议了任务，而您想更改任务顺序或者将任务分配给特定用户，请选择更改、添加或重新排序任务，并执行任何必要的更新。查看完事务后，选择 X 关闭事务窗格。

   > **Tip**
   >
   > 要查看任务的进度，请从事务的任务列表中选择任务。任务不会作为单独的项目显示在面板上，只能通过事务进行访问。如果任务分配给 Amazon Q，则您必须打开该任务才能批准其要执行的任何操作。您还必须打开任务，才能看到任何关联拉取请求，因为它们不会作为链接出现在事务中，而只出现在任务中。要从任务中返回事务，请选择指向该事务的链接。

2. 现在，选择一个分配给 Amazon Q 且处于审核中状态的事务。查看事务描述中记录的背景和方法。查看注释以了解该软件执行的操作。查看为与此事务相关的工作创建的所有任务，包括任务进度、您可能需要采取的任何操作以及任何注释。在拉取请求中，选择打开标签旁边的拉取请求链接以查看代码。

   > **Tip**
   >
   > 为任务生成的拉取请求仅在任务视图中以关联拉取请求形式出现，不会以事务的关联拉取请求形式出现。

3. 在拉取请求中，查看代码更改。有关更多信息，请参阅 审核拉取请求。如果您希望 Amazon Q 更改其建议的任何代码，请在拉取请求上留下注释。为了获得最佳效果，在 Amazon Q 上添加注释时，请务必具体说明。

   例如，在查看为 **`Create another mysfit named Quokkapus`** 创建的拉取请求时，您可能会注意到描述中有一个拼写错误。您可以在 Amazon Q 上添加一条注释，注明“通过在‘needs’和‘a’之间添加一个空格来更改描述，以修复拼写错误‘needsa’”。或者，您也可以添加一条注释，指示 Amazon Q 更新描述，并提供修改后的完整描述以便其采纳。

   如果您将新 mysfit 的图像上传到网站，您可以添加一条注释，让 Amazon Q 更新 mysfit，在其中添加用于新 mysfit 的图像和缩略图指针。

> **Note**
>
> Amazon Q 不会回复个人注释。仅当您在创建事务时选择了在每个步骤后停止的默认选项，Amazon Q 才会将注释中留下的反馈纳入拉取请求中。

4. （可选）在您和其他项目用户留下您希望更改代码的所有注释后，选择创建修订，让 Amazon Q 创建拉取请求的修订，其中包含您在注释中请求的更改。Amazon Q 将在概述中报告修订创建进度，而不是在更改中报告。请务必刷新浏览器，以查看 Amazon Q 关于创建修订的最新更新。

> **Note**
>
> 只有创建了该事务的用户才能创建拉取请求的修订。对于一个拉取请求，您只能请求创建一个修订。在选择创建修订之前，请确保您已经解决了注释中的所有问题，并且对注释的内容感到满意。

5. 在此示例项目中，将为每个拉取请求运行一个工作流。在合并拉取请求之前，请确保看到工作流已成功运行。在合并之前，您也可以选择创建其他工作流和环境来测试代码。有关更多信息，请参阅入门工作流。
6. 如果您对拉取请求的最新修订感到满意，请选择合并。

创建事务并让 Amazon Q 为其推荐任务

问题有时可能包含复杂或漫长的工作量。 CodeCatalyst 包括与 Amazon Q 开发者代理集成，用于软件开发。您可以让 Amazon Q 根据事务的标题和描述来分析事务，并建议将工作按逻辑分解为不同的任务。该软件将尝试创建建议任务的列表，然后可以对这些任务进行审查和修改，以及选择是否创建。这可以帮助您和您的团队以更易于管理的方式将工作的各个部分分配给用户，从而可以更快地完成工作。

创建和查看事务的建议任务列表

1. 在导航窗格中，选择事务，并确保您处于面板视图。
2. 选择创建事务。
3. 给事务起一个标题，用通俗易懂的语言解释您想做什么。例如，对于事务，输入标题 **`Change the get_all_mysfits() API to return mysfits sorted by the Good/Evil attribute`**。对于描述，请提供以下详细信息：

```
Update the API to allow sorting of mysfits by whether they are Good, Neutral, or
 Evil. Add a button on the website that allows users to quickly choose this sort
 and to exclude alignments that they don't want to see.
```

4. 在继续执行下一步之前，请查看描述，并确保其中包含可能需要的所有详细信息。
5. 在被分派人中，选择将事务分配给自己。
6. 选择创建事务。您的视图将更改为“事务”面板。
7. 选择您刚才创建的事务将其打开。选择建议任务。
8. 选择包含此事务的代码的源存储库。选择开始建议任务。

对话框将关闭，Amazon Q 将开始分析事务的复杂性。如果事务很复杂，则会建议将工作分解为单独且连续的任务。列表准备就绪后，选择查看建议的任务。您可以添加其他任务、修改建议的任务以及对任务重新排序。如果您同意这些建议，则选择创建任务可以创建任务。然后，您可以将这些任务分配给用户进行处理，甚至可以分配给 Amazon Q 本身。

清理 资源

完成本教程后，可以考虑采取以下操作来清理在本教程中创建的不再需要的所有资源。

- 从不再处理的任何事务取消分配 Amazon Q。如果 Amazon Q 已完成对某个事务的处理或找不到解决方案，请务必取消分配 Amazon Q，以免达到生成式人工智能功能的最大配额。有关更多信息，请参阅 Managing generative AI features 和 Pricing。
- 将所有已处理完的事务移至完成。
- 如果不再需要该项目，请删除它。

使用空格整理资源 CodeCatalyst

您可以创建一个代表您、您的公司、部门或小组的空间，为您的开发团队提供一个管理项目的场所。您必须创建一个空间来添加您在 Amazon 中创建的项目、成员和关联的云资源 CodeCatalyst。

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。不能重复使用已删除空间的名称。

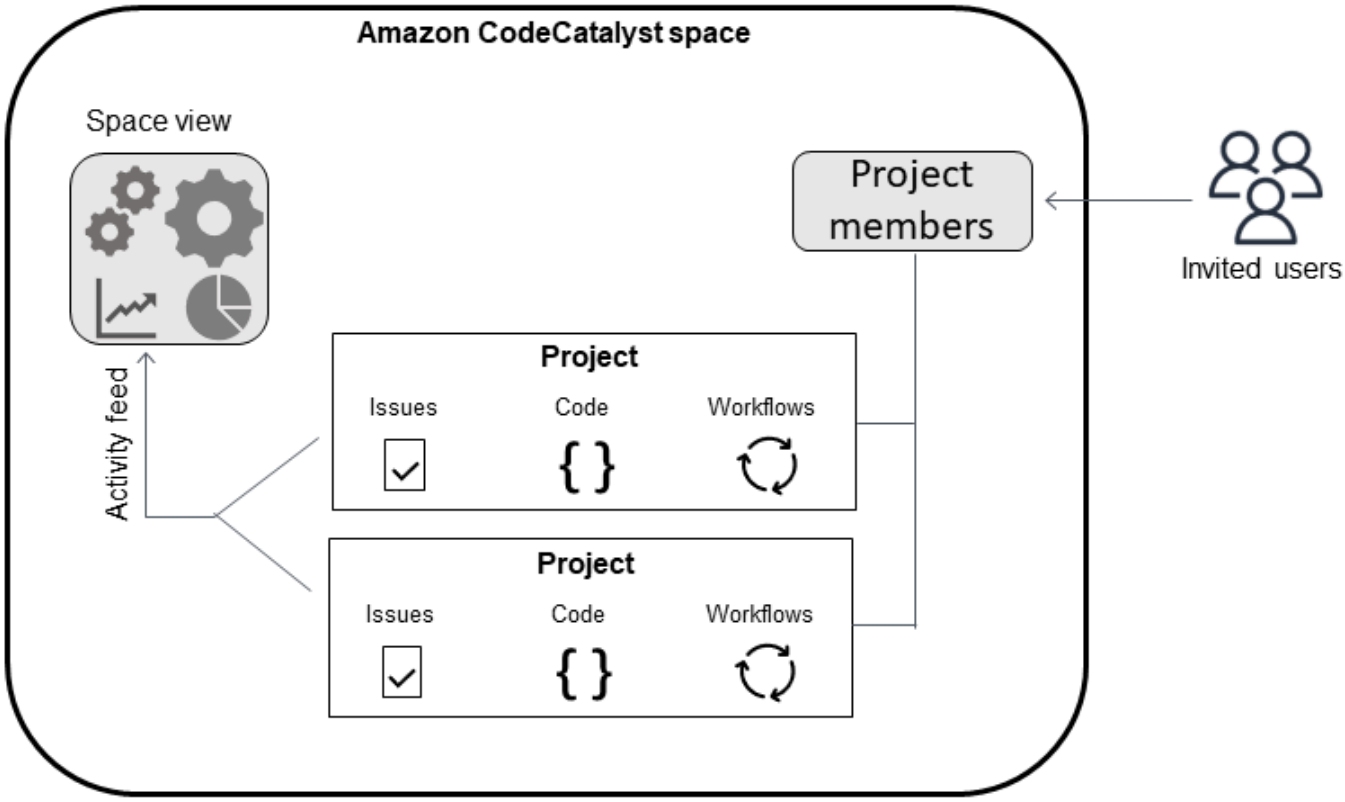
当您创建空间时，系统会自动为您分配空间管理员角色。您可以为空间中的其他用户添加此角色。

使用空间管理员角色，可以对空间进行如下管理：

- 向空间中添加其他空间管理员
- 更改成员角色和权限
- 编辑或删除空间
- 创建项目并邀请成员加入项目
- 查看空间中所有项目的列表
- 查看空间中所有项目的活动源

当您创建空间时，您会被自动添加到空间中，并拥有两个角色：空间管理员角色，和您在创建空间时创建的项目的项目管理员角色。当其他用户接受项目邀请时，会自动将其添加为空间成员。空间中的这种成员资格不授予任何空间权限。用户在空间中能做什么，取决于用户在特定项目中的角色。

有关角色的更多信息，请参阅 使用用户角色授予访问权限。



以下是新增账户的其他注意事项：

- AWS 账户 添加到 CodeCatalyst 空间可以在该空间中的任何项目中使用。
- 虽然每个环境可以支持多个环境 AWS 账户，但在一个操作中，每个环境只能使用一个账户。
- 计费是在空间级别配置的。可以配置多个账户进行计费，但一个 CodeCatalyst 空间中只能有一个账户处于活动状态。 AWS 账户 可以用作中多个空间的结算帐号 CodeCatalyst。指定 AWS 账户 为空间结算账号的配额与 CodeCatalyst 空间的其他账户连接的配额不同。有关更多信息，请参阅 CodeCatalyst 的配额。
- 创建连接后，如果您的工作流程必须通过您的 CodeCatalyst 环境访问这些 AWS IAM 角色，则必须向连接中添加 IAM 角色。有关如何使用环境的更多信息，请参阅部署到 AWS 账户 和 VPCs。

主题

- 创建空间
- 编辑空间
- 删除空间
- 监控空间中用户和资源的活动

- 允许在已连接的情况下访问 AWS 资源 AWS 账户
- 为关联账户配置 IAM 角色
- 向用户授予空间权限
- 允许使用团队访问空间
- 允许机器资源的空间访问权限
- 管理空间的开发环境
- 空间配额

创建空间

首次使用 AWS 建筑商 ID 在 CodeCatalyst Amazon 上注册时，您需要创建一个空间。有关更多信息，请参阅 设置并登录 CodeCatalyst。您可以选择创建其他空间来满足您的业务需求。

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。不能重复使用已删除空间的名称。

本指南中的信息用于在中 CodeCatalyst 创建支持 AWS Builder ID 用户的空间。《CodeCatalyst 管理员指南》中提供了设置和管理支持身份联合的空间的步骤。要使用为身份联合设置的空间，请参阅《Amazon CodeCatalyst 管理员指南》中的 CodeCatalyst 空间设置和管理。

要创建支持 AWS Builder ID 用户的其他空间，必须为您分配空间管理员角色。

> **Note**
>
> 创建其他空间时，系统不会提示您创建项目。要了解如何在空间中创建项目，请参阅创建项目。

创建另一个空间

1. 在中 AWS 管理控制台，请确保您使用要与 CodeCatalyst 空间关联的相同 AWS 账户 用户登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

4. 选择创建空间。
5. 在创建空间页面的空间名称中，输入空间的名称。您以后不能更改此名称。

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。不能重复使用已删除空间的名称。

6. 在 AWS 区域中，选择要在其中存储空间和项目数据的区域。您以后不能更改此区域。
7. 在 AWS 账户 ID 中，输入您要连接到空间的账户的十二位 ID。

   在 AWS 账户验证令牌中，复制生成的令牌 ID。系统会自动为您复制令牌，但您可能需要在批准 AWS 连接请求时将其存储。
8. 选择 “验证” AWS。
9. “验证 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。

   在中 AWS 管理控制台，请务必选择您想要创建空间的相同 AWS 区域 位置。

   要直接访问该页面，请登录 a AWS 管理控制台 t https://console.aws.amazon.com/codecatalyst/home/ 中的 Amazon CodeCatalyst Spaces。

   这将自动在验证令牌中输入验证令牌。成功横幅显示一条消息，表明该令牌是有效令牌。
10. 选择验证空间。

    这将显示账户已验证成功消息，表示已将该账户添加到空间。
11. 留在 “验证 Amazon CodeCatalyst 空间” 页面上。选择以下链接：要为该空间添加 IAM 角色，请查看空间详细信息。

    CodeCatalyst 空间详情页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。
12. 在 “可用的 IAM 角色” 下 CodeCatalyst，选择添加 IAM 角色。

    将显示 “添加可用的 IAM 角色 CodeCatalyst” 页面。

13. 选择在 IAM 中创建 CodeCatalyst 开发管理员角色。此选项将创建一个服务角色，其中包含开发角色的权限策略和信任策略。

    开发人员角色是一 AWS 个 IAM 角色，可让您的 CodeCatalyst 工作流程访问诸如 Amazon S3、Lambda 和之类的 AWS 资源。 CloudFormation该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-spaceName`，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。
14. 选择创建开发角色。
15. 在连接页面的可用的 IAM 角色下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表中的开发者角色。
16. 选择 “前往亚马逊” CodeCatalyst。
17. 在的创建页面上 CodeCatalyst，选择创建空间。

编辑空间

您可以更改空间的描述，以便于用户更好地了解空间的用途。

您必须拥有空间管理员角色才能编辑空间详细信息。

本指南中提供的信息用于编辑支持 AWS 生成器 ID 用户的空间。 CodeCatalyst 要详细了解设置和管理支持身份联合的空间的步骤，请参阅《Amazon CodeCatalyst 管理员指南》中的 CodeCatalyst 空间设置和管理。

编辑空间描述

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

   > **Tip**
   >
   > 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 在空间设置选项卡上，选择编辑。对空间描述进行所需的更改，然后选择保存。

删除空间

您可以删除空间以取消对该空间所有资源的访问权限。您必须拥有空间管理员角色才能删除空间。

> **Note**
>
> 无法撤销空间删除操作。

删除空间后，所有空间成员都将无法访问空间资源。空间资源的计费也将停止，第三方源存储库提示的任何工作流都将停止。

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。不能重复使用已删除空间的名称。

本指南中的信息用于删除支持 AWS 生成器 ID 用户的空间。 CodeCatalyst 要详细了解设置和管理支持身份联合的空间的步骤，请参阅《Amazon CodeCatalyst 管理员指南》中的 CodeCatalyst 空间设置和管理。

删除空间

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

   > **Tip**
   >
   > 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择删除。
4. 键入 **`delete`** 以确认删除。
5. 选择删除。

   > **Note**
   >
   > 如果您属于多个空间，您将被重定向到空间概述页面。如果您属于一个空间，您将被重定向到空间创建页面。

监控空间中用户和资源的活动

要查看最近创建的项目和状态更新，您可以使用 CodeCatalyst 控制台查看显示空间资源更新的活动提要。

在活动源中，您可以查看工作流运行失败和创建的项目等指标。

查看您空间中的活动

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择活动。
4. 在活动中查看信息。
5. 要按活动筛选，请选择右上角的选择器。
6. 要查看空间中的所有活动，请选择任何活动类型。

允许在已连接的情况下访问 AWS 资源 AWS 账户

您可以在 Amazon CodeCatalyst 空间 AWS 账户 中使用您的资源。为此，您必须在 AWS 账户 和您的空间之间建立连接 CodeCatalyst。创建这样的连接意味着您空间中的项目和工作流程可以与您的 CodeCatalyst 空间中的资源进行交互 AWS 账户。您必须为要在 CodeCatalyst 空间中使用的每个 AWS 账户 连接创建一个连接。

创建连接后，您可以选择将 AWS IAM 角色与其关联。

主题

- 向 AWS 账户 空间添加
- 将 IAM 角色添加到账户连接
- 将账户连接和 IAM 角色添加到部署环境
- 查看账户连接
- 删除账户连接（中 CodeCatalyst）

- 为空间配置计费账户

您可以通过将账户添加到您的空间 AWS 账户 来设置 CodeCatalyst 为使用已授权。通过 AWS 账户 添加到您的 CodeCatalyst 空间，您可以为项目工作流程提供对 AWS 账户 资源和账单配置的访问权限。

添加 AWS 账户 会创建授权使用此账户 CodeCatalyst 的连接。你可以使用 adde AWS 账户 d 来执行以下操作：

- 为 CodeCatalyst 空间设置账单。请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。指定 AWS 账户 为空间结算账号的配额与 CodeCatalyst 空间的其他账户连接的配额不同。有关更多信息，请参阅 CodeCatalyst 的配额。
- CodeCatalyst 允许担任 IAM 角色以访问和部署到账户 AWS 服务 中的 AWS 资源。请参阅为关联账户配置 IAM 角色。

通过使用 AWS 账户完成授权来创建账户连接。创建连接后，可以通过添加 IAM 角色来进一步配置要使用的工作流和项目的连接。

有关在 AWS 管理控制台 页面中以管理员 CodeCatalyst 身份为 AWS 账户 和空间配置账户连接的步骤，请参阅《CodeCatalyst 管理员指南》中的 “管理已连接的帐户”。可以针对特定项目的限制来配置账户连接。您只能将工作流或 VPC 连接与有权访问您的项目的人关联。 AWS 账户 有关更多信息，请参阅 Configuring project-restricted account connections。

向 AWS 账户 空间添加

您可以使用 CodeCatalyst 控制台和 AWS 管理控制台 将您的空间连接到 AWS 账户。

在向 AWS 账户 中的空间添加之前 CodeCatalyst，请完成以下先决条件：

- 创建 AWS 账户 并获得在您要连接的账户中创建 AWS IAM 角色的权限。
- 创建要与账户连接关联的一个或多个 IAM 角色，包括具有角色的权限的 IAM 策略。
- 在要创建连接的 CodeCatalyst空间中获取空间管理员角色。

主题

- 后续步骤：为账户连接创建其他 IAM 角色

步骤 1：创建连接请求

在 CodeCatalyst 控制台中创建连接请求会生成一个连接令牌，您可以使用该令牌完成授权。

在要创建连接的空间中，您必须具有 CodeCatalyst 空间管理员或超级用户角色。您还必须拥有要添加的 AWS 账户 的管理权限。

创建连接

1. 在中 AWS 管理控制台，请确保您使用要与之建立连接的相同帐户登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
4. 选择 “添加” AWS 账户。
5. 在 “ AWS 账户 与 Amazon 关联 CodeCatalyst” 页面的 AWS 账户 ID 中，输入您要关联至空间的账户的 12 位数 ID。有关查找您的 AWS 账户 ID 的信息，请参阅您的 AWS 账户 ID 及其别名。
6. 在 Amazon CodeCatalyst 显示名称中，输入账户的参考名称。
7. （可选）在连接描述中，输入账户的描述，这将帮助您选择要应用该账户和一个或多个角色的项目。
8. 选择关联 AWS 账户。
9. 该页面将返回到 AWS 账户 详细信息页面，其中显示了成功横幅。

步骤 2：接受账户连接请求

在 CodeCatalyst 控制台中提交连接请求后 AWS 账户，您可以与 AWS 管理员合作，通过使用提供的连接令牌提交连接请求来接受该请求。

请确保您的账户拥有管理员权限，并且 AWS 管理控制台 使用与创建连接时相同的 AWS 账户 权限登录。

批准连接请求（控制台）

1. 在中 AWS 管理控制台，请确保您使用要与之建立连接的相同帐户登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。

4. 在 AWS 账户 详细信息页面上，选择完成 AWS 管理控制台中的设置。
5. “验证 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。

   要直接访问该页面，请登录 a AWS 管理控制台 t https://console.aws.amazon.com/codecatalyst/home/ 中的 Amazon CodeCatalyst Spaces。

   这将自动在验证令牌中输入验证令牌。成功消息显示一条消息，表明该令牌是有效令牌。
6. （可选）在已授权的付费套餐下，选择授权付费套餐 (标准版、企业版)，为您的计费账户启用付费套餐。

   > **Note**
   >
   > 这不会将计费套餐升级到付费套餐。但是，这会对进行配置， AWS 账户 以便您可以随时更改空间的计费等级。 CodeCatalyst您可以随时启用付费套餐。如果不进行此更改，则空间只能使用免费套餐。
7. 选择验证空间。

   这将显示账户已验证成功消息，表示已将该账户添加到空间。

步骤 3：审查已批准的连接

在连接获得批准后，您可以在控制台中查看该连接以及您向其添加的 IAM 角色。

审查已批准的连接

1. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
2. 这将列出账户连接及其创建日期。
3. 选择账户显示名称。此时将显示 AWS 账户 详细信息页面。

步骤 4：将 IAM 角色添加到连接

如果您使用的是为 CodeCatalyst 部署操作配置的 IAM 角色，请将该角色添加到您的部署环境中。有关更多信息，请参阅 将 IAM 角色添加到账户连接。

后续步骤：为账户连接创建其他 IAM 角色

在创建连接后，您可以创建其他 IAM 角色以添加到该连接。您添加的 IAM 角色取决于您的工作流。例如， CodeCatalyst生成操作需要 CodeCatalyst 生成角色。

要连接您的账户，您将需要已创建角色的 Amazon 资源名称（ARN）。复制角色的 ARN，如此处所述。有关使用 IAM 角色 ARNs 的更多信息，请参阅 A mazon 资源名称 (ARN)。

访问 IAM 角色 ARN

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
3. 在搜索框中，输入要添加的角色的名称。
4. 从列表中选择该角色。

   此时将显示该角色的摘要页面。
5. 在顶部，复制角色 ARN 值。

将 IAM 角色添加到账户连接

创建账户连接的部分工作包括添加要用于您 CodeCatalyst 空间中的项目的一个或多个 IAM 角色。

> **Note**
>
> 要在账户连接中使用 IAM 角色，请确保将信任策略更新为使用 CodeCatalyst 服务委托人。

将 IAM 角色添加到账户连接（控制台）

1. 在中 AWS 管理控制台，请确保您使用要管理的相同帐户登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
4. 选择您的账户连接的 Amazon CodeCatalyst 显示名称，然后从中选择管理角色 AWS 管理控制台。

   将显示 “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面。
5. 请执行以下操作之一：

- 要创建包含开发者角色权限策略和信任策略的服务角色，请选择在 IAM 中创建 CodeCatalyst 开发管理员角色。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-spaceName`，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-**spaceName** 服务角色。

  选择创建开发角色。

- 要添加已在 IAM 中创建的角色，请选择添加现有 IAM 角色。在选择现有 IAM 角色中，从下拉列表中选择该角色。

  选择添加角色。

该页面将在 AWS 管理控制台中打开。您可能需要登录才能访问该页面。

6. 在 Amazon CodeCatalyst Spaces 页面的导航窗格中，选择空间。

   要直接访问该页面，请登录 a AWS 管理控制台 t https://console.aws.amazon.com/codecatalyst/home/ 中的 Amazon CodeCatalyst Spaces。

7. 选择为您的 CodeCatalyst 空间添加的账户。这将显示连接页面。
8. 在连接页面的可用的 IAM 角色下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表。选择将 IAM 角色关联到 CodeCatalyst。
9. 在“关联 IAM 角色”弹出窗口的“角色 ARN”中，输入要与空间关联的 IAM 角色的亚马逊资源名称 (ARN)。 CodeCatalyst

   在用途下，选择一个角色用途来描述您希望如何在账户连接中使用该角色。为用于在工作流中运行操作的角色指定 `RUNNER`。为用于访问其他服务的角色指定 `SERVICE`。

   您可以指定多个用途。

   > **Note**
   >
   > 必须为角色 ARN 选择用途。

10. 选择关联 IAM 角色。对其他 IAM 角色重复这些步骤。

将账户连接和 IAM 角色添加到部署环境

要访问 AWS 资源，例如 Amazon ECS 或用于部署的 AWS Lambda 资源， CodeCatalyst 构建和部署操作需要具有访问这些资源的权限的 IAM 角色。使用 Space 管理员或 Power 用户角色，您可以将自己的 CodeCatalyst 账户与资源创建 AWS 账户 地关联起来。之后，可以将 IAM 角色添加到您的账户连接。要执行部署操作，则必须将 IAM 角色添加到 CodeCatalyst 环境中。

您必须在项目中添加要用于部署环境的 IAM 角色。将角色添加到账户连接不会将角色和连接添加到项目部署环境中。要将您的账户连接和 IAM 角色添加到部署环境中，请确保按步骤 4：将 IAM 角色添加到连接中所述创建账户连接和角色。

然后，使用 CodeCatalyst 控制台中的环境页面将您的账户连接和 IAM 角色添加到项目的部署环境中。

> **Note**
>
> 只有将 IAM 角色用于需要 IAM 角色的 CodeCatalyst 操作时，您才可以向环境中添加 IAM 角色。所有需要 IAM 角色的工作流程操作（包括构建操作）都必须使用环境。 CodeCatalyst

将账户连接和 IAM 角色添加到部署环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到具有要在其中添加账户连接和 IAM 角色的部署环境的项目。
3. 展开 CI/CD，然后选择环境。
4. 选择您的环境，随后将显示其他选项卡。
5. 选择 AWS 账户 连接选项卡。在连接名称下，将列出已添加到环境的账户（如果有）。
6. 选择关联 AWS 账户。这将显示将 AWS 账户 与 <environment_name> 关联页面。
7. 在连接下，选择用于要添加的 IAM 角色的账户连接的名称。选择关联 。

查看账户连接

您可以查看连接列表并查看有关每个连接的详细信息。

您必须具有空间管理员或高级用户角色才能管理空间的连接。

查看 CodeCatalyst 空间的所有连接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到具有要查看的账户连接的空间。
3. 选择 AWS 账户选项卡。
4. 在 AWS 账户下，查看该空间的账户连接列表，包括每个连接的账户 ID 和状态。

查看账户连接详细信息

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 在 Amazon CodeCatalyst 显示名称中，选择连接名称。在详细信息页面上，查看与该连接关联的 IAM 角色的列表以及其他详细信息。

删除账户连接（中 CodeCatalyst）

您可以删除不再需要的账户连接。在本步骤中，您将使用 CodeCatalyst 删除先前添加到空间中的账户连接。这将从您的空间中删除账户连接，前提是该账户不是空间的计费账户。

> **Important**
>
> 删除一个账户连接后，您将无法重新连接它。您必须创建新的账户连接，然后根据需要关联 IAM 角色和环境或设置账单。

必须为您的 CodeCatalyst 空间指定一个账单账户，即使空间的使用量不会超过免费套餐。您需要先为您的空间添加另一个账户，之后才能为用作指定计费账户的账户移除空间。请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

> **Important**
>
> 虽然您可以使用这些步骤来移除账户，但建议不要这样做。也可以将该帐户设置为支持中的工作流程 CodeCatalyst。

要管理空间的账户连接，您必须具有空间管理员或高级用户角色。

稍后可以重新添加已移除的账户，但您必须在账户和空间之间创建新的连接。您将需要将所有 IAM 角色重新关联到已添加的账户。

删除账户连接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 在 Amazon CodeCatalyst 显示名称下，选择要删除的账户关联旁边的选择器。
4. 选择移除 AWS 账户。在字段中输入名称来确认删除，然后选择移除。

   这将显示成功横幅，表明已从连接列表中移除账户连接。

为空间配置计费账户

必须为您的 CodeCatalyst 空间指定一个账单账户，即使空间的使用量不会超过免费套餐。

要配置结算账号，请参阅《CodeCatalyst 管理员指南》中的账单。指定 AWS 账户 为空间结算账号的配额与 CodeCatalyst 空间的其他账户连接的配额不同。有关更多信息，请参阅 CodeCatalyst 的配额。

要移除作为您 CodeCatalyst 房源指定结算账户的账户，请务必先指定一个新的结算账号。

为关联账户配置 IAM 角色

您可以在 AWS Identity and Access Management (IAM) 中为要添加的账户创建角色 CodeCatalyst。如果添加的是计费账户，则无需创建角色。

在您的中 AWS 账户，您必须拥有为要添加到空间中的角色创建角色的权限。 AWS 账户 有关 IAM 角色和策略的更多信息，包括 IAM 参考和策略示例，请参阅 Identity and Access Management 和亚马逊 CodeCatalyst。有关中使用的信任策略和服务主体的更多信息 CodeCatalyst，请参阅了解 CodeCatalyst 信任模型。

在中 CodeCatalyst，您必须使用空间管理员角色登录才能完成向空间添加帐户（以及角色，如果适用）的步骤。

您可以使用以下方法之一为账户连接添加角色。

- 要创建包含 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色的权限策略和信任策略的服务角色，请参阅 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。
- 有关创建角色并添加策略以根据蓝图创建项目的示例，请参阅创建 IAM 角色并使用 CodeCatalyst信任策略。

- 有关创建 IAM 角色时要使用的示例角色策略列表，请参阅使用 IAM 角色授予对项目 AWS 资源的访问权限。
- 有关为工作流操作创建角色的详细步骤，请参阅该操作的工作流教程，如下所示：
  - 教程：将构件上传到 Amazon S3
  - 教程：部署无服务器应用程序
  - 教程：将应用程序部署到 Amazon ECS
  - 教程：使用 GitHub Action 对代码进行 lint 检查

主题

- CodeCatalystWorkflowDevelopmentRole-spaceName 角色
- AWSRoleForCodeCatalystSupport 角色
- 创建 IAM 角色并使用 CodeCatalyst信任策略

CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色

您只需在 IAM 中点击一下，即可创建开发人员角色。您必须在要添加账户的空间中拥有空间管理员或高级用户角色。您还必须对要添加的 AWS 账户 具有管理权限。

在开始以下步骤之前，您必须 AWS 管理控制台 使用要添加到 CodeCatalyst 空间的相同帐户登录。否则，控制台将返回未知账户错误。

要创建并添加 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-*spaceName*

1. 在开始进入 CodeCatalyst 控制台之前，请先打开 AWS 管理控制台，然后确保您的空间使用相同 AWS 账户 的方式登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
4. 选择要创建角色的 AWS 账户 位置的链接。此时将显示 AWS 账户 详细信息页面。
5. 从中选择 “管理角色” AWS 管理控制台。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。
6. 选择在 IAM 中创建 CodeCatalyst 开发管理员角色。此选项将创建一个服务角色，其中包含开发角色的权限策略和信任策略。该角色的名称为

CodeCatalystWorkflowDevelopmentRole-*spaceName*。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。

> **Note**
>
> 此角色仅建议与开发者账户一起使用，并且使用AdministratorAccess AWS 托管策略，授予其在其中创建新策略和资源的完全访问权限 AWS 账户。

7. 选择创建开发角色。
8. 在连接页面的可用的 IAM 角色下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表中的角色。CodeCatalystWorkflowDevelopmentRole-*spaceName*
9. 要返回您的空间，请选择 Go to Amazon CodeCatalyst。

AWSRoleForCodeCatalystSupport 角色

您只需在 IAM 中点击一下，即可创建支持角色。您必须在要添加账户的空间中拥有空间管理员或高级用户角色。您还必须对要添加的 AWS 账户 具有管理权限。

在开始以下步骤之前，您必须 AWS 管理控制台 使用要添加到 CodeCatalyst 空间的相同帐户登录。否则，控制台将返回未知账户错误。

要创建并添加 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在开始进入 CodeCatalyst 控制台之前，请先打开 AWS 管理控制台，然后确保您的空间使用相同 AWS 账户 的方式登录。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 选择要创建角色的 AWS 账户 位置的链接。此时将显示 AWS 账户 详细信息页面。
4. 从中选择 “管理角色” AWS 管理控制台。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。
5. 在CodeCatalyst 空间详情下，选择添加 Su CodeCatalyst pport 角色。此选项将创建一个服务角色，其中包含预览开发角色的权限策略和信任策略。该角色的名称为 AWSRoleForCodeCatalystSupport，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 AWSRoleForCodeCatalystSupport 服务角色。
6. 在 “为 Su CodeCatalyst pport 添加角色” 页面上，将默认角色保留为选中状态，然后选择创建角色。

7. 在 “可用的 IAM 角色” CodeCatalystWorkflowDevelopmentRole-*spaceName* 下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表中的角色。
8. 要返回您的空间，请选择 Go to Amazon CodeCatalyst。

创建 IAM 角色并使用 CodeCatalyst信任策略

在 AWS 账户 连接中 CodeCatalyst 使用的 IAM 角色必须配置为使用此处提供的信任策略。使用这些步骤创建 IAM 角色并附加允许您根据蓝图创建项目的策略。 CodeCatalyst

作为替代方法，您可以创建一个服务角色，其中包含 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色的权限策略和信任策略。有关更多信息，请参阅 将 IAM 角色添加到账户连接。

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 选择 角色，然后选择 创建角色。
3. 选择自定义信任策略。
4. 在自定义信任策略表单下，粘贴以下信任策略。

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/
*"
                }
            }
        }
    ]
```

5. 选择下一步。
6. 在添加权限下，搜索并选择您已在 IAM 中创建的自定义策略。
7. 选择下一步。
8. 对于角色名称，输入角色的名称，例如：`codecatalyst-project-role`
9. 选择创建角色。
10. 复制角色的 Amazon 资源名称（ARN）。在将该角色添加到您的账户关联或环境中时，您需要提供该信息。

向用户授予空间权限

您可以通过查看、添加、移除或更改加入空间的用户的角色来管理空间的成员。

本指南中的信息用于在支持 AWS Builder ID 用户的空间 CodeCatalyst 中邀请和管理用户。要详细了解设置和管理支持身份联合的空间的步骤，请参阅《Amazon CodeCatalyst 管理员指南》中的CodeCatalyst 空间设置和管理。

查看空间中的成员

您可以查看空间中的用户，包括他们的显示名称、别名和空间角色等信息。空间中的成员有三种角色：

- 空间管理员-此角色拥有所有权限 CodeCatalyst，包括创建项目。仅将该角色分配给需要管理空间各个方面（如访问空间中的所有项目）的用户。

  如果不先移除用户，以后就无法更改该角色。有关更多信息，请参阅 空间管理员角色。
- 高级用户 — 此角色是 Amazon CodeCatalyst 空间中第二强大的角色，但它无法访问空间中的项目。该角色专为需要能够在空间中创建项目并帮助管理空间的用户和资源的用户而设计。有关更多信息，请参阅 高级用户角色。
- 有限访问权限 - 对于接受空间项目邀请而加入空间的用户，默认情况下会分配该角色。项目成员在项目中被分配一个角色。有关管理项目成员的信息，请参阅向用户授予项目权限。

空间管理员表显示了具有空间管理员角色的用户。这些用户不会显示在空间成员中，因为他们被自动（隐式）分配给空间中的所有项目，并且在项目中没有角色。

空间成员表显示了空间中所有在项目中具有某个角色但不具有空间管理员角色的成员。

根据用户是否具有空间管理员角色来显示用户， CodeCatalyst 如下所示：

- 具有空间管理员角色的用户如果后来接受了项目邀请和角色，将不会显示在空间下的空间成员表或项目下的项目成员表中。它们将继续显示在这两个位置的空间管理员表中。在每个项目中，所有具有空间管理员角色的用户都会显示在该项目的空间管理员表中。
- 接受项目邀请以项目角色加入的用户将以有限访问权限角色添加到空间中。如果用户的角色后来更改为空间管理员角色，也会从空间成员表移至空间管理员表。在项目下，用户将从项目成员表移至空间管理员表。

查看您空间中的用户和角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择成员。

   作为空间成员的用户显示在空间成员表中。

> **Tip**
>
> 如果您拥有空间管理员角色，则可以查看您直接受邀参与的项目。导航到项目的项目设置，然后选择我的项目。

在状态列中，以下为有效值：

- 已@@ 邀请 — 已 CodeCatalyst 发送邀请，但用户尚未接受或拒绝。
- 成员 - 用户接受了邀请。

直接邀请用户进入空间

您可以直接邀请用户进入您的 CodeCatalyst 空间。当您想通过为用户分配空间管理员或高级用户角色来邀请该用户帮助您管理空间时，这非常有用。将其中一个角色分配给其他用户，有助于您将管理空间的责任分配给更多人，而不必邀请这些用户参与任何项目。

> **Note**
>
> 您必须拥有空间管理员或高级用户角色才能邀请成员。

空间管理员表显示了具有空间管理员角色的用户。这些用户不会显示在空间成员表中，因为他们被自动（隐式）分配给空间中的所有项目，并且在项目中没有角色。

默认情况下，接受项目邀请的成员会被添加到空间中。项目成员表显示空间中具有项目角色的所有成员。

有关如何接受邀请并首次登录的更多信息，请参阅设置并登录 CodeCatalyst。

邀请用户进入您的空间

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。
3. 选择设置，然后选择成员。
4. 选择邀请。
5. 输入您想邀请加入您空间的人员的电子邮件地址。在角色中，选择要在空间中为该用户分配的角色。
6. 选择邀请

取消空间邀请

如果您想取消最近发出的加入空间的邀请，且邀请尚未被接受，您可以取消邀请。

要管理空间邀请，您必须具有空间管理员或高级用户角色。

取消空间成员邀请

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择成员。
4. 验证成员的状态是否为已邀请。

> **Note**
>
> 您只能取消尚未接受的邀请。

5. 选择受邀成员所在行旁边的选项，然后选择取消邀请。
6. 此时将显示一个确认窗口。选择取消邀请进行确认。

更改空间成员的角色

您可以更改为空间成员分配的角色。您必须拥有空间管理员角色才能更改空间中用户的角色。

空间管理员表显示了具有空间管理员角色的用户。这些用户不会显示在空间成员表中，因为他们被自动（隐式）分配给空间中的所有项目。

更改空间中用户的角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择成员。
4. 在空间成员表中，选择要更改其角色的用户。选择更改角色。

移除空间成员

当空间成员不需要访问任何空间资源时，您可以将其从空间中移除。您必须拥有空间管理员角色才能从空间中移除成员。

空间管理员表显示了具有空间管理员角色的用户。这些用户不会显示在空间成员表中，因为他们被自动（隐式）分配给空间中的所有项目，并且在项目中没有角色。您只能直接移除该表中空间的成员。

从项目成员表中移除用户

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择成员。
4. 在项目成员表中选择用户。选择移除 。

> **Note**
>
> 从空间中移除成员会将该用户从空间中的所有项目中移除，同时移除与这些项目中的资源相关的权限。

移除或更改具有空间管理员角色的用户的角色

您可以移除或更改空间中具有空间管理员角色的用户的角色。

您必须具有空间管理员角色才能将具有空间管理员角色的用户从空间中移除。更改具有空间管理员角色的用户的角色实质上是将该用户从空间管理员表中移除。如果该用户在空间中的任何项目中都没有项目角色，则从该用户中移除空间管理员角色会将该用户从空间中移除。

> **Note**
>
> 作为具有空间管理员角色的用户，您无法移除自己。联系其他具有空间管理员角色的用户。

从空间成员表中移除具有空间管理员角色的用户

> **Note**
>
> 对于未明确添加到项目的用户，他们没有任何项目角色（项目管理员或贡献者）。如果空间管理员角色是用户的唯一角色，则该用户将完全从空间中移除。

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中移除或更改具有空间管理员角色的用户的角色的空间。
3. 选择设置，然后选择成员。
4. 查看成员列表的邀请状态，确保列表中没有未经授权的待处理空间邀请（状态为已邀请）。

> **Important**
>
> 在移除具有空间管理员角色的用户之前，您必须确认没有发出任何待处理的邀请。

5. 选择成员选项卡。在空间管理员表中，选择用户，然后选择移除。

   在移除成员对话框中，执行以下操作之一。

   - 选择仅移除用户的空间管理员角色的选项。选择移除 。

   > **Important**
   >
   > 如果没有为用户分配任何其他角色，则从空间管理员更改角色会将该用户从空间中移除。

   - 选择将具有空间管理员角色的用户从空间及其所有项目中移除的选项。选择移除 。
6. 刷新成员选项卡。在用户通过项目角色拥有成员资格的任何项目中，该用户会自动添加到项目成员列表中。如果空间管理员角色是用户的唯一角色，则该用户将完全从空间中移除。

允许使用团队访问空间

创建空间后，您可以添加团队。团队允许您对用户进行分组，以便他们可以共享权限并管理项目、问题跟踪、角色和资源 CodeCatalyst。

您必须拥有空间管理员角色才能管理团队。

团队也按 project/space 级别进行管理 CodeCatalyst。要了解有关空间/项目中的团队的更多信息，请参阅允许使用团队访问空间。

主题

- 创建团队
- 查看团队
- 为团队授予空间角色

- 在空间级别为团队授予项目角色
- 直接向团队添加用户
- 直接从团队中移除用户
- 向团队添加 SSO 组
- 删除团队

创建团队

团队可以在空间中拥有角色权限，例如高级用户。团队还可以在项目中拥有项目权限，例如项目管理员。团队可以与许多项目关联，并在每个项目中具有不同的角色。您可以管理团队，其中团队成员要么是 AWS Builder ID 空间的个人用户，要么是支持身份联合的空间的 SSO 群组。

在空间和项目用户的成员页面上，用户可以拥有多个角色。拥有多个角色的用户在拥有多个角色时将显示一个指示器，并且首先会显示其中权限最多的角色。

> **Note**
>
> 如果您的空间支持身份联合验证，则您必须已在 IAM Identity Center 中设置 SSO 用户或 SSO 组。

如何管理团队成员取决于添加和移除用户的方式。对于管理团队成员，有两种选择：

- 直接添加用户 – 您可以单独添加或移除用户。例如，您可以通过选择 AWS Builder ID 用户或已在 IAM Identity Center 中设置的 SSO 用户来将用户添加到团队中。当您选择通过直接添加 AWS Builder ID 用户或 SSO 用户来管理团队成员时，使用 SSO 群组的选项将不再可用。
- 使用 SSO 组 – 您可以通过已在 IAM Identity Center 中设置的 SSO 组来管理团队成员。在您选择通过使用 SSO 组来管理团队成员时，用于直接添加用户的选项将不再可用。

您必须拥有空间管理员角色才能管理团队。

创建团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择创建团队。

4. 在团队名称中，输入您团队的描述性名称。

> **Note**
>
> 该团队名称在空间内必须是唯一的。

（可选）在团队描述中，输入您团队的描述。

5. 在 “空间角色” 下，从可用空间角色列表中选择 CodeCatalyst 要分配给团队的角色。团队的所有成员都将继承该角色。

   - 空间管理员 – 有关详细信息，请参阅空间管理员角色。
   - 受限访问 – 有关详细信息，请参阅受限的访问角色。
   - 高级用户 – 有关详细信息，请参阅高级用户角色。

6. 在团队成员资格中，选择以下选项之一，来选择向团队添加成员的方法。

   - 选择直接添加成员以单独管理用户。这包括为空间添加 AWS Builder ID 用户或为支持身份联合的空间添加 SSO 用户。
   - 选择使用 SSO 组以选择已在 IAM Identity Center 中设置的 SSO 组。

     在 SSO 组中，选中要添加的组旁边的框。您可以添加最多 5 个 SSO 组。

> **Note**
>
> 您以后不能更改此名称。在您选择通过直接添加 AWS 构建者 ID 用户或 SSO 用户来管理团队成员时，SSO 组选项将不再可用。在您选择通过使用 SSO 组来管理团队成员时，用于直接添加用户的选项将不再可用。

7. 选择创建。

> **Note**
>
> 当您选择使用 SSO 组时，请注意，在创建团队时不会拉出 SSO 组中的用户。用户需要先登录 CodeCatalyst 才能显示在列表中。

查看团队

在中 CodeCatalyst，您可以查看团队的项目和角色。在成员页面上，您可以查看项目角色和用户列表。对于 SSO 组类型的团队，您还可以看到与该团队关联的 SSO 组的列表。

查看团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 在空间角色中，查看分配给此空间的团队的角色。
4. 在项目角色选项卡上，在已将团队添加为成员的空间中，查看为每个 CodeCatalyst 项目分配给团队的项目和项目角色（仅适用于 AWS 建造者 ID 空间）。
5. 在成员选项卡上，查看分配给团队的成员的列表。
6. 在 SSO 组选项卡上，查看分配给团队的 SSO 组的列表（仅适用于支持身份联合验证的空间）。

为团队授予空间角色

团队是一种对用户进行分组的方式，这样您就可以授予和管理团队对中项目的访问权限 CodeCatalyst。例如，您可以给与团队管理用户空间的能力，从而使用团队来快速管理用户的角色和权限。

团队可以在空间中拥有角色权限，例如高级用户。您可以更改团队的空间角色，但请注意，团队的所有成员都将继承这些权限。

您必须拥有空间管理员角色才能管理团队。

更改团队的空间角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 在操作中，选择更改空间角色。您可以将空间角色更改为以下角色之一。这将更改团队中所有成员的角色。

   - 空间管理员 – 有关详细信息，请参阅空间管理员角色。
   - 受限访问 – 有关详细信息，请参阅受限的访问角色。
   - 高级用户 – 有关详细信息，请参阅高级用户角色。

4. 选择保存。

在空间级别为团队授予项目角色

中的 CodeCatalyst 团队与用户类似，因为团队成员可以在项目中拥有角色权限，例如项目管理员。角色更改将应用于团队，并且所有团队成员都将继承这些权限。您可以为每个项目选择一个将自动授予给团队的角色。

您必须拥有空间管理员角色才能管理团队。

添加或更改项目角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择项目角色选项卡。
4. 要更改角色，请选择此列表中项目旁边的选择器，然后选择更改角色。要添加角色，请选择添加项目角色。在项目中，选择要添加的项目，然后在角色中选择角色。选择一个可用的项目角色：
   - 项目管理员 – 有关详细信息，请参阅项目管理员角色。
   - 贡献者 – 有关详细信息，请参阅贡献者角色。
   - 审阅者 – 有关详细信息，请参阅审阅者角色。
   - 只读 – 有关详细信息，请参阅只读角色。
5. 选择保存。

移除项目角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择项目角色选项卡。
4. 选择要移除的角色。

> **Important**
>
> 从团队中移除一个角色会移除团队中所有用户的关联权限。

5. 选择保存。

直接向团队添加用户

您可以向团队添加团队成员。添加用户时，新用户将继承团队中所有现有角色的权限。

无论您的空间是为 AWS Builder ID 用户支持还是身份联合设置的，您都可以将空间设置为直接添加用户。

> **Note**
>
> 当您的空间设置为使用 SSO 组管理团队成员时，无法使用直接添加用户选项。要使用 SSO 组，请参阅向团队添加 SSO 组。

您必须拥有空间管理员角色才能管理团队。

直接添加用户

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择成员选项卡。
4. 选择添加成员。

   > **Note**
   >
   > 要添加到团队的用户必须已经是空间的成员。您无法添加或邀请不具有空间成员身份的团队成员。

5. 在下拉字段中选择一个用户，然后选择保存。选择已在 IAM Identity Center 中设置的 AWS Builder ID 用户或单点登录用户。

直接从团队中移除用户

您可以从团队中移除团队成员。所有权限将不再由用户继承。您可以稍后重新将用户添加到团队中。

> **Note**
>
> 在移除团队成员时，将从空间中的所有项目和资源中移除该用户的关联权限。

您必须拥有空间管理员角色才能管理团队。

移除团队成员

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择成员选项卡。
4. 选择要移除的用户旁边的选择器，然后选择移除。
5. 在文本输入字段中输入 remove，然后选择移除。

向团队添加 SSO 组

如果您的空间已配置为在 IAM Identity Center 中管理 SSO 用户和组的空间，则可以添加一个 SSO 组，该组将作为单独团队加入空间。

> **Note**
>
> 当您选择通过直接添加 AWS Builder ID 用户或 SSO 用户来管理团队成员时，使用 SSO 群组的选项不可用。要直接添加用户，请参阅直接向团队添加用户。

您必须拥有空间管理员角色才能管理团队。

将 SSO 组添加为团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在您的空间页面上，选择团队。选择 SSO 组选项卡。
3. 选择要添加的 SSO 组。您可以添加最多 5 个 SSO 组。

删除团队

您可以删除不再需要的团队。

> **Note**
>
> 在删除团队时，将从空间中的所有项目和资源中移除所有团队成员的关联权限。

您必须拥有空间管理员角色才能管理团队。

删除团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 在操作中，选择删除团队。这将更改整个团队的角色。
4. 选择删除。

允许机器资源的空间访问权限

计算机资源是中 CodeCatalyst 被授予项目或空间权限的特定资源 CodeCatalyst。

> **Note**
>
> “机器资源” 一词并不指诸如 Amazon EC2 实例之类的云基础设施，而是指具有空间或项目权限的蓝图或工作流程资源。

当 CodeCatalyst 通过 SSO 进行访问时，计算机资源代表您来自授权资源的身份。机器资源用于向空间中的资源（例如蓝图和工作流）授予权限。您可以查看空间中的机器资源，也可以选择启用或禁用空间的机器资源。例如，您可能希望禁用机器资源以管理访问权限，并在稍后重新启用机器资源。

在需要撤销或禁用机器资源的情况下，可对机器资源执行这些操作。例如，如果您怀疑凭证可能已被泄露，则可以禁用机器资源。通常，不需要使用这些操作。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理机器资源。

计算机资源也在中的项目级别进行管理 CodeCatalyst。要了解有关项目中的团队的更多信息，请参阅允许机器资源的空间访问权限 。

主题

- 查看机器资源的空间访问权限
- 禁用机器资源的空间访问权限
- 启用机器资源的空间访问权限

查看机器资源的空间访问权限

您可以查看正在空间中使用的机器资源的列表。

您必须拥有空间管理员角色才能管理机器资源。

查看机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间，然后选择设置。选择机器资源。
3. 在下拉列表中，选择工作流操作以仅查看工作流的机器资源。选择蓝图以仅查看蓝图的机器资源。

   您也可以使用筛选条件字段对名称进行筛选。

禁用机器资源的空间访问权限

您可以选择禁用正在空间中使用的机器资源。

> **Important**
>
> 禁用机器资源将移除对空间中所有关联的蓝图或工作流的所有权限。

您必须拥有空间管理员角色才能管理机器资源。

禁用机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间，然后选择设置。选择机器资源。
3. 选择下列选项之一。

   > **Important**
   >
   > 禁用机器资源将移除对空间中所有关联的蓝图或工作流的所有权限。

   - 要单独禁用，请选择要禁用的一个或多个机器资源旁边的选择器。选择禁用，然后选择此资源。
   - 要禁用所有资源，请选择禁用，然后选择所有资源。

- 要禁用所有工作流操作，请选择禁用，然后选择所有工作流操作。
- 要禁用所有蓝图，请选择禁用，然后选择所有蓝图。

启用机器资源的空间访问权限

您可以选择启用正在空间中使用的和已禁用的机器资源。

您必须拥有空间管理员角色才能管理机器资源。

启用机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间，然后选择设置。选择机器资源。
3. 选择下列选项之一。
   - 要单独启用，请选择要启用的一个或多个机器资源旁边的选择器。选择启用，然后选择此资源。
   - 要启用所有资源，请选择启用，然后选择所有资源。
   - 要启用所有工作流操作，请选择启用，然后选择所有工作流操作。
   - 要启用所有蓝图，请选择启用，然后选择所有蓝图。

管理空间的开发环境

所有开发环境都是作为空间中项目的一部分创建的。空间成员可以在源存储库级别的项目中创建自己的开发环境。然后，空间管理员可以使用 Amazon CodeCatalyst 控制台代表空间成员查看、编辑、删除和停止开发环境。简而言之，空间管理员在空间级别维护开发环境。

管理开发环境的注意事项

- 您必须具有空间管理员角色，才能查看设置下的开发环境页面，并在空间级别管理开发环境。
- 空间成员通过自己的 CodeCatalyst 账户管理他们在项目中创建的开发环境。以空间管理员身份管理开发环境时，您将代表空间成员维护这些资源。
- 开发环境默认使用特定的计算和存储配置。有关升级配置的账单和费率的信息，请参阅 Amazon CodeCatalyst 定价页面。

> **Important**
>
> 开发环境不适用于将 Active Directory 用作身份提供商的空间中的用户。有关更多信息，请参阅 当我使用单点登录账户登录 CodeCatalyst 时，无法创建开发环境。

有关开发环境的其他注意事项，包括停止正在运行的实例、默认计算配置、升级计算、产生成本和配置超时，请参阅使用开发环境编写和修改代码 CodeCatalyst。

主题

查看空间的开发环境

您可以查看空间中所有开发环境的类型、状态和详细信息。有关创建和运行开发环境的更多信息，请参阅创建开发环境。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理开发环境。

查看空间中的开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

   > **Tip**
   >
   > 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择开发环境。

   该页面列出了您的空间中的所有开发环境。您可以查看每个开发环境的资源名称、资源别名（如果适用）、IDE 类型、默认或配置的计算和存储以及配置的超时。

为您的空间编辑开发环境

您可以编辑开发环境的配置，例如空闲开发环境停止运行的配置超时长度（如果有）。有关编辑开发环境的更多信息，请参阅编辑开发环境。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理开发环境。

编辑空间中的开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择开发环境。
4. 选择要管理的开发环境旁边的选择器。选择编辑。
5. 对开发环境的计算或不活动超时进行所需的更改。
6. 选择保存。

停止空间的开发环境

如果开发环境配置为会超时，则可以在正在运行的开发环境变为空闲之前将其停止。否则，已超时的开发环境将已经停止。有关停止开发环境的更多信息，请参阅停止开发环境。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理开发环境。

停止空间中的开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择开发环境。

4. 选择要管理的开发环境旁边的选择器。选择停止。

删除空间的开发环境

您可以删除不再需要或不再有拥有者的开发环境。有关删除开发环境的注意事项的更多信息，请参阅删除开发环境。

您必须拥有空间管理员角色才能查看此页面并在空间级别管理开发环境。

删除空间中的开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 选择设置，然后选择开发环境。
4. 选择要管理的开发环境旁边的选择器。选择删除。键入 `delete` 进行确认，然后选择删除。

空间配额

下表描述了 Amazon 空间的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

空间的 Slack 频道的最大数量	500
一个电子邮件地址的最多邀请次数	25
一个用户的最多邀请次数	500
每位用户每位用户的最大活动空间数 AWS 区域	5
每个用户每月在每个区域创建的空间最大数量	5
一个团队的 SSO 组的最大数量	5

一个空间的最大团队数	100
一个团队的最大用户数	1000
空间描述	空间描述是可选的。如果指定，长度必须在 0 到 200 个字符之间。它们可以包含字母、数字、空格、句点、下划线、逗号、短划线和以下特殊字符的任意组合： `? & $ % + = / \ ; : \n \t \r`
空间名称	空间名称在各处必须是唯一的 CodeCatalyst。不能重复使用已删除空间的名称。 空间名称长度必须在 3 至 63 个字符之间。它们还必须以字母数字字符开头。空间名称可包含字母、数字、句点、下划线和短划线的任意组合。它们不能包含以下任何字符： `! ? @ # $ % ^ & * ( ) + = { } [ ] | 八 > < ~ ` ' " ; :`

使用项目来组织工作 CodeCatalyst

您可以使用 Amazon 中的项目 CodeCatalyst 来建立协作空间，开发团队可以在其中使用共享持续 integration/continuous 交付 (CI/CD) 工作流程和存储库执行开发任务。创建项目时，您可以添加、更新或移除资源。您还可以监控团队的工作进度。一个空间内可以有多个项目。

中的空间 CodeCatalyst 由项目组成。您可以查看空间中的每个项目，但只能使用您作为其成员的项目。创建项目时，将生成项目的默认角色，您可将这些角色分配给您邀请加入项目的用户。

- 分配给项目且具有项目角色（例如贡献者角色）的任何人员都能访问项目资源，例如源存储库。
- 任何具有空间管理员或项目管理员角色的人员都可以发送项目加入邀请。
- 具有项目管理员角色的用户可以跟踪共享资源中的活动、状态和其他设置。
- 具有受限访问权限角色的用户可以管理作为 CI/CD 工作流程一部分的功能、代码修复和测试的项目分配。

  工作流用于根据您要设置的活动和输出来构建、测试和发布或更新应用程序，将其作为CI/CD pipeline. You can assemble workflows by adding actions that transfer and work on your source artifacts. When you run actions, your project cloud resources are used to provide on-demand compute ability for your workflow actions. You might configure more CI/CD工作流程。例如，您可以创建一个仅用于构建和测试操作的工作流，在修复错误时，无需部署即可查看测试结果并完成工作流。之后，您可以创建另一个工作流来构建应用程序并将其部署到暂存环境。

创建项目时，您可以使用蓝图创建包含示例代码并创建资源的项目，也可以从空项目开始操作。如果您使用蓝图创建项目，则您选择的蓝图将决定将哪些资源添加到您的项目中，以及用于 CodeCatalyst 创建或配置哪些工具，以便您可以跟踪和使用项目资源。创建项目后，可以手动添加或移除资源。

每个项目都将项目活动作为按用户划分的事件列表进行跟踪，例如在创建项目或修改资源时。将在空间级别监控和汇总项目活动。有关使用活动数据的更多信息，请参阅查看空间中的所有项目。

如果您的项目使用 AWS 资源，则可以将您的 CodeCatalyst 账户关联到一个拥有管理权限的 AWS 账户，以便为项目整合资源。

创建项目后，可以将源存储库、事务和其他资源添加到项目中。您必须具有空间管理员角色才能创建项目。

创建项目

通过 CodeCatalyst 项目，您可以使用共享持续 integration/continuous 交付 (CI/CD) 工作流程和存储库执行开发任务、管理资源、跟踪问题和添加用户。

在创建项目之前，您必须具有空间管理员或高级用户角色。

主题

在 Amazon 中创建一个空项目 CodeCatalyst

您可以创建没有资源的空项目，并在稍后手动添加所需的资源。

在创建项目之前，您必须具有空间管理员或高级用户角色。

创建空项目

1. 导航到要在其中创建项目的空间。
2. 在空间控制面板上，选择创建项目。
3. 选择从头开始。
4. 在为项目命名下，输入要分配给项目的名称。该名称在空间内必须是唯一的。
5. 选择创建项目。

使用链接的第三方存储库创建项目

您可以将项目的源代码保存在首选的第三方提供商中，但仍可使用蓝图、生命周期管理、工作流程等所有 CodeCatalyst 功能。为此，您可以创建一个链接到 GitHub 存储库、Bitbucket 存储库或 CodeCatalyst 项目存储库的新 GitLab 项目。然后，您可以在 CodeCatalyst 项目中使用您的链接源代码库。

在创建 CodeCatalyst 项目之前，您必须拥有 Space 管理员或 Power 用户角色。有关更多信息，请参阅创建空间和直接邀请用户进入空间 。

要在链接到您 GitHub 账户中的 CodeCatalyst 源存储库的项目中创建项目，您需要完成以下三个任务：

1. 安装GitHub 存储库、Bitbucket 存储库或GitLab 存储库扩展。在外部站点中，系统会提示您连接并 CodeCatalyst 提供对存储库的访问权限，这将在下一步中完成。

   > **Important**
   >
   > 要将GitHub 存储库、Bitbuck et GitLab 存储库或存储库扩展安装到您的 CodeCatalyst 空间，您必须使用在该空间中具有空间管理员角色的账户登录。

2. 将您的 GitHub 账户、Bitbucket 工作空间或 GitLab 用户连接到。 CodeCatalyst

   > **Important**
   >
   > 要将您的 GitHub 账户、Bitbucket 工作 CodeCatalyst 空间、 GitLab 用户连接到您的空间，您必须同时是第三方来源的管理员和 CodeCatalyst 空间管理员。

   > **Important**
   >
   > 安装存储库扩展后，您链接到的任何存储库都 CodeCatalyst 将对其代码进行索引和存储。CodeCatalyst这将使代码可在中 CodeCatalyst搜索。要更好地了解在中使用链接存储库时代码的数据保护 CodeCatalyst，请参阅 Amazon CodeCatalyst 用户指南中的数据保护。

3. 创建 CodeCatalyst 链接到您的 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库的项目。

   > **Important**
   >
   > 虽然您可以作为贡献者链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目仓库，但您只能以 Space 管理员或项目管理员的身份取消第三方仓库的链接。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

> **Note**
>
> - GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库只能链接到空间中的一个 CodeCatalyst 项目。
> - 您不能将空仓库或已存档 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库与 CodeCatalyst 项目一起使用。
> - 您不能链接与 GitLab 项目中 GitHub 仓库同名的存储库、Bitbucket 存储库或 CodeCatalyst 项目存储库。
> - GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。
> - Bitbucket 存储库扩展与 Bitbucket Data Center 存储库不兼容。
> - GitLab 存储库扩展与 GitLab 自行管理的项目存储库不兼容。
> - 您无法在已链接存储库中使用为我编写描述或汇总评论功能。这些功能仅在中的拉取请求中可用 CodeCatalyst。

有关更多信息，请参阅 为带有扩展程序的项目添加功能 CodeCatalyst。

安装第三方扩展

1. 导航到要在其中创建项目的空间。
2. 在空间控制面板上，选择创建项目。
3. 选择自带代码。
4. 在 “链接现有存储库” 下，根据要使用的第三方存储库提供商选择GitHub GitLab 存储库、Bitbucket 存储库和存储库。如果您之前没有关联您的 GitHub 账户、Bitbucket 工作空间或 GitLab 账户，则系统会提示您进行关联。如果您选择的第三方扩展尚未安装，系统会显示安装提示。

5. 如果出现提示，请选择安装。查看扩展所需的权限，如果要继续，请再次选择安装。

安装第三方扩展程序后，下一步是将您的 GitHub 账户、Bitbucket 工作空间或 GitLab 用户连接到您的 CodeCatalyst 空间。

将您的 GitHub 账户、Bitbucket 工作空间或 GitLab 用户连接到 CodeCatalyst

根据您选择配置的第三方扩展执行下列操作之一：

- GitHub 存储库：Connect 连接到 GitHub 帐户。

  1. 选择 Connect GitHub 帐户以访问外部站点 GitHub。
  2. 使用您的 GitHub 凭证登录您的 GitHub 账户，然后选择要安装Amazon的账户 CodeCatalyst。

     > **Tip**
     >
     > 如果您之前已将 GitHub 账户关联到该空间，则系统不会提示您重新授权。相反，如果您是多个空间的成员或合作者，则会看到一个对话框，询问您要在哪里安装扩展程序；如果您只属于一个 GitHub 空间，则会看到Amazon CodeCatalyst 应用程序的配置页面。 GitHub 为要允许的存储库访问权限配置应用程序，然后选择保存。如果保存按钮未激活，请更改配置，然后重试。

  3. 选择是 CodeCatalyst 允许访问所有当前和将来的存储库，还是选择要在中使用的特定 GitHub 存储库 CodeCatalyst。默认选项是将 GitHub 账户中的所有 GitHub 存储库包括在内，包括将由访问的 future 存储库 CodeCatalyst。
  4. 查看授予的权限 CodeCatalyst，然后选择 “安装”。

  将您的 GitHub 账户关联到后 CodeCatalyst，系统会将您带到GitHub 存储库扩展详情页面，您可以在其中查看和管理关联的 GitHub 账户和关联的 GitHub 存储库。

- Bitbucket 存储库：连接到 Bitbucket 工作区。

  1. 选择连接 Bitbucket 工作区以转到 Bitbucket 的外部站点。
  2. 使用您的 Bitbucket 凭据登录您的 Bitbucket 工作空间并查看授予的权限。 CodeCatalyst
  3. 从 “授权工作空间” 下拉菜单中，选择要提供 CodeCatalyst 访问权限的 Bitbucket 工作空间，然后选择 “授予访问权限”。

> **Tip**
>
> 如果您之前已将 Bitbucket 工作区连接到空间，则系统不会提示您重新授权。相反，如果您是多个 Bitbucket 工作空间的成员或合作者，则会看到一个对话框，询问您要在哪里安装扩展程序；如果您只属于一个 Bitbucket 工作空间，则会看到亚马逊 CodeCatalyst 应用程序的配置页面。为要允许的工作区访问权限配置应用程序，然后选择授予访问权限。如果授予访问权限按钮未激活，请更改配置，然后重试。

将 Bitbucket 工作空间连接到后 CodeCatalyst，您将被带到 Bitbucket 存储库扩展详细信息页面，您可以在其中查看和管理已连接的 Bitbucket 工作空间和关联的 Bitbucket 存储库。

- GitLab 存储库：Connect 连接到 GitLab 用户。

  1. 选择 Connect GitLab 用户以访问外部站点 GitLab。
  2. 使用您的 GitLab 证书登录您的 GitLab 用户并查看授予的权限 CodeCatalyst。

  > **Tip**
  >
  > 如果您之前已将 GitLab 用户连接到空间，则系统不会提示您重新授权。相反，您将被导航回 CodeCatalyst 控制台。

  3. 选择 “为 AWS 连接器授权” GitLab。

  将 GitLab 用户连接到后 CodeCatalyst，系统会将您带到GitLab 存储库扩展详细信息页面，您可以在其中查看和管理关联的 GitLab 用户和关联的 GitLab 项目存储库。

将第三方来源连接到后 CodeCatalyst，您可以将第三方存储库链接到您的 CodeCatalyst 项目。

创建您的项目

1. 在创建项目页面上，选择您关联的 GitHub 帐户。
2. 根据您连接的第三方存储库提供商，选择GitHub 存储库、Bitbuck et 存储库或GitLab 存储库存储库下拉菜单以查看第三方存储库，然后选择要链接到项目的存储库。
3. 在命名项目文本输入字段中，输入要分配给项目的名称。该名称在空间内必须是唯一的。
4. 选择创建项目。

安装GitHub 存储库、Bitbuck et GitLab 存储库或存储库扩展、连接资源提供商并将第三方存储库与 CodeCatalyst 项目关联后，您可以在 CodeCatalyst 工作流程和开发环境中使用它。您还可以在关联的 GitHub 账户、Bitbucket 工作区或 GitLab 使用蓝图生成的代码的用户中创建第三方存储库。您还可以将已链接存储库与 Amazon Q 开发者版、蓝图等结合使用。有关更多信息，请参阅在第三方存储库事件发生后自动启动工作流运行和创建开发环境。

使用蓝图创建项目

您可以使用项目蓝图预置所有项目资源和示例代码。有关蓝图的信息，请参阅使用 CodeCatalyst 蓝图创建综合项目。

使用蓝图创建项目

1. 在 CodeCatalyst 控制台中，导航到要在其中创建项目的空间。
2. 在空间控制面板上，选择创建项目。
3. 选择从蓝图开始。

> **Tip**
>
> 您可以选择向 Amazon Q 提供您的项目需求，让 Amazon Q 为您推荐蓝图，以此来添加蓝图。有关更多信息，请参阅创建项目或添加功能时使用 Amazon Q 选择蓝图和使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践。此功能仅在美国西部（俄勒冈州）区域中提供。
> 此功能要求为空间启用生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。

4. 在CodeCatalyst 蓝图或空间蓝图选项卡中，选择蓝图，然后选择下一步。
5. 在命名项目下，输入要分配给项目的名称及其关联资源名称。该名称在空间内必须是唯一的。
6. （可选）默认情况下，蓝图创建的源代码存储在存储 CodeCatalyst 库中。此外，您可以选择将蓝图源代码存储在第三方存储库中。有关更多信息，请参阅 为带有扩展程序的项目添加功能 CodeCatalyst。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

根据要使用的第三方存储库提供商，执行以下操作之一：

- GitHub 存储库：Connect GitHub 账户。

  选择 “高级” 下拉菜单，选择 GitHub作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitHub 帐户。

  > **Note**
  >
  > 如果您要关联 GitHub 账户，则必须创建个人连接才能在您的身份和身份之间建立 CodeCatalyst 身份映射。 GitHub 有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

- Bitbucket 存储库：连接 Bitbucket 工作区。

  选择高级下拉菜单，选择 Bitbucket 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 Bitbucket 工作区。

- GitLab 存储库：Connect GitLab 用户。

  选择 “高级” 下拉菜单，选择 GitLab作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitLab 用户。

7. 在项目资源下，配置蓝图参数。根据蓝图的不同，您可以选择命名源存储库名称。
8. （可选）要根据您选择的项目参数查看包含更新的定义文件，请从生成项目预览中选择查看代码或查看工作流。
9. （可选）从蓝图卡中选择查看详细信息，查看蓝图的具体详细信息，如蓝图架构概述、所需连接和权限，以及蓝图创建的资源种类。
10. 选择创建项目。

使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践

当您创建项目或想要向现有项目添加新组件时，您可能不确定要使用哪个蓝图或如何集成功能。CodeCatalyst 包括与名为 Amazon Q 的生成式 AI 助手的集成，该助手可以分析您的项目需求并提出最适合您需求的蓝图。

您可以使用 Amazon Q 来帮助您创建带蓝图的项目，该蓝图可以根据您的要求创建组件，或者您可以使用 Amazon Q 来帮助您将蓝图添加到现有项目中。例如，要将适用于 Web 应用程序或现代应用程序的资源添加到项目，请指定您的要求，之后这些资源将与建议的蓝图一起添加。可以为您创建其余组件的事务。

Amazon Q 还会针对建议的蓝图无法满足的要求创建事务。此外，您可以将这些事务分配给 Amazon Q。如果您将事务分配给 Amazon Q，它将尝试创建解决方案草稿以供您评估。这有助于您和您的团队专注于并优化处理需要关注的事务，而 Amazon Q 则专注于解决您没有资源来立即解决的问题。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

以下是一些最佳实践，可帮助您使用 Amazon Q 创建项目和添加蓝图。

> **Important**
>
> 生成式人工智能功能仅在美国西部（俄勒冈州）区域中可用。

- 使用 Amazon Q 提供的默认提示。Amazon Q 非常适合根据提供的提示选择蓝图。
- 使用 Amazon Q 建议的配置选项来预览蓝图。选择一个蓝图以预览该蓝图将创建的示例代码和资源。
- 使用为 Amazon Q 启用的空间。要使用 Amazon Q 创建项目，或使用 Amazon Q 向带蓝图的项目添加功能，请使用为生成式人工智能功能启用的空间。有关更多信息，请参阅 Enabling or disabling generative AI features for a space。
- 获取有关 Amazon Q 建议的蓝图的更多信息。您可能需要详细了解使用建议的特定蓝图创建的项目资源、示例代码和组件的类型。有关中可用蓝图的更多信息 CodeCatalyst，请参阅使用 CodeCatalyst 蓝图创建综合项目。
- 允许 Amazon Q 处理事务。允许 Amazon Q 为您创建事务、分配这些事务并对其进行跟踪。有关更多信息，请参阅 教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作。

- 取消向 Amazon Q 分配不再处理的事务。完成示例后，取消向 Amazon Q 分配不再处理的任何事务。如果 Amazon Q 已完成对某个事务的处理或找不到解决方案，请务必取消分配 Amazon Q，以免达到生成式人工智能功能的最大配额。有关更多信息，请参阅 Managing generative AI features 和 Pricing。
- 查看 Amazon Q 的使用情况。您可以在用户级别查看生成式人工智能功能的使用情况。转到我的设置以管理生成式人工智能配额，并按您的构建者 ID 或单点登录（SSO）身份查看使用情况。有关更多信息，请参阅 Viewing usage of generative AI features in a space。

> **Important**
>
> 中的生成式 AI 功能 CodeCatalyst 受配额限制。有关更多信息，请参阅 Amazon Q 开发者版定价、Enabling or disabling generative AI features for a space 和 Billing。

将蓝图与项目结合使用的最佳实践

以下是一些最佳实践，可帮助您使用蓝图创建项目或添加蓝图。

- 使用提供的 CodeCatalyst 蓝图创建项目或将其添加到项目中。您可以使用蓝图为开发人员创建包含源代码和资源的完整项目。例如，Web 应用程序蓝图创建应用程序和基础设施资源并部署 Web 应用程序。您可以使用蓝图创建项目，或将自定义蓝图添加到现有项目中。有关更多信息，请参阅 使用蓝图创建项目。在中 CodeCatalyst 查看任何蓝图，预览蓝图将创建的示例代码和资源。
- 使用您的组织设计的自定义蓝图。您可以使用自定义蓝图在空间中创建完整项目。您的组织设计的自定义蓝图可以提供标准化和最佳实践，这也有助于减少设置新项目所需的工作量。作为自定义蓝图作者，您可以详细了解哪些项目正在空间中使用您的蓝图。利用生命周期管理功能，您可以集中管理每个项目的软件开发生命周期，并且蓝图用户可以利用生命周期管理功能从蓝图的更新选项或版本中重新生成代码库。有关更多信息，请参阅 以蓝图作者的身份使用生命周期管理功能。
- 将开发人员角色或相应的 IAM 角色添加到您项目的账户。在完成项目创建步骤期间或之后，您可以通过在已连接到空间的 AWS 账户 中选择或创建 IAM 角色来配置蓝图权限。

向已创建的项目添加资源和任务

在项目准备就绪后，您可以添加资源和任务。

- 要了解使用您的项目创建 CI/CD 的工作流程，请参阅入门工作流。

- 要使用与新项目中的构建操作（这些操作将构建构件部署到 Amazon S3 存储桶），请参阅使用工作流进行构建和教程：将构件上传到 Amazon S3。
- 要从一个空项目开始，然后使用 CloudFormation 堆栈部署来部署类似的无服务器应用程序，请参阅教程：部署无服务器应用程序。
- 要添加事务规划面板，请参阅跟踪和组织处理问题的工作 CodeCatalyst。
- 要查看项目概述、项目状态、最近的团队活动和分配的工作，请参阅获取项目列表。
- 要查看源代码或创建拉取请求，请参阅使用源存储库存储代码并协作处理代码 CodeCatalyst。
- 要设置发送工作流运行成功或失败的状态提醒的通知，请参阅从 CodeCatalyst 发送 Slack 和电子邮件通知。
- 要邀请成员加入您的项目，请参阅向用户授予项目权限。
- 要设置开发环境，请参阅使用开发环境编写和修改代码 CodeCatalyst。

获取项目列表

在您的 CodeCatalyst 空间中，您可以查看您拥有项目权限的每个项目的详细信息。

要查看一个项目，您必须是该项目的成员，或者拥有空间的空间管理员角色。

如果您尚未创建项目，请参阅创建项目。您必须具有要在其中创建项目的空间的空间管理员角色。

- 在项目概述中，您可以查看项目成员、源存储库、工作流运行、打开拉取请求、项目开发环境和事务。
- 在项目设置下，您可以查看和管理项目详细信息、删除项目、邀请新成员加入项目、管理项目成员以及配置通知。

查看项目任务和开发环境

要查看项目任务（例如分配给您或由您创建的未解决事务和拉取请求，以及项目的关联开发环境）的摘要，请使用控制台。

要查看一个项目，您必须是该项目的成员，或者拥有空间的空间管理员角色。

查看您的源存储库、工作流运行、事务、拉取请求和开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择概述。
4. 查看分配给您以及由您创建的项目任务。

   - 查看成员 + 查看全部列表以查看项目成员列表。
   - 查看存储库卡以查看与项目关联的源存储库。
   - 查看工作流运行卡以查看与项目关联的工作流。
   - 查看打开拉取请求卡以查看代码存储库状态摘要，以及分配给您和由您创建的拉取请求。
   - 查看我的开发环境卡以查看与项目关联的开发环境的摘要。
   - 查看事务卡以查看已分配的任务或您创建的任务的摘要。

查看空间中的所有项目

在空间的项目列表中，您可以查看您拥有权限的所有项目。

要查看项目任务（例如分配给您或由您创建的未解决事务和拉取请求，以及项目的关联开发环境）的摘要，请使用控制台。

要查看一个项目，您必须是该项目的成员，或者拥有空间的空间管理员角色。

查看您的源存储库、工作流运行、事务、拉取请求和开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 查看项目名称、路径、项目 ID 和描述。

查看项目设置

在项目设置中，您可以查看项目成员、源存储库、工作流运行、打开拉取请求、项目开发环境和事务。

要查看项目任务（例如分配给您或由您创建的未解决事务和拉取请求，以及项目的关联开发环境）的摘要，请使用控制台。

查看您的源存储库、工作流运行、事务、拉取请求和开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 查看项目名称、路径、项目 ID 和描述。

在中更改为其他项目 CodeCatalyst

要更改为其他项目，请使用控制台从您有权访问的项目的列表中进行选择。

更改为其他项目

1. 在 CodeCatalyst 控制台中，选择顶部的项目选择器。
2. 展开下拉列表，然后选择要导航到的项目。

删除项目

您可以删除项目以移除对该项目的资源的所有访问权限。您必须拥有空间管理员或项目管理员角色才能删除项目。在删除项目后，项目成员将无法访问项目资源，并且第三方源存储库所触发的任何工作流都将停止。

删除项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 选择删除项目。
5. 输入 **`delete`** 以确认删除。
6. 选择删除项目。

向用户授予项目权限

您可以使用 Amazon CodeCatalyst 控制台管理项目中的成员。您可以添加或移除用户、管理当前成员的角色、发送项目加入邀请以及取消尚未接受的邀请。

在空间和项目用户的成员页面上，用户可以拥有多个角色。拥有多个角色的用户在拥有多个角色时将显示一个指示器，并且首先会显示其中权限最多的角色。

获取成员及其项目角色的列表

在向项目添加用户时，您可以分配一个角色来授予项目权限，如下所示：

- 项目管理员角色拥有项目中的所有权限。仅将此角色分配给需要管理项目的各个方面（包括编辑项目设置、管理项目权限和删除项目）的用户。有关更多信息，请参阅 项目管理员角色。
- 贡献者角色拥有在项目中工作所需的权限。将此角色分配给那些需要在项目中处理代码、工作流、事务和操作的用户。有关更多信息，请参阅 贡献者角色。
- 审阅者角色拥有审阅权限。有关更多信息，请参阅 审阅者角色。
- 只读角色拥有读取权限。有关更多信息，请参阅 只读角色。

您无需邀请具有空间管理员角色的用户加入您的项目，因为他们具有对空间中的所有项目的隐式访问权限。

当您邀请一个用户加入您的项目（未分配空间管理员角色）时，该用户将显示在项目下的项目成员表中和空间下的项目成员表中。

查看空间中的用户和角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 选择成员选项卡。

   项目成员表显示项目中具有角色的所有成员。

   > **Tip**
   >
   > 如果您拥有空间管理员角色，则可以查看您直接受邀参与的项目。导航到项目的项目设置，然后选择我的项目。

   空间管理员表显示了具有空间管理员角色的用户。这些用户将被自动（隐式）分配给空间中的所有项目，并且他们在项目中没有角色。

   在状态列中，以下为有效值：

   - 已@@ 邀请 — 已 CodeCatalyst 发送邀请，但用户尚未接受或拒绝。

- 成员 - 用户接受了邀请。

主题

邀请用户加入项目

您可以使用控制台邀请用户加入您的项目。您可以邀请空间成员或添加空间之外的名称。

要邀请用户加入您的项目，您必须使用项目管理员或空间管理员角色登录。

您无需邀请具有空间管理员角色的用户加入您的项目，因为他们具有对空间中的所有项目的隐式访问权限。

当您邀请一个用户加入您的项目（未分配空间管理员角色）时，该用户将显示在项目下的项目成员表中和空间下的项目成员表中。

从“项目设置”选项卡邀请成员加入您的项目

1. 导航到您的项目。

   > **Tip**
   >
   > 您可以在顶部导航栏中选择要查看的项目。

2. 在导航窗格中，选择项目设置。
3. 选择成员选项卡。
4. 在项目成员中，选择邀请新成员。
5. 键入新成员的电子邮件地址，选择该成员的角色，然后选择邀请。有关角色的更多信息，请参阅使用用户角色授予访问权限。

从项目概述页面邀请成员加入您的项目

1. 导航到您的项目。

> **Tip**
>
> 您可以在顶部导航栏中选择要查看的项目。

2. 选择成员 + 按钮。
3. 键入新成员的电子邮件地址，选择该成员的角色，然后选择邀请。有关角色的更多信息，请参阅使用用户角色授予访问权限。

取消邀请

如果您最近发送了一个邀请，只要该邀请未被接受，便可将其取消。

要管理项目邀请，您必须具有项目管理员角色或空间管理员角色。

取消项目成员邀请

1. 导航到要在其中取消已发送的邀请的项目。
2. 在导航窗格中，选择项目设置。
3. 查看成员选项卡，并验证成员的状态是否为已邀请。

> **Note**
>
> 您只能取消尚未接受的邀请。

4. 选择受邀成员所在行旁边的选项，然后选择取消邀请。
5. 此时将显示一个确认窗口。选择取消邀请进行确认。

从项目中移除用户

您可以使用控制台从项目中移除用户。

要从项目中移除用户，您必须使用项目管理员或空间管理员角色登录。

> **Note**
>
> 从空间内的所有项目中移除用户会自动将该用户从空间中移除。

从项目中移除用户

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到具有要查看的项目的空间。在项目下，选择您的项目。
3. 在导航窗格中，选择项目设置。
4. 选择成员选项卡。
5. 选择要移除的配置文件用户旁边的选择器，然后选择移除。
6. 确认要移除用户，然后选择移除。

接受或拒绝项目邀请

您可能会收到一封电子邮件邀请，邀请您加入 Amazon CodeCatalyst 项目。您可以接受或拒绝邀请。

接受或拒绝邀请

1. 打开邀请电子邮件。
2. 在电子邮件中选择项目链接。
3. 选择接受或拒绝。

   如果您选择拒绝，则会向项目管理账户发送一封电子邮件，告知其您已拒绝邀请。

允许使用团队访问项目

创建项目后，您可以添加团队。团队允许您对用户进行分组，以便他们可以共享权限，并以项目和空间成员的 CodeCatalyst 身份管理项目、问题跟踪、角色和资源。

您必须拥有项目管理员角色才能管理项目团队。

团队也在空间层面进行管理 CodeCatalyst。要了解有关空间中的团队的更多信息，请参阅允许使用团队访问空间。

主题

- 向项目添加团队
- 为团队授予项目角色
- 移除团队的项目角色

向项目添加团队

您可以管理团队，该团队中的成员可以访问项目中的资源。

在空间和项目用户的成员页面上，用户可以拥有多个角色。拥有多个角色的用户在拥有多个角色时将显示一个指示器，并且首先会显示其中权限最多的角色。

添加团队

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。选择设置，然后选择团队。
3. 选择添加团队。
4. 在团队中，从可用团队列表中选择一个团队。
5. 在 “项目角色” 下，从中可用的项目角色列表中选择一个角色 CodeCatalyst。
   - 项目管理员 – 有关详细信息，请参阅项目管理员角色。
   - 贡献者 – 有关详细信息，请参阅贡献者角色。
   - 审阅者 – 有关详细信息，请参阅审阅者角色。
   - 只读 – 有关详细信息，请参阅只读角色。
6. 选择添加团队。

为团队授予项目角色

团队可以在空间中拥有角色权限，例如高级用户。您可以更改团队的空间角色，但请注意，团队的所有成员都将继承这些权限。

添加或更改项目角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 要更改角色，请选择此列表中团队旁边的选择器，然后选择更改角色。要添加角色，请选择添加项目角色。在项目中，选择要添加的项目，然后在角色中选择角色。选择一个可用的项目角色：
   - 项目管理员 – 有关详细信息，请参阅项目管理员角色。
   - 贡献者 – 有关详细信息，请参阅贡献者角色。
   - 审阅者 – 有关详细信息，请参阅审阅者角色。

- 只读 – 有关详细信息，请参阅只读角色。

4. 选择保存。

移除团队的项目角色

在中 CodeCatalyst，您可以查看团队的项目角色。您还可以查看团队中的成员。您可以移除团队的项目角色。

移除项目角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的空间。选择设置，然后选择团队。
3. 选择项目角色选项卡。
4. 选择要移除的角色。

> **Important**
>
> 从团队中移除一个角色会移除团队中所有用户的关联权限。

5. 选择保存。

允许机器资源的项目访问权限

计算机资源 CodeCatalystthat 是中被授予项目或空间权限的特定资源 CodeCatalyst。

> **Note**
>
> “机器资源” 一词并不指 EC2 实例等云基础设施，而是指具有空间或项目权限的蓝图或工作流程资源。

一个在项目中使用机器资源的示例包括允许蓝图资源代表您访问项目。

当 CodeCatalyst 通过 SSO 进行访问时，计算机资源代表您来自授权资源的身份。机器资源用于向项目中的资源（例如蓝图和工作流）授予权限。您可以查看项目中的机器资源，也可以选择启用或禁用项目的机器资源。例如，您可能希望禁用机器资源以管理访问权限，并在稍后重新启用机器资源。

在需要撤销或禁用机器资源的情况下，可对机器资源执行这些操作。例如，如果您怀疑凭证可能已被泄露，则可以禁用机器资源。通常，不需要使用这些操作。

您必须拥有空间管理员角色或项目管理员角色才能查看此页面并在项目级别管理机器资源。

计算机资源也在中的空间级别进行管理 CodeCatalyst。要了解有关空间/项目中的团队的更多信息，请参阅允许机器资源的空间访问权限 。

主题

查看机器资源的项目访问权限

您可以查看正在项目中使用的机器资源的列表。

您必须拥有空间管理员角色或项目管理员角色。

查看机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目，然后选择项目设置。选择机器资源。
3. 在下拉列表中，选择工作流操作以仅查看工作流的机器资源。选择蓝图以仅查看蓝图的机器资源。

   您也可以使用筛选条件字段对名称进行筛选。

禁用机器资源的项目访问权限

您可以选择禁用正在项目中使用的机器资源。

> **Important**
>
> 禁用机器资源将移除对空间中所有关联的蓝图或工作流的所有权限。

您必须拥有空间管理员角色或项目管理员角色。

禁用机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目，然后选择项目设置。选择机器资源。
3. 选择下列选项之一。

> **Important**
> 禁用机器资源将移除对空间中所有关联的蓝图或工作流的所有权限。

- 要单独禁用，请选择要禁用的一个或多个机器资源旁边的选择器。选择禁用，然后选择此资源。
- 要禁用所有资源，请选择禁用，然后选择所有资源。
- 要禁用所有工作流操作，请选择禁用，然后选择所有工作流操作。
- 要禁用所有蓝图，请选择禁用，然后选择所有蓝图。

启用机器资源的项目访问权限

您可以选择启用正在项目中使用的和已禁用的机器资源。

您必须拥有空间管理员角色或项目管理员角色。

启用机器资源

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目，然后选择项目设置。选择机器资源。
3. 选择下列选项之一。

- 要单独启用，请选择要启用的一个或多个机器资源旁边的选择器。选择启用，然后选择此资源。
- 要启用所有资源，请选择启用，然后选择所有资源。
- 要启用所有工作流操作，请选择启用，然后选择所有工作流操作。
- 要启用所有蓝图，请选择启用，然后选择所有蓝图。

项目的配额

下表描述了 Amazon 中项目的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

每个空间的最大项目数	100
一个用户可以属于的项目的最大数量	1000
可以属于一个项目的成员的最大数量。	10000
项目名称	一个空间中的项目名称必须唯一。名称长度必须介于 3 到 63 个字符之间。名称区分大小写。项目名称必须以字母数字字符开头。有效字符：A-Z、a-z、0-9、空格、.、,、_（下划线）和 -（连字符） 项目名称不能包含以下任何字符：! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
项目描述	项目描述的长度最多为 200 个字符。有效字符：A-Z、a-z、0-9、空格、.、,、_（下划线）和 -（连字符）。项目描述是可选的。

发送通知来自 CodeCatalyst

您可以在中设置通知来监控您的项目和资源 CodeCatalyst。用户可以在他们所属的任何项目中选择要接收其电子邮件的项目事件。您还可以选择配置在团队消息应用程序（例如 Slack）中发送给整个团队的通知，方法是配置 CodeCatalyst 空间和 Slack 工作区之间的访问权限，然后为要发送到该 Slack 工作区中的一个或多个频道的项目配置通知。在 CodeCatalyst 空间和 Slack 工作区之间配置访问权限后，项目成员还可以选择添加自己的 Slack 成员， IDs 这样他们就可以直接收到有关互联的 Slack 工作空间和频道中的 CodeCatalyst 事件的通知。

> **Note**
>
> 可以发送到 Slack 的项目事件集与用户可以选择通过电子邮件接收通知的事件集不同。

主题

通知的工作原理是什么？

您可以将项目设置为向团队消息收发应用程序（例如 Slack）提供通知。

需要具有通知的哪些权限？

任何项目成员都可以在 CodeCatalyst 中配置、查看、更新或删除频道的通知设置。不过，只有拥有空间管理员角色的用户才能添加或删除 Slack 工作区。所有用户都可以在 CodeCatalyst 中配置他们希望接收有关他们所属项目的电子邮件的项目事件。

我可以配置有关哪些 CodeCatalyst 事件的通知？

您可以将 CodeCatalyst 配置为向一个或多个 Slack 频道发送有关工作流事件的通知。在 CodeCatalyst 项目和 Slack 之间配置通知后，项目用户可以选择添加自己的 Slack 成员 ID，以便在 Slack 频道中接收有关 CodeCatalyst 事件的私信。添加其 Slack 成员 ID 的用户将在 Slack 频道中接收为其项目配置的 ID 的直接提及，这有助于提高对其关注的事件的认识。

您还可以选择要接收有关哪些事件的电子邮件。这些电子邮件发送到为您的 AWS 构建者 ID 配置的电子邮件地址。

如何显示通知？

您可以将 CodeCatalyst 配置为向一个或多个 Slack 频道发送通知。您需要授权 CodeCatalyst 才能授予对您的 Slack 工作区的访问权限。提供授权后，CodeCatalyst 可以向您配置的 Slack 频道发送通知。如果项目成员选择添加其 Slack 成员 ID，他们可以在 Slack 频道中收到有关为该项目配置的 CodeCatalyst 事件的提及。

我如何设置通知？

电子邮件通知将配置为 CodeCatalyst 的一部分。项目用户可以在我的设置页面中选择与他们希望接收的电子邮件相关的事件。

要设置项目资源的 Slack 通知，您必须完成以下高级任务。

设置通知（高级任务）

1. 在 CodeCatalyst 中，您可以在 CodeCatalyst 和消息收发客户端（例如 Slack）之间设置连接。在连接 Slack 工作区后，该工作区将可供空间中的所有项目使用。

   > **Note**
   >
   > 只有拥有空间管理员角色的用户才能添加或删除 Slack 工作区。

2. 在您在 CodeCatalyst 中的项目中，添加频道以便团队接收通知。
3. 在 CodeCatalyst 中，您可以为各种事件（例如工作流运行失败）启用通知，并指定要将通知发送到的频道。

有关详细步骤，请参阅开始使用 Slack 通知。

在 CodeCatalyst 空间和 Slack 之间配置通知后，用户可以选择添加自己的 Slack 成员 ID，以便在 Slack 频道中接收有关为其项目配置的 CodeCatalyst 事件的私信。

开始使用 Slack 通知

创建项目后，您可以设置 Slack 通知，以帮助您的团队监控项目资源。

这些步骤将引导您首次设置Slack通知。 CodeCatalyst如果您已经配置了通知，请参阅从 CodeCatalyst 发送 Slack 和电子邮件通知。

> **Note**
>
> 可以发送到通知频道的项目事件集与用户可以选择通过电子邮件接收通知的事件集不同。有关更多信息，请参阅 从 CodeCatalyst 发送 Slack 和电子邮件通知。

主题

- 步骤 4：后续步骤

先决条件

在开始之前，您需要：

- 一个 CodeCatalyst 空间。有关创建 CodeCatalyst 空间和首次登录的信息，请参阅设置并登录 CodeCatalyst。
- 一个 CodeCatalyst 项目。有关更多信息，请参阅 创建项目。
- 具有项目管理员或空间管理员角色的 CodeCatalyst 帐户。有关更多信息，请参阅 使用用户角色授予访问权限。
- 一个可以访问的 Slack 账户和 Slack 工作空间。 CodeCatalyst
- 一个用于发送通知的 Slack 频道。 CodeCatalyst 频道可以是公有的或私有的。

第 1 步： CodeCatalyst 连接到你的 Slack 工作空间

只有拥有空间管理员角色的用户才能添加或删除 Slack 工作区。添加或删除 Slack 工作区会影响该空间中的所有项目。要在 CodeCatalyst 和 Slack 之间建立连接，请使用您的 Slack 工作区 CodeCatalyst 执行安全的 OAuth 身份验证握手。

按照以下说明 CodeCatalyst 连接到您的 Slack 工作区。

> **Note**
>
> 针对每个 Slack 工作区，只需要执行一次此操作。然后，您可以通过 Slack 频道设置通知。

要 CodeCatalyst 连接到你的 Slack 工作空间

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。
3. 在导航窗格中，选择项目设置。
4. 选择通知选项卡。
5. 选择配置通知。
6. 选择连接到 Slack 工作区。
7. 阅读对话框内容，然后选择连接到 Slack 工作区。

8. 在聊天应用程序中的 Amazon Q 开发者版消息上：

   a. 在右上角，选择包含您的频道的 Slack 工作区。

   b. 选择允许。

   您将返回到 CodeCatalyst 控制台。

9. 继续第 2 步：将你的 Slack 频道添加到 CodeCatalyst。

第 2 步：将你的 Slack 频道添加到 CodeCatalyst

你需要使用 Slack 频道 ID 才能将你的频道添加到。 CodeCatalyst

获取 Slack 频道 ID

1. 登录 Slack。有关更多信息，请参阅登录到 Slack。
2. 前往 Slack 工作区，其中包含您希望通知送达的频道。有关更多信息，请参阅在 Slack 工作区之间切换或登录其他 Slack 工作区。
3. 在导航窗格中，打开希望通知送达的频道的上下文（右键单击）菜单，然后选择打开频道详细信息。

   频道 ID 将显示在对话框底部。

4. 复制频道 ID 值。下一步中需要使用该值。

使用你刚才复制的频道 ID，你现在可以将你的 Slack 频道连接到。 CodeCatalyst

要将你的 Slack 频道添加到 CodeCatalyst

1. 在开始之前，如果您的 Slack 频道是私有频道，请按如下方式将聊天应用程序中的 Amazon Q 开发者版添加到该频道：

   a. 在 Slack 频道的消息框中，输入 **@aws** 并从对话框中选择 AWS 应用程序。

   b. 按 Enter。

   此时会出现 Slackbot 消息，表明聊天应用程序中的 Amazon Q 开发者版不在私有频道中。

   c. 选择邀请他们，邀请聊天应用程序中的 Amazon Q 开发者版访问该频道。

2. 在 CodeCatalyst 控制台中，选择 “下一步”。
3. 在频道 ID 中，粘贴您之前获得的 Slack 频道 ID。

4. 在频道名称中，输入名称。我们建议使用 Slack 频道名称。
5. 选择下一步。
6. 在选择通知事件中，选择要针对其接收通知的事件类型。
7. 选择结束。

第 3 步：测试从 Slack 发送 CodeCatalyst 的通知

将项目配置为发送工作流状态通知后，您可以在 Slack 中查看通知。

在 Slack 中查看通知

1. 在您的 CodeCatalyst 项目中，手动启动工作流程以完成工作流程运行并在运行完成时收到状态通知。
2. 在 Slack 中，查看您为通知设置的频道。您的通知会显示每次工作流运行的最新状态以及工作流是失败还是成功。

步骤 4：后续步骤

为您的 CodeCatalyst 空间配置了 Slack 工作空间后，您可以添加其他 Slack 频道现有 CodeCatalyst 项目，并在创建新项目后将其添加到新项目中。您还可以让项目用户知道他们可以为自己的 Slack 成员配置个人 Slack 通知 IDs，并配置他们将接收电子邮件的事件。有关更多信息，请参阅 从 CodeCatalyst 发送 Slack 和电子邮件通知。

从 CodeCatalyst 发送 Slack 和电子邮件通知

您可以将 CodeCatalyst 配置为发送有关项目中发生的事件的通知。CodeCatalyst 可以向消息收发客户端（例如 Slack 频道）发送通知。让 CodeCatalyst 向 Slack 频道发送消息有助于确保您的整个团队都知道重要事件，例如工作流故障。（可选）您可以选择让 CodeCatalyst 在它发送的 Slack 消息中 @mention 您，这样您就会收到相应的私信（DM）。

CodeCatalyst 还可以通过电子邮件直接向您发送通知。将发送有关任何您具有成员身份的项目中的事件的电子邮件通知。这些电子邮件将发送到在您的 AWS 构建者 ID 中配置的电子邮件地址。

> **Note**
>
> 可以发送到 Slack 频道的事件与通过电子邮件发送的事件不同。

主题

配置电子邮件通知

您可以选择接收有关任何您具有成员身份的项目中的事件的电子邮件通知。这些电子邮件将发送到在您的 AWS 构建者 ID 中配置的电子邮件地址。默认情况下，您将收到有关所有可发送电子邮件的项目事件的电子邮件。

为项目事件配置电子邮件通知

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。这将打开 CodeCatalyst 我的设置页面。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在电子邮件通知中，在列表中找到要配置电子邮件通知的项目，然后选择编辑。
4. 选择要接收其电子邮件的事件，然后选择保存。

向 Slack 频道发送通知

您可以将 CodeCatalyst 配置为向团队的 Slack 频道发送有关项目事件的通知。通过执行此操作，您可以帮助确保整个团队了解重要事件，例如工作流运行失败。

> **Note**
>
> 项目的任何成员都可以管理发送到该项目的频道的通知。不过，只有拥有空间管理员角色的用户才能添加或删除 Slack 工作区。

按照以下说明操作，添加将向其发送通知的 Slack 频道。

为通知添加 Slack 频道

1. 如果您添加的是第一个 Slack 频道，请改为参阅开始使用 Slack 通知。

   设置第一个频道后，请返回此过程以设置其他频道。
2. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
3. 导航到您的项目。
4. 在导航窗格中，选择项目设置。
5. 选择通知选项卡。
6. 选择添加频道。
7. 选择 选择工作区，然后选择包含要向其发送通知的频道的 Slack 工作区。

   如果您的 Slack 工作区不在列表中，则可以按照开始使用 Slack 通知中的说明操作来添加它。
8. 在输入频道 ID 之前，如果要添加的 Slack 频道是私人频道，请完成以下步骤：

   a. 在 Slack 频道的消息框中，输入 **@aws** 并从弹出窗口中选择 AWS 应用程序。

   b. 按 Enter。

      此时会出现 Slackbot 消息，表明聊天应用程序中的 Amazon Q 开发者版不在私有频道中。

   c. 选择邀请他们，邀请聊天应用程序中的 Amazon Q 开发者版访问该频道。
9. 在 CodeCatalyst 的频道 ID 字段中，输入 Slack 频道 ID。要查找 ID，请转到 Slack，再在导航窗格中右键单击频道，然后选择打开频道详细信息。

   频道 ID 将显示在对话框底部。
10. 在频道名称中，输入名称。我们建议使用 Slack 频道名称。
11. 在选择通知事件中，选择要针对其接收通知的事件类型。
12. 选择添加。

配置 Slack 私信

如果已将 CodeCatalyst 项目配置为向 Slack 频道发送通知，则也可以私信（DM）形式发送这些通知。通过以 DM 的形式将通知直接发送给您，有助于提高您对拥有角色的项目中发生的事件的认识。要启用 DM，您必须将您的 Slack 成员 ID 添加到 CodeCatalyst。

配置 Slack 私信

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。这将打开 CodeCatalyst 我的设置页面。

> **Tip**
>
> 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在个人 Slack 通知中，选择连接 Slack ID，然后选择连接到 Slack 工作区。这将打开一个单独的窗口。

> **Tip**
>
> 除非具有空间管理员角色的用户为您的 CodeCatalyst 空间添加了 Slack 工作区，否则无法配置此选项。有关更多信息，请参阅开始使用 Slack 通知和向 Slack 频道发送通知。

4. 在权限请求窗口中，确保工作区的名称与为 CodeCatalyst 空间配置的 Slack 工作区匹配。选择允许以允许聊天应用程序中的 Amazon Q 开发者版访问工作区。此窗口将关闭，并且 Slack 工作区会将连接状态显示为已连接。

> **Tip**
>
> 如果连接状态未发生更改，请检查连接 Slack 工作区时是否出错。您可能需要向上滚动以查看错误。

5. 要停止接收个人 Slack 通知，请选择已连接的 Slack 工作区，然后选择断开连接 Slack ID。

编辑通知频道的通知

您可以更改通知将转至的频道，也可以全局关闭特定通知。

编辑通知

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的项目。

3. 在导航窗格中，选择项目设置。
4. 选择通知选项卡。
5. 选择编辑通知。
6. 请执行以下操作之一：
   - 要向特定频道发送通知，请从下拉列表中选择该频道。
   - 要全局关闭通知，请选择通知旁边的开关。
   - 要停止向特定频道发送通知，请选择该频道上的 X。
7. 选择保存。

移除频道

您可以从 Amazon CodeCatalyst 中移除 Slack 频道。通过移除 Slack 频道，有关所选 CodeCatalyst 项目的通知将不再发送到该频道。

移除频道

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的项目。在导航窗格中，选择项目设置。
3. 在项目设置页面上，选择通知选项卡。
4. 选择要移除的频道旁边的指示器，然后选择移除频道。在确认窗口中，选择确定。

使用蓝图设置 CodeCatalyst 项目

蓝图是代表 CodeCatalyst 项目架构组件的任意代码生成器。从单个文件中的工作流到包含示例代码的整个项目，该组件可以由任何内容组成。蓝图采用一组任意选项，并使用这些选项生成一组任意输出代码，这些代码然后转发到项目中。在使用最新的最佳实践或新选项更新蓝图时，在包含该蓝图的项目中会重新生成代码库的相关部分。

您可以使用 Amazon CodeCatalyst 蓝图创建包含源存储库、示例源代码、 CI/CD 工作流程、构建和测试报告以及集成问题跟踪工具的完整项目。 CodeCatalyst 蓝图根据设置的配置参数生成资源和源代码。使用 CodeCatalyst托管蓝图时，您选择的蓝图决定将哪些资源添加到您的项目中，以及用于 CodeCatalyst 创建或配置的工具，因此您可以跟踪和使用项目资源。作为蓝图用户，您可以使用蓝图创建项目或将其添加到现有 CodeCatalyst 项目中。您可以在项目中添加多个蓝图，每个蓝图都可以作为一个独立的组件来应用。例如，您可以具有使用 Web 应用程序蓝图创建的项目，以后再添加安全性蓝图。某个蓝图有更新时，您可以通过生命周期管理，将更改或修复合并到您的项目中。有关更多信息，请参阅使用 CodeCatalyst 蓝图创建综合项目和以蓝图用户的身份使用生命周期管理功能。

作为蓝图作者，您还可以创建和发布自定义蓝图，供 CodeCatalyst 空间成员使用您的项目资源。您可以开发自定义蓝图，来满足空间项目的特定需求。将自定义蓝图添加到空间的蓝图目录后，您可以管理蓝图并继续更新，从而使空间的项目与最新的最佳实践保持同步。有关更多信息，请参阅 使用自定义蓝图对项目进行标准化 CodeCatalyst。要查看蓝图 SDK 和示例蓝图，请参阅开源 GitHub 存储库。

您可能已实施标准化和最佳实践。您可以选择将包含源代码的现有源存储库转换为自定义蓝图，而不必从头开始创建和开发自定义蓝图。有关更多信息，请参阅 将源存储库转换为自定义蓝图。

主题

使用蓝图创建项目

您可以使用 Amazon 蓝图目录中的蓝图或带有自定义 CodeCatalyst 蓝图的团队空间目录中的蓝图快速创建项目。将根据蓝图使用特定资源创建您的项目。在创建新项目或者向现有项目添加组件时，您还可以与生成式人工智能助手 Amazon Q 协作。您可以在类似聊天的界面中与 Amazon Q 进行交互，向其提供项目要求。根据您的要求，Amazon Q 会推荐蓝图并概述无法满足的要求。然后，如果您对此感到满意，则可根据 Amazon Q 的建议继续执行操作，该工具将创建必要的资源，例如包含满足您要求的代码的源存储库。有关更多信息，请参阅使用蓝图创建项目、使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践和使用 CodeCatalyst 蓝图创建综合项目。

创建项目后，您可以使用自定义蓝图从 CodeCatalyst 目录或空间目录中向 CodeCatalyst 项目添加其他蓝图。蓝图代表架构组件，因此可以在您的项目中同时使用多个蓝图，以便整合您团队的最佳实践。这还使您能够确保您的项目与不断演变的组件的最新更改保持同步。要了解有关在项目中使用蓝图的更多信息，请参阅以蓝图用户的身份使用生命周期管理功能。

在项目中添加蓝图以整合资源

您可以在项目中添加多个蓝图，从而整合功能组件、资源和监管措施。项目支持在单独的蓝图中分开管理的各种元素。向项目中添加蓝图后，可以减少手动创建资源以及支持软件组件正常运行的需求。随着需求的演变，您的项目也可以保持紧跟最新变化。要了解有关如何在项目中添加蓝图的更多信息，请参阅以蓝图用户的身份使用生命周期管理功能。

在配置蓝图详细信息时，您还可以选择将蓝图的源代码存储在首选的第三方存储库中，而您仍可以在该存储库中管理蓝图，并利用生命周期管理功能让项目保持随时更新。有关更多信息，请参阅为带有扩展程序的项目添加功能 CodeCatalyst和以蓝图用户的身份使用生命周期管理功能。

> **Important**
>
> 要在 CodeCatalyst 项目中添加蓝图，您必须使用在空间中具有空间管理员、高级用户或项目管理员角色的帐户登录。

> **Tip**
>
> 将蓝图添加到项目中后，您可以配置电子邮件和 Slack 通知，从而提供有关蓝图最新更改的更新信息。有关更多信息，请参阅 发送通知来自 CodeCatalyst。

向项目添加蓝图

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到空间，然后选择要在其中添加蓝图的项目。
3. 在导航窗格中，选择蓝图，然后选择添加蓝图。

> **Tip**
>
> 您可以选择向 Amazon Q 提供您的项目需求，让 Amazon Q 为您推荐蓝图，以此来添加蓝图。有关更多信息，请参阅创建项目或添加功能时使用 Amazon Q 选择蓝图和使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践。此功能仅在美国西部（俄勒冈州）区域中提供。
> 此功能要求为空间启用生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。

4. 从蓝图选项卡中选择CodeCatalyst 蓝图或从空间蓝图选项卡中选择自定义蓝图，然后选择下一步。
5. 在蓝图详细信息下，从目标版本下拉菜单中选择蓝图版本。系统会自动选择最新的目录版本。
6. （可选）默认情况下，蓝图创建的源代码存储在存储 CodeCatalyst 库中。此外，您可以选择将蓝图源代码存储在第三方存储库中。有关更多信息，请参阅 为带有扩展程序的项目添加功能 CodeCatalyst。

   根据要使用的第三方存储库提供商，执行以下操作之一：

   - GitHub 存储库：Connect GitHub 账户。

     选择 “高级” 下拉菜单，选择 GitHub作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitHub 帐户。

     > **Note**
     >
     > 如果您使用的是 GitHub 账户连接，则必须创建个人连接才能在您的身份和身份之间建立 CodeCatalyst 身份映射。 GitHub 有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

   - Bitbucket 存储库：连接 Bitbucket 工作区。

选择高级下拉菜单，选择 Bitbucket 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 Bitbucket 工作区。

- GitLab 存储库：Connect GitLab 用户。

  选择 “高级” 下拉菜单，选择 GitLab作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitLab 用户。

7. 在配置蓝图下，配置蓝图参数。根据具体的蓝图，您可能能够选择命名源存储库。
8. 查看当前蓝图版本与更新版本之间的差异。拉取请求中显示的差异说明了当前版本与最新版本之间的更改，最新版本是创建拉取请求时所需的版本。如果未显示任何更改，则说明版本可能相同，或者您可能为当前版本和所需版本选择了相同的版本。
9. 在您确定拉取请求中包含了所要审核的代码和更改之后，请选择添加蓝图。创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

> **Note**
>
> 在拉取请求获得批准并且合并之前，不会应用蓝图。有关更多信息，请参阅审核拉取请求和合并拉取请求。

如果对于指定空间，没有蓝图可用来创建新项目或添加到现有项目中，蓝图作者还可以将自定义蓝图添加到这些空间内的项目中。有关更多信息，请参阅 在指定的空间和项目中发布和添加自定义蓝图。

如果您不希望再接收蓝图的更新，可以取消该蓝图与您的项目的关联。有关更多信息，请参阅 取消蓝图与项目的关联以停止更新。

取消蓝图与项目的关联以停止更新

如果不希望接收新的蓝图更新，可以取消该蓝图与您的项目的关联。从蓝图添加到您的项目中的资源和功能软件组件将保留在您的项目中。

> **Important**
>
> 要取消蓝图与 CodeCatalyst 项目的关联，您必须使用空间中具有空间管理员、高级用户或项目管理员角色的帐户登录。

取消蓝图与项目的关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到空间，然后选择要取消与蓝图关联的项目。
3. 在导航窗格中，选择蓝图。
4. 选择具有要取消关联的资源的蓝图，再选择操作下拉菜单，然后选择取消关联蓝图。
5. 输入 `confirm` 以确认取消关联。
6. 选择确认。

更改项目中的蓝图版本

如果您使用蓝图创建了项目或向现有项目中添加了蓝图，则会收到有关该蓝图的新版本的通知。在通过已批准的拉取请求更新蓝图版本之前，您可以查看代码更改和受影响的环境。生命周期管理使您能够更改项目中一个或多个已应用蓝图的版本，因此可以在不影响项目其它区域的情况下更改每个蓝图版本。您也可以覆盖蓝图更新。有关更多信息，请参阅 以蓝图用户的身份使用生命周期管理功能。

> **Important**
>
> 要更改 CodeCatalyst 项目中蓝图的版本，您必须使用空间中具有空间管理员、高级用户或项目管理员角色的帐户登录。

> **Tip**
>
> 将蓝图添加到项目中后，您可以配置电子邮件和 Slack 通知，从而提供有关蓝图最新更改的更新信息。有关更多信息，请参阅 发送通知来自 CodeCatalyst。

将蓝图更新到最新版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要更新蓝图版本的空间。
3. 在空间控制面板上，选择带有要更新的蓝图的项目。
4. 在导航窗格中，选择蓝图。
5. 在状态列中，选择用于更改目录版本的链接（例如，更改目录版本 0.3.109）。

6. 选择操作下拉菜单，然后选择更新版本。系统会自动选择最新版本。
7. （可选）在配置蓝图下，配置蓝图参数。
8. （可选）在代码更改选项卡中，查看当前蓝图版本和更新版本之间的差异。拉取请求中显示的差异是当前版本与最新版本之间的更改，最新版本是创建拉取请求时所需的版本。如果未显示任何更改，则说明版本可能相同，或者您可能为当前版本和所需版本选择了相同的版本。
9. 如果您对拉取请求中包含的要查看的代码和更改满意，请选择应用更新。创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

> **Note**
>
> 在拉取请求获得批准并且合并之前，不会更新蓝图。有关更多信息，请参阅审核拉取请求和合并拉取请求。

> **Note**
>
> 如果您已打开用于更新蓝图版本的拉取请求，请在创建新请求前关闭之前的拉取请求。选择更新版本后，您将看到该蓝图的待处理拉取请求列表。您还可以通过项目设置中的蓝图选项卡和项目摘要页面查看待处理的拉取请求。有关更多信息，请参阅 查看拉取请求。

更改蓝图版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要更新蓝图版本的空间。
3. 在空间控制面板上，选择带有要更新的蓝图的项目。
4. 在导航窗格中，选择蓝图，然后选择要更新的蓝图对应的单选按钮。
5. 选择操作下拉菜单，然后选择配置蓝图。
6. 从目标版本下拉菜单中选择要使用的版本。系统会自动选择最新版本。
7. （可选）在配置蓝图下，配置蓝图参数。
8. （可选）在代码更改选项卡中，查看当前蓝图版本和更新版本之间的差异。拉取请求中显示的差异是当前版本与最新版本之间的更改，最新版本是创建拉取请求时所需的版本。如果未显示任何更改，则说明版本可能相同，或者您可能为当前版本和所需版本选择了相同的版本。

9. 如果您对拉取请求中包含的要查看的代码和更改满意，请选择应用更新。创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

> **Note**
>
> 在拉取请求获得批准并且合并之前，不会更新蓝图。有关更多信息，请参阅审核拉取请求和合并拉取请求。

> **Note**
>
> 如果您已打开用于更新蓝图版本的拉取请求，请在创建新请求前关闭之前的拉取请求。选择更新版本后，您将看到该蓝图的待处理拉取请求列表。您还可以通过项目设置中的蓝图选项卡和项目摘要页面查看待处理的拉取请求。有关更多信息，请参阅 查看拉取请求。

编辑项目中的蓝图描述

您可以编辑用于创建项目或在创建项目后应用的蓝图的描述。一个蓝图可以在项目中多次使用。为了区分项目中蓝图的用途，可以对这些蓝图使用描述。描述还可用于识别从特定蓝图中添加的组件。

> **Important**
>
> 要编辑空间中自定义蓝图的描述，您必须使用在该空间中具有空间管理员、高级用户或项目管理员角色的账户登录。

编辑项目中的蓝图描述

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到您的空间，然后选择包含要更新的蓝图设置的项目。
3. 在导航窗格中，选择蓝图。
4. 选择带有要更新的描述的蓝图，选择操作下拉菜单，然后选择设置。
5. 在蓝图描述文本输入字段中，输入描述以识别项目中的蓝图。
6. 选择保存。

以蓝图用户的身份使用生命周期管理功能

利用生命周期管理功能，可以根据蓝图的更新后的选项或版本重新生成代码库。这可让蓝图作者集中管理每个包含特定蓝图的项目的软件开发生命周期。例如，通过将安全修复推送到 Web 应用程序蓝图，可让每个包含 Web 应用程序蓝图或从 Web 应用程序蓝图创建的项目自动获取该修复。此外，这个相同的管理框架可让您在选择蓝图选项后以蓝图用户的身份对其进行更改。

主题

对现有项目使用生命周期管理功能

您可以对从蓝图创建的项目或未与任何蓝图关联的现有项目使用生命周期管理功能。例如，您可以将标准安全实践蓝图添加到从未根据蓝图创建的 five-year-old Java 应用程序中。蓝图将生成安全扫描工作流和其他相关代码。现在，在对蓝图进行更改时，Java 应用程序中的代码库部分将根据团队的最佳实践自动保持最新。

对项目中的多个蓝图使用生命周期管理功能

由于蓝图代表架构组件，因此通常会在同一项目中结合使用多个蓝图。例如，一个项目可以包含由公司平台工程师构建的中央 Web API 蓝图与由应用程序安全团队构建的版本检查蓝图。所有这些蓝图均可单独更新，并且会记住过去应用于它们的合并解决方法。

> **Note**
>
> 作为任意架构组件，并非所有蓝图在结合使用时都有用或将在逻辑上进行协作，尽管它们仍会尝试相互合并。

处理生命周期拉取请求中的冲突

有时，生命周期拉取请求可能会产生合并冲突。可以手动解决这些冲突。后续蓝图更新中将记住这些解决方法。

选择退出生命周期管理变更

用户可以从项目中移除蓝图，以取消关联对蓝图的所有引用并选择退出生命周期更新。为了安全起见，这不会移除或影响项目的任何代码或资源，包括已从蓝图中添加的代码或资源。有关更多信息，请参阅取消蓝图与项目的关联以停止更新。

在项目中覆盖蓝图的生命周期管理

如果要覆盖蓝图对项目中特定文件的更新，可以在存储库中包含所有权文件。 GitLab的代码所有者规范是推荐的指导方针。蓝图始终遵循代码拥有者文件而不是任何其他文件，并且可以生成如下所示的示例代码：

```
new BlueprintOwnershipFile(sourceRepo, {
      resynthesis: {
        strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied accross all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: MergeStrategies.neverUpdate,
            globs: [
              '**/src/**',
              '**/css/**',
            ],
          },
        ],
      },
    });
```

这会生成一个包含以下内容的 `.ownership-file`：

```
[dont-override-sample-code] @amazon-codecatalyst/blueprints.import-from-git
# This strategy is applied accross all sample code. The blueprint will create sample
 code, but skip attempting to update it.
# Internal merge strategy: neverUpdate
**/src/**
**/css/**
```

使用 CodeCatalyst 蓝图创建综合项目

使用蓝图创建项目时，CodeCatalyst 会创建一个包含源存储库、示例源代码、 CI/CD 工作流程、构建和测试报告以及集成问题跟踪工具的完整项目。项目蓝图使用代码为不同类型的应用程序和框架配置云基础设施、资源和示例源构件。

有关更多信息，请参阅 创建项目。您必须是空间管理员才能创建项目。

主题

可用蓝图

蓝图名称	蓝图描述
ASP.NET Core Web API	该蓝图创建一个 .NET 6 ASP.NET Core Web API 应用程序。该蓝图使用适用于.NET 的 AWS 部署工具，并提供了将 Amazon 弹性容器服务或配置 AWS Elastic Beanstalk 为部署目标的选项。 AWS App Runner
AWS Glue ETL	该蓝图使用 AWS CDK、Glu AWS e、Lambda 和 Amazon Athena AWS 创建了一个示例数据提取转换加载 (ETL) 参考实现，将逗号分隔的值 () 转换为 Apache Parquet。CSVs
DevOps 部署管道	此蓝图使用部署管道参考架构创建 AWS 部署管道，该架构 AWS 跨多个阶段部署参考应用程序。
Java API 带有 AWS Fargate	此蓝图创建一个容器化 Web 服务项目。该项目使用 AWS Copilot CLI 在 Amazon ECS 上构建和部署由 Amazon DynamoDB 支持的容器化 Spring Boot Java Web 服务。该项目将容器化应用程序部署到无服务器计算上 AWS Fargate 的 Amazon ECS 集群。此应用程序将数据存储

蓝图名称	蓝图描述
	在 DynamoDB 表中。在工作流成功运行后，将通过应用程序负载均衡器公开提供示例 Web 服务。
现代三层 Web 应用程序	此蓝图使用 Python 为应用程序层和 Vue 前端框架生成代码，以便构建和部署架构完善的 3 层现代 Web 应用程序。
.NET 无服务器应用程序	此蓝图使用.NET CLI Lambda 工具创建 AWS Lambda 函数。蓝图为 AWS Lambda 函数提供了选项，包括选择 C# 或 F#。
Node.js API 带有 AWS Fargate	此蓝图创建一个容器化 Web 服务项目。该项目使用 AWS Copilot CLI 在 Amazon Elastic Container Service 上构建和部署容器化 Express/Node.js Web 服务。该项目将容器化应用程序部署到无服务器计算上 AWS Fargate 的 Amazon ECS 集群。在工作流成功运行后，将通过应用程序负载均衡器公开提供示例 Web 服务。
无服务器应用程序模型（SAM）	此蓝图创建一个使用无服务器应用程序模型（SAM）来创建和部署 API 的项目。你可以选择适用于 Java 的 SDK 或适用于 Python 的 SDK 作为编程语言。 TypeScript
无服务器微服务 RESTful	此蓝图创建了一个 REST API，该API使用 AWS Lambda 并 Amazon API Gateway 带有待办事项服务参考。你可以选择适用于 Java 的 SDK 或适用于 Python 的 SDK 作为编程语言。 TypeScript
单页应用程序	此蓝图创建一个使用 React、Vue 和 Angular 框架的单页应用程序（SPA）。对于托管，请从 “托 AWS Amplify 管” 或 “Amazon S3” 中 Amazon CloudFront 进行选择。

蓝图名称	蓝图描述
静态网站	此蓝图使用 Hugo 或 Jekyll 静态网站生成器创建一个静态网站。静态网站生成器使用文本输入文件（例如 Markdown）来生成静态网页。这些静态网页非常适合变化较少且内容丰富的内容，例如产品页面、文档和博客。该蓝图使用将静态网页部署 AWS CDK 到任一 AWS Amplify 或 Amazon S3 + CloudFront。
To Do Web 应用程序	此蓝图创建一个包含前端和后端组件的 To Do 无服务器 Web 应用程序。你可以选择适用于 Java 的 SDK 或适用于 Python 的 SDK 作为编程语言。 TypeScript
订阅外部蓝图	此蓝图为每个导入的程序包创建一个工作流。这些工作流每天运行一次，以检查 NPM 中是否有新版本的程序包。如果存在新版本，则工作流程会尝试将其作为自定义蓝图添加到您的 CodeCatalyst 空间。如果找不到程序包或程序包不是蓝图，则该操作将失败。目标程序包必须在 NPM 上且必须是蓝图。必须在支持自定义蓝图的等级订阅空间。
Bedrock 生成式人工智能聊天机器人	此蓝图使用 Amazon Bedrock 和 Anthropic 的 Claude 创建一个生成式人工智能聊天机器人。利用此蓝图，您可以构建和部署自己的安全、登录受保护的 LLM 操场，并且可以根据您的数据对它进行自定义。有关更多信息，请参阅 Bedrock GenAI Chatbot documentation。

查找项目蓝图信息

中 CodeCatalyst提供了多个项目蓝图。每个蓝图均附有一个摘要和自述文件。摘要描述了蓝图所安装的资源，自述文件详细介绍了蓝图并提供了有关如何使用蓝图的说明。

使用自定义蓝图对项目进行标准化 CodeCatalyst

您可以使用自定义蓝图来标准化 CodeCatalyst 空间项目的开发和最佳实践。自定义蓝图可用于定义 CodeCatalyst 项目的各个方面，例如工作流程定义和应用程序代码。在使用自定义蓝图创建新项目或者将其应用于现有项目后，对蓝图的任何更改都将作为拉取请求更新提供给这些项目。作为蓝图作者，您可以查看有关整个空间中哪些项目正在使用您的蓝图的详细信息，这样您就能够查看标准在各个项目中的应用情况。蓝图的生命周期管理功能可让您集中管理每个项目的软件开发生命周期，从而确保您空间中的项目继续遵循最佳实践以及最新的更改或修复。有关更多信息，请参阅 以蓝图作者的身份使用生命周期管理功能。

利用自定义蓝图，可以通过重新合成针对先前的项目更新蓝图版本。重新合成是使用更新的版本重新运行蓝图合成的过程，或是用于将修复和更改合并到现有项目中的功能。有关更多信息，请参阅 自定义蓝图概念。

您可能已实施标准化和最佳实践。您可以选择将包含源代码的现有源存储库转换为自定义蓝图，而不必从头开始创建和开发自定义蓝图。有关更多信息，请参阅 将源存储库转换为自定义蓝图。

要查看蓝图 SDK 和示例蓝图，请参阅开源 GitHub 存储库。

主题

自定义蓝图概念

以下是您在 CodeCatalyst 中使用自定义蓝图时需要了解的一些概念和术语。

主题

蓝图项目

利用蓝图项目，可以开发蓝图并将其发布到您的空间。源存储库是在项目创建过程中创建的，存储库的名称是您在输入项目资源详细信息时选择的名称。在蓝图创建过程中，如果您选择生成工作流版本，则会在蓝图中使用蓝图生成器蓝图创建发布工作流。该工作流会自动发布您的最新版本。

空间蓝图

导航到空间的蓝图部分时，您可以查看和管理空间蓝图表中的所有蓝图。在将蓝图发布到您的空间后，它们将作为空间蓝图提供，以便您能够在空间的蓝图目录中添加和移除它们。您还可以在空间的蓝图部分中管理发布权限和删除蓝图。有关更多信息，请参阅 查看自定义蓝图的详细信息、版本和项目。

空间蓝图目录

您可以在空间的蓝图目录中查看所有已添加的自定义蓝图。在此处，空间成员可以选择您的自定义蓝图来创建新项目。该目录与 CodeCatalyst 目录不同，后者包含对所有空间成员可用的蓝图。有关更多信息，请参阅 使用 CodeCatalyst 蓝图创建综合项目。

合成

合成是生成 CodeCatalyst 项目捆绑包的过程，该捆绑包代表项目中的源代码、配置和资源。之后，CodeCatalyst 部署 API 操作将使用该捆绑包以部署到项目中。该过程可以在开发自定义蓝图时在本地运行，以模拟项目创建过程，而无需在 CodeCatalyst 中创建项目。以下命令可用于执行合成：

```
yarn blueprint:synth            # fast mode
```

```
yarn blueprint:synth --cache     # wizard emulation mode
```

蓝图首先在 defaults.json 合并了该选项的情况下调用主 blueprint.ts 类。在 synth/synth.*[options-name]*/proposed-bundle/ 文件夹下将生成一个新的项目捆绑包。输出包括自定义蓝图根据您设置的选项生成的项目捆绑包，包括您可能已配置的部分选项。

重新合成

重新合成是使用不同的蓝图选项或现有项目的蓝图版本重新生成蓝图的过程。作为蓝图作者，您可以在自定义蓝图代码中定义自定义合并策略。您还可以在 .ownership-file 中定义所有权边界，以便指定允许在代码库的哪些部分更新蓝图。自定义蓝图可以建议对 .ownership-file 的更新，而使用自定义蓝图的项目开发人员可以确定其项目的所有权边界。您可以在本地运行重新合成，并在发布自定义蓝图之前进行测试和更新。使用以下命令可执行重新合成：

```
yarn blueprint:resynth             # fast mode
yarn blueprint:resynth --cache     # wizard emulation mode
```

蓝图首先在 defaults.json 合并了该选项的情况下调用主 blueprint.ts 类。在 synth/resynth.*[options-name]*/ 文件夹下将生成一个新的项目捆绑包。输出包括自定义蓝图根据您设置的选项生成的项目捆绑包，包括您可能已配置的部分选项。

在合成和重新合成过程完成后将创建以下内容：

- proposed-bundle – 使用目标蓝图版本的新选项运行合成时的输出。
- existing-bundle – 对现有项目的模拟。如果此文件夹中没有任何内容，则生成此文件夹时的输出与 proposed-bundle 相同。
- ancestor-bundle – 模拟使用先前版本和/或先前选项运行蓝图时生成的内容。如果此文件夹中没有任何内容，则生成此文件夹时的输出与 proposed-bundle 相同。
- resolved-bundle – 始终会重新生成捆绑包，并且该捆绑包默认为 proposed-bundle、existing-bundle 和 ancestor-bundle 之间的三向合并。此捆绑包提供了重新合成将在本地输出的内容的模拟。

要了解有关蓝图输出捆绑包的更多信息，请参阅使用重新合成功能生成文件。

部分选项

您可以在 src/wizard-configuration/ 下添加选项变体，这些变体不必枚举整个 Options 接口，而且这些选项会在 defaults.json 文件的顶部合并。这可让您跨特定选项定制测试案例。

示例：

`Options` 接口：

```
{
  language: "Python" | "Java" | "Typescript",
  repositoryName: string
  ...
}
```

`defaults.json` 文件：

```
{
  language: "Python",
  repositoryName: "Myrepo"
  ...
}
```

其他配置测试：

- ```
  #wizard-config-typescript-test.json
  {
    language: "Typescript",
  }
  ```
- ```
  #wizard-config-java-test.json
  {
    language: "Java",
  }
  ```

Projen

Projen 是一款开源工具，可供自定义蓝图用来保持最新状态和一致性。蓝图采用 Projen 程序包形式，因为此框架可让您构建、捆绑和发布项目，并且您可以使用该接口来管理项目的配置和设置。

即使在创建蓝图后，也可以使用 Projen 大规模更新蓝图。Projen 工具是生成项目捆绑包的蓝图合成背后的底层技术。Projen 拥有项目的配置，它不会影响您作为蓝图作者执行的操作。您可以在添加依赖项后运行 `yarn projen` 以重新生成项目配置，也可以更改 `projenrc.ts` 文件中的选项。Projen 还是用于合成项目的自定义蓝图的底层生成工具。有关更多信息，请参阅 projen GitHub 页。要了解有关如何使用 Projen 的更多信息，请参阅 Projen 文档。

开始使用自定义蓝图

在创建蓝图的过程中，您可以配置蓝图并生成项目资源的预览。每个自定义蓝图都由一个 CodeCatalyst 项目管理，该项目默认包含用于发布到空间蓝图目录的工作流。

在配置自定义蓝图详细信息时，您还可以选择将蓝图的源代码存储在第三方存储库中，而您仍可以在该存储库中管理自定义蓝图，并利用生命周期管理功能，在修改自定义蓝图时使空间的项目保持同步。有关更多信息，请参阅为带有扩展程序的项目添加功能 CodeCatalyst和以蓝图作者的身份使用生命周期管理功能。

如果您已经拥有具有标准化和最佳实践的源存储库，则可以选择将该源存储库转换为自定义蓝图。有关更多信息，请参阅 将源存储库转换为自定义蓝图。

主题

先决条件

在创建自定义蓝图之前，请考虑以下要求：

- 您的 CodeCatalyst 空间必须是企业级别。有关更多信息，请参阅《Amazon CodeCatalyst Administrator Guide》中的 Managing billing。
- 要创建自定义蓝图，您需要拥有空间管理员或高级用户角色。有关更多信息，请参阅 使用用户角色授予访问权限。

步骤 1：在 CodeCatalyst 中创建自定义蓝图

当您根据空间设置创建自定义蓝图时，系统会为您创建一个存储库。存储库包含在将蓝图发布到空间的蓝图目录之前，开发蓝图所必需的所有资源。

创建自定义蓝图

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。

2. 在 CodeCatalyst 控制台中，导航到要在其中创建自定义蓝图的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 选择创建蓝图。
5. 在命名您的蓝图下，输入要分配给项目的名称及其关联资源名称。该名称在空间内必须是唯一的。
6. （可选）默认情况下，蓝图创建的源代码存储在 CodeCatalyst 存储库中。此外，您可以选择将蓝图源代码存储在第三方存储库中。有关更多信息，请参阅 为带有扩展程序的项目添加功能 CodeCatalyst。

   根据要使用的第三方存储库提供商，执行以下操作之一：

   - GitHub 存储库：连接 GitHub 账户。

     选择高级下拉菜单，选择 GitHub 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 GitHub 账户。

     > **Note**
     >
     > 如果您使用与 GitHub 账户的连接，则必须创建个人连接，以便在您的 CodeCatalyst 身份与 GitHub 身份之间建立身份映射。有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

   - Bitbucket 存储库：连接 Bitbucket 工作区。

     选择高级下拉菜单，选择 Bitbucket 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 Bitbucket 工作区。

   - GitLab 存储库：连接 GitLab 用户。

     选择高级下拉菜单，选择 GitLab 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 GitLab 用户。

7. 在蓝图详细信息下，执行以下操作：

   a. 在蓝图显示名称文本输入字段中，输入一个名称，此名称将出现在空间蓝图目录中。

   b. 在描述文本输入字段中，输入自定义蓝图的描述。

   c. 在作者姓名文本输入字段中，输入自定义蓝图的作者姓名。

   d. （可选）选择高级设置。

      i. 选择 + 添加以添加要添加到 `package.json` 文件中的标签。

ii. 选择许可证下拉菜单，然后为自定义蓝图选择许可证。

iii. 在蓝图包名称文本输入字段中，输入用于标识您的蓝图包的名称。

iv. 默认情况下，发布工作流是使用项目中称为蓝图生成器的发布蓝图生成的。由于发布工作流启用了发布权限，因此当您推送更改时，工作流会将最新的蓝图版本发布到您的空间。要关闭工作流生成功能，请取消选中发布工作流复选框。

8. （可选）蓝图项目附带预定义的代码，用于支持将蓝图发布到空间的蓝图目录中。要根据您选择的项目参数查看包含更新的定义文件，请从生成蓝图预览中选择查看代码或查看工作流。

9. 选择创建蓝图。

如果您没有关闭自定义蓝图的工作流生成功能，则在创建蓝图时，工作流会自动开始运行。工作流运行完成后，默认情况下，您的自定义蓝图可以添加到空间的蓝图目录中。如果您不希望将最新的蓝图版本自动发布到您的空间，则可以关闭发布权限。有关更多信息，请参阅设置自定义蓝图的发布权限和运行工作流。

由于名为 `blueprint-release` 的发布工作流是使用蓝图创建的，因此在您的项目中，该蓝图是作为已应用的蓝图存在的。有关更多信息，请参阅在项目中添加蓝图以整合资源和取消蓝图与项目的关联以停止更新。

步骤 2：使用组件开发自定义蓝图

创建自定义蓝图时会生成蓝图向导，而开发自定义蓝图时可以使用组件对其进行修改。您可以更新 `src/blueprints.js` 和 `src/defaults.json` 文件来修改向导。

> **Important**
>
> 如果要使用来自外部来源的蓝图包，请考虑使用这些包可能带来的风险。您对添加到空间中的自定义蓝图以及这些蓝图生成的代码负责。

在配置蓝图代码之前，在您的 CodeCatalyst 项目中使用受支持的集成式开发环境（IDE）创建开发环境。开发环境是使用所需工具和软件包所必需的。

创建开发环境

1. 在导航窗格中，执行下列操作之一：

   a. 选择概述，然后导航到我的开发环境部分。

b. 选择代码，然后选择开发环境。

c. 依次选择代码、源存储库以及您在创建蓝图时创建的存储库。

2. 选择创建开发环境。
3. 从下拉菜单中选择受支持的 IDE。有关更多信息，请参阅开发环境支持的集成式开发环境。
4. 选择在现有分支中工作，然后从现有分支下拉菜单中选择您创建的功能分支。
5. （可选）在别名 – 可选文本输入字段中，输入别名以标识开发环境。
6. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动；开发环境创建完成后，状态列将显示正在运行。

有关更多信息，请参阅 使用开发环境编写和修改代码 CodeCatalyst。

开发您的自定义蓝图

1. 在工作终端中，使用以下 `yarn` 命令安装依赖项：

```
yarn
```

所需的工具和软件包可通过 CodeCatalyst 开发环境（包括 Yarn）获得。如果您在没有开发环境的情况下使用自定义蓝图，请先将 Yarn 安装到您的系统中。有关更多信息，请参阅 Yarn 安装文档。

2. 开发您的自定义蓝图，根据您的偏好进行配置。您可以通过添加组件来修改蓝图向导。有关更多信息，请参阅开发自定义蓝图以满足项目要求、使用前端向导修改蓝图功能和将自定义蓝图发布到空间。

步骤 3：预览自定义蓝图

设置和开发自定义蓝图后，您可以预览蓝图的预览版本并将其发布到您的空间。预览版使您能够在用蓝图创建新项目或者将蓝图应用于现有项目之前，检查蓝图是否符合您的需求。

预览自定义蓝图

1. 在正运行的终端中，运行以下 `yarn` 命令：

```
yarn blueprint:preview
```

2. 导航到提供的 `See this blueprint at:` 链接以预览您的自定义蓝图。

3. 根据您的配置，检查用户界面（包括文本）是否按预期显示。如果要更改自定义蓝图，可以编辑 `blueprint.ts` 文件，重新同步蓝图，然后再次发布预览版本。有关更多信息，请参阅 重新合成。

（可选）步骤 4：发布自定义蓝图预览版

如果您想将自定义蓝图的预览版添加到空间的蓝图目录中，则可以将预览版发布到您的空间。这样就可以在将非预览版本添加到目录之前，以用户身份查看蓝图。使用预览版可以在不占用实际版本的情况下进行发布。例如，如果您正在处理 `0.0.1` 版本，则可以发布和添加预览版本，这样就可以发布第二个版本的新更新并将其添加为 `0.0.2`。

发布自定义蓝图的预览版

导航到提供的 `Enable version [version number] at:` 链接以启用您的自定义蓝图。此链接是在步骤 3：预览自定义蓝图中运行 `yarn` 命令时提供的。

创建、开发、预览和发布自定义蓝图后，您可以发布最终蓝图版本，并将其添加到空间的蓝图目录中。有关更多信息，请参阅将自定义蓝图发布到空间和将自定义蓝图添加到空间蓝图目录中。

教程：创建和更新 React 应用程序

作为蓝图作者，您可以开发自定义蓝图并将这些蓝图添加到空间的蓝图目录中。然后，空间成员可以使用这些蓝图来创建新项目，或者将蓝图添加到现有项目中。您可以继续对自己的蓝图进行更改，然后通过拉取请求将这些蓝图作为更新提供。

本教程从蓝图作者和蓝图用户的角度提供了演练。本教程演示如何创建 React 单页 Web 应用程序蓝图。然后，使用该蓝图创建新的项目。使用所做的更改来更新蓝图时，根据蓝图创建的项目会通过拉取请求合并这些更改。

主题

- 步骤 8：查看项目中的更改

先决条件

要创建和更新自定义蓝图，您必须按以下方式完成设置并登录 CodeCatalyst 中的任务：

- 拥有用于登录 CodeCatalyst 的 AWS 构建者 ID。
- 属于某个空间，并且在该空间中已为您分配了空间管理员或高级用户角色。有关更多信息，请参阅创建空间、向用户授予空间权限和空间管理员角色。

步骤 1：创建自定义蓝图

当您创建自定义蓝图时，会创建一个 CodeCatalyst 项目，其中包含您的蓝图源代码以及开发工具和资源。您在自己的项目中开发、测试和发布蓝图。

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要在其中创建蓝图的空间。
3. 选择设置以导航到空间设置。
4. 在空间设置选项卡中，选择蓝图，然后选择创建蓝图。
5. 使用以下值更新蓝图创建向导中的字段：
   - 在蓝图名称中，输入 `react-app-blueprint`。
   - 在蓝图显示名称中，输入 `react-app-blueprint`。
6. 您也可以选择查看代码来预览蓝图的蓝图源代码。类似地，选择查看工作流可预览将在构建和发布蓝图的项目中创建的工作流。
7. 选择创建蓝图。
8. 完成蓝图的创建后，您将转到蓝图的项目。该项目包含蓝图源代码，以及开发、测试和发布蓝图所需的各种工具和资源。发布工作流已生成，该工作流会自动将您的蓝图发布到空间。
9. 现在，您的蓝图和蓝图项目已创建，下一步是通过更新源代码来配置蓝图。您可以使用开发环境，直接在浏览器中打开和编辑源存储库。

   在导航窗格中，选择代码，然后选择开发环境。
10. 选择创建开发环境，然后选择 AWS Cloud9（在浏览器中）。
11. 保留其余默认设置，然后选择创建。
12. 在 AWS Cloud9 终端中，运行以下命令，导航到您的蓝图项目目录：

```
cd react-app-blueprint
```

13. 创建蓝图时，会自动创建并填充 `static-assets` 文件夹。在本教程中，您将删除默认文件夹，并为 react 应用程序蓝图生成新文件夹。

    运行以下命令，删除 static-assets 文件夹：

    ```
    rm -r static-assets
    ```

    AWS Cloud9 是在基于 Linux 的平台上构建的。如果您使用 Windows 操作系统，可以改为使用下面的命令：

    ```
    rmdir /s /q static-assets
    ```

14. 现在默认文件夹已删除，请运行以下命令，为 react-app 蓝图创建一个 `static-assets` 文件夹：

    ```
    npx create-react-app static-assets
    ```

    出现提示时，请输入 y 继续。

    在 `static-assets` 文件夹中创建了一个新的 react 应用程序，其中包含必要的软件包。这些更改需要推送到您的远程 CodeCatalyst 源存储库。

15. 确保您具有最新的更改，然后运行以下命令来提交更改并推送到蓝图的 CodeCatalyst 源存储库：

    ```
    git pull
    ```

    ```
    git add .
    ```

    ```
    git commit -m "Add React app to static-assets"
    ```

    ```
    git push
    ```

将更改推送到蓝图源存储库时，发布工作流将会自动启动。此工作流会递增蓝图版本，构建蓝图并将蓝图发布到您的空间。在下一步中，您将导航到运行的发布工作流以查看执行的操作。

步骤 2：查看发布工作流

1. 在 CodeCatalyst 控制台的导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择蓝图发布工作流。
3. 您可以看到工作流包含用于构建和发布蓝图的操作。
4. 在最近的运行下，选择工作流运行链接，查看您所做的代码更改中运行的操作。
5. 运行完成后，您的新蓝图版本就会发布。已发布的蓝图版本可以在空间设置中看到，但在将蓝图添加到空间的蓝图目录中之前，不能在项目中使用。在下一步中，您将向目录中添加蓝图。

步骤 3：将蓝图添加到目录

将蓝图添加到空间的蓝图目录中后，该蓝图就可以在空间内的所有项目中使用。空间成员可以使用蓝图创建新项目，或者将蓝图添加到现有项目中。

1. 在 CodeCatalyst 控制台中，导航回该空间。
2. 选择设置，然后选择蓝图。
3. 选择 react-app-blueprint，然后选择添加到目录。
4. 选择保存。

步骤 4：使用蓝图创建项目

现在，蓝图已添加到目录中，可在项目中使用。在此步骤中，您将使用刚创建的蓝图来创建项目。在后续步骤中，您将通过更新和发布蓝图的新版本来更新此项目。

1. 选择项目选项卡，然后选择创建项目。
2. 选择空间蓝图，然后选择 react-app-blueprint。

   > **Note**
   >
   > 选择蓝图后，您可以看到蓝图的 `README.md` 文件的内容。

3. 选择下一步。
4.

   > **Note**
   >
   > 此项目创建向导的内容可以在蓝图中配置。

以蓝图用户身份输入项目名称。在本教程中，请输入 `react-app-project`。有关更多信息，请参阅 开发自定义蓝图以满足项目要求。

接下来，您将更新蓝图并将新版本添加到目录中，然后使用新版本来更新此项目。

步骤 5：更新蓝图

在使用蓝图创建新项目或者将蓝图应用于现有项目后，您可以继续以蓝图作者的身份进行更新。在此步骤中，您将对蓝图进行更改并自动向空间发布新版本。然后，新版本可以添加作为目录版本。

1. 导航到教程：创建和更新 React 应用程序中创建的 react-app-blueprint 项目。
2. 打开在教程：创建和更新 React 应用程序中创建的开发环境。
   a. 在导航窗格中，选择代码，然后选择开发环境。
   b. 从表中找到“开发环境”，然后选择在 AWS Cloud9 中打开（在浏览器中）。
3. 运行蓝图发布工作流时，会通过更新 `package.json` 文件来递增蓝图版本。通过在 AWS Cloud9 终端中运行以下命令提取更改：

```
git pull
```

4. 通过运行以下命令，导航到 `static-assets` 文件夹：

```
cd /projects/react-app-blueprint/static-assets
```

5. 通过运行以下命令，在 `static-assets` 文件夹中创建 `hello-world.txt` 文件：

```
touch hello-world.txt
```

   AWS Cloud9 是在基于 Linux 的平台上构建的。如果您使用 Windows 操作系统，可以改为使用下面的命令：

```
echo > hello-world.text
```

6. 在左侧导航栏中，双击 `hello-world.txt` 文件以在编辑器中打开文件，然后添加以下内容：

```
Hello, world!
```

保存该文件。

7. 确保您具有最新的更改，然后运行以下命令来提交更改并推送到蓝图的 CodeCatalyst 源存储库：

```
git pull
```

```
git add .
```

```
git commit -m "prettier setup"
```

```
git push
```

在推送更改时已启动了发布工作流，该工作流会自动将新版本的蓝图发布到空间。

步骤 6：将蓝图的已发布目录版本更新为新版本

在使用蓝图创建新项目或者将蓝图应用于现有项目后，您仍能以蓝图作者的身份更新该蓝图。在此步骤中，您将对蓝图进行更改，并更改蓝图的目录版本。

1. 在 CodeCatalyst 控制台中，导航回该空间。
2. 选择设置，然后选择蓝图。
3. 选择 react-app-blueprint，然后选择管理目录版本。
4. 选择新版本，然后选择保存。

步骤 7：使用新蓝图版本更新项目

新版本现已在该空间的蓝图目录中可用。作为蓝图用户，您可以更新在步骤 4：使用蓝图创建项目中创建的项目版本。这样可以确保您获得满足最佳实践所需的最新更改和修复程序。

1. 在 CodeCatalyst 控制台中，导航到在步骤 4：使用蓝图创建项目中创建的 react-app-project 项目。
2. 在导航窗格中，选择蓝图。
3. 在信息框中，选择更新蓝图。
4. 在右侧的代码更改面板中，您可以看到 `hello-world.txt` 和 `package.json` 更新。
5. 选择应用更新。

选择应用更新将在项目中创建拉取请求，其中带有更新的蓝图版本中的更改。要对项目进行更新，您必须合并拉取请求。有关更多信息，请参阅审核拉取请求和合并拉取请求。

1. 在蓝图表中，找到蓝图。在状态列中，选择待处理拉取请求，然后选择指向已打开拉取请求的链接。
2. 查看拉取请求，然后选择合并。
3. 选择快进合并以保留默认值，然后选择合并。

步骤 8：查看项目中的更改

在步骤 7：使用新蓝图版本更新项目后，对蓝图的更改现在已在您的项目中可用。作为蓝图用户，您可以查看源存储库中的更改。

1. 在导航窗格中，选择源存储库，然后选择在创建项目时创建的源存储库的名称。
2. 在文件下，您可以查看在步骤 5：更新蓝图中创建的 `hello-world.txt` 文件。
3. 选择 `hello-world.txt` 以查看内容文件。

借助生命周期管理，蓝图作者能够集中管理包含特定蓝图的每个项目的软件开发生命周期。如本教程所示，您可以推送蓝图更新，然后通过使用蓝图来创建新项目或者将蓝图应用于现有项目的项目，可以合并这些更新。有关更多信息，请参阅 以蓝图作者的身份使用生命周期管理功能。

将源存储库转换为自定义蓝图

使用自定义蓝图可以在 CodeCatalyst 空间的多个项目中整合最佳实践或新选项。您可以从头开始创建和开发新的自定义蓝图，也可以将包含代码和既定最佳实践的现有 CodeCatalyst 或第三方源存储库转换为蓝图项目。您可以避免将相关构件从该现有存储库复制到蓝图项目中。在将一个源存储库转换为一个自定义蓝图后，您可以更新、发布和添加该蓝图，就像对任何其他自定义蓝图执行这些操作一样。

将一个源存储库转换为一个自定义蓝图后，系统会将该源存储库重构为一个蓝图项目，该存储库的内容将移至存储库内的 `static-assets` 文件夹中，并且蓝图所需的相关资产将添加到该存储库中。当使用自定义蓝图创建项目或者将其添加到现有项目时，在转换后的源存储库中存储的工作流定义也会通过蓝图添加到项目中。

> **Note**
>
> 将源存储库转换为自定义蓝图时，不会包含环境和密钥等资源。将源存储库转换为自定义蓝图后，您需要手动复制或添加这些资源。

> **Important**
>
> 要将源存储库转换为自定义蓝图，您必须使用具有空间中的项目管理员、空间管理员或高级用户角色的账户进行登录。

将源存储库转换为源存储库列表中的自定义蓝图

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到空间，然后选择要在其中将源存储库转换为自定义蓝图的项目。
3. 在导航窗格中，依次选择代码、源存储库以及要转换为自定义蓝图的源存储库所对应的单选按钮。
4. 选择转换为蓝图，将您的源存储库转换为自定义蓝图。

您还可以从 CodeCatalyst 控制台中的源存储库页面中，将 CodeCatalyst 存储库转换为自定义蓝图。

从源存储库页面中将源存储库转换为自定义蓝图

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到空间，然后选择要在其中将源存储库转换为自定义蓝图的项目。
3. 在导航窗格中，依次选择代码、源存储库以及要转换为自定义蓝图的 CodeCatalyst 源存储库的名称。
4. 选择更多下拉菜单，然后选择转换为蓝图，将您的源存储库转换为自定义蓝图。

将源存储库转换为自定义蓝图后，发布工作流将立即自动运行。成功完成运行后，自定义蓝图将发布到您空间的自定义蓝图列表中。然后，您可以将转换后的自定义蓝图添加到空间目录中，以便创建新项目或者将该自定义蓝图添加到现有项目中。有关更多信息，请参阅将自定义蓝图发布到空间和将自定义蓝图添加到空间蓝图目录中。

> **Important**
>
> 要将自定义蓝图添加到空间的蓝图目录中，您必须使用具有空间管理员或高级用户角色的账户进行登录。

以蓝图作者的身份使用生命周期管理功能

生命周期管理功能可让您从单一的通用最佳实践来源使大量项目保持同步。这将扩展修复的传播，并在任意数量的项目的整个软件开发生命周期中对这些项目进行维护。生命周期管理功能简化了内部活动、安全修复、审计、运行时升级、最佳实践变更以及其他维护实践，因为这些标准是在同一位置定义的，并且会在发布新标准时自动集中保持最新。

在发布新版本的蓝图时，系统会提示所有包含该蓝图的项目更新到最新版本。作为蓝图作者，您还可以查看每个项目为实现合规性而包含的特定蓝图的版本。当现有源存储库中存在冲突时，生命周期管理功能会创建拉取请求。对于所有其他资源（例如开发环境），所有生命周期管理更新都会严格地创建新资源。用户可以自行选择合并或不合并这些拉取请求。合并待处理的拉取请求后，将更新项目中使用的蓝图的版本（包括选项）。要了解如何以蓝图用户的身份使用生命周期管理功能，请参阅对现有项目使用生命周期管理功能和对项目中的多个蓝图使用生命周期管理功能。

主题

- 测试捆绑包输出和合并冲突的生命周期管理
- 使用合并策略生成捆绑包并指定文件
- 访问上下文对象以获取项目详细信息

测试捆绑包输出和合并冲突的生命周期管理

您可以在本地测试蓝图的生命周期管理和合并冲突解决方案。在 `synth/` 目录下会生成一系列捆绑包，代表生命周期更新的各个阶段。要测试生命周期管理，您可以在蓝图上运行以下 yarn 命令：`yarn blueprint: resynth`。要了解有关重新合成和捆绑包的更多信息，请参阅重新合成和使用重新合成功能生成文件。

使用合并策略生成捆绑包并指定文件

您可以使用合并策略生成带重新合成功能的捆绑包，并指定用于自定义蓝图的生命周期管理更新的文件。通过利用重新合成功能和合并策略，您可以管理更新并控制在部署期间更新的文件。您也可以编写自己的策略来控制如何将更改合并到现有 CodeCatalyst 项目中。

主题

- 使用重新合成功能生成文件
- 使用合并策略
- 指定用于生命周期管理更新的文件
- 编写合并策略

使用重新合成功能生成文件

重新合成功能可以将蓝图生成的源代码与之前由同一蓝图生成的源代码合并，从而使对蓝图所做的更改能够传播到现有项目。通过 resynth() 函数跨蓝图输出捆绑包运行合并。重新合成功能首先会生成三个捆绑包，分别代表蓝图和项目状态的不同方面。可以使用 yarn blueprint:resynth 命令本地手动运行此功能，这将创建捆绑包（如果捆绑包不存在）。通过手动使用捆绑包，您可以在本地模拟和测试重新合成行为。默认情况下，蓝图仅在 src/* 下的存储库中运行重新合成功能，因为通常仅捆绑包的此部分受源代码控制。有关更多信息，请参阅重新合成。

- existing-bundle – 此捆绑包是现有项目状态的表示形式。这是由合成计算人为构造的，旨在提供蓝图上下文来说明它部署到的项目中的内容（如果有）。如果在本地运行重新合成功能时，此位置已存在某些内容，则它将被重置并视为模拟。否则，它将被设置为 ancestor-bundle 的内容。
- ancestor-bundle – 如果蓝图输出是使用一些早期选项和/或版本合成的，则此捆绑包代表蓝图输出。如果这是此蓝图首次被添加到项目中，且其父级不存在，它将被设置为与 existing-bundle 相同的内容。在本地，如果该位置已存在此捆绑包，则它将被视为模拟。
- proposed-bundle – 如果蓝图是使用一些新选项和/或版本合成的，则此捆绑包模拟了蓝图。这与 synth() 函数所生成的捆绑包相同。在本地，此捆绑包始终会被覆盖。

每个捆绑包都是在重新合成阶段创建的，可以从 this.context.resynthesisPhase 下的蓝图类中访问该捆绑包。

- resolved-bundle – 这是最后的捆绑包，它代表已打包并部署到 CodeCatalyst 项目的内容。您可以查看已将哪些文件和差异发送到部署机制。这是解析其他三个捆绑包之间的合并的 resynth() 函数的输出。

通过提取 ancestor-bundle 和 proposed-bundle 之间的差异，然后将该差异添加到 existing-bundle 来生成 resolved-bundle，从而应用三向合并。所有合并策略都会将文件解析为 resolved-bundle。重新合成功能在 resynth() 期间使用蓝图的合并策略来解析这些捆绑包的覆盖范围，并根据结果生成已解析的捆绑包。

使用合并策略

您可以使用蓝图库提供的合并策略。这些策略提供了解析 使用重新合成功能生成文件 部分中提到的文件输出以及解决文件冲突的方法。

- alwaysUpdate – 一种始终解析为建议的文件的策略。
- neverUpdate – 一种始终解析到现有文件的策略。

- `onlyAdd` – 一种在现有文件尚不存在时解析为建议的文件的策略。否则，解析为现有文件。
- `threeWayMerge` – 一种在现有的、建议的和共同的父级文件之间进行三向合并的策略。如果无法干净地合并文件，则已解析的文件可能包含冲突标记。提供的文件的内容必须已通过 UTF-8 进行编码，这样策略才能生成有意义的输出。该策略尝试检测输入文件是否为二进制文件。如果该策略检测到二进制文件中存在合并冲突，则它始终返回建议的文件。
- `preferProposed` – 一种在现有的、建议的和共同的父级文件之间进行三向合并的策略。该策略通过选择每个冲突的相关建议文件来解决冲突。
- `preferExisting` – 一种在现有的、建议的和共同的父级文件之间进行三向合并的策略。该策略通过选择每个冲突的相关现有文件来解决冲突。

要查看合并策略的源代码，请参阅 open-source GitHub repository。

指定用于生命周期管理更新的文件

在重新合成期间，蓝图控制如何将更改合并到现有源存储库中。但您可能不希望将更新推送到蓝图中的每个文件。例如，像 CSS 样式表这样的示例代码是项目特定的。如果您未指定其他策略，则三向合并策略将为默认选项。蓝图可以通过指定有关存储库构造本身的合并策略来指定它们拥有哪些文件和不拥有哪些文件。蓝图可以更新其合并策略，并且可在重新合成期间使用最新策略。

```
const sourceRepo = new SourceRepository(this, {
      title: 'my-repo',
    });
    sourceRepo.setResynthStrategies([
      {
        identifier: 'dont-override-sample-code',
        description: 'This strategy is applied accross all sample code. The blueprint
 will create sample code, but skip attempting to update it.',
        strategy: MergeStrategies.neverUpdate,
        globs: [
          '**/src/**',
          '**/css/**',
        ],
      },
    ]);
```

可以指定多个合并策略，并且使最后一个策略优先。未发现的文件默认为类似于 Git 的三向合并。通过 `MergeStrategies` 构造提供了几种合并策略，但您可以编写自己的合并策略。提供的策略符合 git merge strategy 驱动因素。

编写合并策略

除了使用提供的某个构建合并策略外，您还可以编写自己的策略。策略必须遵循标准策略接口。您必须编写一个策略函数来从 existing-bundle、proposed-bundle 和 ancestor-bundle 中获取文件版本，并将它们合并到单个已解析的文件中。例如：

```
type StrategyFunction = (
   /**
   * file from the ancestor bundle (if it exists)
   */
    commonAncestorFile: ContextFile | undefined,
   /**
   * file from the existing bundle (if it exists)
   */
    existingFile: ContextFile | undefined,
   /**
   * file from the proposed bundle (if it exists)
   */
    proposedFile: ContextFile | undefined,
    options?: {})
    /**
    * Return: file you'd like in the resolved bundle
    * passing undefined will delete the file from the resolved bundle
    */
=> ContextFile | undefined;
```

如果文件不存在（未定义），则该文件路径在该特定位置捆绑包中不存在。

示例：

```
strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied across all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: (ancestor, existing, proposed) => {
                const resolvedfile = ...
                ...
                // do something
                ...
                return resolvedfile
            },
            globs: [
```

```
            '**/src/**',
            '**/css/**',
          ],
        },
      ],
```

访问上下文对象以获取项目详细信息

作为蓝图作者，您可以在合成过程中访问蓝图项目的上下文，以便获取空间名称、项目名称或项目源存储库中的现有文件等信息。您还可以获取诸如蓝图正在生成的重新合成阶段之类的详细信息。例如，您可以访问上下文，以便了解您是要重新同步以生成原级捆绑包还是建议的捆绑包。之后，您可以使用现有的代码上下文来转换存储库中的代码。例如，您可以编写自己的重新合成策略来设置特定的代码标准。可以将策略添加到小型蓝图的 `blueprint.ts` 文件中，也可以为策略创建单独的文件。

以下示例说明如何在项目的上下文中查找文件、设置工作流生成器以及为特定文件设置蓝图提供的重新合成策略：

```
const contextFiles = this.context.project.src.findAll({
      fileGlobs: ['**/package.json'],
    });

    // const workflows = this.context.project.src.findAll({
    //   fileGlobs: ['**/.codecatalyst/**.yaml'],
    // });

    const security = new WorkflowBuilder(this, {
      Name: 'security-workflow',
    });
    new Workflow(this, repo, security.getDefinition());
    repo.setResynthStrategies([
      {
        identifier: 'force-security',
        globs: ['**/.codecatalyst/security-workflow.yaml'],
        strategy: MergeStrategies.alwaysUpdate,
      },
    ]);


    for (const contextFile of contextFiles) {
      const packageObject = JSON.parse(contextFile.buffer.toString());
      new SourceFile(internalRepo, contextFile.path, JSON.stringify({
        ...packageObject,
```

```
      }, null, 2));
    }
  }
```

开发自定义蓝图以满足项目要求

在发布自定义蓝图之前，您可以开发蓝图以满足特定要求。您可以通过在预览时创建项目来开发自定义蓝图并测试蓝图。您可以开发自定义蓝图以包含项目组件，例如特定的源代码、账户连接、工作流、事务或可在 CodeCatalyst 中创建的任何其他组件。

> **Important**
>
> 如果要使用来自外部来源的蓝图包，请考虑使用这些包可能带来的风险。您对添加到空间中的自定义蓝图以及这些蓝图生成的代码负责。

> **Important**
>
> 要在 CodeCatalyst 空间中开发自定义蓝图，您必须使用在该空间中具有空间管理员或高级用户角色的账户进行登录。

开发或更新自定义蓝图

1. 恢复您的开发环境。有关更多信息，请参阅 恢复开发环境。

   如果您没有开发环境，则必须先创建一个。有关更多信息，请参阅 创建开发环境。
2. 在您的开发环境中打开工作终端。
3. 如果您在创建蓝图时选择了发布工作流，则会自动发布最新的蓝图版本。提取更改以确保 `package.json` 文件具有递增版本。使用以下命令：

   ```
   git pull
   ```
4. 在 `src/blueprint.ts` 文件中，编辑自定义蓝图的选项。CodeCatalyst 向导会动态解释 `Options` 接口以生成选择用户界面（UI）。您可以通过添加组件和支持的标签来开发自定义蓝图。有关更多信息，请参阅使用前端向导修改蓝图功能、向蓝图添加环境组件、向蓝图添加区域组件、向蓝图添加存储库和源代码组件、向蓝图添加工作流组件和向蓝图添加开发环境组件。

在开发自定义蓝图时，还可以查看蓝图 SDK 和示例蓝图以获得其他支持。有关更多信息，请参阅 open-source GitHub repository。

在成功合成后，自定义蓝图会提供预览包。项目包代表项目中的源代码、配置和资源，它由 CodeCatalyst 部署 API 操作用来部署到项目中。如果要继续开发自定义蓝图，请重新运行蓝图合成流程。有关更多信息，请参阅 自定义蓝图概念。

使用前端向导修改蓝图功能

CodeCatalyst 上的蓝图选择向导由 `blueprint.ts` 文件中的 `Options` 接口自动生成。前端向导支持使用 JSDOC style comments and tags 对蓝图的 `Options` 进行修改和特征化。您可以使用 JSDOC 样式注释和标签来执行任务。例如，您可以选择选项上方显示的文本，启用输入验证等功能，或使选项可折叠。该向导的工作原理是解释从 `Options` 接口的 TypeScript 类型生成的抽象语法树（AST）。该向导会根据描述的类型自动进行配置。并非所有类型均受支持。其它支持的类型包括区域选择器和环境选择器。

以下是一个在蓝图的 `Options` 中使用 JSDOC 注释和标签的向导示例：

```
export interface Options {
  /**
   * What do you want to call your new blueprint?
   * @validationRegex /^[a-zA-Z0-9_]+$/
   * @validationMessage Must contain only upper and lowercase letters, numbers and
 underscores
   */
  blueprintName: string;

  /**
   * Add a description for your new blueprint.
   */
   description?: string;

   /**
    * Tags for your Blueprint:
    * @collapsed true
    */
  tags?: string[];
}
```

默认情况下，Options 接口的每个选项的显示名称使用 camelCase 显示。在向导中，JSDOC 样式注释中的纯文本将显示为选项上方的文本。

主题

- 支持的标签
- 支持的 TypeScript 类型
- 在合成过程中与用户交流

支持的标签

前端向导中的自定义蓝图的 Options 支持以下 JSDOC 标签。

@inlinePolicy ./path/to/policy/file.json

- 必需 - Role 类型选项。
- 用法 - 使您能够传达角色所需的内联策略。policy.json 路径应位于源代码下。当您需要角色的自定义策略时，请使用此标签。
- 依赖项 - blueprint-cli 0.1.12 及更高版本
- 示例 – @inlinePolicy ./deployment-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @inlinePolicy ./path/to/deployment-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@trustPolicy ./path/to/policy/file.json

- 必需 - Role 类型选项。
- 用法 - 使您能够传达角色所需的信任策略。policy.json 路径应位于源代码下。当您需要角色的自定义策略时，请使用此标签。
- 依赖项 - blueprint-cli 0.1.12 及更高版本
- 示例 – @trustPolicy ./trust-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @trustPolicy ./path/to/trust-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@validationRegex 正则表达式

- 必需 - 字符串选项。
- 用法 - 使用给定的正则表达式对选项执行输入验证并显示 @validationMessage。
- 示例 – @validationRegex /^[a-zA-Z0-9_]+$/
- 建议 - 与 @validationMessage 一起使用。默认情况下，验证消息为空。

@validationMessage 字符串

- 必需 - @validationRegex 或其它错误来查看用法。
- 用法 - @validation* 失败时显示验证消息。
- 示例 - @validationMessage Must contain only upper and lowercase letters, numbers, and underscores。
- 建议 - 与 @validationMessage 一起使用。默认情况下，验证消息为空。

@collapsed 布尔值（可选）

- 必需 - 不适用
- 用法 - 使子选项能够折叠的布尔值。如果存在折叠的注释，则其默认值为 true。将该值设置为 @collapsed false 可创建最初处于打开状态的可折叠部分。
- 示例 – @collapsed true

@displayName 字符串

- 必需 - 不适用
- 用法 - 更改选项的显示名称。使显示名称能够使用除 camelCase 以外的格式。
- 示例 – @displayName Blueprint Name

@displayName 字符串

- 必需 - 不适用
- 用法 - 更改选项的显示名称。使显示名称能够使用除 camelCase 以外的格式。
- 示例 – `@displayName Blueprint Name`

@defaultEntropy 数字

- 必需 - 字符串选项。
- 用法 - 将指定长度的随机字母数字字符串附加到选项。
- 示例 – `@defaultEntropy 5`

@placeholder 字符串（可选）

- 必需 - 不适用
- 用法 - 更改默认文本字段占位符。
- 示例 – `@placeholder type project name here`

@textArea 数字（可选）

- 必需 - 不适用
- 用法 - 将字符串输入转换为文本区域组件，用于显示较大的文本部分。添加一个数字就定义了行数。默认为五行。
- 示例 – `@textArea 10`

@hidden 布尔值（可选）

- 必需 - 不适用
- 用法 - 除非验证检查失败，否则向用户隐藏文件。默认值为 true。
- 示例 – `@hidden`

@button 布尔值（可选）

- 必需 - 不适用
- 用法 - 注释必须是布尔属性的注释。添加一个按钮，选择后将合成为 true。不是切换按钮。

- 示例 – `buttonExample: boolean;`

```
/**
  * @button
  */
buttonExample: boolean;
```

@showName 布尔值（可选）

- 必需 - 不适用
- 用法 - 只能用于账户连接类型。显示隐藏的名称输入。默认值为 `default_environment`。
- 示例 – `@showName true`

```
/**
  * @showName true
  */
accountConnection: AccountConnection<{
    ...
}>;
```

@showEnvironmentType 布尔值（可选）

- 必需 - 不适用
- 用法 - 只能用于账户连接类型。显示隐藏的环境类型下拉菜单。所有连接都默认为 `production`。选项有 Non-production 或 Production。
- 示例 – `@showEnvironmentType true`

```
/**
  * @showEnvironmentType true
  */
accountConnection: AccountConnection<{
    ...
}>;
```

@forceDefault 布尔值（可选）

- 必需 - 不适用

- 用法 - 使用蓝图作者提供的默认值，而不是用户之前使用的值。
- 示例 – `forceDeafultExample: any;`

```
/**
  * @forceDefault
  */
forceDeafultExample: any;
```

@requires blueprintName

- 必需 - `Options` 接口注释。
- 用法 - 警告用户将指定的 `blueprintName` 添加到项目中，作为当前蓝图的要求。
- 示例 – `@requires '@amazon-codecatalyst/blueprints.blueprint-builder'`

```
/*
 * @requires '@amazon-codecatalyst/blueprints.blueprint-builder'
 */
export interface Options extends ParentOptions {
...
```

@filter regex

- 必需 - `Selector` 或 `MultiSelect` 接口注释。
- 用法 - 将向导中的下拉菜单筛选为与指定正则表达式匹配的选项。
- 示例 – `@filter /blueprintPackageName/`

```
 /**
     * @filter /myPackageName/
     */
    blueprintInstantiation?: Selector<BlueprintInstantiation>;
...
```

支持的 TypeScript 类型

前端向导中的自定义蓝图的 `Options` 支持以下 TypeScript 类型。

数字

- 必需 - number 类型选项。
- 用法 - 生成数字输入字段。
- 示例 – age: number

```
{
  age: number
  ...
}
```

字符串

- 必需 - string 类型选项。
- 用法 - 生成字符串输入字段。
- 示例 – name: string

```
{
  age: string
  ...
}
```

字符串列表

- 必需 - string 类型数组选项。
- 用法 - 生成字符串列表输入。
- 示例 – isProduction: boolean

```
{
  isProduction: boolean
  ...
}
```

Checkbox

- 必需 - boolean 选项。

- 用法 - 生成一个复选框。
- 示例 – `isProduction: boolean`

```
{
  isProduction: boolean
  ...
}
```

单选框

- 必需 - 三个或更少字符串的组合选项。
- 用法 - 生成选定的单选框。

> **Note**
>
> 当有四个或更多项目时，此类型将以下拉列表的形式呈现。

- 示例 – `color: 'red' | 'blue' | 'green'`

```
{
  color: 'red' | 'blue' | 'green'
  ...
}
```

下拉菜单

- 必需 - 四个或更多字符串的组合选项。
- 用法 - 生成一个下拉列表。
- 示例 – `runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'`

```
{
  runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'
  ...
}
```

可扩展部分

- 必需 - 对象选项。
- 用法 - 生成可扩展部分。对象中的选项将嵌套在向导中的可扩展部分内。
- 示例 -

```
{
     expandableSectionTitle: {
         nestedString: string;
         nestedNumber: number;
     }
}
```

Tuple

- 必需 - Tuple 类型选项。
- 用法 - 生成键值对输入。
- 示例 – `tuple: Tuple[string, string]>`

```
{
    tuple: Tuple[string, string]>;
    ...
}
```

Tuple 列表

- 必需 - Tuple 类型数组选项。
- 用法 - 生成 tuple 列表输入。
- 示例 – `tupleList: Tuple[string, string]>[]`

```
{
  tupleList: Tuple[string, string]>[];
  ...
}
```

Selector

- 必需 - `Selector` 类型选项。
- 用法 - 生成应用于项目的源存储库或蓝图的下拉列表。
- 示例 – `sourceRepo: Selector<SourceRepository>`

```
{
    sourceRepo: Selector<SourceRepository>;
    sourceRepoOrAdd: Selector<SourceRepository | string>;
    blueprintInstantiation: Selector<BlueprintInstantiation>;
  ...
}
```

Multiselect

- 必需 - `Selector` 类型选项。
- 用法 - 生成多选输入。
- 示例 – `multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>`

```
{
  multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>;
  ...
}
```

在合成过程中与用户交流

作为蓝图作者，您可以与用户交流，而不仅仅是验证消息。例如，空间成员可能会查看一个选项组合，而该组合产生的蓝图并不清晰。自定义蓝图支持通过调用合成向用户反馈错误消息的功能。基本蓝图实现了 `throwSynthesisError(...)` 函数，该函数希望得到明确的错误消息。您可以通过以下方式调用消息：

```
//blueprint.ts
this.throwSynthesisError({
   name: BlueprintSynthesisErrorTypes.BlueprintSynthesisError,
   message: 'hello from the blueprint! This is a custom error communicated to the
 user.'
})
```

生成输入和重新呈现前端向导元素

您可以使用 DynamicKVInput 生成向导输入，并为自定义蓝图动态创建前端向导元素。

主题

创建开发环境

您可以使用 DynamicKVInput 类型，使用自定义蓝图的默认值生成前端向导输入。要查看最新的架构，请参阅 DynamicKVInput definition。

以下示例说明如何使用 Options 塑造对象：

```
import { DynamicKVInput } from '@amazon-codecatalyst/blueprints.blueprint';

export interface Options extends ParentOptions {

  parameters: DynamicKVInput[];

}
```

以下示例说明如何设置多个属性的默认参数：

```
{
"parameters": [
        {
            "key": "AWS_REGION",
            "value": "us-west-2",
            "displayType": "region",
            "possibleValues": [
                "us-west-1",
                "us-west-2",
                "us-east-1",
                "us-east-2"
            ],
            "displayName": "AWS Region",
            "description": "AWS Region to deploy the solution to."
        },
        {
```

```
            "key": "SchedulingActive",
            "value": "Yes",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Scheduling Active",
            "description": "Activate or deactivate scheduling."
        },
        {
            "key": "ScheduledServices",
            "value": "Both",
            "displayType": "dropdown",
            "possibleValues": [
                "EC2",
                "RDS",
                "Both"
            ],
            "displayName": "Scheduled Services",
            "description": "Services to schedule."
        },
        {
            "key": "ScheduleRdsClusters",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Schedule RDS Clusters",
            "description": "Enable scheduling of Aurora clusters for RDS service."
        },
        {
            "key": "CreateRdsSnapshot",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Create RDS Snapshot",
            "description": "Create snapshot before stopping RDS instances (does not
  apply to Aurora Clusters)."
```

```
        },
        {
            "key": "MemorySize",
            "value": "128",
            "displayType": "dropdown",
            "possibleValues": [
                "128",
                "384",
                "512",
                "640",
                "768",
                "896",
                "1024",
                "1152",
                "1280",
                "1408",
                "1536"
            ],
            "displayName": "Memory Size",
            "description": "Size of the Lambda function running the scheduler, increase
 size when processing large numbers of instances."
        },
        {
            "key": "UseCloudWatchMetrics",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Use CloudWatch Metrics",
            "description": "Collect instance scheduling data using CloudWatch metrics."
        },
        {
            "key": "LogRetentionDays",
            "value": "30",
            "displayType": "dropdown",
            "possibleValues": [
                "1",
                "3",
                "5",
                "7",
                "14",
                "30",
```

```
                "60",
                "90",
                "120",
                "150",
                "180",
                "365",
                "400",
                "545",
                "731",
                "1827",
                "3653"
            ],
            "displayName": "Log Retention Days",
            "description": "Retention days for scheduler logs."
        },
        {
            "key": "Trace",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Trace",
            "description": "Enable debug-level logging in CloudWatch logs."
        },
        {
            "key": "EnableSSMMaintenanceWindows",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Enable SSM Maintenance Windows",
            "description": "Enable the solution to load SSM Maintenance Windows, so
that they can be used for EC2 instance Scheduling."
        },
        {
            "key": "DefaultTimezone",
            "value": "UTC",
            "displayType": "string",
            "displayName": "Default Timezone",
            "description": "Default timezone to use for scheduling."
```

```
        },
        {
            "key": "Regions",
            "value": "us-west-2",
            "displayType": "string",
            "displayName": "Regions",
            "description": "List of regions in which instances should be scheduled,
leave blank for current region only."
        },
        {
            "key": "UsingAWSOrganizations",
            "value": "No",
            "displayType": "dropdown",
            "possibleValues": [
                "Yes",
                "No"
            ],
            "displayName": "Using AWS Organizations",
            "description": "Use AWS Organizations to automate spoke account
registration."
        },
        {
            "key": "Principals",
            "displayType": "string",
         "optional": false,
            "displayName": "Principals",
            "description": "(Required) If using AWS Organizations, provide the
Organization ID. Eg. o-xxxxyyy. Else, provide a comma separated list of spoke account
ids to schedule. Eg.: 1111111111, 2222222222 or {param: ssm-param-name}"
        },
        {
            "key": "Namespace",
            "value": "Default",
            "displayType": "string",
            "displayName": "Namespace",
            "description": "Provide unique identifier to differentiate between multiple
solution deployments (No Spaces). Example: Dev"
        },
        {
            "key": "SchedulerFrequency",
            "value": 5,
            "displayType": "number",
            "displayName": "Scheduler Frequency",
            "description": "Scheduler running frequency in minutes."
```

```
        }
    ]
}
```

动态创建向导元素

在合成过程中，可以使用与创建向导输入相同的架构来动态地重新呈现向导。在必要时，这可以用来解决用户的后续问题。

```
//blueprint.ts

export interface Options extends ParentOptions {
...
 dynamicOptions: OptionsSchemaDefinition<'optionsIdentifier', KVSchema>;
}
```

然后，可以在合成期间使用 Options 组件设置向导。

```
import {
  OptionsSchemaDefinition,
  OptionsSchema,
} from '@amazon-codecatalyst/blueprints.blueprint';

...

  // dynamically renders a number in the place where 'optionsIdentifier' was set in the
 original options type.
  new OptionsSchema<KVSchema>(this, 'optionsIdentifier', [
    {
            "key": "SchedulerFrequency",
            "value": 5,
            "displayType": "number",
            "displayName": "Scheduler Frequency",
            "description": "Scheduler running frequency in minutes."
    }
   ]);
```

向蓝图添加环境组件

自定义蓝图向导是通过向导公开的 Options 界面动态生成的。蓝图支持从公开的类型生成用户界面（UI）组件。

导入 Amazon CodeCatalyst 蓝图环境组件

在您的 `blueprint.ts` 文件中，添加以下内容：

```
import {...} from '@amazon-codecatalyst/codecatalyst-environments'
```

主题

- 创建开发环境
- 环境的列表
- 模拟接口示例

创建开发环境

以下示例说明如何将您的应用程序部署到云：

```
export interface Options extends ParentOptions {
        ...
        myNewEnvironment:  EnvironmentDefinition{
            thisIsMyFirstAccountConnection: AccountConnection{
                thisIsARole: Role['lambda', 's3', 'dynamo'];
             };
        };
    }
```

该接口会生成一个 UI 组件，要求使用单一账户连接（`thisIsMyFirstAccountConnection`）的新环境（`myNewEnvironment`）。此外还会在账户连接（`thisIsARole`）上生成一个角色，具备 `['lambda', 's3', 'dynamo']` 作为要求的最低角色功能。并非所有用户都具有账户连接，因此您应该检查用户未连接账户或未将账户与角色连接的情况。角色也可以使用 `@inlinePolicies` 进行注释。有关更多信息，请参阅 @inlinePolicy ./path/to/policy/file.json。

环境组件需要 `name` 和 `environmentType`。以下代码是所需的最低默认设置：

```
{
  ...
  "myNewEnvironment": {
    "name": "myProductionEnvironment",
    "environmentType": "PRODUCTION"
  },
}
```

然后，用户界面组件会提示您在各种字段中输入内容。当您填写字段时，蓝图会完全展开。出于测试和开发目的，在 defaults.json 文件中包含完整的模拟可能会对您有所帮助。

环境的列表

指定 EnvironmentDefinition 类型的数组将在向导 UI 中生成环境列表。

```
export interface Options extends ParentOptions {
    ...
   /**
     @showName readOnly
   */
    myEnvironments:  EnvironmentDefinition<{
        thisIsMyFirstAccountConnection: AccountConnection<{
            thisIsARole: Role<['lambda', 's3', 'dynamo']>;
        }>;
    }>[];

}
```

以下示例说明环境列表的默认值：

```
{
  ...
  "myEnvironments": [
  {
    "name": "myProductionEnvironment",
    "environmentType": "PRODUCTION"
  },
  {
    "name": "myDevelopmentEnvironment",
    "environmentType": "DEVELOPMENT"
  },
  ]
}
```

模拟接口示例

简单的模拟界面

```
{
    ...
```

```
    "thisIsMyEnvironment": {
        "name": "myProductionEnvironment",
        "environmentType": "PRODUCTION",
        "thisIsMySecondAccountConnection": {
            "id": "12345678910",
            "name": "my-account-connection-name",
            "secondAdminRole": {
                "arn": "arn:aws:iam::12345678910:role/ConnectedQuokkaRole",
                "name": "ConnectedQuokkaRole",
                "capabilities": [
                    "lambda",
                    "s3",
                    "dynamo"
                ]
            }
        }
    }
}
```

复杂的模拟界面

```
export interface Options extends ParentOptions {
  /**
   * The name of an environment
   * @displayName This is a Environment Name
   * @collapsed
   */
  thisIsMyEnvironment: EnvironmentDefinition{
    /**
     * comments about the account that is being deployed into
     * @displayName This account connection has an overriden name
     * @collapsed
     */
    thisIsMyFirstAccountConnection: AccountConnection{
      /**
       * Blah blah some information about the role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
       * @displayName This role has an overriden name
       */
      adminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
       * Blah blah some information about the second role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
```

```
       */
      lambdaRole: Role['lambda', 's3'];
    };
    /**
     * comments about the account that is being deployed into
     */
    thisIsMySecondAccountConnection: AccountConnection{
      /**
         * Blah blah some information about the role that I expect
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondAdminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
         * Blah blah some information about the second role that I expect
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondLambdaRole: Role['lambda', 's3'];
    };
  };
}
```

完整的模拟界面

```
{
  ...
  "thisIsMyEnvironment": {
    "name": "my-production-environment",
    "environmentType": "PRODUCTION",
    "thisIsMySecondAccountConnection": {
      "id": "12345678910",
      "name": "my-connected-account",
      "secondAdminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "secondLambdaRole": {
        "name": "LambdaQuokkaRole",
```

```
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
    },
    "thisIsMyFirstAccountConnection": {
      "id": "12345678910",
      "name": "my-connected-account",
      "adminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "lambdaRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
    }
  },
}
```

向蓝图添加密钥组件

可以在 CodeCatalyst 中使用密钥来存储可在工作流中引用的敏感数据。您可以向自定义蓝图添加密钥并在工作流中引用该密钥。有关更多信息，请参阅 使用密钥遮蔽数据。

导入 Amazon CodeCatalyst 蓝图区域类型

在您的 `blueprint.ts` 文件中，添加以下内容：

```
import { Secret, SecretDefinition } from '@amazon-codecatalyst/blueprint-
component.secrets'
```

主题

创建密钥

以下示例创建了一个 UI 组件，该组件提示用户输入密钥值和可选描述：

```
export interface Options extends ParentOptions {
    ...
    mySecret: SecretDefinition;
}


export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    new Secret(this, options.secret);
}
```

密钥组件需要 name。以下代码是所需的最低默认设置：

```
{
    ...
    "secret": {
        "name": "secretName"
    },

}
```

在工作流中引用密钥

以下示例蓝图创建了一个密钥和一个引用该密钥值的工作流。有关更多信息，请参阅 在工作流中引用密钥。

```
export interface Options extends ParentOptions {
    ...
/**
*
* @validationRegex /^\w+$/
*/
  username: string;
```

```
  password: SecretDefinition;
}


export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    const password = new Secret(this, options_.password);

    const workflowBuilder = new WorkflowBuilder(this, {
      Name: 'my_workflow',
    });


    workflowBuilder.addBuildAction({
      actionName: 'download_files',
      input: {
        Sources: ['WorkflowSource'],
      },
      output: {
        Artifacts: [{ Name: 'download', Files: ['file1'] }],
      },
      steps: [
        `curl -u ${options_.username}:${password.reference} https://example.com`,
      ],
    });

    new Workflow(
      this,
      repo,
      workflowBuilder.getDefinition(),
    );

}
```

要了解有关在 CodeCatalyst 中使用密钥的更多信息，请参阅使用密钥遮蔽数据。

向蓝图添加区域组件

可以将区域类型添加到自定义蓝图的 Options 界面，以便在蓝图向导中生成可输入一个或多个 AWS 区域的组件。可以从 blueprint.ts 文件中的基础蓝图导入区域类型。有关更多信息，请参阅 AWS 区域。

导入 Amazon CodeCatalyst 蓝图区域类型

在您的 `blueprint.ts` 文件中，添加以下内容：

```
import { Region } from '@amazon-codecatalyst/blueprints.blueprint'
```

区域类型参数是一组可供选择的 AWS 区域代码，您也可以使用 * 包含所有受支持的 AWS 区域。

主题

注释

可以向 `Options` 界面中的每个字段添加 JSDoc 标签，以自定义字段在向导中的显示和行为方式。对于区域类型，支持以下标签：

- `@displayName` 注释可用于在向导中更改字段的标签。

  示例：`@displayName AWS Region`
- `@placeholder` 注释可用于更改选择/多选组件的占位符。

  示例：`@placeholder Choose AWS Region`

区域组件示例

从指定列表中选择区域

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>;
}
```

从指定列表中选择一个或多个区域

```
export interface Options extends ParentOptions {
```

```
    ...
  /**
   * @displayName Regions
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

选择一个 AWS 区域

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['*']>;
}
```

从指定列表中选择一个或多个区域

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Regions
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

向蓝图添加存储库和源代码组件

Amazon CodeCatalyst 使用存储库来存储代码。存储库将名称用作输入。大多数组件都存储在存储库中，例如源代码文件、工作流和其他组件，例如托管式开发环境（MDE）。源存储库组件还导出用于管理文件和静态资产的各种组件。存储库有名称限制。有关更多信息，请参阅 使用源存储库存储代码并协作处理代码 CodeCatalyst。

```
const repository = new SourceRepository(this, {
  title: 'my-new-repository-title',
});
```

导入 Amazon CodeCatalyst 蓝图存储库和源代码组件

在您的 `blueprint.ts` 文件中，添加以下内容：

```
import {...} from '@caws-blueprint-component/caws-source-repositories'
```

主题

添加文件

您可以使用 SourceFile 构造，将文本文件写入到存储库中。该操作是最常见的应用场景之一，需要使用存储库、文件路径和文本内容。如果存储库中不存在文件路径，则该组件会创建所有必需的文件夹。

```
new SourceFile(repository, `path/to/my/file/in/repo/file.txt`, 'my file contents');
```

> **Note**
>
> 如果您将两个文件写入到同一存储库中的相同位置，则最新的实施会覆盖前一个实施。您可以使用该功能对生成的代码进行分层，这对于扩展自定义蓝图可能已生成的代码尤其有用。

添加通用文件

您可以向自己的存储库中写入任意位。您可以从缓冲区读取并使用 File 构造。

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

复制文件

您可以复制粘贴起始代码，然后在这个基础上生成更多代码，从而开始使用生成的代码。将代码放在 static-assets 目录中，然后使用 StaticAsset 构造来定位该代码。在这种情况下，路径始终从 static-assets 目录的根目录开始。

```
const starterCode = new StaticAsset('path/to/file/file.txt')
const starterCodeText = new StaticAsset('path/to/file/file.txt').toString()
const starterCodeRawContent = new StaticAsset('path/to/image/hello.png').content()

const starterCodePath = new StaticAsset('path/to/image/hello.png').path()
// starterCodePath is equal to 'path/to/image/hello.png'
```

StaticAsset 的子类是 SubstitutionAsset。子类的功能完全相同，不过您可以改为对文件执行 Mustache 替换。这对于执行复制和替换样式生成非常有用。

静态资产替换使用 Mustache 模板引擎来呈现植入到所生成的源存储库中的静态文件。Mustache 模板规则是在呈现期间应用的，这意味着默认情况下，所有值都是 HTML 编码值。要呈现未转义的 HTML，请使用三个花括号语法 {{{name}}}。有关更多信息，请参阅 mustache templating rules。

> **Note**
>
> 对无法使用文本解释的文件运行替换操作时，可能会产生错误。

```
const starterCodeText = new SubstitionAsset('path/to/file/file.txt').subsitite({
  'my_variable': 'subbed value1',
  'another_variable': 'subbed value2'
})
```

定位多个文件

静态资源支持通过 StaticAsset 上的静态函数以及名为 findAll(...) 的子类，来进行 glob 定位，这将返回已预加载路径、内容等的静态资源列表。您可以将列表与 File 结构链接起来，以便在 static-assets 目录中复制和粘贴内容。

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

创建新的存储库并添加文件

您可以使用存储库组件，在生成的项目中创建新的存储库。然后，您可以将文件或工作流添加到创建的存储库中。

```
import { SourceRepository } from '@amazon-codecatalyst/codecatalyst-source-
repositories';
...
const repository = new SourceRepository(this, { title: 'myRepo' });
```

以下示例说明如何向现有存储库添加文件和工作流：

```
import { SourceFile } from '@amazon-codecatalyst/codecatalyst-source-repositories';
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows';
...
new SourceFile(repository, 'README.md', 'This is the content of my readme');
new Workflow(this, repository, {/**...workflowDefinition...**/});
```

将这两段代码组合在一起时，可以生成单个名为 myRepo 的存储库，其根目录中包含一个源文件 README.md 和一个 CodeCatalyst 工作流。

向蓝图添加工作流组件

Amazon CodeCatalyst 项目使用工作流，根据触发器运行操作。您可以使用工作流组件来构建和整理工作流 YAML 文件。有关更多信息，请参阅 工作流 YAML 定义。

导入 Amazon CodeCatalyst 蓝图工作流组件

在您的 blueprint.ts 文件中，添加以下内容：

```
import { WorkflowBuilder, Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
```

主题

- 工作流组件示例
- 连接到环境

工作流组件示例

WorkflowBuilder 组件

您可以使用类来构建工作流定义。可以为工作流组件提供定义，以便在存储库中进行呈现。

```
import { WorkflowBuilder } from '@amazon-codecatalyst/codecatalyst-workflows'
```

```
const workflowBuilder = new WorkflowBuilder({} as Blueprint, {
  Name: 'my_workflow',
});

// trigger the workflow on pushes to branch 'main'
workflowBuilder.addBranchTrigger(['main']);

// add a build action
workflowBuilder.addBuildAction({
  // give the action a name
  actionName: 'build_and_do_some_other_stuff',

  // the action pulls from source code
  input: {
    Sources: ['WorkflowSource'],
  },

  // the output attempts to autodiscover test reports, but not in the node modules
  output: {
    AutoDiscoverReports: {
      Enabled: true,
      ReportNamePrefix: AutoDiscovered,
      IncludePaths: ['**/*'],
      ExcludePaths: ['*/node_modules/**/*'],
    },
  },
  // execute some arbitrary steps
  steps: [
    'npm install',
    'npm run myscript',
    'echo hello-world',
  ],
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

工作流 Projen 组件

以下示例说明如何使用 Projen 组件将工作流 YAML 写入到存储库中：

```
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'

...
```

```
const repo = new SourceRepository
const blueprint = this;
const workflowDef = workflowBuilder.getDefinition()

// creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

连接到环境

许多工作流需要在 AWS 账户连接中运行。工作流可让操作使用账户和角色名称规范连接到环境，从而处理这种情况。

```
import { convertToWorkflowEnvironment } from '@amazon-codecatalyst/codecatalyst-
workflows'


const myEnv = new Environment(...);

// can be passed into a workflow constructor
const workflowEnvironment = convertToWorkflowEnvironment(myEnv);


// add a build action
workflowBuilder.addBuildAction({
  ...
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

向蓝图添加开发环境组件

托管式开发环境（MDE，Managed Development Environment）用于在 CodeCatalyst 中创建和建立 MDE 工作区。该组件生成一个 `devfile.yaml` 文件。有关更多信息，请参阅 Introduction to Devfile 和编辑开发环境的存储库 devfile。

```
new Workspace(this, repository, SampleWorkspaces.default);
```

导入 Amazon CodeCatalyst 蓝图工作区组件

在您的 blueprint.ts 文件中，添加以下内容：

```
import {...} from '@amazon-codecatalyst/codecatalyst-workspaces'
```

向蓝图添加事务组件

在 CodeCatalyst 中，您可以监控项目中涉及的功能、任务、错误以及任何其他工作。每项工作均保存在称为事务的不同记录中。每个事务都可以具有描述、被分派人、状态和其他属性，您可以对这些属性进行搜索、分组和筛选。您可以使用默认视图查看自己的事务，也可以使用自定义筛选、排序或分组来创建自己的视图。有关与事务相关的概念的更多信息，请参阅事务概念和 中的问题配额 CodeCatalyst。

事务组件可以生成事务的 JSON 表示形式。该组件将 ID 字段和事务定义用作输入。

导入 Amazon CodeCatalyst 蓝图事务组件

在您的 blueprint.ts 文件中，添加以下内容：

```
import {...} from '@amazon-codecatalyst/blueprint-component.issues'
```

主题

- 事务组件示例

事务组件示例

创建事务

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myFirstIssue', {
  title: 'myFirstIssue',
  content: 'This is an example issue.',
});
```

创建高优先级事务

```
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
...
const repo = new SourceRepository
const blueprint = this;
```

```
const workflowDef = workflowBuilder.getDefinition()

// Creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// Can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

使用标签创建低优先级事务

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myThirdIssue', {
  title: 'myThirdIssue',
  content: 'This is an example of a low priority issue with a label.',
  priority: 'LOW',
  labels: ['exampleLabel'],
});
```

使用蓝图工具和 CLI

蓝图 CLI 提供了用来管理和使用自定义蓝图的工具。

主题

- 使用蓝图工具
- 图像上传工具

使用蓝图工具

使用蓝图工具

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 恢复您的开发环境。有关更多信息，请参阅 恢复开发环境。

   如果您没有开发环境，则必须先创建一个。有关更多信息，请参阅 创建开发环境。
3. 在正运行的终端中，运行以下命令来安装蓝图 CLI：

   ```
   npm install -g @amazon-codecatalyst/blueprint-util.cli
   ```

4. 在 blueprint.ts 文件中，按以下格式导入要使用的工具：

```
import { <tooling-function-name> } from '@amazon-codecatalyst/blueprint-util.cli/
lib/<tooling-folder-name>/<tooling-file-name>;
```

> **Tip**
>
> 您可以转至 CodeCatalyst blueprints GitHub repository，查找要使用的工具的名称。

如果您要使用图像上传工具，请在脚本中添加以下内容：

```
import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
image-upload-tool/upload-image-to-aws';
```

示例

- 如果您要使用发布功能，请在脚本中添加以下内容：

```
import { publish } from '@amazon-codecatalyst/blueprint-util.cli/lib/publish/
publish';
```

- 如果您要使用图像上传工具，请在脚本中添加以下内容：

```
import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
image-upload-tool/upload-image-to-aws';
```

5. 调用函数。

示例

- 如果您要使用发布功能，请在脚本中添加以下内容：

```
await publish(logger, config.publishEndpoint, {<your publishing options>});
```

- 如果您要使用图像上传工具，请在脚本中添加以下内容：

```
const {imageUrl, imageName} = await uploadImagePublicly(logger, 'path/to/
image'));
```

图像上传工具

使用图像上传工具，您可将图像上传到自己的 AWS 账户中的 S3 存储桶，然后通过 CloudFront 公开分发图像。该工具将本地存储空间中的图像路径（以及可选的存储桶名称）作为输入内容，并返回公开可用的图像的 URL。有关更多信息，请参阅什么是 Amazon CloudFront？以及什么是 Amazon S3？

使用图像上传工具

1. 克隆提供蓝图 SDK 和示例蓝图访问权限的开源蓝图 GitHub 存储库。在正运行的终端中，运行以下命令：

```
git clone https://github.com/aws/codecatalyst-blueprints.git
```

2. 运行以下命令，导航到蓝图 GitHub 存储库：

```
cd codecatalyst-blueprints
```

3. 运行以下命令以安装依赖项：

```
yarn && yarn build
```

4. 运行以下命令，确保已安装了最新的蓝图 CLI 版本：

```
yarn upgrade @amazon-codecatalyst/blueprint-util.cli
```

5. 使用要将图像上传到的 S3 存储桶登录 AWS 账户。有关更多信息，请参阅 Configure the AWS CLI 和 Sign in through the AWS Command Line Interface。
6. 从 CodeCatalyst 存储库的根目录运行以下命令，以便使用蓝图 CLI 导航到该目录：

```
cd packages/utils/blueprint-cli
```

7. 运行以下命令，将您的图像上传到 S3 存储桶。

```
yarn blueprint upload-image-public <./path/to/your/image>
      <optional:optional-bucket-name>
```

此时会生成您的图像的 URL。由于部署 CloudFront 分配需要一些时间，因此该 URL 不会立即可用。检查分配状态，获取最新的部署状态。有关更多信息，请参阅使用分配。

通过快照测试评测接口变化

支持在蓝图的多个配置中生成快照测试。

蓝图支持对您作为蓝图作者提供的配置进行快照测试。配置是部分覆盖，合并在蓝图根目录下的 defaults.json 文件之上。启用并配置快照测试后，构建和测试流程会综合给定的配置，并验证综合输出与参考快照相比是否没有变化。要查看快照测试代码，请访问 CodeCatalyst blueprints GitHub repository。

启用快照测试

1. 在 `.projenrc.ts` 文件中，用要拍摄快照的文件更新 ProjenBlueprint 的输入对象。例如：

```
{
  ....
  blueprintSnapshotConfiguration: {
    snapshotGlobs: ['**', '!environments/**', '!aws-account-to-environment/**'],
  },
}
```

2. 重新合成蓝图，在蓝图项目中创建 TypeScript 文件。不要编辑源文件，因为它们是由 Projen 维护和重新生成的。使用以下命令：

```
yarn projen
```

3. 导航到 `src/snapshot-configurations` 目录，查看带有空对象的 `default-config.json` 文件。用您自己的一个或多个测试配置更新或替换该文件。然后将每个测试配置与项目的 `defaults.json` 文件合并、合成，并在测试时与快照进行比较。使用以下命令进行测试：

```
yarn test
```

首次使用测试命令时，将显示以下消息：`Snapshot Summary › NN snapshots written from 1 test suite`。随后的测试运行验证合成输出与快照相比是否没有变化，并显示以下消息：`Snapshots: NN passed, NN total`。

如果您有意更改蓝图以产生不同的输出，请运行以下命令更新参考快照：

```
yarn test:update
```

快照希望每次运行的合成输出恒定不变。如果您的蓝图生成的文件各不相同，您必须将这些文件排除在快照测试之外。更新 `ProjenBluerpint` 输入对象的 `blueprintSnapshotConfiguration` 对象，添加 `snapshotGlobs` 属性。`snapshotGlobs` 属性是一个 globs 数组，用于确定快照包含或排除哪些文件。

> **Note**
>
> 有一个默认的 globs 列表。如果您指定了自己的列表，可能需要明确恢复默认条目。

将自定义蓝图发布到空间

您必须先将自定义蓝图发布到您的空间，之后才能将该自定义蓝图添加到空间的蓝图目录中。您也可以在发布前在 CodeCatalyst 控制台中查看蓝图。您可以发布蓝图的预览版或正常版本。

> **Important**
>
> 如果要使用来自外部来源的蓝图包，请考虑使用这些包可能带来的风险。您对添加到空间中的自定义蓝图以及这些蓝图生成的代码负责。

> **Important**
>
> 要将自定义蓝图发布到 CodeCatalyst 空间，您必须使用在该空间中具有空间管理员或高级用户角色的账户进行登录。

主题

- 查看和发布自定义蓝图的预览版
- 查看和发布自定义蓝图的正常版本
- 在指定的空间和项目中发布和添加自定义蓝图

查看和发布自定义蓝图的预览版

如果您想将自定义蓝图的预览版添加到空间的蓝图目录中，则可以将预览版发布到您的空间。这样就可以在将非预览版本添加到目录之前，以用户身份查看蓝图。使用预览版可以在不占用实际版本的情况下

进行发布。例如，如果您正在处理 0.0.1 版本，则可以发布和添加预览版本，这样就可以发布第二个版本的新更新并将其添加为 0.0.2。

进行更改后，可以通过运行 package.json 文件来重新构建自定义蓝图的程序包，并预览所做的更改。

查看和发布自定义蓝图的预览版

1. 恢复您的开发环境。有关更多信息，请参阅恢复开发环境。
2. 在您的开发环境中打开工作终端。
3. （可选）在正常运行的终端中，如果您尚未安装项目所需的依赖项，请安装这些依赖项。使用以下命令：

   ```
   yarn
   ```

4. （可选）如果您对 .projenrc.ts 文件进行了更改，请在构建和预览蓝图之前重新生成项目的配置。使用以下命令：

   ```
   yarn projen
   ```

5. 使用以下命令重新构建和预览您的自定义蓝图。使用以下命令：

   ```
   yarn blueprint:preview
   ```

   导航到提供的 See this blueprint at: 链接以预览您的自定义蓝图。根据您的配置，检查用户界面（包括文本）是否按预期显示。如果要更改自定义蓝图，可以编辑 blueprint.ts 文件，重新同步蓝图，然后再次发布预览版本。有关更多信息，请参阅重新合成。
6. （可选）您可以发布自定义蓝图的预览版，随后可将其添加到空间的蓝图目录中。导航到 Enable version *[preview version number]* at: 链接以将预览版发布到您的空间。

您无需在 CodeCatalyst 中创建项目即可模拟项目创建过程。要合成项目，请使用以下命令：

```
yarn blueprint:synth
```

这将在 synth/synth.*[options-name]*/proposed-bundle/ 文件夹中生成蓝图。有关更多信息，请参阅 合成。

如果您要更新自定义蓝图，请改用以下命令来重新合成您的项目：

```
yarn blueprint:resynth
```

这将在 synth/synth.*[options-name]*/proposed-bundle/ 文件夹中生成蓝图。有关更多信息，请参阅 重新合成。

发布预览版后，您可以添加蓝图，以便空间成员能够使用它来创建新项目或将它添加到现有项目。有关更多信息，请参阅 将自定义蓝图添加到空间蓝图目录中。

查看和发布自定义蓝图的正常版本

开发和预览自定义蓝图后，您可以查看和发布要添加到空间的蓝图目录的新版本。创建项目时生成的发布工作流将自动发布已推送的更改。如果您在创建蓝图时禁用了工作流生成，则您的蓝图不会自动添加到空间的蓝图目录中。运行 yarn 命令后，您仍可将自定义蓝图发布到您的空间。

查看和发布自定义蓝图

1. 恢复您的开发环境。有关更多信息，请参阅恢复开发环境。
2. 在您的开发环境中打开工作终端。
3. • 如果您在创建蓝图时选择退出发布工作流生成，请使用以下命令：

   ```
   yarn blueprint:release
   ```

   您仍可导航到提供的 See this blueprint at: 链接以预览您的自定义蓝图。

   发布自定义蓝图的更新版本，随后可将其添加到空间的蓝图目录中。导航到 Enable version *[release version number]* at: 链接以将最新版本发布到您的空间。

   • 如果您在创建蓝图时选择了发布工作流，则会在推送更改时自动发布最新的蓝图版本。使用以下命令：

   ```
   git add .
   ```

   ```
   git commit -m "commit message"
   ```

   ```
   git push
   ```

发布正常版本后，您可以添加蓝图，以便空间成员能够使用它来创建新项目或将它添加到现有项目。有关更多信息，请参阅 将自定义蓝图添加到空间蓝图目录中。

在指定的空间和项目中发布和添加自定义蓝图

默认情况下，`blueprint:preview` 和 `blueprint:release` 命令会发布到已在其中创建蓝图的 CodeCatalyst 空间。如果您有多个企业空间，也可以在这些空间中预览和发布相同的蓝图。您也可以向其他空间的现有项目添加蓝图。

在指定的空间中发布或添加自定义蓝图

1. 恢复您的开发环境。有关更多信息，请参阅 恢复开发环境。
2. 在您的开发环境中打开工作终端。
3. （可选）如果您尚未安装项目所需的依赖项，请安装这些依赖项。使用以下命令：

```
yarn
```

4. 使用 `--space` 标签将预览版或正常版本发布到指定空间。例如：

- ```
  yarn blueprint:preview --space my-awesome-space # publishes under a "preview"
   version tag to 'my-awesome-space'
  ```

  输出示例：

  ```
  Enable version 0.0.1-preview.0 at: https://codecatalyst.aws/spaces/my-awesome-
  space/blueprints
  Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
  blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
  a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.0/projects/create
  ```

- ```
  yarn blueprint:release --space my-awesome-space # publishes normal version to
   'my-awesome-space'
  ```

  输出示例：

  ```
  Enable version 0.0.1 at: https://codecatalyst.aws/spaces/my-awesome-space/
  blueprints
  Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
  blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
  a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1/projects/create
  ```

使用 `--project` 将自定义蓝图的预览版添加到指定空间中的现有项目。例如：

```
yarn blueprint:preview --space my-awesome-space --project my-project # previews
 blueprint application to an existing project
```

输出示例：

```
Enable version 0.0.1-preview.1 at: https://codecatalyst.aws/spaces/my-awesome-
space/blueprints
Blueprint applied to [my-project]: https://codecatalyst.aws/spaces/my-awesome-
space/projects/my-project/blueprints/%40amazon-codecatalyst%2FmySpace.my-blueprint/
publishers/1524817d-a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.1/add
```

设置自定义蓝图的发布权限

默认情况下，如果在项目创建期间生成了工作流版本，则会启用自定义蓝图的权限。启用发布权限后，可以将蓝图发布到空间。您可以禁用该权限，这样便无法发布蓝图。禁用该权限后，蓝图创建期间生成的发布工作流将无法运行。除非启用蓝图的权限，否则无法发布对蓝图的新更改。

> **Important**
>
> 要启用或禁用自定义蓝图项目的发布权限，您必须使用在空间中具有空间管理员或高级用户角色的账户登录。

设置蓝图项目的发布权限

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要在其中管理自定义蓝图的发布权限的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 选择项目发布权限选项卡以查看空间中所有蓝图的发布权限。
5. 选择要管理的蓝图，然后选择启用或禁用以更改发布权限。如果要启用权限，请查看权限更改详细信息，然后选择启用蓝图发布以确认更改。

将自定义蓝图添加到空间蓝图目录中

将自定义蓝图发布到空间后，您可以将蓝图添加到空间蓝图目录中。如果您将自定义蓝图添加到 CodeCatalyst 空间蓝图目录中，则该蓝图可供所有空间成员在创建项目时使用或将其添加到现有项目

中。在将自定义蓝图添加到空间蓝图目录之前，必须启用蓝图的发布权限。如果您选择生成工作流发布版本，则默认情况下会启用发布权限。有关更多信息，请参阅设置自定义蓝图的发布权限和将自定义蓝图发布到空间。

> **Important**
>
> 要将自定义蓝图添加到 CodeCatalyst 空间蓝图目录中，您必须使用在该空间中具有空间管理员或高级用户角色的账户登录。

将蓝图添加到空间蓝图目录中

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 只能从源存储库的默认分支中添加蓝图。如果您在功能分支上开发了蓝图，请将您的功能分支与默认分支更改进行合并。创建拉取请求，将任何更改合并至默认分支。有关更多信息，请参阅 在 Amazon 中使用拉取请求查看代码 CodeCatalyst。
3. 在 CodeCatalyst 控制台中，导航到包含您的自定义蓝图的空间控制面板。
4. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
5. 选择要添加的蓝图名称，然后选择添加到目录。如果您有多个版本，请从目录版本下拉菜单中选择一个版本
6. 选择保存。

更改自定义蓝图的目录版本

作为蓝图作者，您可以管理要发布到空间的蓝图目录的版本。更改蓝图的目录版本不会影响使用其他蓝图版本的项目。

> **Important**
>
> 要在 Amazon CodeCatalyst 空间的蓝图目录中更改自定义蓝图的目录版本，您必须使用在该空间中具有空间管理员或高级用户角色的账户登录。

管理自定义蓝图版本

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。

2. 在 CodeCatalyst 控制台中，导航到要在其中更改自定义蓝图的版本的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 在空间蓝图表中，选择要管理的自定义蓝图的单选按钮。
5. 选择创建目录版本，然后从目录版本下拉菜单中选择版本。
6. 选择保存。

查看自定义蓝图的详细信息、版本和项目

您可以查看空间中已发布的自定义蓝图，包括蓝图的详细信息、版本以及使用该蓝图创建的或添加该蓝图的项目。

主题

查看空间的自定义蓝图

查看空间的自定义蓝图

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要在其中查看自定义蓝图的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图以查看空间蓝图。表中显示了以下详细信息：
   - 名称 - 自定义蓝图的名称。
   - 目录状态 - 自定义蓝图是否已发布到空间的蓝图目录中。
   - 最新版本 - 自定义蓝图的最新版本。
   - 上次修改时间 - 上次更新空间蓝图的日期。

查看使用自定义蓝图创建的或添加自定义蓝图的项目

查看使用自定义蓝图创建的或添加自定义蓝图的项目

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要在其中查看自定义蓝图的空间。

3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 从空间蓝图表中，选择自定义蓝图的名称以查看使用蓝图的项目和不使用蓝图的项目表。

从空间蓝图目录中移除自定义蓝图

如果您不再希望自定义蓝图用于创建新项目或应用于现有项目，则可以从空间的蓝图目录中移除自定义蓝图。

> **Note**
>
> 如果您从空间的蓝图目录中移除自定义蓝图，这不会影响从该蓝图创建的项目或应用了该蓝图的项目。不会从项目中移除蓝图的资源。

> **Important**
>
> 要从 CodeCatalyst 空间的蓝图目录中移除自定义蓝图，您必须使用在该空间中具有空间管理员或高级用户角色的账户登录。

从空间蓝图目录中移除自定义蓝图

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到包含您的自定义蓝图的空间控制面板
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 选择要移除的蓝图名称，然后选择从目录中移除蓝图。

删除已发布的自定义蓝图或版本

在您从 Amazon CodeCatalyst 空间中删除自定义蓝图的版本或蓝图本身时，您对蓝图项目或蓝图版本的资源的所有访问权限都将被移除。在您删除蓝图版本或蓝图后，项目成员将无法访问项目资源，并且第三方源存储库所触发的任何工作流都将停止。

> **Note**
>
> 如果删除蓝图，不会影响已应用该蓝图的项目。不会从项目中移除蓝图的资源。

如果蓝图版本已发布到空间的蓝图目录，请先为目录选择一个新版本，然后再删除已发布版本。

> **Important**
>
> 要从空间中删除已发布的自定义蓝图或自定义蓝图的目录版本，您必须使用在空间中具有空间管理员或高级用户角色的账户进行登录。

删除自定义蓝图的目录版本

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要从中删除自定义蓝图的目录版本的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 选择包含要删除的目录版本的蓝图的名称。
5. 选择要删除的目录版本的单选按钮，然后选择删除版本。
6. 查看详细信息，然后从选择新的蓝图目录版本下拉菜单中选择其他蓝图版本。
7. 输入 `delete` 以确认删除蓝图目录版本。
8. 选择删除。

如果某个蓝图版本不在空间的蓝图目录中，则无需选择新版本即可删除该版本。

删除自定义蓝图版本

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要从中删除自定义蓝图版本的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 选择包含要删除的版本的蓝图的名称。
5. 选择要删除的版本的单选按钮，然后选择删除版本。
6. 输入 `delete` 以确认删除蓝图版本。
7. 选择删除。

从空间的蓝图目录中删除某个蓝图会删除该蓝图的所有版本。空间中的使用蓝图的项目将不会被删除。

删除自定义蓝图版本

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在 CodeCatalyst 控制台中，导航到要从中删除自定义蓝图的空间。
3. 在空间控制面板上，选择设置选项卡，然后选择蓝图。
4. 在空间蓝图表中，选择要删除的自定义蓝图的单选按钮，然后选择删除蓝图。
5. 输入 `delete` 以确认删除自定义蓝图。
6. 选择 Delete。

处理依赖项、不匹配项和工具

依赖项管理不当可能会导致使用自定义蓝图的用户在构建时失败以及出现运行时问题。过时的工具和组件可能会导致蓝图用户无法访问最新功能和错误修复。您可以管理依赖项、处理依赖项不匹配并升级工具和组件，确保所有依赖项都依赖于相同的组件版本，并且已同步组件。

主题

- 添加依赖项
- 处理依赖项类型不匹配
- 使用 yarn 和 npm
- 升级工具和组件

添加依赖项

作为蓝图作者，您可能需要向蓝图添加包，例如 `@amazon-codecatalyst/blueprint-component.environments`。您需要使用该包更新 `projen.ts` 文件，然后使用 Projen 重新生成项目的配置。Projen 充当每个蓝图代码库的项目模型，以便通过更改模型呈现配置文件的方式来推送向后兼容的工具更新。`package.json` 文件是由 Projen 模型部分拥有的文件。Projen 确认 package.json 文件中包含的依赖项版本，但其他选项需源自模型。

添加依赖项和更新 **`projenrc.ts`** 文件

1. 在 `projen.ts` 文件中，导航到 deps 部分。
2. 添加要在蓝图中使用的依赖项。
3. 使用以下命令可重新生成项目的配置：

```
yarn projen && yarn
```

处理依赖项类型不匹配

在 Yarn 更新后，您可能会收到以下有关存储库参数的错误：

```
Type 'SourceRepository' is missing the following properties from type
 'SourceRepository': synthesisSteps, addSynthesisStep
```

导致该错误的原因是，一个组件依赖另一个组件的较新版本，但依赖组件被固定到较旧版本时，发生了依赖项不匹配的情况。可通过以下方式纠正该错误：使所有组件都依赖同一版本，以便该版本在这些组件之间同步。除非您确定如何处理版本，否则最好是将所有蓝图提供的包保持在同一最新版本（`0.0.x`）下。以下示例说明如何配置 `package.json` 文件，以使所有依赖项都依赖同一版本：

```
...
"@caws-blueprint-component/caws-environments": "^0.1.12345",
"@caws-blueprint-component/caws-source-repositories": "^0.1.12345",
"@caws-blueprint-component/caws-workflows": "^0.1.12345",
"@caws-blueprint-component/caws-workspaces": "^0.1.12345",
"@caws-blueprint-util/blueprint-utils": "^0.1.12345",
...
"@caws-blueprint/blueprints.blueprint": "*",
```

为所有依赖项配置版本后，请使用以下命令：

```
yarn install
```

使用 yarn 和 npm

蓝图使用 Yarn 作为工具。使用 npm 和 Yarn 将导致出现工具问题，因为两者解析依赖项树的方式不同。为了避免出现此类问题，最好是仅使用 Yarn。

如果您意外地使用 npm 安装了依赖项，则可以移除生成的 `package-lock.json` 文件，并确保使用所需的依赖项更新 `.projenrc.ts` 文件。您可以使用 Projen 重新生成项目的配置。

使用以下命令可从模型中重新生成：

```
yarn projen
```

在确保使用必要的依赖项更新 .projenrc.ts 文件后，使用以下命令：

```
yarn
```

升级工具和组件

有时，您可能需要升级工具和组件来引入可用的新功能。除非您确定了如何处理版本，否则建议您将所有组件保持在同一版本。版本在组件之间进行同步，因此让所有组件使用同一版本可确保它们之间存在适当的依赖项。

使用 Yarn 工作区 monorepo

使用以下命令可从自定义蓝图的存储库的根目录升级实用工具和组件：

```
yarn upgrade @amazon-codecatalyst/*
```

如果您未使用 monorepo，请使用以下命令：

```
yarn upgrade –pattern @amazon-codecatalyst/*
```

可用于升级工具和组件的其他选项：

- 使用 npm 视图 `@caws-blueprint-component/<some-component>` 获取最新版本。
- 通过在 package.json 文件中设置版本并使用以下命令来手动升级到最新版本：`yarn`。所有组件和实用工具都应具有同一版本。

改进

蓝图软件开发工具包（SDK）是您可以做出贡献的开源库。作为贡献者，请考虑贡献指南、反馈和缺陷。有关更多信息，请参阅 blueprints GitHub repository。

中的蓝图配额 CodeCatalyst

下表描述了 Amazon CodeCatalyst 中蓝图的配额和限制。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

每个项目应用的最大蓝图数量 CodeCatalyst	100

使用源存储库存储代码并协作处理代码 CodeCatalyst

CodeCatalyst 源存储库是托管在亚马逊的 Git 存储库 CodeCatalyst。您可以使用中的源存储库 CodeCatalyst 来安全地存储、版本和管理项目的资产。

CodeCatalyst 存储库中的资产可以包括：

- 文档
- 源代码
- 二进制文件

CodeCatalyst 还使用项目的源存储库来存储项目的配置信息，例如工作流程配置文件。

一个 CodeCatalyst 项目中可以有多个源存储库。例如，您可能希望为前端源代码、后端源代码、实用程序和文档设置单独的源存储库。

以下是处理源代码库、拉取请求和开发环境中代码的一种可能的工作流程 CodeCatalyst：

Mary Major CodeCatalyst 使用蓝图创建了一个 Web 应用程序项目，该蓝图创建了一个包含示例代码的源存储库。她邀请自己的朋友 Li Juan、Saanvi Sarkar 和 Jorge Souza 一起参与这个项目。Li Juan 查看了源存储库中的示例代码，决定简单地进行一些更改，以便在代码中添加测试。Li 创建了一个开发环境，选择 AWS Cloud9 作为 IDE，然后指定一个新分支*`test-code`*。开发环境打开。Li 快速添加代码，然后提交分支并将其推送到源存储库中 CodeCatalyst。接下来，Li 创建了一个拉取请求。在创建该拉取请求时，Li 将 Jorge Souza 和 Saanvi Sarkar 添加为审阅者，以确保代码得到审查。

在查看代码时，Jorge Souza记得他有自己的项目存储库 GitHub ，其中包含他们正在开发的应用程序的原型。他让 Mary Major 安装和配置扩展程序，使他能够将 GitHub 存储库作为额外的源存储库链接到项目。Mary 查看了存储库 GitHub 并与 Jorge 合作配置了 GitHub 扩展，这样他就可以将 GitHub 存储库链接为该项目的额外源存储库。

CodeCatalyst 源代码库支持 Git 的标准功能，可与现有的基于 Git 的工具配合使用。在 Git 客户端或集成开发环境 (PATs) 中克隆和使用源存储库时，您可以创建和使用个人访问令牌 (IDEs) 作为应用程序特定的密码。 PATs 它们与您的 CodeCatalyst用户身份相关联。有关更多信息，请参阅 使用个人访问令牌向用户授予对存储库的访问权限。

CodeCatalyst 源存储库支持拉取请求。这是一种简单的方法，让您和其他项目成员可在将代码更改从一个分支合并到另一个分支之前查看和评论代码更改。您可以在 CodeCatalyst 控制台中查看更改并评论代码行。

推送到 CodeCatalyst 源存储库中的分支可以自动启动工作流程中的运行，在该工作流程中可以构建、测试和部署更改。如果您的源存储库是作为使用项目模板的项目的一部分创建的，则会为您配置一个或多个工作流作为项目的一部分。您可以随时为存储库添加其他工作流。项目中工作流的 YAML 配置文件存储到的源存储库，是在源操作中为这些工作流配置的源存储库。有关更多信息，请参阅 入门工作流。

主题

源存储库概念

以下是您在使用 CodeCatalyst 源存储库时需要了解的一些概念。

主题

Projects

项目代表一种为开发团队和任务提供支持的协作努力。 CodeCatalyst 在拥有项目后，可以添加、更新或移除用户和资源，自定义您的项目控制面板以及监控团队的工作进度。一个空间内可以有多个项目。

源存储库特定于您在空间中创建或链接源存储库的项目。您不能在项目之间共享存储库，也不能将一个存储库链接到一个空间中的多个项目。在项目中具有贡献者或项目管理员角色的用户，可以根据授予给这些角色的权限，与该项目关联的源存储库进行交互。有关更多信息，请参阅 使用用户角色授予访问权限。

源存储库

源存储库用于安全地存储项目的代码和文件。它还存储文件的版本历史记录。默认情况下，源存储库与 CodeCatalyst 项目中的其他用户共享。一个项目可以有多个源存储库。您可以为中的项目创建源存储库 CodeCatalyst，也可以选择链接其他服务托管的现有源存储库（如果已安装的扩展程序支持该服务）。例如，在安装 GitHub 存储库扩展之后，您可以将GitHub 存储库链接到项目。有关更多信息，请参阅将源代码存储在项目的存储库中 CodeCatalyst和快速入门：安装扩展、连接提供商和链接资源 CodeCatalyst。

开发环境

开发环境是一种基于云的开发环境，您可以使用它 CodeCatalyst 来快速处理存储在项目源存储库中的代码。开发环境中包含的项目工具和应用程序库由项目的源存储库中的 devfile 定义。如果您的源存储库中没有 devfile，系统会自动应用默认的 devfile。默认 devfile 包括适用于最常用的编程语言和框架的工具。默认情况下，为开发环境配备了双核处理器、4 GB RAM 和 16 GiB 持久性存储。

您可以选择将源存储库的现有分支克隆到开发环境中，也可以选择在创建开发环境时创建一个新分支。

个人访问令牌 (PATs)

个人访问令牌（PAT）类似于密码。它与您的用户身份相关联，可在中的所有空间和项目中使用 CodeCatalyst。您可以使用 PATs 访问包括集成开发环境 (IDEs) 和基于 Git 的源存储库在内的 CodeCatalyst 资源。 PATs 代表你 CodeCatalyst ，你可以在用户设置中对其进行管理。一个用户可以有多个 PAT。个人访问令牌仅显示一次。最佳实践是，务必将 PAT 安全地存储在本地计算机上。默认情况下，一年后 PATs 过期。

使用集成开发环境 (IDEs) 时，等同 PATs 于 Git 密码。在设置 IDE 与 Git 存储库协同工作时，如果要求输入密码，请提供 PAT。有关如何将 IDE 与基于 Git 的存储库连接的更多信息，请参阅 IDE 的文档。

Branches

分支是指向 Git 和中的提交的指针或引用 CodeCatalyst。您可以使用分支来组织您的工作。例如，您可以使用分支来处理文件的新版本或不同版本，而不会影响其他分支中的文件。您可以使用分支来开发新功能、存储项目的特定版本等。一个源存储库可以有一个或多个分支。使用模板创建项目时，为项目创建的源存储库会在名为 main 的分支中包含示例文件。main 分支是存储库的默认分支。

默认分支

无论您如何创建，中的源存储库都 CodeCatalyst 有一个默认分支。如果您选择使用模板创建项目，在为该项目创建的源存储库中，除了示例代码、工作流定义和其他资源以外，还包括一个 README.md 文件。如果创建源存储库时没有使用模板，则会在第一次提交时为您添加 README.md 文件，并在创建源存储库时为您创建默认分支。该默认分支名为 main。此默认分支在用户克隆存储库时被用作本地存储库的基本或默认分支。您可以更改将哪个分支用作默认分支。有关更多信息，请参阅 管理存储库的默认分支。

您不能删除源存储库的默认分支。搜索结果只包括默认分支的结果。

提交

提交是对一个或一组文件所做的更改。在 Amazon CodeCatalyst 控制台中，提交会保存您的更改并将其推送到源存储库。提交包含有关更改的信息，包括执行了更改的用户的身份、更改的时间和日期、提交标题以及包含的有关更改的任何消息。有关更多信息，请参阅 通过在 Amazon 中提交来了解源代码的变化 CodeCatalyst。

在中的源存储库的上下文中 CodeCatalyst，提交是存储库内容和内容更改的快照。您还可以为提交添加 Git 标签，用于标识特定提交。

拉取请求

拉取请求是您和其他用户在源存储库中审查、评论和合并从一个分支到另一个分支的代码更改的主要方式。您可以使用拉取请求来协同审查代码更改，包括次要的更改或修复、主要功能添加或已发布软件的新版本。在拉取请求中，您可以查看源分支和目标分支之间的更改，或这些分支修订版之间的差异。您可以为单行代码更改添加注释，也可以为整个拉取请求添加注释。

> **Tip**
>
> 在创建拉取请求时，显示的差异是源分支最新块与目标分支最新块之间的差异。创建拉取请求后，显示的差异将是您选择的拉取请求修订版与创建拉取请求时目标分支最新块的提交之间的差异。有关 Git 中的差异和合并基础的更多信息，请参阅 Git 文档git-merge-base中的。

修订

修订版是拉取请求的更新版本。每次向拉取请求的源分支推送，都会创建一个修订版，其中包含该推送中包含的提交所做的更改。除了源分支和目标分支之间的差异外，您还可以查看拉取请求修订版之间的差异。有关更多信息，请参阅 在 Amazon 中使用拉取请求查看代码 CodeCatalyst。

工作流

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

源存储库还可以存储工作流、通知、事务的配置文件和其他信息，以及项目的其他配置信息。在创建需要配置文件的资源时，或将存储库指定为工作流的源操作时，会创建配置文件并将其存储在源存储库中。如果根据蓝图创建项目，配置文件已作为项目的一部分，存储在为您创建的源存储库中。这些配置信息存储在存储库默认分支中名为 `.codecatalyst` 的文件夹中。在创建默认分支的分支时，除了该分支中的所有其他文件和文件夹之外，还会创建该文件夹及其配置的副本。

为使用源存储库进行设置

当您在本地计算机 CodeCatalyst 上使用 Amazon 中的源存储库时，您可以单独使用 Git 或在支持的集成开发环境 (IDE) 中进行代码更改以及推送和拉取代码。作为最佳做法，我们建议您使用最新版本的 Git 和其他软件。

> **Note**
>
> 如果您使用开发环境，则不必安装 Git。您的开发环境中包含最新版本的 Git。

的版本兼容性信息 CodeCatalyst

组件	版本
Git	最新

安装 Git

要在没有 IDE 的情况下通过 Git 客户端处理源存储库中的文件、提交、分支和其他信息，请在本地计算机上安装 Git。

要安装 Git，建议您访问 Git 下载等网站。

创建个人访问令牌

要将源存储库克隆到本地计算机或首选的 IDE，必须创建个人访问令牌（PAT）。

创建个人访问令牌（PAT）

1. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

2. 在 PAT 名称中，为您的 PAT 输入描述性名称。
3. 在到期日期中，保留默认日期，或者选择日历图标以自定义日期。到期日期默认为从当前日期起一年后。
4. 选择创建。

   当为源存储库选择克隆存储库时，也可以创建此令牌。
5. 将 PAT 密钥保存在安全位置。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后将无法再检索该密钥。

开始使用 CodeCatalyst 源存储库和单页应用程序蓝图

按照本教程中的步骤学习如何在 Amazon 中使用源存储库 CodeCatalyst。

要开始使用 Amazon 中的源存储库，最快的方法 CodeCatalyst 是使用模板创建项目。使用模板创建项目时，系统会为您创建资源，包括其中含有示例代码的源存储库。您可以使用此存储库和代码示例来了解如何执行以下操作：

- 查看项目的源存储库并浏览其内容
- 创建一个带有新分支的开发环境，在其中开发代码
- 更改文件，提交并推送更改内容
- 创建拉取请求，并与其他项目成员一起审查代码更改
- 查看自动构建和测试拉取请求源分支中的更改的项目工作流
- 将源分支中的更改合并到目标分支，并关闭拉取请求
- 查看自动构建和部署的合并更改

要充分学习本教程的内容，请邀请他人加入您的项目，以便共同完成拉取请求。您还可以在中探索其他功能 CodeCatalyst，例如创建议题并将其与拉取请求关联，或者配置通知并在关联的工作流程运行时收到提醒。有关全面的探索 CodeCatalyst，请参阅入门教程。

使用蓝图创建项目

要协同工作，首先要创建项目。您可以使用蓝图来创建项目，这还会创建一个包含示例代码的源存储库和一个工作流，当您更改代码时，工作流会自动构建和部署您的代码。在本教程中，我们将指导您完成一个使用单页应用程序蓝图创建的项目，不过您也可以对任何有源存储库的项目使用这些步骤。在创建项目时，确保选择一个 IAM 角色或添加一个 IAM 角色（如果没有）。我们建议您在此项目中使用 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。

如果您已经有一个项目，可以跳到查看项目的存储库。

> **Note**
>
> 只有拥有 Space 管理员或 Power 用户角色的用户才能在中创建项目 CodeCatalyst。如果您没有此角色，但需要一个项目来完成本教程，请让具有这些角色之一的人员为您创建一个项目，并将您添加到已创建的项目中。有关更多信息，请参阅 使用用户角色授予访问权限。

使用蓝图创建项目

1. 在 CodeCatalyst 控制台中，导航到要在其中创建项目的空间。
2. 在空间控制面板上，选择创建项目。
3. 选择从蓝图开始。

> **Tip**
>
> 您可以选择向 Amazon Q 提供您的项目需求，让 Amazon Q 为您推荐蓝图，以此来添加蓝图。有关更多信息，请参阅创建项目或添加功能时使用 Amazon Q 选择蓝图和使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践。此功能仅在美国西部（俄勒冈州）区域中提供。
>
> 此功能要求为空间启用生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。

4. 在CodeCatalyst 蓝图或空间蓝图选项卡中，选择蓝图，然后选择下一步。
5. 在命名项目下，输入要分配给项目的名称及其关联资源名称。该名称在空间内必须是唯一的。
6. （可选）默认情况下，蓝图创建的源代码存储在存储 CodeCatalyst 库中。此外，您可以选择将蓝图源代码存储在第三方存储库中。有关更多信息，请参阅 为带有扩展程序的项目添加功能 CodeCatalyst。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
>
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

根据要使用的第三方存储库提供商，执行以下操作之一：

- GitHub 存储库：Connect GitHub 账户。

  选择 “高级” 下拉菜单，选择 GitHub作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitHub 帐户。

  > **Note**
  >
  > 如果您要关联 GitHub 账户，则必须创建个人连接才能在您的身份和身份之间建立 CodeCatalyst 身份映射。 GitHub 有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

- Bitbucket 存储库：连接 Bitbucket 工作区。

  选择高级下拉菜单，选择 Bitbucket 作为存储库提供商，然后选择用于存储蓝图所创建的源代码的 Bitbucket 工作区。

- GitLab 存储库：Connect GitLab 用户。

  选择 “高级” 下拉菜单，选择 GitLab作为存储库提供者，然后选择要存储蓝图创建的源代码的 GitLab 用户。

7. 在项目资源下，配置蓝图参数。根据蓝图的不同，您可以选择命名源存储库名称。
8. （可选）要根据您选择的项目参数查看包含更新的定义文件，请从生成项目预览中选择查看代码或查看工作流。
9. （可选）从蓝图卡中选择查看详细信息，查看蓝图的具体详细信息，如蓝图架构概述、所需连接和权限，以及蓝图创建的资源种类。
10. 选择创建项目。

创建项目或接受项目邀请并完成登录过程后，项目概述页面就会打开。新项目的项目概述页面不包含待解决事务或拉取请求。您可以选择性创建一个事务并将其分配给自己。您也可以选择邀请其他人加入您的项目。有关更多信息，请参阅在中创建问题 CodeCatalyst和邀请用户加入项目。

查看项目的存储库

作为项目成员，您可以查看项目的源存储库。您还可以选择创建其他存储库。如果拥有 Space 管理员角色的用户安装并配置了GitHub 存储库、Bitbucket 存储库或GitLab 存储库扩展，您还可以在为扩展

程序配置的 GitHub 账户、Bitbucket 工作空间或 GitLab 用户中添加指向第三方存储库的链接。有关更多信息，请参阅创建源存储库和快速入门：安装扩展、连接提供商和链接资源 CodeCatalyst。

> **Note**
>
> 对于使用单页应用程序蓝图创建的项目，包含示例代码的源存储库的默认名称为***spa-app***。

导航到项目的源存储库

1. 导航到您的项目，然后执行下列操作之一：
   - 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。
   - 在导航窗格中，选择代码，然后选择源存储库。在源存储库中，从列表中选择存储库的名称。您可以通过在筛选栏中键入部分存储库名称，筛选存储库列表。
2. 在存储库主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求和工作流的数量。默认情况下，会显示默认分支的内容。您可通过选择此下拉列表中的其他分支来更改视图。

存储库的概述页面包含有关为该存储库的分支及其文件配置的工作流和拉取请求的信息。如果您刚刚创建项目，那么构建、测试和部署代码的初始工作流仍在运行，因为它们需要几分钟才能完成。您可以通过选择相关工作流下方的数字来查看相关工作流及其状态，但这将在 CI/CD 中打开工作流页面。在本教程中，请继续浏览概述页面，并探索存储库中的代码。`README.md` 文件的内容会显示在该页面上的存储库文件下方。在文件中，显示默认分支的内容。如果有其他分支，您可以更改文件视图以显示其他分支的内容。`.codecatalyst` 文件夹包含用于项目其他部分的代码，如工作流 YAML 文件。

要查看文件夹的内容，请选择文件夹名称旁边的箭头展开文件夹。例如，选择 `src` 旁边的箭头可查看该文件夹中包含的单页 Web 应用程序的文件。要查看某个文件的内容，请从列表中选择该文件。这将打开查看文件，您可以在其中浏览多个文件的内容。您也可以在控制台中编辑单个文件，但要编辑多个文件，建议您创建一个开发环境。

创建开发环境

您可以在 Amazon CodeCatalyst 控制台中添加和更改源存储库中的文件。不过，要高效处理多个文件和分支，我们建议使用开发环境或将存储库克隆到本地计算机。在本教程中，我们将创建一个带有名为的分支的 AWS Cloud9 开发环境**`develop`**。您可以选择不同的分支名称，但只要将分支命名为**`develop`**，在本教程稍后创建拉取请求时，工作流就会自动运行，以构建和测试您的代码。

> **Tip**
>
> 如果您决定在本地克隆一个存储库，而不是使用开发环境，请确保您的本地计算机上有 Git 或您的 IDE 包含 Git。有关更多信息，请参阅 为使用源存储库进行设置。

使用新分支创建开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建开发环境的项目。
3. 从项目的源存储库列表中选择存储库的名称。另外，在导航窗格中，依次选择代码、源存储库以及要为其创建开发环境的存储库。
4. 在存储库主页上，选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 选择要克隆的存储库，选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支自下拉菜单中选择要从中创建新分支的分支。
7. 可选操作，为开发环境添加别名。
8. 可选操作，选择开发环境配置编辑按钮，编辑开发环境的计算、存储或超时配置。
9. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动，开发环境创建完成后，状态列将显示正在运行。将在您选择的 IDE 中打开一个新选项卡，其中包含您的开发环境。您可以编辑代码并提交和推送更改。

创建开发环境后，您就可以编辑文件、提交更改并将更改推送到 **`test`** 分支。在本教程中，请编辑 `src` 文件夹中 `App.tsx` 文件的 <p> 标签之间的内容，以更改网页上显示的文本。提交并推送您的更改，然后返回 CodeCatalyst 选项卡。

在 AWS Cloud9 开发环境中进行和推送更改

1. 在中 AWS Cloud9，展开侧面导航菜单以浏览文件。展开 `src`，然后打开 `App.tsx`。
2. 更改 <p> 标签内的文本。
3. 保存文件，然后使用 Git 菜单提交并推送更改。或者，在终端窗口中使用 git commit 和 git push 命令提交和推送更改。

```
git commit -am "Making an example change"
git push
```

> **Tip**
>
> 您可能需要将终端中的目录更改为 Git 存储库目录，然后才能成功运行 Git 命令。

创建拉取请求

您可以使用拉取请求来协同审查代码更改，包括次要的更改或修复、主要功能添加或已发布软件的新版本。在本教程中，您将创建一个拉取请求，以查看与主`test`分支相比，您对分支所做的更改。在使用模板创建的项目中创建拉取请求也会启动相关工作流（如果有的话）的运行。

创建拉取请求

1. 导航到您的项目。
2. 请执行以下操作之一：
   - 在导航窗格中，依次选择代码、拉取请求和创建拉取请求。
   - 在存储库主页上，选择更多，然后选择创建拉取请求。
   - 在项目页面上，选择创建拉取请求。
3. 在源代码库中，确保指定的源存储库是包含所提交代码的存储库。只有在您未从存储库的主页创建拉取请求时，才会显示此选项。
4. 在目标分支中，在审查代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含所提交代码的分支。
6. 在拉取请求标题中，输入一个标题，帮助其他用户了解需要审查的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如事务链接或所做更改的描述之类的信息。

   > **Tip**
   >
   > 你可以选择 “为我写描述”， CodeCatalyst 自动生成拉取请求中包含的更改的描述。您可以在将自动生成的描述添加到拉取请求后，对描述进行更改。
   > 此功能要求为空间启用生成式人工智能功能，并且不适用于链接的存储库中的拉取请求。
   > 有关更多信息，请参阅 Managing generative AI features。

8. （可选）在事务中，选择关联事务，然后从列表中选择事务或输入事务 ID。要取消链接事务，请选择“取消链接”图标。

9. （可选）在必需的审阅者中，选择添加必需审阅者。从项目成员列表中进行选择，添加审阅者。必需的审阅者必须批准更改，才能将拉取请求合并到目标分支。

> **Note**
>
> 您不能将审阅者同时添加为必需的审阅者和可选的审阅者。您不能将自己添加为审阅者。

10. （可选）在可选的审阅者中，选择添加可选审阅者。从项目成员列表中进行选择，添加审阅者。可选的审阅者不必批准更改，这并不是将拉取请求合并到目标分支之前的必备要求。
11. 审查分支之间的差异。拉取请求中显示的差异是源分支中的修订与合并基准之间的更改，而合并基准是创建拉取请求时目标分支的 HEAD 提交。如果未显示任何更改，则分支可能相同，或者您可能为源和目标选择了相同的分支。
12. 如果您对拉取请求中包含的希望审查的代码和更改感到满意，请选择创建。

> **Note**
>
> 创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

您可以通过选择概述，然后查看工作流运行下拉取请求详细信息区域中的信息，来查看创建此拉取请求所启动的相关工作流的信息。要查看工作流运行的情况，请选择该运行。

> **Tip**
>
> 如果您将分支命名为 **`develop`** 以外的名称，工作流将不会自动运行以构建和测试您的更改。如果要对其进行配置，请编辑onPullRequestBuildAndTest工作流程的 YAML 文件。有关更多信息，请参阅 创建工作流。

您可以对此拉取请求发表评论，也可以要求其他项目成员发表评论。您还可以选择添加或更改可选或必需的审阅者。您可以选择对存储库的源分支进行更多更改，并查看这些已提交的更改是如何创建拉取请求修订版的。有关更多信息，请参阅审核拉取请求、更新拉取请求、在 Amazon 中使用拉取请求查看代码 CodeCatalyst和查看工作流运行状态和详细信息。

合并拉取请求

拉取请求经过审核并获得所需审阅者的批准后，即可在 CodeCatalyst控制台中将其源分支合并到目标分支。合并拉取请求也会通过与目标分支相关联的工作流启动更改的运行。在本教程中，您将把测试分支合并到主分支中，这将开始onPushToMainDeployPipeline工作流程的运行。

合并拉取请求（控制台）

1. 在拉取请求中，选择您在上一步中创建的拉取请求。在拉取请求中，选择合并。
2. 从拉取请求的可用合并策略中选择。（可选）选择或取消选择在合并拉取请求后删除源分支的选项，然后选择合并。合并完成后，拉取请求的状态将变为已合并，并不再出现在拉取请求的默认视图中。默认视图显示状态为待解决的拉取请求。您仍然可以查看已合并的拉取请求，但不能审批该拉取请求或更改其状态。

> **Note**
>
> 如果 “合并” 按钮未激活，或者您看到 “不可合并” 标签，则可能是所需的审阅者尚未批准拉取请求，或者拉取请求无法在控制台中合并。 CodeCatalyst 在拉取请求详细信息区域的概述中，使用时钟图标来指示尚未批准拉取请求的审阅者。如果所有必需的审阅者都已批准了拉取请求，但合并按钮仍未激活，则可能存在合并冲突，或超出了空间的存储配额。您可以在开发环境中解决目标分支的合并冲突，推送更改，然后合并拉取请求，也可以解决冲突并在本地合并，然后将包含合并的提交推送到 CodeCatalyst。有关更多信息，请参阅合并拉取请求（Git）和您的 Git 文档。

查看已部署的代码

现在，您可以查看默认分支中最初部署的代码，以及自动构建、测试和部署后合并的更改。为此，您可以返回存储库的概述页面，选择相关工作流图标旁边的数字，或在导航窗格中选择 CI/CD，然后选择工作流。

查看已部署的代码

1. 在工作流的 `onPushToMainDeployPipeline` 中，展开最近的运行。

> **Note**
>
> 这是使用单页应用程序蓝图创建的项目的工作流默认名称。

2. 最近的运行是由您合并到 `main` 分支的拉取请求提交所启动的运行，可能会显示正在进行中状态。从列表中选择成功完成的运行，打开该运行的详细信息。
3. 选择变量。复制 AppURL 的值。这是已部署的单页 Web 应用程序的 URL。打开一个新的浏览器标签页并粘贴该值，以查看已构建和部署的代码。保持标签页打开状态。
4. 返回工作流运行的列表，等待最近的运行完成。运行完成后，返回您打开的标签页，以查看 Web 应用程序并刷新浏览器。您应该能在合并的拉取请求中看到所做的更改。

清理资源

在探索了如何使用源存储库和拉取请求后，建议您删除任何不需要的资源。您不能删除拉取请求，但可以关闭它们。您可以删除任何已创建的分支。

如果您不再需要源存储库或项目，也可以删除这些资源。有关更多信息，请参阅删除源存储库和删除项目。

将源代码存储在项目的存储库中 CodeCatalyst

源存储库用于安全地存储项目的代码和文件。源存储库还储存从首次提交到最新更改的源历史记录。如果您选择包含源存储库的蓝图，则该存储库还包含项目工作流和通知的配置文件和其他信息。这些配置信息存储在名为 .codecatalyst 的文件夹中。

您可以在中创建源存储库，方法 CodeCatalyst 是创建带有蓝图的项目，该蓝图将在创建项目时创建源存储库，也可以通过在现有项目中创建源存储库。项目用户将自动查看并能够使用您为项目创建的存储库。您也可以选择将托管在 Bibucket 上 GitHub的 Git 存储库或 GitLab 您的项目关联起来。当您这样做时，您的项目用户可以在项目存储库列表中查看和访问该链接的存储库。

> **Note**
>
> 在链接存储库之前，必须为托管该存储库的服务安装扩展。您无法链接已存档的存储库。虽然你可以链接一个空存储库，但在你使用创建默认分支的初始提交对其进行初始化 CodeCatalyst 之前，你不能在中使用它。有关更多信息，请参阅 在空间中安装扩展。

默认情况下，源存储库与您的 Amazon CodeCatalyst 项目的其他成员共享。您可以为项目创建其他源存储库或将存储库链接到项目。项目的所有成员都可以查看、添加、编辑和删除项目源存储库中的文件和文件夹。

要快速处理源存储库中的代码，您可以创建一个开发环境来克隆指定的存储库和分支，以便在为开发环境选择的集成式开发环境（IDE）中处理代码。您可以在本地计算机上克隆源存储库，然后在本地存储库和远程存储库之间拉取和推送更改。 CodeCatalyst您也可以通过在首选 IDE 中配置对源存储库的访问权限来使用该存储库，前提是该 IDE 支持凭证管理。

存储库名称在 CodeCatalyst 项目中必须是唯一的。

主题

创建源存储库

当您在 Amazon 中使用蓝图创建项目时 CodeCatalyst， CodeCatalyst 会为您创建一个源存储库。除了为您创建的工作流和其他资源的配置信息外，该源存储库还包含示例代码。这是开始使用中的存储库的推荐方法 CodeCatalyst。您可以选择为项目创建存储库。这些存储库将包含一个 README.md 文件，您可以随时编辑或删除该文件。根据您在创建源存储库时的选择，存储库可能还包含一个 `.gitignore` 文件。

如果要将现有 Git 仓库克隆到 CodeCatalyst 源仓库中，可以考虑改为创建一个空仓库。在您向其中添加内容 CodeCatalyst 之前，该存储库将无法在中使用，只需几个简单的 Git 命令即可完成此操作。或者，您可以直接从 CodeCatalyst 控制台向空存储库添加内容。或者，您可以在支持的 Git 存储库提供程序中链接源存储库。有关更多信息，请参阅 链接源存储库。

创建源存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。

5. 在存储库名称中，输入存储库的名称。在本指南中，我们使用*codecatalyst-source-repository*，但您可以选择其他名称。一个项目中的存储库名称必须唯一。有关存储库名称要求的更多信息，请参阅中的源存储库配额 CodeCatalyst。
6. （可选）在描述中，添加对存储库的描述，这有助于项目中的其他用户了解存储库的用途。
7. 选择创建存储库（默认）。此选项会创建一个存储库，其中包含默认分支和 README.md 文件。与空存储库不同，您可以在创建此存储库后立即使用它。
8. 在默认分支中，除非您有理由选择其他名称，否则请将该名称保留为 main。本指南中的示例对于默认分支均使用名称 main。
9. （可选）为您计划推送的代码类型添加一个 `.gitignore` 文件。
10. 选择创建。

> **Note**
>
> CodeCatalyst 在创建存储库时将`README.md`文件添加到存储库中。 CodeCatalyst还会在名为 main 的默认分支中为存储库创建初始提交。您可以编辑或删除 README.md 文件，但无法删除默认分支。

创建空的源存储库

1. 在 CodeCatalyst 控制台中，导航到要在其中创建空存储库的项目。
2. 在项目的摘要页面上，在源存储库中，选择添加存储库，然后选择创建存储库。或者，在导航窗格中，选择代码，然后选择源存储库。选择添加存储库，然后选择创建存储库。
3. 在存储库名称中，输入存储库的名称。在本指南中，我们使用*codecatalyst-source-repository*，但您可以选择其他名称。一个项目中的存储库名称必须唯一。有关存储库名称要求的更多信息，请参阅中的源存储库配额 CodeCatalyst。
4. （可选）在描述中，添加对存储库的描述，这有助于项目中的其他用户了解存储库的用途。
5. 选择创建空存储库，然后选择创建。

将现有 Git 存储库克隆到源存储库中

您可以将现有 Git 存储库克隆到 Amazon 中的空源存储库 CodeCatalyst。这是一种快速入门的方法，可以快速开始 CodeCatalyst 使用以前托管在另一个 Git 存储库提供程序中的代码。您可以通过创建镜像克隆然后将镜像推送到来克隆存储库的内容 CodeCatalyst。或者，如果您有想要添加内容的存储库的本地存储库 CodeCatalyst，则可以将 CodeCatalyst 源存储库作为另一个远程存储库添加到本地存储

库，然后推送到空源存储库。这两种方法同样有效。使用镜像克隆不仅能映射分支，还能映射所有引用。这是一种在中创建存储库工作副本的简单而干净的方法 CodeCatalyst。向指向空 CodeCatalyst 源存储库的本地存储库添加远程存储库会将仓库内容添加到 CodeCatalyst，但它也允许您从本地存储库推送到 CodeCatalyst 源存储库和原始 Git 远程存储库。如果您想在不同的远程存储库中维护代码，这可能很有用，但如果其他开发人员只向其中一个远程存储库提交代码，则可能会导致冲突。

以下过程使用基本的 Git 命令来完成这项任务。在 Git 中完成任务有很多方法，包括克隆。有关更多信息，请参阅 Git 文档。

> **Important**
>
> 必须 CodeCatalyst 先在中创建一个空存储库，然后才能将内容克隆到该存储库。您还必须拥有个人访问令牌。有关更多信息，请参阅创建空的源存储库和创建个人访问令牌。

用于`git clone --mirror`将现有 Git 存储库克隆到 CodeCatalyst

1. 在 CodeCatalyst 控制台中，导航到您在其中创建空存储库的项目。
2. 在项目的摘要页面上，从列表中选择空存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择空存储库的名称。
3. 复制空存储库的 HTTPS 克隆 URL。您在推送镜像克隆时需要此 URL。例如，如果您在 ExampleCorp 空间中为 MyExampleProject 项目 MyExampleRepo 中的源存储库命名，而您的用户名为 LiJuan，则您的克隆 URL 可能如下所示：

   ```
   https://LiJuan@git.us-west-2.codecatalyst.aws/
   v1/ExampleCorp/MyExampleProject/MyExampleRepo
   ```

4. 在命令行或终端窗口中，使用`git clone --mirror`命令创建要克隆到的 Git 存储库的镜像克隆 CodeCatalyst。例如，如果要在中创建 codetalyst-blueprints 存储库的镜像克隆 GitHub，则需要输入以下命令：

   ```
   git clone --mirror https://github.com/aws/codecatalyst-blueprints.git
   ```

5. 切换到您执行克隆操作的目录。

   ```
   cd codecatalyst-blueprints.git
   ```

6. 运行git push命令，指定目标 CodeCatalyst 源存储库的 URL 和名称以及--all选项。（这是您在步骤 3 中复制的 URL。）例如：

```
git push https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo --all
```

添加远程存储库并将本地存储库推送到 CodeCatalyst

1. 在 CodeCatalyst 控制台中，导航到您在其中创建空存储库的项目。
2. 在项目的摘要页面上，从列表中选择空存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择空存储库的名称。
3. 复制空存储库的 HTTPS 克隆 URL。您在推送镜像克隆时需要此 URL。例如，如果您在 ExampleCorp 空间中为 MyExampleProject 项目 MyExampleRepo 中的源存储库命名，而您的用户名为 LiJuan，则您的克隆 URL 可能如下所示：

   ```
   https://LiJuan@git.us-west-2.codecatalyst.aws/
   v1/ExampleCorp/MyExampleProject/MyExampleRepo
   ```

4. 在命令行或终端窗口中，将目录更改为要推送到 CodeCatalyst的本地存储库。
5. 运行 git remote -v 命令查看本地存储库的现有远程存储库。例如，如果您要在美国东部（俄亥俄州）区域克隆名为 **MyDemoRepo** 的 AWS CodeCommit 存储库的本地存储库，命令输出可能如下所示：

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   如果您想继续使用存储库，可复制远程 URL。
6. 使用`git remote remove`命令移除 CodeCommit 存储库以获取 URLs Origin 并推送：

   ```
   git remote remove origin
   ```

7. 使用 git remot e add 命令将 CodeCatalyst 源存储库 URL 添加为本地存储库的远程获取和推送。例如：

   ```
   git remote add origin https://LiJuan@git.us-west-2.codecatalyst.aws/
   v1/ExampleCorp/MyExampleProject/MyExampleRepo
   ```

这会将 CodeCommit 仓库推送 URL 替换为 CodeCatalyst 源存储库 URL，但不会更改获取 URL。因此，如果你再次运行 git remote-v 命令，你会看到你现在正在从 CodeCommit 远程存储库中提取（获取）代码，但你被配置为将更改从本地存储库推送到 CodeCatalyst 源存储库：

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  https://LiJuan@git.us-west-2.codecatalyst.aws/v1/ExampleCorp/
MyExampleProject/MyExampleRepo (push)
```

如果您想使用以下`git remote set-url`命令推送到两个存储库，则可以选择添加回 CodeCommit 远程 URL：

```
git remote set-url --add --push origin https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
```

8. 运行 `git push` 命令将本地存储库推送到所有已配置的推送远程存储库。或者，运行 git push -u -origin 命令，指定 --all 选项将本地存储库同时推送到两个存储库。例如：

```
git push -u -origin --all
```

> **Tip**
>
> 根据 Git 版本的不同，--all 可能无法将本地存储库的所有分支推送到空存储库。您可能需要分别签出和推送每个分支。

链接源存储库

在将源存储库链接到项目时，如果您的空间安装了该 CodeCatalyst 扩展插件，则可以包括具有托管存储库的服务扩展程序的存储库。只有具有空间管理员角色的用户才能安装扩展。一旦安装了扩展，就可以链接到为该扩展访问而配置的存储库。有关更多信息，请参阅在空间中安装扩展或在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

> **Important**
>
> 安装存储库扩展后，您链接到的任何存储库都 CodeCatalyst 将对其代码进行索引和存储。CodeCatalyst这将使代码可在中 CodeCatalyst搜索。要更好地了解在中使用链接存储库时代码的数据保护 CodeCatalyst，请参阅 Amazon CodeCatalyst 用户指南中的数据保护。

您只能将存储库链接到空间中的一个项目。您无法链接已存档的存储库。虽然你可以链接一个空存储库，但在你使用创建默认分支的初始提交对其进行初始化 CodeCatalyst 之前，你不能在中使用它。此外：

- GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库只能链接到空间中的一个 CodeCatalyst 项目。
- 您不能将空仓库或已存档 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库与 CodeCatalyst 项目一起使用。
- 您不能链接与 GitLab 项目中 GitHub 仓库同名的存储库、Bitbucket 存储库或 CodeCatalyst 项目存储库。
- GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。
- Bitbucket 存储库扩展与 Bitbucket Data Center 存储库不兼容。
- GitLab 存储库扩展与 GitLab 自行管理的项目存储库不兼容。
- 您无法在已链接存储库中使用为我编写描述或汇总评论功能。这些功能仅在中的拉取请求中可用 CodeCatalyst。

虽然您可以作为贡献者链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目仓库，但您只能以 Space 管理员或项目管理员的身份取消第三方仓库的链接。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
>
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

链接源存储库

1. 导航到要链接存储库的项目。

> **Note**
>
> 在链接存储库之前，具有空间管理员角色的用户必须首先为托管存储库的提供程序安装扩展。有关更多信息，请参阅 在空间中安装扩展。

2. 在导航窗格中，选择代码，然后选择源存储库。
3. 选择添加存储库，然后选择链接存储库。
4. 从存储库提供程序下拉菜单中，选择以下第三方存储库提供商之一：GitHub或 Bitbucket。
5. 根据您已选择链接的第三方存储库提供程序，执行下列操作之一：

- GitHub 存储库：链接存储 GitHub 库。
  1. 从GitHub 账户下拉菜单中，选择包含要关联的存储库的 GitHub 账户。
  2. 从GitHub 存储库下拉菜单中，选择要关联 CodeCatalyst 项目的 GitHub 账户。
  3. （可选）如果您在 GitHub 存储库列表中看不到存储库，则可能未在的 Amazon CodeCatalyst 应用程序中将其配置为可以访问存储库 GitHub。您可以配置可在关联账户 CodeCatalyst 中使用哪些 GitHub 存储库。
     a. 导航到您的GitHub帐户，选择 “设置”，然后选择 “应用程序”。
     b. 在 “已安装的 GitHub 应用程序” 选项卡中，选择 Amazon CodeCatalyst 应用程序的配置。
     c. 执行以下任一操作来配置要链接的 GitHub 存储库的访问权限 CodeCatalyst：
        - 要提供对所有当前和未来存储库的访问权限，请选择所有存储库。
        - 要提供对特定存储库的访问权限，请选择 “仅选择存储库”，选择 “选择存储库” 下拉列表，然后选择要允许链接的存储库 CodeCatalyst。
- Bitbucket 存储库：链接 Bitbucket 存储库。
  1. 从 Bitbucket 工作区下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作区。
  2. 从 Bitbucket 存储库下拉菜单中，选择要关联项目的 Bitbucket 存储库。 CodeCatalyst

> **Tip**
>
> 如果存储库的名称显示为灰色，则无法链接该存储库，因为它已经链接到 Amazon CodeCatalyst 中的另一个项目。

6. 选择链接。

如果您不想再在中使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库 CodeCatalyst，则可以取消其与项目的链接。 CodeCatalyst 解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

查看源存储库

您可以在 Amazon 中查看与项目关联的源存储库 CodeCatalyst。对于中的源存储库 CodeCatalyst，存储库的概述页面提供了该存储库中信息和活动的快速概述，包括：

- 存储库的描述（如果有）
- 存储库中的分支数量
- 存储库的未处理拉取请求数量
- 存储库的相关工作流数量
- 默认分支或您选择的分支中的文件和文件夹
- 显示分支的最后一次提交的标题、作者和日期
- 以 Markdown 格式呈现的 README.md 文件的内容（如果包含任何 README.md 文件）

该页面还提供指向存储库的提交、分支和拉取请求的链接，以及打开、查看和编辑单个文件的快速方法。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中查看有关链接仓库的这些信息。要查看链接的存储库的相关信息，请在存储库列表中选择链接，在托管该存储库的服务中打开该存储库。

导航到项目的源存储库

1. 导航到您的项目，然后执行下列操作之一：
   - 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。
   - 在导航窗格中，选择代码，然后选择源存储库。在源存储库中，从列表中选择存储库的名称。您可以通过在筛选栏中键入部分存储库名称，筛选存储库列表。
2. 在存储库主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求和工作流的数量。默认情况下，会显示默认分支的内容。您可通过选择此下拉列表中的其他分支来更改视图。

> **Tip**
>
> 您还可以通过在项目摘要页面中选择查看项目代码来快速导航到项目的存储库。

编辑源存储库的设置

您可以管理存储库的设置，包括编辑存储库的描述、选择默认分支、创建和管理分支规则，以及创建和管理中拉取请求的批准规则 CodeCatalyst。这有助于项目成员了解存储库的用途，并帮助您执行团队使用的最佳实践和流程。

> **Note**
>
> 无法编辑源存储库的名称。
> 您无法在中编辑链接仓库的名称、描述或其他信息 CodeCatalyst。要修改有关链接的存储库的信息，必须在托管链接的存储库的提供程序中对其进行编辑。有关更多信息，请参阅托管链接存储库的服务的文档。

编辑存储库的设置

1. 在 CodeCatalyst 控制台中，导航到包含要编辑其设置的源存储库的项目。
2. 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择存储库的名称。
3. 在存储库的概述页面上，选择更多，然后选择管理设置。
4. 执行以下一个或多个操作：
   - 编辑存储库描述，然后选择保存。

- 要更改存储库的默认分支，请在默认分支中选择编辑。有关更多信息，请参阅 管理存储库的默认分支。
- 要添加、移除或更改关于哪些项目角色有权限在分支中执行某些操作的规则，请在分支规则中选择编辑。有关更多信息，请参阅 使用分支规则管理分支允许的操作。
- 要添加、移除或更改用于将拉取请求合并到分支的审批规则，请在审批规则中选择编辑。有关更多信息，请参阅 管理将拉取请求与审批规则合并的要求。

克隆源存储库

要有效处理源存储库中的多个文件、分支和提交，可将源存储库克隆到本地计算机，然后使用 Git 客户端或集成式开发环境（IDE）进行更改。提交您的更改并将其推送到源存储库，以便使用议题和拉取请求等 CodeCatalyst 功能。您还可以选择创建一个开发环境来处理代码。创建开发环境会自动将您指定的存储库和分支克隆到开发环境中。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中克隆链接存储库，也无法为其创建开发环境。要在本地克隆链接的存储库，请在存储库列表中选择链接，在托管该存储库的服务中打开该存储库，然后克隆它。有关更多信息，请参阅托管链接存储库的服务的文档。

从源存储库创建开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择代码，然后选择源存储库。
3. 选择要在其中处理代码的源存储库。
4. 选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 请执行以下操作之一：
   - 选择在现有分支中工作，然后从现有分支下拉菜单中选择一个分支。
   - 选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支的来源下拉菜单中选择要从中创建新分支的分支。
7. 还可选择为开发环境添加名称或编辑其配置。
8. 选择创建。

克隆源存储库

1. 导航到您的项目。
2. 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择存储库的名称。您可以通过在筛选栏中键入部分存储库名称，筛选存储库列表。
3.
4. 选择克隆存储库。复制存储库的克隆 URL。

> **Note**
>
> 如果您没有个人访问令牌（PAT），请选择创建令牌。复制令牌并将其保存在安全的位置。当 Git 客户端或集成式开发环境（IDE）提示输入密码时，您将使用此 PAT。

5. 请执行以下操作之一：
   - 要将存储库克隆到本地计算机，请打开终端或命令行并运行 git clone 命令，在命令后面加上克隆 URL。例如：

     ```
     git clone https://LiJuan@git.us-west-2.codecatalyst.aws/
     v1/ExampleCorp/MyExampleProject/MyExampleRepo
     ```

     提示输入密码时，请粘贴之前保存的 PAT。

     > **Note**
     >
     > 如果操作系统提供凭证管理功能或已安装凭证管理系统，则只需提供一次 PAT。否则，您可能需要为每次 Git 操作提供 PAT。作为最佳实践，请确保您的凭证管理系统能安全地存储您的 PAT。克隆 URL 字符串中不要包含 PAT。

   - 要使用 IDE 克隆存储库，请按照 IDE 的文档进行操作。选择克隆 Git 存储库并提供 URL 的选项。提示输入密码时，请提供 PAT。

删除源存储库

如果不再需要 Amazon CodeCatalyst 项目的源存储库，则可以将其删除。删除源存储库也会删除存储在存储库中的所有项目信息。如果有任何工作流依赖于源存储库，则在删除该存储库后，这些工作流将

从项目工作流列表中删除。引用源存储库的事务不会被删除或更改，但一旦删除源存储库，添加到事务中的任何指向源存储库的链接都将失效。

> **Important**
>
> 删除源存储库的操作无法撤销。删除源存储库后，就无法再克隆它、从中提取数据或向其推送数据。删除源存储库不会删除该存储库的任何本地副本（本地存储库）。要删除本地存储库，请使用您的本地计算机的目录和文件管理工具。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中删除链接的存储库。要删除链接的存储库，请在存储库列表中选择链接，在托管该存储库的服务中打开该存储库，然后删除它。有关更多信息，请参阅托管链接存储库的服务的文档。
>
> 要从项目中移除链接的存储库，请参阅取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

删除源存储库

1. 导航到包含要删除的源存储库的项目。
2. 在项目的摘要页面上，从列表中选择所需的存储库，然后选择查看存储库。或者，在导航窗格中，选择代码，然后选择源存储库。从项目的源存储库列表中选择存储库的名称。
3. 在存储库主页上，选择更多，选择管理设置，然后选择删除存储库。
4. 查看分支、拉取请求和相关工作流信息，以确保不会删除仍在使用或有未完成工作的存储库。如果要继续，请键入 delete，然后选择删除。

使用 Amazon 中的分支来整理源代码 CodeCatalyst

在 Git 中，分支是指向提交的指针或引用。在开发中，它们是组织工作的便捷方式。您可以使用分支来分离新的或不同版本文件的工作，而不影响其他分支中的工作。您可以使用分支来开发新功能、存储项目的特定版本等。您可以为源存储库中的分支配置规则，将分支上的某些操作限制到该项目中的特定角色。

无论您如何创建 CodeCatalyst ，Amazon 中的源存储库都有内容和默认分支。链接存储库可能没有默认分支或内容，但在初始化它们并创建默认分支 CodeCatalyst 之前无法使用。使用蓝图创建项目

时，CodeCatalyst 会为该项目创建一个源存储库，其中包含 README.md 文件、示例代码、工作流程定义和其他资源。在不使用蓝图的情况下创建源存储库时，在首次提交时，系统会为您添加一个 README.md 文件，并为您创建一个默认分支。该默认分支名为 main。此默认分支在用户克隆存储库时被用作本地存储库的基本或默认分支。

> **Note**
>
> 您不能删除默认分支。为源存储库创建的第一个分支是该存储库的默认分支。此外，搜索只会显示默认分支的结果。您无法在其他分支中搜索代码。

在中创建存储库 CodeCatalyst 还会创建第一次提交，这将创建一个包含一个 README.md 文件的默认分支。该默认分支的名称是 main。这是本指南的示例中使用的默认分支名称。

主题

创建分支

您可以使用 CodeCatalyst 控制台在 CodeCatalyst 存储库中创建分支。其他用户下次从存储库中提取更改时，就能看到您创建的分支。

> **Tip**
>
> 您还可以在创建开发环境的过程中创建分支来处理您的代码。有关更多信息，请参阅 创建开发环境。

您还可以使用 Git 创建分支。有关更多信息，请参阅 针对分支的常用 Git 命令。

创建分支（控制台）

1. 在 CodeCatalyst 控制台中，导航到源存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。
3. 选择要在其中创建分支的存储库。
4. 在存储库的概述页面上，选择更多，然后选择创建分支。
5. 输入分支的名称。
6. 选择要从中创建分支的分支，然后选择创建。

管理存储库的默认分支

您可以在 Amazon 的源存储库中指定哪个分支用作默认分支 CodeCatalyst。无论您如何创建，其中的所有源存储库 CodeCatalyst 都有内容和默认分支。如果您使用蓝图创建项目，为该项目创建的源存储库中的默认分支将命名为 main。默认分支的内容会自动显示在该存储库的概述页面上。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

默认分支的处理方式与源存储库中的所有其他分支略有不同。默认分支的名称旁边有一个特殊的标签，即默认。当用户使用 Git 客户端将存储库克隆到本地计算机时，默认分支是用作本地存储库的基础分支或默认分支的分支。默认分支也是创建工作流时用于存储工作流 YAML 文件和事务信息的默认位置。使用 search in 时 CodeCatalyst，仅搜索存储库的默认分支。由于默认分支对项目的许多方面都至关重要，因此您无法删除指定为默认分支的分支。但是，您可以选择使用不同的分支作为默认分支。如果这样做，任何应用于以前默认分支的分支规则都将自动应用于您指定为默认分支的分支。

> **Note**
>
> 您必须具有项目管理员角色才能更改 CodeCatalyst 项目中源存储库的默认分支。这不适用于链接的存储库。

查看和更改存储库的默认分支

1. 导航到存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要查看其设置的存储库，包括默认分支。
3. 在存储库的概述页面上，选择更多，然后选择管理设置。
4. 在默认分支中，会显示指定为默认分支的分支名称，并在名称旁边显示一个名为默认的标签。在分支中，同样的标签也会显示在分支列表中该分支名称的旁边。
5. 要更改默认分支，请选择编辑。

   > **Note**
   >
   > 您必须在项目中具有项目管理员角色，才能更改默认分支。

6. 从下拉列表中选择要设为默认分支的分支名称，然后选择保存。

使用分支规则管理分支允许的操作

创建分支时，根据该角色的权限，允许对该分支执行某些操作。您可以通过配置分支规则来更改特定分支允许执行的操作。分支规则基于用户在项目中的角色。您可以选择对某些预定义操作（如将提交推送到分支）进行限制，使得只有项目中具有特定角色的用户才能执行这些操作。通过限制允许哪些角色执行某些操作，有助于您保护项目中的特定分支。例如，如果配置了一个分支规则，只允许具有项目管理员角色的用户合并或推送到该分支，那么在项目中具有其他角色的用户将无法更改该分支中的代码。

您应该仔细考虑为分支创建规则的所有影响。例如，如果您选择限制只有具有项目管理员角色的用户才能向分支推送，那么具有贡献者角色的用户将无法在该分支中创建或编辑工作流，因为工作流 YAML 存储在该分支中，而这些用户无法提交和推送对 YAML 的更改。最佳做法是，在创建分支规则后对其进行测试，以确保它们不会产生任何预料之外的影响。您还可以将分支规则与拉取请求的审批规则结合使用。有关更多信息，请参阅 管理将拉取请求与审批规则合并的要求。

> **Note**
>
> 您必须具有项目管理员角色才能管理 CodeCatalyst 项目中源存储库的分支规则。您无法为链接的存储库创建分支规则。

您只能创建比角色默认权限更严格的分支规则。您创建的分支规则，不能比用户在项目中的角色所允许的权限更宽松。例如，您不能创建分支规则，来允许具有审阅者角色的用户推送到分支。

应用于源存储库默认分支的分支规则，与应用于其他分支的分支规则在行为上有些不同。应用于默认分支的任何规则，都将自动应用于您指定为默认分支的任何分支。以前被设置为默认分支的分支仍将保留应用于它的规则，只是不再具有防止删除的保护功能。这种保护只适用于当前的默认分支。

分支规则有两种状态：标准和自定义。标准表示允许在分支上执行的操作与用户在分支操作中的角色权限相匹配 CodeCatalyst 的操作。要进一步了解哪些角色拥有哪些权限，请参阅使用用户角色授予访问权限。自定义表示一个或多个分支中的操作具有特定的角色列表来授予执行该操作的权限，与项目中用户角色授予的默认权限不同。

Note

如果创建了分支规则来限制分支的一个或多个操作，则删除分支操作会自动设置为只允许具有项目管理员角色的用户删除该分支。

下表列出了操作以及允许在分支上执行这些操作的角色的默认设置。

分支操作和角色

分支操作	在未应用分支规则时，允许执行此操作的角色
合并到分支（这包括将拉取请求合并到分支）	项目管理员、贡献者
推送到分支	项目管理员、贡献者
删除分支	项目管理员、贡献者
删除分支（默认分支）	不允许

您不能删除分支规则，但可以更新这些规则来允许所有角色在分支上执行该操作，这样实际上就是移除了规则。

> **Note**
>
> 您必须具有项目管理员角色才能为 CodeCatalyst 项目中的源存储库配置分支规则。这不适用于链接的存储库。链接存储库不支持中的分支规则 CodeCatalyst。

查看和编辑存储库的分支规则

1. 导航到存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要在其中查看分支规则的存储库。
3. 在存储库的概述页面上，选择分支。
4. 在分支规则列中，查看存储库中每个分支的规则状态。标准表示分支操作规则是在源存储库中创建的任何分支的默认规则，并与项目中授予给这些角色的权限相匹配。自定义表示一个或多个分支操作所具备的规则，只允许一组不同的角色执行该分支中的一个或多个操作。

   要查看分支规则的具体内容，请选择要查看的分支旁边的标准或自定义字样。
5. 要创建或更改分支规则，请选择管理设置。在源存储库的设置页面上，在分支规则中，选择编辑。
6. 在分支中，从下拉列表中选择要为其配置规则的分支名称。对于每种允许的操作类型，从下拉列表中选择要允许执行该操作的角色，然后选择保存。

分支的 Git 命令

您可以使用 Git 创建、管理和删除计算机（本地存储库）或开发环境中的源代码库克隆中的分支，然后提交更改并将其推送到 CodeCatalyst 源存储库（远程存储库）。例如：

针对分支的常用 Git 命令

列出本地存储库中的所有分支，并在当前分支旁边显示一个星号 (*)。	`git branch`
将远程存储库中所有现有分支的信息提取到本地存储库。	`git fetch`
列出本地存储库中的所有分支和本地存储库中的远程跟踪分支。	`git branch -a`

只列出本地存储库中的远程跟踪分支。	`git branch -r`
使用指定的分支名称在本地存储库中创建一个分支。在您提交并推送更改之前，此分支不会出现在远程存储库中。	`git branch branch-name`
使用指定的分支名称在本地存储库中创建一个分支，然后切换到该分支。	`git checkout -b branch-name`
使用指定的分支名称切换到本地存储库中的另一个分支。	`git checkout other-branch-name`
使用本地存储库为远程存储库指定的昵称和指定的分支名称，将一个分支从本地存储库推送到远程存储库。同时为本地存储库中的分支设置上游跟踪信息。	`git push -u remote-name branch-name`
将本地存储库中另一个分支的更改合并到本地存储库中的当前分支。	`git merge from-other-branch-name`
删除本地存储库中的某个分支，除非其包含尚未合并的作业。	`git branch -d branch-name`

使用本地存储库为远程存储库指定的昵称和指定的分支名称，删除远程存储库中的一个分支。（注意冒号（:）的用法。）或者，在命令中指定 `--delete`。	`git push remote-name :branch-name`
	
	`git push remote-name --delete branch-name`

有关更多信息，请参阅您的 Git 文档。

查看分支和详细信息

您可以在 Amazon CodeCatalyst 控制台中查看有关亚马逊 CodeCatalyst远程分支的信息，包括文件、文件夹和特定分支的最新提交的详细信息。您也可以使用 Git 命令和本地操作系统来查看远程和本地分支的这些信息。

查看分支（控制台）

1. 在 CodeCatalyst 控制台中，导航到包含要在其中查看分支的源存储库的项目。选择代码，选择源存储库，然后选择源存储库。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要查看其中分支的存储库。
3. 此时将显示存储库的默认分支。您可以看到分支中的文件和文件夹列表、最新提交的信息，以及 README.md 文件的内容（如果分支中存在）。要查看不同分支的信息，请从存储库分支的下拉列表中选择该分支。
4. 要查看存储库的所有分支，请选择查看全部。分支页面显示每个分支的名称、最新提交和规则等信息。

有关如何使用 Git 和操作系统查看分支和详细信息的信息，请参阅针对分支的常用 Git 命令、Git 文档和操作系统文档。

删除分支

如果您不再需要某个分支，可以删除它。例如，如果您将一个有功能更改的分支合并到默认分支中，而该功能已经发布，那么您可能想删除原来的功能分支，因为这些更改已经是默认分支的一部分。保持较少的分支数量让用户可以更容易找到包含他们想要处理的更改的分支。删除分支后，该分支的副本会保留在本地计算机上的存储库克隆中，直到用户提取并同步这些更改。

删除分支（控制台）

1. 导航到存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要删除其中分支的存储库。
3. 在存储库的概述页面上，选择分支名称旁边的下拉选择器，然后选择查看全部。
4. 选择要删除的分支，然后选择删除分支。

> **Note**
>
> 您不能删除存储库的默认分支。

5. 您将看到确认对话框。其中显示与分支相关联的版本库、未处理的拉取请求数量和工作流数量。
6. 要确认删除该分支，请在文本框中键入 delete，然后选择删除。

您也可以使用 Git 删除分支。有关更多信息，请参阅 针对分支的常用 Git 命令。

在 Amazon 中管理源代码文件 CodeCatalyst

在 Amazon 中 CodeCatalyst，文件是一种受版本控制的独立信息，可供您和存储文件的源存储库和分支的其他用户使用。您可以使用目录结构来组织存储库文件。 CodeCatalyst自动跟踪对文件的每一次提交的更改。您可以在不同的存储库分支中存储文件的不同版本。

要在源存储库中添加或编辑多个文件，您可以使用 Git 客户端、开发环境或集成式开发环境（IDE）。要添加或编辑单个文件，您可以使用 CodeCatalyst 控制台。

主题

创建或添加文件

要创建文件并将其添加到源存储库，您可以使用 Amazon CodeCatalyst 控制台、开发环境、互联集成开发环境 (IDE) 或 Git 客户端。 CodeCatalyst 控制台包括用于创建文件的代码编辑器。使用该编辑器可方便地在存储库的某个分支中创建或编辑简单文件，如 README.md 文件。处理多个文件时，可以考虑创建开发环境。

从源存储库创建开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择代码，然后选择源存储库。
3. 选择要在其中处理代码的源存储库。
4. 选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。

6. 请执行以下操作之一：

   - 选择在现有分支中工作，然后从现有分支下拉菜单中选择一个分支。
   - 选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支的来源下拉菜单中选择要从中创建新分支的分支。

7. 还可选择为开发环境添加名称或编辑其配置。
8. 选择创建。

在 CodeCatalyst 控制台中创建文件

1. 导航到要在其中创建文件的项目。有关如何导航到存储库的更多信息，请参阅查看源存储库。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要在其中创建文件的存储库。
3. （可选）如果要在不同于默认分支的分支中创建文件，请选择要在其中创建文件的分支。
4. 选择创建文件。
5. 在文件名中输入文件的名称。在编辑器中添加文件内容。

   > **Tip**
   >
   > 如果要在分支根目录的子文件夹或子目录中创建文件，请将该结构作为文件名的一部分。

   对更改感到满意后，选择提交。
6. 在文件名中，查看文件名并根据需要进行更改。（可选）在分支中，从可用的分支列表中选择要在其中创建文件的分支。在提交消息中，可选择输入简短但翔实的描述，说明做出此更改的原因。这将显示为将文件添加到源存储库的提交的基本提交信息。
7. 选择提交，提交并将文件推送到源存储库。

您还可以通过将源存储库克隆到本地计算机，然后使用 Git 客户端或互联集成式开发环境（IDE）来推送文件和更改，从而将文件添加到源存储库中。

> **Note**
>
> 如果要添加 Git 子模块，您必须使用 Git 客户端或开发环境并运行 git submodule add 命令。您无法在 CodeCatalyst 控制台中添加或查看 Git 子模块，也无法在拉取请求中查看 Git 子模块的差异。有关 Git 子模块的更多信息，请参阅 Git 文档。

使用 Git 客户端或互联集成式开发环境（IDE）添加文件

1. 将源存储库克隆到本地计算机。有关更多信息，请参阅 克隆源存储库。
2. 在本地存储库中创建文件，或将文件复制到本地存储库中。
3. 执行以下操作之一，创建并推送提交：

   - 如果您使用的是 Git 客户端，请在终端或命令行下运行 git add 命令，指定要添加的文件的名称。或者，要添加所有已添加或更改的文件，请运行 git add 命令，然后加上单句点或双句点，以指示您是要包含当前目录级别的所有更改（单句点），还是要包含当前目录及所有子目录的所有更改（双句点）。要提交更改，请运行 git commit -m 命令并提供提交消息。要将您的更改推送到中的源存储库 CodeCatalyst，请运行git push。有关 Git 命令的更多信息，请参阅 Git 文档和分支的 Git 命令。
   - 如果您使用的是开发环境或 IDE，请在 IDE 中创建文件和添加文件，然后提交并推送您的更改。有关更多信息，请参阅使用开发环境编写和修改代码 CodeCatalyst或 IDE 文档。

查看文件

您可以在 Amazon CodeCatalyst 控制台中查看源存储库中的文件。您可以查看默认分支和任何其他分支中的文件。文件内容可能因您选择查看的分支而异。

在 CodeCatalyst 控制台中查看文件

1. 导航到要在其中查看文件的项目。有关更多信息，请参阅 查看源存储库。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要查看其中文件的存储库。
3. 此时将显示默认分支的文件和文件夹列表。文件用纸张图标表示，文件夹用文件夹图标表示。
4. 执行以下任一操作：

- 要查看不同分支中的文件和文件夹，请从分支列表中选择该分支。
- 要展开文件夹，请从列表中选择该文件夹。

5. 要查看特定文件的内容，请从列表中选择该文件。该文件的内容将显示在分支中。要查看不同分支中的文件内容，请从分支选择器中选择所需的分支。

> **Tip**
>
> 查看文件内容时，可以从查看文件中选择其他文件进行查看。要编辑文件，请选择编辑。

您可以在控制台中查看多个文件。您还可以使用 Git 客户端或集成式开发环境（IDE），查看已克隆到本地计算机的文件。有关更多信息，请参阅 Git 客户端或 IDE 的文档。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中查看 Git 子模块。有关 Git 子模块的更多信息，请参阅 Git 文档。

查看文件更改历史记录

您可以在 Amazon CodeCatalyst 控制台中查看源存储库中文件的更改历史记录。这有助于您了解在选择查看文件历史记录的分支上，不同提交对文件所做的更改。例如，如果查看源存储库的 **`main`** 分支中 **`readme.md`** 文件的更改历史记录，就会看到该分支中包含该文件更改的提交的列表。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中查看链接存储库中文件的历史记录。

在 CodeCatalyst 控制台中查看文件的历史记录

1. 导航到要查看其中文件历史记录的项目。有关更多信息，请参阅 查看源存储库。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。
3. 选择要在其中查看文件历史记录的存储库。选择要在其中查看文件历史记录的分支，然后从列表中选择文件。选择 View history (查看历史记录)。

4. 查看指定分支中包含对该文件更改的提交的列表。要查看特定提交中包含的更改详细信息，请在列表中选择该提交的提交消息。此时显示该提交与其父提交之间的差异。
5. 要查看另一个分支中文件的更改历史记录，请使用分支选择器将视图切换到该分支，从文件列表中选择文件，然后选择查看历史记录。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中查看 Git 子模块的更改历史记录。有关 Git 子模块的更多信息，请参阅 Git 文档。

编辑文件

您可以在 Amazon CodeCatalyst 控制台中编辑单个文件。要同时编辑多个文件，您可以创建一个开发环境或者克隆存储库，然后使用 Git 客户端或集成式开发环境（IDE）进行更改。有关更多信息，请参阅 使用开发环境编写和修改代码 CodeCatalyst或 克隆源存储库。

在 CodeCatalyst 控制台中编辑文件

1. 导航到要在其中编辑文件的项目。有关如何导航到存储库的更多信息，请参阅查看源存储库。
2. 选择要编辑其中文件的存储库。选择查看分支，然后选择要在其中工作的分支。从该分支的文件和文件夹列表中选择文件。

   文件内容将显示出来。
3. 选择编辑。
4. 在编辑器中，编辑文件内容，然后选择提交。（可选）在提交更改中，在提交消息中添加有关更改的更多信息。对更改感到满意后，选择提交。

重命名或删除文件

您可以在开发环境、本地计算机或集成式开发环境（IDE）中重命名或删除文件。重命名或删除文件后，提交并将这些更改推送到源存储库。您无法在 Amazon CodeCatalyst 控制台中重命名或删除文件。

在 Amazon 中使用拉取请求查看代码 CodeCatalyst

拉取请求是您和其他项目成员用于审查、注释和合并分支间的代码更改的主要方式。您可以使用拉取请求来协同审查代码更改，包括次要的更改或修复、主要功能添加或已发布软件的新版本。如果您使用事务来跟踪项目工作，您可以将特定事务与拉取请求相关联，以便于您跟踪拉取请求中的代码更改处理了哪些事务。当您创建、更新、评论、合并或关闭拉取请求时，系统会自动向拉取请求的作者以及该拉取请求的任何必需或可选的审阅者发送一封电子邮件。

> **Tip**
>
> 您可以在个人资料中配置您将收到哪些拉取请求事件的相关电子邮件。有关更多信息，请参阅从 CodeCatalyst 发送 Slack 和电子邮件通知。

拉取请求需要在源存储库中建立两个分支：一个是源分支，其中包含您希望审查的代码；另一个是目标分支，您希望在其中合并已审查的代码。源分支包含 AFTER 提交，该提交包含要合并到目标分支中的更改。目标分支包含 BEFORE 提交，表示代码在拉取请求分支合并到目标分支中之前的状态。

> **Note**
>
> 在创建拉取请求时，显示的差异是源分支最新块与目标分支最新块之间的差异。创建拉取请求后，显示的差异将是您选择的拉取请求修订版与创建拉取请求时目标分支最新块的提交之间的差异。有关 Git 中的差异和合并基础的更多信息，请参阅 Git 文档git-merge-base中的。

虽然拉取请求是为特定源存储库和分支创建的，但作为项目工作的一部分，您可以创建、查看、审查和关闭拉取请求。您不必查看源存储库就能查看和处理拉取请求。创建拉取请求时，拉取请求状态会设置为 Open。拉取请求将一直处于打开状态，直到您在 CodeCatalyst 控制台中将其合并（将状态更改为“已合并”）或将其关闭（将其状态更改为“已关闭”）。

代码通过审查后，您可以通过以下几种方式之一更改拉取请求状态：

- 在 CodeCatalyst 控制台中合并拉取请求。拉取请求源分支中的代码将被合并到目标分支中。拉取请求状态将变为已合并。无法改回 Open 状态。
- 在本地合并分支并推送您的更改，然后在 CodeCatalyst 控制台中关闭拉取请求。
- 使用 CodeCatalyst 控制台关闭拉取请求而不进行合并。这会将状态更改为已关闭，但不会将代码从源分支合并到目标分支。

在创建拉取请求之前，请：

- 将要审查的代码更改提交和推送至一个分支（源分支）。
- 为您的项目设置通知，这样其他用户就能收到您创建拉取请求时运行的任何工作流的相关通知。（此步骤为可选步骤，但建议使用。）

主题

创建拉取请求

创建拉取请求有助于在您将代码更改合并到另一分支之前，让其他用户查看和审核您所做的更改。首先，您需要为代码更改创建一个分支，这称作拉取请求的源分支。提交更改并将其推送到存储库后，您可以创建一个拉取请求来将源分支的内容与目标分支的内容进行比较。

您可以在 Amazon CodeCatalyst 控制台中从特定分支、拉取请求页面或项目概述创建拉取请求。从特定分支创建拉取请求会自动在拉取请求创建页面上提供存储库名称和源分支。创建一个拉取请求时，您将自动收到有关该拉取请求的任何更新以及拉取请求的合并或关闭时间的电子邮件。

> **Note**
>
> 在创建拉取请求时，显示的差异是源分支最新块与目标分支最新块之间的差异。创建拉取请求后，显示的差异将是您选择的拉取请求修订版与创建拉取请求时目标分支最新块的提交之间的差异。有关 Git 中的差异和合并基础的更多信息，请参阅 Git 文档git-merge-base中的。

在创建拉取请求时，您可以使用为我编写描述功能，让 Amazon Q 自动为拉取请求中包含的更改创建描述。当您选择此选项时，Amazon Q 会分析包含代码更改的源分支与要合并这些更改的目标分支之间的差异。然后，它会总结了这些更改的内容，以及对这些更改的意图和效果的优秀解释。此功能仅在美

国西部（俄勒冈）地区适用于 CodeCatalyst 拉取请求。为我编写描述功能不适用于已链接存储库中的拉取请求。

> **Note**
>
> > **Note**
> >
> > 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

创建拉取请求

1. 导航到您的项目。
2. 请执行以下操作之一：
   - 在导航窗格中，依次选择代码、拉取请求和创建拉取请求。
   - 在存储库主页上，选择更多，然后选择创建拉取请求。
   - 在项目页面上，选择创建拉取请求。
3. 在源代码库中，确保指定的源存储库是包含所提交代码的存储库。只有在您未从存储库的主页创建拉取请求时，才会显示此选项。
4. 在目标分支中，在审查代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含所提交代码的分支。
6. 在拉取请求标题中，输入一个标题，帮助其他用户了解需要审查的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如事务链接或所做更改的描述之类的信息。

> **Tip**
>
> 你可以选择 “为我写描述”， CodeCatalyst 自动生成拉取请求中包含的更改的描述。您可以在将自动生成的描述添加到拉取请求后，对描述进行更改。
> 此功能要求为空间启用生成式人工智能功能，并且不适用于链接的存储库中的拉取请求。
> 有关更多信息，请参阅 Managing generative AI features。

8. （可选）在事务中，选择关联事务，然后从列表中选择事务或输入事务 ID。要取消链接事务，请选择“取消链接”图标。
9. （可选）在必需的审阅者中，选择添加必需审阅者。从项目成员列表中进行选择，添加审阅者。必需的审阅者必须批准更改，才能将拉取请求合并到目标分支。

> **Note**
>
> 您不能将审阅者同时添加为必需的审阅者和可选的审阅者。您不能将自己添加为审阅者。

10. （可选）在可选的审阅者中，选择添加可选审阅者。从项目成员列表中进行选择，添加审阅者。可选的审阅者不必批准更改，这并不是将拉取请求合并到目标分支之前的必备要求。
11. 审查分支之间的差异。拉取请求中显示的差异是源分支中的修订与合并基准之间的更改，而合并基准是创建拉取请求时目标分支的 HEAD 提交。如果未显示任何更改，则分支可能相同，或者您可能为源和目标选择了相同的分支。
12. 如果您对拉取请求中包含的希望审查的代码和更改感到满意，请选择创建。

> **Note**
>
> 创建拉取请求后，可以添加备注。可以将备注添加到拉取请求中或文件内的单独行中，也可以为整个拉取请求添加备注。您可以使用 @ 符号后跟文件名的方式，添加指向文件等资源的链接。

从分支创建拉取请求

1. 导航到要在其中创建拉取请求的项目。
2. 在导航窗格中，选择源存储库，然后选择包含要审阅其代码更改的分支的存储库。
3. 选择默认分支名称旁边的下拉箭头，然后从列表中选择所需分支。要查看存储库的所有分支，请选择查看全部。
4. 选择更多，然后选择创建拉取请求。
5. 系统将为您预先选择存储库和源分支。在目标分支中，选择在审阅代码后要将代码合并到的分支。在拉取请求标题中，输入一个标题来帮助其他用户了解必须审阅的内容及其审阅原因。（可选）在拉取请求描述中提供更多信息，例如粘贴中相关问题的链接 CodeCatalyst，或者添加对您所做更改的描述。

> **Note**
>
> 如果拉取请求的目标分支与工作流中指定的某个分支匹配，则配置为针对拉取请求创建事件运行的工作流将在拉取请求创建后运行。

6. 审查分支之间的差异。如果未显示任何更改，则分支可能相同，或者您可能为源和目标选择了相同的分支。
7. （可选）在事务中，选择关联事务，然后从列表中选择事务或输入事务 ID。要取消链接事务，请选择“取消链接”图标。
8. （可选）在必需的审阅者中，选择添加必需审阅者。从项目成员列表中进行选择，添加审阅者。必需的审阅者必须批准更改，才能将拉取请求合并到目标分支。

> **Note**
>
> 您不能将审阅者同时添加为必需的审阅者和可选的审阅者。您不能将自己添加为审阅者。

9. （可选）在可选的审阅者中，选择添加可选审阅者。从项目成员列表中进行选择，添加审阅者。可选的审阅者不必批准更改，即可将拉取请求合并到目标分支中。
10. 如果您对拉取请求包含要审阅的更改以及所需的审阅者感到满意后，请选择创建。

如果您将任何工作流配置为在分支与拉取请求中的目标分支匹配时运行，则创建拉取请求后，您将在拉取请求详细信息区域的概述中看到有关这些工作流运行的信息。有关更多信息，请参阅添加触发器到工作流。

查看拉取请求

您可以在 Amazon CodeCatalyst 控制台中查看项目的拉取请求。项目摘要页面将显示项目的所有待处理的拉取请求。要查看所有拉取请求（无论状态如何），请导航到项目的拉取请求页面。在查看拉取请求时，您可以选择汇总针对为您创建的拉取请求的更改留下的所有评论。

> **Note**
>
> > **Note**
> >
> > 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配

给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

查看待处理的拉取请求

1. 导航到要在其中查看拉取请求的项目。
2. 在项目页面上，将显示待处理的拉取请求，包括有关拉取请求的创建者、包含拉取请求分支的存储库以及拉取请求的创建日期的信息。您可以按源存储库筛选待处理的拉取请求视图。
3. 要查看所有拉取请求，请选择查看全部。您可以使用选择器在各个选项之间进行选择。例如，要查看所有拉取请求，请选择任何状态和任何作者。

   或者，在导航窗格中，选择代码，再选择拉取请求，然后使用选择器来优化视图。
4. 在拉取请求页面上，您可以按 ID、标题、状态等对拉取请求进行排序。要自定义拉取请求页面上显示的信息的内容和数量，请选择齿轮图标。
5. 要查看特定的拉取请求，请从列表中选择该拉取请求。
6. 要查看与该拉取请求关联的工作流运行的状态（如果有），请选择概述，然后在工作流运行下查看拉取请求的拉取请求详细信息区域中的信息。

   如果为工作流配置了拉取请求创建或修订事件，并且工作流中的目标分支要求与拉取请求中指定的目标分支相匹配，则会运行工作流。有关更多信息，请参阅 添加触发器到工作流。
7. 要查看链接的事务（如果有），请选择概述，然后查看事务下的拉取请求详细信息中的信息。如果您想查看链接的事务，请从列表中选择其 ID。
8. （可选）要创建针对拉取请求修订中的代码更改留下的评论的摘要，请选择创建内容摘要。摘要将不包括针对整个拉取请求留下的任何评论。

   > **Note**
   >
   > 此功能要求为空间启用生成式人工智能功能，它不适用于链接的存储库并且仅在美国西部（俄勒冈州）区域提供。有关更多信息，请参阅 Managing generative AI features。
9. 要查看拉取请求中的代码更改，请选择更改。您可以在文件已更改中快速查看拉取请求中发生更改的文件数，以及拉取请求中具有评论的文件。文件夹旁边显示的评论数表示该文件夹中文件的评论数。展开文件夹可查看文件夹中每个文件的评论数。您还可以查看为特定代码行留下的任何评论。

> **Note**
>
> 并非拉取请求中的所有更改都可以在控制台中显示。例如，您无法在控制台中查看 Git 子模块，因此无法在拉取请求中查看子模块中的差异。有些差异可能太大而无法显示。有关更多信息，请参阅中的源存储库配额 CodeCatalyst和查看文件。

10. 要查看此拉取请求的质量报告，请选择报告。

> **Note**
>
> 必须配置工作流来生成报告，以使报告显示在拉取请求中。有关更多信息，请参阅 使用工作流进行测试。

管理将拉取请求与审批规则合并的要求

创建拉取请求时，您可以选择将必需或可选的审阅者添加到该单个拉取请求中。但是，您也可以创建所有拉取请求在合并到特定的目标分支时必须满足的要求。这些要求称为审批规则。审批规则是为存储库中的分支配置的。当您创建目标分支已配置审批规则的拉取请求时，除了获得任何必需的审阅者的批准外，还必须满足该规则的要求，然后才能将拉取请求合并到该分支。创建审批规则有助于您保持合并到分支（如默认分支）的质量标准。

应用于源存储库默认分支的审批规则与应用于其他分支的审批规则在行为上有些不同。应用于默认分支的任何规则，都将自动应用于您指定为默认分支的任何分支。以前设置为默认分支的分支仍将保留应用于它的规则。

创建审批规则时，您应该考虑项目用户在现在和将来如何满足该规则。例如，如果您的项目中有六个用户，而您创建的审批规则要求在合并到目标分支前需要五次审批，那么您实际上创建了一条规则，要求除创建拉取请求的人员以外的所有人在合并拉取请求前批准该请求。

> **Note**
>
> 您必须具有项目管理员角色才能在 CodeCatalyst 项目中创建和管理批准规则。无法为链接的存储库创建审批规则。

您不能删除审批规则，但可以将其更新为要求零审批，这样就有效地删除了规则。

查看和编辑拉取请求目标分支的审批规则

1. 导航到存储库所在的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要在其中查看审批规则的存储库。
3. 在存储库的概述页面上，选择分支。
4. 在审批规则栏中，选择查看可查看存储库每个分支的任何规则的状态。

   在最小批准数中，该数字与拉取请求合并到该分支前所需的批准数相对应。
5. 要创建或更改审批规则，请选择管理设置。在源存储库的设置页面上，在审批规则中，选择编辑。

   > **Note**
   >
   > 您必须具有项目管理员角色才能编辑审批规则。
6. 在分支中，从下拉列表中选择要为其配置审批规则的分支名称。在最小批准数中，输入一个数字，然后选择保存。

审核拉取请求

您可以使用 Amazon CodeCatalyst 控制台协作查看并评论拉取请求中包含的更改。您可以向单个代码行中添加评论，说明源分支和目标分支之间的差异以及各个拉取请求修订之间的差异。您可以选择创建在拉取请求中更改代码时留下的评论的摘要，以帮助您快速了解其他用户留下的反馈。您还可以选择创建一个开发环境来处理代码。

> **Note**
>
> > **Note**
> >
> > 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

> **Tip**
>
> 您可以在个人资料中配置您将收到哪些拉取请求事件的相关电子邮件。有关更多信息，请参阅从 CodeCatalyst 发送 Slack 和电子邮件通知。

拉取请求显示了拉取请求修订与创建拉取请求时作为目标分支最新块的提交之间的差异。这称作合并基础。有关 Git 中的差异和合并基础的更多信息，请参阅 Git 文档git-merge-base中的。

> **Tip**
>
> 在控制台中工作时，特别是在您已打开拉取请求一段时间的情况下，请考虑刷新浏览器以确保在开始审核拉取请求之前，有最新的修订可供拉取请求使用。

在 CodeCatalyst 控制台中查看拉取请求

1. 导航到您的项目。
2. 通过执行下列操作之一来导航到拉取请求：
   - 如果项目页面上列出了拉取请求，请从列表中选择它。
   - 如果项目页面上未列出拉取请求，请选择查看全部。使用筛选条件和排序来查找拉取请求，然后从列表中选择它。
   - 在导航窗格中，选择代码，然后选择拉取请求。
3. 从列表中选择要审核的拉取请求。您可以通过在筛选条件栏中键入拉取请求的部分名称来筛选拉取请求列表。
4. 在概述中，您可以审核拉取请求的名称和标题。您可以创建和查看拉取请求本身留下的评论。您还可以查看拉取请求的详细信息，包括有关工作流运行、链接的事务、审阅者、拉取请求作者和可行的合并策略的信息。

   > **Note**
   >
   > 特定代码行上留下的评论会显示在更改中。

5. （可选）要添加适用于整个拉取请求的评论，请展开针对拉取请求的评论，然后选择创建评论。
6. （可选）要查看对此拉取请求修订中的更改留下的所有评论的摘要，请选择创建评论摘要。

> **Note**
>
> 此功能要求为空间启用生成式人工智能功能，并且仅在美国西部（俄勒冈州）区域提供。有关更多信息，请参阅 Managing generative AI features。

7. 在更改中，您可以查看目标分支与拉取请求的最新修订之间的差异。如果有多个修订，您可以在它们之间的差异中更改要比较的修订。有关修订的更多信息，请参阅修订。

> **Tip**
>
> 您可以在文件已更改中快速查看拉取请求中发生更改的文件数，以及拉取请求中具有评论的文件。文件夹旁边显示的评论数表示该文件夹中文件的评论数。展开文件夹可查看文件夹中每个文件的评论数。

8. 要更改差异的显示方式，请在统一和拆分之间进行选择。
9. 要向拉取请求中的行添加评论，请转到要评论的行。选择为该行显示的评论图标，输入评论，然后选择保存。
10. 要查看拉取请求中的各个修订之间的更改或其源分支和目标分支之间的更改，请从比较中的可用选项中进行选择。修订中各行的评论将保留在这些修订中。
11. 如果您已将工作流配置为生成有关拉取请求触发器的代码覆盖率报告，则可以在相关的拉取请求中查看行和分支覆盖率结果。要隐藏代码覆盖率结果，请选择隐藏代码覆盖率。有关更多信息，请参阅 代码覆盖率报告。
12. 如果要对拉取请求进行代码更改，则可以从拉取请求创建开发环境。选择创建开发环境。可选择为开发环境添加名称或编辑其配置，然后选择创建。
13. 在报告中，您可以查看此拉取请求中的质量报告。如果有多个修订，您可以在它们之间的差异中更改要比较的修订。您可以按名称、状态、工作流、操作和类型筛选报告。

> **Note**
>
> 必须配置工作流来生成报告，以使报告显示在拉取请求中。有关更多信息，请参阅 在操作中配置质量报告。

14. 要查看特定报告，请从列表中选择该报告。有关更多信息，请参阅 使用工作流进行测试。
15. 如果您被列为拉取请求的审阅者并想批准更改，请确保您正在查看最新修订，然后选择批准。

> **Note**
>
> 所有必需的审阅者都必须批准拉取请求，之后才能合并拉取请求。

更新拉取请求

通过更新拉取请求，可让其他项目成员更轻松地审核代码。您可以更新拉取请求以更改其审阅者、事务链接、拉取请求的标题或描述。例如，您可能需要更改拉取请求的必需的审阅者，以移除正在度假的人员并添加其他人员。您还可以将提交推送到处于打开状态的拉取请求的源分支，从而通过进一步的代码更改来更新拉取请求。每次推送到源存储库中拉取请求的 CodeCatalyst 源分支都会创建一个修订版。项目成员可以查看拉取请求中的各个修订之间的差异。

更新拉取请求的审阅者

1. 导航到要在其中更新拉取请求审阅者的项目。
2. 在项目页面上的打开拉取请求下，选择要更新其审阅者的拉取请求。或者，在导航窗格中，选择代码，再选择拉取请求，然后选择要更新的拉取请求。
3. （可选）在概述中的拉取请求详细信息区域中，选择加号以添加必需或可选的审阅者。选择审阅者旁边的 X 可将其作为可选或必需的审阅者移除。
4. （可选）在概述中的拉取请求详细信息区域中，选择链接的事务以将事务链接到拉取请求，然后从列表中选择事务或输入事务 ID。要取消链接某个事务，请选择要取消链接的事务旁边的取消链接图标。

更新拉取请求的源分支中的文件和代码

1. 要更新多个文件，您可以创建开发环境，也可以克隆存储库及其源分支，然后使用 Git 客户端或集成式开发环境（IDE）对源分支中的文件进行更改。提交更改并将其推送到源存储库中的 CodeCatalyst 源分支，以使用更改自动更新拉取请求。有关更多信息，请参阅克隆源存储库和通过在 Amazon 中提交来了解源代码的变化 CodeCatalyst。
2. 可以使用 Git 客户端或 IDE 更新源分支中的单个文件，就像更新多个文件一样。您也可以直接在 CodeCatalyst控制台中对其进行编辑。有关更多信息，请参阅 编辑文件。

更新拉取请求的标题和描述

1. 导航到要在其中更新拉取请求的标题或描述的项目。
2. 项目页面将显示待处理的拉取请求，包括有关拉取请求的创建者、包含拉取请求分支的存储库以及拉取请求的创建时间的信息。您可以按源存储库筛选待处理的拉取请求视图。从列表中选择要更改的拉取请求。
3. 要查看所有拉取请求，请选择查看全部。或者，在导航窗格中，选择代码，然后选择拉取请求。使用筛选框或排序功能查找要更改的拉取请求，然后将其选定。
4. 在概述部分中，选择编辑。
5. 更改标题或描述，然后选择保存。

合并拉取请求

在您的代码经过审核并且所有必需的审阅者都已批准之后，您可以使用支持的合并策略（例如快进）在 CodeCatalyst 控制台中合并拉取请求。并非 CodeCatalyst 控制台支持的所有合并策略都可用作所有拉取请求的选项。 CodeCatalyst 评估合并，并且仅允许您在控制台中可用且能够将源分支合并到目标分支的合并策略之间进行选择。您还可以使用所选的 Git 合并策略来合并拉取请求，方法是在本地计算机或开发环境中运行 git merge 命令来将源分支合并到目标分支中。然后，您可以将目标分支中的这些更改推送到中的源存储库 CodeCatalyst。

> **Note**
>
> 合并分支并在 Git 中推送更改不会自动关闭拉取请求。

如果您具有项目管理员角色，则还可以选择合并尚未满足所有审批要求和审批规则的拉取请求。

合并拉取请求（控制台）

如果源分支和目标分支之间没有合并冲突，并且所有必需的审阅者都批准了拉取请求，则可以在 CodeCatalyst 控制台中合并拉取请求。如果发送冲突或者无法完成合并，则合并按钮将处于非活跃状态，并显示不可合并标签。在这种情况下，您必须获得所有必需的批准者的审批，本地解决冲突（如有必要）并推送这些更改，之后才能进行合并。合并拉取请求会自动向拉取请求的创建者以及任何所需或可选的审阅者发送一封电子邮件。这不会自动关闭或更改与拉取请求链接的任何事务的状态。

> **Tip**
>
> 您可以在个人资料中配置您将收到哪些拉取请求事件的相关电子邮件。有关更多信息，请参阅从 CodeCatalyst 发送 Slack 和电子邮件通知。

合并拉取请求

1. 导航到要在其中合并拉取请求的项目。
2. 在项目页面上的打开拉取请求下，选择要合并的拉取请求。如果您未看到拉取请求，请选择查看所有拉取请求，然后从列表中选择该拉取请求。或者，在导航窗格中，选择代码，再选择拉取请求，然后选择要合并的拉取请求。选择合并。
3. 从拉取请求的可用合并策略中选择。（可选）选择或取消选择在合并拉取请求后删除源分支的选项，然后选择合并。

> **Note**
>
> 如果“合并”按钮处于非活动状态，或者您看到“不可合并”标签，则表示必需的审阅者尚未批准拉取请求，或者无法在控制台中合并拉取请求。 CodeCatalyst 概述中的拉取请求详细信息区域中的时钟图标会指示尚未批准拉取请求的审阅者。如果所有必需的审阅者都已批准拉取请求，但合并按钮仍处于非活跃状态，则可能存在合并冲突。选择带下划线的不可合并标签，以查看有关拉取请求无法合并的原因的更多详细信息。您可以在开发环境或 CodeCatalyst 控制台中解决目标分支的合并冲突，然后合并拉取请求，也可以解决冲突并在本地合并，然后将包含合并的提交推送到中的源分支 CodeCatalyst。有关更多信息，请参阅合并拉取请求（Git）和您的 Git 文档。

覆盖合并要求

如果您具有项目管理员角色，则可以选择合并尚未满足所有必需的审批要求和审批规则的拉取请求。这称为覆盖拉取请求的要求。如果必需的审阅者不可用，或者迫切需要将特定的拉取请求合并到具有无法快速满足的审批规则的分支中，则可以选择执行此操作。

合并拉取请求

1. 在要覆盖要求并合并的拉取请求中，选择合并按钮旁边的下拉箭头。选择覆盖审批要求。
2. 在覆盖原因中，详细说明为什么要在拉取请求未达到审批规则和必需的审阅者要求的情况下覆盖此拉取请求。虽然这是一项可选操作，但强烈建议您这样做。

3. （可选）选择合并策略或接受默认值。您也可以选择使用更多详细信息更新自动生成的提交消息。
4. 选择或取消选择该选项可在合并时删除源分支。建议您在覆盖合并拉取请求的要求时保留源分支，直到您有机会与其他团队成员一起审阅该决定。
5. 选择合并。

合并拉取请求（Git）

Git 支持许多用于合并和管理分支的选项。以下命令是您可使用的一些选项。有关更多信息，请参阅 Git 网站上的可用文档。在合并并推送更改后，请手动关闭拉取请求。有关更多信息，请参阅 关闭拉取请求。

用于合并分支的常用 Git 命令

将本地存储库中的源分支的更改合并到本地存储库中的目标分支。	`git checkout destination-branch-name`
	
	`git merge source-branch-name`

将源分支合并到目标分支，并指定快进式合并。这会合并分支并将目标分支指针移动到源分支的最新块。	`git checkout destination-branch-name`
	
	`git merge --ff-only  source-branch-name`

将源分支合并到目标分支，并指定压缩合并。这会将源分支中的所有提交合并到目标分支中的单个合并提交。	`git checkout destination-branch-name`
	
	`git merge --squash source-branch-name`

将源分支合并到目标分支，并指定三向合并。这会创建合并提交，并将源分支中的各个提交添加到目标分支。	`git checkout destination-branch-name`
	
	`git merge --no-ff source-branch-name`

删除本地存储库中的源分支。在合并到目标分支并将更改推送到源存储库之后，执行此操作对于清理本地存储库非常有用。	`git branch -d source-branch-name`

使用本地存储库为远程存储库指定的昵称删除远程存储库（中的源存储库 CodeCatalyst）中的源分支。（注意冒号（：）的用法。）或者，在命令中指定 `--delete`。	`git push remote-name :source-branch-name`
	
	`git push remote-name --delete source-branch-name`

关闭拉取请求

您可以将拉取请求标记为已关闭。这不会合并拉取请求，但有助于您确定哪些拉取请求需要操作，以及哪些拉取请求不再相关。如果您不再计划合并这些更改，或者更改已由另一个拉取请求合并，建议您关闭拉取请求。

关闭拉取请求会自动向拉取请求的创建者以及任何所需或可选的审阅者发送一封电子邮件。这不会自动更改与拉取请求链接的任何事务的状态。

> **Note**
>
> 拉取请求一经关闭便无法重新打开。

关闭拉取请求

1. 导航到要在其中关闭拉取请求的项目。
2. 在项目页面上，将显示打开拉取请求。选择要关闭的拉取请求。
3. 选择关闭。
4. 检查信息，然后选择关闭拉取请求。

通过在 Amazon 中提交来了解源代码的变化 CodeCatalyst

提交是存储库内容以及对该内容进行的更改的快照。每次用户提交并向分支推送更改时，都会保存该信息。Git 提交信息包括提交作者、提交更改的人、日期和时间，以及所做的更改。当您在亚马逊 CodeCatalyst 控制台中创建或编辑文件时，系统会自动包含类似信息，但作者姓名是您的 CodeCatalyst 用户名。您还可以为提交添加 Git 标签，以便于识别特定提交。

在 Amazon 中 CodeCatalyst，您可以：

- 查看分支的提交列表。

- 查看单个提交，包括与一个或多个父提交相比，某个提交中所做的更改。

您还可以查看文件和文件夹。有关更多信息，请参阅 在 Amazon 中管理源代码文件 CodeCatalyst。

主题

查看分支的提交情况

您可以通过在 CodeCatalyst 控制台中查看分支的提交来查看对分支所做更改的历史记录。这有助于您了解谁在何时对分支进行了更改。您还可以查看特定提交中所做的更改。

> **Tip**
>
> 以及查看对特定文件进行更改的提交的历史记录。有关更多信息，请参阅 查看文件。

您还可以使用 Git 客户端查看提交。有关更多信息，请参阅您的 Git 文档。

查看提交（控制台）

1. 导航到包含要查看其中提交的源存储库的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要在其中查看分支提交的存储库。
3. 此时将显示存储库的默认分支，包括该分支的最新提交信息。选择提交。或者，选择更多，然后选择查看提交。
4. 要查看不同分支的提交，请选择分支选择器，然后选择分支名称。
5. 要查看特定提交的详细信息，请从提交标题中选择其标题。此时将显示提交的详细信息，包括其父提交的信息，以及通过对父提交与指定提交进行的对比，确认的对代码所做的更改。

> **Tip**
>
> 如果一个提交有多个父提交，您可以通过父提交 ID 旁边的下拉图标，选择要查看其信息并显示更改的父提交。

更改提交的显示方式（CodeCatalyst控制台）

您可以更改提交视图中显示的信息。您可以选择隐藏或显示作者和提交 ID 等列。

更改提交的显示方式（控制台）

1. 导航到包含要查看其中提交的源存储库的项目。
2. 从项目的源存储库列表中选择存储库的名称。或者，在导航窗格中，选择代码，然后选择源存储库。

   选择要在其中更改提交查看方式的存储库。
3. 此时将显示存储库的默认分支，包括该分支的最新提交信息。选择提交。
4. 选择齿轮图标。
5. 在首选项中，选择要显示的提交次数，并选择是否显示提交作者、提交日期和提交 ID 的信息。

   > **Note**
   >
   > 在信息显示中无法隐藏提交标题。
6. 完成更改后，选择保存以保存更改，或选择取消放弃更改。

中的源存储库配额 CodeCatalyst

下表描述了 Amazon 中源存储库的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

资源	信息
分支名称	任意允许使用的字符组合，长度在 1 到 256 个字符之间，且必须在存储库中唯一。分支名称不能： • 以斜杠 (/) 或点号 (.) 开头或结尾 • 只包含单个字符 @ • 包含两个或多个连续的点号 (..)、正斜杠 (//) 或以下字符的组合：@{ • 包含空格或以下任意字符：？ ^ * [\ ~ : 分支名称是引用。分支名称的很多限制基于 Git 引用标准。有关更多信息，请参阅 Git 内部结构和git-check-ref-format。
关于拉取请求的注释	拉取请求最多 1000 个。
提交消息	最多 1024 个字符。
文件路径	允许的字符的任意组合，长度在 1 到 4,096 个字符之间。文件路径必须是一个明确的名称，用于指定文件和确切的文件位置。文件路径深度不能超过 20 个目录。此外，文件路径不能： • 包含空字符串 • 是相对文件路径 • 包含以下任意字符组合： /./ /../ // • 以尾随斜杠或反斜杠结尾

资源	信息
	文件名和路径必须是完全限定的。本地计算机上文件的名称和路径必须遵循该操作系统的标准。在指定存储库中文件的路径时，请使用 Amazon Linux 的标准。
文件大小	使用 CodeCatalyst控制台时，任何单个文件的最大容量为 6 MB。
可在控制台中查看文件大小 CodeCatalyst	使用 CodeCatalyst控制台时，任何单个文件的最大容量为 6 MB。
Git blob 大小	最大 2 GB。 **Note** 单个提交中的所有文件的数量和总大小没有限制，只要元数据不超过 6 MB 并且单个 blob 不超过 2 GB 即可。不过，作为最佳实践，可以考虑进行多次较小的提交，而不是一次大的提交。
提交的元数据	对于组合的提交的元数据（例如，作者信息、日期、父提交列表和提交消息的组合），最多 6 MB。 **Note** 单个提交中的所有文件的数量和总大小没有限制，只要数据不超过 20 MB，单个文件不超过 6 MB，并且单个 blob 不超过 2 GB。
可以关联到拉取请求 CodeCatalyst 的问题数量	50
可关联到拉取请求的 Jira 事务数	50

资源	信息
空间中未处理的拉取请求数	亚马逊 CodeCatalyst 空间的最大值为 1,000。
空间中的总拉取请求数	亚马逊 CodeCatalyst 空间的最大值为 10,000。
单个推送中的引用数	最多 4000 个，包括创建、删除和更新。存储库中的引用总数没有限制。
空间中的存储库数量	亚马逊 CodeCatalyst 空间上限为 5,000。
存储库描述	任意字符组合，长度在 0 到 1000 个字符之间。存储库描述是可选的。
存储库名称	一个项目中的存储库名称必须唯一。它们可以包含字母、数字、句点、下划线和短划线的任意组合，长度在 1 到 100 个字符之间。名称不区分大小写。存储库名称不能以 .git 结尾，不能包含空格，也不能包含以下任何字符：! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
存储库大小	存储库大小受空间总体存储限制的影响。有关更多信息，请参阅 Pricing 和排查源存储库的问题。
拉取请求的审阅者	拉取请求总共最多有 100 位审阅者（可选或必需）。
拉取请求的书面摘要	拉取请求的最大书面摘要数量取决于您空间的计费套餐。有关更多信息，请参阅 定价。

使用开发环境编写和修改代码 CodeCatalyst

开发环境是基于云的开发环境。在 Amazon 中 CodeCatalyst，您可以使用开发环境来处理存储在项目源存储库中的代码。创建开发环境时，您有以下几种选择：

- 在中创建特定于项目的开发环境 CodeCatalyst ，以便使用支持的集成开发环境 (IDE) 来处理代码。
- 创建一个空的开发环境，从源存储库中将代码克隆到此开发环境中，并使用受支持的 IDE 处理这些代码。
- 在 IDE 中创建一个开发环境，并将源存储库克隆到此开发环境中。

devfile 是一个开放的标准 YAML 文件，可用于标准化您的开发环境。换句话说，此文件编纂了开发环境所需的开发工具。因此，您可以快速设置开发环境，在各个项目之间切换，并在各个团队成员之间复制开发环境配置。开发环境可以最大限度地减少您创建和维护本地开发环境所花费的时间，因为它们使用 devfile 来配置对给定项目编码、测试和调试所需的所有工具。

> **Important**
>
> 开发环境可以通过有权访问您的 CodeCatalyst 凭据的开发文件运行脚本。在打开不受信任的资源之前，请检查存储库。

开发环境中包含的项目工具和应用程序库由项目的源存储库中的 devfile 定义。如果您的源存储库中没有开发文件，则 CodeCatalyst 会自动应用默认的开发文件。此默认 devfile 包括适用于最常用的编程语言和框架的工具。如果您的项目是使用蓝图创建的，则开发文件将由 CodeCatalyst自动创建。有关 devfile 的更多信息，请参阅 https://devfile.io。

创建一个开发环境后，只有您能够访问该开发环境。在开发环境中，您可以在受支持的 IDE 中查看和处理源存储库的代码。

默认情况下，开发环境配置为具有双核处理器、4 GB RAM 和 16 GB 持久性存储。如果您具有空间管理员权限，则可以更改空间的计费等级，以使用其他开发环境配置选项并管理计算和存储限制。

主题

- 编辑开发环境
- 删除开发环境
- 使用 SSH 连接到开发环境
- 配置开发环境的 devfile
- 将 VPC 连接关联到开发环境
- 中开发环境的配额 CodeCatalyst

创建开发环境

您可以通过多种方式创建开发环境：

- CodeCatalyst 使用 “概述”、“开发环境” 或 “源代码库” 页面中的源存储库或链接源存储库创建开发环境 CodeCatalyst
- 在 “开发环境” 页面 CodeCatalyst 中创建一个未连接到源存储库的空开发环境
- 在您选择的 IDE 中创建一个开发环境，然后将任何源存储库克隆到该开发环境中

> **Important**
>
> 开发环境不适用于将 Active Directory 用作身份提供商的空间中的用户。有关更多信息，请参阅 当我使用单点登录账户登录 CodeCatalyst 时，无法创建开发环境。

您可以为存储库的每个分支创建一个开发环境。一个项目可以有多个存储库。您创建的开发环境只能使用您的 CodeCatalyst 账户进行管理，但您可以打开开发环境并使用任何支持的环境在其中工作 IDEs。必须安装 AWS Toolkit 才能在 IDE 中使用开发环境。有关更多信息，请参阅 开发环境支持的集成式开发环境。默认情况下，系统使用双核处理器、4 GB RAM 和 16 GB 持久性存储创建开发环境。

> **Note**
>
> 如果您创建了与源存储库关联的开发环境，则资源列将始终显示在创建此开发环境时指定的分支。即使您创建另一个分支、切换到开发环境中的另一个分支或克隆其他存储库，这也适用。如果您创建了一个空开发环境，则资源列将为空。

开发环境支持的集成式开发环境

您可以将开发环境与以下支持的集成开发环境 (IDEs) 配合使用：

- AWS Cloud9
- JetBrains IDEs
  - IntelliJ IDEA Ultimate
  - GoLand
  - PyCharm专业人士
- Visual Studio Code

在中创建开发环境 CodeCatalyst

要开始在中使用开发环境 CodeCatalyst，请使用您的AWS 生成器 ID 或 SSO 进行身份验证并登录。

从分支创建开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建开发环境的项目。
3. 在导航窗格中，执行下列操作之一：
   - 选择概述，然后导航到我的开发环境部分。
   - 选择代码，然后选择开发环境。
   - 选择代码，再选择源存储库，然后选择要为其创建开发环境的存储库。
4. 选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 选择克隆存储库。
7. 请执行以下操作之一：
   a. 选择要克隆的存储库，选择在现有分支中工作，然后从现有分支下拉菜单中选择一个分支。

> **Note**
>
> 如果您选择第三方存储库，则必须在现有分支中工作。

b. 选择要克隆的存储库，选择在新分支中工作，在分支名称字段中输入分支名称，然后从创建分支自下拉菜单中选择要从中创建新分支的分支。

> **Note**
>
> 如果您从源存储库页面或从特定的源存储库创建开发环境，则无需选择存储库。这将从您在源存储库页面中选择的源存储库创建开发环境。

8. （可选）在别名 – 可选中，输入开发环境的别名。
9. （可选）选择开发环境配置编辑按钮，编辑开发环境的计算、存储或超时配置。
10. （可选）在 Amazon Virtual Private Cloud (Amazon VPC) - 可选中，从下拉菜单中选择要与开发环境关联的 VPC 连接。

    如果为您的空间设置了默认 VPC，则开发环境将连接到该 VPC。您可以通过关联其他 VPC 连接来覆盖此设置。另请注意，与 VPC 连接的开发环境不支持 AWS Toolkit。

    如果您要使用的 VPC 连接未列出，则可能是因为该 AWS 账户 连接包含您的项目中不允许的连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

    > **Note**
    >
    > 当您创建具有 VPC 连接的开发环境时，会在 VPC 内创建一个新的网络接口。CodeCatalyst 使用关联的 VPC 角色与该接口交互。此外，请确保您的 IPv4 CIDR 块未配置为 `172.16.0.0/12` IP 地址范围。

11. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动，开发环境创建完成后，状态列将显示正在运行。

创建空的开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建开发环境的项目。
3. 在导航窗格中，执行下列操作之一：

   - 选择概述，然后导航到我的开发环境部分。
   - 选择代码，然后选择开发环境。

4. 选择创建开发环境。
5. 从下拉菜单中选择受支持的 IDE。请参阅开发环境支持的集成式开发环境了解更多信息。
6. 选择创建空的开发环境。
7. （可选）在别名 – 可选中，输入开发环境的别名。
8. （可选）选择开发环境配置编辑按钮，编辑开发环境的计算、存储或超时配置。
9. （可选）在 Amazon Virtual Private Cloud (Amazon VPC) - 可选中，从下拉菜单中选择要与开发环境关联的 VPC 连接。

   如果为您的空间设置了默认 VPC，则开发环境将连接到该 VPC。您可以通过关联其他 VPC 连接来覆盖此设置。另请注意，与 VPC 连接的开发环境不支持 AWS Toolkit。

   如果您要使用的 VPC 连接未列出，则可能是因为该 AWS 账户 连接包含您的项目中不允许的连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

   > **Note**
   >
   > 当您创建具有 VPC 连接的开发环境时，会在 VPC 内创建一个新的网络接口。CodeCatalyst 使用关联的 VPC 角色与该接口交互。此外，请确保您的 IPv4 CIDR 块未配置为 `172.16.0.0/12` IP 地址范围。

10. 选择创建。在创建开发环境时，开发环境状态列将显示正在启动，开发环境创建完成后，状态列将显示正在运行。

> **Note**
>
> 首次创建和打开开发环境可能需要花费一到两分钟的时间。

> **Note**
>
> 在 IDE 中打开开发环境后，您可能需要先将目录更改为源存储库，然后再提交和推送对代码的更改。

在 IDE 中创建开发环境

您可以使用开发环境来快速处理存储在项目源存储库中的代码。开发环境可加快开发速度，因为您可以使用受支持的集成式开发环境（IDE）立即在项目特定的、功能齐全的云开发环境中开始编码。

有关在 IDE CodeCatalyst 中使用的信息，请参阅以下文档。

- Amazon f CodeCatalyst or JetBrains IDEs
- 亚马逊 V CodeCatalyst S Code 版
- Amazon f CodeCatalyst or AWS Cloud9

停止开发环境

开发环境的 `/projects` 目录存储从源存储库中提取的文件和用于配置开发环境的 devfile。`/home` 目录（创建开发环境时，该目录为空）存储您在使用开发环境时创建的文件。开发环境的 `/projects` 和 `/home` 目录中的所有文件都是永久性存储的，因此，您可以在需要切换到其他开发环境、存储库或项目时停止开发环境中的工作。

> **Warning**
>
> 如果任何实例（包括 Web 浏览器、远程 shell 和）保持连接状态 IDEs，则开发环境不会超时。因此，请务必关闭所有连接的实例，以免产生额外费用。

如果开发环境处于空闲状态的时间达到了在创建开发环境期间为超时字段选择的时间长度，它将自动停止。您可以在开发环境处于空闲状态之前将其停止。如果您在创建开发环境时选择了无超时，则开发环境将不会自动停止。相反，它将持续运行。

> **Warning**
>
> 如果您停止与已删除的 VPC 连接关联的开发环境，则该开发环境将无法恢复。

从开发环境页面停止开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中停止开发环境的项目。

3. 在导航窗格中，选择代码。
4. 选择开发环境。
5. 选择要停止的开发环境的单选按钮。
6. 从操作菜单中，选择停止。

> **Note**
>
> 计算使用仅在开发环境运行时计费，而存储使用按开发环境存在的整个时长计费。在未使用开发环境时将其停用以停止计算计费。

恢复开发环境

开发环境的 `/projects` 目录存储从源存储库中提取的文件和用于配置开发环境的 devfile。`/home` 目录（创建开发环境时，该目录为空）存储您在使用开发环境时创建的文件。开发环境的 `/projects` 和 `/home` 目录中的所有文件都是永久性存储的，因此，您可以在需要切换到其他开发环境、存储库或项目时停止开发环境中的工作，并稍后恢复开发环境中的工作。

如果开发环境处于空闲状态的时间达到了在创建开发环境期间为超时字段选择的时间长度，它将自动停止。必须关闭 AWS Cloud9 浏览器选项卡，开发环境才会处于空闲状态。

> **Note**
>
> 即使您删除了用于创建开发环境的分支，开发环境仍可用并正常运行。如果要在已删除分支的开发环境中恢复工作，请创建一个新分支并将更改推送到该分支。

从概述页面恢复开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中恢复开发环境的项目，然后导航到我的开发环境部分。
3. 选择在 (IDE) 中恢复。

   - 对于 JetBrains IDEs，当提示选择要打开 JetBrains -gateway链接的应用程序时，请选择 Gateway-e ap。 JetBrains当系统提示时，请选择打开链接以进行确认。
   - 对于 VS Code IDE，当系统提示您选择应用程序以打开 VS Code 链接时，请选择 VS Code。选择打开链接以进行确认。

从源存储库恢复开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要从中恢复开发环境的项目。
3. 在导航窗格中，选择代码。
4. 选择源存储库。
5. 选择包含要恢复的开发环境的源存储库。
6. 选择分支名称以查看分支的下拉菜单，然后选择您的分支。
7. 选择恢复开发环境。
   - 对于 JetBrains IDEs，当系统提示允许此站点通过 Gateway 打开 JetBrains-gateway 链接时，选择 Open Link 进行确认 JetBrains？ 。
   - 对于 VS Code IDE，当系统提示您允许此站点通过 Visual Studio Code 打开 VS Code 链接? 时，选择打开链接以进行确认。

从开发环境页面恢复开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要从中恢复开发环境的项目。
3. 在导航窗格中，选择代码。
4. 选择开发环境。
5. 从 IDE 列中，为开发环境选择在 (IDE) 中恢复。
   - 对于 JetBrains IDEs，当系统提示允许此站点通过 Gateway 打开 JetBrains-gateway 链接时，选择 Open Link 进行确认 JetBrains？ 。
   - 对于 VS Code IDE，当系统提示您允许此站点通过 Visual Studio Code 打开 VS Code 链接? 时，选择打开链接以进行确认。

> **Note**
>
> 恢复开发环境需要几分钟的时间。

编辑开发环境

当 IDE 正在运行时，您可以编辑开发环境。如果您编辑计算或非活动超时，则您的开发环境将在您保存更改后重新启动。

编辑开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中编辑开发环境的项目。
3. 在导航窗格中，选择代码。
4. 选择开发环境。
5. 选择要编辑的开发环境。
6. 选择编辑。
7. 根据需要对计算或非活动超时进行更改。
8. 选择保存。

删除开发环境

处理完存储在开发环境中的内容后，您可以删除开发环境。创建一个新的开发环境来处理新内容。如果您删除开发环境，则将永久删除保留的内容。在删除开发环境之前，请确保代码更改提交并推送到开发环境的原始源存储库。删除开发环境后，将停止对开发环境的计算和存储计费。

删除开发环境后，可能需要几分钟时间才能更新存储配额。如果您已达到存储配额，则此时将无法创建新的开发环境。

> **Important**
>
> 删除开发环境的操作无法撤消。删除开发环境后，您无法再将其恢复。

删除开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中删除开发环境的项目。
3. 在导航窗格中，选择代码。
4. 选择开发环境。

5. 选择要删除的开发环境。
6. 选择删除。
7. 输入 **delete** 以确认删除开发环境。
8. 选择删除。

> **Note**
>
> 在删除空间中的 VPC 连接之前，请务必移除与该 VPC 关联的开发环境。
> 即使您删除开发环境，也可能无法删除 VPC 中的网络接口。确保根据需要清理资源。如果删除与 VPC 连接的开发环境时出错，则必须分离过时连接，并在确认未使用该连接后将其删除。

使用 SSH 连接到开发环境

您可以使用 SSH 连接到您的开发环境以不受限制地执行操作，例如端口转发、上传和下载文件以及使用其他 IDEs。

> **Note**
>
> 如果要在关闭 IDE 选项卡或窗口后继续使用 SSH 一段较长时间，请务必为开发环境设置较高的超时值，这样它就不会因在 IDE 中处于非活动状态而停止。

先决条件

- 您需要下列操作系统之一：
  - Windows 10 或更高版本且已启用 OpenSSH
  - macOS 和 Bash 版本 3 或更高版本
  - 带 `yum`、`dpkg` 或 `rpm` 包管理器以及 Bash 版本 3 或更高版本的 Linux
- 您还需要 AWS CLI 版本 2.9.4 或更高版本。

使用 SSH 连接到开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到要在其中使用 SSH 连接到开发环境的项目。
3. 在导航窗格中，选择代码。
4. 选择开发环境。
5. 使用 SSH 选择要连接到的正在运行的开发环境。
6. 选择通过 SSH 连接，选择所需的操作系统，然后执行以下操作：

   - 如果尚未这样做，请在指定的终端中粘贴并执行第一条命令。此命令将下载脚本并在本地环境中执行以下修改，以便您能够使用 SSH 连接到开发环境：
     - 安装会话管理器插件 AWS CLI
     - 修改您的本地 AWS Config 并添加 CodeCatalyst 个人资料，以便您可以执行 SSO 登录。有关更多信息，请参阅 设置为 AWS CLI 与一起使用 CodeCatalyst。
     - 修改您的本地 SSH 配置并添加使用 SSH 连接到开发环境所需的配置。
     - 在`~/.aws/codecatalyst-dev-env` 目录中添加 SSH 客户端用于连接到开发环境的脚本。此脚本调用 CodeCatalyst StartDevEnvironmentSession API 并使用 AWS Systems Manager Session Manager 插件与您的开发环境建立 AWS Systems Manager 会话，本地 SSH 客户端使用该会话安全地连接到远程开发环境。
   - 使用第二个命令 CodeCatalyst 使用 AWS SSO 登录 Amazon。此命令请求和检索凭证，以便`~/.aws/codecatalyst-dev-env`目录中的脚本可以调用 CodeCatalyst StartDevEnvironmentSession API。每当您的凭证过期时，都应执行此命令。当您在模式（ssh<destination>）中执行最后一条命令时，如果您的凭证已过期或者您尚未按照此步骤中的说明执行 SSO 登录，则会收到错误。
   - 使用第三条命令通过 SSH 连接到您指定的开发环境。此命令具有以下结构：

     ```
     ssh codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
     ```

     您还可以使用此命令执行 SSH 客户端所允许的其他操作，例如端口转发或上传和下载文件：

     - 端口转发：

       ```
       ssh -L <local-port>:127.0.0.1:<remote-port> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
       ```

     - 将文件上传到开发环境的主目录：

       ```
       scp -O </path-to-local-file> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>:</path-to-remote-file-or-directory>
       ```

配置开发环境的 devfile

devfile 是一种开放标准，可帮助您在整个团队中自定义开发环境。devfile 是一个 YAML 文件，用于对所需的开发工具进行编码。通过配置开发文件，您可以预先确定所需的项目工具和应用程序库，然后 Amazon CodeCatalyst 将它们安装到您的开发环境中。存储库与为它创建的 devfile 相对应，您可以为每个存储库创建一个单独的 devfile。您的开发环境支持命令和事件，并提供默认的通用 devfile 映像。

如果您使用空蓝图创建项目，则可以手动创建 devfile。如果您使用不同的蓝图创建项目，则会自动 CodeCatalyst 创建一个 devfile。开发环境的 `/projects` 目录存储从源存储库中提取的文件和 devfile。`/home` 目录（首次创建开发环境时，该目录为空）存储您在使用开发环境时创建的文件。开发环境的 `/projects` 和 `/home` 目录中的所有内容都将持久存储。

> **Note**
>
> 仅在更改 devfile 名称或 devfile 组件名称时，`/home` 文件夹才会更改。如果更改 devfile 或 devfile 组件名称，则 `/home` 目录的内容将被替换，并且无法恢复之前的 `/home` 目录数据。

如果您创建的开发环境的源存储库中的根目录不包含 devfile，或者创建的开发环境不具有源存储库，则系统会自动将默认的通用 devfile 应用于源存储库。所有 IDEs用户都使用相同的默认通用 devfile 镜像。 CodeCatalyst 目前支持 devfile 版本 2.0.0。有关 devfile 的更多信息，请参阅 Devfile schema - Version 2.0.0。

> **Note**
>
> 您只能在 devfile 中包含公共容器映像。

请注意，与 VPC 连接的开发环境仅支持以下 devfile 映像：

- 通用映像
- 私有 Amazon ECR 映像（如果存储库与 VPC 位于同一区域）

主题

- 编辑开发环境的存储库 devfile
- 支持的 Devfile 功能 CodeCatalyst
- 开发环境的 devfile 示例

- 使用恢复模式排查存储库 devfile 的问题
- 为开发环境指定通用 devfile 映像
- Devfile 命令
- Devfile 事件
- Devfile 组件

编辑开发环境的存储库 devfile

使用以下过程编辑开发环境的存储库 devfile。

在中编辑开发环境的存储库开发文件 CodeCatalyst

编辑存储库 devfile

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含要编辑其 devfile 的源存储库的项目。
3. 在导航窗格中，选择代码。
4. 选择源存储库。
5. 选择包含要编辑的 devfile 的源存储库。
6. 从文件列表中选择 `devfile.yaml` 文件。
7. 选择编辑。
8. 编辑 devfile。
9. 选择提交，或者创建拉取请求，以便团队成员能够审查和批准更改。

> **Note**
>
> 如果您编辑 devfile，则必须重新启动 devfile 以使更改生效。这可以通过运行 `/aws/mde/mde start --location devfile.yaml` 来完成。如果启动 devfile 时出现问题，它将进入恢复模式。但是，如果您编辑与 VPC 连接的开发环境关联的 devfile，则必须重新启动开发环境才能使更改生效。

您可以通过运行 `/aws/mde/mde status` 来查看正在使用的 devfile。位置字段包含 devfile 相对于环境的 `/projects` 文件夹的路径。

```
{
            "status": "STABLE",
            "location": "devfile.yaml"
        }
```

您还可以将 /projects/devfile.yaml 中的默认 devfile 移至您的源代码存储库。要更新 devfile 的位置，请使用以下命令：/aws/mde/mde start --location *repository-name*/devfile.yaml。

在 IDE 中编辑开发环境的存储库 devfile

要更改开发环境的配置，您必须编辑 devfile。我们建议您在支持的 IDE 中编辑开发文件，然后更新您的开发环境，但也可以从中源存储库的根目录编辑开发文件。 CodeCatalyst如果您在受支持的 IDE 中编辑 devfile，则必须提交更改并将其推送到源存储库或创建拉取请求，以便团队成员能够审查和批准 devfile 编辑内容。

- 在中编辑开发环境的存储库 devfile AWS Cloud9
- 在 VS Code 中编辑开发环境的存储库 devfile
- 在中编辑开发环境的存储库 devfile JetBrains

支持的 Devfile 功能 CodeCatalyst

CodeCatalyst 在 2.0.0 版本上支持以下开发文件功能。有关 devfile 的更多信息，请参阅 Devfile schema - Version 2.0.0。

功能	Type
exec	命令
postStart	事件
container	组件
args	组件属性
env	组件属性
mountSources	组件属性

功能	Type
volumeMounts	组件属性

开发环境的 devfile 示例

以下是简单 devfile 的示例。

```
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

将捕获 devfile 的启动、命令和事件日志并将其存储在 /aws/mde/logs 中。要调试 devfile 行为，请使用有效的 devfile 启动开发环境并访问日志。

使用恢复模式排查存储库 devfile 的问题

如果启动 devfile 时出现问题，它将进入恢复模式，这样您仍能连接到您的环境并修复 devfile。在恢复模式中，运行 `/aws/mde/mde status` 将不会包含 devfile 的位置。

```
{
            "status": "STABLE"
        }
```

您可以查看 `/aws/mde/logs` 下的日志中的错误，修复 devfile，然后重试运行 `/aws/mde/mde start`。

为开发环境指定通用 devfile 映像

默认通用映像包括可用于 IDE 的最常用的编程语言和相关工具。如果未指定映像，则 CodeCatalyst 会提供此映像并包含由 CodeCatalyst 维护的工具。要接收有关新映像版本的通知，请参阅使用 SNS 订阅通用映像通知。

Amazon CodeCatalyst 有效支持以下 devfile 映像：

映像版本	映像标识符
Universal image 4.0	public.ecr.aws/aws-mde/universal-image:4.0
Universal image 5.0	public.ecr.aws/aws-mde/universal-image:5.0

> **Note**
>
> 您还可以使用 `public.ecr.aws/aws-mde/universal-image:latest` 获取最新映像，当前为 `public.ecr.aws/aws-mde/universal-image:5.0`。

CodeCatalyst 已弃用以下映像。您仍然可以使用这些映像，但它们不会缓存在构建主机上，因此将导致开发环境的启动时间更长。

映像版本	映像标识符	弃用日期
Universal image 1.0	public.ecr.aws/aws-mde/universal-image:1.0	2024 年 8 月 16 日
Universal image 2.0	public.ecr.aws/aws-mde/universal-image:2.0	2024 年 8 月 16 日
Universal image 3.0	public.ecr.aws/aws-mde/universal-image:3.0	2025 年 7 月 30 日

> **Note**
>
> 如果您使用的是 AWS Cloud9，则在升级到 `universal-image:3.0` 后，自动完成功能将不适用于 PHP、Ruby 和 CSS。

主题

使用 SNS 订阅通用映像通知

CodeCatalyst 提供通用映像通知服务。您可以使用它订阅 Amazon Simple Notification Service（SNS）主题，以便在发布 CodeCatalyst 通用映像更新时收到通知。有关 SNS 主题的更多信息，请参阅 What is Amazon Simple Notification Service?。

当发布新的通用映像时，我们会向订阅用户发送通知；此部分介绍如何订阅 CodeCatalyst 通用映像更新。

示例消息

```
{
```

```
    "Type": "Notification",
    "MessageId": "123456789",
    "TopicArn": "arn:aws:sns:us-east-1:1234657890:universal-image-updates",
    "Subject": "New Universal Image Release",
    "Message": {
        "v1": {
            "Message": "A new version of the Universal Image has been released. You are
 now able to launch new DevEnvironments using this image.",
            "image ": {
                "release_type": "MAJOR VERSION",
                "image_name": "universal-image",
                "image_version": "2.0",
                "image_uri": "public.ecr.aws/amazonlinux/universal-image:2.0"
            }
        }
    },
    "Timestamp": "2021-09-03T19:05:57.882Z",
    "UnsubscribeURL": "example url"
}
```

使用 Amazon SNS 控制台订阅 CodeCatalyst 通用映像更新

1. 打开 Amazon SNS 控制台以显示控制面板。
2. 在导航栏中，选择您的 AWS 区域。
3. 在导航窗格中，选择订阅，然后选择创建订阅。
4. 在主题 ARN 中，输入 `arn:aws:sns:us-east-1:089793673375:universal-image-updates`。
5. 在协议中，选择电子邮件。
6. 在端点中，提供一个电子邮件地址。此电子邮件地址将用于接收通知。
7. 选择创建订阅。
8. 您将收到一封主题行为“AWS 通知 - 订阅确认”的确认电子邮件。打开这封电子邮件，然后选择确认订阅。

使用 Amazon SNS 控制台取消订阅 CodeCatalyst 通用映像更新

1. 打开 Amazon SNS 控制台以显示控制面板。
2. 在导航栏中，选择您的 AWS 区域。
3. 在导航窗格中，选择订阅，然后选择要取消的订阅。

4. 选择操作，然后选择删除订阅。
5. 选择删除。

通用映像 4.0 运行时版本

下表列出了对 `universal-image:4.0` 可用的运行时。

`universal-image:4.0` 运行时版本

运行时名称	版本	特定主要和最新次要版本
aws cli	2.1.1	`aws-cli: 2.x`
docker compose	2.17	`docker-compose: 2.x`
dotnet	8.0	`dotnet: 8.x`
golang	1.22	`golang: 1.x`
java	corretto21	`java: corretto21.x`
nodejs	20.6	`nodejs: 20.x`
php	8.2	`php: 8.x`
python	3.9	`python: 3.x`
	3.12	
ruby	3.3	`ruby: 3.x`
terraform	1.5	`terraform: 1.x`

通用映像 5.0 运行时版本

下表列出了对 `universal-image:5.0` 可用的运行时。

universal-image:5.0 运行时版本

运行时名称	版本	特定主要和最新次要版本
aws cli	2.25	`aws-cli: 2.x`
docker compose	2.34	`docker-compose: 2.x`
dotnet	8.0	`dotnet: 8.x`
golang	1.24	`golang: 1.x`
java	corretto21	`java: corretto21.x`
nodejs	22.0	`nodejs: 22.x`
php	8.3.16	`php: 8.x`
python	3.12	`python: 3.x`
ruby	3.4.2	`ruby: 3.x`
terraform	1.10.5	`terraform: 1.x`

Devfile 命令

目前，CodeCatalyst 仅支持你的开发文件中的`exec`命令。有关更多信息，请参阅 Devfile.io 文档中的 Adding commands。

以下示例说明如何在 devfile 中指定 `exec` 命令。

```
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
```

```
    exec:
      component: test
      commandLine: "yum -y update --security"
```

连接到开发环境后，您可以通过终端执行定义的命令。

```
/aws/mde/mde command <command-id>
/aws/mde/mde command executescript
```

对于长时间运行的命令，您可以使用流标志 `-s` 来实时输出命令的执行。

```
/aws/mde/mde -s command <command-id>
```

> **Note**
>
> `command-id` 必须小写。

支持的 exec 参数 CodeCatalyst

CodeCatalyst 在 devfile 版本 2.0.0 上支持以下`exec`参数。

- `commandLine`
- `component`
- `id`
- `workingDir`

Devfile 事件

目前， CodeCatalyst 仅支持开发文件中的`postStart`事件。有关更多信息，请参阅 devFile.io 文档postStartObject中的。

以下示例说明如何在 devfile 中添加 `postStart` 事件绑定。

```
commands:
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - updateyum
    - executescript
```

启动后，您的开发环境将按照指定的 `postStart` 命令的定义顺序来执行这些命令。如果命令失败，开发环境将继续运行，并且执行输出将存储在 `/aws/mde/logs` 下的日志中。

Devfile 组件

目前， CodeCatalyst 仅支持开发文件中的`container`组件。有关更多信息，请参阅 Devfile.io 文档中的 Adding components。

以下示例说明如何将启动命令添加到 devfile 中的容器。

```
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      command: ['sleep', 'infinity']
```

> **Note**
>
> 当容器具有短效入口命令时，您必须包含 `command: ['sleep', 'infinity']` 以使容器保持运行。

CodeCatalyst 还支持容器组件中的以下属性：`args`、`env`、`mountSources`、和`volumeMounts`。

将 VPC 连接关联到开发环境

VPC 连接是一种 CodeCatalyst 资源，其中包含您的工作流程访问 VPC 所需的所有配置。空间管理员可以代表空间成员在 Amazon CodeCatalyst 控制台中添加自己的 VPC 连接。通过添加 VPC 连接，空间成员可以运行工作流操作和创建符合网络规则的开发环境，并能访问已关联 VPC 中的资源。

> **Important**
>
> 具有 VPC 连接的开发环境不支持链接到的第三方源存储库 CodeCatalyst。

只有在创建开发环境后才能将开发环境关联到 VPC 连接。创建开发环境后，无法更改关联到开发环境的 VPC 连接。如果要使用其他 VPC 连接，则必须删除当前的开发环境并创建新的开发环境。

> **Note**
>
> 开发环境只能与具有项目访问权限的 AWS 账户的 VPC 连接关联。有关更多信息，请参阅《Amazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

请注意，开发环境在创建时会使用多种 AWS 资源和服务。这意味着开发环境可以连接到以下 AWS 服务：

- Amazon CodeCatalyst
- AWS SSM
- AWS KMS
- Amazon ECR
- Amazon CloudWatch
- Amazon ECS

> **Note**
>
> AWS Toolkit 不支持使用关联的 VPC 连接创建开发环境。另请注意，如果您使用除之外的 IDE AWS Cloud9，则加载时间可能约为五分钟。

您必须具有空间管理员角色或高级用户角色才能在空间级别管理 VPC 连接。有关更多信息 VPCs，请参阅《CodeCatalyst 管理员指南》 VPCs CodeCatalyst中的 “管理 Amazon”。

中开发环境的配额 CodeCatalyst

下表描述了 Amazon 中开发环境的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

每月开发环境小时数	开发环境的小时数受空间的总体存储限制的影响。有关更多信息，请参阅 Pricing 和排查开发环境的问题。
每个空间的开发环境存储量	开发环境存储受空间的总体存储限制的影响。有关更多信息，请参阅 Pricing 和排查开发环境的问题。
开发环境计算量	开发环境计算受空间的总体存储限制的影响。有关更多信息，请参阅 Pricing 和排查开发环境的问题。

在中发布和共享软件包 CodeCatalyst

Amazon CodeCatalyst 包含完全托管的软件包存储库服务，可让您的开发团队轻松安全地存储和共享用于应用程序开发的软件包。这些包存储在包存储库中，这些存储库是在中的项目中创建和组织的 CodeCatalyst。

单个软件包存储库可以存储每种支持的软件包类型的软件包。 CodeCatalyst 支持以下软件包格式：

- npm
- Maven
- NuGet
- Python

可以发现程序包存储库中的程序包，并在包含该存储库的项目的成员之间共享此程序包。

要将程序包发布到存储库和使用存储库中的程序包，请将程序包管理器配置为使用存储库端点（URL）。然后，您可以使用程序包管理器将程序包发布到存储库。您可以使用 Maven、Gradle、npm、yarn、nuget、dotnet、pip 和 twine 等程序包管理器。

您也可以将 CodeCatalyst 工作流程配置为使用 CodeCatalyst 软件包存储库。有关在工作流中使用程序包的更多信息，请参阅将程序包存储库连接到工作流。

您可以将一个程序包存储库中的程序包提供给同一项目中的另一个存储库，方法是将前者添加为上游存储库。上游存储库可用的所有程序包版本也可供下游存储库使用。有关更多信息，请参阅 配置并使用上游存储库。

您可以通过创建一种名为网关的特殊类型的存储 CodeCatalyst 库来向仓库提供开源软件包。上游到网关存储库允许您使用来自热门公共存储库（例如 npmjs.com 和 pypi.org）的软件包，并自动将其缓存在存储库中。 CodeCatalyst 有关更多信息，请参阅 连接到公共外部存储库。

主题

- 发布和修改程序包
- 使用 npm
- 使用 Maven
- 使用 NuGet
- 使用 Python
- 程序包配额

程序包概念

以下是管理、发布或使用中的软件包时需要了解的一些概念和术语 CodeCatalyst。

软件包

软件包是一个包含安装软件和解决任何依赖关系所需的软件和元数据的捆绑包。 CodeCatalyst 支持 npm 包格式。

程序包中包括：

- 一个名称（例如，webpack 是一个流行的 npm 程序包的名称）
- 一个可选的命名空间（例如，@types/node 中的 @types）
- 一组版本（例如，1.0.0、1.0.1、1.0.2）
- 程序包级别的元数据（例如 npm dist 标签）

程序包命名空间

某些程序包格式支持分层程序包名称，用于将程序包组织成逻辑组，有助于避免名称冲突。具有相同名称的程序包可以存储在不同的命名空间中。例如，npm 支持作用域，而 npm 程序包 @types/node 的作用域为 @types，名称为 node。@types 作用域中还有许多其他的程序包名称。在中 CodeCatalyst，作用域（“类型”）被称为包命名空间，名称（“节点”）被称为包名称。对于 Maven 程序包，程序包命名空间与 Maven GroupId 相对应。Maven 程序包 org.apache.logging.log4j:log4j 的 groupID（程序包命名空间）为 org.apache.logging.log4j，artifactID（程序包名称）为 log4j。某些程序包格式（例如 Python）不支持其概念类似于 npm 作用域或 Maven groupID 的分层名称。如果无法对程序包名称进行分组，则可能更难避免名称冲突。

程序包版本

程序包版本标识程序包的特定版本，例如 `@types/node@12.6.9`。版本号格式和语义因不同的程序包格式而异。例如，npm 程序包版本必须符合语义版本控制规范。在中 CodeCatalyst，软件包版本由版本标识符、 package-version-level元数据和一组资源组成。

资产

资产是存储在中与软件包版本关联 CodeCatalyst 的单个文件，例如 npm `.tgz` 文件或 Maven POM 或 JAR 文件。

程序包存储库

CodeCatalyst 包存储库包含一组包，其中包含软件包版本，每个版本都映射到一组资产。程序包存储库是多语言的，这意味着单个存储库可以包含任何受支持类型的程序包。每个包存储库都公开端点，用于使用 (、)、CLI `nuget`、Maven CL `npm` l NuGet CLIs (`dotnet`) 和 Pyth CLIs on `mvnpip`（`twine`和）等工具获取和发布包。有关中的 CodeCatalyst软件包配额（包括在每个空间中可以创建多少个软件包存储库）的信息，请参阅程序包配额。

您可以通过将一个程序包存储库设置为上游来将它链接到另一个程序包存储库。在将存储库设置为上游时，可以使用来自该上游的任何程序包以及链中的任何其他上游存储库。有关更多信息，请参阅 上游存储库。

网关存储库是一种特殊类型的程序包存储库，用于从官方外部程序包授权机构处拉取和存储程序包。有关更多信息，请参阅 网关存储库。

上游存储库

您可以使用 CodeCatalyst 在两个软件包存储库之间创建上游关系。当可以从下游存储库的程序包存储库端点访问其中的程序包版本时，一个程序包存储库将为另一个程序包存储库的上游。利用上游关系，可从客户端的角度有效地合并两个程序包存储库的内容。

例如，如果软件包管理器请求存储库中不存在的软件包版本，则 CodeCatalyst 会在已配置的上游存储库中搜索该软件包的版本。按配置顺序搜索上游存储库，一旦找到软件包， CodeCatalyst 就会停止搜索。

网关存储库

网关存储库是一种特殊类型的程序包存储库，它与支持的外部官方程序包授权机构相连。在将网关存储库添加为上游存储库时，可以从相应的官方程序包授权机构处使用程序包。您的下游存储库不与公有存

储库进行通信，相反，所有内容都由网关存储库进行中继。通过此方式使用的程序包同时存储在网关存储库和收到原始请求的下游存储库中。

网关存储库是预定义的，但必须在要使用的每个项目中创建它们。以下列表包含可以在中创建的每个网关存储库 CodeCatalyst 以及它们所连接的包权限。

- npm-public-registry-gateway提供来自 npmjs.com 的 npm 软件包。
- maven-central-gateway提供来自 Maven Central 存储库的 Maven 软件包。
- google-android-gateway提供来自谷歌安卓系统的 Maven 软件包。
- commonsware-gateway 提供的 Maven CommonsWare
- gradle-plugins-gateway提供来自 Gradle 插件的 Maven 软件包。
- nuget-gallery-gateway提供 NuGet 图库中的 NuGet 软件包。
- pypi-gateway 提供来自 Python 包索引的 Python 程序包。

配置并使用程序包存储库

在中 CodeCatalyst，软件包存储在软件包存储库中并进行管理。要将包发布到 CodeCatalyst 或使用来自 CodeCatalyst （或任何支持的公共包存储库）的包，您必须创建一个包存储库并将包管理器连接到该存储库。

主题

创建程序包存储库

要在中创建软件包存储库，请执行以下步骤 CodeCatalyst。

创建程序包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建程序包存储库的项目。
3. 在导航窗格中，选择程序包。

4. 在程序包存储库页面上，选择创建程序包存储库。
5. 在程序包存储库详细信息部分中，添加以下项：

   a. 存储库名称。考虑使用描述性名称，其中包含诸如项目或团队名称或存储库的使用方式之类的详细信息。

   b. （可选）描述。当您在一个项目中拥有跨多个团队的多个存储库时，存储库描述特别有用。
6. 在 “上游存储库” 部分，选择 “选择上游存储库”，添加要通过软件包存储库访问的任何 CodeCatalyst 软件包存储库。您可以添加 Gateway 存储库以连接到外部软件包存储库或其他CodeCatalyst 存储库。

   - 当从程序包存储库请求程序包时，将按照上游存储库在此列表中的显示顺序搜索这些上游存储库。找到包裹后， CodeCatalyst 将停止搜索。要更改上游存储库的顺序，您可以将存储库拖放到列表中。
7. 选择创建以创建您的程序包存储库。

连接到程序包存储库

要发布到或使用其中的软件包 CodeCatalyst，您必须使用软件包存储库端点信息和 CodeCatalyst 凭据配置软件包管理器。如果您尚未创建存储库，则可以按照创建程序包存储库中的说明执行此操作。

有关如何将包管理器连接到软件 CodeCatalyst 包存储库的说明，请参阅以下文档。

- 配置并使用 Gradle Groovy
- 配置并使用 mvn
- 配置并使用 nuget 或 dotnet CLI
- 配置并使用 npm
- 配置 pip 并安装 Python 程序包
- 配置 Twine 并发布 Python 程序包

删除程序包存储库

要在中删除软件包存储库，请执行以下步骤 CodeCatalyst。

删除程序包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到包含要删除的程序包存储库的项目。
3. 在导航窗格中，选择程序包。
4. 在程序包存储库页面上，选择要删除的存储库。
5. 选择删除。
6. 查看提供的有关删除程序包存储库的效果的信息。
7. 在输入框中输入 `delete`，然后选择删除。

配置并使用上游存储库

您可以将网关存储库和其他 CodeCatalyst 软件包存储库作为上游连接到您的软件包存储库。这样，程序包管理器客户端就可以使用单个程序包存储库端点访问多个程序包存储库中包含的程序包。以下是使用上游存储库的主要好处：

- 您只需要为程序包管理器配置一个存储库端点，即可从多个来源进行提取。
- 从上游存储库中使用的程序包存储在下游存储库中，这可确保即使上游存储库发生意外中断或上游存储库中的程序包被删除，您的程序包也可用。

在创建程序包存储库时，可以添加上游存储库。您还可以在 CodeCatalyst 控制台中从现有软件包存储库中添加或删除上游存储库。

当您将网关存储库添加为上游存储库时，程序包存储库将连接到网关存储库的相应公共程序包存储库。有关支持的公共程序包存储库的列表，请参阅支持的外部程序包存储库及其网关存储库。

您可以将多个存储库链接起来以作为上游存储库。例如，假设您的团队创建了一个名为的存储库，`project-repo`并且已经在使用另一个名为的存储库`team-repo`，该存储库已npm-public-registry-gateway添加为上游存储库，该存储库已连接到公共 npm 存储库。`npmjs.com`您可以将 `team-repo` 作为上游存储库添加到 `project-repo`。在此情况下，您只需将程序包管理器配置为使用 `project-repo` 从 `project-repo`、`team-repo`、`npm-public-registry-gateway` 和 `npmjs.com` 中拉取程序包即可。

主题

- 添加上游存储库
- 编辑上游存储库的搜索顺序
- 请求包含上游存储库的程序包版本
- 移除上游存储库

添加上游存储库

将公共包存储库或其他 CodeCatalyst 软件包存储库作为上游存储库添加到下游存储库中，可以将上游存储库中的所有包都提供给连接到下游存储库的包管理器。

添加上游存储库

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择要将上游存储库添加到的程序包存储库。
3. 在程序包存储库的名称下，选择上游，然后选择选择上游存储库。
4. 在选择上游类型中，选择下列选项之一：
   - 网关存储库

     您可以从可用网关存储库的列表中进行选择。

     > **Note**
     >
     > 要连接到公共外部包颁发机构，例如 Maven Central、npmjs.com 或 Nuget Gallery，请 CodeCatalyst 使用网关存储库作为中间存储库，用于搜索和存储从外部存储库提取的包。这样可以减少时间和数据传输，因为项目中的所有程序包存储库都将使用网关中间存储库中的程序包。有关更多信息，请参阅 连接到公共外部存储库。

   - CodeCatalyst 存储库

     您可以从项目中可用的 CodeCatalyst 软件包存储库列表中进行选择。
5. 选择所有要添加作为上游存储库的存储库后，选择选择，然后选择保存。

   有关更改上游存储库的搜索顺序的更多信息，请参阅编辑上游存储库的搜索顺序。

添加上游存储库后，您可以使用连接到本地存储库的程序包管理器，从该上游存储库中获取程序包。您无需更新程序包管理器配置。有关从上游存储库请求程序包版本的更多信息，请参阅请求包含上游存储库的程序包版本。

编辑上游存储库的搜索顺序

CodeCatalyst 按其配置的搜索顺序搜索上游存储库。找到包裹后， CodeCatalyst 停止搜索。您可以更改在上游存储库中搜索程序包时搜索这些存储库的顺序。

编辑上游存储库的搜索顺序

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择要编辑器上游存储库搜索顺序的程序包存储库。
3. 在程序包存储库的名称下，选择上游。
4. 在上游存储库部分中，您可以查看上游存储库及其搜索顺序。要更改搜索顺序，请将存储库拖放到列表中。
5. 编辑完上游存储库的搜索顺序后，选择保存。

请求包含上游存储库的程序包版本

以下示例显示了包管理器从具有上游存储库的软件包存储库请求 CodeCatalyst 包时可能出现的情况。

在此示例中，程序包管理器（例如 npm）从名为 downstream 的具有多个上游存储库的程序包存储库中请求程序包版本。在请求程序包时，可能会出现以下情况：

- 如果 downstream 包含请求的程序包版本，则将该版本返回给客户端。
- 如果downstream不包含请求的软件包版本，则按其配置downstream的 CodeCatalyst 搜索顺序在上游存储库中搜索该版本。如果找到了程序包版本，则会将对该版本的引用复制到 downstream，并将程序包版本返回给客户端。
- 如果 downstream 和其上游存储库都不包含程序包版本，则会向客户端返回 HTTP 404 Not Found 响应。

一个存储库允许的直接上游存储库的最大数量为 10。请求软件包版本时 CodeCatalyst 搜索的最大存储库数为 25。

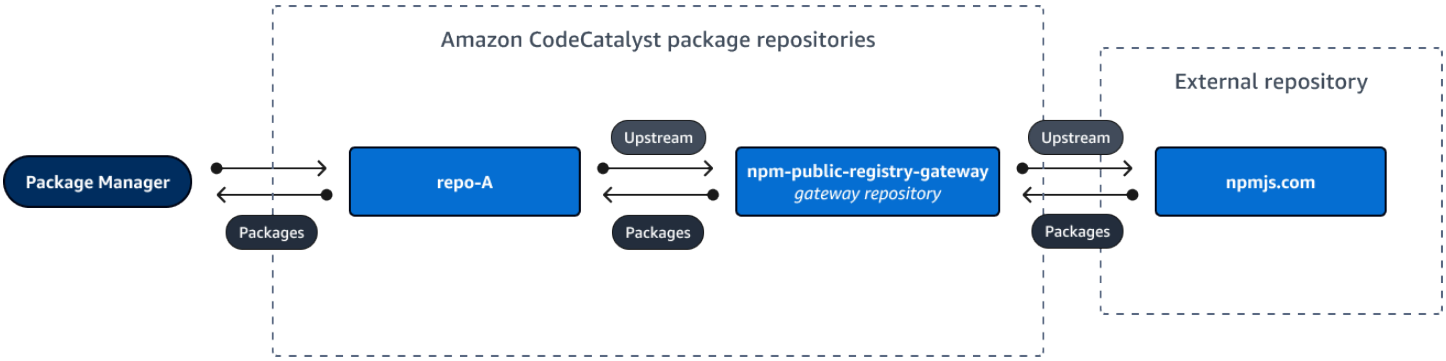
上游存储库的程序包保留

如果在上游存储库中找到了请求的程序包版本，则会保留对该版本的引用，并且该引用在请求它的存储库中始终可用。这将确保您能够在上游存储库意外中断时访问您的程序包。保留的程序包版本不受以下任何因素的影响：

- 删除上游存储库。
- 断开上游存储库与下游存储库的连接。
- 从上游存储库中删除程序包版本。
- 在上游存储库中编辑程序包版本（例如，向其中添加新资产）。

通过上游关系提取程序包

CodeCatalyst 可以通过多个名为上游存储库的链接存储库获取软件包。如果一个 CodeCatalyst 包存储库与另一个包存储库有上游连接，而另一个 CodeCatalyst 包存储库与网关存储库有上游连接，则会从外部存储库中复制对不在上游存储库中的软件包的请求。例如，考虑以下配置：名为的存储库与网关存储库repo-A有上游连接npm-public-registry-gateway。 npm-public-registry-gateway与公共软件包存储库有上游连接https://npmjs.com。



如果配置npm为使用repo-A存储库，则运行npm install会启动将软件包从复制https://npmjs.com到npm-public-registry-gateway。已安装的版本也会提取到 repo-A。下面的示例会安装 lodash。

```
$ npm config get registry
https://packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/
$ npm install lodash
+ lodash@4.17.20
added 1 package from 2 contributors in 6.933s
```

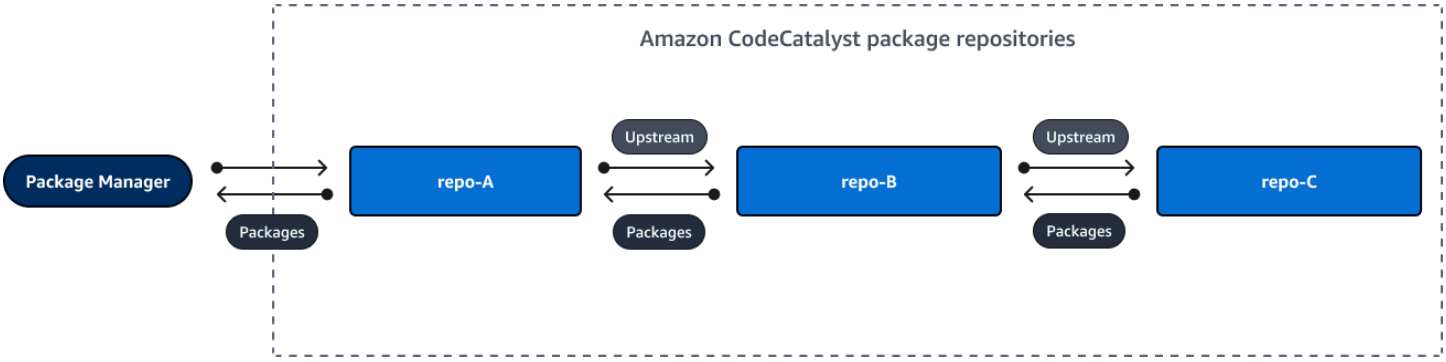
运行 npm install 后，repo-A 仅包含最新版本（lodash 4.17.20），因为这是 npm 从 repo-A 提取的版本。

由于与npm-public-registry-gateway有外部上游连接 https://npmjs.com，因此所有从中导入的软件包版本https://npmjs.com都存储在中npm-public-registry-gateway。任何具有指向 npm-public-registry-gateway 的上游连接的下游存储库，均可能已提取这些程序包版本。

的内容为您npm-public-registry-gateway提供了一种查看一段时间以来导入的所有软件包和软件包版本的方法。https://npmjs.com

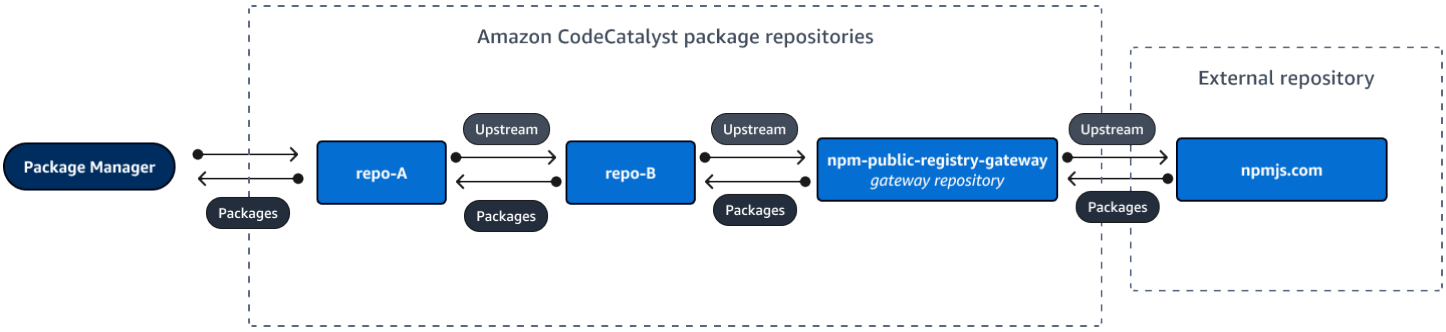
中间存储库中的程序包保留

CodeCatalyst 允许您链接上游存储库。例如，repo-A 可以使用 repo-B 作为上游存储库，而 repo-B 可以使用 repo-C 作为上游存储库。这个配置意味着在 repo-A 中可以获取 repo-B 和 repo-C 中的程序包版本。



当程序包管理器连接到存储库 repo-A 并从存储库 repo-C 中提取程序包版本时，程序包版本不会保留在存储库 repo-B 中。程序包版本仅保留在最下游的存储库中，在此示例中为 repo-A。程序包版本不会保留在任何中间存储库中。对于较长的链也是如此；例如，如果有四个存储库 repo-A、repo-B、repo-C 和 repo-D，一个连接到 repo-A 的程序包管理器从 repo-D 提取程序包版本，则程序包版本将保留在 repo-A 中，但不会保留在 repo-B 或 repo-C 中。

从公共程序包存储库拉取程序包版本时，程序包保留行为与之类似，不同之处在于程序包版本始终保留在具有与公共存储库的直接上游连接的网关存储库中。例如，repo-A 使用 repo-B 作为上游存储库。repo-B 使用 npm-public-registry-gateway 作为上游存储库，它具有与公共存储库 npmjs.com 的上游连接；请参见下图。



如果连接到 repo-A 的程序包管理器请求特定的程序包版本（例如，lodash 4.17.20），而三个存储库中的任何一个都不存在该程序包版本，则将从 npmjs.com 提取该程序包版本。当提取 lodash 4.17.20 时，该版本将保留在 repo-A（因为这是最下游的存储库）和 npm-public-registry-gateway（因为它具有与公共外部存储库 npmjs.com 的上游连接）中。lodash 4.17.20 不会保留在 repo-B 中（因为这是中间存储库）。

移除上游存储库

如果您不再需要访问上游存储库中的程序包，可以从程序包存储库中移除上游存储库。

> **Warning**
> 移除上游存储库时，可能会中断上游关系链，从而破坏您的项目或构建。

移除上游存储库

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择要从中移除上游存储库的程序包存储库。
3. 在程序包存储库的名称下，选择上游。
4. 在编辑上游存储库部分中，找到要移除的上游存储库，然后选择 。
5. 完成对上游存储库的移除后，选择保存。

连接到公共外部存储库

您可以通过将相应的网关存储库添加为上游存储库，将 CodeCatalyst 软件包存储库连接到支持的公共外部存储库。网关存储库充当中间存储库，用于搜索和存储从外部存储库提取的程序包。这样可以减少时间和数据传输，因为项目中的所有程序包存储库都可以使用网关存储库中存储的程序包。

使用网关存储库连接到公共存储库

1. 在导航窗格中，选择程序包。
2. 在程序包中，选择网关存储库页面。您可以查看受支持的网关存储库及其描述的列表。
3. 必须先创建一个网关存储库，之后才能使用该网关存储库。如果已创建网关存储库，则将显示其创建日期和时间。如果未创建，请选择创建以创建它。
4. 要使用网关存储库中的软件包，必须设置从存储库到网关 CodeCatalyst 存储库的上游连接。选择程序包存储库，然后选择要连接到的程序包存储库。
5. 要连接到公共存储库，请选择上游，然后选择选择上游存储库。
6. 选择网关存储库，选择与要连接到的公共存储库对应的网关存储库作为上游存储库。
7. 选择所有要添加为上游存储库的网关存储库后，选择选择。

8. 完成对上游存储库的排序后，选择保存。

有关上游存储库的更多信息，请参阅配置并使用上游存储库。

在将网关存储库添加为上游存储库后，您可以使用连接到本地存储库的程序包管理器从与其对应的公共外部程序包存储库中获取程序包。您无需更新程序包管理器配置。通过此方式使用的程序包同时存储在网关存储库和本地程序包存储库中。有关从上游存储库请求程序包版本的更多信息，请参阅请求包含上游存储库的程序包版本。

支持的外部程序包存储库及其网关存储库

CodeCatalyst 支持使用网关存储库向以下官方包授权机构添加上游连接。

存储库程序包类型	说明	网关存储库名称
npm	npm 公有注册表	`npm-public-registry-gateway`
Python	Python 包索引	`pypi-gateway`
Maven	Maven Central	`maven-central-gateway`
Maven	Google Android 存储库	`google-android-gateway`
Maven	CommonsWare	`commonsware-gateway`
Maven	Gradle 插件存储库	`gradle-plugins-gateway`
NuGet	NuGet 画廊	`nuget-gallery-gateway`

发布和修改程序包

中的软件包 CodeCatalyst 是解决依赖关系和安装软件所需的软件和元数据的捆绑包。有关中支持的软件包格式列表 CodeCatalyst，请参阅在中发布和共享软件包 CodeCatalyst。本节提供有关发布、查看、删除程序包和更新程序包版本状态的信息。

主题

将包发布到 CodeCatalyst 包存储库

您可以使用包管理器工具将任何支持的软件 CodeCatalyst 包类型的版本发布到包存储库。发布程序包版本的步骤如下：

将软件包版本发布到 CodeCatalyst 包存储库

1. 如果还没有程序包存储库，请创建一个程序包存储库。
2. 将您的程序包管理器连接到您的程序包存储库。有关如何将 npm 包管理器连接到软件 CodeCatalyst 包存储库的说明，请参阅配置并使用 npm。
3. 使用连接的程序包管理器发布您的程序包版本。

目录

发布和上游存储库

在中 CodeCatalyst，您无法发布存在于可访问的上游存储库或公共存储库中的软件包版本。例如，假设您要将 npm 程序包 `lodash@1.0` 发布到程序包存储库 `myrepo`，并且 `myrepo` 通过配置为上

游存储库的网关存储库连接到 npmjs.com。如果存在`lodash@1.0`于上游存储库或 npmjs.com 中，则`myrepo`通过发出 CodeCatalyst 409 冲突错误来拒绝任何向其发布的尝试。这有助于防止您意外发布与上游存储库中的程序包的名称和版本相同的程序包，这可能会导致意外行为。

您仍可以发布具有上游存储库中存在的程序包名称的其他版本。例如，如果 `lodash@1.0` 存在于上游存储库中，但 `lodash@1.1` 不存在该存储库中，则可以将 `lodash@1.1` 发布到下游存储库。

私有程序包和公有存储库

CodeCatalyst 不会将存储在存储 CodeCatalyst 库中的软件包发布到公共存储库，例如 npmjs.com 或 Maven Central。 CodeCatalyst 将软件包从公共存储库导入 CodeCatalyst 存储库，但它不会将软件包朝相反的方向移动。您发布到 CodeCatalyst 存储库的包将保持私有状态，并且仅适用于存储库所属的 CodeCatalyst 项目。

覆盖程序包资产

您无法重新发布已存在且包含不同内容的程序包资产。例如，假定您已经发布了一个具有 JAR 资产 `mypackage-1.0.jar` 的 Maven 程序包。仅当新旧资产的校验和完全相同时，您才能再次发布该资产。要重新发布包含新内容的相同资产，请先删除该程序包版本。尝试重新发布具有不同内容但名称相同的资产会导致出现 HTTP 409 冲突错误。

对于支持多种资产（Python 和 Maven）的程序包格式，您可以向现有程序包版本添加使用不同名称的新资产，前提是您拥有所需的权限。由于 npm 和每个软件包版本 NuGet 仅支持一个资产，因此要修改已发布的软件包版本，必须先将其删除。

如果您尝试重新发布已存在的资产（例如 `mypackage-1.0.jar`），并且已发布资产和新资产的内容相同，则因为该操作具有幂等性，所以操作会成功。

查看程序包版本详细信息

您可以使用 CodeCatalyst 控制台查看有关特定软件包版本的详细信息。

查看程序包版本详细信息

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择包含要查看其详细信息的程序包版本的存储库。
3. 在程序包表中搜索程序包版本。您可以使用搜索栏按程序包名称和格式筛选程序包。从列表中选择程序包。
4. 在程序包详细信息页面上，选择版本，然后选择您要查看的版本。

删除程序包版本

您可以从 CodeCatalyst 控制台的 Package 版本详情页面删除软件包版本。

删除程序包版本

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择包含要删除的程序包版本的存储库。
3. 从表中搜索并选择程序包。
4. 在程序包详细信息页面上，选择版本，然后选择您要删除的版本。
5. 在程序包版本详细信息页面上，选择版本操作，然后选择删除。
6. 在文本字段中输入删除，然后选择删除。

更新程序包版本的状态

中的每个软件包版本都 CodeCatalyst 有一个描述软件包版本的当前状态和可用性的状态。您可以在CodeCatalyst控制台中更改软件包版本状态。有关程序包版本可能的状态值及其含义的更多信息，请参阅程序包版本状态。

更新程序包版本的状态

1. 在导航窗格中，选择程序包。
2. 在程序包存储库页面上，选择包含要更新状态的程序包版本的存储库。
3. 从表中搜索并选择程序包。
4. 在程序包详细信息页面上，选择版本，然后选择要查看的版本。
5. 在程序包版本详细信息页面上，选择操作，然后选择未列出、存档或处置。有关每种程序包版本状态的更多信息，请参阅程序包版本状态。
6. 在文本字段中输入确认文本，然后根据要更新到的状态，选择未列出、存档或处置。

程序包版本状态

程序包版本状态的可能值如下所示。您可以在控制台中更改程序包版本状态。有关更多信息，请参阅更新程序包版本的状态。

- 已发布 – 已成功发布程序包版本，可以使用程序包管理器来请求版本。程序包版本将包括在返回给程序包管理器的程序包版本列表中，例如，在 `npm view <package-name> versions` 的输出中。程序包版本的所有资产均可从存储库中获得。
- 未完成：上次发布尝试未完成。当前，只有 Maven 程序包版本可以处于未完成状态。当客户端上传程序包版本的一个或多个资源，但没有为包括该版本的程序包发布 `maven-metadata.xml` 文件时，就会发生这种情况。
- 未列出：程序包版本的资产可从存储库下载，但该程序包版本未包含在向程序包管理器返回的版本列表中。例如，对于 npm 程序包，`npm view <package-name> versions` 的输出不包括该程序包版本。因为在可用版本列表中未显示该版本，这意味着 npm 的依赖项解析逻辑不会选择该程序包版本。但是，如果 `npm package-lock.json` 文件中已经引用了未列出的程序包版本，则仍然可以下载和安装该版本，例如在运行 `npm ci` 时。
- 已存档：该程序包版本的资产无法再下载。在返回给程序包管理器的版本列表中不会包括该程序包版本。由于资产不可用，因此会阻止客户端使用程序包版本。如果您的应用程序构建依赖于更新为已存档的版本，那么构建就会失败，除非该程序包版本已在本地缓存。您不能使用程序包管理器或构建工具来重新发布已存档的程序包版本，因为它仍然存在于存储库中。但是，您可以在控制台中将程序包版本状态更改回未列出或已发布。
- 已处置：程序包版本未显示在列表中，也无法从存储库下载资产。“已处置” 和 “已存档” 之间的主要区别在于，如果状态为 “已处置”，则软件包版本的资产将被永久删除 CodeCatalyst。因此，您无法将程序包版本从已处置更改为已存档、未列出或已发布。由于已删除资产，因此无法使用该程序包版本。将程序包版本标记为已处置后，您无需再支付程序包资产的存储费用。

除了前面列表中的状态外，程序包版本还可以删除。删除程序包版本后，存储库中将没有该版本，您可以使用程序包管理器或构建工具随意地重新发布该程序包版本。

程序包名称、程序包版本和资产名称规范化

CodeCatalyst 在存储软件包名称、软件包版本和资源名称之前对其进行标准化，这意味着中的名称或版本 CodeCatalyst 可能与发布软件包时提供的名称或版本不同。有关如何标准化每种软件包类型的名称和版本 CodeCatalyst 的更多信息，请参阅以下文档。

- Python 程序包名称规范化
- NuGet 软件包名称、版本和资产名称标准化

CodeCatalyst 不对其他包格式执行标准化。

编辑程序包来源控制

在 Amazon 中 CodeCatalyst，可以通过直接发布软件包版本、从上游存储库中提取包版本或通过网关从外部公共存储库摄取包版本来将其添加到包存储库中。如果您允许通过直接发布和从公有存储库摄取来添加程序包的版本，则您会容易受到依赖项替换攻击。有关更多信息，请参阅依赖项替换攻击。为了保护自己免受依赖项替换攻击，请对存储库中的程序包配置程序包来源控制，从而限制将该程序包的版本添加到存储库的方式。

您应考虑配置程序包来源控制以使不同程序包的新版本同时来自内部来源（例如直接发布）和外部来源（例如公有存储库）。默认情况下，根据程序包的第一个版本添加到存储库的方式来配置程序包来源控制。

程序包来源控制设置

使用程序包来源控制，您可以配置将程序包版本添加到存储库的方式。以下列表包括可用的程序包来源控制设置和值。

发布

此设置配置了是否可以使用程序包管理器或类似工具将程序包版本直接发布到存储库。

- 允许：可以直接发布程序包版本。
- 阻止：不可以直接发布程序包版本。

上游

此设置配置了在程序包管理器发出请求时，是从外部的公有存储库中摄取程序包版本，还是在上游存储库中保留程序包版本。

- 允许：任何软件包版本都可以从配置为上游 CodeCatalyst 存储库的其他存储库中保留，也可以通过外部连接从公共来源获取。
- BLOCK：Package 版本不能从配置为上游存储 CodeCatalyst 库的其他存储库中保留，也不能从具有外部连接的公共来源获取。

默认程序包来源控制设置

程序包的默认程序包来源控制基于该程序包的第一个版本添加到程序包存储库中的方式。

- 如果第一个程序包版本由程序包管理器直接发布，则设置将为发布：允许和上游：阻止。

- 如果第一个程序包版本是从公有来源摄取，则设置将为发布：阻止和上游：允许。

常见的程序包访问控制场景

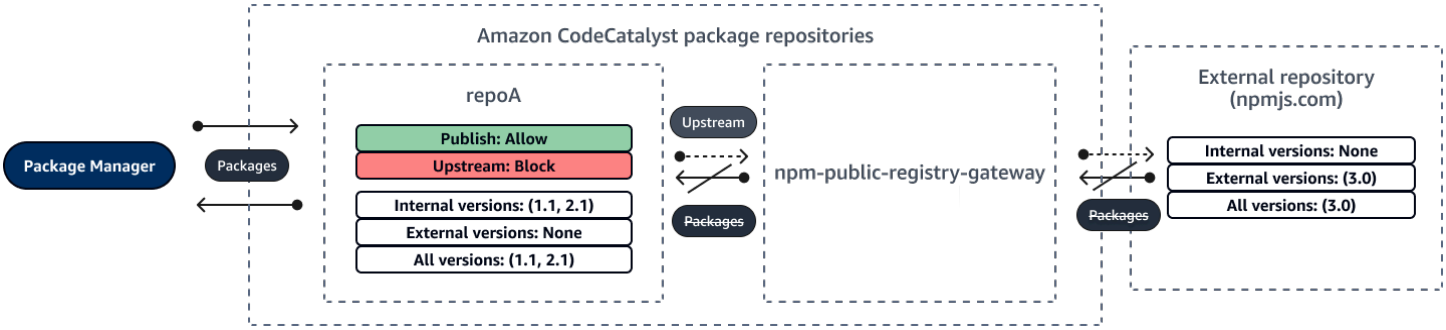
本节介绍将软件包版本添加到软件 CodeCatalyst 包存储库时的一些常见场景。根据第一个程序包版本的添加方式为新程序包设置程序包来源控制设置。

在以下场景中，内部程序包会直接从程序包管理器发布到存储库，例如您维护的程序包。外部程序包是存在于公有存储库中的程序包，可以通过网关存储库上游将其摄取到您的存储库中。

为现有内部程序包发布了外部程序包版本

在此场景中，考虑一个内部程序包 packageA。您的团队将 PackageA 的第一个软件包版本发布到 CodeCatalyst 软件包存储库。由于这是该程序包的第一个程序包版本，因此程序包来源控制设置会自动设置为发布：允许和上游：阻止。在您的存储库中发布软件包后，同名的包将发布到与您的软件 CodeCatalyst 包存储库连接的公共存储库中。这可能是针对内部程序包的企图依赖项替换攻击，也可能只是巧合。无论如何，配置程序包来源控制来阻止摄取新的外部版本，从而保护自己免受潜在的攻击。

在下图中，RepoA 是您的 CodeCatalyst 软件包存储库，与存储库有上游连接。`npm-public-registry-gateway`您的存储库包含 packageA 的版本 1.1 和 2.1，但版本 3.0 已发布到公有存储库。通常，repoA 会在程序包管理器请求程序包后摄取版本 3.0。由于软件包提取设置为 “阻止”，因此版本 3.0 不会提取到您的 CodeCatalyst 软件包存储库中，也无法供与其连接的包管理员使用。

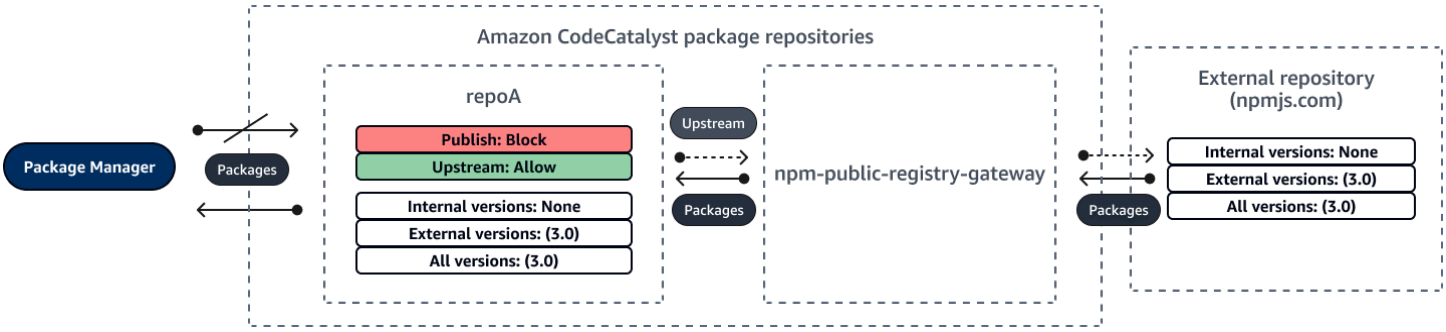


已为现有外部程序包发布内部程序包版本

在此场景中，在外部的公有存储库中有一个名为 packageB 的程序包，已将该公有存储库连接到您的存储库。当连接到您的存储库的程序包管理器请求 packageB 时，从公有存储库中将程序包版本摄取到您的存储库中。由于这是 packageB 添加到存储库中的第一个程序包版本，因此程序包来源设置配置为发布：阻止和上游：允许。稍后，您尝试将具有相同程序包名称的版本发布到存储库。您可能不知道该公共软件包并试图发布一个不相关的同名包，或者您可能正在尝试发布已修补的版本，或者您可能正在尝

试直接发布外部已经存在的确切软件包版本。 CodeCatalyst 拒绝您尝试发布的版本，但如有必要，您可以明确覆盖拒绝并发布该版本。

在下图中，RepoA 是您的 CodeCatalyst 软件包存储库，与存储库有上游连接。npm-public-registry-gateway您的程序包存储库包含从公有存储库中摄取的版本 3.0。您想将版本 1.2 发布到您的程序包存储库。通常，您可以将版本 1.2 发布到 repoA，但是由于发布设置为阻止，因此无法发布版本 1.2。



发布现有外部程序包的已修补程序包版本

在此场景中，在外部的公有存储库中有一个名为 packageB 的程序包，已将该公有存储库连接到您的程序包存储库。当连接到您的存储库的程序包管理器请求 packageB 时，从公有存储库中将程序包版本摄取到您的存储库中。由于这是 packageB 添加到存储库中的第一个程序包版本，因此程序包来源设置配置为发布：阻止和上游：允许。您的团队决定将此程序包的已修补程序包版本发布到存储库。为了能够直接发布程序包版本，您的团队将程序包来源控制设置更改为发布：允许和上游：阻止。此程序包的版本现在可以直接发布到您的存储库并从公有存储库中摄取。在您的团队发布已修补的程序包版本后，您的团队会将程序包来源设置恢复为发布：阻止和上游：允许。

编辑程序包来源控制

根据程序包的第一个程序包版本添加到程序包存储库的方式来自动配置程序包来源控制。有关更多信息，请参阅 默认程序包来源控制设置。要在包存储库中为包添加或编辑 CodeCatalyst 包源控件，请执行以下过程中的步骤。

添加或编辑程序包来源控制

1. 在导航窗格中，选择程序包。
2. 选择包含要编辑的程序包的程序包存储库。
3. 在程序包表中，搜索并选择要编辑的程序包。
4. 在程序包摘要页面中，选择来源控制。

5. 在来源控制中，选择要为此程序包设置的程序包来源控制。必须同时设置两个程序包来源控制设置（发布和上游）。

   - 要允许直接发布程序包版本，请在发布中选择允许。要阻止发布程序包版本，请选择阻止。
   - 要允许从外部存储库摄取程序包和从上游存储库提取程序包，请在上游来源中选择允许。要阻止所有从外部存储库和上游存储库进行的程序包版本摄取和提取，请选择阻止。

6. 选择保存。

发布和上游存储库

在中 CodeCatalyst，您无法发布存在于可访问的上游存储库或公共存储库中的软件包版本。例如，假设您要将 npm 程序包 `lodash@1.0` 发布到存储库 `myrepo`，并且 `myrepo` 有一个与 npmjs.com 进行外部连接的上游存储库。考虑以下场景。

1. `lodash` 上的程序包来源控制设置为发布：允许和上游：允许。如果存在`lodash@1.0`于上游存储库或 npmjs.com 中，则`myrepo`通过发出 CodeCatalyst 409 冲突错误来拒绝任何向其发布的尝试。您仍然可以发布另一个版本，例如 `lodash@1.1`。
2. `lodash` 上的程序包来源控制设置为发布：允许和上游：阻止。您可以将 `lodash` 的任何版本发布到由于无法访问程序包版本而尚不存在该版本的存储库中。
3. `lodash` 上的程序包来源控制设置为发布：阻止和上游：允许。您不能直接将任何程序包版本发布到您的存储库。

依赖项替换攻击

程序包管理器简化了打包和共享可重用代码的过程。这些程序包可能是组织为了在其应用程序中使用而开发的私有程序包，也可能是公有程序包（通常是在组织外部开发并由公有程序包存储库分发的开源程序包）。在请求程序包时，开发人员依靠他们的程序包管理器来提取其依赖项的新版本。依赖项替换攻击也称为依赖项混淆攻击，该攻击利用了这样一个事实，即程序包管理器通常无法区分程序包的合法版本和恶意版本。

依赖项替换攻击是被称为软件供应链攻击的黑客攻击的其中一种形式。软件供应链攻击是一种利用软件供应链中任何地方的漏洞进行的攻击。

依赖项替换攻击可以针对任何人，包括使用内部开发的程序包和从公有存储库提取的程序包的用户。攻击者识别内部程序包名称，然后策略性地将同名的恶意代码放置在公有程序包存储库中。通常，恶意代码在版本号较高的程序包中发布。因为程序包管理器认为恶意程序包是程序包的最新版本，所以从这些

公有源中提取恶意代码。这种行为会导致期望程序包和恶意程序包之间发生“混淆”或“替换”，进而导致代码被篡改。

为了防止依赖替换攻击，Amazon CodeCatalyst 提供了包裹来源控制。程序包来源控制是用于控制如何将程序包添加到存储库的设置。将新软件包的第一个软件包版本添加到 CodeCatalyst 存储库时，会自动配置控件。这些控件可以确保软件包版本不能既直接发布到您的存储库，也不能从公共来源获取，从而保护您免受依赖关系替换攻击。有关程序包来源控制以及如何更改该设置的更多信息，请参阅编辑程序包来源控制。

使用 npm

这些主题描述了如何使用 npm Node.js 软件包管理器 CodeCatalyst。

> **Note**
>
> CodeCatalyst 支持`node v4.9.1`以及以后`npm v5.0.0`和以后。

主题

- 配置并使用 npm
- npm 标签处理

配置并使用 npm

要npm与一起使用 CodeCatalyst，您必须npm连接到软件包存储库并提供用于身份验证的个人访问令牌 (PAT)。您可以在 CodeCatalyst 控制台中查看有关npm连接到软件包存储库的说明。

目录

- 不受支持的命令

使用以下命令配置 npm CodeCatalyst

以下说明说明了如何进行身份验证并npm连接到您的 CodeCatalyst 软件包存储库。有关更多信息，请参阅 npm 官方文档。

npm连接到您的 CodeCatalyst 软件包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。
3. 在导航窗格中，选择程序包。
4. 从列表中选择您的程序包存储库。
5. 选择连接到存储库。
6. 在配置详细信息中，在程序包管理器客户端中，选择 npm 客户端。
7. 选择您的操作系统以查看相应的配置步骤。
8. 需要个人访问令牌 (PAT) 才能对 npm 进行 CodeCatalyst身份验证。如果您已有令牌，则可使用它。如果没有，您可以使用以下步骤创建一个令牌。

   a. （可选）：更新 PAT 名称和到期日期。

   b. 选择创建令牌。

   c. 将 PAT 复制并存储在安全位置。

   > **Warning**
   >
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。凭证应是短期有效的，以最大限度地减少攻击者在盗用凭证后有效地使用凭证的时间。

9. 从项目的根目录运行以下命令，对程序包存储库配置 npm。这些命令将执行以下操作：

   - 如果您的项目没有项目级 `.npmrc` 文件，则会创建该文件。
   - 将程序包存储库端点信息添加到您的项目级 `.npmrc` 文件中。
   - 将您的凭证（PAT）添加到您的用户级 `.npmrc` 文件中。

   替换以下值。

> **Note**
>
> 如果通过控制台指令进行复制，则以下命令中的值将进行更新且无需更改。

- *username*用您的 CodeCatalyst 用户名替换。
- *PAT*用您的 CodeCatalyst PAT 替换。
- *space_name*替换为您的 CodeCatalyst 空间名称。
- *proj_name*用您的 CodeCatalyst 项目名称替换。
- *repo_name*替换为你的 CodeCatalyst 软件包存储库名称。

```
npm set registry=https://packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/ --location project
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/:_authToken=username:PAT
```

对于 npm 6 或更低版本：要让 npm 始终将身份验证令牌传递给 CodeCatalyst，即使是GET请求也是如此，请按如下方式设置 always-auth 配置变量。npm config set

```
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/:always-auth=true --location project
```

从软件包存储库安装 npm CodeCatalyst 软件包

在按照使用以下命令配置 npm CodeCatalyst 中的步骤操作来将 npm 连接到存储库后，您可以在存储库上运行 npm 命令。

您可以使用npm install命令安装位于软件 CodeCatalyst 包存储库或其上游存储库之一中的 npm 软件包。

```
npm install lodash
```

通过 npmjs 安装 npm 软件包 CodeCatalyst

你可以通过 CodeCatalyst 存储库从 npmjs.com 安装来自 npmjs.com 的 npm 软件包，方法是将存储库配置为与连接到 npmjs.com 的网关存储库的上游连接。npm-public-registry-gateway从 npmjs 安装的程序包被提取并存储在网关存储库和最远的下游程序包存储库中。

从 npmjs 安装程序包

1. 如果您尚未执行此操作，请npm按照中的步骤使用 CodeCatalyst软件包存储库进行配置使用以下命令配置 npm CodeCatalyst。
2. 检查您的存储库是否已将网关存储库添加为上游连接。npm-public-registry-gateway按照中的说明添加上游存储库并选择npm-public-registry-gateway存储库，您可以检查添加了哪些上游源或npm-public-registry-gateway将其添加为上游源。
3. 使用 `npm install` 命令安装程序包。

```
npm install package_name
```

有关从上游存储库请求程序包的更多信息，请参阅请求包含上游存储库的程序包版本。

将 npm 包发布到你的软件 CodeCatalyst 包存储库

完成使用以下命令配置 npm CodeCatalyst 后，您可以运行 npm 命令。

你可以使用`npm publish`命令将 npm 包发布到 CodeCatalyst 软件包存储库。

```
npm publish
```

有关如何创建 npm 程序包的信息，请参阅 npm Docs 上的 Creating Node.js Modules。

npm 命令支持

以下各节除了列出不支持的特定npm命令外，还总结了 CodeCatalyst软件包存储库支持的命令。

主题

- 与程序包存储库进行交互的受支持命令
- 支持的客户端命令
- 不受支持的命令

与程序包存储库进行交互的受支持命令

此部分列出了 npm 命令，其中 npm 客户端向其配置到的注册表（例如 `npm config set registry`）发出一个或多个请求。这些命令已经过验证，在针对 CodeCatalyst 软件包存储库调用时可以正常运行。

命令	说明
bugs	猜测程序包的错误跟踪器 URL 的位置，然后尝试打开它。
ci	从零开始安装一个项目。
deprecate	弃用程序包的某个版本。
dist-tag	修改程序包分发标签。
docs	猜测程序包的文档 URL 的位置，然后尝试使用 `--browser` 配置参数打开它。
doctor	运行一组检查以验证你的 npm 安装是否可以管理你的 JavaScript 软件包。
install	安装程序包。
install-ci-test	从零开始安装一个项目并运行测试。别名：`npm cit`。此命令运行 `npm ci`，然后立即运行 `npm test`。
install-test	安装程序包并运行测试。运行 `npm install`，然后立即运行 `npm test`。
outdated	检查已配置的注册表，确定是否有任何已安装的程序包已过时。
ping	ping 已配置或给定的 npm 注册表并验证身份验证。
publish	将程序包版本发布到注册表。

命令	说明
update	猜测程序包存储库 URL 的位置，然后尝试使用 `--browser` 配置参数来打开它。
view	显示程序包元数据。也可用于输出元数据属性。

支持的客户端命令

这些命令不需要与软件包存储库进行任何直接交互，因此 CodeCatalyst不需要任何东西来支持它们。

命令	说明
bin（旧版）	显示 npm `bin` 目录。
build	构建程序包。
cache	操作程序包缓存。
completion	在所有 npm 命令中启用制表符自动完成功能。
config	更新用户和全局 `npmrc` 文件的内容。
dedupe	搜索本地程序包树，并尝试通过将依赖项进一步向上移动来简化结构，这样多个依赖程序包就可以更有效地共享依赖项。
edit	编辑已安装的程序包。在当前工作目录中选择一个依赖项，然后在默认编辑器中打开程序包目录。
explore	浏览已安装的程序包。在指定的已安装程序包目录中创建一个子 Shell。如果指定了一条命令，则此命令将在该子 Shell 中运行，然后立即停止运行。
help	获取有关 npm 的帮助。
help-search	搜索 npm 帮助文档。

命令	说明
init	创建 package.json 文件。
link	创建指向程序包目录的符号链接。
ls	列出已安装的程序包。
pack	将程序包打包成 tarball。
prefix	显示前缀。除非同时也指定了 -g，否则这是包含 package.json 文件的最近一级父目录。
prune	删除未在父程序包依赖项列表中列出的程序包。
rebuild	对匹配的文件夹运行 npm build 命令。
restart	运行程序包的停止、重启和启动脚本以及相关的前置和后置脚本。
root	将有效的 node_modules 目录输出到标准输出。
run-script	运行任意程序包脚本。
shrinkwrap	锁定依赖项版本以供发布。
uninstall	卸载程序包。

不受支持的命令

CodeCatalyst 软件包存储库不支持这些npm命令。

命令	说明	备注
access	设置已发布程序包的访问级别。	CodeCatalyst 使用的权限模型不同于公共 npmjs 存储库。
adduser	添加注册表用户账户	CodeCatalyst 使用的用户模型不同于公共 npmjs 存储库。

命令	说明	备注
audit	运行安全审核。	CodeCatalyst 目前不提供安全漏洞数据。
hook	管理 npm 钩子，包括添加、移除、列出和更新。	CodeCatalyst 目前不支持任何变更通知机制。
login	对用户进行身份验证。这是 `npm adduser` 的一个别名。	CodeCatalyst 使用的身份验证模型不同于公共 npmjs 存储库。有关信息，请参阅使用以下命令配置 npm CodeCatalyst。
logout	注销注册表。	CodeCatalyst 使用的身份验证模型不同于公共 npmjs 存储库。无法从 CodeCatalyst 存储库中注销，但是身份验证令牌将在其可配置的到期时间后过期。默认令牌持续时间为 12 小时。
owner	管理程序包所有者。	CodeCatalyst 使用的权限模型不同于公共 npmjs 存储库。
profile	更改注册表配置文件的设置。	CodeCatalyst 使用的用户模型不同于公共 npmjs 存储库。
search	在注册表中搜索与搜索词匹配的程序包。	CodeCatalyst 不支持该`search`命令。
star	标记您喜欢的程序包。	CodeCatalyst 目前不支持任何收藏夹机制。
stars	查看已标记为收藏的程序包。	CodeCatalyst 目前不支持任何收藏夹机制。

命令	说明	备注
team	管理团队和团队成员资格。	CodeCatalyst 使用的用户和组成员资格模型不同于公共 npmjs 存储库。
token	管理您的身份验证令牌。	CodeCatalyst 使用不同的模型来获取身份验证令牌。有关信息，请参阅使用以下命令配置 npm CodeCatalyst。
unpublish	从注册表中移除程序包。	CodeCatalyst 不支持使用 npm 客户端从存储库中删除软件包版本。您可以在控制台中删除程序包。
whoami	显示 npm 用户名。	CodeCatalyst 使用的用户模型不同于公共 npmjs 存储库。

npm 标签处理

npm 注册表支持标签，这些标签是程序包版本的字符串别名。您可以使用标签来提供别名而不是使用版本号。例如，您有一个包含多个开发流的项目，并且为每个流使用不同的标签（例如 `stable`、`beta`、`dev`、`canary`）。有关更多信息，请参阅 npm Docs 上的 dist-tag。

默认情况下，npm 使用 `latest` 标签来标识程序包的当前版本。`npm install pkg`（不带 `@version` 或 `@tag` 说明符）会安装最新的标签。通常，项目仅对稳定发行版使用最新标签。对于不稳定版本或预发行版本使用其他标签。

使用 npm 客户端编辑标签

这三个`npm dist-tag`命令（`add`、`rm`、和`ls`）在 CodeCatalyst 软件包存储库中的功能与它们在默认 npm 注册表中的功能相同。

npm 标签和上游存储库

当npm请求某个软件包的标签以及该软件包的版本也存在于上游存储库中时，会先 CodeCatalyst 合并这些标签，然后再将其返回给客户端。例如，名为 `R` 的存储库有一个名为 `U` 的上游存储库。下表显示了两个存储库中都存在的名为 `web-helper` 的程序包的标签。

Repository	程序包名称	程序包标签
R	web-helper	最新（版本 1.0.0 的别名）
U	web-helper	alpha（版本 1.0.1 的别名）

在这种情况下，当 npm 客户端从存储库 R 获取 web-helper 程序包的标签时，它会同时收到最新标签和 alpha 标签。标签指向的版本不会改变。

如果上游和本地存储库中的同一个软件包上都存在相同的标签，则 CodeCatalyst 使用上次更新的标签。例如，假设 webhelper 上的标签已修改为如下所示。

Repository	程序包名称	程序包标签	上次更新时间
R	web-helper	最新（版本 1.0.0 的别名）	2023 年 1 月 1 日
U	web-helper	最新（版本 1.0.1 的别名）	2023 年 6 月 1 日

在这种情况下，当 npm 客户端从存储库 R 提取程序包 web-helper 的标签时，最新标签将作为版本 1.0.1 的别名，因为它上次已进行更新。这样就可以运行 npm update，在上游存储库中更轻松地使用尚不存在于本地存储库中的新程序包版本。

使用 Maven

许多不同的语言（包括 Java、Kotlin、Scala 和 Clojure）都使用 Maven 存储库格式。许多不同的构建工具（包括 Maven、Gradle、Scala SBT、Apache Ivy 和 Leiningen）都支持 CodeCatalyst。

我们已经测试并确认了与以下版本 CodeCatalyst 的兼容性：

- 最新的 Maven 版本：3.6.3。
- 最新的 Gradle 版本：6.4.1。我们还测试了 5.5.1 版本。

主题

- 配置并使用 Gradle Groovy

- 配置并使用 mvn
- 使用 curl 发布程序包
- 使用 Maven 校验和与快照

配置并使用 Gradle Groovy

要将 Gradle Groovy 与配合使用 CodeCatalyst，您必须将 Gradle Groovy 连接到您的软件包存储库，并提供用于身份验证的个人访问令牌 (PAT)。你可以在控制台中查看将 Gradle Groovy 连接到软件包存储库的说明。 CodeCatalyst

目录

从中获取依赖关系 CodeCatalyst

以下说明说明了如何配置 Gradle Groovy 以获取软件包存储 CodeCatalyst 库的依赖项。

使用 Gradle Groovy 从软件包存储库中获取依赖项 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。
3. 在导航窗格中，选择程序包。
4. 从程序包存储库列表中，选择您的程序包存储库。
5. 选择连接到存储库。
6. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 Gradle Groovy。
7. 你需要一个个人访问令牌 (PAT) 来对 Gradle Groovy 进行身份验证。 CodeCatalyst如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

a. 选择创建令牌。

b. 选择复制以复制您的 PAT。

> **Warning**
>
> 关闭此对话框后，您将无法再次查看或复制您的 PAT。

8. 使用您的访问凭证更新您的 gradle 属性文件。*username*替换为您的 CodeCatalyst 用户名，然后*PAT*用您的 CodeCatalyst 个人访问令牌替换。只要在以下步骤中使用相同的值，就可以为*spaceUsername*和*spacePassword*使用任何值。

```
spaceUsername=username
spacePassword=PAT
```

9. 要从 Gradle 版本 CodeCatalyst 中获取依赖关系，请复制maven代码片段并将其添加到项目文件的相应repositories部分。build.gradle替换以下值。*spaceName*只要在以下步骤中使用相同的值，就可以使用任何值。

> **Note**
>
> 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *space_name*替换为您的 CodeCatalyst空间名称。
- *proj_name*用您的 CodeCatalyst项目名称替换。
- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。

```
maven {
  name = 'spaceName'
  url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
  credentials(PasswordCredentials)
}
```

10. （可选）要将 CodeCatalyst 包存储库用作项目依赖关系的唯一来源，请从build.gradle文件中删除存储库中的任何其他部分。如果您有多个存储库，Gradle 会按照列出的顺序在每个存储库中搜索依赖项。

从中获取插件 CodeCatalyst

默认情况下，Gradle 会解析来自公有 Gradle 插件门户的插件。以下步骤将您的 Gradle 项目配置为解析 CodeCatalyst 软件包存储库中的插件。

使用 Gradle 从 CodeCatalyst 软件包存储库中获取插件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目。
3. 在导航窗格中，选择程序包。
4. 从程序包存储库列表中，选择您的程序包存储库。
5. 选择连接到存储库。
6. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 Gradle。
7. 您需要使用个人访问令牌 (PAT) 来对 Gradle 进行 CodeCatalyst身份验证。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。
   a. 选择创建令牌。
   b. 选择复制以复制您的 PAT。

   > **Warning**
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

8. 使用您的访问凭证更新您的 gradle 属性文件。*username*替换为您的 CodeCatalyst 用户名，然后*PAT*用您的 CodeCatalyst 个人访问令牌替换。只要在以下步骤中使用相同的值，就可以为*spaceUsername*和*spacePassword*使用任何值。

   ```
   spaceUsername=username
   spacePassword=PAT
   ```

9. 在 `settings.gradle` 文件中添加一个 `pluginManagement` 块。`pluginManagement` 块必须出现在 `settings.gradle` 中的任何其他语句之前。替换以下值。

   > **Note**
   > 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *spaceName*替换为上一步中使用的名称值。
- *space_name*替换为您的 CodeCatalyst空间名称。
- *proj_name*用您的 CodeCatalyst项目名称替换。
- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。

```
pluginManagement {
    repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

这将确保 Gradle 会解析来自指定存储库的插件。存储库必须已配置与 Gradle 插件门户（`gradle-plugins-store`）的上游连接，以便构建可以使用通常所需的 Gradle 插件。有关更多信息，请参阅 Gradle 文档。

通过以下方式从外部软件包存储库获取软件包 CodeCatalyst

您可以通过存储库从公共存储库安装 Maven 软件包，方法是将其配置为与代表网关存储库的网关的上游连接。 CodeCatalyst从网关存储库安装的软件包会被提取并存储在您的存储 CodeCatalyst库中。

CodeCatalyst 支持以下公共 Maven 软件包存储库。

- maven-central-gateway
- google-android-gateway
- gradle-plugins-gateway
- commonsware-gateway

从公共 Maven 程序包存储库安装程序包

1. 如果您还没有，请按照从中获取依赖关系 CodeCatalyst或从中获取插件 CodeCatalyst中的步骤使用您的 CodeCatalyst 软件包存储库配置 Gradle。
2. 确保您的存储库已将要从中安装的网关存储库添加为上游连接。为此，您可以按照添加上游存储库中的说明操作，并选择要作为上游添加的公共程序包存储库。

有关从上游存储库请求程序包的更多信息，请参阅请求包含上游存储库的程序包版本。

将包发布到 CodeCatalyst

本节介绍如何将使用 Gradle Groovy 构建的 Java 库发布到存储库。 CodeCatalyst

使用 Gradle Groovy 将软件包发布到软件包存储库 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。
3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 Gradle Groovy。
6. 您需要使用个人访问令牌 (PAT) 来对 Gradle 进行 CodeCatalyst身份验证。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

   a. 选择创建令牌。

   b. 选择复制以复制您的 PAT。

   > **Warning**
   >
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 使用您的访问凭证更新您的 gradle 属性文件。*username*替换为您的 CodeCatalyst 用户名，然后*PAT*用您的 CodeCatalyst 个人访问令牌替换。只要在以下步骤中使用相同的值，就可以为*spaceUsername*和*spacePassword*使用任何值。

   ```
   spaceUsername=username
   spacePassword=PAT
   ```

8. 将 `maven-publish` 插件添加到项目的 `build.gradle` 文件的 `plugins` 部分。

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

9. 接下来，在项目 build.gradle 文件中添加一个 publishing 部分。替换以下值。

> **Note**
>
> 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *space_name*替换为您的 CodeCatalyst空间名称。
- *proj_name*用您的 CodeCatalyst项目名称替换。
- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。

```
publishing {
    publications {
        mavenJava(MavenPublication) {
            groupId = 'group-id'
            artifactId = 'artifact-id'
            version = 'version'
            from components.java
        }
    }
    repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

maven-publish 插件根据 publishing 部分中指定的 groupId、artifactId 和 version 生成 POM 文件。

10. 对 build.gradle 作出的这些更改完成后，运行以下命令来构建项目并将其上传到存储库。

```
./gradlew publish
```

11. 在 CodeCatalyst 控制台中导航到您的软件包存储库，以检查软件包是否已成功发布。您应在程序包存储库的程序包列表中看到该程序包。

有关更多信息，请参阅 Gradle 网站上的以下主题：

- 构建 Java 库
- 将项目作为模块发布

在 IntelliJ IDEA 中运行 Gradle 构建

你可以在 IntelliJ IDEA 中运行 Gradle 构建，从中提取依赖关系。 CodeCatalyst要对 Gradle 进行身份验证 CodeCatalyst，必须使用个人访问令牌 (PAT)。您可以将 CodeCatalyst PAT 存储在自己选择的单独文件中`gradle.properties`或单独存储。

方法 1：将 PAT 放入 **`gradle.properties`**

如果您未使用 `gradle.properties` 文件并且能够用 PAT 覆盖其内容，请使用此方法。如果您使用 `gradle.properties`，则可以修改此方法以添加 PAT，而不是覆盖文件内容。

> **Note**
>
> 该示例显示了位于 `GRADLE_USER_HOME` 中的 `gradle.properties` 文件。

首先，创建一个 PAT（如果您没有 PAT）。

创建个人访问令牌（PAT）

1. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。

> **Tip**
>
> 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

2. 在 PAT 名称中，为您的 PAT 输入描述性名称。

3. 在到期日期中，保留默认日期，或者选择日历图标以自定义日期。到期日期默认为从当前日期起一年后。
4. 选择创建。

   当为源存储库选择克隆存储库时，也可以创建此令牌。
5. 将 PAT 密钥保存在安全位置。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后将无法再检索该密钥。

接下来，使用以下代码段来更新 `build.gradle` 文件：

```
repositories {
    maven {
        name = 'spaceName'
        url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
        credentials(PasswordCredentials)
    }
}
```

方法 2：将 PAT 放入单独的文件中

如果您不想修改 `gradle.properties` 文件，请使用此方法。

首先，创建一个 PAT（如果您没有 PAT）。

创建个人访问令牌（PAT）

1. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。

> **Tip**
>
> 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

2. 在 PAT 名称中，为您的 PAT 输入描述性名称。

3. 在到期日期中，保留默认日期，或者选择日历图标以自定义日期。到期日期默认为从当前日期起一年后。
4. 选择创建。

   当为源存储库选择克隆存储库时，也可以创建此令牌。
5. 将 PAT 密钥保存在安全位置。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后将无法再检索该密钥。

将 PAT 放在单独的文件中

1. 使用以下代码段来更新 `build.gradle` 文件。将*space_name*、*proj_name*和*repo_name*，替换为您的 CodeCatalyst 用户名、空间名称、项目名称和软件包存储库名称。

```
def props = new Properties()
file("fileName").withInputStream { props.load(it) }

repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

2. 将 PAT 写入在 `build.gradle` 文件中指定的文件：

```
echo "codecatalystArtifactsToken=PAT" > fileName
```

配置并使用 mvn

您可以使用 `mvn` 命令来运行 Maven 构建。您必须将 `mvn` 配置为使用程序包存储库，并提供用于身份验证的个人访问令牌（PAT）。

目录

从中获取依赖关系 CodeCatalyst

mvn要配置为从 CodeCatalyst 存储库获取依赖项，必须编辑 Maven 配置文件`settings.xml`，也可以编辑项目的项目模型对象 (POM) 文件。POM 文件包含有关项目的信息以及 Maven 构建项目所需的配置信息，例如依赖项、构建目录、源目录、测试源目录、插件和目标。

用于**mvn**从 CodeCatalyst 软件包存储库中获取依赖项

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。
3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 mvn。
6. 您需要使用个人访问令牌 (PAT) mvn 进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

    a. 选择创建令牌。

    b. 选择复制以复制您的 PAT。

    > **Warning**
    >
    > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 将包含存储库的配置文件添加到 `settings.xml` 文件。替换以下值。

    > **Note**
    >
    > 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *space_name*替换为您的 CodeCatalyst空间名称。
- *proj_name*用您的 CodeCatalyst项目名称替换。
- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。

```
<profiles>
  <profile>
    <id>repo_name</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <repositories>
        <repository>
          <id>repo_name</id>
          <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
        </repository>
    </repositories>
  </profile>
</profiles>
```

8. 将服务器添加到 `settings.xml` 文件中的服务器列表。替换以下值。

> **Note**
>
> 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。
- *username*用您的 CodeCatalyst 用户名替换。
- *PAT*用您的 CodeCatalyst PAT 替换。

```
<servers>
  <server>
    <id>repo_name</id>
    <username>username</username>
    <password>PAT</password>
  </server>
```

```
</servers>
```

9. （可选）在 settings.xml 文件中设置镜像，以捕获所有连接并将它们路由到您的存储库，而不是网关存储库。

> **Note**
>
> 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

- *space_name*替换为您的 CodeCatalyst空间名称。
- *proj_name*用您的 CodeCatalyst项目名称替换。
- *repo_name*替换为你的 CodeCatalyst软件包存储库名称。

```
<mirrors>
  <mirror>
    <id>repo_name</id>
    <name>repo_name</name>
    <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
    <mirrorOf>*</mirrorOf>
  </mirror>
</mirrors>
```

> **Important**
>
> 可以在 <id> 元素中使用任何值，但必须与 <server> 和 <repository> 元素中的值相同。这样，就可以将指定的凭据包含在对的请求中 CodeCatalyst。

更改这些配置后，就可以构建项目了。

```
mvn compile
```

通过以下方式从外部软件包存储库获取软件包 CodeCatalyst

您可以通过存储库从公共存储库安装 Maven 软件包，方法是将其配置为与代表网关存储库的网关的上游连接。 CodeCatalyst从网关存储库安装的软件包会被提取并存储在您的存储 CodeCatalyst库中。

目前， CodeCatalyst 支持以下公共 Maven 软件包存储库。

- maven-central-gateway
- google-android-gateway
- gradle-plugins-gateway
- commonsware-gateway

从公共 Maven 程序包存储库安装程序包

1. 如果您还没有，请mvn按照中的步骤使用您的 CodeCatalyst软件包存储库进行配置从中获取依赖关系 CodeCatalyst。
2. 确保您的存储库已将要从中安装的网关存储库添加为上游连接。要查看添加了哪些上游源或要将网关存储库添加为上游源，请按照添加上游存储库中的说明操作。

有关从上游存储库请求程序包的更多信息，请参阅请求包含上游存储库的程序包版本。

将包发布到 CodeCatalyst

要将 Maven 包发布mvn到 CodeCatalyst 存储库，还必须编辑~/.m2/settings.xml和项目 POM。

用于将包发布**mvn**到您的软件 CodeCatalyst 包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。
3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 mvn。
6. 您需要使用个人访问令牌 (PAT) mvn 进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

    a. 选择创建令牌。

b. 选择复制以复制您的 PAT。

> **Warning**
>
> 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 在本地机器上使用 PAT 配置环境变量。您将在 `setting.xml` 文件中使用此环境变量。

```
export CODECATALYST_ARTIFACTS_TOKEN=your_PAT
```

8. 在 `settings.xml` 中添加一个引用 `CodeCatalyst_ARTIFACTS_TOKEN` 环境变量的 `<servers>` 部分，以便 Maven 可在 HTTP 请求中传递令牌。

```
<settings>
...
    <servers>
        <server>
            <id>repo-name</id>
            <username>username</username>
            <password>${env.CodeCatalyst_ARTIFACTS_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

9. 将 `<distributionManagement>` 部分添加到项目的 `pom.xml`。

```
<project>
...
     <distributionManagement>
         <repository>
             <id>repo_name</id>
             <name>repo_name</name>
             <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
         </repository>
     </distributionManagement>
...
</project>
```

更改这些配置后，就可以开始构建项目并将项目发布到指定的存储库。

```
mvn deploy
```

您可以在 CodeCatalyst 控制台中导航到您的软件包存储库，以检查软件包是否已成功发布。

发布第三方程序包

您可以使用将第三方 Maven 包发布到 CodeCatalyst 存储库。`mvn deploy:deploy-file`这对于想要发布程序包且只有 JAR 文件且无权访问程序包源代码或 POM 文件的用户很有用。

`mvn deploy:deploy-file` 命令会根据命令行中传递的信息生成 POM 文件。

首先，创建一个 PAT（如果您没有 PAT）。

创建个人访问令牌（PAT）

1. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

2. 在 PAT 名称中，为您的 PAT 输入描述性名称。
3. 在到期日期中，保留默认日期，或者选择日历图标以自定义日期。到期日期默认为从当前日期起一年后。
4. 选择创建。

   当为源存储库选择克隆存储库时，也可以创建此令牌。
5. 将 PAT 密钥保存在安全位置。

   > **Important**
   >
   > PAT 密钥仅显示一次。关闭窗口后将无法再检索该密钥。

发布第三方 Maven 程序包

1. 创建 `~/.m2/settings.xml` 文件并输入以下内容：

```
<settings>
    <servers>
        <server>
            <id>repo_name</id>
            <username>username</username>
            <password>PAT}</password>
        </server>
    </servers>
</settings>
```

2. 运行 mvn deploy:deploy-file 命令：

```
mvn deploy:deploy-file -DgroupId=commons-cli          \
-DartifactId=commons-cli       \
-Dversion=1.4                  \
-Dfile=./commons-cli-1.4.jar   \
-Dpackaging=jar                \
-DrepositoryId=repo-name      \
-Durl=https://packages.region.codecatalyst.aws/maven/space-name/proj-name/repo-
name/
```

> **Note**
>
> 前面的示例发布 commons-cli 1.4。修改 groupID、artifactID、version 和 file 参数来发布另一个 JAR。

这些说明基于 Apache Maven 文档中关于 JARs 将第三方部署到远程存储库的指南中的示例。

有关更多信息，请参阅 Apache Maven Project 网站上的以下主题：

- 设置多个存储库
- 设置参考
- 分配管理
- 配置文件

使用 curl 发布程序包

本节介绍如何使用 HTTP 客户端将 Maven 软件包发布curl到软件 CodeCatalyst 包存储库。如果您的环境中没有 Maven 客户端或想要安装 Maven 客户端，则使用 curl 发布程序包会很有用。

使用 **curl** 发布 Maven 程序包

1. 您必须将个人访问令牌 (PAT) 存储到环境变量中才能curl进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，则可以创建一个 PAT 并配置环境变量。

   a. 请按照使用个人访问令牌向用户授予对存储库的访问权限中的步骤操作来创建 PAT。复制 PAT 以将其存储在环境变量中。

   b. 在本地机器的命令行上，使用 PAT 配置环境变量。

   ```
   export CodeCatalyst_ARTIFACTS_TOKEN=your_PAT
   ```

2. 使用以下curl命令将 JAR 发布到 CodeCatalyst存储库。
   将*username*、*space_nameproj_name*、和*repo_name*，替换为您的 CodeCatalyst 用户名、空间名称、项目名称和软件包存储库名称。

   ```
   curl --request PUT https://packages.region.codecatalyst.aws/maven/space-name/proj-
   name/repo-name/com/mycompany/app/my-app/1.0/my-app-1.0.jar \
        --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
    application/octet-stream" \
        --data-binary @target/path/to/my-app-1.0.jar
   ```

3. 使用以下curl命令将 POM 发布到 CodeCatalyst存储库。
   将*username*、*space_nameproj_name*、和*repo_name*，替换为您的 CodeCatalyst 用户名、空间名称、项目名称和软件包存储库名称。

   ```
   curl --request PUT https://packages.region.codecatalyst.aws/maven/space-name/proj-
   name/repo-name/com/mycompany/app/my-app/1.0/my-app-1.0.pom \
        --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
    application/octet-stream" \
        --data-binary @target/my-app-1.0.pom
   ```

4. 此时，Maven 软件包将在您的 CodeCatalyst 存储库中，状态为。Unfinished为了能够使用程序包，程序包必须处于 Published 状态。您可以Published通过将maven-metadata.xml文件上传Unfinished到您的包中或在 CodeCatalyst 控制台中更改状态来将包从移动到。

a. 选项 1：使用以下 curl 命令将 maven-metadata.xml 文件添加到您的程序包中。将*username*、*space_nameproj_name*、和*repo_name*，替换为您的 CodeCatalyst 用户名、空间名称、项目名称和软件包存储库名称。

```
curl --request PUT https://packages.region.codecatalyst.aws/maven/space-
name/proj-name/repo-name/com/mycompany/app/my-app/maven-metadata.xml \
     --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
 application/octet-stream" \
     --data-binary @target/maven-metadata.xml
```

以下是 maven-metadata.xml 文件的内容示例：

```
<metadata modelVersion="1.1.0">
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <versioning>
        <latest>1.0</latest>
        <release>1.0</release>
        <versions>
            <version>1.0</version>
        </versions>
        <lastUpdated>20200731090423</lastUpdated>
    </versioning>
</metadata>
```

b. 选项 2：在 CodeCatalyst 控制台Published中将包裹状态更新为。有关如何更新程序包版本的状态的信息，请参阅更新程序包版本的状态。

如果您只有包的 JAR 文件，则可以使用将消耗包版本发布到 CodeCatalyst 存储库。mvn如果您无法访问程序包的源代码或 POM，此方法会很有用。有关详细信息，请参阅发布第三方程序包。

使用 Maven 校验和与快照

以下各节介绍如何在中使用 Maven 校验和和 Maven 快照。 CodeCatalyst

使用 Maven 校验和

将 Maven 包发布到 CodeCatalyst 包存储库时，将使用与包中每个资产或文件关联的校验和来验证上传。资产的例子包括 jar、pom 和 war 文件。对于每个资产，Maven 程序包都包含多个校验和文件，这些文件使用带有附加扩展名（例如 md5 或 sha1）的资产名称。例如，名为 my-

maven-package.jar 的文件的校验和文件可能是 my-maven-package.jar.md5 和 my-maven-package.jar.sha1。

每个 Maven 程序包还包含一个 maven-metadata.xml 文件。必须上传此文件才能成功发布。如果在上传任何程序包文件期间检测到校验和不匹配，则发布将停止。这可能会阻止上传 maven-metadata.xml。在发生此情况时，Maven 程序包的状态将设置为 Unfinished。您无法下载具有此状态的程序包中的资产。

如果在发布 Maven 程序包时出现校验和不匹配的情况，请记住以下几点：

- 如果在上传 maven-metadata.xml 之前出现校验和不匹配的情况，则程序包的状态不会设置为 Unfinished。程序包不可见，也无法使用其资产。在发生此情况时，请尝试以下方法之一，然后尝试再次下载资产。
  - 再次运行发布 Maven 程序包的命令。如果因网络问题导致校验和文件在下载期间损坏，此方法可能会起作用。如果已解决网络问题以进行重试，则校验和匹配且下载成功。
  - 如果重新发布 Maven 程序包不起作用，请删除该程序包，然后重新发布它。
- 如果在上传 maven-metadata.xml 之后出现校验和不匹配的情况，则程序包的状态会设置为 Published。您可以使用程序包中的任何资产，包括校验和不匹配的资产。下载资源时，生成的校验和会随 CodeCatalyst 之下载。如果下载的文件与校验和不匹配关联，则其下载的校验和文件可能与发布程序包时上传的校验和不匹配。

使用 Maven 快照

Maven 快照是 Maven 程序包的特殊版本，该版本引用了最新的生产分支代码。它是先于最终发布版本的开发版本。您可以通过附加到程序包版本的后缀 SNAPSHOT 来识别 Maven 程序包的快照版本。例如，版本 1.1 的快照是 1.1-SNAPSHOT。有关更多信息，请参阅 Apache Maven Project 网站上的 What is a SNAPSHOT version?。

CodeCatalyst 支持发布和使用 Maven 快照。您可以将 Maven 快照发布到 CodeCatalyst 存储库，或者（如果您直接连接）发布到上游存储库。但是，不支持程序包存储库及其某个上游存储库中的快照版本。例如，如果您将带有版本的 Maven 包上传1.2-SNAPSHOT到包存储库，则 CodeCatalyst 不支持将具有相同快照版本的 Maven 包上传到其上游存储库之一。在此场景中，可能会返回不可预测的结果。

在发布 Maven 快照时，其先前版本将保留在名为构建的新版本中。每次发布 Maven 快照时，都会创建一个新的构建版本。快照的所有先前版本都保留在其构建版本中。发布 Maven 快照时，其状态将设置为 Published，包含先前版本的构建的状态设置为 Unlisted。

如果您请求快照，则会返回状态为 `Published` 的版本。这始终是 Maven 快照的最新版本。您也可以请求快照的特定构建。

要删除 Maven 快照的所有构建版本，请使用 CodeCatalyst 控制台。

使用 NuGet

这些主题描述了如何使用和发布`NuGet`软件包 CodeCatalyst。

> **Note**
>
> CodeCatalyst 支持 4.8 及更高NuGet版本。

主题

CodeCatalyst 与视觉工作室一起使用

您可以 CodeCatalyst 直接在 Visual Studio 中使用软件包。

要配置和 NuGet 使用`dotnet`或等 CLI 工具`nuget`，请参阅配置并使用 nuget 或 dotnet CLI。

目录

使用 CodeCatalyst

Windows

使用 Visual Studio 配置 CodeCatalyst

1. 需要使用个人访问令牌 (PAT) 进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，请按照使用个人访问令牌向用户授予对存储库的访问权限中的说明创建一个 PAT。
2. 使用 `nuget` 或 `dotnet` 配置程序包存储库和凭证。

   dotnet

   Linux 和 MacOS 用户：由于在非 Windows 平台上不支持加密，因此您必须在以下命令中添加 `--store-password-in-clear-text` 标志。请注意，这会将您的密码以纯文本形式存储在配置文件中。

   ```
   dotnet nuget add source https://packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/v3/index.json --name repo_name --password PAT --username user_name
   ```

   nuget

   ```
   nuget sources add -name repo_name -Source https://packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/v3/index.json -password PAT --username user_name
   ```

   输出示例：

   ```
   Package source with Name: repo_name added successfully.
   ```
3. 将 Visual Studio 配置为使用新的程序包来源。在 Visual Studio 中，选择工具，然后选择选项。
4. 在 “选项” 菜单中，展开 “Package Man NuGet ager” 部分，然后选择 Package S ources。
5. 在 “可用包源” 列表中，确保您的*repo_name*源已启用。如果您已将包存储库配置为与 NuGet 图库的上游连接，请禁用 nuget.org 源代码。

macOS

使用 Visual Studio 配置 CodeCatalyst

1. 需要使用个人访问令牌 (PAT) 进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，请按照使用个人访问令牌向用户授予对存储库的访问权限中的说明创建一个 PAT。
2. 在菜单栏上，选择首选项。
3. 在该NuGet部分中，选择来源。
4. 选择添加并添加您的存储库信息。
   a. 在 “名称” 中，输入您的 CodeCatalyst 软件包存储库名称。
   b. 在 “位置” 中，输入您的 CodeCatalyst 包存储库端点。下面的代码片段显示了示例端点。将*space-name*、*proj-name*和*repo-name*，替换为您的 CodeCatalyst 空间名称、项目名称和存储库名称。

      ```
      https://packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/
      ```

   c. 对于用户名，输入任何有效值。
   d. 对于密码，输入您的 PAT。
5. 选择 添加源。
6. 如果您已将包存储库配置为与 NuGet图库的上游连接，请禁用 nuget.org 源代码。

配置完成后，Visual Studio 可以使用存储 CodeCatalyst 库、其任何上游存储库或 NuGet.org 中的软件包（如果您将其配置为上游源）。有关在 Visual Studio 中浏览和安装 NuGet 包的更多信息，请参见NuGet 文档中的使用 Package Manager 在 Visual Studio 中安装和管理 NuGet 软件包。

配置并使用 nuget 或 dotnet CLI

您可以使用NuGet和之类的 CLI 工具dotnet来发布和使用来自的包 CodeCatalyst。本文档提供有关配置 CLI 工具以及使用这些工具来发布或使用程序包的信息。

目录

- 将 NuGet 包发布到 CodeCatalyst

使用配置 NuGet CodeCatalyst

要 NuGet 使用进行配置 CodeCatalyst，请在 NuGet 配置文件中添加存储库端点和个人访问令牌，以允许`nuget`或`dotnet`连接到您的 CodeCatalyst 软件包存储库。

NuGet 使用您的 CodeCatalyst 软件包存储库进行配置

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。
3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在 “连接到存储库” 对话框中，从包管理器客户端列表中选择NuGet或 dotnet。
6. 您需要使用个人访问令牌 (PAT) NuGet 进行身份验证 CodeCatalyst。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

   a. 选择创建令牌。

   b. 选择复制以复制您的 PAT。

   > **Warning**
   >
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 配置`nuget`或`dotnet`以使用存储库的 NuGet 终端节点和 CodeCatalyst PAT。替换以下值。

   > **Note**
   >
   > 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

   - *`username`*用您的 CodeCatalyst 用户名替换。
   - *`PAT`*用您的 CodeCatalyst PAT 替换。
   - *`space_name`*替换为您的 CodeCatalyst 空间名称。
   - *`proj_name`*用您的 CodeCatalyst 项目名称替换。
   - *`repo_name`*替换为你的 CodeCatalyst 软件包存储库名称。

a. 对于 nuget，请使用 nuget sources add 命令。

```
nuget sources add -name "repo_name" -Source "https://
packages.region.codecatalyst.aws/nuget/space_name/proj_name/repo_name/v3/
index.json" -username "username" -password "PAT"
```

b. 对于 dotnet，请使用 dotnet nuget add source 命令。

Linux 和 macOS 用户：由于在非 Windows 平台上不支持加密，因此您必须在以下命令中添加 --store-password-in-clear-text 标志。请注意，这会将您的密码以纯文本形式存储在配置文件中。

```
dotnet nuget add source "https://packages.region.codecatalyst.aws/
nuget/space_name/proj_name/repo_name/v3/index.json" -n "proj_name/repo_name" -u
 "username" -p "PAT" --store-password-in-clear-text
```

配置 NuGet 完毕后 CodeCatalyst，您可以使用存储在存储 CodeCatalyst 库或其中一个上游存储库中的软件 NuGet 包，并将 NuGet 包发布到您的 CodeCatalyst 存储库。

使用 CodeCatalyst存储库中的 NuGet 软件包

配置 NuGet 完毕后 CodeCatalyst，即可使用存储在存储 CodeCatalyst 库或其上游存储库中的 NuGet 软件包。

要使用 nuget 或 dotnet 使用存储库或其上游存储库中的软件包版本，请运行以下命令。 CodeCatalyst *packageName*替换为要使用的软件包的*packageSourceName*名称以及 NuGet 配置文件中 CodeCatalyst 软件包存储库的源名称，后者应该是存储库名称。

使用 **dotnet** 安装程序包

```
dotnet add packageName --source packageSourceName
```

使用 **nuget** 安装程序包

```
nuget install packageName --source packageSourceName
```

有关更多信息，请参阅 Microsoft 文档中的使用 nuget CLI 管理程序包或使用 dotnet CLI 安装和管理程序包。

使用来自 NuGet .org 的 NuGet 软件包 CodeCatalyst

您可以通过 CodeCatalyst 存储库使用 NuGet.org 中的 NuGet 软件包，方法是将存储库配置为与 NuGet. org 的上游连接。从 NuGet.org 使用的软件包会被提取并存储在您的 CodeCatalyst 存储库中。

使用 NuGet .org 中的软件包

1. 如果还没有，请 NuGet 按照中的步骤使用 CodeCatalyst 软件包存储库配置软件包管理器使用配置 NuGet CodeCatalyst。
2. 确保您的存储库已将 NuGet.org 添加为上游连接。您可以按照中的添加上游存储库说明并选择存储库来检查添加了哪些上游源代码或将 Nuget.org 添加为上NuGet 游源。

将 NuGet 包发布到 CodeCatalyst

配置完毕后 NuGet CodeCatalyst，就可以使用`nuget`或`dotnet`将包版本发布到 CodeCatalyst 存储库。

要将软件包版本推送到 CodeCatalyst 存储库，请运行以下命令，并在 NuGet 配置`.nupkg`文件中包含文件的完整路径和 CodeCatalyst 存储库的源名称。

使用 **`dotnet`** 发布程序包

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

使用 **`nuget`** 发布程序包

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

NuGet 软件包名称、版本和资产名称标准化

CodeCatalyst 在存储软件包和资源名称以及软件包版本之前对其进行标准化，这意味着中的名称或版本 CodeCatalyst 可能与发布软件包或资源时提供的名称或版本不同。

P@@ ackage 名称 CodeCatalyst 标准化：通过将所有字母转换为小写来标准化 NuGet 软件包名称。

Package 版本 CodeCatalyst 标准化：使用与相同的 NuGet模式对 NuGet 软件包版本进行标准化。以下信息来自 NuGet 文档中的标准化版本号。

- 从版本号中删除前导零：

  - `1.00` 视为 `1.0`
  - `1.01.1` 视为 `1.1.1`
  - `1.00.0.1` 视为 `1.0.0.1`
- 版本号第四部分中的零会省略掉：
  - `1.0.0.0` 视为 `1.0.0`
  - `1.0.01.0` 视为 `1.0.1`
- SemVer 2.0.0 版本元数据已删除：
  - `1.0.7+r3456` 视为 `1.0.7`

P@@ ackage as CodeCatalyst set 名称标准化：根据标准化的 NuGet 软件包名称和软件包版本构造软件包资产名称。

NuGet 兼容性

本指南包含有关与不同 NuGet 工具和版本 CodeCatalyst的兼容性的信息。

主题

一般 NuGet 兼容性

CodeCatalyst 支持 NuGet 4.8 及更高版本。

CodeCatalyst 仅支持 NuGet HTTP 协议的 V3 版本。这意味着不支持某些依赖协议 V2 版本的 CLI 命令。有关更多信息，请参阅下文的 nuget 命令支持部分。

CodeCatalyst 不支持 PowerShellGet 2.x。

NuGet 命令行支持

CodeCatalyst 支持 NuGet (`nuget`) 和.NET 核心 (`dotnet`) CLI 工具。

nuget 命令支持

由于 CodeCatalyst 仅支持 V3 NuGet 的 HTTP 协议，因此以下命令在对 CodeCatalyst 资源使用时将不起作用：

- `list`：`nuget list` 命令显示来自给定来源的程序包列表。要获取软件包存储库中的软件 CodeCatalyst 包列表，请在 CodeCatalyst 控制台中导航到该存储库。

使用 Python

这些主题描述了如何使用 `pip` Python 包管理器以及 `twine` Python 包发布实用工具 CodeCatalyst。

主题

- 配置 pip 并安装 Python 程序包
- 配置 Twine 并发布 Python 程序包
- Python 程序包名称规范化
- Python 兼容性

配置 pip 并安装 Python 程序包

要`pip`与一起使用 CodeCatalyst，您必须`pip`连接到您的包存储库并提供用于身份验证的个人访问令牌。您可以在 CodeCatalyst 控制台中查看有关`pip`连接到软件包存储库的说明。在您进行身份验证并`pip`连接到之后 CodeCatalyst，您可以运行`pip`命令。

目录

CodeCatalyst 使用 pip 从中安装 Python 软件包

以下说明说明了`pip`如何配置为从您的 CodeCatalyst 包存储库或其上游存储库之一安装 Python 包。

配置并使用**`pip`**从软件包存储库安装 Python CodeCatalyst 软件包

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。

3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 pip。
6. 您需要一个个人访问令牌 (PAT) 来对 pip 进行 CodeCatalyst身份验证。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

   a. 选择创建令牌。

   b. 选择复制以复制您的 PAT。

   > **Warning**
   >
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 使用`pip config`命令设置 CodeCatalyst 注册表 URL 和凭据。替换以下值。

   > **Note**
   >
   > 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

   - *`username`*用您的 CodeCatalyst 用户名替换。
   - *`PAT`*用您的 CodeCatalyst PAT 替换。
   - *`space_name`*替换为您的 CodeCatalyst 空间名称。
   - *`proj_name`*用您的 CodeCatalyst 项目名称替换。
   - *`repo_name`*替换为你的 CodeCatalyst 软件包存储库名称。

   ```
   pip config set global.index-url https://username:PAT@https://
   packages.region.codecatalyst.aws/pypi/space_name/proj_name/repo_name/simple/
   ```

8. 假设您的存储库或其中一个上游存储库中存在程序包，则可以使用 `pip install` 来安装。例如，使用以下命令来安装 `requests` 程序包。

   ```
   pip install requests
   ```

   使用该`-i`选项可暂时恢复为从 https://pypi.org 安装软件包，而不是从软件包存储库安装 CodeCatalyst 软件包。

```
pip install -i https://pypi.org/simple requests
```

从 PyPI 开始使用 Python 软件包 CodeCatalyst

通过为仓库配置与 PyPI 的上游连接，你可以通过 CodeCatalyst 存储库使用 Python 包索引 (PyPI) 中的 Python 包。从 PyPI 消耗的包会被摄取并存储在你的仓库中。 CodeCatalyst

使用来自 PyPI 的程序包

1. 如果您还没有，请按照中的步骤在 CodeCatalyst 软件包存储库中 CodeCatalyst 使用 pip 从中安装 Python 软件包配置 pip。
2. 确保您的存储库已将 PyPI 添加为上游来源。您可以按照添加上游存储库中的说明操作并选择 PyPI 存储存储库来检查添加了哪些上游来源或将 PyPI 添加为上游来源。

有关从上游存储库请求程序包的更多信息，请参阅请求包含上游存储库的程序包版本。

pip 命令支持

以下各节总结了 CodeCatalyst 存储库支持的 pip 命令以及不支持的特定命令。

主题

- 与存储库进行交互的受支持命令
- 支持的客户端命令

与存储库进行交互的受支持命令

此部分列出了 `pip` 命令，其中 `pip` 客户端向其配置的注册表发出一个或多个请求。这些命令已经过验证，在针对 CodeCatalyst 软件包存储库调用时可以正常运行。

命令	说明
install	安装程序包。
download	下载程序包。

CodeCatalyst 不实现`pip search`。如果您配置`pip`了 CodeCatalyst 包存储库，则运行`pip search`将搜索并显示来自 PyP I 的包。

支持的客户端命令

这些命令不需要与存储库进行任何直接交互，因此 CodeCatalyst 无需执行任何操作即可支持它们。

命令	说明
uninstall	卸载程序包。
freeze	按要求格式输出已安装的程序包。
list	列出已安装程序包。
show	显示有关已安装程序包的信息。
check	验证已安装的程序包是否具有兼容的依赖项。
config	管理本地和全局配置。
wheel	根据您的要求构建 wheel。
hash	计算程序包存档的哈希值。
completion	协助完成命令。
debug	显示对调试有用的信息。
help	显示命令的帮助。

配置 Twine 并发布 Python 程序包

要`twine`与一起使用 CodeCatalyst，您必须`twine`连接到您的包存储库并提供用于身份验证的个人访问令牌。您可以在 CodeCatalyst 控制台中查看有关`twine`连接到软件包存储库的说明。在您进行身份验证并`twine`连接到之后 CodeCatalyst，您可以运行`twine`命令。

CodeCatalyst 使用 Twine 将软件包发布到

以下说明说明了如何进行身份验证并`twine`连接到您的 CodeCatalyst 软件包存储库。

配置并使用**twine**将包发布到您的软件 CodeCatalyst 包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在项目的概述页面上，选择程序包。
3. 从程序包存储库列表中，选择您的程序包存储库。
4. 选择连接到存储库。
5. 在连接到存储库对话框中，从程序包管理器客户端列表中选择 Twine。
6. 您将需要一个个人访问令牌 (PAT) 来对 twine 进行 CodeCatalyst身份验证。如果您已有一个 PAT，则可以使用它。如果没有 PAT，您可以在此处创建一个。

   a. 选择创建令牌。

   b. 选择复制以复制您的 PAT。

   > **Warning**
   >
   > 关闭此对话框后，您将无法再次查看或复制您的 PAT。

7. 您可以使用 `.pypirc` 文件或环境变量配置 twine。

   a. 使用 **`.pypirc`** 文件进行配置。

   在选定编辑器中打开 `~/.pypirc`。

   为其添加索引服务器 CodeCatalyst，包括您在上一步中创建和复制的存储库、用户名和 PAT。替换以下值。

   > **Note**
   >
   > 如果通过控制台指令进行复制，则以下值将进行更新且不应更改。

   - *`username`*用您的 CodeCatalyst 用户名替换。
   - *`PAT`*用您的 CodeCatalyst PAT 替换。
   - *`space_name`*替换为您的 CodeCatalyst 空间名称。
   - *`proj_name`*用您的 CodeCatalyst 项目名称替换。
   - *`repo_name`*替换为你的 CodeCatalyst 软件包存储库名称。

```
[distutils]
index-servers = proj-name/repo-name

[proj-name/repo-name]
repository = https://packages.region.codecatalyst.aws/
pypi/space_name/proj_name/repo_name/
password = PAT
username = username
```

b. 使用环境变量进行配置。

设置以下环境变量。在TWINE_REPOSITORY_URL值中，*repo_name*使用您的 CodeCatalyst 空间*space_name**proj_name*、项目和包存储库名称更新、和。

```
export TWINE_USERNAME=username
```

```
export TWINE_PASSWORD=PAT
```

```
export TWINE_REPOSITORY_URL="https://packages.region.codecatalyst.aws/
pypi/space_name/proj_name/repo_name/"
```

8. 使用 `twine upload` 命令发布 Python 发行版。

Python 程序包名称规范化

CodeCatalyst 在存储软件包名称之前对其进行标准化，这意味着中的软件包名称 CodeCatalyst 可能与发布软件包时提供的名称不同。

对于 Python 程序包，在执行规范化时，程序包名称转为小写，所有 `.`、`-` 和 `_` 字符都替换为单个 `-` 字符。因此，程序包名称 `pigeon_cli` 和 `pigeon.cli` 会规范化并存储为 `pigeon-cli`。pip 和 twine 可以使用未规范化的名称。有关 Python 程序包名称规范化的更多信息，请参阅 Python 文档中的 PEP 503。

Python 兼容性

虽然 CodeCatalyst 不支持 `/simple/` API，但它确实支持 `Legacy` API 操作。 CodeCatalyst 不支持 PyPI `XML-RPC` 或 `JSON` API 操作。

有关更多信息，请参阅 Python 打包管理机构 GitHub存储库中的以下内容。

- Legacy API
- XML-RPC API
- JSON API

程序包配额

下表描述了 Amazon 中包裹的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

资源	默认配额
程序包存储库	每个空间最多 1000 个。
直接上游存储库	每个程序包存储库最多 10 个。
已搜索的上游程序包存储库	每个请求的程序包版本最多 25 个已搜索的上游存储库。
程序包资产文件大小	每个程序包资产最多 5GB。
程序包资产	每个程序包版本最多 150 个。

使用工作流进行构建、测试和部署

在CodeCatalyst开发环境中编写应用程序代码并将其推送到CodeCatalyst 源存储库后，就可以部署它了。可以通过工作流自动执行此操作。

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

关于工作流定义文件

工作流定义文件是描述您的工作流的 YAML 文件。默认情况下，该文件存储在源存储库根目录下的 `~/.codecatalyst/workflows/` 文件夹中。该文件的扩展名可以为 .yml 或 .yaml，并且扩展名必须小写。

下面是一个简单的工作流定义文件示例。我们将在下表中逐行解释该示例。

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest .
```

行	说明
`Name: MyWorkflow`	指定工作流的名称。有关 Name 属性的更多信息，请参阅顶级属性。
`SchemaVersion: 1.0`	指定工作流架构版本。有关 SchemaVersion 属性的更多信息，请参阅顶级属性。
`RunMode: QUEUED`	表示如何 CodeCatalyst 处理多次运行。有关运行模式的更多信息，请参阅配置运行的排队行为。
`Triggers:`	指定将导致工作流运行启动的逻辑。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

`- Type: PUSH`
`  Branches:`
`  - main`
	指示只要向默认源存储库的 main 分支推送代码，工作流就必须启动。有关工作流源的更多信息，请参阅将源存储库连接到工作流。

`Actions:`	定义工作流运行期间要执行的任务。在此示例中，Actions 部分定义一个名为 Build 的操作。有关操作的更多信息，请参阅配置工作流操作。
`Build:`	定义 Build 操作的属性。有关构建操作的更多信息，请参阅使用工作流进行构建。
`Identifier: aws/build@v1`	为构建操作指定唯一的硬编码标识符。

`Inputs:`
`  Sources:`
`  - WorkflowSource`
	指示构建操作应在 WorkflowSource 源存储库中查找完成处理所需的文件。有关更多信息，请参阅 将源存储库连接到工作流。

`Configuration:`	包含特定于构建操作的配置属性。

行	说明

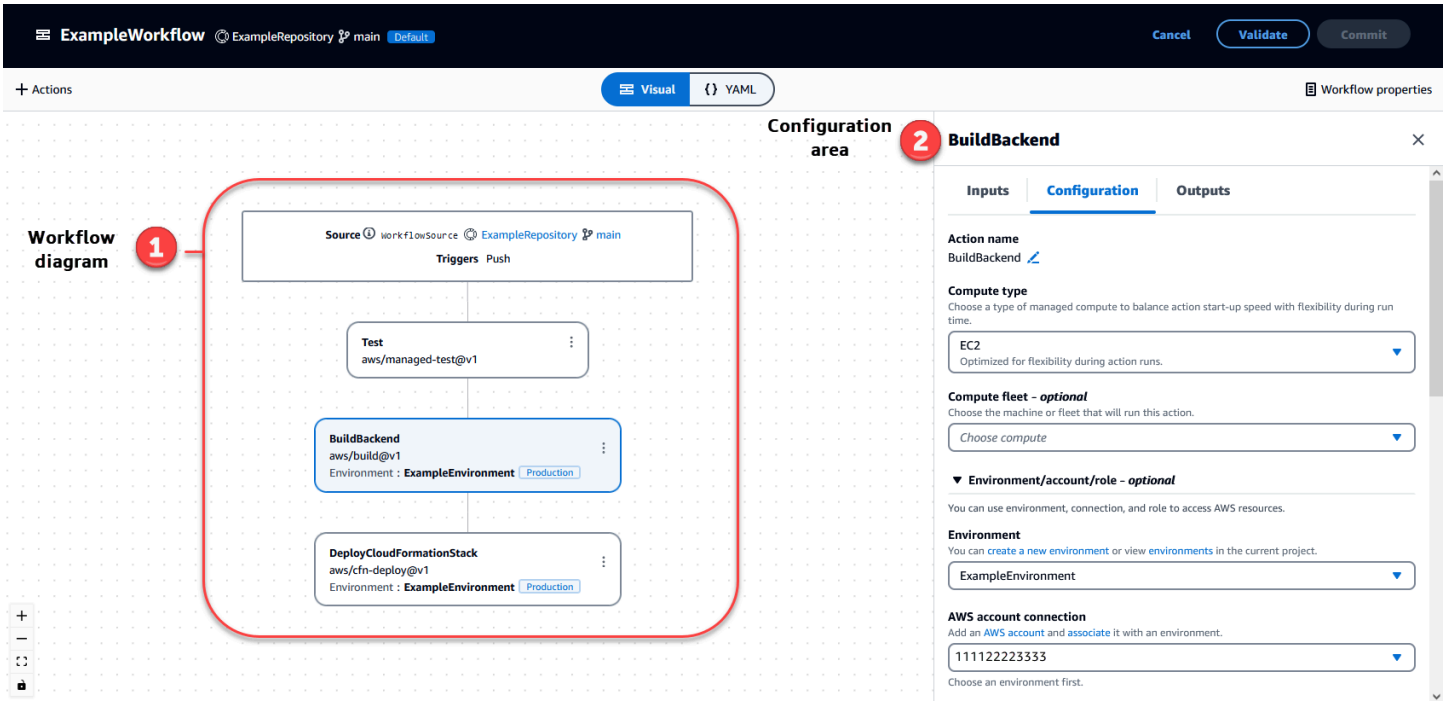
`Steps:`
`  - Run: docker build -t MyApp:lat`
`est .`
	告诉构建操作构建名为 MyApp 的 Docker 映像并使用 `latest` 标记。

有关工作流定义文件中所有可用属性的完整列表，请参阅工作流 YAML 定义。

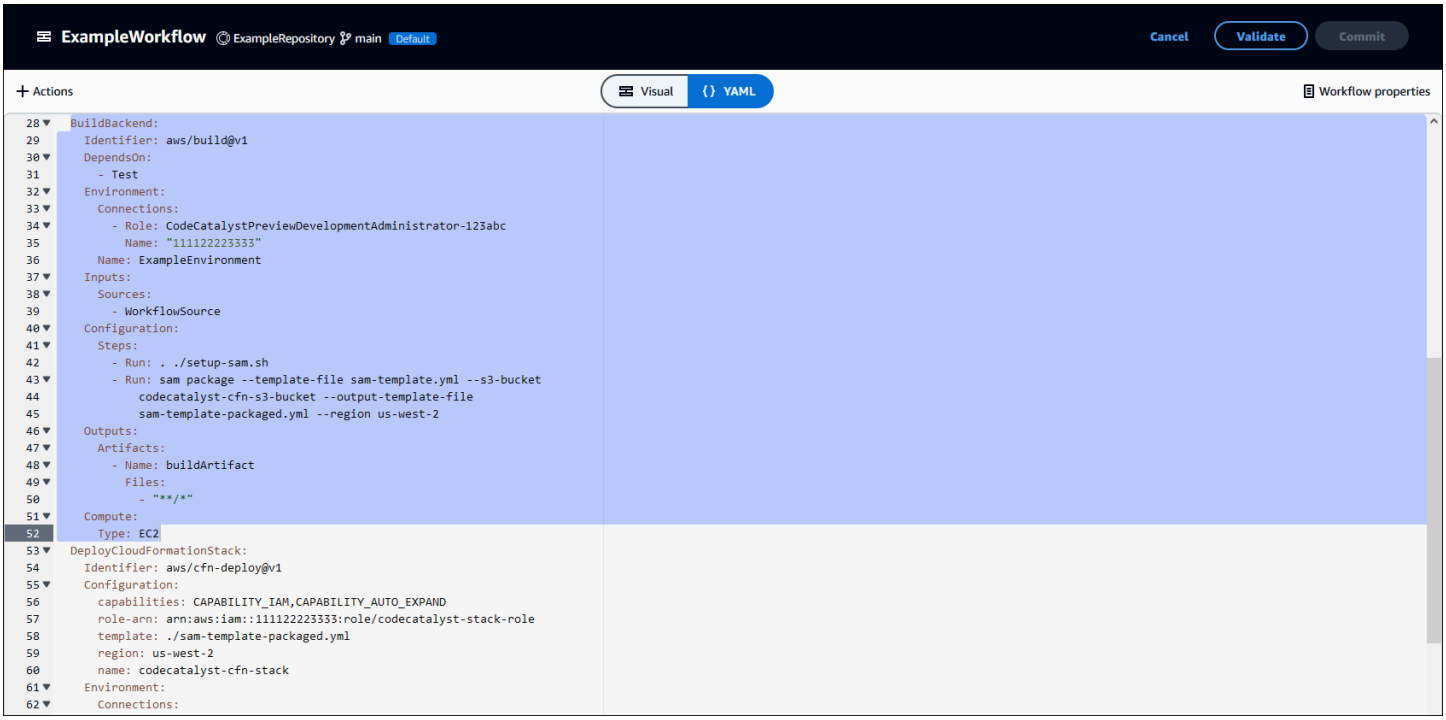
使用 CodeCatalyst 控制台的视觉编辑器和 YAML 编辑器

要创建和编辑工作流程定义文件，您可以使用首选编辑器，但我们建议使用 CodeCatalyst 控制台的可视化编辑器或 YAML 编辑器。这些编辑器提供有用的文件验证功能，有助于确保 YAML 属性名称、值、嵌套、间距、大小写等正确无误。

下图显示了可视化编辑器中的工作流。可视化编辑器为您提供了一个完整的用户界面，您可以通过该界面创建和配置工作流定义文件。可视化编辑器包括一个显示工作流主要组件的工作流程图（1）和一个配置区域（2）。



您也可以使用 YAML 编辑器，如下图所示。使用 YAML 编辑器粘贴大型代码块（例如教程中的代码块），或添加可视化编辑器不提供的高级属性。

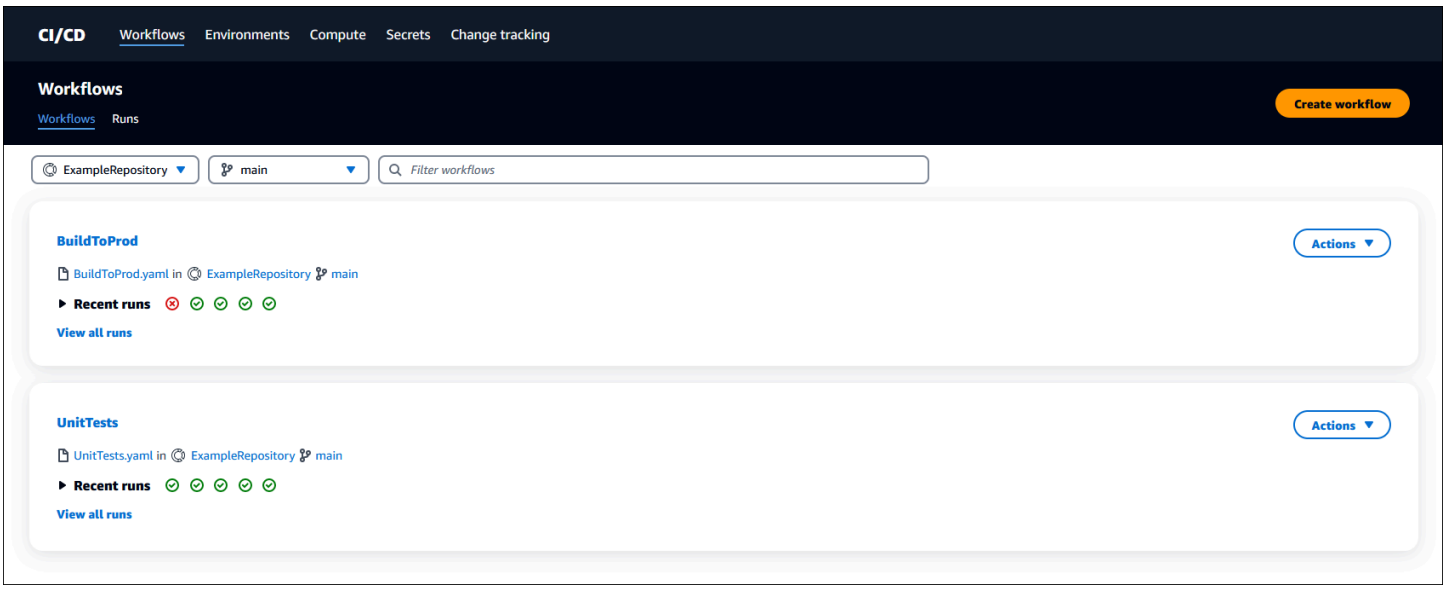


您可以从可视化编辑器切换到 YAML 编辑器，查看您的配置对底层 YAML 代码的影响。

发现工作流

您可以在工作流摘要页面查看您的工作流，以及您在同一项目中设置的其它工作流。

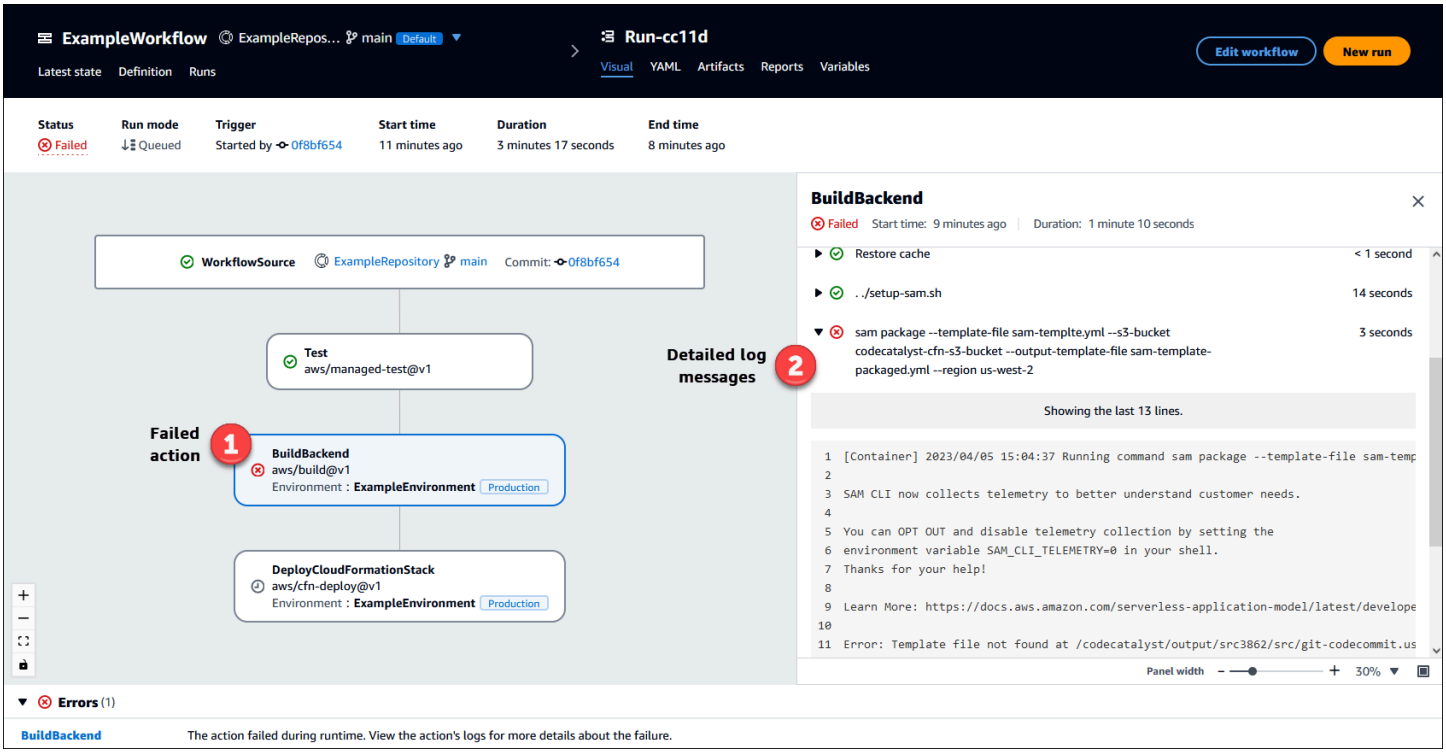
下图显示了工作流摘要页面。它包含两个工作流程：BuildToProd和UnitTests。您可以看到两者都运行过几次。您可以选择最近的运行以快速查看运行历史记录，也可以选择工作流名称以查看工作流的 YAML 代码和其它详细信息。



查看工作流运行详细信息

您可以通过在工作流摘要页面中选择运行来查看工作流运行的详细信息。

下图显示了名为 Run-cc11d 的工作流运行的详细信息，该工作流是在提交到源时自动启动的。工作流程图表明某项操作已失败（1）。您可以导航到日志（2）以查看详细的日志消息并解决问题。有关工作流运行的更多信息，请参阅运行工作流。



后续步骤

要了解有关工作流概念的更多信息，请参阅工作流概念。

要创建您的第一个工作流，请参阅入门工作流。

工作流概念

以下是使用工作流程构建、测试或部署代码时需要了解的一些概念和术语 CodeCatalyst。

工作流

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流

启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

工作流定义文件

工作流定义文件是描述您的工作流的 YAML 文件。默认情况下，该文件存储在源存储库根目录下的 `~/.codecatalyst/workflows/` 文件夹中。该文件的扩展名可以为 .yml 或 .yaml，并且扩展名必须小写。

有关工作流定义文件的更多信息，请参阅工作流 YAML 定义。

操作

操作是工作流的主要构建基块，它定义了工作流运行期间要执行的逻辑工作单元（又称任务）。通常，一个工作流包括多个按顺序运行或并行运行的操作，具体取决于您配置这些操作的方式。

有关操作的更多信息，请参阅配置工作流操作。

操作组

一个操作组包含一个或多个操作。将操作分组为操作组有助于将工作流保持得井井有条，还可以配置不同组之间的依赖关系。

有关操作组的更多信息，请参阅将操作分组为操作组。

构件

构件是工作流操作的输出，通常由文件夹或文件存档组成。构件之所以重要，是因为它们让您可以在操作之间共享文件和信息。

有关构件的更多信息，请参阅在操作之间共享构件和文件。

计算

计算是指为运行工作流操作而管理和维护的计算引擎（CPU、内存和操作系统）。 CodeCatalyst

有关计算的更多信息，请参阅配置计算和运行时映像。

环境

不要将 CodeCatalyst 环境与开发环境混淆，它定义了 CodeCatalyst 工作流程所连接的目标 AWS 账户 和可选的 Amazon VPC。环境还定义了工作流程访问目标账户中的 AWS 服务和资源所需的 IAM 角色。

您可以设置多个环境并为它们指定名称，例如开发、测试、暂存和生产。在这些环境中部署时，有关部署的信息会显示在环境中的 CodeCatalyst 部署活动和部署目标选项卡上。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs。

阶段门

阶段门是一个工作流组件，您可以用它来要求工作流必须满足特定条件才能继续运行。门禁的一个例子是批准门，在该门禁中，用户必须在 CodeCatalyst控制台中提交批准，然后才能允许工作流程继续运行。

您可以在工作流中的操作序列之间或在第一个操作（源下载后立即运行）之前添加阶段门。如果需要的话，您也可以在最后一个操作之后添加阶段门。

有关阶段门的更多信息，请参阅用阶段门控制工作流运行。

Reports

报告包含有关工作流运行期间所执行测试的详细信息。您可以创建报告，例如测试报告、代码覆盖率报告、软件组成分析报告和静态分析报告。您可以使用测试报告帮助解决在工作流运行期间发生的问题。如果您有来自多个工作流的许多报告，则可以使用报告来查看趋势和故障率，以帮助您优化应用程序和部署配置。

有关报告的更多信息，请参阅质量报告类型。

运行

运行是一个工作流的单次迭代。在运行期间， CodeCatalyst执行工作流配置文件中定义的操作并输出关联的日志、构件和变量。

有关运行的更多信息，请参阅运行工作流。

来源

源也称为输入源，是一个源存储库，工作流操作连接到该存储库以获取执行操作所需的文件。例如，工作流操作可能连接到源存储库来获取应用程序源文件，以便构建应用程序。

有关来源的更多信息，请参阅将源存储库连接到工作流。

变量

变量是一个键值对，其中包含您可以在 Ama CodeCatalyst zon 工作流程中引用的信息。工作流运行时，变量的值部分将替换为实际值。

有关变量的更多信息，请参阅在工作流中使用变量。

工作流触发器

工作流触发器简称触发器，使您可以在发生某些事件（例如代码推送）时自动启动工作流运行。您可能需要配置触发器，使软件开发人员不必通过 CodeCatalyst 控制台手动启动工作流程。

您可以使用三种类型的触发器：

- 推送 – 每当推送提交时，代码推送触发器都会启动工作流运行。
- 拉取请求 – 每当创建、修改或关闭拉取请求时，拉取请求触发器都会启动工作流运行。
- 计划 – 计划触发器使工作流运行按您定义的计划启动。您可以考虑使用计划触发器在夜间运行软件的版本，这样在第二天早上可以准备好最新的版本供开发人员处理。

您可以单独使用推送、拉取请求和计划触发器，也可以在同一个工作流中组合使用这些触发器。

触发器是可选的，如果您未配置任何触发器，则只能手动启动工作流。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

入门工作流

在本教程中，您将学习如何创建和配置您的第一个工作流。

> **Tip**
>
> 喜欢使用预配置的工作流开始？ 请参阅使用蓝图创建项目，其中包含有关使用正常运行的工作流设置项目的说明、示例应用程序和其他资源。

主题

先决条件

开始前的准备工作：

- 你需要一个 CodeCatalyst 空间。有关更多信息，请参阅 创建空间。
- 在你的 CodeCatalyst 空间里，你需要一个名为：

  ```
  codecatalyst-project
  ```

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。
- 在你的项目中，你需要一个 CodeCatalyst 名为：

  ```
  codecatalyst-source-repository
  ```

  有关更多信息，请参阅 创建源存储库。

> **Note**
>
> 如果您有现有的项目和源存储库，则可以使用它们；但是，创建新的项目和源存储库后，在本教程结束时的清理工作会更轻松。

步骤 1：创建并配置工作流

在此步骤中，您将创建和配置一个工作流，该工作流会在有更改时自动构建和测试源代码。

创建工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。

2. 选择创建工作流。

   工作流程定义文件显示在 CodeCatalyst 控制台的 YAML 编辑器中。

配置工作流

您可以在可视化编辑器或 YAML 编辑器中配置工作流。让我们从 YAML 编辑器开始，然后再转到可视化编辑器。

1. 选择 + 操作以查看可添加到工作流中的工作流操作列表。
2. 在构建操作中，选择 + 可将操作的 YAML 添加到您的工作流定义文件中。工作流现在应类似于如下所示。

```
Name: Workflow_fe47
SchemaVersion: "1.0"

# Optional - Set automatic triggers.
Triggers:
  - Type: Push
    Branches:
      - main

# Required - Define action configurations.
Actions:
  Build_f0:
    Identifier: aws/build@v1

    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this workflow as
 a source

    Outputs:
      AutoDiscoverReports:
        Enabled: true
        # Use as prefix for the report files
        ReportNamePrefix: rpt

    Configuration:
      Steps:
        - Run: echo "Hello, World!"
        - Run: echo "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" >> report.xml
```

```
        - Run: echo "<testsuite tests=\"1\" name=\"TestAgentJunit\" >" >>
report.xml
        - Run: echo "<testcase classname=\"TestAgentJunit\" name=\"Dummy
            Test\"/></testsuite>" >> report.xml
```

该工作流程将WorkflowSource源存储库中的文件复制到运行Build_f0操作的计算机上，打印Hello, World!到日志，在计算机上发现测试报告，然后将其输出到 CodeCatalyst 控制台的“报告” 页面。

3. 选择可视化以在可视化编辑器中查看工作流定义文件。您可以通过可视化编辑器中的字段来配置 YAML 编辑器中显示的 YAML 属性。

步骤 2：通过提交来保存工作流

在此步骤中，您将保存所做更改。由于工作流以 .yaml 文件的形式存储在存储库中，因此您可以通过提交来保存更改。

提交工作流更改

1. （可选）选择验证以确保工作流 YAML 代码有效。
2. 选择提交。
3. 在工作流文件名中，输入工作流配置文件的名称，例如 **my-first-workflow**。
4. 在提交消息中，输入消息来标识您的提交，例如 **create my-first-workflow.yaml**。
5. 在存储库中，选择要在其中保存工作流的存储库（codecatalyst-repository）。
6. 在分支名称中，选择要将工作流保存到的分支（main）。
7. 选择提交。

您的新工作流将显示在工作流列表中。它可能需要一点时间才能显示。

由于工作流与提交保存在一起，并且工作流配置了代码推送触发器，因此保存工作流会自动启动工作流运行。

步骤 3：查看运行结果

在此步骤中，您将导航到从您的提交启动的运行，并查看结果。

查看运行结果

1. 选择工作流的名称，例如 Workflow_fe47。

   显示源存储库标签 (WorkflowSource) 和生成操作（例如 build_F 0）的工作流程图。
2. 在工作流运行图表中，选择构建操作（例如 Build_f0）。
3. 查看日志、报告、配置和变量选项卡的内容。这些选项卡显示构建操作的结果。

   有关更多信息，请参阅查看构建操作的结果。

（可选）步骤 4：清理

在此步骤中，您将清除在本教程中创建的资源。

删除资源

- 如果您为本教程创建了新项目，请将其删除。有关说明，请参阅删除项目。删除项目还会删除源存储库和工作流。

使用工作流进行构建

使用CodeCatalyst 工作流程，您可以构建应用程序和其他资源。

主题

- 如何构建应用程序？
- 构建操作的益处
- 构建操作的替代方案
- 添加构建操作
- 查看构建操作的结果
- 教程：将构件上传到 Amazon S3
- 构建和测试操作 YAML

如何构建应用程序？

要在中构建应用程序或资源 CodeCatalyst，请先创建一个工作流程，然后在其中指定构建操作。

部署操作是一个工作流构建基块，可编译源代码、运行单元测试以及生成可供部署的构件。

您可以使用 CodeCatalyst 控制台的可视化编辑器或 YAML 编辑器将生成操作添加到工作流程中。

构建应用程序或资源的步骤大致如下。

构建应用程序（高级别任务）

1. 在中 CodeCatalyst，您可以为要构建的应用程序添加源代码。有关更多信息，请参阅 将源代码存储在项目的存储库中 CodeCatalyst。
2. 在中 CodeCatalyst，您可以创建工作流程。在工作流中，您可以定义如何构建、测试和部署应用程序。有关更多信息，请参阅 入门工作流。
3. （可选）在工作流中，您可以添加触发器，该触发器指示将导致工作流自动启动的事件。有关更多信息，请参阅使用触发器自动启动工作流运行
4. 在工作流中，您可以添加构建操作来编译和打包应用程序或资源源代码。（可选）如果您不希望将测试操作或部署操作用于这些目的，也可以让构建操作运行单元测试、生成报告和部署应用程序。有关测试操作和部署操作的更多信息，请参阅添加构建操作。
5. （可选）在工作流中，您可以添加测试操作和部署操作，来测试和部署您的应用程序或资源。您可以从多个预先配置的操作中进行选择，将应用程序部署到不同的目标，例如 Amazon ECS。有关更多信息，请参阅使用工作流进行测试和使用工作流进行部署。
6. 您可以手动启动工作流，也可以通过触发器自动启动工作流。该工作流按顺序运行构建、测试和部署操作，以便构建和测试您的应用程序和资源，并将其部署到目标。有关更多信息，请参阅 手动启动工作流运行。

构建操作的益处

在工作流中使用构建操作有以下益处：

- 完全托管 – 构建操作可消除设置、修补、更新和管理自己的编译服务器的需要。
- 按需 – 构建操作可以按需扩展，以满足您的构建需求。您只需为使用的构建分钟数付费。有关更多信息，请参阅 配置计算和运行时映像。
- 开箱即用 — CodeCatalyst 包括预先打包的运行时环境 Docker 镜像，这些镜像用于运行所有工作流程操作，包括构建操作。这些图像预先配置了用于构建应用程序（例如， AWS CLI 和 Node.js）的有用工具。您可以配置 CodeCatalyst 为使用从公共或私有注册表提供的构建映像。有关更多信息，请参阅 指定运行时环境映像。

构建操作的替代方案

如果您使用生成操作来部署应用程序，请考虑改用 CodeCatalyst部署操作。部署操作会执行 behind-the-scenes配置，否则如果您使用的是构建操作，则必须手动编写这些配置。有关可用的部署操作的更多信息，请参阅部署操作列表。

您也可以使用 AWS CodeBuild 来构建应用程序。有关更多信息，请参阅什么是 CodeBuild？。

添加构建操作

使用以下步骤将生成操作添加到您的 CodeCatalyst 工作流程中。

Visual

使用可视化编辑器添加构建操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择可视化。
6. 选择操作。
7. 在操作中，选择构建。
8. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅构建和测试操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加构建操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。

3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择 YAML。
6. 选择操作。
7. 在操作中，选择构建。
8. 根据需求修改 YAML 代码中的属性。构建和测试操作 YAML中提供了每个可用属性的解释。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

构建操作定义

构建操作定义为工作流定义文件中的一组 YAML 属性。有关这些属性的信息，请参阅工作流 YAML 定义中的构建和测试操作 YAML。

查看构建操作的结果

按照以下说明操作，查看构建操作的结果，包括生成的日志、报告和变量。

查看构建操作的结果

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
3. 在工作流图表中，选择构建操作的名称，例如构建。
4. 要查看构建运行的日志，请选择日志。这将显示各个构建阶段的日志。您可以根据需要展开或折叠日志。
5. 要查看构建操作生成的测试报告，请选择报告，或者在导航窗格中选择报告。有关更多信息，请参阅 质量报告类型。
6. 要查看用于构建操作的配置，请选择配置。有关更多信息，请参阅 添加构建操作。
7. 要查看构建操作使用的变量，请选择变量。有关更多信息，请参阅 在工作流中使用变量。

教程：将构件上传到 Amazon S3

在本教程中，您将学习如何使用包含几个构建操作的 Amazon CodeCatalyst 工作流程将项目上传到 Amazon S3 存储桶。当工作流启动时，这些操作将按顺序运行。第一个构建操作生成两个文件（`Hello.txt` 和 `Goodbye.txt`），并将它们捆绑到一个构建构件中。第二个构建操作将构件上传到 Amazon S3。您需要将工作流配置为每次将提交命令推送到源存储库时运行。

主题

先决条件

在开始之前，您需要：

- 你需要一个带有关联 AWS 账户的 CodeCatalyst 空间。有关更多信息，请参阅 创建空间。
- 在您的空间中，您需要一个空项目，其名称为：

  ```
  codecatalyst-artifact-project
  ```

  使用从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。
- 在你的项目中，你需要一个 CodeCatalyst 名为：

  ```
  codecatalyst-artifact-environment
  ```

  按如下方式配置此环境：
  - 选择任何类型，例如开发。
  - 将您的 AWS 账户与之关联。

- 对于默认 IAM 角色，选择任何角色。稍后需要指定另一个角色。

有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。

步骤 1：创建 AWS 角色

在此步骤中，您将创建一个 AWS IAM 角色，稍后将该角色分配给工作流程中的构建操作。此角色授予 CodeCatalyst 构建操作访问您的 AWS 账户和写入存储项目的 Amazon S3 的权限。该角色被称为构建角色。

> **Note**
>
> 如果您已经为其他教程创建了构建角色，也可以在本教程中使用该角色。只要确保该角色具有以下过程中显示的权限和信任策略即可。

有关 IAM 角色的更多信息，请参阅AWS AWS Identity and Access Management 用户指南中的 IAM 角色。

创建构建角色

1. 按如下步骤操作，为角色创建策略：
   a. 登录到 AWS。
   b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   c. 在导航窗格中，选择策略。
   d. 选择创建策略。
   e. 选择 JSON 选项卡。
   f. 删除现有代码。
   g. 粘贴以下代码：

      JSON

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

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-s3-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-s3-build-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-s3-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst build role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

步骤 2：创建 Amazon S3 存储桶

在此步骤中，您将创建一个 Amazon S3 存储桶，`Hello.txt` 和 `Goodbye.txt` 构件会上传到该存储桶中。

创建 Amazon S3 存储桶

1. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 在主窗格中，选择创建存储桶。
3. 对于存储桶名称，输入：

```
codecatalyst-artifact-bucket
```

4. 对于 AWS 区域，选择一个区域。本教程假设您选择了美国西部（俄勒冈州）us-west-2。有关 Amazon S3 支持的区域的信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service endpoints and quotas。
5. 在页面底部选择创建存储桶。
6. 复制您刚刚创建的存储桶的名称，例如：

```
codecatalyst-artifact-bucket
```

现在，您已经在美国西部（俄勒冈州）us-west-2 区域中创建了一个名为 **codecatalyst-artifact-bucket** 的存储桶。

步骤 3：创建源存储库

在此步骤中，您将在中创建源存储库 CodeCatalyst。此存储库用于存储本教程的工作流定义文件。

有关源存储库的更多信息，请参阅创建源存储库。

创建源存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目 `codecatalyst-artifact-project`。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。
5. 在存储库名称中，输入：

   ```
   codecatalyst-artifact-source-repository
   ```

6. 选择创建。

现在，您已经创建了一个名为 `codecatalyst-artifact-source-repository` 的存储库。

步骤 4：创建工作流

在此步骤中，您将创建一个工作流，其中包含以下按顺序运行的构建基块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- 一个名为 `GenerateFiles` 的构建操作 – `GenerateFiles` 操作在触发时，会创建两个文件（`Hello.txt` 和 `Goodbye.txt`），并将这两个文件打包到一个名为 `codecatalystArtifact` 的输出构件中。
- 另一个名为 `Upload` 的构建操作 – `GenerateFiles` 操作完成后，`Upload` 操作会运行 AWS CLI 命令 `aws s3 sync`，将 `codecatalystArtifact` 和源存储库中的文件上传到您的 Amazon S3 存储桶。已在 AWS CLI CodeCatalyst计算平台上预安装和预先配置，因此您无需安装或配置它。

  有关 CodeCatalyst 计算平台上预打包软件的更多信息，请参阅指定运行时环境映像。有关`aws s3 sync`命令 AWS CLI的更多信息，请参阅《AWS CLI 命令参考》中的 sync。

有关构建操作的更多信息，请参阅使用工作流进行构建。

创建工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择创建工作流。
3. 删除 YAML 示例代码。
4. 添加以下 YAML 代码：

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略此部分，则必须确保您环境的默认 IAM 角色字段中指定的角色包含步骤 1：创建 AWS 角色中描述的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。

```
Name: codecatalyst-artifact-workflow
SchemaVersion: 1.0

Triggers:
  - Type: Push
    Branches:
      - main
Actions:
  GenerateFiles:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        # Create the output files.
        - Run: echo "Hello, World!" > "Hello.txt"
        - Run: echo "Goodbye!" > "Goodbye.txt"
    Outputs:
      Artifacts:
        - Name: codecatalystArtifact
          Files:
            - "**/*"
  Upload:
    Identifier: aws/build@v1
    DependsOn:
      - GenerateFiles
    Environment:
      Name: codecatalyst-artifact-environment
      Connections:
```

```
        - Name: codecatalyst-account-connection
          Role: codecatalyst-s3-build-role
    Inputs:
      Artifacts:
        - codecatalystArtifact
    Configuration:
      Steps:
        # Upload the output artifact to the S3 bucket.
        - Run: aws s3 sync . s3://codecatalyst-artifact-bucket
```

在以上代码中，进行如下替换：

- *codecatalyst-artifact-environment*使用您在中创建的环境的名称先决条件。
- *codecatalyst-account-connection*使用您在中创建的账户连接的名称先决条件。
- 将 *codecatalyst-s3-build-role* 替换为您在步骤 1：创建 AWS 角色中创建的构建角色的名称。
- *codecatalyst-artifact-bucket*使用您在中创建的 Amazon S3 的名称步骤 2：创建 Amazon S3 存储桶。

有关此文件中的属性的信息，请参阅构建和测试操作 YAML。

5. （可选）选择验证，确保 YAML 代码在提交之前有效。
6. 选择提交。
7. 在提交工作流对话框中，输入以下内容：

   a. 对于工作流文件名，保留默认值 codecatalyst-artifact-workflow。

   b. 对于提交消息，输入：

   ```
   add initial workflow file
   ```

   c. 对于存储库，选择 codecatalyst-artifact-source-repository。

   d. 对于分支名称，选择主。

   e. 选择提交。

现在，您已创建工作流。由于在工作流顶部定义了触发器，因此工作流运行会自动启动。具体而言，当您将 codecatalyst-artifact-workflow.yaml 文件提交（并推送）到源存储库时，触发器启动了工作流运行。

查看正在运行的工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择您刚刚创建的工作流：codecatalyst-artifact-workflow。
3. 选择GenerateFiles查看第一个构建操作的进度。
4. 选择上传以查看第二个构建操作的进度。
5. 上传操作完成后，请执行以下操作：
   - 如果工作流运行成功，请转到下一过程。
   - 如果工作流运行失败，请选择日志来解决该问题。

步骤 5：验证结果

工作流程运行后，转到 Amazon S3 服务并查看您的*codecatalyst-artifact-bucket*存储桶。现在，该存储桶应包含以下文件和文件夹：

```
.
|— .aws/
|— .git/
|Goodbye.txt
|Hello.txt
|REAME.md
```

已上传 Goodbye.txt 和 Hello.txt 文件，因为它们是 codecatalystArtifact 构件的一部分。已上传 .aws/、.git/ 和 README.md 文件，因为它们位于您的源存储库中。

清理

清理干净 CodeCatalyst AWS ，避免为这些服务收费。

要清理干净 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 删除 codecatalyst-artifact-source-repository 源存储库。
3. 删除 codecatalyst-artifact-workflow 工作流。

要清理干净 AWS

1. 在 Amazon S3 中进行清理，如下所示：
   a. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
   b. 删除 `codecatalyst-artifact-bucket` 存储桶内的文件。
   c. 删除 `codecatalyst-artifact-bucket` 存储桶。
2. 在 IAM 中进行清理，如下所示：
   a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 删除 `codecatalyst-s3-build-policy`。
   c. 删除 `codecatalyst-s3-build-role`。

构建和测试操作 YAML

下面是构建和测试操作的 YAML 定义。两个操作只有一个参考，因为它们的 YAML 属性非常相似。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

在以下代码中选择一个 YAML 属性可查看其描述。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier: aws/build@v1 | aws/managed-test@v1
    DependsOn:
```

```
      - dependent-action-name-1
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Caching:
      FileCaching:
        key-name-1:
          Path: file1.txt
          RestoreKeys:
            - restore-key-1
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Outputs:
      Artifacts:
        - Name: output-artifact-1
          Files:
            - build-output/artifact-1.jar
            - "build-output/build*"
        - Name: output-artifact-2
          Files:
            - build-output/artifact-2.1.jar
            - build-output/artifact-2.2.jar
      Variables:
        - variable-name-1
        - variable-name-2
      AutoDiscoverReports:
        Enabled: true | false
        ReportNamePrefix: AutoDiscovered
        IncludePaths:
```

```
        - "**/*"
      ExcludePaths:
        - node_modules/cdk/junit.xml
      SuccessCriteria:
        PassRate: percent
        LineCoverage: percent
        BranchCoverage: percent
        Vulnerabilities:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisBug:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisSecurity:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisQuality:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisFinding:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
    Reports:
      report-name-1:
        Format: format
        IncludePaths:
          - "*.xml"
        ExcludePaths:
          - report2.xml
          - report3.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
            Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
            Number: whole-number
          StaticAnalysisBug:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisSecurity:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisQuality:
```

```
                Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
                Number: whole-number
            StaticAnalysisFinding:
                Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
                Number: whole-number
    Configuration:
      Container:
        Registry: registry
        Image: image
      Steps:
        - Run: "step 1"
        - Run: "step 2"
      Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: package-repository
              Scopes:
                - "@scope"
        ExportAuthorizationToken: true | false
```

action-name

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

对应的 UI：“配置”选项卡/操作名称

Identifier

(*action-name*/Identifier)

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

将 aws/build@v1 用于构建操作。

将 aws/managed-test@v1 用于测试操作。

对应的用户界面：工作流程图/操作名称/标签 *aws/build@v1|aws/managed-test@v1*

DependsOn

(*action-name*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*action-name*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

（*action-name*/Compute/类型）

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

对应的 UI：“配置”选项卡/计算类型

Fleet

(*action-name*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

对应的 UI：“配置”选项卡/计算实例集

Timeout

(*action-name*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Environment

(*action-name*/Environment)

（可选）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*action-name*/Environment/Name)

（可选）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*action-name*/Environment/Connections)

（可选）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Name

(*action-name*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Role

(*action-name*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定 IAM 角色的名称，该操作使用此角色来访问 Amazon S3 和 Amazon ECR 等 AWS 服务并在其中进行处理。确保将此角色添加到空间中的 AWS 账户 连接。要向账户连接添加 IAM 角色，请参阅将 IAM 角色添加到账户连接。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

> **Note**
>
> 您可以将 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色用于此操作。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

> **Warning**
>
> 仅将权限授予构建和测试操作所需的那些角色。使用具有更广泛权限的角色可能会带来安全风险。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Caching

(*action-name*/Caching)

（可选）

一个部分，可在其中指定缓存来保存磁盘上的文件，并在后续的工作流运行中从该缓存中恢复这些文件。

有关文件缓存的更多信息，请参阅在工作流运行之间缓存文件。

对应的 UI：“配置”选项卡/文件缓存 – 可选

FileCaching

(*action-name*/Caching/FileCaching)

（可选）

一个部分，用于指定缓存序列的配置。

相应的 UI：配置 tab/File 缓存-可选/添加缓存

key-name-1

(*action-name*/Caching/FileCaching/***key-name-1***)

（可选）

指定主缓存属性的名称。缓存属性名称在您的工作流内必须是唯一的。每个操作在 `FileCaching` 中最多可以有五个条目。

相应的 UI：配置 tab/File 缓存-可选/添加缓存/密钥

Path

(*action-name*/Caching/FileCaching/*key-name-1*/Path)

（可选）

为您的缓存指定关联路径。

相应的 UI：配置 tab/File 缓存-可选/添加缓存/路径

RestoreKeys

(*action-name*/Caching/FileCaching/*key-name-1*/RestoreKeys)

（可选）

指定在找不到主缓存属性时用作回退手段的恢复键。恢复键名称在您的工作流内必须是唯一的。每个缓存在 `RestoreKeys` 中最多可以有五个条目。

相应的 UI：配置 tab/File 缓存-可选/添加缓存/还原密钥-可选

Inputs

(*action-name*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间操作所需的数据。

> **Note**
>
> 每个构建操作或测试操作最多有四个输入（一个源和三个构件）。变量不计入此总数。

如果您需要引用驻留在不同输入（例如源和构件）中的文件，则源输入是主输入，构件是辅助输入。辅助输入中对文件的引用采用特殊前缀，以与主输入中的文件区分开来。有关更多信息，请参阅 示例：引用多个构件中的文件。

对应的 UI：输入选项卡

Sources

(*action-name*/Inputs/Sources)

（可选）

指定表示操作所需的源存储库的标签。当前，支持的唯一标签是 `WorkflowSource`，它表示存储工作流定义文件的源存储库。

如果省略源，则必须在 *action-name*/Inputs/Artifacts 下至少指定一个输入构件。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：无

Artifacts - input

(*action-name*/Inputs/Artifacts)

（可选）

指定以前操作中的一些构件，您希望将这些构件用作此操作的输入。这些构件必须已在以前的操作中定义为输出构件。

如果未指定任何输入构件，则必须在 *action-name*/Inputs/Sources 下指定至少一个源存储库。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

> **Note**
>
> 如果构件 – 可选下拉列表不可用（可视化编辑器），或者在验证 YAML 时出现错误（YAML 编辑器），这可能是因该操作仅支持一个输入导致的。在这种情况下，请尝试移除源输入。

对应的 UI：“输入”选项卡/构件 – 可选

Variables - input

(*action-name*/Inputs/Variables)

（可选）

指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输入”选项卡/变量 – 可选

Outputs

(*action-name*/Outputs)

（可选）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts - output

(*action-name*/Outputs/Artifacts)

（可选）

指定操作生成的构件的名称。构件名称在工作流内必须是唯一的，并且仅限于字母数字字符（a-z、A-Z、0-9）和下划线（_）。不允许使用空格、连字符（-）和特殊字符。不能使用引号以使输出构件名称包含空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件

Name

(*action-name*/Outputs/Artifacts/Name)

（如果包含 Artifacts - output，则为必需）

指定操作生成的构件的名称。构件名称在工作流内必须是唯一的，并且仅限于字母数字字符（a-z、A-Z、0-9）和下划线（_）。不允许使用空格、连字符（-）和特殊字符。不能使用引号以使输出构件名称包含空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的用户界面：输出tab/Artifacts/New输出/ 构建构件名称

Files

(*action-name*/Outputs/Artifacts/Files)

（如果包含 Artifacts - output，则为必需）

指定由操作输出的构件中 CodeCatalyst 包含的文件。这些文件由工作流操作在运行时生成，也可在您的源存储库中找到。文件路径可以位于源存储库或先前操作的构件中，并且相对于源存储库或构件根目录。您可以使用 glob 模式来指定路径。示例：

- 要指定位于构建位置或源存储库位置根目录中的单个文件，请使用 `my-file.jar`。
- 要在子目录中指定单个文件，请使用 `directory/my-file.jar` 或 `directory/subdirectory/my-file.jar`。
- 要指定所有文件，请使用 `"**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件和目录，请使用 `"directory/**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件，而非其任意子目录，请使用 `"directory/*"`。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（*）或其他特殊字符，请用双引号（""）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个构件或源中查找它。有关更多信息，请参阅引用源存储库文件和在构件中引用文件。

对应的用户界面：tab/Artifacts/New输出输出/编译生成的文件

Variables - output

(*action-name*/Outputs/Variables)

（可选）

指定希望操作导出的变量，以便后续操作可以使用这些变量。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输出”选项卡/变量/添加变量

variable-name-1

(*action-name*/Outputs/Variables/*variable-name-1*)

（可选）

指定希望操作导出的变量的名称。此变量必须已在同一操作的 `Inputs` 或 `Steps` 部分中定义。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的用户界面：输出tab/Variables/Add变量/名称

AutoDiscoverReports

(*action-name*/Outputs/AutoDiscoverReports)

（可选）

定义自动发现功能的配置。

启用自动发现后，会 CodeCatalyst 搜索操作中`Inputs`传递的所有文件以及操作本身生成的所有文件，以查找测试、代码覆盖率和软件组合分析 (SCA) 报告。对于找到的每个报告，将其 CodeCatalyst 转换为 CodeCatalyst 报告。CodeCatalyst 报告是完全集成到 CodeCatalyst 服务中的报告，可以通过 CodeCatalyst 控制台查看和操作。

> **Note**
>
> 默认情况下，自动发现功能会检查所有文件。您可以使用 IncludePaths 或 ExcludePaths 属性限制检查哪些文件。

对应的 UI：“输出”选项卡/报告/自动发现报告

Enabled

(*action-name*/Outputs/AutoDiscoverReports/Enabled)

（可选）

启用或禁用自动发现功能。

有效值为 `true` 或 `false`。

如果省略 `Enabled`，则默认值为 `true`。

对应的 UI：“输出”选项卡/报告/自动发现报告

ReportNamePrefix

(*action-name*/Outputs/AutoDiscoverReports/ReportNamePrefix)

（如果包含并启用 AutoDiscoverReports，则为必需）

为其找到的所有报告指定一个前缀，以便命名其关联 CodeCatalyst 的报告。 CodeCatalyst 例如，如果您将前缀指定为`AutoDiscovered`，并 CodeCatalyst自动发现两个测试报告`TestSuiteTwo.xml`，`TestSuiteOne.xml`则关联 CodeCatalyst 的报告将命名为 `AutoDiscoveredTestSuiteOne` and。`AutoDiscoveredTestSuiteTwo`

对应的 UI：“输出”选项卡/报告/前缀名称

IncludePaths

(*action-name*/Outputs/AutoDiscoverReports/IncludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/IncludePaths)

（如果包含并启用 AutoDiscoverReports，或者包含 Reports，则为必需）

指定搜索原始报告时 CodeCatalyst 包含的文件和文件路径。例如，如果您指定`"/test/report/*"`，则会在操作使用的整个构建映像中 CodeCatalyst 搜索该`/test/report/*`目录。当它找到该目录时， CodeCatalyst 然后在该目录中查找报告。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（`*`）或其他特殊字符，请用双引号（`""`）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

如果省略此属性，则默认值为 `"**/*"`，这意味着搜索范围包括所有路径的所有文件。

> **Note**
>
> 对于手动配置的报告，`IncludePaths` 必须是与单个文件匹配的 glob 模式。

对应的 UI：

- 输出tab/Reports/Auto-discover reports/Include/exclude路径/ 包含路径
- 输出tab/Reports/Manually配置报告//包含/排除路*report-name-1*径/包含路径

ExcludePaths

(*action-name*/Outputs/AutoDiscoverReports/ExcludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/ExcludePaths)

（可选）

指定搜索原始报告时 CodeCatalyst 排除的文件和文件路径。例如，如果您指定`"/test/my-reports/**/*"`，则 CodeCatalyst 不会在`/test/my-reports/`目录中搜索文件。要忽略某个目录中的所有文件，请使用 `**/*` glob 模式。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（`*`）或其他特殊字符，请用双引号（`""`）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

对应的 UI：

- 输出tab/Reports/Auto-discover reports/Include/exclude路径/ 排除路径
- 输出tab/Reports/Manually配置报告 *report-name-1* //包含/排除路径/排除路径

SuccessCriteria

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria)

（可选）

为测试、代码覆盖率、软件组成分析（SCA）和静态分析（SA）报告指定成功标准。

有关更多信息，请参阅 配置报告的成功标准。

对应的 UI：“输出”选项卡/报告/成功标准

PassRate

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/PassRate)

（可选）

指定测试报告中必须通过测试的百分比，关联 CodeCatalyst的报告才会被标记为通过。有效值包括十进制数字。例如：50、60.5。通过率标准仅适用于测试报告。有关测试报告的更多信息，请参阅测试报告。

对应的用户界面：输出tab/Reports/Success标准/通过率

LineCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/LineCoverage)

（可选）

指定代码覆盖率报告中必须覆盖的行数百分比，关联 CodeCatalyst 的报告才会被标记为通过。有效值包括十进制数字。例如：50、60.5。行覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

对应的用户界面：输出tab/Reports/Success标准/线路覆盖率

BranchCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/BranchCoverage)

（可选）

指定代码覆盖率报告中必须覆盖的分支百分比才能将关联 CodeCatalyst 报告标记为已通过。有效值包括十进制数字。例如：50、60.5。分支覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

相应的 UI：输出tab/Reports/Success标准/分支覆盖率

Vulnerabilities

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/Vulnerabilities)

（可选）

指定 SCA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大漏洞数量和严重性。要指定漏洞，您必须指定：

- 要计入的漏洞的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

  例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 漏洞的总数。
- 您希望允许的具有指定严重性的漏洞的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

漏洞标准仅适用于 SCA 报告。有关 SCA 报告的更多信息，请参阅软件组成分析报告。

要指定最低严重性，请使用 Severity 属性。要指定最大漏洞数，请使用 Number 属性。

相应的 UI：输出tab/Reports/Success标准/漏洞

StaticAnalysisBug

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisBug)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisBug)

（可选）

指定 SA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大错误数量和严重性。要指定错误，您必须指定：

- 要计入的错误的最低严重性。有效值（按严重程度从高到低）为 `CRITICAL`、`HIGH`、`MEDIUM`、`LOW` 和 `INFORMATIONAL`。

  例如，如果您选择 `HIGH`，则将计算 `HIGH` 和 `CRITICAL` 错误的总数。
- 您希望允许的具有指定严重性的错误的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

错误标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

要指定最低严重性，请使用 `Severity` 属性。要指定最大漏洞数，请使用 `Number` 属性。

对应的用户界面：输出tab/Reports/Success标准/错误

StaticAnalysisSecurity

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisSecurity)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisSecurity)

（可选）

指定 SA 报告中允许将关联 CodeCatalyst 报告标记为已通过的安全漏洞的最大数量和严重性。要指定安全漏洞，您必须指定：

- 要计入的安全漏洞的最低严重性。有效值（按严重程度从高到低）为 `CRITICAL`、`HIGH`、`MEDIUM`、`LOW` 和 `INFORMATIONAL`。

例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 安全漏洞的总数。

- 您希望允许的具有指定严重性的安全漏洞的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

安全漏洞标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

要指定最低严重性，请使用 Severity 属性。要指定最大漏洞数，请使用 Number 属性。

相应的 UI：输出tab/Reports/Success标准/安全漏洞

StaticAnalysisQuality

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisQuality)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisQuality)

（可选）

指定 SA 报告中允许将相关 CodeCatalyst 报告标记为已通过的最大质量问题数量和严重性。要指定质量问题，您必须指定：

- 要计入的质量问题的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

  例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 质量问题的总数。
- 您希望允许的具有指定严重性的质量问题的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

质量问题标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

要指定最低严重性，请使用 Severity 属性。要指定最大漏洞数，请使用 Number 属性。

相应的用户界面：输出tab/Reports/Success标准/质量问题

StaticAnalysisFinding

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisFinding)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisFinding)

（可选）

指定 SA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大结果数量和严重性。要指定调查发现，您必须指定：

- 要计入的调查发现的最低严重性。有效值（按严重程度从高到低）为 `CRITICAL`、`HIGH`、`MEDIUM`、`LOW` 和 `INFORMATIONAL`。

  例如，如果您选择 `HIGH`，则将计算 `HIGH` 和 `CRITICAL` 调查发现的总数。
- 您希望允许的具有指定严重性的调查发现的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

调查发现仅适用于 SARIF SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

要指定最低严重性，请使用 `Severity` 属性。要指定最大漏洞数，请使用 `Number` 属性。

相应的 UI：输出tab/Reports/Success标准/调查结果

Reports

(*action-name*/Outputs/Reports)

（可选）

一个部分，用于指定测试报告的配置。

对应的 UI：“输出”选项卡/报告

report-name-1

（*action-name*/Outputs/Reports/报告名称 1）

（如果包含 Reports，则为必需）

您要为将从原始 CodeCatalyst 报告生成的报告命名。

相应的 UI：输出tab/Reports/Manually配置报告/ 报告名称

Format

(*action-name*/Outputs/Reports/*report-name-1*/Format)

（如果包含 Reports，则为必需）

指定您用于报告的文件格式。可能值如下所示。

- 对于测试报告：
  - 对于 Cucumber JSON，请指定 Cucumber（可视化编辑器）或 CUCUMBERJSON（YAML 编辑器）。
  - 对于 JUnit XML，请指定 JUnit（可视化编辑器）或JUNITXML（YAML 编辑器）。
  - 对于 NUnit XML，请指定 NUnit（可视化编辑器）或NUNITXML（YAML 编辑器）。
  - 对于 NUnit 3 XML，请指定 NUnit3（可视化编辑器）或NUNIT3XML（YAML 编辑器）。
  - 对于 Visual Studio TRX，请指定 Visual Studio TRX（可视化编辑器）或 VISUALSTUDIOTRX（YAML 编辑器）。
  - 对于 TestNG XML，请指定 TestNG（可视化编辑器）或 TESTNGXML（YAML 编辑器）。
- 对于代码覆盖率报告：
  - 对于 Clover XML，请指定 Clover（可视化编辑器）或 CLOVERXML（YAML 编辑器）。
  - 对于 Cobertura XML，请指定 Cobertura（可视化编辑器）或 COBERTURAXML（YAML 编辑器）。
  - 对于 JaCoCo XML，请指定 JaCoCo（可视化编辑器）或JACOCOXML（YAML 编辑器）。
  - 对于由 s implecov 生成的 SimpleCov JSON，而不是 s implecov-json，请指定 S implecov（可视化编辑器）或（YAML 编辑器）。SIMPLECOV
- 对于软件组成分析（SCA）报告：
  - 对于 SARIF，请指定 SARIF（可视化编辑器）或 SARIFSCA（YAML 编辑器）。

对应的用户界面：输出tab/Reports/Manually configure reports/Add/configure报告*report-name-1*//报告类型和报告格式

Configuration

(*action-name*/Configuration)

（必需）可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

Container

(*action-name*/Configuration/Container)

（可选）

指定操作用于完成其处理的 Docker 映像或容器。您可以指定随附的活动图像之一 CodeCatalyst，也可以使用自己的图像。如果您选择使用自己的映像，则该映像可以位于 Amazon ECR、Docker Hub 或其他注册表中。如果您未指定 Docker 映像，则该操作会将其中一个活动映像用于其处理。有关默认使用哪个活动映像的信息，请参阅活动映像。

有关指定自己的 Docker 映像的更多信息，请参阅为操作分配自定义运行时环境 Docker 映像。

对应的 UI：运行时环境 Docker 映像 – 可选

Registry

(*action-name*/Configuration/Container/Registry)

（如果包含 `Container`，则为必需）

指定存储映像的注册表。有效值包括：

- `CODECATALYST`（YAML 编辑器）

  图像存储在 CodeCatalyst 注册表中。
- Docker Hub（可视化编辑器）或 `DockerHub`（YAML 编辑器）

  映像存储在 Docker Hub 映像注册表中。
- 其他注册表（可视化编辑器）或 `Other`（YAML 编辑器）

  映像存储在自定义映像注册表中。可以使用任何公开可用的注册表。
- Amazon Elastic Container Registry（可视化编辑器）或 `ECR`（YAML 编辑器）

  映像存储在 Amazon Elastic Container Registry 映像存储库中。要使用 Amazon ECR 存储库中的映像，此操作需要对 Amazon ECR 的访问权限。要启用此访问权限，您必须创建包含以下权限和自定义信任策略的 IAM 角色。（如果需要，可以修改现有角色以包含这些权限和策略。）

  IAM 角色必须在其角色策略中包含以下权限：

- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:GetAuthorizationToken`
- `ecr:GetDownloadUrlForLayer`

IAM 角色必须包含以下自定义信任策略：

有关如何创建 IAM 角色的更多信息，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

创建该角色后，必须通过环境将该角色分配给操作。有关更多信息，请参阅 将环境与操作关联。

对应的 UI：Amazon Elastic Container Registry、Docker Hub 和其他注册表选项

Image

(*action-name*/Configuration/Container/Image)

（如果包含 `Container`，则为必需）

指定下列项之一：

- 如果您使用的是 `CODECATALYST` 注册表，请将映像设置为以下活动映像之一：
  - `CodeCatalystLinux_x86_64:2024_03`
  - `CodeCatalystLinux_x86_64:2022_11`
  - `CodeCatalystLinux_Arm64:2024_03`
  - `CodeCatalystLinux_Arm64:2022_11`
  - `CodeCatalystLinuxLambda_x86_64:2024_03`
  - `CodeCatalystLinuxLambda_x86_64:2022_11`
  - `CodeCatalystLinuxLambda_Arm64:2024_03`
  - `CodeCatalystLinuxLambda_Arm64:2022_11`
  - `CodeCatalystWindows_x86_64:2022_11`
- 如果您使用的是 Docker Hub 注册表，请将映像设置为 Docker Hub 映像名称和可选标签。

  示例：`postgres:latest`
- 如果您使用的是 Amazon ECR 注册表，请将映像设置为 Amazon ECR 注册表 URI。

示例：111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo

- 如果您使用的是自定义注册表，请将映像设置为自定义注册表所期望的值。

对应的 UI：运行时环境 Docker 映像（如果注册表是 CODECATALYST）、Docker Hub 映像（如果注册表是 Docker Hub）、ECR 映像 URL（如果注册表是 Amazon Elastic Container Registry）和映像 URL（如果注册表是其他注册表）。

Steps

(*action-name*/Configuration/Steps)

（必需）

指定要在操作期间运行的 shell 命令，这些命令可用于安装、配置和运行构建工具。

以下是有关如何构建 npm 项目的示例：

```
Steps:
  - Run: npm install
  - Run: npm run build
```

以下是有关如何指定文件路径的示例：

```
Steps:
  - Run: cd $ACTION_BUILD_SOURCE_PATH_WorkflowSource/app  && cat file2.txt
  - Run: cd $ACTION_BUILD_SOURCE_PATH_MyBuildArtifact/build-output/  && cat file.txt
```

有关指定文件路径的更多信息，请参阅引用源存储库文件和在构件中引用文件。

对应的 UI：“配置”选项卡/Shell 命令

Packages

(*action-name*/Configuration/Packages)

（可选）

一个部分，可在其中指定操作用于解析依赖关系的程序包存储库。使用程序包可以安全地存储和共享用于开发应用程序的软件包。

有关程序包的更多信息，请参阅在中发布和共享软件包 CodeCatalyst。

对应的 UI：“配置”选项卡/程序包

NpmConfiguration

(*action-name*/Configuration/Packages/NpmConfiguration)

（如果包含 Packages，则为必需）

一个用于定义 npm 程序包格式的配置的部分。此配置在工作流运行期间由操作使用。

有关 npm 程序包配置的更多信息，请参阅使用 npm。

对应的用户界面：配置tab/Packages/Add配置/ npm

PackageRegistries

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries)

（如果包含 Packages，则为必需）

一个部分，可在其中定义一系列程序包存储库的配置属性。

对应的 UI：配置tab/Packages/Add configuration/npm/添加软件包存储库

PackagesRepository

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries/PackagesRepository)

（如果包含 Packages，则为必需）

指定您希望该操作使用的 CodeCatalyst 软件包存储库的名称。

如果您指定多个默认存储库，则优先使用最后一个存储库。

有关程序包存储库的更多信息，请参阅程序包存储库。

相应的 UI：配置tab/Packages/Add configuration/npm/Add包存储库/ Package 存储库

Scopes

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries/Scopes)

（可选）

指定要在程序包注册表中定义的范围序列。定义范围时，将指定的程序包存储库配置为所有列出范围的注册表。如果通过 npm 客户端请求具有范围的程序包，此程序包将使用该存储库而不是默认存储库。每个范围名称必须以“@”为前缀。

如果您包含覆盖范围，则优先使用最后一个存储库。

如果省略 `Scopes`，则将指定的程序包存储库配置为该操作使用的所有程序包的默认注册表。

有关范围的更多信息，请参阅程序包命名空间和范围限定的程序包。

相应的 UI：配置tab/Packages/Add configuration/npm/Add包存储库/ 作用域-可选

ExportAuthorizationToken

(*action-name*/Configuration/Packages/ExportAuthorizationToken)

（可选）

启用或禁用导出授权令牌功能。如果启用，则可以使用导出的授权令牌来手动配置软件包管理器以使用软件 CodeCatalyst 包存储库进行身份验证。您可以将该令牌用作可在操作中引用的环境变量。

有效值为 `true` 或 `false`。

如果省略 `ExportAuthorizationToken`，则默认值为 `false`。

有关导出授权令牌的更多信息，请参阅在工作流操作中使用授权令牌。

对应的 UI：“配置”选项卡/程序包/导出授权令牌

使用工作流进行测试

在中 CodeCatalyst，您可以将测试作为不同工作流程操作（例如生成和测试）的一部分来运行。这些工作流操作都可以生成质量报告。测试操作是生成测试、代码覆盖率、软件组成分析和静态分析报告的工作流操作。这些报告显示在 CodeCatalyst 控制台中。

主题

- 质量报告类型
- 添加测试操作

- 查看测试操作的结果
- 跳过操作中失败的测试
- 与集成 universal-test-runner
- 在操作中配置质量报告
- 测试的最佳实践
- 支持的 SARIF 属性

质量报告类型

Amazon CodeCatalyst 测试操作支持以下类型的质量报告。有关如何在 YAML 中格式化这些报告的示例，请参阅质量报告 YAML 示例。

主题

- 测试报告
- 代码覆盖率报告
- 软件组成分析报告
- 静态分析报告

测试报告

在中 CodeCatalyst，您可以配置在生成期间运行的单元测试、集成测试和系统测试。然后 CodeCatalyst 可以创建包含测试结果的报告。

您可以使用测试报告来帮助解决测试问题。如果您有来自多个构建的许多测试报告，您可以使用测试报告查看失败率，以帮助您优化构建。

您可以使用以下测试报告文件格式：

- Cucumber JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX（.trx、.xml）

代码覆盖率报告

在中 CodeCatalyst，您可以为测试生成代码覆盖率报告。 CodeCatalyst 提供了以下代码覆盖率指标：

行覆盖率

衡量测试涵盖的语句数量。语句是一条指令，不包括注释。

```
line coverage = (total lines covered)/(total number of lines)
```

分支覆盖率

衡量在 `if` 或 `case` 语句等控制结构的每个可能分支中，测试覆盖了多少分支。

```
branch coverage = (total branches covered)/(total number of branches)
```

支持以下代码覆盖率报告文件格式：

- JaCoCo XML (.xml)
- SimpleCov JSON（由 s implecov 生成，而不是 simplecov-json，.js on）
- Clover XML（版本 3，.xml）
- Cobertura XML（.xml）
- LCOV（.info）

软件组成分析报告

在中 CodeCatalyst，您可以使用软件组成分析 (SCA) 工具来分析应用程序的组件并检查是否存在已知的安全漏洞。您可以发现并解析 SARIF 报告，其中详细列出了不同严重程度的漏洞以及修复方法。有效的严重程度值（按严重程度从高到低）为 `CRITICAL`、`HIGH`、`MEDIUM`、`LOW` 和 `INFORMATIONAL`。

支持以下 SCA 报告文件格式：

- SARIF（.sarif、.json）

静态分析报告

您可以使用静态分析（SA）报告来识别源代码级缺陷。在中 CodeCatalyst，您可以在部署代码之前生成 SA 报告以帮助解决代码中的问题。这些问题包括错误、安全漏洞、质量问题和其它漏洞。有效的严重程度值（按严重程度从高到低）为 `CRITICAL`、`HIGH`、`MEDIUM`、`LOW` 和 `INFORMATIONAL`。

CodeCatalyst 提供了以下 SA 指标：

错误

识别源代码中可能存在的错误。这些错误可能包括有关内存安全的问题。下面是一个错误示例。

```
// The while loop will inadvertently index into array x out-of-bounds
int x[64];
while (int n = 0; n <= 64; n++) {
  x[n] = 0;
}
```

安全漏洞

识别源代码中可能存在的安全漏洞。这些安全漏洞可能包括以明文方式存储密钥令牌等问题。

质量问题

识别源代码中可能存在的质量问题。这些质量问题可能包括风格惯例方面的问题。下面是一个质量问题示例。

```
// The function name doesn't adhere to the style convention of camelCase
int SUBTRACT(int x, int y) {
  return x-y
}
```

其它漏洞

识别源代码中可能存在的其它漏洞。

CodeCatalyst 支持以下 SA 报告文件格式：

- PyLint (.py)
- ESLint （.js、.jsx、.ts、.tsx）
- SARIF（.sarif、.json）

添加测试操作

使用以下步骤向您的 CodeCatalyst 工作流程添加测试操作。

Visual

使用可视化编辑器添加测试操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择可视化。
6. 选择操作。
7. 在操作中，选择测试。
8. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅构建和测试操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加构建操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择 YAML。
6. 选择操作。
7. 在操作中，选择测试。
8. 根据需求修改 YAML 代码中的属性。构建和测试操作 YAML中提供了每个可用属性的解释。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

测试操作定义

测试操作被定义为工作流定义文件中的一组 YAML 属性。有关这些属性的信息，请参阅工作流 YAML 定义中的构建和测试操作 YAML。

查看测试操作的结果

按照以下说明查看测试操作的结果，包括生成的日志、报告和变量。

查看测试操作的结果

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
3. 在工作流程图中，选择测试操作的名称，例如测试。
4. 要查看操作生成的日志，请选择日志。将显示各个操作阶段的日志。您可以根据需要展开或折叠日志。
5. 要查看测试操作生成的测试报告，请选择报告，或者在导航窗格中选择报告。有关更多信息，请参阅 质量报告类型。
6. 要查看用于测试操作的配置，请选择配置。有关更多信息，请参阅 添加测试操作。
7. 要查看测试操作使用的变量，请选择变量。有关更多信息，请参阅 在工作流中使用变量。

跳过操作中失败的测试

如果您的操作有多条测试命令，您可能希望使操作中的后续测试命令能够运行，即使前一条命令失败。例如，在以下命令中，您可能希望 `test2` 始终运行，即使 `test1` 失败。

```
Steps:
- Run: npm install
- Run: npm run test1
- Run: npm run test2
```

通常，当步骤返回错误时，Amazon CodeCatalyst 会停止该工作流程操作并将其标记为失败。通过将错误输出重定向为 `null`，可让操作步骤继续运行。您可以在命令中添加 `2>/dev/null` 来做到这一点。经过这样的修改，前面的示例就变成了下面的样子。

```
Steps:
- Run: npm install
```

```
- Run: npm run test1 2>/dev/null
- Run: npm run test2
```

在第二个代码片段中，npm install 命令的状态将被保留，但 npm run test1 命令返回的任何错误将被忽略。因此，npm run test2 命令就会运行。这样，无论是否发生错误，您都可以同时查看两份报告。

与集成 universal-test-runner

测试操作与开源命令行工具 universal-test-runner 集成。universal-test-runner 使用测试执行协议，在给定的框架内为任何语言运行测试。universal-test-runner 支持以下框架：

- Gradle
- Jest
- Maven
- pytest
- .NET

universal-test-runner 只安装在为测试操作策管的映像上。如果将测试操作配置为使用自定义 Docker Hub 或 Amazon ECR，则必须手动安装 universal-test-runner 才能启用高级测试功能。为此，请在映像上安装 Node.js（14 或更高版本），然后使用 Shell 命令 - Run: npm install -g @aws/universal-test-runner 通过 npm 安装 universal-test-runner。有关通过 Shell 命令在容器中安装 Node.js 的更多信息，请参阅 Installing and Updating Node Version Manager。

有关 universal-test-runner 的更多信息，请参阅什么是 universal-test-runner？

Visual

universal-test-runner在可视化编辑器中使用

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。
4. 选择编辑。
5. 选择可视化。
6. 选择操作。
7. 在操作中，选择测试。

8. 在配置选项卡上，使用您选择的支持框架更新示例代码，完成 Shell 命令字段。例如，要使用支持的框架，可以使用类似下面的 Run 命令。

```
- Run: run-tests <framework>
```

如果您想要的框架不受支持，请考虑提供自定义适配器或运行程序。有关 Shell 命令字段的描述，请参阅 Steps。

9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

YAML

要 universal-test-runner在 YAML 编辑器中使用

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。
4. 选择编辑。
5. 选择 YAML。
6. 选择操作。
7. 在操作中，选择测试。
8. 根据需要修改 YAML 代码。例如，要使用支持的框架，可以使用类似下面的 Run 命令。

```
Configuration:
  Steps:
    - Run: run-tests <framework>
```

如果您想要的框架不受支持，请考虑提供自定义适配器或运行程序。有关 Steps 属性的描述，请参阅 Steps。

9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

在操作中配置质量报告

本节介绍如何在操作中配置质量报告。

主题

自动发现和手动报告

启用自动发现后，将 CodeCatalyst 搜索传递给操作的所有输入以及操作本身生成的所有文件，以查找测试、代码覆盖率、软件组合分析 (SCA) 和静态分析 (SA) 报告。您可以在中查看和操作每个报告 CodeCatalyst。

您还可以手动配置生成哪些报告。您可以指定要生成的报告类型和文件格式。有关更多信息，请参阅质量报告类型。

配置报告的成功标准

您可以设置决定测试、代码覆盖率、软件组成分析（SCA）或静态分析（SA）报告成功标准的值。

成功标准是决定报告通过还是失败的阈值。 CodeCatalyst 首先生成您的报告，该报告可以是测试、代码覆盖率、SCA 或 SA 报告，然后将成功标准应用于生成的报告。然后说明是否达到了成功标准以及达到的程度。如果任何报告不符合指定的成功标准，则指定成功标准的 CodeCatalyst 操作将失败。

例如，当您为 SCA 报告设置成功标准时，从最严重到最不严重的有效漏洞值是：`CRITICAL`、`HIGH`、`MEDIUM`、`LOW`、`INFORMATIONAL`。如果您设置的标准是扫描一个严重程度为 `HIGH` 的漏洞，那么如果至少有一个严重程度为 `HIGH` 的漏洞，或没有严重程度为 `HIGH` 的漏洞，但至少有一个严重程度更高的漏洞，如一个严重程度为 `CRITICAL` 的漏洞，报告就会失败。

如果您不指定成功标准，那么：

- 根据您的原始 CodeCatalyst 报告生成的报告不会显示成功标准。
- 成功标准不会用于确定相关工作流操作是通过还是失败。

Visual

配置成功标准

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择包含生成报告的操作的工作流。这是您想要为其应用成功标准的报告。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

3. 选择编辑。
4. 选择可视化。
5. 在工作流程图中，选择已配置为生成 CodeCatalyst 报告的操作。
6. 选择输出选项卡。
7. 在自动发现报告或手动配置报告下，选择成功标准。

   成功标准出现。根据您之前的选择，您可能会看到以下任一或全部选项：

   通过率

   指定测试报告中必须通过测试的百分比，关联 CodeCatalyst的报告才能标记为通过。有效值包括十进制数字。例如：50、60.5。通过率标准仅适用于测试报告。有关测试报告的更多信息，请参阅测试报告。

   行覆盖率

   指定代码覆盖率报告中必须覆盖的行数百分比，关联 CodeCatalyst 的报告才会被标记为通过。有效值包括十进制数字。例如：50、60.5。行覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

   分支覆盖率

   指定代码覆盖率报告中必须覆盖的分支百分比才能将关联 CodeCatalyst 报告标记为已通过。有效值包括十进制数字。例如：50、60.5。分支覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

   漏洞（SCA）

   指定 SCA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大漏洞数量和严重性。要指定漏洞，您必须指定：

   - 要计入的漏洞的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

     例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 漏洞的总数。
   - 您希望允许的具有指定严重性的漏洞的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

   漏洞标准仅适用于 SCA 报告。有关 SCA 报告的更多信息，请参阅软件组成分析报告。

错误

指定 SA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大错误数量和严重性。要指定错误，您必须指定：

- 要计入的错误的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

  例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 错误的总数。
- 您希望允许的具有指定严重性的错误的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

错误标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

安全漏洞

指定 SA 报告中允许将关联 CodeCatalyst 报告标记为已通过的安全漏洞的最大数量和严重性。要指定安全漏洞，您必须指定：

- 要计入的安全漏洞的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

  例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 安全漏洞的总数。
- 您希望允许的具有指定严重性的安全漏洞的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

安全漏洞标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

质量问题

指定 SA 报告中允许将相关 CodeCatalyst 报告标记为已通过的最大质量问题数量和严重性。要指定质量问题，您必须指定：

- 要计入的质量问题的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 质量问题的总数。

- 您希望允许的具有指定严重性的质量问题的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

质量问题标准仅适用于 PyLint 和 ESLint SA 报告。有关 SA 报告的更多信息，请参阅静态分析报告。

8. 选择提交。
9. 运行您的工作流程以 CodeCatalyst 将成功标准应用于原始报告，然后重新生成包含成功标准信息的关联 CodeCatalyst 报告。有关更多信息，请参阅 手动启动工作流运行。

YAML

配置成功标准

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择包含生成报告的操作的工作流。这是您想要为其应用成功标准的报告。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
3. 选择编辑。
4. 选择 YAML。
5. 在工作流程图中，选择已配置为生成 CodeCatalyst 报告的操作。
6. 在详细信息窗格中，选择输出选项卡。
7. 在操作、`AutoDiscoverReports`部分或`Reports`部分中，添加一个`SuccessCriteria`属性以及、`PassRate`、`LineCoverage`、`BranchCoverageVulnerabilities`、`StaticAnalysisBug`和`StaticAnalysisQuality`属性。

   有关这些属性的说明，请参阅构建和测试操作 YAML。
8. 选择提交。
9. 运行您的工作流程以 CodeCatalyst 将成功标准应用于原始报告，然后重新生成包含成功标准信息的关联 CodeCatalyst 报告。有关启动工作流的更多信息，请参阅手动启动工作流运行。

质量报告 YAML 示例

下面的示例显示了如何手动配置四份报告：测试报告、代码覆盖率报告、软件组成分析报告和静态分析报告。

```
Reports:
  MyTestReport:
    Format: JUNITXML
    IncludePaths:
      - "*.xml"
    ExcludePaths:
      - report1.xml
      SuccessCriteria:
        PassRate: 90
  MyCoverageReport:
    Format: CLOVERXML
    IncludePaths:
      - output/coverage/jest/clover.xml
      SuccessCriteria:
        LineCoverage: 75
        BranchCoverage: 75
  MySCAReport:
    Format: SARIFSCA
    IncludePaths:
      - output/sca/reports.xml
      SuccessCriteria:
        Vulnerabilities:
          Number: 5
          Severity: HIGH
  MySAReport:
    Format: ESLINTJSON
    IncludePaths:
      - output/static/eslint.xml
      SuccessCriteria:
        StaticAnalysisBug:
          Number: 10
          Severity: MEDIUM
        StaticAnalysisSecurity:
          Number: 5
          Severity: CRITICAL
        StaticAnalysisQuality:
          Number: 0
          Severity: INFORMATIONAL
```

测试的最佳实践

在使用提供的测试功能时 CodeCatalyst，我们建议您遵循以下最佳实践。

主题

自动发现

在中配置操作时 CodeCatalyst，自动发现允许您自动发现各种工具（例如 JUnit 测试报告）的输出，并从中生成相关 CodeCatalyst报告。自动发现功能有助于确保，即使已发现输出的名称或路径发生变化，也能继续生成报告。添加新文件后， CodeCatalyst会自动发现它们并生成相关报告。不过，如果您使用自动发现功能，则必须考虑到该功能的以下几个方面：

- 在操作中激活自动发现功能后，所有自动发现的同类型报告将共享相同的成功标准。例如，最低通过率等共享标准将适用于所有自动发现的测试报告。如果同类型的报告需要不同的标准，则必须明确配置每种报告。
- 自动发现功能还可以找到由依赖项生成的报告，如果配置了成功标准，可能会导致对这些报告的操作失败。这个问题可以通过更新排除路径配置来解决。
- 自动发现功能不能保证每次都能生成相同的报告列表，因为它会在运行时扫描操作。如果您希望始终生成特定报告，则应明确配置报告。例如，如果测试在构建过程中停止运行，测试框架就不会产生任何输出，因此也就不会生成测试报告，而且该操作可能会成功。如果您希望操作的成功与否取决于该特定测试，则必须明确配置该报告。

> **Tip**
>
> 开始执行新项目或现有项目时，请对整个项目目录（包括 **/*）使用自动发现功能。这将调用项目中的所有文件生成报告，包括子目录中的文件。

有关更多信息，请参阅 在操作中配置质量报告。

成功标准

您可以通过配置成功标准对报告执行质量阈值。例如，如果自动发现了两份代码覆盖率报告，一份的行覆盖率为 80%，另一份的行覆盖率为 60%，则您有以下选项：

- 将行覆盖率的自动发现成功标准设置为 80%。这将导致第一份报告通过，第二份报告失败，从而导致整体操作失败。要解除工作流的阻塞，请在项目中添加新测试，直到第二份报告的行覆盖率超过 80%。
- 将行覆盖率的自动发现成功标准设置为 60%。这将导致两份报告都通过，从而使操作成功。然后，您可以在第二份报告中努力提高代码覆盖率。然而，采用这种方法，您无法保证第一份报告的覆盖率不会降至 80% 以下。
- 使用可视化编辑器或为每份报告添加明确的 YAML 部分和路径，明确配置一份或两份报告。这样就可以为每份报告配置单独的成功标准和自定义名称。但是，采用这种方法时，如果报告路径发生变化，操作可能会失败。

有关更多信息，请参阅 配置报告的成功标准。

包含/排除路径

查看操作结果时，您可以调整 CodeCatalyst 通过配置`IncludePaths`和生成的报告列表`ExcludePaths`。

- 用于`IncludePaths`指定搜索报告时 CodeCatalyst 要包含的文件和文件路径。例如，如果您指定`"/test/report/*"`，则会在操作使用的整个构建映像中 CodeCatalyst 搜索该`/test/report/`目录。当它找到该目录时， CodeCatalyst 然后在该目录中查找报告。

  > **Note**
  >
  > 对于手动配置的报告，`IncludePaths` 必须是与单个文件匹配的 glob 模式。

- 用于`ExcludePaths`指定搜索报告时 CodeCatalyst 要排除的文件和文件路径。例如，如果您指定`"/test/reports/**/*"`，则 CodeCatalyst不会在`/test/reports/`目录中搜索文件。要忽略某个目录中的所有文件，请使用 `**/*` glob 模式。

以下是可能的 glob 模式示例。

模式	说明
`*.*`	匹配当前目录中所有包含点的对象名称
`*.xml`	匹配当前目录中所有以 `.xml` 结尾的对象名称

模式	说明
*.{xml,txt}	匹配当前目录中所有以 .xml 或 .txt 结尾的对象名称
**/*.xml	匹配所有目录中以 .xml 结尾的对象名称
testFolder	匹配名为 testFolder 的对象，将其视为文件
testFolder/*	匹配 testFolder 的子文件夹一级中的对象，如 testFolder/file.xml
testFolder/*/*	匹配 testFolder 的子文件夹两级中的对象，如 testFolder/reportsFolder/file.xml
testFolder/**	匹配子文件夹 testFolder 以及 testFolder 下的文件，如 testFolder/file.xml 和 testFolder/otherFolder/file.xml

CodeCatalyst 按如下方式解释全局模式：

- 斜杠 (/) 字符用于分隔文件路径中的目录。
- 星号 (*) 字符与不跨越文件夹边界的名称组分的零个或多个字符匹配。
- 双星号 (**) 与所有目录中名称组分的零个或多个字符匹配。

> **Note**
>
> ExcludePaths 优先于 IncludePaths。如果 IncludePaths 和 ExcludePaths 都包含同一个文件夹，则不会扫描该文件夹以获取报告。

支持的 SARIF 属性

静态分析结果交换格式 (SARIF) 是一种输出文件格式，可用于 Amazon 的软件组成分析 (SCA) 和静态分析报告。 CodeCatalyst下面的示例显示了如何在静态分析报告中手动配置 SARIF：

```
Reports:
MySAReport:
Format: SARIFSA
IncludePaths:
    - output/sa_report.json
SuccessCriteria:
    StaticAnalysisFinding:
    Number: 25
    Severity: HIGH
```

CodeCatalyst 支持以下 SARIF 属性，这些属性可用于优化分析结果在报告中的显示方式。

主题

`sarifLog` 对象

Name	必需	描述
`$schema`	是	版本 2.1.0 的 SARIF JSON 架构的 URI。
`version`	是	CodeCatalyst 仅支持 SARIF 版本 2.1.0。
`runs[]`	是	SARIF 文件包含一个或多个运行组成的数组，每个运行代表分析工具的一次运行。

`run` 对象

Name	必需	描述
`tool.driver`	是	描述分析工具的 `toolComponent` 对象。
`tool.name`	否	表示用于执行分析的工具名称的属性。
`results[]`	是	上显示的分析工具的结果 CodeCatalyst。

`toolComponent` 对象

Name	必需	描述
`name`	是	分析工具的名称。
`properties.artifactScanned`	否	工具分析的构件总数。
`rules[]`	是	代表规则的 `reportingDescriptor` 对象数组。根据这些规则，分析工具会发现所分析代码中存在的问题。

`reportingDescriptor` 对象

Name	必需	描述
`id`	是	用于引用调查发现的规则的唯一标识符。 最大长度：1024 个字符
`name`	否	规则的显示名称。

Name	必需	描述
		最大长度：1024 个字符
shortDescription.text	否	规则的简短描述。 最大长度：3000 个字符
fullDescription.text	否	规则的完整描述。 最大长度：3000 个字符
helpUri	否	可以本地化的字符串，包含规则主要文档的绝对 URI。 最大长度：3000 个字符
properties.unscore	否	表示扫描调查发现是否已评分的标志。
properties.score.severity	否	一组固定的字符串，用于指定调查发现的严重程度。 最大长度：1024 个字符
properties.cvssv3_baseSeverity	否	Common Vulnerability Scoring System v3.1 的定性严重性评级。
properties.cvssv3_baseScore	否	CVSS v3 基本分值范围为 0.0 - 10.0。
properties.cvssv2_severity	否	如果 CVSS v3 的值不可用，则会 CodeCatalyst 搜索 CVSS v2 值。
properties.cvssv2_score	否	CVSS v2 基本分值范围为 0.0 - 10.0。

Name	必需	描述
properties.severity	否	一组固定的字符串，用于指定调查发现的严重程度。 最大长度：1024 个字符
defaultConfiguration.level	否	规则的默认严重性。

result 对象

Name	必需	描述
ruleId	是	用于引用调查发现的规则的唯一标识符。 最大长度：1024 个字符
ruleIndex	是	工具组件 rules[] 中相关规则的索引。
message.text	是	一条消息，描述结果并显示每个调查发现的消息。 最大长度：3000 个字符
rank	否	一个介于 0.0 到 100.0（含）之间的值，表示结果的优先级或重要性。0.0 表示最低优先级，100.0 表示最高优先级。
level	否	结果的严重性。 最大长度：1024 个字符
properties.unscore	否	表示扫描调查发现是否已评分的标志。

Name	必需	描述
properties.score.severity	否	一组固定的字符串，用于指定调查发现的严重程度。 最大长度：1024 个字符
properties.cvssv3_baseSeverity	否	Common Vulnerability Scoring System v3.1 的定性严重性评级。
properties.cvssv3_baseScore	否	CVSS v3 基本分值范围为 0.0 - 10.0。
properties.cvssv2_severity	否	如果 CVSS v3 的值不可用，则会 CodeCatalyst 搜索 CVSS v2 值。
properties.cvssv2_score	否	CVSS v2 基本分值范围为 0.0 - 10.0。
properties.severity	否	一组固定的字符串，用于指定调查发现的严重程度。 最大长度：1024 个字符
locations[]	是	检测到结果的位置集。除非只能通过在每个指定位置进行更改来纠正问题，否则只能包括一个位置。 CodeCatalyst 使用位置数组中的第一个值来注释结果。 location 对象的最大数量：10

Name	必需	描述
relatedLocations[]	否	调查发现中参考的其它位置列表。 location 对象的最大数量：50
fixes[]	否	代表扫描工具提供的建议的fix对象数组。 CodeCatalyst 使用数fixes组中的第一个建议。

location 对象

Name	必需	描述
physicalLocation	是	标识构件和区域。
logicalLocations[]	否	用名称描述的一组位置，不提及构件。

physicalLocation 对象

Name	必需	描述
artifactLocation.uri	是	表示构件位置的 URI，通常是存储库中的文件或在构建过程中生成的文件。
fileLocation.uri	否	表示文件位置的回退 URI。如果 artifactLocation.uri 返回空，则使用此项。
region.startLine	是	区域中第一个字符的行号。

Name	必需	描述
region.startColumn	是	区域中第一个字符的列号。
region.endLine	是	区域中最后一个字符的行号。
region.endColumn	是	区域中最后一个字符的列号。

logicalLocation 对象

Name	必需	说明
fullyQualifiedName	否	描述结果位置的其它信息。 最大长度：1024 个字符

fix 对象

Name	必需	说明
description.text	否	显示每个调查发现的建议的消息。 最大长度：3000 个字符
artifactChanges.[0].artifactLocation.uri	否	表示需要更新的构件位置的 URI。

使用工作流进行部署

使用CodeCatalyst 工作流程，您可以将应用程序和其他资源部署到各种目标， AWS Lambda例如 Amazon ECS 等。

如何部署应用程序？

要通过部署应用程序或资源 CodeCatalyst，首先要创建一个工作流，然后在其中指定部署操作。部署操作是一个工作流程构造块，它定义了要部署的内容、部署位置以及部署方式（例如，使用 blue/green 方案）。您可以使用 CodeCatalyst 控制台的可视化编辑器或 YAML 编辑器向工作流程添加部署操作。

部署应用程序或资源的步骤大致如下。

部署应用程序（高级别任务）

1. 在您的 CodeCatalyst 项目中，您可以为要部署的应用程序添加源代码。有关更多信息，请参阅 将源代码存储在项目的存储库中 CodeCatalyst。
2. 在您的 CodeCatalyst 项目中，您可以添加一个环境来定义目标 AWS 账户 和要部署到的可选亚马逊虚拟私有云 (VPC)。有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。
3. 在您的 CodeCatalyst 项目中，您可以创建工作流程。在工作流中，您可以定义如何构建、测试和部署应用程序。有关更多信息，请参阅 入门工作流。
4. 在工作流中，您可以添加触发器、构建操作以及（可选）测试操作。有关更多信息，请参阅使用触发器自动启动工作流运行、添加构建操作和添加测试操作。
5. 在工作流中，您可以添加部署操作。您可以从 CodeCatalyst提供的多个将应用程序部署到不同目标（例如 Amazon ECS）的操作中进行选择。（您也可以使用生成操作或 GitHub 操作来部署应用程序。 有关生成 GitHub 操作和操作的更多信息，请参阅部署操作的替代方案。）
6. 您可以手动启动工作流，也可以通过触发器自动启动工作流。该工作流按顺序运行构建、测试和部署操作，以将您的应用程序和资源部署到目标。有关更多信息，请参阅 手动启动工作流运行。

部署操作列表

提供了以下部署操作：

- 部署 CloudFormation 堆栈-此操作 AWS 基于您提供的CloudFormation 模板或AWS Serverless Application Model 模板在中创建 CloudFormation 堆栈。有关更多信息，请参阅 部署 CloudFormation 堆栈。
- 部署到 Amazon ECS – 此操作将注册您提供的任务定义文件。有关更多信息，请参阅 使用工作流部署到 Amazon ECS。
- 部署到 Kubernetes 集群 – 此操作将应用程序部署到 Amazon Elastic Kubernetes Service 集群。有关更多信息，请参阅 使用工作流部署到 Amazon EKS。

- AWS CDK 部署-此操作将AWS CDK 应用程序部署到。 AWS有关更多信息，请参阅 使用工作流程部署 AWS CDK 应用程序。

> **Note**
>
> 还有其他可以部署资源的 CodeCatalyst 操作；但是，它们不被视为部署操作，因为它们的部署信息不会显示在 “环境” 页面上。要详细了解环境页面和查看部署，请参阅部署到 AWS 账户和 VPCs和查看部署信息。

部署操作的优势

在工作流中使用部署操作有以下益处：

- 部署历史记录 – 查看部署历史记录，帮助管理和传达已部署软件中的更改。
- 可追溯性-通过 CodeCatalyst 控制台跟踪部署状态，并查看每个应用程序修订的部署时间和地点。
- 回滚 – 如果出现错误，则自动回滚部署。您还可以配置警报以激活部署回滚。
- 监控 – 观察工作流的各个阶段的部署进展。
- 与其他 CodeCatalyst 功能集成 — 存储源代码，然后通过一个应用程序构建、测试和部署源代码。

部署操作的替代方案

未强制您使用部署操作，但建议您这样做，因为部署操作可提供上一部分中列明的好处。相反，您可以使用以下CodeCatalyst 操作：

- 构建操作。

  通常，如果要部署到没有相应的部署操作的目标，或者要对部署过程进行更多控制，则可以使用构建操作。有关使用构建操作来部署资源的更多信息，请参阅使用工作流进行构建。
- 一个GitHub 动作。

  您可以在 CodeCatalyst 工作流程中使用GitHub 操作来部署应用程序和资源（而不是 CodeCatalyst 操作）。有关如何在 CodeCatalyst 工作流程中使用 GitHub 操作的信息，请参阅 与 GitHub 操作集成

如果您不想使用 CodeCatalyst 工作流程来部署应用程序，也可以使用以下 AWS 服务来部署应用程序：

- AWS CodeDeploy — 看看什么是 CodeDeploy？
- AWS CodeBuild 而且 AWS CodePipeline — 参见什么是 AWS CodeBuild？ 还有什么 AWS CodePipeline？
- CloudFormation — 看看什么是 CloudFormation？

使用 CodeDeploy、 CodeBuild CodePipeline、和 CloudFormation 服务进行复杂的企业部署。

主题

使用工作流部署到 Amazon ECS

本节介绍如何使用工作流程将容器化应用程序部署到 Amazon 弹性容器服务集群中。 CodeCatalyst 为此，您必须将部署到 Amazon ECS 操作添加到工作流。此操作将注册您提供的任务定义文件。注册后，任务定义将由在 Amazon ECS 集群中运行的 Amazon ECS 服务实例化。“实例化任务定义”等效于将应用程序部署到 Amazon ECS 中。

要使用此操作，您必须已拥有 Amazon ECS 集群、服务和任务定义文件。

有关 Amazon ECS 的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》。

> **Tip**
>
> 有关说明如何使用部署到 Amazon ECS 操作的教程，请参阅教程：将应用程序部署到 Amazon ECS。

> **Tip**
>
> 对于部署到 Amazon ECS 操作的工作示例，请使用带 AWS Fargate的 Node.js API 或带 AWS Fargate的 Java API 蓝图创建一个项目。有关更多信息，请参阅 使用蓝图创建项目。

主题

“部署到 Amazon ECS”操作使用的运行时映像

部署到 Amazon ECS 操作在 2022 年 11 月版映像上运行。有关更多信息，请参阅 活动映像。

教程：将应用程序部署到 Amazon ECS

在本教程中，您将学习如何使用工作流程、Amazon ECS 和其他一些服务将无服务器应用程序部署到亚马逊弹性容器服务 (Amazon ECS) 中。 AWS 部署的应用程序是一个基于 Apache Web 服务器 Docker 映像构建的简单 Hello World 网站。本教程将引导您完成所需的准备工作（例如设置集群），然后介绍如何创建用于构建和部署应用程序的工作流。

> **Tip**
>
> 您可以使用蓝图来执行完整的 Amazon ECS 设置，而不是按照本教程的说明操作。您将需要使用带 AWS Fargate的 Node.js API 或带 AWS Fargate的 Java API 蓝图。有关更多信息，请参阅 使用蓝图创建项目。

主题

先决条件

开始前的准备工作：

- 你需要一个带有关联 AWS 账户的 CodeCatalyst 空间。有关更多信息，请参阅 创建空间。
- 在您的空间中，您需要一个空项目，其名称为：

  ```
  codecatalyst-ecs-project
  ```

  使用从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。
- 在你的项目中，你需要一个 CodeCatalyst 名为：

  ```
  codecatalyst-ecs-environment
  ```

  按如下方式配置此环境：
  - 选择任何类型，例如非生产。
  - 将您的 AWS 账户与之关联。
  - 对于默认 IAM 角色，选择任何角色。稍后需要指定另一个角色。

有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。

步骤 1：设置 AWS 用户和 AWS CloudShell

本教程的第一步是在中创建用户 AWS IAM Identity Center，并以该用户的身份启动 AWS CloudShell 实例。在本教程中， CloudShell 是您的开发计算机，也是您配置 AWS 资源和服务的地方。完成本教程后，请删除此用户。

> **Note**
>
> 在本教程中，请不要使用根用户。您必须创建单独的用户，否则以后在 AWS Command Line Interface (CLI) 中执行操作时可能会遇到问题。

有关 IAM Identity Center 用户的更多信息以及 CloudShell，请参阅AWS IAM Identity Center 用户指南和AWS CloudShell 用户指南。

创建 IAM Identity Center 用户

1. 登录 AWS 管理控制台 并打开 AWS IAM Identity Center 控制台，网址为https://console.aws.amazon.com/singlesignon/。

   > **Note**
   >
   > 请务必使用与您的 CodeCatalyst空间 AWS 账户 相连的登录。您可以通过导航到您的空间并选择 AWS 账户选项卡来确认已连接哪个账户。有关更多信息，请参阅 创建空间。

2. 在导航窗格中，选择用户，然后选择添加用户。
3. 在用户名中，输入：

   ```
   CodeCatalystECSUser
   ```

4. 在密码下，选择生成可与此用户共享的一次性密码。
5. 在电子邮件地址和确认电子邮件地址中，输入 IAM Identity Center 中不存在的电子邮件地址。
6. 在名字和姓氏中，输入：

   ```
   CodeCatalystECSUser
   ```

7. 在显示名称中，保留自动生成的名称：

```
CodeCatalystECSUser CodeCatalystECSUser
```

8. 选择下一步。
9. 在将用户添加到组页面上，选择下一步。
10. 在查看并添加用户页面上，检查相应信息，然后选择添加用户。

    这将显示一次性密码对话框。
11. 选择复制，然后粘贴登录信息，包括 AWS 访问门户 URL 和一次性密码。
12. 选择关闭。

创建权限集

您稍后会将此权限集分配给 CodeCatalystECSUser。

1. 在导航窗格中，选择权限集，然后选择创建权限集。
2. 选择 “预定义权限集”，然后选择AdministratorAccess。该策略为所有 AWS 服务提供完全权限。
3. 选择下一步。
4. 在权限集名称中，输入：

```
CodeCatalystECSPermissionSet
```

5. 选择下一步。
6. 在查看和创建页面上，检查相应信息，然后选择创建。

将权限集分配给 CodeCatalyst ECSUser

1. 在导航窗格中 AWS 账户，选择，然后选中您当前登录 AWS 账户 的旁边的复选框。
2. 选择分配用户或组。
3. 选择用户选项卡。
4. 选中 CodeCatalystECSUser 旁边的复选框。
5. 选择下一步。
6. 选中 CodeCatalystECSPermissionSet 旁边的复选框。
7. 选择下一步。

8. 检查相应信息，然后选择提交。

   现在，你已经CodeCatalystECSPermissionSet将CodeCatalystECSUser和分配给你的 AWS 账户，将它们绑定在一起。

要注销并以身份重新登录 CodeCatalyst ECSUser

1. 在注销之前，请确保您拥有 AWS 访问门户 URL 以及的用户名和一次性密码CodeCatalystECSUser。您应在早些时候将此信息复制到文本编辑器中。

   > **Note**
   >
   > 如果您没有这些信息，请转至 IAM Identity Center 中的 CodeCatalystECSUser 详细信息页面，选择重置密码和生成一次性密码 [...]，然后再次选择重置密码以在屏幕上显示信息。

2. 退出 AWS。
3. 将 AWS 访问门户 URL 粘贴到浏览器的地址栏中。
4. 使用 CodeCatalystECSUser 的用户名和一次性密码进行登录。
5. 在新密码中，输入一个密码，然后选择设置新密码。

   屏幕上会出现一个 AWS 账户框。
6. 选择 AWS 账户，然后选择 AWS 账户 向其分配CodeCatalystECSUser用户和权限集的名称。
7. 在 CodeCatalystECSPermissionSet 旁，选择管理控制台。

   AWS 管理控制台 出现了。现在，您已经以具有适当权限的 CodeCatalystECSUser 的身份登录。

启动实 AWS CloudShell 例

1. 在顶部导航栏中，选择 AWS 图标
   ()。Co

   的主页 AWS 管理控制台 随即出现。
2. 在顶部导航栏中，选择图 AWS CloudShell 标
   ()。

CloudShell 打开。等待 CloudShell 环境创建完成。

> **Note**
>
> 如果您没有看到该 CloudShell 图标，请确保您所在的区域由支持 CloudShell。本教程假设您位于美国西部（俄勒冈州）区域。

验证 AWS CLI 是否已安装

1. 在终 CloudShell 端中输入：

```
aws --version
```

2. 检查是否显示了版本。

   已经 AWS CLI 为当前用户配置了`CodeCatalystECSUser`，因此无需像往常那样配置 AWS CLI 密钥和证书。

步骤 2：将占位符应用程序部署到 Amazon ECS 中

在此部分中，您手动将占位符应用程序部署到 Amazon ECS 中。此占位符应用程序将替换为由您的工作流部署的 Hello World 应用程序。占位符应用程序是 Apache Web 服务器。

有关 Amazon ECS 的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》。

完成以下一系列过程可部署占位符应用程序。

创建任务执行角色

此角色授予 Amazon ECS 和代表您进行 API 调用的 AWS Fargate 权限。

1. 创建信任策略：

   a. 在中 AWS CloudShell，输入以下命令：

   ```
   cat > codecatalyst-ecs-trust-policy.json
   ```

   CloudShell 终端中会出现闪烁的提示。

   b. 在提示符处，输入以下代码：

JSON

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

c. 将光标置于最后一个大括号（}）之后。

d. 按 **Enter** 和 **Ctrl+d** 以保存文件并退出 cat。

2. 创建任务执行角色：

```
aws iam create-role \
      --role-name codecatalyst-ecs-task-execution-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

3. 将 AWS 托管AmazonECSTaskExecutionRolePolicy策略附加到角色：

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-task-execution-role \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

4. 显示角色的详细信息：

```
aws iam get-role \
      --role-name codecatalyst-ecs-task-execution-role
```

5. 请记下角色的 "Arn": 值，例如 arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role。稍后您将需要此 Amazon 资源名称（ARN）。

要创建 Amazon ECS 集群

该集群将包含 Apache 占位符应用程序，稍后将包含 Hello World 应用程序。

1. 在中 CodeCatalystECSUser AWS CloudShell，创建一个空集群：

```
aws ecs create-cluster --cluster-name codecatalyst-ecs-cluster
```

2. （可选）验证是否已成功创建集群：

```
aws ecs list-clusters
```

codecatalyst-ecs-cluster 集群的 ARN 应显示在列表中，这表示创建成功。

创建任务定义文件

任务定义文件指示运行从中提取的 Apache 2.4 Web 服务器 Docker 镜像 (httpd:2.4)。 DockerHub

1. 在中 CodeCatalystECSUser AWS CloudShell，创建任务定义文件：

```
cat > taskdef.json
```

2. 在提示符处，粘贴以下代码：

```
{
    "executionRoleArn": "arn:aws:iam::111122223333:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            "image": "httpd:2.4",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
```

```
        "FARGATE"
    ],
    "cpu": "256",
    "family": "codecatalyst-ecs-task-def",
    "memory": "512",
    "networkMode": "awsvpc"
}
```

在前面的代码中，替换 *arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role*

替换为您在创建任务执行角色中记下的任务执行角色的 ARN。

3. 将光标置于最后一个大括号（}）之后。
4. 按 **Enter** 和 **Ctrl+d** 以保存文件并退出 cat。

将任务定义文件注册到 Amazon ECS

1. 在中 CodeCatalystECSUser AWS CloudShell，注册任务定义：

```
aws ecs register-task-definition \
    --cli-input-json file://taskdef.json
```

2. （可选）验证是否已注册任务定义：

```
aws ecs list-task-definitions
```

codecatalyst-ecs-task-def 任务定义应显示在列表中。

创建 Amazon ECS 服务

Amazon ECS 服务运行 Apache 占位符应用程序的任务（和关联的 Docker 容器），然后运行 Hello World 应用程序的任务。

1. 与 CodeCatalystECSUser 一样，切换到 Amazon Elastic Container Service 控制台（如果您尚未这样做）。
2. 选择您之前创建的集群 codecatalyst-ecs-cluster。
3. 在服务选项卡上，选择创建。
4. 在创建页面上，执行以下操作：

a. 保留所有默认设置，但接下来列出的设置除外。

b. 对于 Launch type (启动类型)，选择 FARGATE。

c. 在任务定义下的系列下拉列表中，选择：

```
codecatalyst-ecs-task-def
```

d. 对于服务名称，输入：

```
codecatalyst-ecs-service
```

e. 对于预期任务数，输入：

```
3
```

在本教程中，每个任务均启动一个 Docker 容器。

f. 展开联网部分。

g. 对于 VPC，请选择任意 VPC。

h. 对于子网，请选择任意子网。

> **Note**
>
> 仅指定一个子网。这就是本教程要求执行的所有操作。

> **Note**
>
> 如果您没有 VPC 和子网，请创建它们。请参阅《Amazon VPC 用户指南》中的创建 VPC 和在 VPC 中创建子网。

i. 对于安全组，选择创建新安全组，然后执行以下操作：

i. 对于安全组名称，输入：

```
codecatalyst-ecs-security-group
```

ii. 对于安全组描述，输入：

```
CodeCatalyst ECS security group
```

iii. 选择添加规则。对于类型，选择 HTTP；对于来源，选择任何位置。

j. 选择底部的创建。

k. 创建服务时请等待。该过程可能需要几分钟。

5. 选择任务选项卡，然后选择刷新按钮。确认所有三个任务的最后状态列都设置为正在运行。

（可选）验证 Apache 占位符应用程序是否正在运行

1. 在任务选项卡中，选择三个任务之一。
2. 在公有 IP 字段中，选择开放地址。

   此时将显示 `It Works!` 页面。这表明 Amazon ECS 服务已成功启动一项任务，该任务会启动带 Apache 映像的 Docker 容器。

   在本教程中，此时您已手动部署 Amazon ECS 集群、服务和任务定义以及 Apache 占位符应用程序。在所有这些项目准备就绪后，便可创建一个工作流以将 Apache 占位符应用程序替换为教程中的 Hello World 应用程序。

步骤 3：创建 Amazon ECR 映像存储库

在此部分中，您将在 Amazon Elastic Container Registry（Amazon ECR）中创建私有映像存储库。此存储库存储教程中的 Docker 映像，该映像将替换您之前部署的 Apache 占位符映像。

有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

在 Amazon ECR 中创建映像存储库

1. 在中 `CodeCatalystECSUser` AWS CloudShell，在 Amazon ECR 中创建一个空存储库：

```
aws ecr create-repository --repository-name codecatalyst-ecs-image-repo
```

2. 显示 Amazon ECR 存储库的详细信息：

```
aws ecr describe-repositories \
      --repository-names codecatalyst-ecs-image-repo
```

3. 记下 “repositoryUri”：值，例如 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo。

   稍后在向工作流添加存储库时需要使用它。

步骤 4：创建 AWS 角色

在本节中，您将创建 CodeCatalyst 工作流程运行所需的 AWS IAM 角色。这些角色是：

- 构建角色-授予 CodeCatalyst 构建操作（在工作流程中）访问您的 AWS 账户并写入 Amazon ECR 和 Amazon 的权限。 EC2
- 部署角色-授予 “ CodeCatalyst 部署到 ECS” 操作（在工作流程中）访问您的 AWS 账户、Amazon ECS 和其他一些 AWS 服务的权限。

有关 IAM 角色的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的 IAM 角色。

> **Note**
>
> 要节省时间，您可以创建一个名为 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色的角色，而不是前面列出的两个角色。有关更多信息，请参阅 为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有非常广泛的权限，这可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。本教程假定您创建的是前面列出的两个角色。

要创建生成和部署角色，您可以使用 AWS 管理控制台 或 AWS CLI。

AWS 管理控制台

要创建构建和部署角色，请完成以下一系列过程。

创建构建角色

1. 按如下步骤操作，为角色创建策略：

   a. 登录到 AWS。

b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

c. 在导航窗格中，选择策略。

d. 选择创建策略。

e. 选择 JSON 选项卡。

f. 删除现有代码。

g. 粘贴以下代码：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-ecs-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-ecs-build-policy，然后选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-ecs-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst ECS build role
```

j. 选择创建角色。

现在，您已创建一个具有权限策略和信任策略的构建角色。

3. 按如下步骤操作，获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-ecs-build-role）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。稍后您将需要用到它。

创建部署角色

1. 按如下步骤操作，为角色创建策略：

a. 登录到 AWS。

b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

c. 在导航窗格中，选择策略。

d. 选择创建策略。

e. 选择 JSON 选项卡。

f. 删除现有代码。

g. 粘贴以下代码：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
```

```
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用通配符。之后，您可以在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-ecs-deploy-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建部署角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-ecs-deploy-policy，然后选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-ecs-deploy-role
```

i. 对于角色描述，输入：

```
CodeCatalyst ECS deploy role
```

j. 选择创建角色。

现在，您已创建具有信任策略的部署角色。

3. 按如下步骤操作，获取部署角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-ecs-deploy-role）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。稍后您将需要用到它。

AWS CLI

要创建构建和部署角色，请完成以下一系列过程。

为两个角色创建信任策略

在中 CodeCatalystECSUser AWS CloudShell，创建信任策略文件：

1. 创建文件：

```
cat > codecatalyst-ecs-trust-policy.json
```

2. 在终端提示符处，粘贴以下代码：
3. 将光标置于最后一个大括号（}）之后。
4. 按 **Enter** 和 **Ctrl+d** 以保存文件并退出 cat。

创建构建策略和构建角色

1. 创建构建策略：
   a. 在中 CodeCatalystECSUser AWS CloudShell，创建生成策略文件：

```
cat > codecatalyst-ecs-build-policy.json
```

   b. 在提示符处，输入以下代码：

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

c. 将光标置于最后一个大括号（}）之后。

d. 按 **Enter** 和 **Ctrl+d** 以保存文件并退出 cat。

2. 将构建策略添加到 AWS：

```
aws iam create-policy \
    --policy-name codecatalyst-ecs-build-policy \
    --policy-document file://codecatalyst-ecs-build-policy.json
```

3. 在命令输出中，记下 "arn": 值，例如 arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy。稍后您将需要此 ARN。

4. 创建构建角色并向其附加信任策略：

```
aws iam create-role \
      --role-name codecatalyst-ecs-build-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 将构建策略附加到构建角色：

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy
```

其中*arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy*，替换为你之前提到的构建策略的 ARN。

6. 显示构建角色的详细信息：

```
aws iam get-role \
      --role-name codecatalyst-ecs-build-role
```

7. 请记下角色的 "Arn": 值，例如 arn:aws:iam::111122223333:role/codecatalyst-ecs-build-role。稍后您将需要此 ARN。

创建部署策略和部署角色

1. 创建部署策略：

a. 在中 AWS CloudShell，创建部署策略文件：

```
cat > codecatalyst-ecs-deploy-policy.json
```

b. 在提示符处，输入以下代码：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
```

```
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

c. 将光标置于最后一个大括号（}）之后。

d. 按 **Enter** 和 **Ctrl+d** 以保存文件并退出 cat。

2. 将部署策略添加到 AWS：

```
aws iam create-policy \
```

```
    --policy-name codecatalyst-ecs-deploy-policy \
    --policy-document file://codecatalyst-ecs-deploy-policy.json
```

3. 在命令输出中，记下部署策略的 "arn": 值，例如 arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy。稍后您将需要此 ARN。
4. 创建部署角色并向其附加信任策略：

```
aws iam create-role \
      --role-name codecatalyst-ecs-deploy-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 将部署策略附加到部署角色，其中*arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy*替换为您之前记下的部署策略的 ARN。

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-
policy
```

6. 显示部署角色的详细信息：

```
aws iam get-role \
      --role-name codecatalyst-ecs-deploy-role
```

7. 请记下角色的 "Arn": 值，例如 arn:aws:iam::111122223333:role/codecatalyst-ecs-deploy-role。稍后您将需要此 ARN。

步骤 5：将 AWS 角色添加到 CodeCatalyst

在此步骤中，您将构建角色 (codecatalyst-ecs-build-role) 和部署角色 (codecatalyst-ecs-deploy-role) 添加到空间中的 CodeCatalyst 账户连接。

将构建角色和部署角色添加到账户连接

1. 在中 CodeCatalyst，导航到您的空间。
2. 选择 AWS accounts (账户)。此时将显示账户连接列表。
3. 选择代表您在其中创建构建和部署角色的 AWS 账户的账户连接。
4. 从管理控制台中选择 “ AWS 管理角色”。

将出现 “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面。您可能需要登录才能访问该页面。

5. 选择添加您在 IAM 中创建的现有角色。

   这将显示一个下拉列表。该列表显示所有具有包含 `codecatalyst-runner.amazonaws.com` 和 `codecatalyst.amazonaws.com` 服务主体的信任策略的 IAM 角色。

6. 在该下拉列表中，选择 `codecatalyst-ecs-build-role`，然后选择添加角色。

   > **Note**
   >
   > 如果您看到的是 `The security token included in the request is invalid`，则可能是因为您不具有适当的权限。要解决此问题，请使用您在创建 CodeCatalyst空间时使用的 AWS 账号退出并重新登录。 AWS

7. 选择添加 IAM 角色，再选择添加您在 IAM 中创建的现有角色，然后在下拉列表中选择 `codecatalyst-ecs-deploy-role`。选择添加角色。

   现在，您已将构建角色和部署角色添加到您的空间。

8. 复制 A mazon CodeCatalyst 显示名称的值。您稍后在创建工作流时将需要此值。

步骤 6：创建源存储库

在此步骤中，您将在中创建源存储库 CodeCatalyst。此存储库将存储教程的源文件，例如任务定义文件。

有关源存储库的更多信息，请参阅创建源存储库。

创建源存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目 `codecatalyst-ecs-project`。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。
5. 在存储库名称中，输入：

   ```
   codecatalyst-ecs-source-repository
   ```

6. 选择创建。

步骤 7：添加源文件

在本节中，您将将 Hello World 源文件添加到您的 CodeCatalyst 存储库中`codecatalyst-ecs-source-repository`。它们包括：

- `index.html` 文件 – 在浏览器中显示 Hello World 消息。
- Dockerfile – 描述用于 Docker 映像的基本映像以及应用于该映像的 Docker 命令。
- `taskdef.json` 文件 – 定义在集群中启动任务时要使用的 Docker 映像。

文件夹结构如下所示：

```
.
|– public-html
|   |– index.html
|– Dockerfile
|– taskdef.json
```

> **Note**
>
> 以下说明向您展示了如何使用 CodeCatalyst 控制台添加文件，但如果您愿意，也可以使用 Git。有关更多信息，请参阅 克隆源存储库。

主题

- index.html
- Dockerfile
- taskdef.json

index.html

`index.html` 文件在浏览器中显示 Hello World 消息。

添加 index.html 文件

1. 在 CodeCatalyst 控制台中，转到您的源存储库`codecatalyst-ecs-source-repository`。

2. 在文件中，选择创建文件。
3. 对于文件名，输入：

```
public-html/index.html
```

> **Important**
>
> 请务必包含 public-html/ 前缀以创建同名文件夹。index.html 应位于此文件夹中。

4. 在文本框中，输入以下代码：

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

5. 选择提交，然后再次选择提交。

   这会将 index.html 添加到存储库中的 public-html 文件夹中。

Dockerfile

Dockerfile 描述要使用的基本 Docker 映像以及应用于该映像的 Docker 命令。有关 Dockerfile 的更多信息，请参阅 Dockerfile Reference。

此处指定的 Dockerfile 指示使用 Apache 2.4 基本映像（httpd）。它还包括用于将名为 index.html 的源文件复制到提供网页的 Apache 服务器上的文件夹中的指令。Dockerfile 中的 EXPOSE 指令告知 Docker 容器正在端口 80 上侦听。

添加 Dockerfile

1. 在源存储库中，选择创建文件。
2. 对于文件名，输入：

   ```
   Dockerfile
   ```

   不要包含文件扩展名。

   > **Important**
   >
   > Dockerfile 必须位于存储库的根目录文件夹中。工作流的 `Docker build` 命令期望它在该位置。

3. 在文本框中，输入以下代码：

   ```
   FROM httpd:2.4
   COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
   EXPOSE 80
   ```

4. 选择提交，然后再次选择提交。

   这会将 Dockerfile 添加到您的存储库中。

taskdef.json

您在此步骤中添加的 `taskdef.json` 文件与您在步骤 2：将占位符应用程序部署到 Amazon ECS 中中指定的文件相同，但有以下区别：

此处的任务定义使用变量 `$REPOSITORY_URI` 和 `$IMAGE_TAG` 来表示映像，而不是在 `image:` 字段（`httpd:2.4`）中指定硬编码的 Docker 映像名称。当您在下一个步骤中运行工作流时，这些变量将被替换为工作流的构建操作所生成的实际值。

有关任务定义参数的详细信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的任务定义参数。

添加 taskdef.json 文件

1. 在源存储库中，选择创建文件。
2. 对于文件名，输入：

taskdef.json

3. 在文本框中，输入以下代码：

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            # The $REPOSITORY_URI and $IMAGE_TAG variables will be replaced
            # by the workflow at build time (see the build action in the
            # workflow)
            "image": $REPOSITORY_URI:$IMAGE_TAG,
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "codecatalyst-ecs-task-def"
}
```

在上述代码中，将

arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role

替换为您在创建任务执行角色中记下的任务执行角色的 ARN。

4. 选择提交，然后再次选择提交。

这会将 taskdef.json 文件添加到您的存储库中。

步骤 8：创建并运行工作流

在此步骤中，您将创建一个工作流来提取源文件，将源文件构建到 Docker 映像中，然后将该映像部署到 Amazon ECS 集群。此部署将替换现有的 Apache 占位符应用程序。

工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- 构建操作（`BuildBackend`）– 此操作在触发后会使用 Dockerfile 构建 Docker 映像并将该映像推送到 Amazon ECR。构建操作还将使用正确的 `image` 字段值更新 `taskdef.json`，然后创建此文件的输出构件。此构件将用作接下来的部署操作的输入。

  有关构建操作的更多信息，请参阅使用工作流进行构建。
- 部署操作（`DeployToECS`）– 构建操作完成后，部署操作将查找构建操作（`TaskDefArtifact`）所生成的输出构件，查找其中包含的 `taskdef.json` 并将它注册到您的 Amazon ECS 服务。之后，该服务按照 `taskdef.json` 文件中的说明，在您的 Amazon ECS 集群中运行三个 Amazon ECS 任务以及关联的 Hello World Docker 容器。

创建工作流

1. 在 CodeCatalyst 控制台的导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择创建工作流。
3. 对于源存储库，选择 `codecatalyst-ecs-source-repository`。
4. 对于分支，选择 `main`。
5. 选择创建。
6. 删除 YAML 示例代码。
7. 添加以下 YAML 代码：

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 `Connections:` 部分。如果您省略这些部分，则必须确保您环境的默认 IAM 角色字段中指定的角色包含步骤 5：将 AWS 角色添加到 CodeCatalyst中描述的两个角色的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in taskdef.json
```

```
        - Run: find taskdef.json -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find taskdef.json -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|
g"
        - Run: cat taskdef.json
        # The output artifact will be a zip file that contains a task definition
 file.
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - taskdef.json
  DeployToECS:
    DependsOn:
      - BuildBackend
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-deploy-role
    Inputs:
      Sources: []
      Artifacts:
        - TaskDefArtifact
    Configuration:
      region: us-west-2
      cluster: codecatalyst-ecs-cluster
      service: codecatalyst-ecs-service
      task-definition: taskdef.json
```

在上述代码中，进行如下替换：

- 的两个实例都*codecatalyst-ecs-environment*与您在中创建的环境名称相同先决条件。
- 的两个实例都*codecatalyst-account-connection*带有您的账户连接的显示名称。显示名称可能是数字。有关更多信息，请参阅 步骤 5：将 AWS 角色添加到 CodeCatalyst。
- *codecatalyst-ecs-build-role*使用您在中创建的构建角色的名称步骤 4：创建 AWS 角色。
- *111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo*（在Value:属性中），其中包含您在中创建的 Amazon ECR 存储库的 URI。步骤 3：创建 Amazon ECR 映像存储库

- *111122223333.dkr.ecr.us-west-2.amazonaws.com*（在Run: aws ecr命令中）使用 Amazon ECR 存储库的 URI，不带图像后缀 () /codecatalyst-ecs-image-repo。
- *codecatalyst-ecs-deploy-role*使用您在中创建的部署角色的名称步骤 4：创建 AWS 角色。
- 两个实例都*us-west-2*使用您的 AWS 地区代码。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。

> **Note**
>
> 如果您决定不创建生成和部署角色，请*codecatalyst-ecs-deploy-role*使用CodeCatalystWorkflowDevelopmentRole-*spaceName*角色名称替换*codecatalyst-ecs-build-role*和。有关该角色的更多信息，请参阅步骤 4：创建 AWS 角色。

> **Tip**
>
> 您可以使用渲染 Amazon ECS 任务定义操作来更新存储库和映像名称，而不是使用前面的工作流代码中显示的 find 和 sed 命令。有关更多信息，请参阅 修改 Amazon ECS 任务定义。

8. （可选）选择验证，确保 YAML 代码在提交之前有效。
9. 选择提交。
10. 在提交工作流对话框中，输入以下内容：

    a. 对于提交消息，移除文本并输入：

    ```
    Add first workflow
    ```

    b. 对于存储库，选择 codecatalyst-ecs-source-repository。

    c. 对于分支名称，选择“主”。

    d. 选择提交。

现在，您已创建工作流。由于在工作流顶部定义了触发器，因此工作流运行会自动启动。具体而言，当您将 workflow.yaml 文件提交（并推送）到源存储库时，触发器启动了工作流运行。

查看工作流运行进度

1. 在 CodeCatalyst 控制台的导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择您刚刚创建的工作流：`codecatalyst-ecs-workflow`。
3. 选择BuildBackend查看构建进度。
4. 选择 DeployToECS 以查看部署进度。

   有关查看运行详细信息的更多信息，请参阅查看工作流运行状态和详细信息。

验证部署

1. 打开 Amazon ECS 经典控制台，网址为https://console.aws.amazon.com/ecs/。
2. 选择您的集群：`codecatalyst-ecs-cluster`。
3. 选择 Tasks 选项卡。
4. 选择三项任务中的任一项。
5. 在公有 IP 字段中，选择开放地址。

   浏览器中会出现“Hello World”页面，这表示 Amazon ECS 服务已成功部署您的应用程序。

步骤 9：对源文件进行更改

在此部分中，您将对源存储库中的 `index.html` 文件进行更改。此更改会促使工作流构建新的 Docker 映像，使用提交 ID 对它进行标记，将它推送到 Amazon ECR，然后将它部署到 Amazon ECS。

更改 index.html

1. 在 CodeCatalyst 控制台的导航窗格中，选择代码，然后选择源存储库，然后选择您的存储库`codecatalyst-ecs-source-repository`。
2. 选择 `public-html`，然后选择 `index.html`。

   这将显示 `index.html` 的内容。
3. 选择编辑。
4. 在第 14 行上，将 `Hello World` 文本更改为 `Tutorial complete!`。
5. 选择提交，然后再次选择提交。

   提交会促使启动新的工作流运行。

6. （可选）转到源存储库的主页，选择查看提交，然后记下 index.html 更改的提交 ID。
7. 查看部署进度：
   a. 在导航窗格中，选择 CI/CD，然后选择工作流。
   b. 选择 codecatalyst-ecs-workflow 以查看最新运行。
   c. 选择BuildBackend、和 DeployToECS 以查看工作流程的运行进度。
8. 验证是否已更新您的应用程序，如下所示：
   a. 打开 Amazon ECS 经典控制台，网址为https://console.aws.amazon.com/ecs/。
   b. 选择您的集群：codecatalyst-ecs-cluster。
   c. 选择 Tasks 选项卡。
   d. 选择三项任务中的任一项。
   e. 在公有 IP 字段中，选择开放地址。

   此时将显示 Tutorial complete! 页面。
9. （可选）在中 AWS，切换到 Amazon ECR 控制台，确认新 Docker 镜像已使用步骤 6 中的提交 ID 进行标记。

清理

清理本教程中使用的文件和服务以免被收取费用。

在中 AWS 管理控制台，按以下顺序进行清理：

1. 在 Amazon ECS 中，执行以下操作：
   a. 删除 codecatalyst-ecs-service。
   b. 删除 codecatalyst-ecs-cluster。
   c. 取消注册 codecatalyst-ecs-task-definition。
2. 在 Amazon ECR 中，删除 codecatalyst-ecs-image-repo。
3. 在亚马逊中 EC2，删除codecatalyst-ecs-security-group。
4. 在 IAM Identity Center 中，删除：
   a. CodeCatalystECSUser
   b. CodeCatalystECSPermissionSet

在 CodeCatalyst 控制台中，按如下方式进行清理：

1. 删除 codecatalyst-ecs-workflow。
2. 删除 codecatalyst-ecs-environment。
3. 删除 codecatalyst-ecs-source-repository。
4. 删除 codecatalyst-ecs-project。

在本教程中，您学习了如何使用 CodeCatalyst 工作流程和 “部署到 Amazon ECS” 操作将应用程序部署到 Amazon ECS 服务。

添加“部署到 Amazon ECS”操作

按照以下说明，将部署到 Amazon ECS 操作添加到工作流。

Visual

使用可视化编辑器添加“部署到 Amazon ECS”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索部署到 Amazon ECS 操作，然后执行下列操作之一：
   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或
   - 选择部署到 Amazon ECS。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“部署到 Amazon ECS”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。

11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加“部署到 Amazon ECS”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索部署到 Amazon ECS 操作，然后执行下列操作之一：
   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或
   - 选择部署到 Amazon ECS。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 根据需求修改 YAML 代码中的属性。“部署到 Amazon ECS”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

“部署到 Amazon ECS”变量

部署到 Amazon ECS 操作在运行时会生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 使用预定义变量。

键	值
cluster	在工作流运行期间部署到的 Amazon ECS 集群的名称。 示例：`codecatalyst-ecs-cluster`
deployment-platform	部署平台的名称。 硬编码为 `AWS:ECS`。
service	在工作流运行期间部署到的 Amazon ECS 服务的名称。 示例：`codecatalyst-ecs-service`
task-definition-arn	在工作流运行期间注册的任务定义的 Amazon 资源名称（ARN）。 示例：`arn:aws:ecs:us-west-2:111122223333:task-definition/codecatalyst-task-def:8` 前面的示例中的 `:8` 表示已注册的修订。
deployment-url	指向 Amazon ECS 控制台的事件选项卡的链接，可以在其中查看与工作流运行关联的 Amazon ECS 部署的详细信息。 示例：`https://console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/codecatalyst-ecs-cluster/services/codecatalyst-ecs-service/events`
region	在工作流程运行期间部署到的的区域代码。AWS 区域 示例：`us-west-2`

“部署到 Amazon ECS”操作 YAML

下面是部署到 Amazon ECS 操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流部署到 Amazon ECS。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSDeployAction_nn:
    Identifier: aws/ecs-deploy@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - task-definition-artifact
    Configuration:
```

```
      region: us-east-1
      cluster: ecs-cluster
      service: ecs-service
      task-definition: task-definition-path
      force-new-deployment: false|true
      codedeploy-appspec: app-spec-file-path
      codedeploy-application: application-name
      codedeploy-deployment-group: deployment-group-name
      codedeploy-deployment-description: deployment-description
```

ECSDeployAction

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：ECSDeployAction_nn。

对应的 UI：“配置”选项卡/操作显示名称

Identifier

(*ECSDeployAction*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/ecs-deploy@v1。

对应的用户界面：工作流程图/ ECSDeploy action_nn/ aws/ecs-deploy @v1 标签

DependsOn

(*ECSDeployAction*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*ECSDeployAction*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*ECSDeployAction*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：配置 tab/Advanced -可选/计算类型

Fleet

(*ECSDeployAction*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

相应的 UI：配置 tab/Advanced -可选/计算舰队

Timeout

(*ECSDeployAction*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Environment

(*ECSDeployAction*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*ECSDeployAction*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*ECSDeployAction*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*ECSDeployAction*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*ECSDeployAction*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定部署到 Amazon ECS 操作用于访问 AWS的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
>
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
```

```
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用该角色时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

- 以下自定义信任策略：

> **Note**
>
> 如果需要，可以在此操作中使用 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Inputs

(*ECSDeployAction*/Inputs)

（可选）

Inputs 部分中定义了工作流运行期间 ECSDeployAction 所需的数据。

> **Note**
>
> 每个部署到 Amazon ECS 操作只能有一个输入（可以是源或构件）。

对应的 UI：输入选项卡

Sources

(*ECSDeployAction*/Inputs/Sources)

（如果您的任务定义文件存储在源存储库中，则为必需项）

如果您的任务定义文件存储在源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 `WorkflowSource`。

如果您的任务定义文件不包含在源存储库中，则必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*ECSDeployAction*/Inputs/Artifacts)

（如果您的任务定义文件存储在上一操作生成的输出构件中，则为必需项）

如果要部署的任务定义文件包含在上一操作生成的构件中，请在此处指定该构件。如果您的任务定义文件不包含在构件中，则必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Configuration

(*ECSDeployAction*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

region

(Configuration/region)

（必需）

指定您的 Amazon ECS 集群和服务所在的 AWS 区域。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。

对应的 UI：“配置”选项卡/区域

cluster

(*ECSDeployAction*/Configuration/cluster)

（必需）

指定现有 Amazon ECS 集群的名称。部署到 Amazon ECS 操作会将容器化应用程序作为任务部署到该集群中。有关 Amazon ECS 集群的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的集群。

对应的 UI：“配置”选项卡/集群

service

(*ECSDeployAction*/Configuration/service)

（必需）

指定将实例化任务定义文件的现有 Amazon ECS 服务的名称。此服务必须位于 `cluster` 字段中指定的集群下。有关 Amazon ECS 服务的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 服务。

对应的 UI：“配置”选项卡/服务

task-definition

(*ECSDeployAction*/Configuration/task-definition)

（必需）

指定现有任务定义文件的路径。如果文件位于源存储库中，则该路径相对于源存储库根文件夹。如果文件位于上一工作流操作生成的构件中，则该路径相对于构件根文件夹。有关任务定义文件的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的任务定义。

对应的 UI：“配置”选项卡/任务定义

force-new-deployment

(*ECSDeployAction*/Configuration/force-new-deployment)

（必需）

如果启用此项，则 Amazon ECS 服务无需更改服务定义即可启动新的部署。强制部署会导致服务停止所有当前正在运行的任务并启动新任务。有关强制重新部署更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的更新服务。

默认值：`false`

对应的 UI：“配置”选项卡/强制重新部署该服务

codedeploy-appspec

(*ECSDeployAction*/Configuration/codedeploy-appspec)

（如果您已将 Amazon ECS 服务配置为使用 blue/green 部署，则为必填项，否则省略）

指定现有 CodeDeploy 应用程序规范 (AppSpec) 文件的名称和路径。此文件必须位于 CodeCatalyst 源存储库的根目录中。有关 AppSpec 文件的更多信息，请参阅《AWS CodeDeploy 用户指南》中的CodeDeploy 应用程序规范 (AppSpec) 文件。

> **Note**
>
> 只有在您已将 Amazon ECS 服务配置为执行 blue/green 部署时，才提供 CodeDeploy 信息。对于滚动更新部署（默认），请省略 CodeDeploy 信息。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 部署类型。

> **Note**
>
> 这些CodeDeploy字段可能隐藏在可视化编辑器中。要显示该字段，请参阅为什么可视化编辑器中缺少 CodeDeploy 字段？。

对应的 UI：“配置” 选项卡/ CodeDeploy AppSpec

codedeploy-application

(*ECSDeployAction*/Configuration/codedeploy-application)

（如果包含 `codedeploy-appspec`，则为必需）

指定现有 CodeDeploy 应用程序的名称。有关 CodeDeploy 应用程序的更多信息，请参阅《AWS CodeDeploy 用户指南》 CodeDeploy中的使用应用程序。

相应的 UI：配置选项卡/应用程序 CodeDeploy

codedeploy-deployment-group

(*ECSDeployAction*/Configuration/codedeploy-deployment-group)

（如果包含 `codedeploy-appspec`，则为必需）

指定现有 CodeDeploy 部署组的名称。有关 CodeDeploy 部署组的更多信息，请参阅《AWS CodeDeploy 用户指南》 CodeDeploy中的使用部署组。

相应的 UI：“配置” 选项卡/ CodeDeploy 部署组

codedeploy-deployment-description

(*ECSDeployAction*/Configuration/codedeploy-deployment-description)

（可选）

指定此操作将创建的部署的描述。有关更多信息，请参阅《AWS CodeDeploy 用户指南》 CodeDeploy中的使用部署。

对应的 UI：“配置” 选项卡/ CodeDeploy 部署描述

使用工作流部署到 Amazon EKS

> **Tip**
>
> 有关说明如何使用部署到 Kubernetes 集群操作的教程，请参阅教程：将应用程序部署到 Amazon EKS。

本节介绍如何使用工作流程将容器化应用程序部署到 Kubernetes 集群中。 CodeCatalyst 为此，您必须将部署到 Kubernetes 集群操作添加到工作流。此操作使用一个或多个 Kubernetes 清单文件将您的应用程序部署到您在 Amazon Elastic Kubernetes Service（EKS）中设置的 Kubernetes 集群。有关示例清单，请参阅教程：将应用程序部署到 Amazon EKS 中的 部署 .yaml。

有关 Kubernetes 的更多信息，请参阅 Kubernetes 文档。

有关 Amazon EKS 的更多信息，请参阅《Amazon EKS 用户指南》中的什么是 Amazon EKS？

主题

- “部署到 Kubernetes 集群”操作的工作原理

- “部署到 Amazon EKS”操作使用的运行时映像
- 教程：将应用程序部署到 Amazon EKS
- 添加“部署到 Kubernetes 集群”操作
- “部署到 Kubernetes 集群”变量
- “部署到 Kubernetes 集群”操作 YAML

“部署到 Kubernetes 集群”操作的工作原理

部署到 Kubernetes 集群操作的工作原理如下：

1. 在运行时，该操作将 Kubernetes `kubectl` 实用程序安装到运行该操作的 CodeCatalyst计算机上。该操作将 `kubectl` 配置为指向您在配置该操作时提供的 Amazon EKS 集群。接下来，`kubectl` 实用工具是运行 `kubectl apply` 命令所必需的。
2. 该操作运行`kubectl apply -f my-manifest.yaml`命令，该命令执行中的说明，将您的应用程序作为一组容器和容器部署`my-manifest.yaml`到已配置的集群中。有关此命令的更多信息，请参阅《Kubernetes 参考文档》中的 kubectl 应用主题。

“部署到 Amazon EKS”操作使用的运行时映像

部署到 Amazon EKS 操作在 2022 年 11 月版映像上运行。有关更多信息，请参阅 活动映像。

教程：将应用程序部署到 Amazon EKS

在本教程中，您将学习如何使用亚马逊工作流程、Amazon EKS 和其他一些服务将容器化应用程序部署到亚马逊 Elastic Kubernetes S CodeCatalyst ervice 中。 AWS 部署的应用程序是简单“Hello, World!” 基于 Apache Web 服务器 Docker 映像构建的网站。本教程将引导您完成所需的准备工作（例如设置开发机器和 Amazon EKS 集群），然后介绍如何创建用于构建应用程序并将其部署到集群中的工作流。

初始部署完成后，本教程将指导您对应用程序源进行更改。此更改会构建新的 Docker 映像并将其与新修订信息一起推送到 Docker 映像存储库。之后，Docker 映像的新修订将部署到 Amazon EKS 中。

> **Tip**
>
> 您可以使用蓝图来执行完整的 Amazon EKS 设置，而不是按照本教程的说明操作。您将需要使用 EKS 应用程序部署蓝图。有关更多信息，请参阅 使用蓝图创建项目。

主题

先决条件

在开始本教程之前：

- 您需要一个带有关联 AWS 账户的 Amazon CodeCatalyst 空间。有关更多信息，请参阅 创建空间。
- 在您的空间中，您需要一个空项目，其名称为：

  ```
  codecatalyst-eks-project
  ```

  使用从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。
- 在你的项目中，你需要一个空的 CodeCatalyst 源代码库，名为：

  ```
  codecatalyst-eks-source-repository
  ```

  有关更多信息，请参阅 使用源存储库存储代码并协作处理代码 CodeCatalyst。
- 在你的项目中，你需要一个名为 CodeCatalyst CI/CD 环境（不是开发环境）：

  ```
  codecatalyst-eks-environment
  ```

  按如下方式配置此环境：

- 选择任何类型，例如非生产。
- 将您的 AWS 账户与之关联。
- 对于默认 IAM 角色，选择任何角色。稍后需要指定另一个角色。

有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。

步骤 1：设置开发机器

本教程中的第一步是使用将在本教程中使用的几种工具来配置开发机器。这些工具是：

- `eksctl` 实用工具 – 用于创建集群
- `kubectl` 实用工具 – `eksctl` 的先决条件
- 这 AWS CLI 个 — 也是前提条件 `eksctl`

如果有的话，可以在现有的开发计算机上安装这些工具，也可以使用基于云的 CodeCatalyst 开发环境。 CodeCatalyst 开发环境的好处是，它易于启动和关闭，并且与其他 CodeCatalyst 服务集成，因此您可以用更少的步骤完成本教程。

本教程假设你将使用 CodeCatalyst 开发环境。

以下说明描述了启动 CodeCatalyst 开发环境并使用所需工具对其进行配置的快速方法，但如果您需要详细说明，请参阅：

- 本指南中的创建开发环境。
- 《Amazon EKS 用户指南》中的安装 kubectl。
- 《Amazon EKS 用户指南》中的安装或升级 eksctl。
- 《AWS Command Line Interface User Guide》中的 Installing or updating the latest version of the AWS CLI。

启动开发环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的项目 `codecatalyst-eks-project`。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择您的源存储库的名称：`codecatalyst-eks-source-repository`。

5. 在顶部附近，选择创建开发环境，然后选择 AWS Cloud9 (在浏览器中)。
6. 确保选中在现有分支中工作和主要，然后选择创建。

   您的开发环境将在新的浏览器标签页中启动，并且您的存储库（codecatalyst-eks-source-repository）将克隆到其中。

安装和配置 kubectl

1. 在开发环境终端中，输入：

   ```
   curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
   ```

2. 输入：

   ```
   chmod +x ./kubectl
   ```

3. 输入：

   ```
   mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
   ```

4. 输入：

   ```
   echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
   ```

5. 输入：

   ```
   kubectl version --short --client
   ```

6. 检查是否显示了版本。

   您现在已安装 kubectl。

安装和配置 eksctl

> **Note**
>
> eksctl 不是硬性要求，因为您可以改用 kubectl。不过，eksctl 的优势在于能够自动执行大部分集群配置，因此建议在本教程中使用此工具。

1. 在开发环境终端中，输入：

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. 输入：

```
sudo cp /tmp/eksctl /usr/bin
```

3. 输入：

```
eksctl version
```

4. 检查是否显示了版本。

   您现在已安装 `eksctl`。

验证 AWS CLI 是否已安装

1. 在开发环境终端中，输入：

```
aws --version
```

2. 检查是否显示了版本以验证 AWS CLI 是否已安装。

   完成其余步骤，为 AWS CLI 其配置必要的访问权限 AWS。

要配置 AWS CLI

您必须 AWS CLI 使用访问密钥和会话令牌配置才能使其访问 AWS 服务。以下说明提供了配置密钥和令牌的快速方法，但如果您需要详细说明，请参阅《AWS Command Line Interface User Guide》中的 Configuring the AWS CLI。

1. 按如下方式创建 IAM Identity Center 用户：

   a. 登录 AWS 管理控制台 并打开 AWS IAM Identity Center 控制台，网址为https://console.aws.amazon.com/singlesignon/。

      （如果您之前从未登录过 IAM Identity Center，则可能需要选择启用。）

> **Note**
>
> 请务必使用与您的 CodeCatalyst空间 AWS 账户 相连的登录。您可以通过导航到您的空间并选择 AWS 账户选项卡来确认已连接哪个账户。有关更多信息，请参阅 创建空间。

b. 在导航窗格中，选择用户，然后选择添加用户。

c. 在用户名中，输入：

```
codecatalyst-eks-user
```

d. 在密码下，选择生成可与此用户共享的一次性密码。

e. 在电子邮件地址和确认电子邮件地址中，输入 IAM Identity Center 中不存在的电子邮件地址。

f. 在名字中，输入：

```
codecatalyst-eks-user
```

g. 在姓氏中，输入：

```
codecatalyst-eks-user
```

h. 在显示名称中，保留：

```
codecatalyst-eks-user codecatalyst-eks-user
```

i. 选择下一步。

j. 在将用户添加到组页面上，选择下一步。

k. 在查看并添加用户页面上，检查相应信息，然后选择添加用户。

这将显示一次性密码对话框。

l. 选择复制，然后将登录信息粘贴到文本文件中。登录信息由 AWS 访问门户 URL、用户名和一次性密码组成。

m. 选择关闭。

2. 按如下所示创建权限集：

a. 在导航窗格中，选择权限集，然后选择创建权限集。

b. 选择 “预定义权限集”，然后选择AdministratorAccess。该策略为所有 AWS 服务提供完全权限。

c. 选择下一步。

d. 在权限集名称中，移除 AdministratorAccess 并输入：

```
codecatalyst-eks-permission-set
```

e. 选择下一步。

f. 在查看和创建页面上，检查相应信息，然后选择创建。

3. 按如下方式将权限集分配给 codecatalyst-eks-user：

a. 在导航窗格中 AWS 账户，选择，然后选中您当前登录 AWS 账户 的旁边的复选框。

b. 选择分配用户或组。

c. 选择用户选项卡。

d. 选中 codecatalyst-eks-user 旁边的复选框。

e. 选择下一步。

f. 选中 codecatalyst-eks-permission-set 旁边的复选框。

g. 选择下一步。

h. 检查相应信息，然后选择提交。

现在，你已经codecatalyst-eks-permission-set将codecatalyst-eks-user和分配给你的 AWS 账户，将它们绑定在一起。

4. 按如下方式获取 codecatalyst-eks-user 的访问密钥和会话令牌：

a. 确保您有 AWS 访问门户 URL 以及的用户名和一次性密码codecatalyst-eks-user。您应在早些时候将此信息复制到文本编辑器中。

> **Note**
>
> 如果您没有这些信息，请转至 IAM Identity Center 中的 codecatalyst-eks-user 详细信息页面，选择重置密码和生成一次性密码 [...]，然后再次选择重置密码以在屏幕上显示信息。

b. 退出 AWS。

c. 将 AWS 访问门户 URL 粘贴到浏览器的地址栏中。

d. 使用以下项进行登录：

- 用户名：

```
codecatalyst-eks-user
```

- 密码：

one-time-password

e. 在设置新密码中，输入一个新密码，然后选择设置新密码。

屏幕上会出现一个 AWS 账户框。

f. 选择 AWS 账户，然后选择您为其分配了 codecatalyst-eks-user 用户和权限集的 AWS 账户 的名称。

g. 在 codecatalyst-eks-permission-set 旁边，选择命令行访问或以编程方式访问。

h. 复制页面中间的命令。其内容与以下内容类似：

```
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_SESSION_TOKEN="session-token"
```

... 其中*session-token*是一个长随机字符串。

5. 将访问密钥和会话令牌添加到 AWS CLI，如下所示：

a. 返回您的 CodeCatalyst 开发环境。

b. 在终端提示符处，粘贴您复制的命令。按 Enter。

现在，您已经为配置 AWS CLI 了访问密钥和会话令牌。现在，您可以使用 AWS CLI 来完成本教程要求的任务。

> **Important**
>
> 在本教程中，如果您在任何时候看到与以下内容类似的消息：
> `Unable to locate credentials. You can configure credentials by running "aws configure".`
> 或：
> `ExpiredToken: The security token included in the request is expired`

... 这是因为您的 AWS CLI 会话已过期。在此情况下，请不要运行 `aws configure` 命令。相反，请按照本过程的步骤 4（以 `Obtain codecatalyst-eks-user's access key and session token` 开头）中的说明操作来刷新会话。

步骤 2：创建 Amazon EKS 集群

在此部分中，您将在 Amazon EKS 中创建集群。以下说明介绍使用 `eksctl` 创建集群的快速方法，但如果您需要详细说明，请参阅：

- 《Amazon EKS 用户指南》中的开始使用 eksctl

  或者

- 《Amazon EKS 用户指南》中的开始使用控制台和 AWS CLI（本主题提供用于创建集群的 `kubectl` 指令）

> **Note**
>
> 与 Amazon EKS 的 CodeCatalyst 集成不支持@@ 私有集群。

开始之前

确保您已在开发机器上完成以下任务：

- 已安装 `eksctl` 实用工具。
- 已安装 `kubectl` 实用工具。
- 已安装 AWS CLI 并使用访问密钥和会话令牌对其进行配置。

有关如何完成这些任务的信息，请参阅步骤 1：设置开发机器。

创建集群

> **Important**
>
> 请勿使用 Amazon EKS 服务的用户界面来创建集群，因为这将无法正确配置集群。执行以下步骤来使用 `eksctl` 实用工具。

1. 转到您的开发环境。
2. 创建集群和节点：

```
eksctl create cluster --name codecatalyst-eks-cluster --region us-west-2
```

其中：

- *codecatalyst-eks-cluster*已替换为您要为集群命名的名称。
- *us-west-2*已替换为您所在的地区。

10-20 分钟后，将显示与以下内容类似的消息：

```
EKS cluster "codecatalyst-eks-cluster" in "us-west-2" region is ready
```

> **Note**
>
> 在 AWS 创建集群时，您将看到多条 `waiting for CloudFormation stack` 消息。这是预期行为。

3. 验证是否已成功创建集群：

```
kubectl cluster-info
```

您将看到与以下内容类似的消息，这表示已成功创建集群：

```
Kubernetes master is running at https://long-string.gr7.us-west-2.eks.amazonaws.com
CoreDNS is running at https://long-string.gr7.us-west-2.eks.amazonaws.com/api/v1/
namespaces/kube-system/services/kube-dns:dns/proxy
```

步骤 3：创建 Amazon ECR 映像存储库

在此部分中，您将在 Amazon Elastic Container Registry（Amazon ECR）中创建私有映像存储库。此存储库将存储教程的 Docker 映像。

有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

在 Amazon ECR 中创建映像存储库

1. 转到您的开发环境。
2. 在 Amazon ECR 中创建一个空存储库：

   ```
   aws ecr create-repository --repository-name codecatalyst-eks-image-repo
   ```

   *codecatalyst-eks-image-repo*替换为您想要为 Amazon ECR 存储库提供的名称。

   本教程假定您已将存储库命名为 codecatalyst-eks-image-repo。
3. 显示 Amazon ECR 存储库的详细信息：

   ```
   aws ecr describe-repositories \
         --repository-names codecatalyst-eks-image-repo
   ```
4. 记下 “repositoryUri”: 值，例如 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo。

   稍后在向工作流添加存储库时需要使用它。

步骤 4：添加源文件

在此部分中，您将应用程序源文件添加到源存储库（codecatalyst-eks-source-repository）。它们包括：

- index.html 文件 – 在浏览器中显示“Hello, World!” 消息。
- Dockerfile – 描述用于 Docker 映像的基本映像以及应用于该映像的 Docker 命令。
- deployment.yaml 文件 – 定义 Kubernetes 服务和部署的 Kubernetes 清单。

文件夹结构如下所示：

```
|— codecatalyst-eks-source-repository
   |— Kubernetes
      |— deployment.yaml
   |— public-html
   |  |— index.html
   |— Dockerfile
```

主题

- index.html
- Dockerfile
- 部署 .yaml

index.html

index.html 文件在浏览器中显示“Hello, World!” 消息。

添加 index.html 文件

1. 转到您的开发环境。
2. 在 codecatalyst-eks-source-repository 中，创建一个名为 public-html 的文件夹。
3. 在 /public-html 中，创建一个名为 index.html 的包含以下内容的文件：

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

4. 在终端提示符下，输入：

```
cd /projects/codecatalyst-eks-source-repository
```

5. 添加、提交和推送：

```
git add .
git commit -m "add public-html/index.html"
```

```
git push
```

这会将 index.html 添加到存储库中的 public-html 文件夹中。

Dockerfile

Dockerfile 描述要使用的基本 Docker 映像以及应用于该映像的 Docker 命令。有关 Dockerfile 的更多信息，请参阅 Dockerfile Reference。

此处指定的 Dockerfile 指示使用 Apache 2.4 基本映像（httpd）。它还包括用于将名为 index.html 的源文件复制到提供网页的 Apache 服务器上的文件夹中的指令。Dockerfile 中的 EXPOSE 指令告知 Docker 容器正在端口 80 上侦听。

添加 Dockerfile

1. 在 codecatalyst-eks-source-repository 中，创建一个名为 Dockerfile 的包含以下内容的文件：

   ```
   FROM httpd:2.4
   COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
   EXPOSE 80
   ```

   不要包含文件扩展名。

   > **Important**
   >
   > Dockerfile 必须位于存储库的根目录文件夹中。工作流的 Docker build 命令期望它在该位置。

2. 添加、提交和推送：

   ```
   git add .
   git commit -m "add Dockerfile"
   git push
   ```

   这会将 Dockerfile 添加到您的存储库中。

部署 .yaml

在此部分中，您将 deployment.yaml 文件添加到存储库。deployment.yaml 文件是一个 Kubernetes 清单，它定义两种要运行的 Kubernetes 资源类型或类别：“服务”和“部署”。

- “服务” 将负载均衡器部署到 Amazon EC2。负载均衡器为您提供面向 Internet 的公有 URL 和标准端口（端口 80），您可以使用它们浏览找到“Hello, World!” 应用程序的修订。
- “部署”部署三个容器组（pod），每个容器组（pod）将包含一个带“Hello, World!”的 Docker 容器 应用程序的修订。这三个容器组（pod）将部署到您创建集群时创建的节点上。

本教程中的清单较短；但清单可以包含任意数量的 Kubernetes 资源类型，例如容器组（pod）、作业、入口和网络策略。此外，如果您的部署复杂，则可以使用多个清单文件。

添加 deployment.yaml 文件

1. 在 codecatalyst-eks-source-repository 中，创建一个名为 Kubernetes 的文件夹。
2. 在 /Kubernetes 中，创建一个名为 deployment.yaml 的包含以下内容的文件：

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
    app: my-app
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    app: my-app
spec:
  replicas: 3
```

```
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: codecatalyst-eks-container
        # The $REPOSITORY_URI and $IMAGE_TAG placeholders will be replaced by
 actual values supplied by the build action in your workflow
        image: $REPOSITORY_URI:$IMAGE_TAG
        ports:
        - containerPort: 80
```

3. 添加、提交和推送：

```
git add .
git commit -m "add Kubernetes/deployment.yaml"
git push
```

这会将 deployment.yaml 文件添加到存储库中名为 Kubernetes 的文件夹中。

现在，您已添加所有源文件。

请花点时间仔细检查您的工作，确保已将所有文件置于正确的文件夹中。文件夹结构如下所示：

```
|— codecatalyst-eks-source-repository
   |— Kubernetes
      |— deployment.yaml
   |— public-html
   |  |— index.html
   |— Dockerfile
```

步骤 5：创建 AWS 角色

在本节中，您将创建 CodeCatalyst 工作流程运行所需的 AWS IAM 角色。这些角色是：

- 构建角色-授予 CodeCatalyst 构建操作（在工作流程中）访问您的 AWS 账户并写入 Amazon ECR 和 Amazon 的权限。 EC2

- 部署角色 — 向 CodeCatalyst 部署到 Kubernetes 集群操作（在工作流程中）授予访问您的账户 AWS 和 Amazon EKS 的权限。

有关 IAM 角色的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的 IAM 角色。

> **Note**
>
> 要节省时间，您可以创建一个名为 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色的角色，而不是前面列出的两个角色。有关更多信息，请参阅 为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色具有非常广泛的权限，这可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。本教程假定您创建的是前面列出的两个角色。

要创建构建和部署角色，请完成以下一系列过程。

1. 为两个角色创建信任策略

1. 转到您的开发环境。
2. 在 `Cloud9-long-string` 目录中，创建一个名为 `codecatalyst-eks-trust-policy.json` 的包含以下内容的文件：

2. 为构建角色创建构建策略

- 在 `Cloud9-long-string` 目录中，创建一个名为 `codecatalyst-eks-build-policy.json` 的包含以下内容的文件：

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

3. 为部署角色创建部署策略

- 在 Cloud9-*long-string* 目录中，创建一个名为 codecatalyst-eks-deploy-policy.json 的包含以下内容的文件：

  JSON

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

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

现在，您已将三个策略文档添加到开发环境。现在，您的目录结构应如下所示：

```
|— Cloud9-long-string
   |— .c9
   |— codecatalyst-eks-source-repository
      |— Kubernetes
      |— public-html
      |— Dockerfile
   codecatalyst-eks-build-policy.json
   codecatalyst-eks-deploy-policy.json
   codecatalyst-eks-trust-policy.json
```

4. 将生成策略添加到 AWS

1. 在开发环境终端中，输入：

   ```
   cd /projects
   ```

2. 输入：

   ```
   aws iam create-policy \
       --policy-name codecatalyst-eks-build-policy \
       --policy-document file://codecatalyst-eks-build-policy.json
   ```

3. 按 Enter 键。

4. 在命令输出中，记下 "arn": 值，例如 arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy。稍后您将需要此 ARN。

5. 将部署策略添加到 AWS

1. 输入：

```
aws iam create-policy \
    --policy-name codecatalyst-eks-deploy-policy \
    --policy-document file://codecatalyst-eks-deploy-policy.json
```

2. 按 Enter 键。
3. 在命令输出中，记下部署策略的 "arn": 值，例如 arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy。稍后您将需要此 ARN。

6. 创建构建角色

1. 输入：

```
aws iam create-role \
      --role-name codecatalyst-eks-build-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. 按 Enter 键。
3. 输入：

```
aws iam attach-role-policy \
      --role-name codecatalyst-eks-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy
```

其中*arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy*，替换为你之前提到的构建策略的 ARN。

4. 按 Enter 键。
5. 在终端提示符下，输入：

```
aws iam get-role \
      --role-name codecatalyst-eks-build-role
```

6. 按 Enter 键。
7. 请记下角色的 "Arn": 值，例如 arn:aws:iam::111122223333:role/codecatalyst-eks-build-role。稍后您将需要此 ARN。

7. 创建部署角色

1. 输入：

```
aws iam create-role \
      --role-name codecatalyst-eks-deploy-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. 按 Enter 键。
3. 输入：

```
aws iam attach-role-policy \
      --role-name codecatalyst-eks-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy
```

其中*arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy*，替换为你之前提到的部署策略的 ARN。

4. 按 Enter 键。
5. 输入：

```
aws iam get-role \
      --role-name codecatalyst-eks-deploy-role
```

6. 按 Enter 键。
7. 请记下角色的 "Arn": 值，例如 arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role。稍后您将需要此 ARN。

现在，您已经创建了生成和部署角色并记下了它们 ARNs。

第 6 步：将 AWS 角色添加到 CodeCatalyst

在此步骤中，将构建角色 (codecatalyst-eks-build-role) 和部署角色 (codecatalyst-eks-deploy-role) 添加到连接到空间 AWS 账户 的角色中。这将使这两个角色能够在工作流中使用。

向您的添加生成和部署角色 AWS 账户

1. 在 CodeCatalyst 控制台中，导航到您的空间。
2. 在顶部，选择设置。
3. 在导航窗格中，选择 AWS 账户。这将显示账户列表。

4. 在 Amazon CodeCatalyst 显示名称列中，复制您创建构建和部署角色的显示名称。 AWS 账户（它可能是一个数字。）您稍后在创建工作流时将需要此值。
5. 选择该显示名称。
6. 从管理控制台中选择 “ AWS 管理角色”。

   此时将出现 “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面。您可能需要登录才能访问该页面。
7. 选择添加您在 IAM 中创建的现有角色。

   这将显示一个下拉列表。该列表显示构建角色和部署角色，以及任何其他具有包含 `codecatalyst-runner.amazonaws.com` 和 `codecatalyst.amazonaws.com` 服务主体的信任策略的 IAM 角色。
8. 从下拉列表中，添加：

   - `codecatalyst-eks-build-role`
   - `codecatalyst-eks-deploy-role`

   > **Note**
   >
   > 如果您看到的是 `The security token included in the request is invalid`，则可能是因为您不具有适当的权限。要解决此问题，请使用您在创建 CodeCatalyst空间时使用的 AWS 账号退出并重新登录。 AWS
9. 返回 CodeCatalyst 控制台并刷新页面。

   现在，构建角色和部署角色应显示在 IAM 角色下。

   这些角色现在可以在工作 CodeCatalyst 流程中使用。

第 7 步：更新 ConfigMap

您必须将您在步骤 5：创建 AWS 角色中创建的部署角色添加到 Kubernetes `ConfigMap` 文件中，这样部署到 Kubernetes 集群操作（在工作流中）才能访问您的集群并与之交互。您可以使用 `eksctl` 或 `kubectl` 来执行此任务。

使用 eksctl 配置 Kubernetes 文件 ConfigMap

- 在开发环境终端中，输入：

```
eksctl create iamidentitymapping --cluster codecatalyst-eks-cluster --
arn arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role --group
 system:masters --username codecatalyst-eks-deploy-role --region us-west-2
```

其中：

- *codecatalyst-eks-cluster*已替换为 Amazon EKS 集群的集群名称。
- *arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role*将替换为您在中创建的部署角色的 ARN。步骤 5：创建 AWS 角色
- *codecatalyst-eks-deploy-role*（旁边--username）将替换为您在中创建的部署角色的名称步骤 5：创建 AWS 角色。

> **Note**
>
> 如果您决定不创建部署角色，请*codecatalyst-eks-deploy-role*替换为该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。有关该角色的更多信息，请参阅步骤 5：创建 AWS 角色。

- *us-west-2*已替换为您所在的地区。

有关此命令的详细信息，请参阅 Manage IAM users and roles。

这将显示一条与以下内容类似的消息：

```
2023-06-09 00:58:29 [#]  checking arn arn:aws:iam::111122223333:role/codecatalyst-
eks-deploy-role against entries in the auth ConfigMap
2023-06-09 00:58:29 [#]  adding identity "arn:aws:iam::111122223333:role/
codecatalyst-eks-deploy-role" to auth ConfigMap
```

使用 kubectl 配置 Kubernetes 文件 ConfigMap

1. 在开发环境终端中，输入：

```
kubectl edit configmap -n kube-system aws-auth
```

ConfigMap 文件出现在屏幕上。

2. 添加红色斜体文本：

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/eksctl-codecatalyst-eks-cluster-n-
NodeInstanceRole-16BC456ME6YR5
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - system:masters
      rolearn: arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role
      username: codecatalyst-eks-deploy-role
  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2023-06-08T19:04:39Z"
  managedFields:
  ...
```

其中：

- *arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role*将替换为您在中创建的部署角色的 ARN。步骤 5：创建 AWS 角色
- *codecatalyst-eks-deploy-role*（旁边username:）将替换为您在中创建的部署角色的名称步骤 5：创建 AWS 角色。

> **Note**
>
> 如果您决定不创建部署角色，请*codecatalyst-eks-deploy-role*替换为该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。有关该角色的更多信息，请参阅步骤 5：创建 AWS 角色。

有关详细信息，请参阅《Amazon EKS 用户指南》中的让 IAM 主体访问您的集群。

现在，您已向部署角色以及部署到 Amazon EKS 操作对您的 Kubernetes 集群的 `system:masters` 权限。

步骤 8：创建并运行工作流

在此步骤中，您将创建一个工作流来提取源文件，将源文件构建到 Docker 映像中，然后将该映像部署到 Amazon EKS 集群中的三个容器组（pod）中。

工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- 构建操作（`BuildBackend`）– 此操作在触发后会使用 Dockerfile 构建 Docker 映像并将该映像推送到 Amazon ECR。构建操作还使用正确的值更新 `deployment.yaml` 文件中的 `$REPOSITORY_URI` 和 `$IMAGE_TAG` 变量，然后创建此文件的输出构件并在 `Kubernetes` 文件夹中创建任何其他文件。在本教程中，`Kubernetes` 文件夹仅包含 `deployment.yaml` 文件，但您可以包含更多文件。此构件将用作接下来的部署操作的输入。

  有关构建操作的更多信息，请参阅使用工作流进行构建。
- 部署操作（`DeployToEKS`）– 构建操作完成后，部署操作将查找构建操作（`Manifests`）所生成的输出构件，并查找其中包含的 `deployment.yaml` 文件。之后，该操作按照 `deployment.yaml` 文件中的指令运行三个容器组（pod），每个容器组（pod）包含一个“Hello, World!” Docker 容器 – 在 Amazon EKS 集群中。

创建工作流

1. 转到 CodeCatalyst 控制台。
2. 导航到您的项目（`codecatalyst-eks-project`）。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择创建工作流。
5. 对于源存储库，选择 `codecatalyst-eks-source-repository`。
6. 对于分支，选择 `main`。
7. 选择创建。

8. 删除 YAML 示例代码。
9. 添加以下 YAML 代码以创建新的工作流定义文件：

> **Note**
>
> 有关工作流定义文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略这些部分，则必须确保您环境的默认 IAM 角色字段中指定的角色包含第 6 步：将 AWS 角色添加到 CodeCatalyst中描述的两个角色的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。

```
Name: codecatalyst-eks-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
```

```
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in deployment.yaml
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|g"
        - Run: cat Kubernetes/*
        # The output artifact will be a zip file that contains Kubernetes manifest
 files.
    Outputs:
      Artifacts:
        - Name: Manifests
          Files:
            - "Kubernetes/*"
  DeployToEKS:
    DependsOn:
      - BuildBackend
    Identifier: aws/kubernetes-deploy@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-deploy-role
    Inputs:
      Artifacts:
        - Manifests
    Configuration:
      Namespace: default
      Region: us-west-2
```

```
      Cluster: codecatalyst-eks-cluster
      Manifests: Kubernetes/
```

在上述代码中，进行如下替换：

- 的两个实例都*codecatalyst-eks-environment*与您在中创建的环境名称相同先决条件。
- 的两个实例都*codecatalyst-account-connection*带有您的账户连接的显示名称。显示名称可能是数字。有关更多信息，请参阅 第 6 步：将 AWS 角色添加到 CodeCatalyst。
- *codecatalyst-eks-build-role*使用您在中创建的构建角色的名称步骤 5：创建 AWS 角色。
- *111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo*（在Value:属性中），其中包含您在中创建的 Amazon ECR 存储库的 URI。步骤 3：创建 Amazon ECR 映像存储库
- *111122223333.dkr.ecr.us-west-2.amazonaws.com*（在Run: aws ecr命令中）使用 Amazon ECR 存储库的 URI，不带图像后缀 () /codecatalyst-eks-image-repo。
- *codecatalyst-eks-deploy-role*使用您在中创建的部署角色的名称步骤 5：创建 AWS 角色。
- 两个实例都*us-west-2*使用您的 AWS 地区代码。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。

> **Note**
>
> 如果您决定不创建生成和部署角色，请*codecatalyst-eks-deploy-role*使用CodeCatalystWorkflowDevelopmentRole-*spaceName*角色名称替换*codecatalyst-eks-build-role*和。有关该角色的更多信息，请参阅步骤 5：创建 AWS 角色。

10. （可选）选择验证，确保 YAML 代码在提交之前有效。
11. 选择提交。
12. 在提交工作流对话框中，输入以下内容：

    a. 对于提交消息，移除文本并输入：

    ```
    Add first workflow
    ```

    b. 对于存储库，选择 codecatalyst-eks-source-repository。

c. 对于分支名称，选择“主”。

d. 选择提交。

现在，您已创建工作流。由于在工作流顶部定义了触发器，因此工作流运行会自动启动。具体而言，当您将 `workflow.yaml` 文件提交（并推送）到源存储库时，触发器启动了工作流运行。

查看工作流运行进度

1. 在 CodeCatalyst 控制台的导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择您刚刚创建的工作流：`codecatalyst-eks-workflow`。
3. 选择BuildBackend查看构建进度。
4. 选择 DeployToEKS 以查看部署进度。

有关查看运行详细信息的更多信息，请参阅查看工作流运行状态和详细信息。

验证部署

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在底部左侧附近，选择负载均衡器。
3. 选择已在 Kubernetes 部署过程中创建的负载均衡器。如果您无法确定要选择哪个负载均衡器，请在标签选项卡下查找以下标签：
   - `kubernetes.io/service-name`
   - `kubernetes.io/cluster/ekstutorialcluster`
4. 选定正确的负载均衡器后，选择描述选项卡。
5. 将 DNS 名称值复制并粘贴到浏览器的地址栏中。

“Hello, World!” 网页将显示在浏览器中，这表示您已成功部署应用程序。

步骤 9：对源文件进行更改

在此部分中，您将对源存储库中的 `index.html` 文件进行更改。此更改会促使工作流构建新的 Docker 映像，使用提交 ID 对它进行标记，将它推送到 Amazon ECR，然后将它部署到 Amazon ECS。

更改 index.html

1. 转到您的开发环境。
2. 在终端提示符下，更改为您的源存储库：

```
cd /projects/codecatalyst-eks-source-repository
```

3. 拉取最新的工作流更改：

```
git pull
```

4. 打开 codecatalyst-eks-source-repository/public-html/index.html。
5. 在第 14 行上，将 Hello, World! 文本更改为 Tutorial complete!。
6. 添加、提交和推送：

```
git add .
git commit -m "update index.html title"
git push
```

工作流运行将自动启动。

7. （可选）输入：

```
git show HEAD
```

记下 index.html 更改的提交 ID。这将使用此提交 ID 标记将由您刚刚启动的工作流运行部署的 Docker 映像。

8. 查看部署进度：

a. 在 CodeCatalyst 控制台的导航窗格中，选择 C I/CD，然后选择工作流程。
b. 选择 codecatalyst-eks-workflow 以查看最新运行。
c. 选择 BuildBackend，然后选择 DeployToEKS 以查看工作流程的运行进度。

9. 验证是否已更新您的应用程序，如下所示：

a. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
b. 在底部左侧附近，选择负载均衡器。
c. 选择已在 Kubernetes 部署过程中创建的负载均衡器。
d. 将 DNS 名称值复制并粘贴到浏览器的地址栏中。

“Tutorial Complete!” 网页将显示在浏览器中，这表示您已成功部署新版本的应用程序。

10. （可选）在中 AWS，切换到 Amazon ECR 控制台，确认新 Docker 映像已使用此过程步骤 7 中的提交 ID 标记。

清理

您应清理环境，以免因本教程使用的存储和计算资源而产生不必要的费用。

清理

1. 删除集群：

   - 在开发环境终端中，输入：

     ```
     eksctl delete cluster --region=us-west-2 --name=codecatalyst-eks-cluster
     ```

     其中：

     - *us-west-2*已替换为您所在的地区。
     - *codecatalyst-eks-cluster*将替换为您创建的集群的名称。

     5-10 分钟后，集群和相关资源将被删除，包括但不限于 CloudFormation 堆栈、节点组（在 Amazon 中 EC2）和负载均衡器。

   > **Important**
   >
   > 如果该`eksctl delete cluster`命令不起作用，则可能需要刷新您的 AWS 凭据或`kubectl`凭据。如果您不确定要刷新哪些凭据，请先刷新 AWS 凭据。要刷新您的 AWS 凭证，请参阅如何修复“找不到凭证”和“ExpiredToken”错误？。要刷新您的 `kubectl` 凭证，请参阅如何修复“无法连接到服务器”错误？。

2. 在 AWS 控制台中，按如下方式进行清理：

   1. 在 Amazon ECR 中，删除 `codecatalyst-eks-image-repo`。
   2. 在 IAM Identity Center 中，删除：

      a. `codecatalyst-eks-user`

b. codecatalyst-eks-permission-set

3. 在 IAM 中，删除：
   - codecatalyst-eks-build-role
   - codecatalyst-eks-deploy-role
   - codecatalyst-eks-build-policy
   - codecatalyst-eks-deploy-policy

3. 在 CodeCatalyst 控制台中，按如下方式进行清理：
   1. 删除 codecatalyst-eks-workflow。
   2. 删除 codecatalyst-eks-environment。
   3. 删除 codecatalyst-eks-source-repository。
   4. 删除您的开发环境。
   5. 删除 codecatalyst-eks-project。

在本教程中，您学习了如何使用 CodeCatalyst 工作流程和部署到 Kubernetes 集群操作将应用程序部署到 Amazon EK S 服务。

添加“部署到 Kubernetes 集群”操作

按照以下说明操作，将部署到 Kubernetes 集群操作添加到工作流。

开始之前

在将部署到 Kubernetes 集群操作添加到工作流之前，您必须做好以下准备：

> **Tip**
> 要快速设置这些先决条件，请按照教程：将应用程序部署到 Amazon EKS 中的说明进行操作。

- Amazon EKS 中的 Kubernetes 集群。有关集群的信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 集群。
- 至少一个 Dockerfile，它描述如何将您的应用程序组装成 Docker 映像。有关 Dockerfile 的更多信息，请参阅 Dockerfile reference。
- 至少一个 Kubernetes 清单文件，该文件在 Kubernetes 文档中称作配置文件或配置。有关更多信息，请参阅 Kubernetes 文档中的管理资源。

- 一个 IAM 角色，它可让部署到 Kubernetes 集群操作访问您的 Amazon EKS 集群并与之交互。有关更多信息，请参阅“部署到 Kubernetes 集群”操作 YAML 中的Role主题。

  创建此角色后，您必须将它添加到：

  - 你的 Kubernetes 文件 ConfigMap 。要了解如何向 ConfigMap 文件添加角色，请参阅 A mazon EKS 用户指南中的启用 IAM 委托人访问您的集群。
  - CodeCatalyst。 要了解如何向添加 IAM 角色 CodeCatalyst，请参阅将 IAM 角色添加到账户连接。
- CodeCatalyst 空间、项目和环境。空间和环境都必须与要部署应用程序的 AWS 账户相关联。有关更多信息，请参阅 创建空间、在 Amazon 中创建一个空项目 CodeCatalyst 和 部署到 AWS 账户 和 VPCs。
- 支持的源存储库 CodeCatalyst。存储库将存储您的应用程序源文件、Dockerfile 和 Kubernetes 清单。有关更多信息，请参阅 使用源存储库存储代码并协作处理代码 CodeCatalyst。

Visual

使用可视化编辑器添加“部署到 Kubernetes 集群”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索部署到 Kubernetes 集群操作，然后执行下列操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择部署到 Kubernetes 集群。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“部署到 Kubernetes 集群”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加“部署到 Kubernetes 集群”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索部署到 Kubernetes 集群操作，然后执行下列操作之一：
   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择部署到 Kubernetes 集群。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 根据需求修改 YAML 代码中的属性。“部署到 Kubernetes 集群”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

“部署到 Kubernetes 集群”变量

部署到 Kubernetes 集群操作在运行时会生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 使用预定义变量。

键	值
cluster	在工作流运行期间部署到的 Amazon EKS 集群的 Amazon 资源名称（ARN）。 示例：`arn:aws:eks:us-west-2:111122223333:cluster/codecatalyst-eks-cluster`
deployment-platform	部署平台的名称。 硬编码为 `AWS:EKS`。
元数据	预留。与工作流运行期间部署的集群相关的 JSON 格式的元数据。
namespace	已将集群部署到的 Kubernetes 命名空间。 示例：`default`
resources	预留。与工作流运行期间部署的资源相关的 JSON 格式的元数据。
server	API 服务器端点的名称，可使用管理工具（例如 `kubectl`）通过此端点与集群进行通信。 有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 集群端点访问控制。 示例：`https://random-string .gr7.us-west-2.eks.amazonaws.com`

“部署到 Kubernetes 集群”操作 YAML

下面是部署到 Kubernetes 集群操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流部署到 Amazon EKS。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployToKubernetesCluster_nn:
    Identifier: aws/kubernetes-deploy@v1
    DependsOn:
      - build-action
    Compute:
        - Type: EC2 | Lambda
        - Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployToEKS
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - manifest-artifact
    Configuration:
```

```
      Namespace: namespace
      Region: us-east-1
      Cluster: eks-cluster
      Manifests: manifest-path
```

DeployToKubernetesCluster

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：DeployToKubernetesCluster_nn。

对应的 UI：“配置”选项卡/操作显示名称

Identifier

(*DeployToKubernetesCluster*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/kubernetes-deploy@v1。

对应的 UI：工作流图表/DeployToKubernetesCluster_nn/aws/kubernetes-deploy@v1 标签

DependsOn

(*DeployToKubernetesCluster*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*DeployToKubernetesCluster*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*DeployToKubernetesCluster*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：配置 tab/Advanced -可选/计算类型

Fleet

(*DeployToKubernetesCluster*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

相应的 UI：配置 tab/Advanced -可选/计算舰队

Timeout

(*DeployToKubernetesCluster*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Environment

(*DeployToKubernetesCluster*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*DeployToKubernetesCluster*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*DeployToKubernetesCluster*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*DeployToKubernetesCluster*/Environment/Connections/Name)

（可选）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*DeployToKubernetesCluster*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定部署到 Kubernetes 集群操作用于访问 AWS的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
>
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

JSON

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

> **Note**
>
> 第一次使用该角色时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

- 以下自定义信任策略：

确保将此角色添加到：

- 您的账户连接。要了解有关将 IAM 角色添加到账户连接的更多信息，请参阅将 IAM 角色添加到账户连接。
- 你的 Kubernetes ConfigMap。要了解有关向添加 IAM 角色的更多信息 ConfigMap，请参阅`eksctl`文档中的管理 IAM 用户和角色。

> **Tip**
>
> 另请参阅教程：将应用程序部署到 Amazon EKS，了解有关向账户连接添加 IAM 角色的说明，以及 ConfigMap。

> **Note**
>
> 如果需要，可以在此操作中使用 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***`spaceName`*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*`my-environment`*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Inputs

(*`DeployToKubernetesCluster`*/Inputs)

（如果包含 Connections，则为必需）

`Inputs` 部分中定义了工作流运行期间 `DeployToKubernetesCluster` 所需的数据。

> **Note**
>
> 每个部署到 Amazon EKS 操作只能有一个输入（可以是源或构件）。

对应的 UI：输入选项卡

Sources

(*DeployToKubernetesCluster*/Inputs/Sources)

（如果您的清单文件存储在源存储库中，则为必需）

如果您的 Kubernetes 清单文件存储在源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 `WorkflowSource`。

如果您的清单文件不包含在源存储库中，则必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*DeployToKubernetesCluster*/Inputs/Artifacts)

（如果您的清单文件存储在上一操作生成的输出构件中，则为必需）

如果一个或多个 Kubernetes 清单文件包含在上一操作生成的构件中，请在此处指定该构件。如果您的清单文件不包含在构件中，则必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Configuration

(*DeployToKubernetesCluster*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

Namespace

(*DeployToKubernetesCluster*/Configuration/Namespace)

（可选）

指定 Kubernetes 应用程序将部署到的 Kubernetes 命名空间。如果未将命名空间与集群结合使用，请使用 `default`。有关命名空间的更多信息，请参阅 Kubernetes 文档中的 Subdividing your cluster using Kubernetes namespaces。

如果省略命名空间，则使用值 `default`。

对应的 UI：“配置”选项卡/命名空间

Region

(*DeployToKubernetesCluster*/Configuration/Region)

（必需）

指定您的 Amazon EKS 集群和服务所在的 AWS 区域。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。

对应的 UI：“配置”选项卡/区域

Cluster

(*DeployToKubernetesCluster*/Configuration/Cluster)

（必需）

指定现有 Amazon EKS 集群的名称。部署到 Kubernetes 集群操作会将容器化应用程序部署到该集群中。有关 Amazon EKS 集群的更多信息，请参阅《Amazon EKS 用户指南》中的集群。

对应的 UI：“配置”选项卡/集群

Manifests

(*DeployToKubernetesCluster*/Configuration/Manifests)

（必需）

指定 YAML 格式的 Kubernetes 清单文件的路径，这些文件在 Kubernetes 文档中称作配置文件、config 文件，或简称为配置。

如果您使用了多个清单文件，请将它们置于一个文件夹中并引用该文件夹。Kubernetes 会按字母数字顺序处理清单文件，因此请务必在文件名前添加递增数字或字母，以便控制处理顺序。例如：

`00-namespace.yaml`

`01-deployment.yaml`

如果清单文件位于源存储库中，则该路径相对于源存储库根文件夹。如果该文件位于上一工作流操作生成的构件中，则该路径相对于构件根文件夹。

示例：

```
Manifests/

deployment.yaml

my-deployment.yml
```

请勿使用通配符（*）。

> **Note**
>
> 不支持 Helm 图表和 kustomization 文件。

有关清单文件的更多信息，请参阅 Kubernetes 文档中的 Organizing resource configurations。

对应的 UI：“配置”选项卡/清单

部署 CloudFormation 堆栈

本节介绍如何使用 CodeCatalyst 工作流程部署 AWS CloudFormation 堆栈。为此，您必须将 “部署 CloudFormation 堆栈” 操作添加到您的工作流程中。该操作会 AWS 根据您提供的模板将资源 CloudFormation 堆栈部署到中。模板可以是：

- CloudFormation 模板-有关更多信息，请参阅使用 CloudFormation 模板。
- AWS SAM 模板-有关更多信息，请参阅 AWS Serverless Application Model (AWS SAM) 规范。

> **Note**
>
> 要使用 AWS SAM 模板，必须先使用`sam package`操作打包 AWS SAM 应用程序。有关向您展示如何在 Amazon CodeCatalyst 工作流程中自动进行打包的教程，请参阅教程：部署无服务器应用程序。

如果堆栈已经存在，则该操作将运行 CloudFormation `CreateChangeSet`操作，然后运行该`ExecuteChangeSet`操作。之后，该操作会等待部署完更改，并根据结果自行标记为成功或失败。

如果您已经有包含要部署的资源的 CloudFormation 或 AWS SAM 模板，或者您计划使用 AWS SAM 和之类的工具在工作流程构建操作中自动生成一个模板，请使用 “部署 CloudFormation 堆栈” 操作AWS Cloud Development Kit (AWS CDK)。

你可以使用的模板没有任何限制，无论你能在其中创作什么， CloudFormation 或者 AWS SAM 你可以在 De ploy CloudFormation stack 操作中使用什么。

> **Tip**
>
> 有关向您展示如何使用 “部署 CloudFormation 堆栈” 操作部署无服务器应用程序的教程，请参阅教程：部署无服务器应用程序。

主题

- “部署 CloudFormation 堆栈” 操作使用的运行时镜像
- 教程：部署无服务器应用程序
- 添加 “部署 CloudFormation 堆栈” 操作
- 配置回滚
- '部署 CloudFormation 堆栈'变量
- '部署 CloudFormation 堆栈'动作 YAML

“部署 CloudFormation 堆栈” 操作使用的运行时镜像

部署 CloudFormation 堆栈操作在 2022 年 11 月的映像上运行。有关更多信息，请参阅 活动映像。

教程：部署无服务器应用程序

在本教程中，您将学习如何使用工作流程构建、测试和部署无服务器应用程序作为 CloudFormation 堆栈。

本教程中的应用程序是一个输出“Hello World”消息的简单 Web 应用程序。它由一个 AWS Lambda 函数和一个 Amazon API Gateway 组成，你可以使用 AWS Serverless Application Model (AWS SAM) 来构建它，后者是的扩展CloudFormation。

主题

- 先决条件
- 步骤 1：创建源存储库

先决条件

开始前的准备工作：

- 你需要一个带有关联 AWS 账户的 CodeCatalyst 空间。有关更多信息，请参阅 创建空间。
- 在您的空间中，您需要一个空项目，其名称为：

  ```
  codecatalyst-cfn-project
  ```

  使用从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。
- 在你的项目中，你需要一个 CodeCatalyst 名为：

  ```
  codecatalyst-cfn-environment
  ```

  按如下方式配置此环境：
  - 选择任何类型，例如非生产。
  - 将您的 AWS 账户与之关联。
  - 对于默认 IAM 角色，选择任何角色。稍后需要指定另一个角色。

  有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。

步骤 1：创建源存储库

在此步骤中，您将在中创建源存储库 CodeCatalyst。此存储库用于存储教程的源文件，例如 Lambda 函数文件。

有关源存储库的更多信息，请参阅创建源存储库。

创建源存储库

1. 在 CodeCatalyst导航窗格中，选择代码，然后选择源存储库。
2. 选择添加存储库，然后选择创建存储库。
3. 在存储库名称中，输入：

```
codecatalyst-cfn-source-repository
```

4. 选择创建。

现在，您已经创建了一个名为 `codecatalyst-cfn-source-repository` 的存储库。

步骤 2：创建 AWS 角色

在此步骤中，您将创建以下 AWS IAM 角色：

- 部署角色-授予 CodeCatalyst Deploy CloudFormation stack 操作访问您的 AWS 账户和 CloudFormation 服务的权限，您将在其中部署无服务器应用程序。部署 CloudFormation 堆栈操作是您的工作流程的一部分。
- 构建角色-授予 CodeCatalyst 构建操作访问您的 AWS 账户并写入存储无服务器应用程序包的 Amazon S3 的权限。构建操作是您的工作流的一部分。
- 堆栈角色- CloudFormation 授予读取和修改您稍后将提供的 AWS SAM 模板中指定的资源的权限。还授予权限 CloudWatch。

有关 IAM 角色的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的 IAM 角色。

> **Note**
>
> 要节省时间，您可以创建一个名为 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色的角色，而不是前面列出的三个角色。有关更多信息，请参阅 为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-spaceName` 角色具有非常广泛的权限，这可

能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。本教程假定您创建的是前面列出的三个角色。

> **Note**
>
> 还需要一个 Lambda 执行角色，但您无需立即创建此角色，因为当您在步骤 5 中运行工作流时，`sam-template.yml` 文件会为您创建它。

创建部署角色

1. 按如下步骤操作，为角色创建策略：

   a. 登录到 AWS。

   b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   c. 在导航窗格中，选择策略。

   d. 选择创建策略。

   e. 选择 JSON 选项卡。

   f. 删除现有代码。

   g. 粘贴以下代码：

   > **Note**
   >
   > 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
   >
   > ```
   > "Resource": "*"
   > ```

   h. 选择下一步：标签。

   i. 选择下一步：审核。

   j. 在名称中，输入：

   ```
   codecatalyst-deploy-policy
   ```

   k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建部署角色：

   a. 在导航窗格中，选择角色，然后选择创建角色。

   b. 选择自定义信任策略。

   c. 删除现有的自定义信任策略。

   d. 添加以下自定义信任策略：

   e. 选择下一步。

   f. 在权限策略中，搜索 codecatalyst-deploy-policy 并选中其复选框。

   g. 选择下一步。

   h. 对于角色名称，输入：

   ```
   codecatalyst-deploy-role
   ```

   i. 对于角色描述，输入：

   ```
   CodeCatalyst deploy role
   ```

   j. 选择创建角色。

   现在，您已创建一个具有信任策略和权限策略的部署角色。

3. 按如下步骤操作，获取部署角色 ARN：

   a. 在导航窗格中，选择角色。

   b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-deploy-role）。

   c. 从列表中选择该角色。

      此时将显示该角色的摘要页面。

   d. 在顶部，复制 ARN 值。

   现在，您已创建具有相应权限的部署角色并已获取其 ARN。

创建构建角色

1. 按如下步骤操作，为角色创建策略：

a. 登录到 AWS。

b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

c. 在导航窗格中，选择策略。

d. 选择创建策略。

e. 选择 JSON 选项卡。

f. 删除现有代码。

g. 粘贴以下代码：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-build-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst build role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

3. 按如下步骤操作，获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（`codecatalyst-build-role`）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。

现在，您已创建具有相应权限的构建角色并已获取其 ARN。

创建堆栈角色

1. AWS 使用要部署堆栈的账户登录。
2. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 按如下步骤操作，创建堆栈角色：

a. 在导航窗格中，选择角色。

b. 选择创建角色。

c. 选择 AWS 服务。

d. 在 “用例” 部分，CloudFormation从下拉列表中进行选择。

e. 选择 CloudFormation 单选按钮。

f. 选择底部的下一步。

g. 使用搜索框找到以下权限策略，然后选中它们对应的复选框。

> **Note**
>
> 如果您搜索一个策略，但该策略未显示，请务必选择清除筛选条件，然后重试。

- CloudWatchFullAccess
- AWS CloudFormationFullAccess
- IAMFull访问
- AWS Lambda_ FullAccess
- 亚马逊APIGateway管理员
- 亚马逊 3 FullAccess
- AmazonEC2ContainerRegistryFullAccess

第一个策略允许访问 CloudWatch 以在警报发生时启用堆栈回滚。

其余策略 AWS SAM 允许访问堆栈中将在本教程中部署的服务和资源。有关授予权限的更多信息，请参阅 AWS Serverless Application Model 开发人员指南的 权限。

h. 选择下一步。

i. 对于角色名称，输入：

```
codecatalyst-stack-role
```

j. 选择创建角色。

4. 按如下步骤操作，获取堆栈角色的 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（`codecatalyst-stack-role`）。

c. 从列表中选择该角色。

d. 在摘要部分中，复制 ARN 值。稍后您将需要用到它。

现在，您已创建具有相应权限的堆栈角色并已获取其 ARN。

步骤 3：将 AWS 角色添加到 CodeCatalyst

在此步骤中，您将构建角色 (codecatalyst-build-role) 和部署角色 (codecatalyst-deploy-role) 添加到空间中的 CodeCatalyst 账户连接。

> **Note**
>
> 您无需将堆栈角色（codecatalyst-stack-role）添加到连接。这是因为在部署角色之间 CodeCatalyst 已经建立连接之后 CloudFormation（不是 CodeCatalyst） AWS 使用堆栈角色。由于堆栈角色不用于获取 CodeCatalyst 访问权限 AWS，因此无需将其与账户连接相关联。

将构建角色和部署角色添加到账户连接

1. 在中 CodeCatalyst，导航到您的空间。
2. 选择 AWS accounts (账户)。此时将显示账户连接列表。
3. 选择代表您在其中创建构建和部署角色的 AWS 账户的账户连接。
4. 从管理控制台中选择 “ AWS 管理角色”。

   此时将出现 “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面。您可能需要登录才能访问该页面。
5. 选择添加您在 IAM 中创建的现有角色。

   这将显示一个下拉列表。该列表显示所有具有包含 codecatalyst-runner.amazonaws.com 和 codecatalyst.amazonaws.com 服务主体的信任策略的 IAM 角色。
6. 在该下拉列表中，选择 codecatalyst-build-role，然后选择添加角色。
7. 选择添加 IAM 角色，再选择添加您在 IAM 中创建的现有角色，然后在下拉列表中选择 codecatalyst-deploy-role。选择添加角色。

   现在，您已将构建角色和部署角色添加到您的空间。
8. 复制 A mazon CodeCatalyst 显示名称的值。您稍后在创建工作流时将需要此值。

步骤 4：创建 Amazon S3 存储桶

在此步骤中，您将创建可在其中存储无服务器应用程序的部署包 .zip 文件的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 在主窗格中，选择创建存储桶。
3. 对于存储桶名称，输入：

   ```
   codecatalyst-cfn-s3-bucket
   ```

4. 对于 AWS 区域，选择一个区域。本教程假设您选择了美国西部（俄勒冈州）us-west-2。有关 Amazon S3 支持的区域的信息，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service endpoints and quotas。
5. 在页面底部选择创建存储桶。

现在，您已经在美国西部（俄勒冈州）us-west-2 区域中创建了一个名为 **codecatalyst-cfn-s3-bucket** 的存储桶。

步骤 5：添加源文件

在此步骤中，您将向源存储库中添加多个应用程序 CodeCatalyst 源文件。`hello-world` 文件夹包含您将部署的应用程序文件。`tests` 文件夹包含单元测试。文件夹结构如下所示：

```
.
|— hello-world
|   |— tests
|      |— unit
|         |— test-handler.js
|   |— app.js
|— .npmignore
|— package.json
|— sam-template.yml
|— setup-sam.sh
```

.npmignore 文件

`.npmignore` 文件指明 npm 应从应用程序包中排除哪些文件和文件夹。在本教程中，npm 将排除 `tests` 文件夹，因为它不是应用程序的一部分。

添加 .npmignore 文件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目 codecatalyst-cfn-project。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 从源存储库列表中，选择您的存储库 codecatalyst-cfn-source-repository。
5. 在文件中，选择创建文件。
6. 对于文件名，输入：

```
.npmignore
```

7. 在文本框中，输入以下代码：

```
tests/*
```

8. 选择提交，然后再次选择提交。

现在，您已在存储库的根目录中创建名为 .npmignore 的文件。

package.json 文件

package.json 文件包含有关您的 Node 项目的重要元数据，例如项目名称、版本号、描述、依赖项以及描述如何运行应用程序并与之交互的其他详细信息。

本教程中的 package.json 包括依赖项列表和 test 脚本。测试脚本执行以下操作：

- 通过使用 mocha，测试脚本运行 hello-world/tests/unit/ 中指定的单元测试，并使用 xunit 报告器将结果写入 junit.xml 文件。
- 通过使用 Istanbul (nyc)，测试脚本使用 clover 报告器生成代码覆盖率报告（clover.xml）。有关更多信息，请参阅 Istanbul 文档中的 Using alternative reporters。

添加 package.json 文件

1. 在存储库中的文件中，选择创建文件。
2. 对于文件名，输入：

```
package.json
```

3. 在文本框中，输入以下代码：

```
{
```

```
  "name": "hello_world",
  "version": "1.0.0",
  "description": "hello world sample for NodeJS",
  "main": "app.js",
  "repository": "https://github.com/awslabs/aws-sam-cli/tree/develop/samcli/local/
init/templates/cookiecutter-aws-sam-hello-nodejs",
  "author": "SAM CLI",
  "license": "MIT",
  "dependencies": {
    "axios": "^0.21.1",
    "nyc": "^15.1.0"
  },
  "scripts": {
    "test": "nyc --reporter=clover mocha hello-world/tests/unit/ --reporter xunit
 --reporter-option output=junit.xml"
  },
  "devDependencies": {
    "aws-sdk": "^2.815.0",
    "chai": "^4.2.0",
    "mocha": "^8.2.1"
  }
}
```

4. 选择提交，然后再次选择提交。

   现在，您已将名为 `package.json` 的文件添加到存储库的根目录。

sam-template.yml 文件

`sam-template.yml` 文件包含有关部署 Lambda 函数和 API Gateway 并将它们一起配置的说明。它遵循AWS Serverless Application Model 模板规范，该规范扩展了 CloudFormation 模板规范。

在本教程中，您将使用 AWS SAM 模板而不是常规 CloudFormation 模板，因为 AWS SAM 提供了一种有用的:: Serverless AWS:: Function 资源类型。这种类型执行许多 behind-the-scenes配置，你通常必须写出这些配置才能使用基本 CloudFormation 语法。例如，`AWS::Serverless::Function` 创建一个 Lambda 函数、Lambda 执行角色和启动该函数的事件源映射。如果你想用 basic 来编写，你必须对所有这些代码进行编码 CloudFormation。

本教程使用的是预先编写的模板，您可以使用构建操作在工作流中生成一个模板。有关更多信息，请参阅 部署 CloudFormation 堆栈。

添加 sam-template.yml 文件

1. 在存储库中的文件中，选择创建文件。
2. 对于文件名，输入：

```
sam-template.yml
```

3. 在文本框中，输入以下代码：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  serverless-api

  Sample SAM Template for serverless-api

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # For details on this resource type,
 see https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello-world/
      Handler: app.lambdaHandler
      Runtime: nodejs12.x
      Events:
        HelloWorld:
          Type: Api # For details on this event source type, see
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

Outputs:
  # ServerlessRestApi is an implicit API created out of the events key under
 Serverless::Function
```

```
  # Find out about other implicit resources you can reference within AWS SAM at
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/
internals/generated_resources.rst#api
  HelloWorldApi:
    Description: "API Gateway endpoint URL for the Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.
${AWS::Region}.amazonaws.com/Prod/hello/"
  HelloWorldFunction:
    Description: "Hello World Lambda function ARN"
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: "Implicit Lambda execution role created for the Hello World
 function"
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

4. 选择提交，然后再次选择提交。

   现在，您已将名为 sam-template.yml 的文件添加到存储库的根文件夹下。

setup-sam.sh 文件

该setup-sam.sh文件包含下载和安装 AWS SAM CLI 实用程序的说明。工作流使用此实用工具来打包 hello-world 源。

添加 setup-sam.sh 文件

1. 在存储库中的文件中，选择创建文件。
2. 对于文件名，输入：

   setup-sam.sh

3. 在文本框中，输入以下代码：

```
#!/usr/bin/env bash
echo "Setting up sam"

yum install unzip -y

curl -LO https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
linux-x86_64.zip
unzip -qq aws-sam-cli-linux-x86_64.zip -d sam-installation-directory
```

```
./sam-installation-directory/install; export AWS_DEFAULT_REGION=us-west-2
```

在前面的代码中，*us-west-2*替换为您所在 AWS 的地区。

4. 选择提交，然后再次选择提交。

现在，您已将名为 setup-sam.sh 的文件添加到存储库的根目录。

app.js 文件

app.js 包含 Lambda 函数代码。在本教程中，代码返回文本 hello world。

添加 app.js 文件

1. 在存储库中的文件中，选择创建文件。
2. 对于文件名，输入：

```
hello-world/app.js
```

3. 在文本框中，输入以下代码：

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;

/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
    try {
```

```
        // const ret = await axios(url);
        response = {
            'statusCode': 200,
            'body': JSON.stringify({
                message: 'hello world',
                // location: ret.data.trim()
            })
        }
    } catch (err) {
        console.log(err);
        return err;
    }

    return response
};
```

4. 选择提交，然后再次选择提交。

现在，您已创建名为 hello-world 的文件夹和名为 app.js 的文件。

test-handler.js 文件

test-handler.js 文件包含 Lambda 函数的单元测试。

添加 test-handler.js 文件

1. 在存储库中的文件中，选择创建文件。
2. 对于文件名，输入：

hello-world/tests/unit/test-handler.js

3. 在文本框中，输入以下代码：

```
'use strict';

const app = require('../../app.js');
const chai = require('chai');
const expect = chai.expect;
var event, context;

describe('Tests index', function () {
    it('verifies successful response', async () => {
```

```
        const result = await app.lambdaHandler(event, context)

        expect(result).to.be.an('object');
        expect(result.statusCode).to.equal(200);
        expect(result.body).to.be.an('string');

        let response = JSON.parse(result.body);

        expect(response).to.be.an('object');
        expect(response.message).to.be.equal("hello world");
        // expect(response.location).to.be.an("string");
    });
});
```

4. 选择提交，然后再次选择提交。

   现在，您已将名为 `test-handler.js` 的文件添加到 `hello-world/tests/unit` 文件夹下。

现在，您已添加所有源文件。

请花点时间仔细检查您的工作，确保已将所有文件置于正确的文件夹中。文件夹结构如下所示：

```
.
|— hello-world
|  |— tests
|     |— unit
|        |— test-handler.js
|  |— app.js
|— .npmignore
|— README.md
|— package.json
|— sam-template.yml
|— setup-sam.sh
```

步骤 6：创建并运行工作流

在此步骤中，您将创建一个工作流来打包 Lambda 源代码并对其进行部署。工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

- 测试操作（Test）– 触发时，此操作将安装 Node package manager (npm)，然后运行 npm run test 命令。此命令告知 npm 运行 package.json 文件中定义的 test 脚本。test 脚本反过来运行单元测试并生成两个报告：测试报告（junit.xml）和代码覆盖率报告（clover.xml）。有关更多信息，请参阅 package.json 文件。

  接下来，测试操作将 XML 报告转换为 CodeCatalyst 报告，并将其显示在 CodeCatalyst 控制台中，位于测试操作的 “报告” 选项卡下。

  有关测试操作的更多信息，请参阅使用工作流进行测试。

- 构建操作 (BuildBackend)-测试操作完成后，构建操作将下载并安装 AWS SAM CLI，打包hello-world源代码，然后将包复制到您的 Amazon S3 存储桶（Lambda 服务期望的位置）。该操作还会输出一个名为的新 AWS SAM 模板文件，sam-template-packaged.yml并将其放置在名为的输出构件中buildArtifact。

  有关构建操作的更多信息，请参阅使用工作流进行构建。

- 部署操作 (DeployCloudFormationStack)-生成操作完成后，部署操作将查找生成操作 (buildArtifact) 生成的输出对象，在其中找到 AWS SAM 模板，然后运行该模板。该 AWS SAM 模板创建了一个用于部署无服务器应用程序的堆栈。

创建工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择创建工作流。
3. 对于源存储库，选择 codecatalyst-cfn-source-repository。
4. 对于分支，选择 main。
5. 选择创建。
6. 删除 YAML 示例代码。
7. 添加以下 YAML 代码：

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略这些部分，则必须确保您环境的默认 IAM 角色字段中指定的角色包含步骤 2：创建 AWS 角色中描述的两个角色的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。

```
Name: codecatalyst-cfn-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Test:
    Identifier: aws/managed-test@v1
    Inputs:
      Sources:
        - WorkflowSource
    Outputs:
      Reports:
        CoverageReport:
          Format: CLOVERXML
          IncludePaths:
            - "coverage/*"
        TestReport:
          Format: JUNITXML
          IncludePaths:
            - junit.xml
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
  BuildBackend:
    Identifier: aws/build@v1
    DependsOn:
      - Test
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-build-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: . ./setup-sam.sh
```

```
        - Run: sam package --template-file sam-template.yml --s3-
bucket codecatalyst-cfn-s3-bucket --output-template-file sam-template-packaged.yml
 --region us-west-2
    Outputs:
      Artifacts:
        - Name: buildArtifact
          Files:
            - "**/*"
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    DependsOn:
      - BuildBackend
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-deploy-role
    Inputs:
      Artifacts:
        - buildArtifact
      Sources: []
    Configuration:
      name: codecatalyst-cfn-stack
      region: us-west-2
      role-arn: arn:aws:iam::111122223333:role/StackRole
      template: ./sam-template-packaged.yml
      capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

在上述代码中，进行如下替换：

- 的两个实例均*codecatalyst-cfn-environment*以您的环境名称命名。
- 的两个实例都*codecatalyst-account-connection*带有您的账户连接的显示名称。显示名称可能是数字。有关更多信息，请参阅 步骤 3：将 AWS 角色添加到 CodeCatalyst。
- 将 *codecatalyst-build-role* 替换为您在步骤 2：创建 AWS 角色中创建的构建角色的名称。
- *codecatalyst-cfn-s3-bucket*使用您在中创建的 Amazon S3 存储桶的名称步骤 4：创建 Amazon S3 存储桶。
- *us-west-2*包含您的 Amazon S3 存储桶所在区域（第一个实例）和堆栈将部署的位置（第二个实例）的两个实例。这两个区域可能不同。本教程假定两个区域都设置为 us-west-2。有关

Amazon S3 和支持的区域的详细信息 CloudFormation，请参阅中的服务终端节点和配额AWS一般参考。

- 将 *codecatalyst-deploy-role* 替换为您在步骤 2：创建 AWS 角色中创建的部署角色的名称。
- *codecatalyst-cfn-environment*使用您在中创建的环境的名称先决条件。
- *arn:aws:iam::111122223333:role/StackRole*使用您在中创建的堆栈角色的 Amazon 资源名称 (ARN)。步骤 2：创建 AWS 角色

> **Note**
>
> 如果您决定不创建构建、部署和堆叠角色 *codecatalyst-build-rolecodecatalyst-deploy-role*，请使用角色的*arn:aws:iam::111122223333:role/StackRole*名称或 ARN 替换、和。CodeCatalystWorkflowDevelopmentRole-*spaceName*有关该角色的更多信息，请参阅步骤 2：创建 AWS 角色。

有关前面显示的代码中的属性的信息，请参阅'部署 CloudFormation 堆栈'动作 YAML。

8. （可选）选择验证，确保 YAML 代码在提交之前有效。
9. 选择提交。
10. 在提交工作流对话框中，输入以下内容：

    a. 对于工作流文件名，保留默认值 codecatalyst-cfn-workflow。

    b. 对于提交消息，输入：

    ```
    add initial workflow file
    ```

    c. 对于存储库，选择 codecatalyst-cfn-source-repository。

    d. 对于分支名称，选择主。

    e. 选择提交。

现在，您已创建工作流。由于在工作流顶部定义了触发器，因此工作流运行会自动启动。具体而言，当您将 codecatalyst-cfn-workflow.yaml 文件提交（并推送）到源存储库时，触发器启动了工作流运行。

查看正在运行的工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择您刚刚创建的工作流：`codecatalyst-cfn-workflow`。
3. 选择运行选项卡。
4. 在运行 ID 列中，选择运行 ID。
5. 选择测试以查看测试进度。
6. 选择BuildBackend查看构建进度。
7. 选择DeployCloudFormationStack查看部署进度。

   有关查看运行详细信息的更多信息，请参阅查看工作流运行状态和详细信息。
8. DeployCloudFormationStack操作完成后，请执行以下操作：
   - 如果工作流运行成功，请转到下一过程。
   - 如果工作流程在测试或BuildBackend操作中运行失败，请选择日志来解决问题。
   - 如果DeployCloudFormationStack操作的工作流程运行失败，请选择部署操作，然后选择摘要选项卡。滚动至CloudFormation 事件部分以查看详细的错误消息。如果发生了回滚， AWS 请在重新运行工作流程之前通过 CloudFormation 控制台删除`codecatalyst-cfn-stack`堆栈。

验证部署

1. 在部署成功后，从顶部附近的水平菜单栏中选择变量 (7)。（请勿在右侧窗格中选择变量。）
2. 旁边 HelloWorldApi，将 `https://` URL 粘贴到浏览器中。

   这将显示来自 Lambda 函数的 hello world JSON 消息，表示工作流已成功部署和配置 Lambda 函数以及 API Gateway。

   > **Tip**
   >
   > 您可以通过一些小配置在工作流程图中 CodeCatalyst 显示此 URL。有关更多信息，请参阅 在工作流图中显示应用程序 URL。

验证单元测试结果和代码覆盖率

1. 在工作流图中，选择测试，然后选择报告。

2. 选择TestReport查看单元测试结果，或者选择CoverageReport查看正在测试的文件的代码覆盖率详细信息，在本例中为app.js和test-handler.js。

验证已部署的资源

1. 登录 AWS 管理控制台 并打开 API Gateway 控制台，网址为https://console.aws.amazon.com/apigateway/。
2. 观察 AWS SAM 模板创建的 codecatalyst-cfn-stackAPI。API 名称来自工作流定义文件（codecatalyst-cfn-workflow.yaml）中的 Configuration/name 值。
3. 打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
4. 在导航窗格中，选择函数。
5. 选择您的 Lambda 函数 codecatalyst-cfn-stack-HelloWorldFunction-*string*。
6. 您可以查看 API Gateway 是如何成为该函数的触发器的。此集成是根据 AWS SAM AWS::Serverless::Function资源类型自动配置的。

步骤 7：进行更改

在此步骤中，您对 Lambda 源代码进行更改，然后提交它。此提交将启动新的工作流运行。此运行在蓝绿方案中部署新的 Lambda 函数，该方案使用 Lambda 控制台中指定的默认流量转移配置。

更改您的 Lambda 源

1. 在中 CodeCatalyst，导航到您的项目。
2. 在导航窗格中，选择代码，然后选择源存储库。
3. 选择您的源存储库 codecatalyst-cfn-source-repository。
4. 更改应用程序文件：
   a. 选择 hello-world 文件夹。
   b. 选择 app.js 文件。
   c. 选择编辑。
   d. 在第 23 行上，将 hello world 更改为 **Tutorial complete!**。
   e. 选择提交，然后再次选择提交。

   提交会促使启动工作流运行。此运行将失败，因为您尚未更新单元测试来反映名称更改。
5. 更新单元测试：

a. 选择 hello-world\tests\unit\test-handler.js。
b. 选择编辑。
c. 在第 19 行上，将 hello world 更改为 **Tutorial complete!**。
d. 选择提交，然后再次选择提交。

提交会促使启动另一个工作流运行。此运行将成功。

6. 在导航窗格中，选择 CI/CD，然后选择工作流。
7. 选择 codecatalyst-cfn-workflow，然后选择运行。
8. 选择最新运行的运行 ID。该运行应仍在进行中。
9. 选择 “测试” BuildBackend、“和” DeployCloudFormationStack以查看工作流程的运行进度。
10. 在工作流完成后，选择顶部附近的变量 (7)。
11. 旁边 HelloWorldApi，将 https:// URL 粘贴到浏览器中。

一条 Tutorial complete! 消息将显示在浏览器中，这表示您已成功部署新应用程序。

清理

清理本教程中使用的文件和服务以免被收取费用。

在 CodeCatalyst 控制台中进行清理

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 删除 codecatalyst-cfn-workflow。
3. 删除 codecatalyst-cfn-environment。
4. 删除 codecatalyst-cfn-source-repository。
5. 删除 codecatalyst-cfn-project。

要在里面清理干净 AWS 管理控制台

1. 清理一下 CloudFormation，如下所示：

a. 在 https://console.aws.amazon.com/cloudformation 上打开 CloudFormation 控制台。
b. 删除 codecatalyst-cfn-stack。

删除堆栈将从 API Gateway 和 Lambda 服务中移除所有教程资源。

2. 在 Amazon S3 中进行清理，如下所示：

   a. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
   b. 选择 codecatalyst-cfn-s3-bucket。
   c. 删除存储桶内容。
   d. 删除 存储桶。

3. 在 IAM 中进行清理，如下所示：

   a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 删除 codecatalyst-deploy-policy。
   c. 删除 codecatalyst-build-policy。
   d. 删除 codecatalyst-stack-policy。
   e. 删除 codecatalyst-deploy-role。
   f. 删除 codecatalyst-build-role。
   g. 删除 codecatalyst-stack-role。

在本教程中，您学习了如何使用 CodeCatalyst 工作流程和部署 CloudFormation 堆栈操作将无服务器应用程序部署为 CloudFormation 堆栈。

添加 “部署 CloudFormation 堆栈” 操作

按照以下说明将 “部署 CloudFormation 堆栈” 操作添加到您的工作流程中。

Visual

使用可视化编辑器添加 “部署 CloudFormation 堆栈” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。

8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署 CloudFormation 堆栈” 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 “部署 CloudFormation 堆栈”。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅'部署 CloudFormation 堆栈'动作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加 “部署 CloudFormation 堆栈” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署 CloudFormation 堆栈” 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 “部署 CloudFormation 堆栈”。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。

- 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 根据需求修改 YAML 代码中的属性。'部署 CloudFormation 堆栈'动作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

配置回滚

默认情况下，如果 Deploy CloudFormation 堆栈操作失败，它将导致堆栈回滚 CloudFormation 到上次已知的稳定状态。您可以更改行为，使回滚不仅在操作失败时发生，而且还会在出现指定的 Amazon CloudWatch 警报时发生。有关 CloudWatch 警报的更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。

您也可以更改默认行为，以便在操作失败时 CloudFormation 不会回滚堆栈。

按照以下说明操作来配置回滚。

> **Note**
>
> 您无法手动启动回滚。

Visual

开始前的准备工作

1. 确保您的工作流程包含有效的 De ploy CloudFormation 堆栈操作。有关更多信息，请参阅 部署 CloudFormation 堆栈。
2. 在 “部署 CloudFormation 堆栈” 操作的 “堆栈角色-可选” 字段中指定的角色中，确保包含CloudWatchFullAccess权限。有关创建此具有适当权限的角色的信息，请参阅步骤 2：创建 AWS 角色。

为 “部署 CloudFormation 堆栈” 操作配置回滚警报

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。

4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 选择您的部署 CloudFormation 堆栈操作。
8. 在详细信息窗格中，选择配置。
9. 在底部，展开高级。
10. 在 “监控警报” 下 ARNs，选择 “添加警报”。
11. 在以下字段中输入信息。

    - 警报 ARN

      指定要用作回滚触发器的亚马逊 CloudWatch 警报的亚马逊资源名称 (ARN)。例如 `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`。您可以拥有最多 5 个回滚触发器。

      > **Note**
      >
      > 如果您指定 CloudWatch 警报 ARN，则还需要配置其他权限才能使操作能够访问。CloudWatch有关更多信息，请参阅 配置回滚。

    - 监控时间

      指定 CloudFormation 监视指定警报的时间段，介于 0 到 180 分钟之间。在部署完所有堆栈资源后开始监控。如果警报发生在指定的监控时间内，则部署将失败，并回 CloudFormation 滚整个堆栈操作。

      默认值：0。 CloudFormation 仅在部署堆栈资源时监控警报，而不在部署堆栈资源之后监视警报。

YAML

为 “部署 CloudFormation 堆栈” 操作配置回滚触发器

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。

4. 选择包含部署 CloudFormation 堆栈操作的工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 YAML 代码中添加 `monitor-alarm-arns` 和 `monitor-timeout-in-minutes` 属性以添加回滚触发器。有关每个属性的说明，请参阅'部署 CloudFormation 堆栈'动作 YAML。
8. 在 De ploy CloudFormation 堆栈操作的`role-arn`属性中指定的角色中，确保包含CloudWatchFullAccess权限。有关创建此具有适当权限的角色的信息，请参阅步骤 2：创建 AWS 角色。

Visual

关闭 “部署 CloudFormation 堆栈” 操作的回滚功能

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择包含部署 CloudFormation 堆栈操作的工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 选择您的部署 CloudFormation 堆栈操作。
8. 在详细信息窗格中，选择配置。
9. 在底部，展开高级。
10. 启用禁用回滚。

YAML

关闭 “部署 CloudFormation 堆栈” 操作的回滚功能

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。

4. 选择包含部署 CloudFormation 堆栈操作的工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 YAML 代码中添加 `disable-rollback: 1` 属性以停止回滚。有关每个属性的说明，请参阅'部署 CloudFormation 堆栈'动作 YAML。

'部署 CloudFormation 堆栈'变量

部署 CloudFormation 堆栈操作在运行时生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 使用预定义变量。

键	值
deployment-platform	部署平台的名称。 硬编码为 `AWS:CloudFormation` 。
region	在工作流程运行期间部署到的的区域代码。AWS 区域 示例：`us-west-2`
stack-id	部署的堆栈的 Amazon 资源名称（ARN）。 示例：`arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cfn-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`

'部署 CloudFormation 堆栈'动作 YAML

以下是 “部署 CloudFormation 堆栈” 操作的 YAML 定义。要了解如何使用此操作，请参阅部署 CloudFormation 堆栈。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployRole
    Inputs:
      Sources:
        - source-name-1
      Artifacts:
        - CloudFormation-artifact
    Configuration:
      name: stack-name
      region: us-west-2
      template: template-path
      role-arn: arn:aws:iam::123456789012:role/StackRole
      capabilities: CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND
      parameter-overrides: KeyOne=ValueOne,KeyTwo=ValueTwo | path-to-JSON-file
      no-execute-changeset: 1|0
      fail-on-empty-changeset: 1|0
      disable-rollback: 1|0
```

```
      termination-protection: 1|0
      timeout-in-minutes: minutes
      notification-arns: arn:aws:sns:us-east-1:123456789012:MyTopic,arn:aws:sns:us-
east-1:123456789012:MyOtherTopic
      monitor-alarm-arns: arn:aws:cloudwatch::123456789012:alarm/
MyAlarm,arn:aws:cloudwatch::123456789012:alarm/MyOtherAlarm
      monitor-timeout-in-minutes: minutes
      tags: '[{"Key":"MyKey1","Value":"MyValue1"},{"Key":"MyKey2","Value":"MyValue2"}]'
```

DeployCloudFormationStack

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：DeployCloudFormationStack_nn。

对应的 UI：“配置”选项卡/操作显示名称

Identifier

(*DeployCloudFormationStack*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/cfn-deploy@v1。

对应的 UI：工作流图表/DeployCloudFormationStack_nn/aws/cfn-deploy@v1 标签

DependsOn

(*DeployCloudFormationStack*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*DeployCloudFormationStack*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*DeployCloudFormationStack*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：配置 tab/Advanced -可选/计算类型

Fleet

(*DeployCloudFormationStack*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示

例：Linux.x86-64.Large、Linux.x86-64.XLarge。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 Fleet，则默认值为 Linux.x86-64.Large。

相应的 UI：配置 tab/Advanced -可选/计算舰队

Timeout

(*DeployCloudFormationStack*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 (分钟) – 可选

Environment

(*DeployCloudFormationStack*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*DeployCloudFormationStack*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*DeployCloudFormationStack*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*DeployCloudFormationStack*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*DeployCloudFormationStack*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定 De ploy CloudFormation 堆栈操作用于访问的 IAM 角色的名称 AWS 和 CloudFormation 服务。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
>
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

> **Note**
>
> 第一次使用该角色时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

- 以下自定义信任策略：

> **Note**
>
> 如果需要，可以在此操作中使用 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解

CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Inputs

(*DeployCloudFormationStack*/Inputs)

（可选）

Inputs 部分中定义了工作流运行期间 DeployCloudFormationStack 所需的数据。

> **Note**
>
> 每个 Deplo y CloudFormation 堆栈操作最多允许四个输入（一个源和三个工件）。

如果您需要引用驻留在不同输入（例如源和构件）中的文件，则源输入是主输入，构件是辅助输入。辅助输入中对文件的引用采用特殊前缀，以与主输入中的文件区分开来。有关更多信息，请参阅 示例：引用多个构件中的文件。

对应的 UI：输入选项卡

Sources

(*DeployCloudFormationStack*/Inputs/Sources)

（如果您的 CloudFormation 或 AWS SAM 模板存储在源存储库中，则为必填项）

如果您的 CloudFormation 或 AWS SAM 模板存储在源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 WorkflowSource。

如果您的 CloudFormation 或 AWS SAM 模板不包含在源存储库中，则它必须位于其他操作生成的项目或 Amazon S3 存储桶中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*DeployCloudFormationStack*/Inputs/Artifacts)

（如果您的 CloudFormation 或 AWS SAM 模板存储在先前操作的输出对象中，则为必填项）

如果要部署的 CloudFormation 或 AWS SAM 模板包含在先前操作生成的对象中，请在此处指定该对象。如果您的 CloudFormation 模板不包含在项目中，则该模板必须位于您的源存储库或 Amazon S3 存储桶中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Configuration

(*DeployCloudFormationStack*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

name

(*DeployCloudFormationStack*/Configuration/name)

（必需）

为 De ploy CloudFormation 堆栈操作创建或更新的 CloudFormation 堆栈指定名称。

对应的 UI：“配置”选项卡/堆栈名称

region

(*DeployCloudFormationStack*/Configuration/region)

（必需）

指定堆栈将部署 AWS 区域 到哪里。有关区域代码的列表，请参阅区域端点。

对应的 UI：“配置”选项卡/堆栈区域

template

(*DeployCloudFormationStack*/Configuration/template)

（必需）

指定您的文件 CloudFormation 或 AWS SAM 模板文件的名称和路径。模板可以采用 JSON 或 YAML 格式，并且可以位于源存储库、上一操作生成的构件或 Amazon S3 存储桶中。如果模板文件位于源存储库或构件中，则路径相对于源存储库或构件根目录。如果模板位于 Amazon S3 存储桶中，则路径为模板的对象 URL 值。

示例：

```
./MyFolder/MyTemplate.json
```

```
MyFolder/MyTemplate.yml
```

```
https://MyBucket.s3.us-west-2.amazonaws.com/MyTemplate.yml
```

> **Note**
>
> 您可能需要在模板的文件路径中添加前缀，以指明要在哪个构件或源中查找它。有关更多信息，请参阅引用源存储库文件和在构件中引用文件。

对应的 UI：“配置”选项卡/模板

role-arn

(*DeployCloudFormationStack*/Configuration/role-arn)

（必需）

指定堆栈角色的亚马逊资源名称 (ARN)。 CloudFormation 使用此角色访问和修改堆栈中的资源。例如：`arn:aws:iam::123456789012:role/StackRole`。

确保堆栈角色包括：

- 一个或多个权限策略。这些策略取决于您在堆栈中拥有的资源。例如，如果您的堆栈包含一个 AWS Lambda 函数，则需要添加授予对 Lambda 访问权限的权限。如果您按照教程：部署无服务器应用程序中描述的教程进行操作，则其中包含一个名为创建堆栈角色的过程，该过程列出堆栈角色在部署典型无服务器应用程序堆栈时所需的权限。

  > **Warning**
  >
  > 将权限限制为 CloudFormation 服务访问堆栈中资源所需的权限。使用具有更广泛权限的角色可能会带来安全风险。

- 以下信任策略：

  JSON

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

（可选）将此角色与您的账户连接关联。要了解有关将 IAM 角色与账户连接关联的更多信息，请参阅将 IAM 角色添加到账户连接。如果您未将堆栈角色与账户连接关联，则堆栈角色将不会出现在可视化编辑器中的堆栈角色下拉列表中；但仍可以使用 YAML 编辑器在 `role-arn` 字段中指定角色 ARN。

> **Note**
>
> 如果需要，可以在此操作中使用 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解

CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：“配置”选项卡/堆栈角色 – 可选

capabilities

(*DeployCloudFormationStack*/Configuration/capabilities)

（必需）

指定允许 CloudFormation 创建特定堆栈所需的 IAM 功能列表。在大多数情况下，您可以将 `capabilities` 保留默认值 `CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND`。

如果您在部署 CloudFormation 堆栈操作的日志中看到 `##[error] requires capabilities: [`*capability-name*`]`，请参阅如何修复 IAM 功能错误？以获取有关如何修复该问题的信息。

有关 IAM 功能的更多信息，请参阅 IAM 用户指南中的在 CloudFormation 模板中确认 IAM 资源。

对应的 UI：“配置”选项卡/高级/功能

parameter-overrides

(*DeployCloudFormationStack*/Configuration/parameter-overrides)

（可选）

在您的 CloudFormation 或 AWS SAM 模板中指定没有默认值的参数，或者您要为其指定非默认值的参数。有关参数的更多信息，请参阅《AWS CloudFormation 用户指南》中的参数。

`parameter-overrides` 属性接受：

- 包含参数和值的 JSON 文件。
- 参数和值的逗号分隔列表。

指定 JSON 文件

1. 确保 JSON 文件使用下列语法之一：

```
{
  "Parameters": {
    "Param1": "Value1",
    "Param2": "Value2",
    ...
  }
}
```

或...

```
[
  {
     "ParameterKey": "Param1",
     "ParameterValue": "Value1"
  },
  ...
]
```

（还有其他语法，但在撰写本文时尚不支持这些语法。）CodeCatalyst 有关在 JSON 文件中指定 CloudFormation 参数的更多信息，请参阅AWS CLI 命令参考中支持的 JSON 语法。

2. 使用下列格式之一指定 JSON 文件的路径：

   - 如果 JSON 文件位于上一操作的输出构件中，请使用：

     `file:///artifacts/`*`current-action-name`*`/`*`output-artifact-name`*`/`*`path-to-json-file`*

     有关详细信息，请参阅示例 1。

   - 如果 JSON 文件位于源存储库中，请使用：

     `file:///sources/WorkflowSource/`*`path-to-json-file`*

     有关详细信息，请参阅示例 2。

     示例 1 – JSON 文件位于输出构件中

     ```
     ##My workflow YAML
     ...
     Actions:
       MyBuildAction:
     ```

```
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ParamArtifact
          Files:
            - params.json
    Configuration:
    ...
  MyDeployCFNStackAction:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: file:///artifacts/MyDeployCFNStackAction/
ParamArtifact/params.json
```

示例 2 – JSON 文件位于源存储库中的名为 `my/folder` 的文件夹中

```
##My workflow YAML
...
Actions:
  MyDeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      parameter-overrides: file:///sources/WorkflowSource/my/folder/params.json
```

使用参数的逗号分隔列表

- 使用以下格式在 `parameter-overrides` 属性中添加参数名称/值对：

  param-1=*value-1*,*param-2*=*value-2*

  例如，假设使用以下 CloudFormation 模板：

```
##My CloudFormation template

Description: My CloudFormation template

Parameters:
```

```
  InstanceType:
    Description: Defines the Amazon EC2 compute for the production server.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t3.medium

Resources:
...
```

...您可以设置 parameter-overrides 属性，如下所示：

```
##My workflow YAML
...
Actions:
...
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: InstanceType=t3.medium,UseVPC=true
```

> **Note**
>
> 您可以使用 undefined 作为值来指定不带相应值的参数名称。例如：
> parameter-overrides: MyParameter=undefined
> 其效果是，在堆栈更新期间， CloudFormation 使用现有参数值作为给定参数名称。

对应的 UI：

- “配置”选项卡/高级/参数覆盖
- 配置tab/Advanced/Parameter覆盖/ 使用文件指定覆盖
- 配置tab/Advanced/Parameter覆盖/使用值集指定覆盖

no-execute-changeset

(*DeployCloudFormationStack*/Configuration/no-execute-changeset)

（可选）

指定是否 CodeCatalyst 要创建 CloudFormation 更改集，然后在运行之前将其停止。这使您有机会在 CloudFormation 控制台中查看更改集。如果您确定更改集看起来不错，请禁用此选项，然后重新运行工作流程，这样 CodeCatalyst 就可以不停地创建和运行变更集。默认设置为不停地创建和运行更改集。有关更多信息，请参阅《AWS CLI 命令参考》中的 deplo CloudFormation y 参数。有关查看更改集的更多信息，请参阅《AWS CloudFormation 用户指南》中的查看更改集。

对应的 UI：“配置”选项卡/高级/没有执行更改集

fail-on-empty-changeset

(*DeployCloudFormationStack*/Configuration/fail-on-empty-changeset)

（可选）

指定如果 CloudFormation 更改集为空，是否 CodeCatalyst 要让 “部署 CloudFormation 堆栈” 操作失败。（如果更改集为空，则表示在最新部署期间未对堆栈进行任何更改。）默认设置为，在更改集为空时允许操作继续执行，并返回一条 UPDATE_COMPLETE 消息（即使未更新堆栈）。

有关此设置的更多信息，请参阅《AWS CLI 命令参考》中的 dep CloudFormation loy 参数。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的使用更改集更新堆栈。

对应的 UI：“配置”选项卡/高级/空更改集会失败

disable-rollback

(*DeployCloudFormationStack*/Configuration/disable-rollback)

（可选）

指定在堆栈部署失败时是否 CodeCatalyst 要回滚堆栈部署。回滚会使堆栈返回到上一个已知的稳定状态。默认设置为启用回滚。有关此设置的更多信息，请参阅《AWS CLI 命令参考》中的 dep CloudFormation loy 参数。

有关 Depl oy CloudFormation 堆栈操作如何处理回滚的更多信息，请参阅配置回滚。

有关回滚堆栈的更多信息，请参阅《AWS CloudFormation 用户指南》中的堆栈故障选项。

对于的 UI：“配置”选项卡/高级/禁用回滚

termination-protection

(*DeployCloudFormationStack*/Configuration/termination-protection)

（可选）

指定是否希望 Deploy CloudFormation 堆栈为其正在部署的堆栈添加终止保护。如果用户尝试删除已启用终止保护的堆栈，则删除操作会失败，并且堆栈及其状态将保持不变。默认设置为禁用终止保护。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的保护堆栈不被删除。

对应的 UI：“配置”选项卡/高级/终止保护

timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/timeout-in-minutes)

（可选）

指定在堆栈创建操作超时并将堆栈状态设置为之前 CloudFormation 应分配的时间（以CREATE_FAILED分钟为单位）。如果 CloudFormation 无法在分配的时间内创建整个堆栈，它将因超时而导致堆栈创建失败并回滚该堆栈。

默认情况下，堆栈创建从不超时。但是，单个资源可能会根据它们所实施服务的性质而具有自己的超时。例如，如果堆栈中的单个资源发生超时，则堆栈创建也会超时，即使尚未达到您为堆栈创建指定的超时也不例外。

对应的用户界面：配置选项卡/高级/超CloudFormation时

notification-arns

(*DeployCloudFormationStack*/Configuration/notification-arns)

（可选）

指定您 CodeCatalyst 要向其发送通知消息的 Amazon SNS 主题的 ARN。例如 `arn:aws:sns:us-east-1:111222333:MyTopic`。当 Deploy CloudFormation 堆栈操作运行时，与 CodeCatalyst CloudFormation 协调，为堆栈创建或更新过程中发生的每个 CloudFormation 事件发送一条通知。（事件显示在 CloudFormation 控制台堆栈的 “事件” 选项卡中。）最多可以指定五个主题。有关更多信息，请参阅 Amazon SNS 是什么？。

对应的用户界面：“配置” 选项卡/高级/ “通知” ARNs

monitor-alarm-arns

(*DeployCloudFormationStack*/Configuration/monitor-alarm-arns)

（可选）

指定要用作回滚触发器的亚马逊 CloudWatch 警报的亚马逊资源名称 (ARN)。例如 `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`。您可以拥有最多 5 个回滚触发器。

> **Note**
>
> 如果您指定 CloudWatch 警报 ARN，则还需要配置其他权限才能使操作能够访问。CloudWatch有关更多信息，请参阅 配置回滚。

对应的用户界面：“配置” 选项卡/高级/监控警报 ARNs

monitor-timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/monitor-timeout-in-minutes)

（可选）

指定 CloudFormation 监视指定警报的时间段，介于 0 到 180 分钟之间。在部署完所有堆栈资源后开始监控。如果警报发生在指定的监控时间内，则部署将失败，并回 CloudFormation 滚整个堆栈操作。

默认值：0。 CloudFormation 仅在部署堆栈资源时监控警报，而不在部署堆栈资源之后监视警报。

对应的 UI：“配置”选项卡/高级/监控时间

tags

(*DeployCloudFormationStack*/Configuration/tags)

（可选）

指定要附加到 CloudFormation 堆栈的标签。标签是任意键值对，可用于针对成本分配等目的来标识堆栈。有关标签是什么以及如何使用标签的更多信息，请参阅 Amazon EC2 用户指南中的为资源添加标签。有关在中添加标签的更多信息 CloudFormation，请参阅《AWS CloudFormation 用户指南》中的设置 CloudFormation 堆栈选项。

密钥可包含字母数字字符或空格，最多包含 127 个字符。值可包含字母数字字符或空格，最多包含 255 个字符。

您可以为每个堆栈添加最多 50 个唯一标签。

对应的 UI：“配置”选项卡/高级/标签

使用工作流程部署 AWS CDK 应用程序

本节介绍如何使用工作流程将 AWS Cloud Development Kit (AWS CDK) 应用程序部署到您的 AWS 账户。为此，您必须将 AWS CDK 部署操作添加到工作流中。AWS CDK 部署操作会合成您的 AWS Cloud Development Kit (AWS CDK) 应用程序并将其部署到。 AWS如果您的应用程序已存在于中 AWS，则操作会在必要时对其进行更新。

有关使用编写应用程序的一般信息 AWS CDK，请参阅什么是 AWS CDK？ 在《AWS Cloud Development Kit (AWS CDK) 开发人员指南》中。

主题

何时使用 “AWS CDK 部署” 操作

如果您使用开发了应用程序 AWS CDK，并且现在想要将其作为自动化持续集成和交付 (CI/CD) 工作流程的一部分自动部署，请使用此操作。例如，当有人合并与您的 AWS CDK 应用程序来源相关的拉取请求时，您可能希望自动部署您的 AWS CDK 应用程序。

“AWS CDK 部署” 操作的工作原理

AWS CDK 部署的工作方式如下：

1. 在运行时，如果您指定了 1.0.12 或更早版本的操作，则该操作会将最新的 CDK CLI（也称为 AWS CDK Tookit）下载到 CodeCatalyst 运行时环境映像。

   如果您指定了 1.0.13 或更高版本，则该操作会与特定版本的 CDK CLI 捆绑在一起，因此不会下载。

2. 该操作使用 CDK CLI 来运行 `cdk deploy` 命令。此命令将您的 AWS CDK 应用程序合成并部署到中。 AWS有关更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 AWS CDK Toolkit (cdk command) 主题。

“AWS CDK 部署”操作使用的 CDK CLI 版本

下表显示了不同版本的 AWS CDK 部署操作默认使用哪个版本的 CDK CLI。

> **Note**
>
> 您也许能够覆盖默认值。有关更多信息，请参阅“AWS CDK 部署”操作 YAML 中的 CdkCliVersion。

“AWS CDK 部署”操作版本	AWS CDK CLI 版本
1.0.0 – 1.0.12	最新
1.0.13 或更高版本	2.99.1

“AWS CDK 部署” 操作使用的运行时镜像

下表显示了 CodeCatalyst 用于运行不同版本的AWS CDK 部署操作的运行时环境映像。这些映像包括不同的预安装工具集。有关更多信息，请参阅 活动映像。

> **Note**
>
> 我们建议将您的 AWS CDK 部署操作升级到 2.x 版，从而利用 2024 年 3 月版映像中提供的最新工具。要升级操作，请在工作流定义文件中将其 `Identifier` 属性设置为 `aws/cdk-deploy@v2`。有关更多信息，请参阅 “AWS CDK 部署”操作 YAML。

“AWS CDK 部署”操作版本	运行时环境映像
1.x	2022 年 11 月版映像
2.x	2024 年 3 月版映像

操作可以部署多少个堆栈？

AWS CDK 部署只能部署单个堆栈。如果您的 AWS CDK 应用程序由多个堆栈组成，则必须创建包含嵌套堆栈的父堆栈，然后使用此操作部署父堆栈。

示例：部署 AWS CDK 应用程序

以下示例工作流包括 AWS CDK 部署操作以及 AWS CDK 引导操作。工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。此存储库包含您的 AWS CDK 应用程序。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- AWS CDK 引导操作 (CDKBootstrap)-触发后，该操作会将CDKToolkit引导堆栈部署到。 AWS如果环境中已存在 CDKToolkit 堆栈，则将在必要时对堆栈进行升级；否则，不会发生任何操作，并且该操作将标记为成功。
- AWS CDK 部署操作 (AWS CDK Deploy)-完成AWS CDK 引导操作后，AWS CDK 部署操作会将您的 AWS CDK 应用程序代码合成到模板中，并将 CloudFormation 模板中定义的堆栈部署到中。 AWS

> **Note**
>
> 以下工作流示例仅用于说明目的，如果不执行附加配置，则无法运行。

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略这些部分，则必须确保在您的环境的默认 IAM 角色字段中指定的角色包含 AWS CDK 引导和 AWS CDK 部署操作所需的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。有关 AWS CDK 引导和 AWS CDK 部署操作所需的权限和信任策略的更多信息，请参阅“AWS CDK 引导”操作 YAML 和“AWS CDK 部署”操作 YAML 中 Role 属性的说明。

```
Name: codecatalyst-cdk-deploy-workflow
SchemaVersion: 1.0

Triggers:
```

```
  - Type: PUSH
    Branches:
      - main
Actions:
  CDKBootstrap:
    Identifier: aws/cdk-bootstrap@v2
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-bootstrap-role
    Configuration:
      Region: us-west-2

  CDKDeploy:
    Identifier: aws/cdk-deploy@v2
    DependsOn:
      - CDKBootstrap
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-deploy-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: my-app-stack
      Region: us-west-2
```

添加 “AWS CDK 部署” 操作

按照以下说明，将 AWS CDK 部署操作添加到您的工作流中。

开始之前

在将 AWS CDK 部署操作添加到工作流之前，请先完成以下任务：

1. 准备好 AWS CDK 应用程序。您可以使用 AWS CDK v1 或 v2，使用支持的任何编程语言编写 AWS CDK 应用程序。 AWS CDK确保您的 AWS CDK 应用程序文件在以下位置可用：

- CodeCatalyst 源存储库，或
- 由另一个工作流程操作生成的 CodeCatalyst 输出对象

2. 引导您的 AWS 环境。要进行引导，您可以：

- 使用《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 How to bootstrap 介绍的方法之一。
- 使用 AWS CDK 引导操作。您可以在与 AWS CDK 部署相同的工作流中添加此操作，也可以在不同的工作流中添加。这是为了确保引导操作在运行 AWS CDK 部署操作之前至少运行一次，以便必要的资源准备就位。有关AWS CDK 引导操作的更多信息，请参阅使用工作流程引导 AWS CDK 应用程序。

  有关引导的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Bootstrapping。

Visual

使用可视化编辑器添加AWS CDK “部署” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS CDK 部署操作，然后执行以下操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 AWS CDK 部署。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“AWS CDK 部署”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

> **Note**
>
> 如果您的 AWS CDK 部署操作失败并出现 `npm install` 错误，请参阅如何修复“npm install”错误？，了解有关如何修复错误的信息。

YAML

使用 YAML 编辑器添加AWS CDK “部署” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS CDK 部署操作，然后执行以下操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 AWS CDK 部署。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择下载以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 根据需求修改 YAML 代码中的属性。“AWS CDK 部署”操作 YAML中提供了每个可用属性的解释。

11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

> **Note**
>
> 如果您的 AWS CDK 部署操作失败并出现 `npm install` 错误，请参阅[如何修复“npm install”错误？](#)，了解有关如何修复错误的信息。

“AWS CDK 部署”变量

AWS CDK 部署操作在运行时会生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 [使用预定义变量](#)。

键	值
stack-id	在工作流程运行期间部署到的 AWS CDK 应用程序堆栈的 Amazon 资源名称 (ARN)。 示例：`arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-app-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
deployment-platform	部署平台的名称。 硬编码为 `AWS:CloudFormation` 。
region	在工作流程运行期间部署到的的区域代码。AWS 区域 示例：`us-west-2`
SKIP-DEPLOYMENT	值为`true`表示在工作流程运行期间跳过了 AWS CDK 应用程序堆栈的部署。如果自上次部署以来，堆栈没有发生变化，则将跳过堆栈部署。

键	值
	只有当该变量的值为 `true` 时，才会生成该变量。 硬编码为 `true`。
CloudFormation variables	除了生成前面列出的变量外，AWS CDK 部署操作还将CloudFormation输出变量作为工作流变量公开，以便在后续的工作流操作中使用。默认情况下，该操作仅公开它找到的前四个（或更少）CloudFormation变量。要确定哪些变量已公开，请运行一次 AWS CDK 部署操作，然后在运行详细信息页面的变量选项卡中查看。如果变量选项卡上列出的变量不是您需要的，则可以使用 `CfnOutputVariables` YAML 属性配置其他变量。有关更多信息，请参阅“AWS CDK 部署”操作 YAML 中 CfnOutputVariables 属性的描述。

“AWS CDK 部署”操作 YAML

下面是 AWS CDK 部署操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流程部署 AWS CDK 应用程序。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
```

```
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKDeploy_nn:
    Identifier: aws/cdk-deploy@v2
    DependsOn:
      - CDKBootstrap
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_artifact
          Files:
            - "cdk.out/**/*"
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      StackName: my-cdk-stack
      Region: us-west-2
      Tags: '{"key1": "value1", "key2": "value2"}'
      Context: '{"key1": "value1", "key2": "value2"}'
      CdkCliVersion: version
      CdkRootPath: directory-containing-cdk.json-file
      CfnOutputVariables: '["CnfOutputKey1","CfnOutputKey2","CfnOutputKey3"]'
      CloudAssemblyRootPath: path-to-cdk.out
```

CDKDeploy

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：CDKDeploy_nn。

对应的 UI：“配置”选项卡/操作名称

Identifier

(*CDKDeploy*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅 指定要使用的操作版本。

> **Note**
>
> 指定 aws/cdk-deploy@v2 可以让操作在 2024 年 3 月版映像上运行，其中包括较新的工具，例如 Node.js 18。指定 aws/cdk-deploy@v1 可以让操作在 2022 年 11 月版映像上运行，其中包括较旧的工具，例如 Node.js 16。

默认值：aws/cdk-deploy@v2。

对应的 UI：工作流图表/CDKDeploy_nn/aws/cdk-deploy@v2 标签

DependsOn

(*CDKDeploy*/DependsOn)

（可选）

指定必须成功运行才能使 AWS CDK 部署操作运行的操作或操作组。我们建议在 DependsOn 属性中指定 AWS CDK 引导操作，如下所示：

```
CDKDeploy:
  Identifier: aws/cdk-deploy@v2
  DependsOn:
    - CDKBootstrap
```

> **Note**
>
> 引导是部署应用程序的必备先决条件。 AWS CDK 如果您未在工作流中包含 AWS CDK 引导操作，则在运行 AWS CDK 部署操作之前，必须找到另一种用于部署 AWS CDK 引导堆栈的方法。有关更多信息，请参阅使用工作流程部署 AWS CDK 应用程序中的添加 “AWS CDK 部署” 操作。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*CDKDeploy*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*CDKDeploy*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：配置 tab/Advanced -可选/计算类型

Fleet

(*CDKDeploy*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

相应的 UI：配置 tab/Advanced -可选/计算舰队

Timeout

(*CDKDeploy*/Timeout)

（必需）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Inputs

(*CDKDeploy*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间 `CDKDeploy` 所需的数据。

> **Note**
>
> 每个 AWS CDK 部署操作只能有一个输入（可以是源或构件）。

对应的 UI：输入选项卡

Sources

(*CDKDeploy*/Inputs/Sources)

（如果您要部署的 AWS CDK 应用程序存储在源存储库中，则为必填项）

如果您的 AWS CDK 应用程序存储在源存储库中，请指定该源存储库的标签。在启动部署过程之前，AWS CDK 部署操作会合成此存储库中的应用程序。目前，唯一支持的标签是 `WorkflowSource`。

如果您的 AWS CDK 应用程序不包含在源存储库中，则它必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*CDKDeploy*/Inputs/Artifacts)

（如果您要部署的 AWS CDK 应用程序存储在先前操作的输出项目中，则为必填项）

如果您的 AWS CDK 应用程序包含在先前操作生成的构件中，请在此处指定该构件。在开始AWS CDK 部署过程之前，部署操作会将指定构件中的应用程序合成到 CloudFormation模板中。如果您的 AWS CDK 应用程序不包含在工件中，则它必须位于您的源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输入”选项卡/构件 – 可选

Outputs

(*CDKDeploy*/Outputs)

（可选）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts - output

(*CDKDeploy*/Outputs/Artifacts

（可选）

指定操作生成的构件。您可以在其他操作中将这些构件作为输入来引用。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件

Name

(*CDKDeploy*/Outputs/Artifacts/Name)

（如果包含 Artifacts - output，则为必需）

指定将包含在运行时由AWS CDK 部署操作合成的 CloudFormation 模板的对象的名称。默认值为 `cdk_artifact`。如果您未指定构件，则该操作会合成模板，但不会将模板保存在构件中。考虑将合成的模板保存在构件中，以便保留其记录，供测试或故障排除之用。

对应的用户界面：输出tab/Artifacts/Add构件/构建构件名称

Files

(*CDKDeploy*/Outputs/Artifacts/Files)

（如果包含 Artifacts - output，则为必需）

指定要包含在构件中的文件。您`"cdk.out/**/*"`必须指定包含 AWS CDK 应用程序的合成 CloudFormation 模板。

> **Note**
>
> `cdk.out` 是保存已合成文件的默认目录。如果您指定了输出目录，而不是 `cdk.json` 文件中的 `cdk.out`，请在此处指定该目录，而不是 `cdk.out`。

相应的 UI：输出tab/Artifacts/Add构件/编译生成的文件

Environment

(*CDKDeploy*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*CDKDeploy*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*CDKDeploy*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*CDKDeploy*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*CDKDeploy*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定账户连接的名称。

指定AWS CDK 部署操作用于访问 AWS 和部署 AWS CDK 应用程序堆栈的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

JSON

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

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111122223333:role/cdk-*"
        }
    ]
}
```

- 以下自定义信任策略：

> **Note**
>
> 如果需要，可以在此操作中使用 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Configuration

(*CDKDeploy*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

StackName

(*CDKDeploy*/Configuration/StackName)

（必需）

AWS CDK 应用程序堆栈的名称，显示在 AWS CDK 应用程序目录的入口点文件中。bin以下示例显示了 TypeScript入口点文件的内容，其中突出显示了堆栈名称。*red italics*如果您的入口点文件使用的是不同语言，该文件看起来会很相似。

```
import * as cdk from 'aws-cdk-lib';
import { CdkWorksopTypescriptStack } from '../lib/cdk_workshop_typescript-stack';

const app = new cdk.App();
new CdkWorkshopTypescriptStack(app, 'CdkWorkshopTypescriptStack');
```

您只能指定一个堆栈。

> **Tip**
>
> 如果您具有多个堆栈，则可以创建带有嵌套堆栈的父堆栈。然后，您可以在此操作中指定父堆栈，以便部署所有堆栈。

对应的 UI：“配置”选项卡/堆栈名称

Region

(*CDKDeploy*/Configuration/Region)

（可选）

指定要 AWS 区域 将 AWS CDK 应用程序堆栈部署到哪里。有关区域代码的列表，请参阅区域端点。

如果您未指定区域，则AWS CDK 部署操作将部署到您的 AWS CDK 代码中指定的区域。有关更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Environments。

对应的 UI：“配置”选项卡/区域

Tags

(*CDKDeploy*/Configuration/Tags)

（可选）

指定要应用于 AWS CDK 应用程序堆栈中 AWS 资源的标签。标签应用于堆栈自身以及堆栈中的各个资源。有关标记的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Tagging。

对应的用户界面：配置 tab/Advanced -可选/标签

Context

(*CDKDeploy*/Configuration/Context)

（可选）

以键值对的形式指定要与 AWS CDK 应用程序堆栈关联的上下文。有关上下文的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Runtime contexts。

相应的 UI：配置 tab/Advanced -可选/上下文

CdkCliVersion

(*CDKDeploy*/Configuration/CdkCliVersion)

（可选）

此属性适用于 AWS CDK 部署操作的 1.0.13 或更高版本，以及 AWS CDK 引导操作的 1.0.8 或更高版本。

指定下列项之一：

- 您希望此操作使用的 AWS Cloud Development Kit (AWS CDK) 命令行界面 (CLI)（也称为 AWS CDK 工具包）的完整版本。示例：`2.102.1`。在构建和部署您的应用程序时，请考虑指定完整版本，从而确保一致性和稳定性。

  Or

- `latest`。请考虑指定 `latest`，从而利用 CDK CLI 的最新功能和修复。

该操作会将指定版本（或最新版本）的 AWS CDK CLI 下载到 CodeCatalyst 构建映像，然后使用此版本运行部署 CDK 应用程序或引导环境所需的命令。 AWS

有关可使用的受支持 CDK CLI 版本的列表，请参阅 AWS CDK 版本。

如果省略此属性，则该操作将使用以下主题之一中描述的默认 AWS CDK CLI 版本：

- “AWS CDK 部署”操作使用的 CDK CLI 版本
- “AWS CDK 引导”操作使用的 CDK CLI 版本

对应的 UI：“配置” 选项卡/ AWS CDK CLI 版本

CdkRootPath

(*CDKDeploy*/Configuration/CdkRootPath)

（可选）

包含 AWS CDK 项目`cdk.json`文件的目录的路径。AWS CDK 部署操作从该文件夹运行，并且该操作创建的所有输出都将添加到此目录中。如果未指定，则AWS CDK 部署操作假定该`cdk.json`文件位于 AWS CDK 项目的根目录中。

对应的 UI：“配置选项卡”/cdk.json 所在的目录

CfnOutputVariables

(*CDKDeploy*/Configuration/CfnOutputVariables)

（可选）

指定要在 AWS CDK 应用程序代码中将哪些`CfnOutput`构造显示为工作流程输出变量。然后，您可以在工作流的后续操作中引用工作流输出变量。有关中变量的更多信息 CodeCatalyst，请参阅在工作流中使用变量。

例如，如果您的 AWS CDK 应用程序代码如下所示：

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'amzn-s3-demo-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
```

```
    });
    new CfnOutput(this, 'bucketName', {
      value: bucket.bucketName,
      description: 'The name of the s3 bucket',
      exportName: 'amzn-s3-demo-bucket',
    });
    const table = new dynamodb.Table(this, 'todos-table', {
      partitionKey: {name: 'todoId', type: dynamodb.AttributeType.NUMBER},
      billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
      removalPolicy: RemovalPolicy.DESTROY,
    })
    new CfnOutput(this, 'tableName', {
      value: table.tableName,
      description: 'The name of the dynamodb table',
      exportName: 'myDynamoDbTable',
    });
    ...
  }
}
```

... 并且您的 `CfnOutputVariables` 属性如下：

```
Configuration:
  ...
  CfnOutputVariables: '["bucketName","tableName"]'
```

... 则该操作会生成以下工作流输出变量：

键	值
bucketName	`bucket.bucketName`
tableName	`table.tableName`

然后，您可以在后续操作中引用 `bucketName` 和 `tableName` 变量。要了解如何在后续操作中引用工作流输出变量，请参阅 引用预定义变量。

如果您未在`CfnOutputVariables`属性中指定任何`CfnOutput`构造，则该操作会将其找到的前四个（或更少）CloudFormation输出变量显示为工作流输出变量。有关更多信息，请参阅 “AWS CDK 部署”变量。

> **Tip**
>
> 要获取操作生成的所有 CloudFormation 输出变量的列表，请运行一次包含AWS CDK 部署操作的工作流，然后查看该操作的“日志”选项卡。日志包含与您的 AWS CDK 应用程序关联的所有 CloudFormation 输出变量的列表。知道所有 CloudFormation 变量是什么之后，就可以使用`CfnOutputVariables`属性指定要将哪些变量转换为工作流程输出变量。

有关 CloudFormation 输出变量的更多信息，请参阅 AWS Cloud Development Kit (AWS CDK) API 参考中的`CfnOutput`类 CfnOutput （构造）中提供的构造文档。

相应的 UI：配置选项卡/ CloudFormation 输出变量

CloudAssemblyRootPath

(*CDKDeploy*/Configuration/CloudAssemblyRootPath)

（可选）

如果您已经将 AWS CDK 应用程序的堆栈合成到云程序集中（使用`cdk synth`操作），请指定云程序集目录的根路径（`cdk.out`）。位于指定云程序集目录中的 CloudFormation 模板将通过AWS CDK 部署操作 AWS 账户 使用`cdk deploy --app`命令部署到您的中。当存在 `--app` 选项时，不会发生 `cdk synth` 操作。

如果您未指定云程序集目录，则 AWS CDK 部署操作将运行不带 `--app` 选项的 `cdk deploy` 命令。如果没有该`--app`选项，则该`cdk deploy`操作既会合成 (`cdk synth`)，又会将您的 AWS CDK 应用程序部署到您的 AWS 账户。

当“AWS CDK 部署”操作可以在运行时进行合成时，我为什么要指定现有的合成云组件？

您可能需要将现有的合成云程序集指定为：

- 确保每次运行“部署”操作时AWS CDK 部署完全相同的资源集

  如果您未指定云程序集，则 AWS CDK 部署操作可能会根据运行时间合成和部署不同的文件。例如，AWS CDK 部署操作可能会在测试阶段合成具有一组依赖项的云程序集，而在生产阶段使用另一组依赖项（如果这些依赖项在各个阶段之间发生了变化）。为了确保测试的内容和部署的内容之间完全对等，我们建议合成一次，然后使用云程序集目录路径字段（可视化编辑器）或 `CloudAssemblyRootPath` 属性（YAML 编辑器），指定已合成的云程序集。
- 将非标准的软件包管理器和工具与 AWS CDK 应用程序结合使用

在 synth 操作期间，AWS CDK 部署操作会尝试使用 npm 或 pip 等标准工具运行您的应用程序。如果操作未能使用这些工具成功运行您的应用程序，则合成将不会执行，操作将失败。要解决此问题，您可以在应用程序cdk.json的文件中指定成功运行应用程序所需的确切命令，然后使用不涉及AWS CDK 部署操作的方法合成应用程序。 AWS CDK 生成云程序集后，可以在 AWS CDK 部署操作的云程序集目录路径字段（可视化编辑器）或 CloudAssemblyRootPath 属性（YAML 编辑器）中指定该程序集。

有关配置cdk.json文件以包含用于安装和运行 AWS CDK 应用程序的命令的信息，请参阅指定应用程序命令。

有关 cdk deploy 和 cdk synth 命令以及 --app 选项的信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Deploying stacks、Synthesizing stacks 和 Skipping synthesis。

有关云程序集的信息，请参阅《AWS Cloud Development Kit (AWS CDK) API Reference》中的 Cloud Assembly。

对应的 UI：“配置”选项卡/云程序集目录路径

使用工作流程引导 AWS CDK 应用程序

本节介绍如何使用 CodeCatalyst 工作流程引导 AWS CDK 应用程序。为此，您必须将 AWS CDK 引导操作添加到工作流中。AWS CDK 引导操作使用现代模板，在您的 AWS 环境中预置引导堆栈。如果引导堆栈已存在，则操作会在必要时更新该堆栈。在中存在引导堆栈 AWS 是部署 AWS CDK 应用程序的先决条件。

有关引导的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) Developer Guide》中的 Bootstrapping。

主题

- 何时使用 “AWS CDK bootstrap” 操作
- “AWS CDK bootstrap” 操作的工作原理
- “AWS CDK 引导”操作使用的 CDK CLI 版本
- “AWS CDK bootstrap” 操作使用的运行时镜像
- 示例：引导应用程序 AWS CDK
- 添加 “AWS CDK 引导” 操作
- “AWS CDK 引导”变量

- “AWS CDK 引导”操作 YAML

何时使用 “AWS CDK bootstrap” 操作

如果您有部署 AWS CDK 应用程序的工作流程，并且想要同时部署（并在需要时更新）引导堆栈，请使用此操作。在这种情况下，您可以将AWS CDK 引导操作添加到与部署应用程序的工作流程相同的工作流程中。 AWS CDK

如果符合以下任一情况，请不要使用此操作：

- 您已使用另一种机制部署了引导堆栈，并且希望保持其原样（不更新）。
- 您想要使用自定义引导模板，但 AWS CDK 引导操作不支持该模板。

“AWS CDK bootstrap” 操作的工作原理

AWS CDK 引导操作的工作方式如下：

1. 在运行时，如果您指定了 1.0.7 或更早版本的操作，则该操作会将最新的 CDK CLI（也称为 AWS CDK Tookit）下载到构建映像。 CodeCatalyst

   如果您指定了 1.0.8 或更高版本，则该操作会与特定版本的 CDK CLI 捆绑在一起，因此不会下载。
2. 该操作使用 CDK CLI 来运行 `cdk bootstrap` 命令。此命令执行《AWS Cloud Development Kit (AWS CDK) Developer Guide》的 Bootstrapping 主题中介绍的引导任务。

“AWS CDK 引导”操作使用的 CDK CLI 版本

下表显示了不同版本的 AWS CDK 引导操作默认使用哪个版本的 CDK CLI。

> **Note**
>
> 您也许能够覆盖默认值。有关更多信息，请参阅“AWS CDK 引导”操作 YAML 中的 CdkCliVersion。

“AWS CDK 引导”操作版本	AWS CDK CLI 版本
1.0.0 – 1.0.7	最新

“AWS CDK 引导”操作版本	AWS CDK CLI 版本
1.0.8 或更高版本	2.99.1

“AWS CDK bootstrap” 操作使用的运行时镜像

下表显示了 CodeCatalyst 用于运行不同版本的AWS CDK 引导操作的运行时环境映像。这些映像包括不同的预安装工具集。有关更多信息，请参阅 活动映像。

> **Note**
>
> 我们建议您将 AWS CDK 引导操作升级到 2.x 版，从而利用 2024 年 3 月版映像中提供的最新工具。要升级操作，请在工作流定义文件中将其 `Identifier` 属性设置为 `aws/cdk-bootstrap@v2`。有关更多信息，请参阅 “AWS CDK 部署”操作 YAML。

“AWS CDK 引导”操作版本	运行时环境映像
1.x	2022 年 11 月版映像
2.x	2024 年 3 月版映像

示例：引导应用程序 AWS CDK

有关包含 AWS CDK 引导操作的工作流，请参阅使用工作流程部署 AWS CDK 应用程序中的示例：部署 AWS CDK 应用程序。

添加 “AWS CDK 引导” 操作

按照以下说明，将 AWS CDK 引导操作添加到您的工作流中。

开始之前

在使用 AWS CDK 引导操作之前，请确保已准备好 AWS CDK 应用程序。引导操作将在引导之前合成 AWS CDK 应用程序。您可以使用 AWS CDK支持的任何编程语言编写应用程序。

确保您的 AWS CDK 应用程序文件在以下位置可用：

- CodeCatalyst 源存储库，或
- 由另一个工作流程操作生成的 CodeCatalyst 输出对象

Visual

使用可视化编辑器添加 “AWS CDK bootstrap” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS CDK 引导操作，然后执行以下操作之一：
   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或
   - 选择 AWS CDK 引导。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入、配置和输出选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“AWS CDK 引导”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

> **Note**
>
> 如果您的 AWS CDK 引导操作失败并出现 `npm install` 错误，请参阅如何修复“npm install”错误？，了解有关如何修复错误的信息。

YAML

使用 YAML 编辑器添加 “AWS CDK bootstrap” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS CDK 引导操作，然后选择 +，将操作添加到工作流图表中并打开其配置窗格。
10. 根据需求修改 YAML 代码中的属性。“AWS CDK 引导”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

> **Note**
>
> 如果您的 AWS CDK 引导操作失败并出现 `npm install` 错误，请参阅如何修复“npm install”错误？，了解有关如何修复错误的信息。

“AWS CDK 引导”变量

AWS CDK 引导操作在运行时会生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 使用预定义变量。

键	值
deployment-platform	部署平台的名称。 硬编码为 `AWS:CloudFormation` 。

键	值
region	在工作流程运行期间部署 AWS CDK 引导堆栈的区域代码。 AWS 区域 示例：`us-west-2`
stack-id	已部署的 AWS CDK 引导堆栈的 Amazon 资源名称 (ARN)。 示例：`arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-bootstrap-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
SKIP-DEPLOYMENT	值为`true`表示在工作流程运行期间跳过了 AWS CDK 引导堆栈的部署。如果自上次部署以来，堆栈没有发生变化，则将跳过堆栈部署。 只有当该变量的值为 `true` 时，才会生成该变量。 硬编码为 `true`。

“AWS CDK 引导”操作 YAML

下面是 AWS CDK 引导操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流程引导 AWS CDK 应用程序。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKBootstrapAction_nn:
    Identifier: aws/cdk-bootstrap@v2
    DependsOn:
      - action-name
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_bootstrap_artifacts
          Files:
            - "cdk.out/**/*"
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      Region: us-west-2
      CdkCliVersion: version
```

CDKBootstrapAction

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：CDKBootstrapAction_nn。

对应的 UI：“配置”选项卡/操作显示名称

Identifier

(*CDKBootstrapAction*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅 指定要使用的操作版本。

> **Note**
>
> 指定 aws/cdk-bootstrap@v2 可以让操作在 2024 年 3 月版映像上运行，其中包括较新的工具，例如 Node.js 18。指定 aws/cdk-bootstrap@v1 可以让操作在 2022 年 11 月版映像上运行，其中包括较旧的工具，例如 Node.js 16。

默认值：aws/cdk-bootstrap@v2。

对应的 UI：工作流图表/CDKBootstrapAction_nn/aws/cdk-bootstrap@v2 标签

DependsOn

(*CDKBootstrapAction*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*CDKBootstrapAction*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*CDKBootstrapAction*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：配置 tab/Advanced -可选/计算类型

Fleet

(*CDKBootstrapAction*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：Linux.x86-64.Large、Linux.x86-64.XLarge。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 Fleet，则默认值为 Linux.x86-64.Large。

相应的 UI：配置 tab/Advanced -可选/计算舰队

Timeout

(*CDKBootstrapAction*/Timeout)

（必需）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Inputs

(*CDKBootstrapAction*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间 AWS CDK 引导操作所需的数据。

对应的 UI：输入选项卡

> **Note**
>
> 每个 AWS CDK 引导操作只能有一个输入（可以是源或构件）。

Sources

(*CDKBootstrapAction*/Inputs/Sources)

（如果您的 AWS CDK 应用程序存储在源存储库中，则为必填项）

如果您的 AWS CDK 应用程序存储在源存储库中，请指定该源存储库的标签。在启动引导过程之前，AWS CDK 引导操作会合成此存储库中的应用程序。目前唯一支持的存储库标签是 `WorkflowSource`。

如果您的 AWS CDK 应用程序不包含在源存储库中，则它必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*CDKBootstrapAction*/Inputs/Artifacts)

（如果您的 AWS CDK 应用程序存储在先前操作的输出构件中，则为必填项）

如果您的 AWS CDK 应用程序包含在先前操作生成的构件中，请在此处指定该构件。在AWS CDK 启动引导过程之前，bootstrap 操作会将指定工件中的应用程序合成到 CloudFormation 模板中。如果您的 AWS CDK 应用程序不包含在构件中，则必须驻留在您的源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输入”选项卡/构件 – 可选

Outputs

(*CDKBootstrapAction*/Outputs)

（可选）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts - output

(*CDKBootstrapAction*/Outputs/Artifacts)

（可选）

指定操作生成的构件。您可以在其他操作中将这些构件作为输入来引用。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件

Name

(*CDKBootstrapAction*/Outputs/Artifacts/Name)

（如果包含 Artifacts - output，则为必需）

指定将包含在运行时由AWS CDK 引导操作合成的 CloudFormation 模板的工件的名称。默认值为 `cdk_bootstrap_artifacts`。如果您未指定构件，则该操作会合成模板，但不会将模板保存在构件中。考虑将合成的模板保存在构件中，以便保留其记录，供测试或故障排除之用。

对应的用户界面：输出tab/Artifacts/Add构件/构建构件名称

Files

(*CDKBootstrapAction*/Outputs/Artifacts/Files)

（如果包含 Artifacts - output，则为必需）

指定要包含在构件中的文件。您`"cdk.out/**/*"`必须指定包含 AWS CDK 应用程序的合成 CloudFormation 模板。

> **Note**
>
> `cdk.out` 是保存已合成文件的默认目录。如果您指定了输出目录，而不是 `cdk.json` 文件中的 `cdk.out`，请在此处指定该目录，而不是 `cdk.out`。

相应的 UI：输出tab/Artifacts/Add构件/编译生成的文件

Environment

(*CDKBootstrapAction*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*CDKBootstrapAction*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*CDKBootstrapAction*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*CDKBootstrapAction*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*CDKBootstrapAction*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定引导操作用于访问 AWS 和添加AWS CDK 引导堆栈的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有合适的策略。

如果需要，可以在此操作中使用 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Configuration

(*CDKBootstrapAction*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

Region

(*CDKBootstrapAction*/Configuration/Region)

（必需）

指定要部署 AWS 区域 引导堆栈的。该区域应与您的 AWS CDK 应用程序部署的区域相匹配。有关区域代码的列表，请参阅区域端点。

对应的 UI：“配置”选项卡/区域

CdkCliVersion

(*CDKBootstrapAction*/Configuration/CdkCliVersion)

（可选）

此属性适用于 AWS CDK 部署操作的 1.0.13 或更高版本，以及 AWS CDK 引导操作的 1.0.8 或更高版本。

指定下列项之一：

- 您希望此操作使用的 AWS Cloud Development Kit (AWS CDK) 命令行界面 (CLI)（也称为 AWS CDK 工具包）的完整版本。示例：`2.102.1`。在构建和部署您的应用程序时，请考虑指定完整版本，从而确保一致性和稳定性。

  Or

- `latest`。请考虑指定 `latest`，从而利用 CDK CLI 的最新功能和修复。

该操作会将指定版本（或最新版本）的 AWS CDK CLI 下载到 CodeCatalyst 构建映像，然后使用此版本运行部署 CDK 应用程序或引导环境所需的命令。 AWS

有关可使用的受支持 CDK CLI 版本的列表，请参阅 AWS CDK 版本。

如果省略此属性，则该操作将使用以下主题之一中描述的默认 AWS CDK CLI 版本：

- “AWS CDK 部署”操作使用的 CDK CLI 版本
- “AWS CDK 引导”操作使用的 CDK CLI 版本

对应的 UI：“配置” 选项卡/ AWS CDK CLI 版本

使用工作流将文件发布到 Amazon S3

本节介绍如何使用 CodeCatalyst 工作流程将文件发布到 Amazon S3。为此，您必须将 Amazon S3 发布操作添加到工作流。Amazon S3 发布操作会将源目录中的文件复制到 Amazon S3 存储桶。源目录可以位于：

- 源存储库，或
- 由另一个工作流操作生成的输出构件

主题

何时使用“Amazon S3 发布”操作

在以下情况下使用此操作：

- 您有一个生成要存储在 Amazon S3 中的文件的工作流。

  例如，您可能有一个用于构建要在 Amazon S3 中托管的静态网站的工作流。在这种情况下，您的工作流将包括一个用于构建站点的 HTML 和支持文件的构建操作，以及一个用于将文件复制到 Amazon S3 的 Amazon S3 发布操作。
- 您有一个包含要存储在 Amazon S3 中的文件的源存储库。

  例如，您可能有一个包含应用程序源文件的源存储库，您每晚都要将这些文件存档到 Amazon S3。

“Amazon S3 发布”操作使用的运行时映像

Amazon S3 发布操作在 2022 年 11 月版映像上运行。有关更多信息，请参阅 活动映像。

示例：将文件发布到 Amazon S3

以下示例工作流包括 Amazon S3 发布操作以及构建操作。该工作流会构建一个静态文档网站，然后将该网站发布到用于托管它的 Amazon S3。工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- 构建操作（BuildDocs）– 触发后，该操作会构建一个静态文档网站（mkdocs build），并将关联的 HTML 文件和支持元数据添加到名为 MyDocsSite 的构件中。有关构建操作的更多信息，请参阅使用工作流进行构建。
- Amazon S3 发布操作（PublishToS3）– 在构建操作完成后，此操作会将 MyDocsSite 构件中的站点复制到 Amazon S3 以进行托管。

> **Note**
>
> 以下工作流示例仅用于说明目的，如果不执行附加配置，则无法运行。

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略此部分，则必须确保您环境的默认 IAM 角色字段中指定的角色包含 Amazon S3 发布操作所需的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。有关 Amazon S3 发布操作所需的权限和信任策略的更多信息，请参阅“Amazon S3 发布”操作 YAML 中的 Role 属性的说明。

```
Name: codecatalyst-s3-publish-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildDocs:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: echo BuildDocs started on `date`
```

```
      - Run: pip install --upgrade pip
      - Run: pip install mkdocs
      - Run: mkdocs build
      - Run: echo BuildDocs completed on `date`
  Outputs:
    Artifacts:
    - Name: MyDocsSite
      Files:
        - "site/**/*"

PublishToS3:
  Identifier: aws/s3-publish@v1
  Environment:
    Name: codecatalyst-s3-publish-environment
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-s3-publish-build-role
  Inputs:
    Sources:
      - WorkflowSource
    Artifacts:
      - MyDocsSite
  Configuration:
    DestinationBucketName: amzn-s3-demo-bucket
    SourcePath: /artifacts/PublishToS3/MyDocSite/site
    TargetPath: my/docs/site
```

添加“Amazon S3 发布”操作

按照以下说明操作，将 Amazon S3 发布操作添加到工作流。

Visual

使用可视化编辑器添加“Amazon S3 发布”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。

6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 Amazon S3 发布操作，然后执行下列操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 Amazon S3 发布。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入、配置和输出选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“Amazon S3 发布”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加“Amazon S3 发布”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 Amazon S3 发布操作，然后执行下列操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

- 选择 Amazon S3 发布。此时会显示操作详细信息对话框。在此对话框中：
  - （可选）选择查看源以查看操作的源代码。
  - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 根据需求修改 YAML 代码中的属性。“Amazon S3 发布”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

“Amazon S3 发布”操作 YAML

下面是 Amazon S3 发布操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流将文件发布到 Amazon S3。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  S3Publish_nn:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
```

```
    Inputs:
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      SourcePath: my/source
      DestinationBucketName: amzn-s3-demo-bucket
      TargetPath: my/target
```

S3Publish

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：S3Publish_nn。

对应的 UI：“配置”选项卡/操作名称

Identifier

(*S3Publish*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/s3-publish@v1。

对应的 UI：工作流图表/S3Publish_nn/aws/s3-publish@v1 标签

DependsOn

(*S3Publish*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*S3Publish*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*S3Publish*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

对应的 UI：“配置”选项卡/计算类型

Fleet

(*S3Publish*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

对应的 UI：“配置”选项卡/计算实例集

Timeout

(*S3Publish*/Timeout)

（必需）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Inputs

(*S3Publish*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间 `S3Publish` 所需的数据。

> **Note**
>
> 每个 AWS CDK 部署操作最多有四个输入（一个源和三个构件）。变量不计入此总数。

如果您需要引用驻留在不同输入（例如源和构件）中的文件，则源输入是主输入，构件是辅助输入。辅助输入中对文件的引用采用特殊前缀，以与主输入中的文件区分开来。有关更多信息，请参阅 示例：引用多个构件中的文件。

对应的 UI：输入选项卡

Sources

(*S3Publish*/Inputs/Sources)

（如果要发布到 Amazon S3 的文件存储在源存储库中，则为必需）

如果要发布到 Amazon S3 的文件存储在源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 `WorkflowSource`。

如果要发布到 Amazon S3 的文件不包含在源存储库中，则必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*S3Publish*/Inputs/Artifacts)

（如果要发布到 Amazon S3 的文件存储在先前操作生成的输出构件中，则为必需）

如果要发布到 Amazon S3 的文件包含在上一操作生成的构件中，请在此处指定该构件。如果您的文件不包含在构件中，则必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Variables - input

(*S3Publish*/Inputs/Variables)

（可选）

指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输入”选项卡/变量 – 可选

Environment

(*S3Publish*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*S3Publish*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*S3Publish*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*S3Publish*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*S3Publish*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定 A mazon S3 发布操作用于访问 AWS 和将文件复制到 Amaz on S3 的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
>
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

- 以下自定义信任策略：

> **Note**
>
> 如果需要，可以在此操作中使用 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Configuration

(*S3Publish*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

SourcePath

(*S3Publish*/Configuration/SourcePath)

（必需）

指定要发布到 Amazon S3 的目录或文件的名称和路径。目录或文件可以位于源存储库或先前操作的构件中，并且相对于源存储库或构件根目录。

示例：

指定 `./myFolder/` 会将 `/myFolder` 的内容复制到 Amazon S3，并保留底层目录结构。

指定 `./myFolder/myfile.txt` 仅将 `myfile.txt` 复制到 Amazon S3。（这将移除目录结构。）

您无法使用通配符。

> **Note**
>
> 您可能需要在目录或文件路径中添加前缀，以指明要在哪个构件或源中查找它。有关更多信息，请参阅引用源存储库文件和在构件中引用文件。

对应的 UI：“配置”选项卡/源路径

DestinationBucketName

(*S3Publish*/Configuration/DestinationBucketName)

（必需）

指定要将文件发布到的 Amazon S3 存储桶的名称。

对应的 UI：“配置”选项卡/目标存储桶 – 可选

TargetPath

(*S3Publish*/Configuration/TargetPath)

（可选）

指定要将文件发布到的 Amazon S3 中的目录的名称和路径。如果该目录不存在，系统会创建它。该目录路径不得包含存储桶名称。

示例：

```
myS3Folder
```

```
./myS3Folder/myS3Subfolder
```

对应的 UI：“配置”选项卡/目标目录 – 可选

部署到 AWS 账户 和 VPCs

使用CodeCatalyst 工作流程，您可以将应用程序和其他资源部署到 AWS 云 VPCs 中的目标和 AWS 账户 Amazon。要启用这些部署，必须设置 CodeCatalyst 环境。

不要将 CodeCatalyst 环境与开发环境混淆，它定义了 CodeCatalyst 工作流程所连接的目标 AWS 账户 和可选的 Amazon VPC。环境还定义了工作流程访问目标账户中的 AWS 服务和资源所需的 IAM 角色。

您可以设置多个环境并为它们指定名称，例如开发、测试、暂存和生产。在这些环境中部署时，有关部署的信息会显示在环境中的 CodeCatalyst 部署活动和部署目标选项卡上。

如何开始使用环境？

添加和使用 CodeCatalyst 环境的高级步骤如下：

1. 在您的 CodeCatalyst 空间中，关联一个或多个 AWS 账户。在此过程中，添加工作流访问 AWS 账户中的资源所需的 IAM 角色。有关更多信息，请参阅 允许在已连接的情况下访问 AWS 资源 AWS 账户。
2. 在您的 CodeCatalyst 项目中，创建一个包含步骤 1 中的 AWS 账户 s 和 IAM 角色之一的环境。有关更多信息，请参阅 创建环境。

3. 在您的 CodeCatalyst 项目中，在工作流程中，添加指向您在步骤 2 中创建的环境的操作。有关更多信息，请参阅 添加操作到工作流。

   现在，您已配置一个环境。此操作现在可将资源部署到环境中指定的 AWS 账户 中。

> **Note**
>
> 您也可以将 Amazon VPC 添加到环境中。有关更多信息，请参阅《CodeCatalyst 管理指南》中的为空间添加 VPC 连接和将 VPC 与环境关联。

单个工作流中是否能有多个环境？

可以。如果工作流包含多个操作，则可以为每个操作分配一个环境。例如，您的工作流可能包含两个部署操作，为一个部署操作分配了 `my-staging-enviroment` 环境，为另一个部署操作分配了 `my-production-environment` 环境。

哪些工作流操作支持环境？

任何将资源部署到 AWS 云端或出于其他原因（例如监控和报告）与 AWS 服务通信的工作流程操作都支持环境。

哪些操作支持在中显示其部署信息 CodeCatalyst？

在支持环境的工作流程操作中，只有少数支持将其部署信息显示在 CodeCatalyst 控制台的 “部署活动” 和 “部署目标” 页面上。

以下工作流操作支持显示其部署信息：

- 部署 CloudFormation 堆栈-有关更多信息，请参阅 部署 CloudFormation 堆栈
- 部署到 Amazon ECS – 有关更多信息，请参阅使用工作流部署到 Amazon ECS
- 部署到 Kubernetes 集群 – 有关更多信息，请参阅使用工作流部署到 Amazon EKS
- AWS CDK 部署-有关更多信息，请参阅 使用工作流程部署 AWS CDK 应用程序

支持的区域：

环境页面可以显示任何 AWS 区域中的资源。

环境是强制性的吗？

如果分配给环境的工作流程操作将资源部署到 AWS 云中，或者出于其他原因（例如监控和报告）与 AWS 服务通信，则该环境是必需的。

例如，如果您的构建操作可以构建应用程序，但不需要与您的 AWS 账户 或 Amazon VPC 通信，则无需为该操作分配环境。但是，如果构建操作将日志发送到您的 Amazon CloudWatch 服务 AWS 账户，则必须为该操作分配环境。

主题

创建环境

按照以下说明操作来创建稍后能够与工作流操作关联的环境。

开始前的准备工作

您需要以下项：

- 一个 CodeCatalyst 空间。有关更多信息，请参阅 设置并登录 CodeCatalyst。
- 一个 CodeCatalyst 项目。有关更多信息，请参阅 使用蓝图创建项目。
- 包含您的工作流程操作需要访问的 IAM 角色的 AWS 账户连接 AWS。有关创建账户连接的信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。对于每个环境，最多使用一个账户连接。

> **Note**
>
> 您可以在没有账户连接的情况下创建环境；但您稍后需添加该连接。

- 以下 CodeCatalyst 角色之一：
  - 空间管理员
  - 项目管理员
  - 贡献者

> **Note**
>
> 如果您具有贡献者角色，则可以创建一个环境，但无法将该环境与 AWS 账户 连接关联。你需要请具有空间管理员或项目管理员角色的人员将环境与 AWS 账户 连接关联。

有关权限和角色的更多信息，请参阅向用户授予项目权限。

创建环境

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 在环境名称中，输入一个名称，例如 **Production** 或 **Staging**。
5. 在环境类型中，选择下列选项之一：

   - 非生产 – 在将应用程序投入生产之前，可在其中测试应用程序以确保其按预期运行的环境。
   - 生产 – 一个公开可用的“实时”环境，用于托管您最终的应用程序。

   如果您选择生产，则 UI 中与环境关联的所有操作旁边都会显示一个生产徽章。该徽章可帮助您快速查看哪些操作正在部署到生产中。除了徽章的外观外，生产环境和非生产环境之间没有区别。
6. （可选）在描述中，输入描述，例如 **Production environment for the hello-world app**。
7. 在AWS 账户 连接中-可选，选择要与此环境关联的 AWS 账户连接。分配了此环境的工作流操作将能够连接到关联的 AWS 账户。有关在中创建 AWS 账户 连接的更多信息 CodeCatalyst，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。

   如果您要使用的 AWS 账户 连接未列出，则可能是因为您的项目中不允许使用该连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。
8. 在默认 IAM 角色中，选择要与此环境关联的 IAM 角色。分配给此环境的工作流程操作将继承此 IAM 角色，并能够使用该角色连接到您的中的服务和资源 AWS 账户。

   如果您需要将环境分配给多个操作，并且这些操作需要的 IAM 角色与此处指定的默认角色不同，则可以使用切换角色选项在每个操作的配置选项卡上指定不同的角色。有关更多信息，请参阅 更改操作的 IAM 角色。

如果您要用作默认角色的 IAM 角色未列出，则可能是因为您尚未将其添加到 AWS 账户 连接中。要向账户连接添加 IAM 角色，请参阅将 IAM 角色添加到账户连接。

9. （可选）在 VPC 连接中，选择要与此环境关联的 VPC 连接。有关创建 VPC 连接的更多信息，请参阅《亚马逊 CodeCatalyst 管理员指南》中的管理亚马逊虚拟私有云。

   如果您要使用的 VPC 连接未列出，则可能是因为它包含了您的项目中不允许的 AWS 账户 连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

10. 选择 “创建环境”。 CodeCatalyst 创建一个空环境。

后续步骤

- 现在您已创建一个环境，可以将该环境与工作流操作关联。有关更多信息，请参阅 将环境与操作关联。

将环境与操作关联

当您将环境与支持的工作流程操作关联时，该环境的 AWS 账户默认 IAM 角色和可选的 Amazon VPC 将分配给该操作。之后，此操作可以使用 IAM 角色连接和部署到 AWS 账户 ，还可以连接到可选 Amazon VPC。

按照以下说明操作来将环境与操作关联。

步骤 1：将环境与工作流操作关联

使用以下过程将环境与工作流操作关联。

Visual

使用可视化编辑器将环境与工作流操作关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。

7. 在工作流图中，选择环境支持的操作。有关更多信息，请参阅 哪些操作支持在中显示其部署信息 CodeCatalyst？。
8. 选择配置选项卡，然后在环境字段中指定信息，如下所示。

   环境

   指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 Amazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

   > **Note**
   >
   > 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

   有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。
9. （可选）更改与操作关联的 IAM 角色。如果角色包含操作的一组错误权限，则可能需要更改角色。

   要更改角色，请执行以下操作：

   1. 在 W hat's in***my-environment***？ 框，然后选择垂直省略号图标 (⋮)。
   2. 选择下列选项之一：
      - 切换角色。选择此选项能且只能更改此操作使用的 IAM 角色。其他操作继续使用其关联环境中指定的默认 IAM 角色。有关更多信息，请参阅 更改操作的 IAM 角色。
      - 编辑环境。选择此选项可更改环境中列出的默认 IAM 角色。选择此选项后，您的操作以及与同一环境关联的任何其他操作都将开始使用新的默认 IAM 角色。

      > **Important**
      >
      > 更新默认 IAM 角色时要谨慎。如果角色具备的权限不足，无法执行所有共享环境的操作，则更改角色可能会导致操作失败。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器将环境与工作流操作关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要与环境关联的工作流操作中，添加与以下内容类似的代码：

```
action-name:
  Environment:
    Name: environment-name
```

有关更多信息，请参阅操作类型主题。本主题包含每个操作的文档链接，包括其 YAML 参考。

8. （可选）如果您希望操作使用与环境中列出的默认 IAM 角色不同的角色，请添加包含您要使用的角色的 `Connections:` 部分。有关更多信息，请参阅 更改操作的 IAM 角色。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

步骤 2：填充部署活动页面

将环境与工作流操作关联后，可以在 CodeCatalyst 控制台的 “环境” 部分的 “部署” 活动和 “部署目标” 页面中填充部署信息。按照以下说明操作来填充这些页面。

> **Note**
>
> 只有少数操作支持在 CodeCatalyst控制台中显示其部署信息。有关更多信息，请参阅 哪些操作支持在中显示其部署信息 CodeCatalyst？。

将部署信息添加到 CodeCatalyst

1. 如果您在步骤 1：将环境与工作流操作关联中提交更改时工作流运行未自动启动，请执行以下操作来手动启动运行：

   a. 在导航窗格中，选择 CI/CD，然后选择工作流。

   b. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

   c. 选择运行。

   工作流程运行会启动新的部署，这会导致 CodeCatalyst 向添加部署信息 CodeCatalyst。

2. 确认部署活动已添加到 CodeCatalyst 控制台：

   a. 在导航窗格中，选择 CI/CD，然后选择环境。

   b. 选择您的环境（例如 `Production`）。

   c. 选择部署活动选项卡，并确认部署的状态为成功。这表示工作流运行已成功部署您的应用程序资源。

   d. 选择部署目标选项卡，并确认已显示您的应用程序资源。

将 VPC 与环境关联

在使用具有 VPC 连接的环境配置操作时，该操作将在连接到 VPC 的情况下运行，遵守网络规则并访问关联的 VPC 所指定的资源。一个或多个环境可使用同一 VPC 连接。

按照以下说明操作来将 VPC 连接与环境关联。

将 VPC 连接与环境关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 选择您的环境（例如 `Production`）。
5. 选择环境属性选项卡。
6. 选择管理 VPC 连接，再选择所需的 VPC 连接，然后选择确认。这会将选定的 VPC 连接与此环境关联。

> **Note**
>
> 如果您要使用的 VPC 连接未列出，则可能是因为该 AWS 账户 连接包含您的项目中不允许的连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

有关更多信息，请参阅《CodeCatalyst 管理员指南》中的管理 Amazon 虚拟私有云。

将 AWS 账户 与环境关联

按照以下说明将 AWS 账户 与环境相关联。当您将 AWS 账户 与环境关联时，分配给该环境的工作流操作将能够连接到该环境 AWS 账户。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。

开始前的准备工作

您需要以下项：

- 包含您的工作流程操作需要访问的 IAM 角色的 AWS 账户连接 AWS。有关创建账户连接的信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。对于每个环境，最多使用一个账户连接。
- 以下 CodeCatalyst 角色之一：空间管理员或项目管理员。有关更多信息，请参阅 向用户授予项目权限。

将 AWS 账户 与环境关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 选择您的环境（例如 `Production`）。
5. 选择编辑环境。
6. 在环境属性下的 AWS 账户 连接 – 可选下拉列表中，选择所需的 AWS 账户。

   如果您要使用的 AWS 账户 连接未列出，则可能是因为您的项目中不允许使用该连接。有关更多信息，请参阅《A mazon CodeCatalyst 管理员指南》中的配置受项目限制的账户连接。

7. 在默认 IAM 角色中，选择要与此环境关联的 IAM 角色。分配给此环境的工作流程操作将继承此 IAM 角色，并能够使用该角色连接到您的中的服务和资源 AWS 账户。

   如果您要用作默认角色的 IAM 角色未列出，则可能是因为您尚未将其添加到 AWS 账户 连接中。要向账户连接添加 IAM 角色，请参阅将 IAM 角色添加到账户连接。

更改操作的 IAM 角色

默认情况下，在将环境与工作流操作关联时，该操作将继承环境中指定的默认 IAM 角色。您可以更改此行为，让操作使用其他角色。如果默认 IAM 角色缺少操作在 AWS 云中运行所需的权限，则可能需要让操作使用其他角色。

要向操作分配其他 IAM 角色，您可以使用可视化编辑器中的切换角色选项或 YAML 编辑器中的 `Connections:` 属性。新角色将覆盖环境中指定的默认 IAM 角色，可让您将默认 IAM 角色保持原样。如果有其他操作使用默认 IAM 角色，则可能需要将它保持原样。

按照以下说明操作，将操作配置为使用与其环境中指定的 IAM 角色不同的 IAM 角色。

Visual

向操作分配其他 IAM 角色（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择代表要更新其 IAM 角色的操作的框。
7. 选择配置选项卡。
8. 在 W hat's in***my-environment***？ 框中，选择垂直省略号图标
   (⋮)。
9. 选择切换角色。
10. 在切换角色对话框的 IAM 角色下拉列表中，选择您希望该操作使用的 IAM 角色。此角色将覆盖环境中指定的默认 IAM 角色。如果要使用的角色不在列表中，请确保已将它添加到空间。有关更多信息，请参阅 将 IAM 角色添加到账户连接。

所选角色现在出现在 W hat's in***my-environment***？ 框以及 “在工作流程中定义” 徽章。此角色还显示在工作流定义文件的 `Connections:` 部分中。

11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

向操作分配其他 IAM 角色（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要在其中使用其他 IAM 角色的工作流操作中，添加一个类似于以下内容的 `Connections:` 部分：

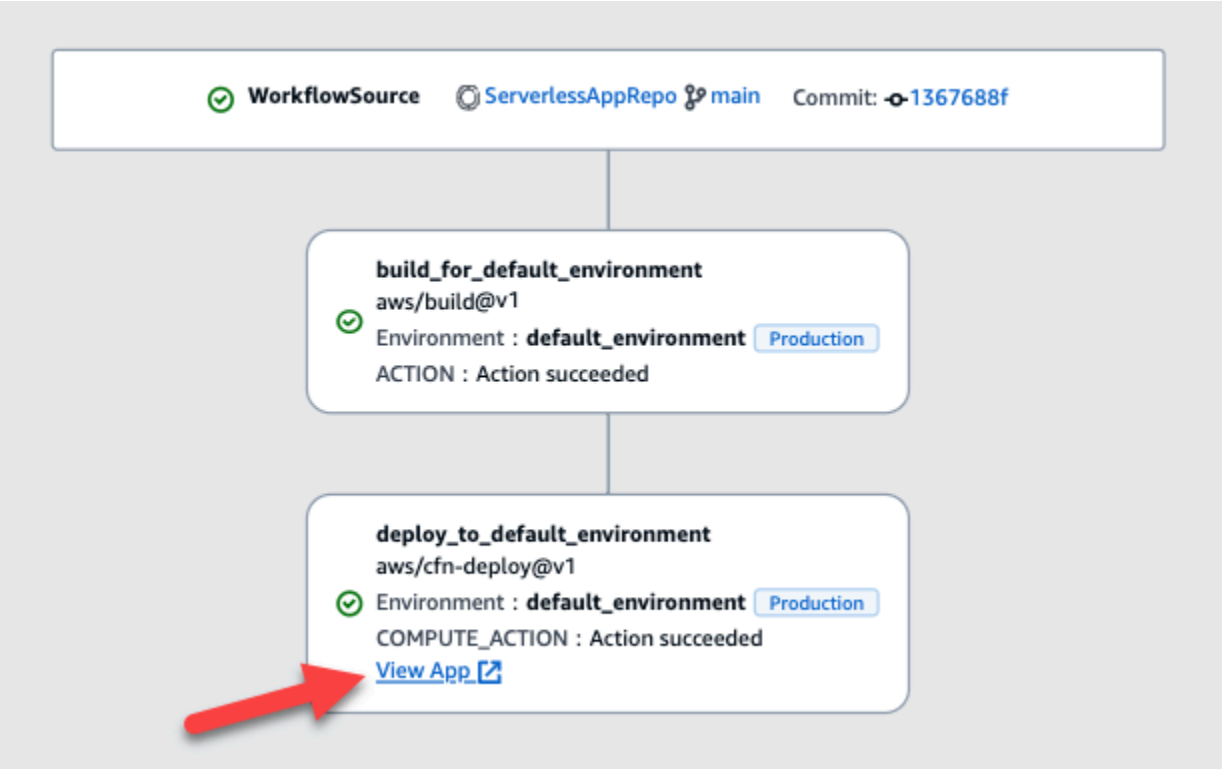
```
action-name:
  Environment:
    Name: environment-name
    Connections:
      - Name: account-connection-name
        Role: iam-role-name
```

在前面的代码中，*account-connection-name*替换为包含 IAM 角色的账户连接的名称，并*iam-role-name*替换为您希望操作使用的 IAM 角色的名称。此角色将覆盖环境中指定的默认 IAM 角色。确保您已将此角色添加到空间。有关更多信息，请参阅 将 IAM 角色添加到账户连接。

有关更多信息，请参阅操作类型主题。本主题包含每个操作的文档链接，包括其 YAML 参考。

在工作流图中显示应用程序 URL

如果您的工作流程部署了应用程序，则可以将 Amazon 配置 CodeCatalyst 为将应用程序的 URL 显示为可点击的链接。此链接出现在 CodeCatalyst 控制台中，位于部署该链接的操作中。以下工作流图显示了位于操作底部的查看应用程序 URL。



通过在 CodeCatalyst 控制台中将此 URL 设置为可点击，您可以快速验证您的应用程序部署。

> **Note**
>
> 部署到 Amazon ECS 操作不支持应用程序 URL。

要启用此功能，请向操作添加名称包含 `appurl` 或 `endpointurl` 的输出变量。您可以使用带有连接符（`-`）、下划线（`_`）或空格（ ）的名称，也可以不使用这些字符。此字符串不区分大小写。将变量的值设置为已部署的应用程序的 `http` 或 `https` URL。

> **Note**
>
> 如果要更新现有输出变量以包含 `app url` 或 `endpoint url` 字符串，请更新对该变量的所有引用以使用新的变量名。

有关详细步骤，请参阅下列过程之一：

- 在 “AWS CDK 部署” 操作中显示应用程序网址
- 在 “部署 CloudFormation 堆栈” 操作中显示应用程序 URL
- 在所有其他操作中显示应用程序 URL

配置完 URL 后，请按照以下说明操作来验证它是否按预期显示：

- 验证是否已添加应用程序 URL

在 “AWS CDK 部署” 操作中显示应用程序网址

1. 如果您使用的是AWS CDK 部署操作，请在 AWS CDK 应用程序代码中添加一个`CfnOutput`构造（这是键值对）：

   - 键名称必须包含 `appurl` 或 `endpointurl`，可以带有连接符（`-`）、下划线（`_`）或空格（ ），也可以不使用这些字符。此字符串不区分大小写。
   - 值必须是已部署的应用程序的 `http` 或 `https` URL。

   例如，你的 AWS CDK 代码可能如下所示：

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'amzn-s3-demo-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'APP-URL', {
      value: https://mycompany.myapp.com,
      description: 'The URL of the deployed application',
      exportName: 'myApp',
    });
    ...
  }
```

```
}
```

有关该CfnOutput构造的更多信息，请参阅 AWS Cloud Development Kit (AWS CDK) API 参考CfnOutputProps中的接口。

2. 保存并提交您的代码。
3. 继续执行验证是否已添加应用程序 URL。

在 “部署 CloudFormation 堆栈” 操作中显示应用程序 URL

1. 如果您使用的是部署 CloudFormation 堆栈操作，请向 CloudFormation 模板或 AWS SAM 模板中的Outputs部分添加具有以下特征的输出：

   - 键（也称作逻辑 ID）必须包含 appurl 或 endpointurl，可以带有连接符（-）、下划线（_）或空格（ ），也可以不使用这些字符。此字符串不区分大小写。
   - 值必须是已部署的应用程序的 http 或 https URL。

   例如，您的 CloudFormation 模板可能如下所示：

```
"Outputs" : {
  "APP-URL" : {
    "Description" : "The URL of the deployed app",
    "Value" : "https://mycompany.myapp.com",
    "Export" : {
      "Name" : "My App"
    }
  }
}
```

   有关 CloudFormation 输出的更多信息，请参阅《AWS CloudFormation 用户指南》中的输出。
2. 保存并提交您的代码。
3. 继续执行验证是否已添加应用程序 URL。

在所有其他操作中显示应用程序 URL

如果您使用其他操作来部署应用程序，例如构建GitHub 操作或操作，请执行以下操作以显示应用程序网址。

1. 在工作流定义文件中操作的 Inputs 或 Steps 部分中定义环境变量。此变量必须具有以下特征：

   - name 必须包含 appurl 或 endpointurl，可以带有连接符（-）、下划线（_）或空格（ ），也可以不使用这些字符。此字符串不区分大小写。
   - 值必须是已部署的应用程序的 http 或 https URL。

   例如，构建操作可能与以下内容类似：

   ```
   Build-action:
     Identifier: aws/build@v1
     Inputs:
       Variables:
         - Name: APP-URL
           Value: https://mycompany.myapp.com
   ```

   ...或与以下内容类似：

   ```
   Actions:
     Build:
       Identifier: aws/build@v1
       Configuration:
         Steps:
           - Run: APP-URL=https://mycompany.myapp.com
   ```

   有关定义环境变量的更多信息，请参阅定义变量。

2. 导出变量。

   例如，您的构建操作可能与以下内容类似：

   ```
   Build-action:
     ...
     Outputs:
       Variables:
         - APP-URL
   ```

   有关导出变量的信息，请参阅导出变量以便其他操作使用。

3. （可选）选择验证，在提交之前验证工作流的 YAML 代码。

4. 选择提交，输入提交消息，然后再次选择提交。

5. 继续执行验证是否已添加应用程序 URL。

验证是否已添加应用程序 URL

- 启动工作流运行（如果它未自动启动）。新运行的应用程序 URL 应在其工作流图中显示为可单击链接。有关启动运行的更多信息，请参阅手动启动工作流运行。

移除部署目标

您可以从 CodeCatalyst 控制台的部署目标页面中移除部署目标，例如 Amazon ECS 集群或 CloudFormation 堆栈。

> **Important**
>
> 移除部署目标时，它会从 CodeCatalyst 控制台中删除，但在托管它的 AWS 服务中仍然可用（如果它仍然存在）。

如果部署目标已过时，可以考虑移除该目标。 CodeCatalyst在以下情况下，目标可能会过时：

- 您删除了已部署到目标的工作流。
- 您更改了要部署到的堆栈或集群。
- 您在 AWS 控制台中从或 Amazon ECS 服务中删除了堆栈 CloudFormation 或集群。

移除部署目标

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 选择包含要移除的部署目标的环境的名称。有关环境的信息，请参阅部署到 AWS 账户 和 VPCs。
5. 选择部署目标选项卡。
6. 选中要移除的部署目标旁边的单选按钮。
7. 选择移除 。

   这将从页面中移除目标。

通过提交跟踪部署状态

在开发生命周期的任何时候，了解特定提交（如错误修复、新功能或其它有影响的变更）的部署状态都很重要。考虑以下场景，在这些场景中，部署状态跟踪功能会对开发团队有所帮助：

- 作为一名开发人员，您已经修复了一个错误，并希望报告其在团队部署环境中的部署状态。
- 作为发布管理者，您希望查看已部署提交的列表，以跟踪和报告其部署状态。

CodeCatalyst 提供了一个视图，您可以用来一目了然地确定各个提交或更改的部署位置以及部署到哪个环境。此视图包括：

- 提交列表。
- 包含提交的部署的状态。
- 成功部署提交的环境。
- 根据 CI/CD 工作流程中的提交运行的任何测试的状态。

下面的步骤将详细介绍如何导航到该视图并使用该视图跟踪项目中的更改。

> **Note**
>
> 只有CodeCatalyst 存储库才支持通过提交跟踪部署状态。您不能将此功能用于GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库。

通过提交跟踪部署状态

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择更改跟踪。
4. 在主窗格顶部的两个下拉列表中，选择包含要查看其发布状态的提交的源存储库和分支。
5. 选择查看更改。

   出现提交列表。

   对于每个提交，您可以查看以下内容：

- 提交信息，如 ID、作者、消息和提交时间。有关更多信息，请参阅 使用源存储库存储代码并协作处理代码 CodeCatalyst。
- 部署到每个环境的状态。有关更多信息，请参阅 部署到 AWS 账户 和 VPCs。
- 测试和代码覆盖率结果。有关更多信息，请参阅 使用工作流进行测试。

> **Note**
>
> 不显示软件组成分析（SCA）结果。

6. （可选）要查看与特定提交相关的更多更改信息，包括最新部署以及详细的代码覆盖率和单元测试信息，请选择该提交对应的查看详细信息。

查看部署日志

您可以查看与特定部署操作相关的日志，以解决 Amazon 中的问题 CodeCatalyst。

您可以从工作流或环境开始查看日志。

查看从工作流开始的部署操作的日志

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择运行。
6. 选择已部署应用程序的工作流运行。
7. 在工作流图中，选择要查看其日志的操作。
8. 选择日志选项卡并展开各个部分以显示日志消息。
9. 要查看更多日志，请选择 “摘要” 选项卡，然后选择 “查看于” CloudFormation（如果可用），在那里查看更多日志。您可能需要登录 AWS。

查看从环境开始的部署操作的日志

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 选择已将应用程序部署到的环境。
5. 在部署活动中，找到工作流运行 ID 列，然后选择已部署堆栈的工作流运行。
6. 在工作流图中，选择要查看其日志的操作。
7. 选择日志选项卡并展开各个部分以显示日志消息。
8. 要查看更多日志，请选择 “摘要” 选项卡，然后选择 “查看于” CloudFormation（如果可用），在那里查看更多日志。您可能需要登录 AWS。

查看部署信息

您可以在 Amazon 中查看有关部署的以下信息 CodeCatalyst：

- 部署活动，包括部署状态、开始时间、结束时间、历史记录和事件持续时间。
- 堆栈名称 AWS 区域、上次更新时间和相关工作流程。
- 提交和拉取请求。
- 特定于操作的信息，例如 CloudFormation 事件和输出。

您可以从工作流、环境或工作流操作开始查看部署信息。

从工作流开始查看部署信息

- 转到已部署应用程序的工作流运行。有关说明，请参阅查看工作流运行状态和详细信息。

从环境开始查看部署信息

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择环境。
4. 选择已在其中部署堆栈的环境，例如 `Production`。
5. 选择部署活动可查看堆栈的部署历史记录、部署状态（例如，成功或失败）以及其他与部署相关的信息。
6. 选择部署目标可查看有关已部署到环境中的堆栈、集群或其他目标的信息。您可以查看堆栈名称、区域、提供商和标识符等信息。

从操作开始查看部署信息

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 在工作流图中，选择已部署应用程序的工作流操作。例如，你可以选择DeployCloudFormationStack。
6. 查看右侧窗格中的内容以了解特定于操作的部署信息。

创建工作流

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

使用以下步骤在中创建工作流程 CodeCatalyst。工作流将作为 YAML 文件，存储在所选源存储库的 `~/.codecatalyst/workflows/` 文件夹中。或者，您可以将工作流存储在 `~/.codecatalyst/workflows/` 的子文件夹中，方法是在提交时在工作流文件名前面加上文件夹名称。有关更多信息，请参阅以下说明。

有关工作流的更多信息，请参阅使用工作流进行构建、测试和部署。

Visual

使用可视化编辑器创建工作流

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。

3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择创建工作流。

   此时将显示创建工作流对话框。
5. 在源存储库字段中，选择工作流定义文件所在的源存储库。如果源存储库不存在，请创建一个存储库。
6. 在分支字段中，选择工作流定义文件所在的分支。
7. 选择创建。

   Amazon CodeCatalyst 将存储库和分支信息保存在内存中，但工作流程尚未提交。
8. 选择可视化。
9. 构建工作流：

   a. （可选）在工作流图表中，选择源和触发器框。此时将出现触发器窗格。选择添加触发器以添加触发器。有关更多信息，请参阅 添加触发器到工作流。

   b. 选择 + 操作（左上角）。此时将出现操作目录。

   c. 选择某个操作中的加号（+）可将该操作添加到工作流中。使用右侧的窗格配置操作。有关更多信息，请参阅添加操作到工作流。

   d. （可选）选择工作流属性（右上角）。此时将出现工作流属性窗格。配置工作流名称、运行模式和计算。有关更多信息，请参阅配置运行的排队行为和配置计算和运行时映像。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，然后在提交工作流对话框中执行以下操作：

    a. 对于工作流文件名，保留默认名称或输入您自己的名称。文件将存储在所选源存储库和分支的 `~/.codecatalyst/workflows/` 文件夹中。您可以在文件名前面加上文件夹或子文件夹。示例：

       - 指定 `my-workflow`（无文件夹）会将文件存储为 `~/.codecatalyst/workflows/my-workflow.yaml`
       - 指定 `folder/subfolder/my-workflow` 会将文件存储为 `~/.codecatalyst/workflows/folder/subfolder/my-workflow.yaml`

    b. 对于提交消息，请保留默认消息或输入您自己的消息。

    c. 对于存储库和分支，为工作流定义文件选择源存储库和分支。这些字段应设置为您之前在创建工作流对话框中指定的存储库和分支。如果愿意，您现在可以更改存储库和分支。

> **Note**
>
> 提交工作流定义文件后，该文件无法与其他存储库或分支关联，因此请务必谨慎地选择存储库和分支。

d. 选择提交以提交工作流定义文件。

YAML

使用 YAML 编辑器创建工作流

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择创建工作流。

   此时将显示创建工作流对话框。
5. 在源存储库字段中，选择工作流定义文件所在的源存储库。如果源存储库不存在，请创建一个存储库。
6. 在分支字段中，选择工作流定义文件所在的分支。
7. 选择创建。

   Amazon CodeCatalyst 将存储库和分支信息保存在内存中，但工作流程尚未提交。
8. 选择 YAML。
9. 构建工作流：

   a. （可选）在 YAML 代码中添加触发器。有关更多信息，请参阅添加触发器到工作流。

   b. 选择 + 操作（左上角）。此时将出现操作目录。

   c. 选择某个操作中的加号（+）可将该操作添加到工作流中。使用右侧的窗格配置操作。有关更多信息，请参阅添加操作到工作流。

   d. （可选）选择工作流属性（右上角）。此时将出现工作流属性窗格。配置工作流名称、运行模式和计算。有关更多信息，请参阅配置运行的排队行为和配置计算和运行时映像。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，然后在提交工作流对话框中执行以下操作：

a. 对于工作流文件名，保留默认名称或输入您自己的名称。文件将存储在所选源存储库和分支的 ~/.codecatalyst/workflows/ 文件夹中。您可以在文件名前面加上文件夹或子文件夹。示例：

   - 指定 my-workflow（无文件夹）会将文件存储为 ~/.codecatalyst/workflows/my-workflow.yaml
   - 指定 folder/subfolder/my-workflow 会将文件存储为 ~/.codecatalyst/workflows/folder/subfolder/my-workflow.yaml

b. 对于提交消息，请保留默认消息或输入您自己的消息。

c. 对于存储库和分支，为工作流定义文件选择源存储库和分支。这些字段应设置为您之前在创建工作流对话框中指定的存储库和分支。如果愿意，您现在可以更改存储库和分支。

> **Note**
>
> 提交工作流定义文件后，该文件无法与其他存储库或分支关联，因此请务必谨慎地选择存储库和分支。

d. 选择提交以提交工作流定义文件。

运行工作流

运行是一个工作流的单次迭代。在运行期间， CodeCatalyst执行工作流配置文件中定义的操作并输出关联的日志、构件和变量。

您可以手动启动运行，也可以通过工作流触发器来自动启动运行。工作流触发器的一个例子是软件开发人员将提交推送到您的主分支。

如果您错误地启动了工作流，也可以在工作流的处理过程中手动停止工作流运行。

如果工作流运行在大约相同的时间启动，您可以配置这些运行的排队方式。您可以使用默认排队行为，即运行按启动顺序一个接一个地排队，也可以让较晚的运行取代（或“接管”），以加快整个运行的速度。您可以将工作流设置为并行运行，这样就不用等待任何其他运行。

手动或自动启动工作流运行后，您可以查看运行的状态和其他详细信息。例如，您可以查看运行何时启动、谁启动的运行以及是否仍在运行中。

主题

手动启动工作流运行

在 Amazon 中 CodeCatalyst，您可以从 CodeCatalyst 控制台手动启动工作流程。

有关工作流运行的更多信息，请参阅运行工作流。

> **Note**
>
> 您也可以通过配置触发器来自动启动工作流运行。

手动启动工作流运行

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择运行。

使用触发器自动启动工作流运行

您可以使用 CodeCatalyst 工作流程触发器自动启动 Amazon 工作流程。

工作流触发器简称触发器，使您可以在发生某些事件（例如代码推送）时自动启动工作流运行。您可能需要配置触发器，使软件开发人员不必通过 CodeCatalyst 控制台手动启动工作流程。

您可以使用三种类型的触发器：

- 推送 – 每当推送提交时，代码推送触发器都会启动工作流运行。
- 拉取请求 – 每当创建、修改或关闭拉取请求时，拉取请求触发器都会启动工作流运行。
- 计划 – 计划触发器使工作流运行按您定义的计划启动。您可以考虑使用计划触发器在夜间运行软件的版本，这样在第二天早上可以准备好最新的版本供开发人员处理。

您可以单独使用推送、拉取请求和计划触发器，也可以在同一个工作流中组合使用这些触发器。

触发器是可选的，如果您未配置任何触发器，则只能手动启动工作流。

> **Tip**
>
> 要查看触发器的实际作用，请启动带有蓝图的项目。大多数蓝图都包含带有触发器的工作流。在蓝图的工作流定义文件中查找 `Trigger` 属性。有关蓝图的更多信息，请参阅使用蓝图创建项目。

主题

示例：工作流中的触发器

以下示例演示如何在 Amazon CodeCatalyst 工作流定义文件中添加不同类型的触发器。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

主题

- 示例：带有计划、推送和分支的触发器
- 示例：带有拉取和分支的触发器
- 示例：带有拉取、分支和“CLOSED”事件的触发器
- 示例：带有推送、分支和文件的触发器
- 示例：手动触发器
- 示例：CI/CD 多工作流设置中的触发器

示例：一个简单的代码推送触发器

以下示例显示了一个触发器，在代码被推送到源存储库中的任何分支时，该触发器将启动工作流运行。

激活此触发器后，CodeCatalyst 会使用您要推送至的分支（即目标分支）中的文件启动工作流运行。

例如，如果您将提交推送到 `main`，则 CodeCatalyst 会使用 `main` 上的工作流定义文件和其他源文件启动工作流运行。

再举一个例子，如果您将提交推送到 `feature-branch-123`，则 CodeCatalyst 会使用 `feature-branch-123` 上的工作流定义文件和其他源文件启动工作流运行。

```
Triggers:
  - Type: PUSH
```

> **Note**
>
> 如果您希望只有在推送到 `main` 时才启动工作流运行，请参阅示例：一个简单的“push to main”触发器。

示例：一个简单的“push to main”触发器

以下示例显示了一个触发器，在代码被推送到源存储库中的 `main` 分支（而且仅在推送到 `main` 分支）时，该触发器将启动工作流运行。

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

示例：一个简单的拉取请求触发器

以下示例显示了一个触发器，在源存储库中创建或修订了任何拉取请求时，该触发器将启动工作流运行。

激活此触发器后，CodeCatalyst 会使用您拉取自的分支（即源分支）中的工作流定义文件和其他源文件启动工作流运行。

例如，如果您使用名为 `feature-123` 的源分支和名为 `main` 的目标分支创建拉取请求，则 CodeCatalyst 会使用 `feature-123` 上的工作流定义文件和其他源文件启动工作流运行。

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
      - REVISION
```

示例：一个简单的计划触发器

以下示例演示了在每星期一到星期五的午夜（UTC+0）启动工作流运行的触发器。

激活此触发器后，对于源存储库中包含带有此触发器的工作流定义文件的每个分支，CodeCatalyst 将启动一个工作流运行。

例如，如果您的源存储库中有 `main`、`release-v1`、`feature-123` 三个分支，并且每个分支都包含一个带有触发器的工作流定义文件，则 CodeCatalyst 会启动三个工作流运行：一个使用 `main` 中的文件，一个使用 `release-v1` 中的文件，另一个使用 `feature-123` 中的文件。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 ? * MON-FRI *"
```

有关可在 `Expression` 属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有计划和分支的触发器

以下示例演示了在每天下午 6:15（UTC+0）启动工作流运行的触发器。

激活此触发器后，CodeCatalyst 会使用 `main` 分支中的文件启动工作流运行，并对以 `release-` 开头的每个分支启动额外的运行。

例如，如果您的源存储库中有名为 main、release-v1、bugfix-1 和 bugfix-2 的分支，CodeCatalyst 会启动两个工作流运行：一个使用 main 中的文件，另一个使用 release-v1 中的文件。它不会为 bugfix-1 和 bugfix-1 分支启动工作流运行。

```
Triggers:
  - Type: SCHEDULE
    Expression: "15 18 * * ? *"
    Branches:
      - main
      - release\-.*
```

有关可在 Expression 属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有计划、推送和分支的触发器

以下示例演示了一个触发器，该触发器在每天午夜（UTC+0）以及每当有代码推送到 main 分支时启动工作流运行。

在本示例中：

- 工作流运行在每天午夜启动。工作流运行使用 main 分支中的工作流定义文件和其他源文件。
- 每当您将提交推送到 main 分支时，也会启动工作流运行。该工作流运行使用目标分支（main）中的工作流定义文件和其他源文件。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 * * ? *"
    Branches:
      - main
  - Type: PUSH
    Branches:
      - main
```

有关可在 Expression 属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有拉取和分支的触发器

以下示例演示了一个触发器，每当有人对名为 main 的目标分支打开或修改拉取请求时，该触发器就会启动工作流运行。尽管 Triggers 配置中指定的分支是 main，但工作流运行将使用源分支（即您拉取自的分支）中的工作流定义文件和其他源文件。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

示例：带有拉取、分支和“CLOSED”事件的触发器

以下示例演示了一个触发器，每当有人对以 main 开头的分支关闭拉取请求时，该触发器就会启动工作流运行。

在本示例中：

- 当您关闭对以 main 开头的目标分支的拉取请求时，就会自动启动工作流运行，该工作流将使用（现已关闭）源分支中的工作流定义文件和其他源文件。
- 如果您已将源存储库配置为在合并拉取请求后自动删除分支，则这些分支将永远没有机会进入 CLOSED 状态。这意味着合并的分支不会激活拉取请求 CLOSED 触发器。在这种情况下，激活 CLOSED 触发器的唯一方法是关闭拉取请求而不合并。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - CLOSED
```

示例：带有推送、分支和文件的触发器

以下示例演示了一个触发器，每当对 main 分支上的 filename.txt 文件或 src 目录中的任何文件进行更改时，该触发器就会启动工作流运行。

激活此触发器后，CodeCatalyst 会使用 main 分支中的工作流定义文件和其他源文件启动工作流运行。

```
Triggers:
  - Type: PUSH
    Branches:
```

```
      - main
    FilesChanged:
      - filename.txt
      - src\/.*
```

示例：手动触发器

要配置手动触发器，请在工作流定义文件中省略 `Triggers` 部分。如果没有此部分，用户将被强制要求通过在 CodeCatalyst 控制台中选择运行按钮来手动启动工作流。有关更多信息，请参阅手动启动工作流运行。

示例：CI/CD 多工作流设置中的触发器

此示例介绍当您想要使用单独的 Amazon CodeCatalyst 工作流进行持续集成（CI）和持续部署（CD）时，如何设置触发器。

在此场景中，您设置两个工作流：

- CI 工作流 – 在创建或修订拉取请求时，此工作流会构建和测试您的应用程序。
- CD 工作流 – 在合并拉取请求时，此工作流会构建和部署您的应用程序。

CI 工作流的定义文件如下所示：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
Actions:
  BuildAction:
    instructions-for-building-the-app
  TestAction:
    instructions-for-test-the-app
```

`Triggers` 代码表明，每当软件开发人员创建拉取请求（或修改拉取请求）来要求将其功能分支合并到分支 `main` 时，就会自动启动工作流运行。CodeCatalyst 使用源分支（即功能分支）中的源代码启动工作流运行。

CD 工作流的定义文件如下所示：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildAction:
    instructions-for-building-the-app
  DeployAction:
    instructions-for-deploying-the-app
```

Triggers 代码指示在出现合并到 main 时自动启动工作流。CodeCatalyst 使用 main 分支中的源代码启动工作流运行。

触发器和分支的使用准则

本节介绍在设置包含分支的 Amazon CodeCatalyst 触发器时的一些主要准则。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

- 准则 1：对于推送和拉取请求触发器，如果要指定分支，您必须在触发器配置中指定目标（即“至”）分支。切勿指定源（即“自”）分支。

  在以下示例中，从任何分支推送至 main 都会激活工作流。

  ```
  Triggers:
    - Type: PUSH
      Branches:
        - main
  ```

  在以下示例中，自任何分支拉取请求到 main 中都会激活工作流。

  ```
  Triggers:
    - Type: PULLREQUEST
      Branches:
        - main
      Events:
        - OPEN
        - REVISION
  ```

- 准则 2：对于推送触发器，激活工作流后，工作流将使用目标分支中的工作流定义文件和源文件运行。

- 准则 3：对于拉取请求触发器，激活工作流后，工作流将使用源分支中的工作流定义文件和源文件运行（即使您在触发器配置中指定了目标分支）。
- 准则 4：完全相同的触发器，在一个分支中能够运行，但在另一个分支中可能无法运行。

  请考虑以下推送触发器：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

  如果包含此触发器的工作流定义文件存在于 `main` 中，然后被克隆到 `test`，则工作流永远不会使用 `test` 中的文件自动启动（尽管您可以手动启动工作流以使其使用 `test` 中的文件）。请查看准则 2 来了解为什么工作流永远不会使用 `test` 中的文件自动运行。

  还要考虑以下拉取请求触发器：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

  如果包含此触发器的工作流定义文件位于 `main` 中，则工作流将永远不会使用 `main` 中的文件运行。（但是，如果您在 `main` 中创建 `test` 分支，则工作流将使用 `test` 中的文件运行。）请查看准则 3 以了解原因。

添加触发器到工作流

按照以下说明在您的 Amazon CodeCatalyst 工作流程中添加推送、拉取或计划触发器。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

Visual

添加触发器（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择源和触发器框。
8. 在配置窗格中，选择添加触发器。
9. 在添加触发器对话框中，在字段中提供信息，如下所示。

   触发器类型

   指定触发器的类型。可以使用下列值之一：

   - 推送（可视化编辑器）或 PUSH（YAML 编辑器）

     当更改推送到源存储库时，推送触发器会启动工作流运行。工作流运行将使用您推送到的分支（即目标分支）中的文件。
   - 拉取请求（可视化编辑器）或 PULLREQUEST（YAML 编辑器）

     在源存储库中打开、更新或关闭拉取请求时，拉取请求触发器会启动工作流运行。工作流运行将使用您拉取自的分支（即源分支）中的文件。
   - 计划（可视化编辑器）或 SCHEDULE（YAML 编辑器）

     计划触发器按您指定的 cron 表达式定义的计划来启动工作流运行。对于源存储库中的每个分支，将使用分支的文件启动单独的工作流运行。[要限制在其上激活触发器的分支，请使用分支字段（可视化编辑器）或 `Branches` 属性（YAML 编辑器）。]

     配置计划触发器时，请遵循以下指南：

     - 每个工作流只能使用一个计划触发器。
     - 如果您在自己的 CodeCatalyst 空间中定义了多个工作流程，我们建议您安排的同时启动不超过 10 个工作流程。
     - 确保将触发器的 cron 表达式配置为在两次运行之间留出足够的时间。有关更多信息，请参阅Expression。

   有关示例，请参阅 示例：工作流中的触发器。

拉取请求的事件

仅当您选择了拉取请求触发器类型时，此字段才会显示。

指定将启动工作流运行的拉取请求事件的类型。有效值如下所示：

- 拉取请求已创建（可视化编辑器）或 OPEN（YAML 编辑器）

  工作流运行将在拉取请求创建后启动。

- 拉取请求已关闭（可视化编辑器）或 CLOSED（YAML 编辑器）

  工作流运行将在拉取请求关闭后启动。CLOSED 事件的行为不太好说明，最好通过一个例子来理解。请参阅示例：带有拉取、分支和“CLOSED”事件的触发器了解更多信息。

- 对拉取请求进行了新的修订（可视化编辑器）或 REVISION（YAML 编辑器）

  工作流运行将在对拉取请求的修订创建后启动。在创建拉取请求时，将创建第一个修订。之后，每当有人将新的提交推送到拉取请求中指定的源分支时，就会创建一个新的修订。如果您在拉取请求触发器中包含 REVISION 事件，则可以忽略 OPEN 事件，因为 REVISION 是 OPEN 的超集。

您可以在同一个拉取请求触发器中指定多个事件。

有关示例，请参阅 示例：工作流中的触发器。

计划

仅当您选择了计划触发器类型时，此字段才会显示。

指定 cron 表达式，用于描述您希望何时执行计划的工作流运行。

中的 Cron 表达式 CodeCatalyst 使用以下六字段语法，其中每个字段用空格分隔：

minutes hours days-of-month month days-of-week year

cron 表达式示例

分钟	小时	一个月中的第几天	Month	星期几	年	意义
0	0	?	*	MON-FRI	*	每个星期一到星期五的午夜（UTC+0）运行工作流。
0	2	*	*	?	*	每天凌晨 2:00（UTC+0）运行工作流。
15	22	*	*	?	*	每天晚上 10:15（UTC+0）运行工作流。
0/30	22-2	?	*	SAT-SUN	*	星期六至星期日，从起始日晚上 10:00 至次日凌晨 2:00（UTC+0）每隔 30 分钟运行一次工作流。

分钟	小时	一个月中的第几天	Month	星期几	年	意义
45	13	L	*	?	2023-2027	2023 年至 2027 年（含）之间，在每个月的最后一天下午 1:45（UTC +0）运行工作流。

在中指定 cron 表达式时 CodeCatalyst，请务必遵循以下准则：

- 为每个 SCHEDULE 触发器指定一个 cron 表达式。
- 在 YAML 编辑器中，将 cron 表达式用双引号（"）括起来。
- 使用协调世界时（UTC）格式指定时间。不支持其他时区。
- 两次运行之间应至少配置 30 分钟的运行间隔。不支持更快的节奏。
- 指定*days-of-month*或字*days-of-week*段，但不能同时指定两者。如果您在其中一个字段中指定值或星号（*），则必须在另一个字段中使用问号（？）。星号表示“全部”，问号表示“任何”。

有关 cron 表达式的更多示例以及有关通配符（如?、和）的信息 *L，请参阅《亚马逊 EventBridge 用户指南》中的 Cron 表达式参考。 EventBridge 和中的 Cron 表达式 CodeCatalyst 的工作方式完全相同。

有关计划触发器的示例，请参阅示例：工作流中的触发器。

分支和分支模式

（可选）

指定触发器监控的源存储库中的分支，以便知道何时启动工作流运行。您可以使用正则表达式模式来定义分支名称。例如，使用 `main.*` 来匹配以 `main` 开头的所有分支。

根据触发器的类型，要指定的分支会有所不同：

- 对于推送触发器，请指定要推送至的分支，即目标分支。对于每个匹配的分支，将使用匹配分支中的文件，启动一个工作流运行。

  示例：`main.*`、`mainline`
- 对于拉取请求触发器，请指定要推送至的分支，即目标分支。对于每个匹配的分支，将使用工作流定义文件和源分支（不是匹配的分支）中的文件，启动一个工作流运行。

  示例：`main.*`、`mainline`、`v1\-.*`（匹配以 `v1-` 开头的分支）
- 对于计划触发器，请指定包含您希望计划运行使用的文件的分支。对于每个匹配的分支，将使用工作流定义文件和匹配分支中的源文件，启动一个工作流运行。

  示例：`main.*`、`version\-1\.0`

> **Note**
>
> 如果您未指定分支，则触发器会监控源存储库中的所有分支，并将使用以下位置中的工作流定义文件和源文件启动工作流运行：
>
> - 您要推送至的分支（对于推送触发器）。有关更多信息，请参阅示例：一个简单的代码推送触发器。
> - 您要拉取自的分支（对于拉取请求触发器）。有关更多信息，请参阅示例：一个简单的拉取请求触发器。
> - 所有分支（对于计划触发器）。源存储库中的每个分支将启动一个工作流运行。有关更多信息，请参阅示例：一个简单的计划触发器。

有关分支和触发器的更多信息，请参阅触发器和分支的使用准则。

有关更多示例，请参阅示例：工作流中的触发器。

文件已更改

仅当您选择了推送或拉取请求触发器类型时，才会显示此字段。

指定触发器监控的源存储库中的文件或文件夹，以便知道何时启动工作流运行。您可以使用正则表达式来匹配文件名或路径。

有关示例，请参阅 示例：工作流中的触发器。

10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

添加触发器（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 根据以下示例的指导，添加 `Triggers` 部分和底层属性。有关更多信息，请参阅 工作流 YAML 定义中的Triggers。

   代码推送触发器类似于下文：

   ```
   Triggers:
     - Type: PUSH
       Branches:
         - main
   ```

   拉取请求触发器类似于下文：

   ```
   Triggers:
     - Type: PULLREQUEST
       Branches:
         - main.*
       Events:
         - OPEN
         - REVISION
         - CLOSED
   ```

   计划触发器类似于下文：

```
Triggers:
  - Type: SCHEDULE
    Branches:
      - main.*
    # Run the workflow at 1:15 am (UTC+0) every Friday until the end of 2023
    Expression: "15 1 ? * FRI 2022-2023"
```

有关可在 Expression 属性中使用的 cron 表达式的更多示例，请参阅Expression。

有关推送、拉取请求和计划触发器的更多示例，请参阅示例：工作流中的触发器。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

配置仅限手动触发的触发器

您可以限制工作流程，使其只能由您的团队使用 CodeCatalyst 控制台中的 “运行” 按钮手动启动。要配置此功能，您必须删除工作流定义文件中的 Triggers 部分。创建工作流默认包含 Triggers 部分，但该部分是可选的，可以删除。

按照以下说明删除工作流定义文件中的 Triggers 部分，使得工作流只能手动启动。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

有关运行工作流的更多信息，请参阅运行工作流。

Visual

删除“触发器”部分（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中选择源框。

8. 在触发器下，选择垃圾桶图标以将 `Triggers` 部分从工作流中删除。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

YAML

删除“触发器”部分（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 找到 `Triggers` 部分并将其删除。
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

停止工作流运行

使用以下过程停止正在进行的工作流运行。如果您意外启动了运行，则可能需要停止运行。

停止工作流程运行时， CodeCatalyst 等待正在进行的操作完成，然后才会在控制台中将运行标记为“已停止” CodeCatalyst 。任何还没有机会启动的操作都不会启动，而且会被标记为已放弃。

> **Note**
>
> 如果运行已排队（也就是说，没有正在进行的操作），则运行将立即停止。

有关工作流运行的更多信息，请参阅运行工作流。

停止工作流运行

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 在工作流下，选择运行，然后从列表中选择正在进行的运行。
5. 选择停止。

用阶段门控制工作流运行

阶段门是一个工作流组件，您可以用它来要求工作流必须满足特定条件才能继续运行。例如，批准门禁是允许工作流程继续运行之前，用户必须在 CodeCatalyst控制台中提交批准。

您可以在工作流中的操作序列之间或在第一个操作（源下载后立即运行）之前添加阶段门。如果需要的话，您也可以在最后一个操作之后添加阶段门。

有关工作流运行的更多信息，请参阅运行工作流。

主题

阶段门类型

目前，Amazon CodeCatalyst 支持一种门禁：批准门。有关更多信息，请参阅 要求批准工作流运行。

我可以设置阶段门与其他操作并行运行吗？

不能。阶段门只能在操作之前或之后运行。有关更多信息，请参阅按顺序执行阶段门和操作。

我能否使用阶段门来阻止工作流运行的启动？

可以，有资格要求。

您可以阻止工作流运行去执行任务，这与阻止其启动略有不同。

要阻止工作流执行任务，请在工作流中的第一个操作之前添加阶段门。在这种情况下，工作流运行将启动，这意味着它将下载您的源存储库文件，但在阶段门解锁之前，它无法执行任务。

> **Note**
>
> 在启动后被阶段门阻止的工作流仍计入每个空间的并行工作流运行最大数量配额和其他配额。为确保您不会超过工作流配额，请考虑使用工作流触发器来有条件地启动工作流，而不是使用阶段门。还可以考虑使用拉取请求批准规则代替阶段门。有关配额、触发器和拉取请求批准规则的更多信息，请参阅 中的工作流程配额 CodeCatalyst、使用触发器自动启动工作流运行和管理将拉取请求与审批规则合并的要求。

阶段门的限制

阶段门具有以下限制：

- 阶段门不能与计算共享功能结合使用。有关此特征的更多信息，请参阅跨操作共享计算。
- 阶段门不能在操作组中使用。有关操作组的更多信息，请参阅将操作分组为操作组。

向工作流添加阶段门

在 Amazon CodeCatalyst 中，您可以向工作流添加阶段门，让工作流在满足某些条件的情况下才能运行。按照以下说明向工作流添加阶段门。

有关阶段门的更多信息，请参阅用阶段门控制工作流运行。

添加和配置阶段门

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。

7. 在左侧选择阶段门。
8. 在阶段门目录中，搜索阶段门，然后选择加号（+），将该阶段门添加到您的工作流中。
9. 配置阶段门。选择可视化以使用可视化编辑器，或者选择 YAML 以使用 YAML 编辑器。有关详细说明，请参阅：

   - 添加“批准”阶段门

10. （可选）选择验证以确保 YAML 代码有效。
11. 选择提交以提交您的更改。

按顺序执行阶段门和操作

在 Amazon CodeCatalyst 中，您可以将某个阶段门设置为在工作流操作、操作组或阶段门之前或之后运行。例如，您可以将 Approval 阶段门设置为在 Deploy 操作之前运行。在这种情况下，Deploy 操作可以说是依赖于 Approval 阶段门。

要在阶段门和操作之间设置依赖关系，请配置阶段门或操作的依赖于属性。有关说明，请参阅设置操作之间的依赖关系。所引用的说明介绍的是工作流操作，但同样适用于阶段门。

有关如何设置阶段门依赖于属性的示例，请参阅示例：“批准”阶段门。

有关阶段门的更多信息，请参阅用阶段门控制工作流运行。

有关工作流操作的更多信息，请参阅配置工作流操作。

指定阶段门的版本

默认情况下，当您向工作流添加阶段门时，CodeCatalyst 会使用以下格式将完整版本添加到工作流定义文件中：

v*major*.*minor*.*patch*

例如：

```
My-Gate:
  Identifier: aws/approval@v1
```

您可以使用加长版本，使工作流使用阶段门的特定主要或次要版本。有关说明，请参阅指定要使用的操作版本。所引用的主题说明的是工作流操作，但同样适用于阶段门。

有关 CodeCatalyst 中阶段门的更多信息，请参阅用阶段门控制工作流运行。

要求批准工作流运行

您可以将工作流运行配置为需要先获得批准，然后才能继续。为此，您必须在工作流中添加批准阶段门。批准阶段门可在一个用户或一组用户在 CodeCatalyst 控制台中提交一个或多个批准之前，阻止工作流的继续。在提供了所有的批准后，阶段门将“解锁”，允许恢复工作流运行。

在工作流中使用批准阶段门，可以让开发、运营和领导团队有机会先对更改进行审查，然后再面向更广泛的受众进行部署。

有关工作流运行的更多信息，请参阅运行工作流。

主题

如何解锁批准阶段门？

要解锁批准阶段门，必须满足以下所有条件：

- 条件 1：必须提交所需数量的批准。所需的批准数量是可配置的，并且每个用户只允许提交一次批准。
- 条件 2：所有批准必须在阶段门超时之前提交。阶段门在激活 14 天后超时。此时间段不可配置。
- 条件 3：没有任何人拒绝工作流运行。一个拒绝就会导致工作流运行失败。
- 条件 4：（仅在使用取代运行模式时适用。）该运行不得被以后的运行所取代。有关更多信息，请参阅对于排队、取代和并行运行模式，工作流批准如何运行？。

如果有任何一个条件未满足，CodeCatalyst 将停止工作流并将运行状态设置为失败（不满足条件 1 到 3 的情况）或取代（不满足条件 4 的情况下）。

何时使用“批准”阶段门

通常，在工作流将应用程序和其他资源部署到生产服务器或任何必须验证质量标准的环境中时，您可以在工作流中使用批准阶段门。通过在部署到生产环境之前放置阶段门，您向审阅者提供了机会，在新的软件修订版面向公众发布之前对其进行验证。

谁可以提供批准？

任何用户，只要是您的项目的成员且具有贡献者或项目管理员角色，就可以提供批准。具有空间管理员角色且属于您的项目空间的用户也可以提供批准。

> **Note**
>
> 具有审阅者角色的用户无法提供批准。

如何通知用户需要该用户的批准？

要通知用户需要该用户的批准，您必须：

- 让 CodeCatalyst 向用户发送 Slack 通知。有关更多信息，请参阅配置批准通知。
- 转到 CodeCatalyst 控制台中提供了批准和拒绝按钮的页面，然后将该页面的 URL 粘贴到发给审批者的电子邮件或消息收发应用程序中。有关如何导航到此页面的更多信息，请参阅批准或拒绝工作流运行。

我能否使用“批准”阶段门来阻止工作流运行的启动？

可以，有资格要求。有关更多信息，请参阅我能否使用阶段门来阻止工作流运行的启动？。

对于排队、取代和并行运行模式，工作流批准如何运行？

在排队、取代或并行运行模式时，批准阶段门的工作方式与操作类似。我们建议您阅读关于排队运行模式、关于取代运行模式、关于并行运行模式部分以熟悉这些运行模式。在对这些模式有基本的了解后，请返回本节，了解当存在批准阶段门时，这些运行模式是如何工作的。

存在批准阶段门时，将按以下方式处理运行：

- 如果您使用排队运行模式，则运行将排队在当前正等待阶段门批准的运行之后。当该阶段门解锁（即所有批准都已给出）时，队列中的下一个运行将进入阶段门，等待批准。此过程继续进行，排队的运行逐个通过阶段门进行处理。Figure 1 说明了这个过程。
- 如果您使用取代运行模式，则其行为与排队运行模式相同，不同之处在于较新的运行不会堆积在阶段门的队列中，而是取代（接管）较早的运行。这里没有队列，任何目前在阶段门等待批准的运行都将被取消并被更新的运行所取代。Figure 2 说明了这个过程。
- 如果您使用并行运行模式，则运行将并行启动，不会形成队列。由于每个运行的前面都没有其他运行，因此阶段门会立即处理每个运行。Figure 3 说明了这个过程。

图 1：“排队运行模式”和批准阶段门

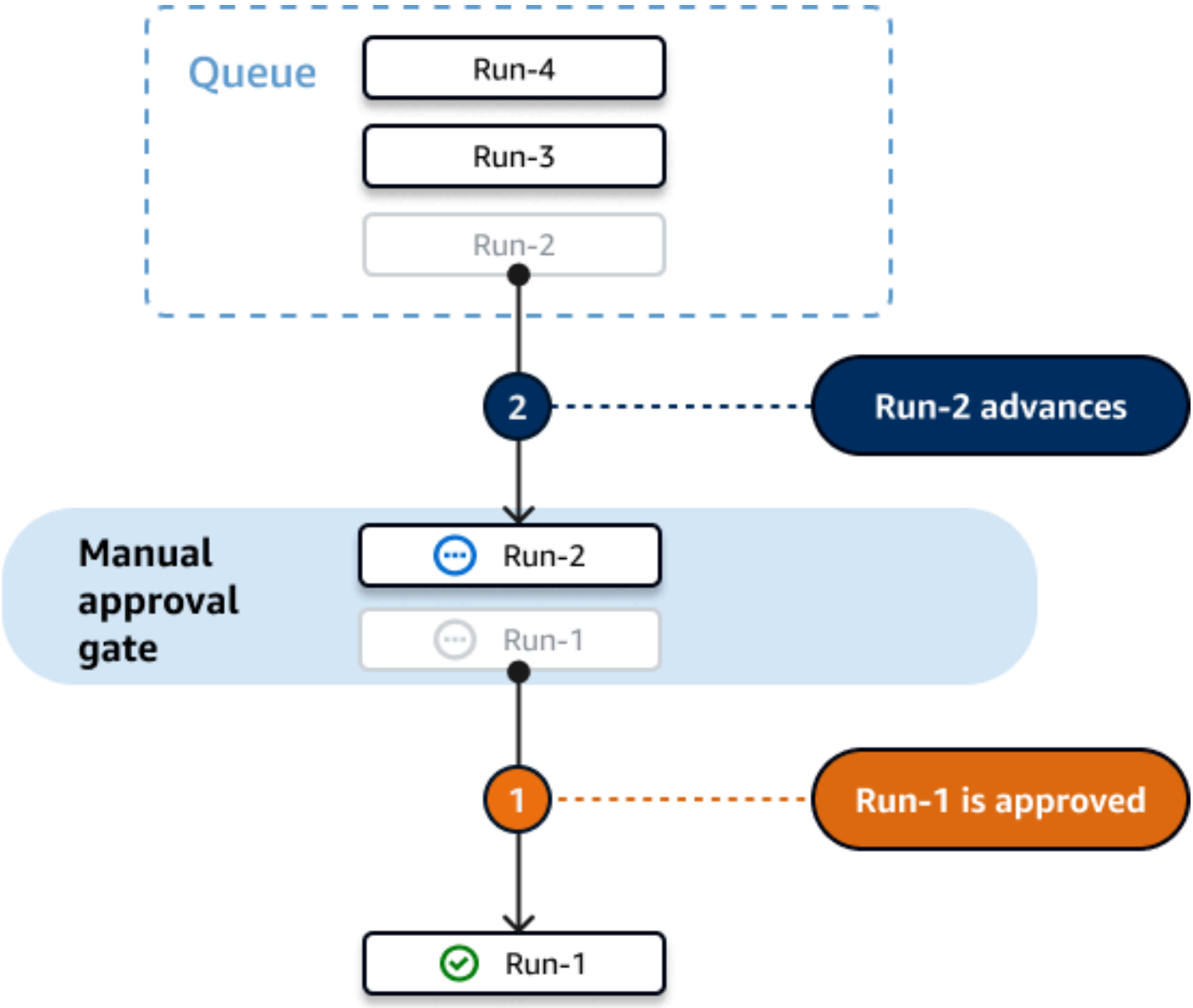


图 2：“取代运行模式”和批准阶段门

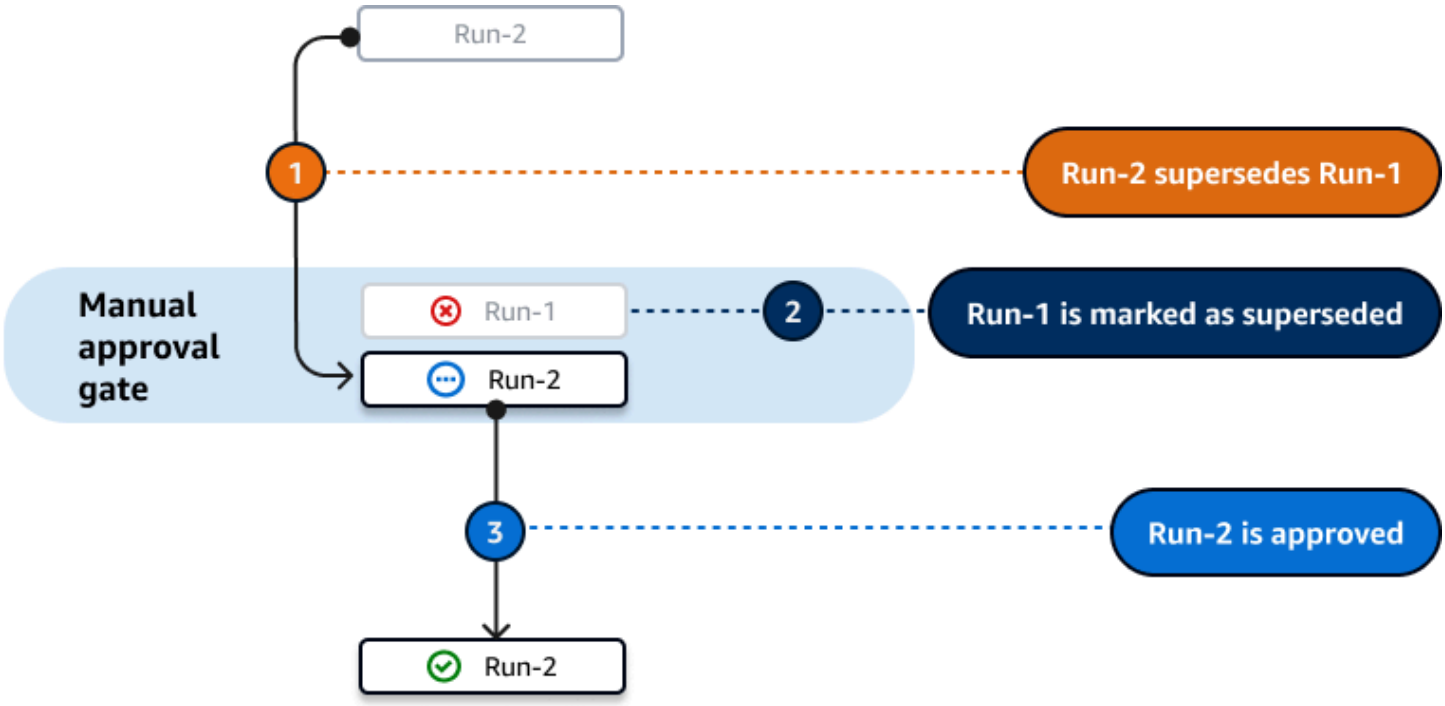
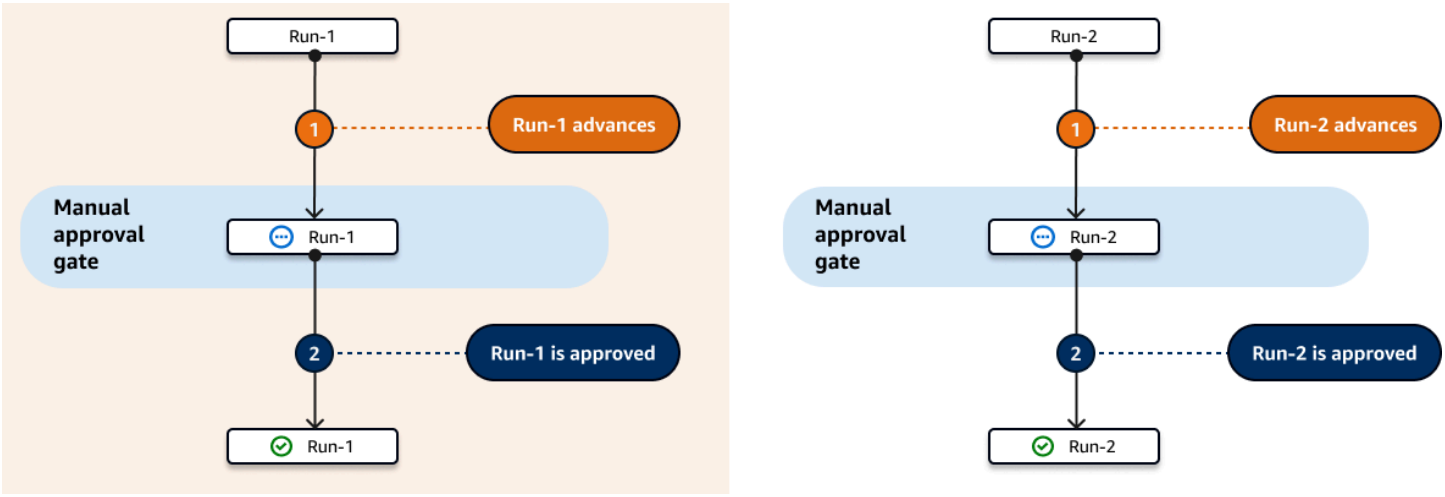


图 3：“并行运行模式”和批准阶段门



示例：“批准”阶段门

以下示例说明如何在名为 Staging 和 Production 的两个操作之间添加一个名为 Approval_01 的批准阶段门。Staging 操作首先运行，其次是 Approval_01 阶段门，最后是 Production 操作。只有在 Approval_01 阶段门解锁时才会执行 Production 操作。DependsOn 属性可确保 Staging、Approval_01 和 Production 阶段按顺序运行。

有关批准阶段门的更多信息，请参阅要求批准工作流运行。

```
Actions:
  Staging: # Deploy to a staging server
    Identifier: aws/ecs-deploy@v1
    Configuration:
    ...
  Approval_01:
    Identifier: aws/approval@v1
    DependsOn:
      - Staging
    Configuration:
      ApprovalsRequired: 2
  Production: # Deploy to a production server
    Identifier: aws/ecs-deploy@v1
    DependsOn:
      - Approval_01
    Configuration:
    ...
```

添加“批准”阶段门

要将工作流配置为需要批准，您必须在工作流中添加批准阶段门。按照以下说明向您的工作流添加批准阶段门。

有关此阶段门的更多信息，请参阅要求批准工作流运行。

Visual

向工作流添加“批准”阶段门（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 在左上角，选择阶段门。
7. 在阶段门目录的批准中，选择加号（+）。
8. 选择输入，然后在依赖于字段中执行以下操作。

指定必须成功运行才能使该阶段门运行的操作、操作组或阶段门。默认情况下，当您向工作流添加阶段门时，阶段门的设置取决于工作流中的最后一个操作。如果删除此属性，则阶段门将不依赖于任何设置，并且将首先运行，然后再执行其他操作。

> **Note**
>
> 阶段门必须配置为在操作、操作组或阶段门之前或之后运行。阶段门无法设置为与其他操作、操作组和阶段门并行运行。

有关依赖于功能的更多信息，请参阅 按顺序执行阶段门和操作。

9. 选择配置选项卡。
10. 在阶段门名称字段中，执行以下操作。

    指定要为阶段门指定的名称。在工作流中，所有阶段门名称都必须唯一。阶段门名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。您不能使用引号在阶段门名称中启用特殊字符和空格。
11. （可选）在批准数字段中，执行以下操作。

    指定解锁批准阶段门所需的最小审批数。最小值为 1。最大值为 2。如果省略，则默认值为 1。

    > **Note**
    >
    > 如果要省略 `ApprovalsRequired` 属性，请从工作流定义文件中删除该阶段门的 `Configuration` 部分。
12. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
13. 选择提交，输入提交消息，然后再次选择提交。

YAML

向工作流添加“批准”阶段门（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。

3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 根据以下示例的指导，添加 Approval 部分和底层属性。有关更多信息，请参阅 工作流 YAML 定义 中的 “批准”阶段门 YAML。

```
Actions:
  MyApproval_01:
    Identifier: aws/approval@v1
    DependsOn:
      - PreviousAction
    Configuration:
      ApprovalsRequired: 2
```

有关另一个示例，请参阅示例：“批准”阶段门。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

配置批准通知

您可以让 CodeCatalyst 向 Slack 频道发送通知，通知用户有工作流运行需要批准。用户看到通知并单击其中的链接。该链接将他们转到 CodeCatalyst 批准页面，他们可以在页面中批准或拒绝工作流。

您还可以配置通知来通知用户，工作流已获得批准、已被拒绝或批准请求已过期。

按照以下说明设置 Slack 通知。

开始前的准备工作

确保已在工作流中添加了批准阶段门。有关更多信息，请参阅添加“批准”阶段门。

向 Slack 频道发送工作流批准通知

1. 使用 Slack 配置 CodeCatalyst。有关更多信息，请参阅开始使用 Slack 通知。
2. 在包含需要批准的工作流的 CodeCatalyst 项目中，如果尚未启用通知则启用。要启用通知，请执行以下操作：

a. 导航到您的项目，然后在导航窗格中，选择项目设置。
b. 在顶部，选择通知。
c. 在通知事件中，选择编辑通知。
d. 开启待处理的工作流批准，然后选择一个 CodeCatalyst 发送通知的 Slack 频道。
e. （可选）开启其他通知，提醒用户有关已批准、已拒绝和已过期批准的情况。您可以启用工作流运行已批准、工作流运行已拒绝、工作流批准已取代和工作流批准已超时通知。在每条通知旁边，选择 CodeCatalyst 将发送通知的 Slack 频道。
f. 选择保存。

批准或拒绝工作流运行

包含批准阶段门的工作流运行将需要得到批准或拒绝。用户可以通过以下方式提供批准或拒绝：

- CodeCatalyst 控制台
- 团队成员提供的链接
- 自动发送的 Slack 通知

在用户提供了批准或拒绝后，此决定无法撤消。

> **Note**
>
> 只有某些用户可以批准或拒绝工作流运行。有关更多信息，请参阅谁可以提供批准？。

开始前的准备工作

确保已在工作流中添加了批准阶段门。有关更多信息，请参阅添加“批准”阶段门。

从 CodeCatalyst 控制台批准或拒绝启动工作流运行

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

5. 在工作流图表中，选择代表批准阶段门的方框。

   侧面板显示。

   > **Note**
   >
   > 此时，您可以根据需要将此页面的 URL 发送给其他审批者。

6. 在审阅决定下，选择批准或拒绝。
7. （可选）在评论 – 可选中，输入评论，说明您批准或拒绝工作流运行的原因。
8. 选择提交。

从团队成员提供的链接批准或拒绝启动工作流运行

1. 选择团队成员发送给您的链接。（您可以让团队成员阅读前面的步骤以获取链接。）
2. 在系统要求时，登录 CodeCatalyst。

   您将被重定向到工作流运行批准页面。
3. 在审阅决定下，选择批准或拒绝。
4. （可选）在评论 – 可选中，输入评论，说明您批准或拒绝工作流运行的原因。
5. 选择提交。

从自动化 Slack 通知批准或拒绝启动工作流运行

1. 确保已设置 Slack 通知。请参阅配置批准通知。
2. 在 Slack 中，在发送批准通知的频道中，选择批准通知中的链接。
3. 在系统要求时，登录 CodeCatalyst。

   您将被重定向到工作流运行页面。
4. 在工作流图表中，选择批准阶段门。
5. 在审阅决定下，选择批准或拒绝。
6. （可选）在评论 – 可选中，输入评论，说明您批准或拒绝工作流运行的原因。
7. 选择提交。

“批准”阶段门 YAML

下面是批准阶段门的 YAML 定义。要了解如何使用此阶段门，请参阅要求批准工作流运行。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The 'Approval' gate definition starts here.
  Approval:
    Identifier: aws/approval@v1
    DependsOn:
      - another-action
    Configuration:
      ApprovalsRequired: number
```

Approval

（必需）

指定要为阶段门指定的名称。在工作流中，所有阶段门名称都必须唯一。阶段门名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。您不能使用引号在阶段门名称中启用特殊字符和空格。

默认值：Approval_nn。

对应的 UI：“配置”选项卡/阶段门名称

Identifier

(*Approval*/Identifier)

（必需）

标识阶段门。批准阶段门支持版本 `1.0.0`。除非您希望缩短版本，否则不要更改此属性。有关更多信息，请参阅 指定要使用的操作版本。

默认值：`aws/approval@v1`。

对应的 UI：工作流图表/Approval_nn/aws/approval@v1 标签

DependsOn

(*Approval*/DependsOn)

（可选）

指定必须成功运行才能使该阶段门运行的操作、操作组或阶段门。默认情况下，当您向工作流添加阶段门时，阶段门的设置取决于工作流中的最后一个操作。如果删除此属性，则阶段门将不依赖于任何设置，并且将首先运行，然后再执行其他操作。

> **Note**
>
> 阶段门必须配置为在操作、操作组或阶段门之前或之后运行。阶段门无法设置为与其他操作、操作组和阶段门并行运行。

有关依赖于功能的更多信息，请参阅 按顺序执行阶段门和操作。

对应的 UI：“输入”选项卡/依赖于

Configuration

(*Approval*/Configuration)

（可选）

可在其中定义阶段门的配置属性的部分。

对应的 UI：配置选项卡

ApprovalsRequired

(*Approval*/Configuration/ApprovalsRequired)

（可选）

指定解锁批准阶段门所需的最小审批数。最小值为 1。最大值为 2。如果省略，则默认值为 1。

> **Note**
>
> 如果要省略 `ApprovalsRequired` 属性，请从工作流定义文件中删除该阶段门的 `Configuration` 部分。

对应的 UI：“配置”选项卡/批准数量

配置运行的排队行为

默认情况下，在 Amazon CodeCatalyst 中，当多个工作流运行同时进行时，CodeCatalyst 会将它们排队，并按照其启动顺序逐一处理。您可以通过指定运行模式来更改此默认行为。运行模式分为几种：

- （默认）排队运行模式 – CodeCatalyst 进程逐个运行
- 取代运行模式 – CodeCatalyst 进程逐个运行，较新的运行将取代较旧的运行
- 并行运行模式 – CodeCatalyst 进程并行运行

有关工作流运行的更多信息，请参阅运行工作流。

主题

关于排队运行模式

在排队运行模式下，运行是串行进行的，处于等待中的运行形成队列。

队列在操作和操作组的入口处形成，因此在同一个工作流中可以有多个队列（请参阅Figure 1）。当排队的运行进入某个操作时，该操作将被锁定，其他运行都无法进入。当运行完成并退出操作时，该操作将解锁并准备好进行下一次运行。

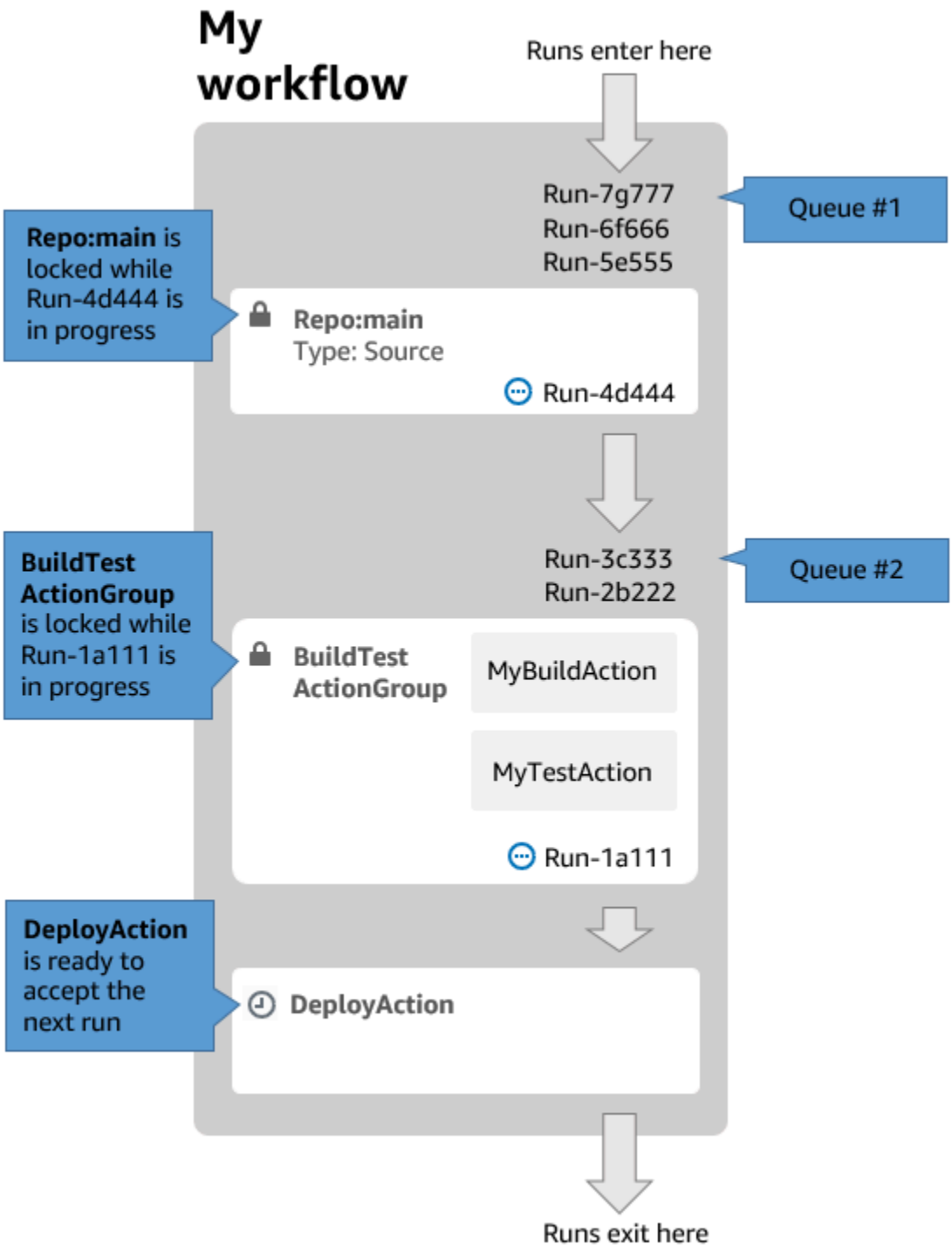
Figure 1 说明了配置为排队运行模式的工作流。它显示了：

- 有七个运行在通过工作流。
- 两个队列：一个在输入源（Repo:main）的入口之外，另一个在 BuildTestActionGroup 操作的入口之外。
- 两个锁定的块：输入源（Repo:main）和 BuildTestActionGroup。

以下是工作流运行的处理完成后会发生的情况：

- 当 Run-4d444 完成源存储库的克隆后，它将退出输入源并加入队列，位于 Run-3c333 之后。然后，Run-5e555 将进入输入源。
- 当 Run-1a111 完成构建和测试后，它将退出 BuildTestActionGroup 操作并进入 DeployAction 操作。然后，Run-2b222 将进入 BuildTestActionGroup 操作。

图 1：配置为“排队运行模式”的工作流



在以下情况下使用排队运行模式：

- 您希望在功能和运行之间保持一对一的关系，在使用取代模式时，这些功能可能会被分组。例如，当您在提交 1 中合并功能 1 时，运行 1 会启动；当您在提交 2 中合并功能 2 时，运行 2 会启动，依此类推。如果您使用取代模式而不是排队模式，则您的功能（和提交）将一起分组到运行中，并会取代其他运行。

- 您需要避免使用并行模式时可能出现的争用情况和意外问题。例如，如果有两位软件开发人员 Wang 和 Saanvi，他们在大致相同的时间启动工作流运行以部署到 Amazon ECS 集群，那么 Wang 的运行可能会在集群上启动集成测试，而 Saanvi 的运行会向集群部署新的应用程序代码，从而导致 Wang 的测试失败或测试了错误的代码。再举一个例子，您的目标可能没有锁定机制，在这种情况下，两次运行可能会以意想不到的方式覆盖彼此的更改。
- 您想限制 CodeCatalyst 用于处理运行的计算资源上的负载。例如，如果工作流中有三个操作，则您最多可能会同时进行三个运行。通过对一次可以进行的运行数施加限制，可以使运行吞吐量更具可预测性。
- 您想限制工作流向第三方服务发出的请求数量。例如，您的工作流可能有一个构建操作，其中包括从 Docker Hub 提取映像的说明。Docker Hub 对每个账户每小时可以发出的拉取请求数量有一定的限制，在超过限制时会被阻止。使用排队运行模式可降低运行吞吐量，这样就可以减少每小时向 Docker Hub 发出的请求数，从而避免可能出现的锁定情况以及导致的构建和运行失败。

最大队列大小：50

有关最大队列大小的注意事项：

- 最大队列大小是指在工作流的所有队列中，允许的运行的最大数量。
- 如果队列中的运行超过 50 个，则 CodeCatalyst 会丢弃第 51 个及随后的运行。

失败行为：

如果某个运行在操作处理时变得没有响应，则其后面的运行将暂停在队列中，直到操作超时。操作在 1 小时后超时。

如果运行在操作内部失败，则允许排队在其后面的第一个运行继续进行。

关于取代运行模式

取代运行模式与排队运行模式大致相同，不同之处在于：

- 如果排队的运行赶上了队列中的另一个运行，则后一个运行将取代（接管）先前的运行，而先前的运行将被取消并标记为“取代”。
- 作为第一个要点中描述的行为的结果，使用取代运行模式时，队列只能包含一个运行。

以Figure 1 中的工作流为指南，对此工作流应用取代运行模式将导致以下结果：

- Run-7g777 将取代队列中的另外两个运行，并且将是 Queue #1 中唯一剩下的运行。Run-6f666 和 Run-5e555 将被取消。
- Run-3c333 将取代 Run-2b222，成为 Queue #2 中唯一剩下的运行。Run-2b222 将被取消。

在有以下需求时，请使用取代运行模式：

- 比排队模式更好的吞吐量
- 相比排队模式向第三方服务发出更少的请求；如果第三方服务有速率限制（例如 Docker Hub），这将非常有利

关于并行运行模式

在并行运行模式中，运行是相互独立的，不会等待其他运行完成后才开始。其中没有队列，运行吞吐量仅受工作流内部的操作完成速度的限制。

在开发环境中使用并行运行模式，每个用户都有自己的功能分支，并部署到不与其他用户共享的目标。

> **Important**
>
> 如果您有多个用户可以部署到的共享目标，例如生产环境中的 Lambda 函数，请不要使用并行模式，因为可能会导致争用情况。当并行工作流运行尝试同时更改共享资源时，就会出现争用情况，导致不可预测的结果。

最大并行运行数量：每个 CodeCatalyst 空间 1000 个

配置运行模式

您可以将运行设置为排队、取代或并行模式。默认使用排队模式。

当您将运行从排队或取代模式更改为并行模式时，CodeCatalyst 会取消排队中的运行，并使当前由操作处理的运行能够完成，而不是取消。

当您将运行从并行模式更改为排队或取代模式时，CodeCatalyst 会让所有当前正在进行的并行运行完成。将运行更改为排队或取代模式后开始的任何运行都使用新模式。

Visual

使用可视化编辑器更改运行模式

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 在右上角，选择工作流属性。
7. 展开高级，在运行模式下，选择以下选项之一：
   a. 排队 – 请参阅关于排队运行模式
   b. 取代 – 请参阅关于取代运行模式
   c. 并行 – 请参阅关于并行运行模式
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器更改运行模式

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 添加 RunMode 属性，如下所示：

```
Name: Workflow_6d39
SchemaVersion: "1.0"
```

```
RunMode: QUEUED|SUPERSEDED|PARALLEL
```

有关更多信息，请参阅工作流 YAML 定义的顶级属性部分中对 RunMode 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

在工作流运行之间缓存文件

启用文件缓存后，生成和测试操作会将磁盘上的文件保存到缓存中，并在后续的工作流运行中从缓存中恢复这些文件。缓存可以减少因构建或下载两次运行之间未更改的依赖项而导致的延迟。CodeCatalyst 还支持后备缓存，可用于还原包含一些所需依赖项的部分缓存。这有助于减少缓存未命中造成的延迟影响。

> **Note**
>
> 文件缓存仅在 Amazon CodeCatalyst 构建和测试操作中可用，并且仅在配置为使用EC2计算类型时才可用。

主题

- 关于文件缓存
- 创建缓存
- 文件缓存限制

关于文件缓存

文件缓存让您可以将数据组织到多个缓存中，每个缓存都在 `FileCaching` 属性下引用。每个缓存都会保存在给定路径指定的目录中。指定的目录将在未来的工作流运行中恢复。以下是使用名为 `cacheKey1` 和 `cacheKey2` 的多个缓存进行缓存的 YAML 代码片段示例。

```
Actions:
  BuildMyNpmApp:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
```

```
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
    Caching:
      FileCaching:
        cacheKey1:
          Path: file1.txt
          RestoreKeys:
             - restoreKey1
        cacheKey2:
          Path: /root/repository
          RestoreKeys:
             - restoreKey2
             - restoreKey3
```

> **Note**
>
> CodeCatalyst 使用多层缓存，由本地缓存和远程缓存组成。当预置实例集或按需计算机在本地缓存中遇到缓存未命中的情况时，将从远程缓存中恢复依赖项。因此，某些操作运行可能会因下载远程缓存而出现延迟。

CodeCatalyst 应用缓存访问限制，以确保一个工作流程中的操作无法修改其他工作流程中的缓存。这样可以防止各个工作流收到其他工作流中可能推送的错误数据，影响其构建或部署。限制是通过缓存范围来强制执行的，该范围将缓存隔离到每个工作流和分支对。例如，分支 `feature-A` 中的 `workflow-A` 与同级分支 `feature-B` 中的 `workflow-A` 具有不同的文件缓存。

当工作流查找指定的文件缓存但无法找到时，就会发生缓存未命中。出现这种情况可能有多种原因，例如创建了新分支或引用了新缓存但新缓存尚未创建时。这种情况也可能发生在缓存过期时，默认情况下，缓存在上次使用的 14 天后过期。为了减少缓存未命中率并提高缓存命中率， CodeCatalyst 支持后备缓存。回退缓存是备用缓存，它提供了恢复部分缓存的机会，部分缓存可能是某个缓存的较旧版本。在恢复缓存时，首先在 `FileCaching` 下搜索属性名称的匹配，如果未找到，则评估 `RestoreKeys`。如果属性名称和所有 `RestoreKeys` 均出现缓存未命中，则工作流将继续运行，因为缓存是尽力提供服务，并不提供保证。

创建缓存

您可以按照以下说明向工作流添加缓存。

Visual

使用可视化编辑器添加缓存

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要添加缓存的操作。
8. 选择配置。
9. 在文件缓存 – 可选下，选择添加缓存，然后在字段中输入信息，如下所示：

   键

   指定主缓存属性的名称。缓存属性名称在您的工作流内必须是唯一的。每个操作在 `FileCaching` 中最多可以有五个条目。

   路径

   为您的缓存指定关联路径。

   恢复键 – 可选

   指定在找不到主缓存属性时用作回退手段的恢复键。恢复键名称在您的工作流内必须是唯一的。每个缓存在 `RestoreKeys` 中最多可以有五个条目。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加缓存

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。

3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流操作中，添加类似于下文的代码：

```
action-name:
  Configuration:
    Steps: ...
  Caching:
    FileCaching:
      key-name:
        Path: file-path
        # # Specify any additional fallback caches
        # RestoreKeys:
        #   - restore-key
```

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

文件缓存限制

以下是属性名称和 RestoreKeys 的限制：

- 名称在工作流中必须唯一。
- 名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。
- 名称最多可以包含 180 个字符。
- 每个操作在 FileCaching 中最多可以有五个缓存。
- 每个缓存在 RestoreKeys 中最多可以有五个条目。

以下是路径的限制：

- 不允许使用星号（*）。
- 路径最多可以包含 255 个字符。

查看工作流运行状态和详细信息

在 Amazon CodeCatalyst 中，您可以查看单个工作流运行的状态和详细信息，也可以同时查看多个运行。

有关可能的运行状态列表，请参阅工作流运行状态。

> **Note**
>
> 您还可以查看工作流状态，该状态不同于工作流运行状态。有关更多信息，请参阅查看工作流的状态。

有关工作流运行的更多信息，请参阅运行工作流。

主题

查看单个运行的状态和详细信息

您可能需要查看单个工作流运行的状态和详细信息，以查看运行是否成功、其完成时间或者查看谁或什么启动了它。

查看单个运行的状态和详细信息

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 在工作流的名称下，选择运行。
6. 在运行历史记录的运行 ID 列中，选择一个运行。例如 Run-95a4d。
7. 在运行的名称下，执行以下操作之一：

- 可视化，可以查看显示工作流运行的操作及其状态的工作流图表（请参阅工作流运行状态）。此视图还显示运行期间使用的源存储库和分支。

  在工作流图表中，选择一个操作以查看该操作在运行期间生成的日志、报告和输出等详细信息。显示的信息取决于所选择的操作类型。有关查看构建或部署日志的详细信息，请参阅查看构建操作的结果或查看部署日志。
- YAML，可查看运行所使用的工作流定义文件。
- 构件，可查看工作流运行生成的构件。有关构件的更多信息，请参阅在操作之间共享构件和文件。
- 报告，可查看工作流运行生成的测试报告和其他类型的报告。有关报告的更多信息，请参阅质量报告类型。
- 变量，可查看工作流运行生成的输出变量。有关变量的更多信息，请参阅在工作流中使用变量。

> **Note**
>
> 如果删除了运行的父工作流，则在运行详细信息页面的顶部会显示一条消息说明这一事实。

查看项目中所有运行的状态和详细信息

您可能需要查看项目中所有工作流运行的状态和详细信息，以了解项目中当前有多少个工作流活动正在进行，并了解工作流的整体运行状况。

查看项目中所有运行的状态和详细信息

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 在工作流下，选择运行。

   此时将显示项目中所有存储库的所有分支中的所有工作流的所有运行。

   页面中包括以下列：

   - ID – 运行的唯一标识符。选择运行 ID 链接以查看有关运行的详细信息。

- 状态 – 工作流运行的处理状态。有关运行状态的更多信息，请参阅工作流运行状态。
- 触发器 – 启动工作流运行的人员、提交、拉取请求（PR，Pull Request）或计划。有关更多信息，请参阅使用触发器自动启动工作流运行。
- 工作流 – 启动运行的工作流的名称，以及工作流定义文件所在的源存储库和分支。您可能需要展开列宽才能看到此信息。

> **Note**
>
> 如果此列设置为不可用，则通常是因为关联的工作流已删除或已移动。

- 开始时间 – 工作流运行的启动时间。
- 持续时间 – 处理工作流运行所花费的时间。持续时间过长或过短都可能表示存在问题。
- 结束时间 – 工作流运行的结束时间。

查看特定工作流所有运行的状态和详细信息

您可能需要查看与特定工作流关联的所有运行的状态和详细信息，以查看是否有任何运行在工作流中造成瓶颈，或者查看哪些运行当前正在进行或已完成。

查看特定工作流的所有运行的状态和详细信息

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 在工作流的名称下，选择运行。

   将显示与所选工作流关联的运行。

   此页面分为两个部分：

   - 活跃运行 – 显示正在进行的运行。这些运行将处于以下状态：进行中。
   - 运行历史记录 – 显示已完成（即不在进行中）的运行。

   有关运行状态的更多信息，请参阅工作流运行状态。

在工作流图表中查看工作流的运行

当多个运行一起在工作流中执行时，您可以查看工作流中所有运行的状态。运行结果显示在工作流图表中（而不是列表视图）。图表直观地显示哪些运行正在由哪些操作处理，哪些运行正在队列中等待。

查看多个运行在工作流中一起执行的状态

> **Note**
>
> 仅当您的工作流使用排队或取代运行模式时，此过程才适用。有关更多信息，请参阅配置运行的排队行为。

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

   > **Note**
   >
   > 确保您查看的是工作流页面，而不是运行页面。

5. 选择左上角的最新状态选项卡。

   此时将出现工作流图表。
6. 查看工作流图表。图表显示工作流中当前正在进行的所有运行，以及已完成的最新运行。更具体地说：

   - 显示在顶部，位于源之前的运行在排队等待启动。
   - 显示在操作之间的运行在排队，等待由下一个操作处理。
   - 在操作中显示的运行的状态是 1. 当前正在由该操作处理，2. 该操作已完成处理，或者 3. 未由该操作处理（通常是因为之前的操作失败）。

配置工作流操作

操作是工作流的主要构建基块，它定义了工作流运行期间要执行的逻辑工作单元（又称任务）。通常，一个工作流包括多个按顺序运行或并行运行的操作，具体取决于您配置这些操作的方式。

主题

操作类型

在 Amazon CodeCatalyst 工作流程中，您可以使用以下类型的操作。

操作类型

CodeCatalyst 行动

CodeCatalyst 动作是由 CodeCatalyst 开发团队创作、维护和全力支持的动作。

有一些用于构建、测试和部署应用程序的 CodeCatalyst 操作，以及用于执行其他任务（例如调用 AWS Lambda 函数）的操作。

以下 CodeCatalyst 操作可用：

- 构建

  此操作会构建您的构件并在 Docker 容器中运行单元测试。有关更多信息，请参阅 添加构建操作。

- 测试

  此操作会针对您的应用程序或构件运行集成和系统测试。有关更多信息，请参阅 添加测试操作。

- Amazon S3 发布

  此操作会将应用程序构件复制到 Amazon S3 存储桶。有关更多信息，请参阅 使用工作流将文件发布到 Amazon S3。

- AWS CDK bootstrap

  此操作会预置部署 CDK 应用程序 AWS CDK 所需的资源。有关更多信息，请参阅 使用工作流程引导 AWS CDK 应用程序。

- AWS CDK 部署

  此操作合成并部署应用程序。 AWS Cloud Development Kit (AWS CDK) 有关更多信息，请参阅 使用工作流程部署 AWS CDK 应用程序。

- AWS Lambda 调用

  此操作调用一个函数。 AWS Lambda 有关更多信息，请参阅 使用工作流调用 Lambda 函数。

- GitHub 操作

  此操作允许您在 CodeCatalyst 工作流程中运行 GitHub 操作。CodeCatalyst有关更多信息，请参阅 使用工作流调用 Lambda 函数。

- 部署 CloudFormation 堆栈

  此操作会部署 CloudFormation 堆栈。有关更多信息，请参阅 部署 CloudFormation 堆栈。

- 部署到 Amazon ECS

  此操作会注册 Amazon ECS 任务定义并将其部署到 Amazon ECS 服务。有关更多信息，请参阅 使用工作流部署到 Amazon ECS。

- 部署到 Kubernetes 集群

  此操作会将应用程序部署到 Kubernetes 集群。有关更多信息，请参阅 使用工作流部署到 Amazon EKS。

- 渲染 Amazon ECS 任务定义

  此操作会将容器映像 URI 插入到 Amazon ECS 任务定义 JSON 文件中，从而创建新的任务定义文件。有关更多信息，请参阅 修改 Amazon ECS 任务定义。

CodeCatalyst 操作文档可在本指南和每个操作的自述文件中找到。

有关可用 CodeCatalyst 操作以及如何向工作流程添加操作的信息，请参阅添加操作到工作流。

CodeCatalyst 实验室行动

CodeCatalyst 实验室操作是 Amazon Labs 的一部分，Amazon CodeCatalyst Labs 是实验性应用程序的试验场。 CodeCatalyst 已经开发了实验室操作来展示与 AWS 服务的集成。

以下 CodeCatalyst 实验室操作可用：

- 部署到 AWS Amplify 主机

  该操作会将应用程序部署到 Amplify Hosting。
- 部署到 AWS App Runner

  此操作会将源映像存储库中的最新映像部署到 App Runner。
- 部署到亚马逊 CloudFront 和亚马逊 S3

  此操作会将应用程序部署到 CloudFront 和 Amazon S3。
- 使用部署 AWS SAM

  此操作使用 AWS Serverless Application Model （AWS SAM）部署无服务器应用程序。
- 使 Amazon CloudFront 缓存失效

  此操作会使给定路径集的 CloudFront 缓存失效。
- 传出 Webhook

  此操作使用户能够使用 HTTPS 请求将工作流中的消息发送到任意 Web 服务器。
- 发布到 AWS CodeArtifact

  此操作将包发布到 CodeArtifact 存储库。
- 发布到 Amazon SNS

此操作使用户能够通过创建主题、发布到主题或订阅主题来与 Amazon SNS 集成。

- 推送到 Amazon ECR

  此操作会构建 Docker 映像并将其发布到 Amazon Elastic Container Registry（Amazon ECR）存储库。

- 使用 Amazon CodeGuru 安全软件进行扫描

  此操作会创建已配置代码路径的 zip 存档，并使用 “ CodeGuru 安全” 来运行代码扫描。

- Terraform Community Edition

  此操作会运行 Terraform Community Edition 的 `plan` 和 `apply` 操作。

CodeCatalyst 实验室操作的文档可在每个操作的自述文件中找到。

有关向工作流程添加 CodeCatalyst 实验室操作和查看其自述文件的信息，请参阅添加操作到工作流。

GitHub 行动

Acti GitHub on 很像一个CodeCatalyst 动作，不同之处在于它是为与 GitHub 工作流程一起使用而开发的。有关 GitHub 操作的详细信息，请参阅GitHub 操作文档。

在 CodeCatalyst 工作流程中，您可以将 GitHub CodeCatalyst 操作与原生操作一起使用。

为方便起见， CodeCatalyst 控制台提供对多个常用 GitHub操作的访问权限。您也可以使用 GitHub Marketplace 中列出的任何 GitHub 操作（但有一些限制）。

操作文档可在每个 GitHub 操作的自述文件中找到。

有关更多信息，请参阅 与 GitHub 操作集成。

第三方操作

第三方操作是由第三方供应商创作并在 CodeCatalyst 控制台中提供的操作。第三方操作的示例包括分别由 Mend 和 Sonar 编写的 “修复 SCA” 和 “SonarCloud 扫描” 操作。

第三方操作的文档可在每个操作的自述文件中找到。第三方供应商也可能提供其它文档。

有关向工作流中添加第三方操作和查看操作自述文件的信息，请参阅添加操作到工作流。

添加操作到工作流

按照以下说明向工作流添加操作，然后进行配置。

添加和配置操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 在左上角，选择 + 操作后将显示操作目录。
7. 在下拉列表中，执行以下操作之一：
   - 选择 Amazon CodeCatalyst 进行查看CodeCatalyst、选择CodeCatalyst 实验室或第三方操作。
     - CodeCatalyst 动作有 b y AWS 标签。
     - CodeCatalyst 实验室操作带有 “按 CodeCatalyst 实验室” 标签。
     - 第三方操作有 b y ***vendor*** 标签，其中*vendor*是第三方供应商的名称。
   - 选择GitHub查看精选的 GitHub 操作列表。
8. 在操作目录中，搜索操作，然后执行以下操作之一：
   - 选择加号（+）可将操作添加到您的工作流中。
   - 选择操作的名称以查看其自述文件。
9. 配置操作。选择可视化以使用可视化编辑器，或者选择 YAML 以使用 YAML 编辑器。有关详细说明，请参阅以下链接。

   有关添加CodeCatalyst操作的说明，请参阅：

   - 添加构建操作
   - 添加测试操作
   - 添加“部署到 Amazon ECS”操作
   - 添加“部署到 Kubernetes 集群”操作
   - 添加 “部署 CloudFormation 堆栈” 操作
   - 添加 “AWS CDK 部署” 操作

- 添加 “AWS CDK 引导” 操作
- 添加“Amazon S3 发布”操作
- 添加 “AWS Lambda 调用” 操作
- 添加“渲染 Amazon ECS 任务定义”操作

有关添加CodeCatalyst 实验室操作的说明，请参阅：

- 操作的自述文件。您可以通过在操作目录中选择操作名称来找到自述文件。

有关添加GitHub 操作的说明，请参阅：

- 与 GitHub 操作集成

有关添加第三方操作的说明，请参阅：

- 操作的自述文件。您可以通过在操作目录中选择操作名称来找到自述文件。

10. （可选）选择验证以确保 YAML 代码有效。
11. 选择提交以提交您的更改。

从工作流中删除操作

按照以下说明从工作流中删除操作。

Visual

使用可视化编辑器删除操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，在要删除的操作中，选择垂直省略号图标，然后选择删除。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器删除操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 找到 YAML 中包含要删除的操作的部分。

   选择该部分并按键盘上的 Delete 键。
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

开发自定义操作

您可以使用动作开发套件 (ADK) 开发自定义 CodeCatalyst 操作以在工作流程中使用。然后，您可以将该操作发布到 CodeCatalyst 操作目录，以便其他 CodeCatalyst 用户可以在其工作流程中查看和使用它。

开发、测试和发布操作（高级任务）

1. 安装开发操作所需的工具和程序包。
2. 创建 CodeCatalyst 存储库来存储您的操作代码。
3. 初始化操作。这会列出操作所需的源文件，包括您可以用自己的代码更新的操作定义文件（`action.yml`）。
4. 引导操作代码以获取构建、测试和发布操作项目所需的工具和库。
5. 在本地计算机上生成操作，然后将更改推送到存储 CodeCatalyst库。

6. 在本地使用单元测试测试操作，然后在中运行 ADK 生成的工作流程。 CodeCatalyst
7. 在 CodeCatalyst 控制台中选择“发布”按钮，将 CodeCatalyst 操作发布到操作目录。

有关详细步骤，请参阅 Amazon Acti CodeCatalyst on 开发套件开发者指南。

将操作分组为操作组

一个操作组包含一个或多个操作。将操作分组为操作组有助于将工作流保持得井井有条，还可以配置不同组之间的依赖关系。

> **Note**
>
> 您不能将操作组嵌套到其他操作组或操作中。

主题

- 定义操作组
- 示例：定义两个操作组

定义操作组

按照以下说明定义 CodeCatalyst 操作组。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

定义组

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。

6. 选择 YAML。
7. 在 Actions 中，添加类似于以下代码的代码：

```
Actions:
  action-group-name:
    Actions:
      action-1:
        Identifier: aws/build@v1
        Configuration:
          ...
      action-2:
        Identifier: aws/build@v1
        Configuration:
          ...
```

有关另一个示例，请参阅示例：定义两个操作组。有关更多信息，请参阅工作流 YAML 定义的操作中对 action-group-name 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

示例：定义两个操作组

以下示例说明如何定义两个 Amazon CodeCatalyst 操作组：BuildAndTest和Deploy。BuildAndTest 组包括两个操作（Build 和 Test），Deploy 组同样包括两个操作（DeployCloudFormationStack 和 DeployToECS）。

```
Actions:
  BuildAndTest: # Action group 1
    Actions:
      Build:
        Identifier: aws/build@v1
        Configuration:
          ...
      Test:
        Identifier: aws/managed-test@v1
        Configuration:
  Deploy: #Action group 2
    Actions:
      DeployCloudFormationStack:
        Identifier: aws/cfn-deploy@v1
```

```
    Configuration:
      ...
  DeployToECS:
    Identifier: aws/ecs-deploy@v1
    Configuration:
      ...
```

顺序操作

默认情况下，当您将操作添加到工作流时，它们将并排添加到可视化编辑器中。这意味着，当您启动工作流运行时，这些操作将并行运行。如果要让操作按顺序运行（并垂直显示在可视化编辑器中），您必须设置它们之间的依赖关系。例如，您可以将 Test 操作设置为依赖于 Build 操作，这样就让测试操作在构建操作之后运行。

您可以在操作和操作组之间设置依赖关系。您还可以配置 one-to-many依赖关系，以便一个操作依赖于其他几个操作才能启动。请查阅设置操作之间的依赖关系中的准则，确保您的依赖项设置符合工作流的 YAML 语法。

主题

- 如何在操作之间配置依赖关系的示例
- 设置操作之间的依赖关系

如何在操作之间配置依赖关系的示例

以下示例说明如何在工作流定义文件中，配置操作和组之间的依赖关系。

主题

- 示例：配置简单依赖关系
- 示例：将操作组配置为依赖于操作
- 示例：将一个操作组配置为依赖于另一个操作组
- 示例：将操作组配置为依赖于多个操作

示例：配置简单依赖关系

以下示例说明如何使用 DependsOn 属性将 Test 操作配置为依赖于 Build 操作。

```
Actions:
```

```
  Build:
    Identifier: aws/build@v1
    Configuration:
      ...
  Test:
    DependsOn:
      - Build
    Identifier: aws/managed-test@v1
     Configuration:
       ...
```

示例：将操作组配置为依赖于操作

以下示例说明如何将 DeployGroup 操作组配置为依赖于 FirstAction 操作。请注意，操作和操作组处于同一级别。

```
Actions:
  FirstAction: #An action outside an action group
    Identifier: aws/github-actions-runner@v1
    Configuration:
      ...
  DeployGroup: #An action group containing two actions
    DependsOn:
      - FirstAction
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

示例：将一个操作组配置为依赖于另一个操作组

以下示例说明如何将 DeployGroup 操作组配置为依赖于 BuildAndTestGroup 操作组。请注意，这些操作组处于同一级别。

```
Actions:
  BuildAndTestGroup: # Action group 1
    Actions:
      BuildAction:
      ...
      TestAction:
      ...
```

```
  DeployGroup: #Action group 2
    DependsOn:
      - BuildAndTestGroup
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

示例：将操作组配置为依赖于多个操作

以下示例说明如何将 DeployGroup 操作组配置为依赖于 FirstAction 操作、SecondAction 操作和 BuildAndTestGroup 操作组。请注意，DeployGroup 与 FirstAction、SecondAction 和 BuildAndTestGroup 处于同一级别。

```
Actions:
  FirstAction: #An action outside an action group
    ...
  SecondAction: #Another action
    ...
  BuildAndTestGroup: #Action group 1
    Actions:
      Build:
      ...
      Test:
      ...
  DeployGroup: #Action group 2
    DependsOn:
      - FirstAction
      - SecondAction
      - BuildAndTestGroup
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

设置操作之间的依赖关系

按照以下说明设置工作流中操作之间的依赖关系。

配置依赖关系时，请按照以下准则操作：

- 如果某个操作位于组中，则该操作只能依赖于同一组中的其他操作。
- 操作和操作组可以依赖于 YAML 层次结构中同一级别的其他操作和操作组，但不能依赖于其他级别中的操作和操作组。

Visual

使用可视化编辑器设置依赖关系

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择将依赖于其他操作的操作。
8. 选择输入选项卡。
9. 在依赖于 – 可选中，执行以下操作：

   指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

   有关“依赖于”功能的更多信息，请参阅顺序操作。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器设置依赖关系

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

5. 选择编辑。
6. 选择 YAML。
7. 在依赖于另一个操作的操作中，添加类似于下文的代码：

```
action-name:
  DependsOn:
    - action-1
```

有关更多示例，请参阅如何在操作之间配置依赖关系的示例。有关一般准则，请参阅设置操作之间的依赖关系。有关更多信息，请参阅工作流 YAML 定义 中相应操作 DependsOn 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

在操作之间共享构件和文件

构件是工作流操作的输出，通常由文件夹或文件存档组成。构件之所以重要，是因为它们让您可以在操作之间共享文件和信息。

例如，您可能有一个构建操作生成了 sam-template.yml 文件，但您希望部署操作使用该文件。在这种情况下，您将使用构件来允许构建操作与部署操作共享 sam-template.yml 文件。代码类似于如下所示：

```
Actions:
  BuildAction:
    Identifier: aws/build@v1
    Steps:
      - Run: sam package --output-template-file sam-template.yml
    Outputs:
      Artifacts:
        - Name: MYARTIFACT
          Files:
            - sam-template.yml
  DeployAction:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Artifacts:
        - MYARTIFACT
    Configuration:
```

```
      template: sam-template.yml
```

在前面的代码中，构建操作（BuildAction）生成一个 sam-template.yml 文件，然后将其添加到名为 MYARTIFACT 的输出构件中。随后的部署操作（DeployAction）指定 MYARTIFACT 作为输入，向其提供了对 sam-template.yml 文件的访问权限。

主题

我能否在不将构件指定为输出和输入的情况下共享它们？

可以，您可以在操作之间共享构件，而无需在操作的 YAML 代码的 Outputs 和 Inputs 部分中指定构件。为此，您必须启用计算共享。有关计算共享以及如何在启用计算共享时指定构件的更多信息，请参阅跨操作共享计算。

> **Note**
>
> 尽管计算共享功能通过消除对 Outputs 和 Inputs 部分的需求来简化工作流的 YAML 代码，但该功能存在一些局限性，您应在启用该功能之前了解这些局限性。有关这些限制的信息，请参阅计算共享注意事项。

我能否在工作流之间共享构件？

不可以，您不能在不同的工作流之间共享构件；但是，您可以在同一工作流中的操作之间共享构件。

构件示例

以下示例展示了如何在 Amazon CodeCatalyst 工作流定义文件中输出、输入和引用构件。

主题

示例：输出构件

以下示例演示如何输出包含两个 .jar 文件的构件。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT1
          Files:
            - build-output/file1.jar
            - build-output/file2.jar
```

示例：输入由其他操作生成的构件

以下示例向您展示了如何在 BuildActionA 中输出名为 ARTIFACT4 的构件并将其输入到 BuildActionB。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT4
          Files:
            - build-output/file1.jar
            - build-output/file2.jar
  BuildActionB:
    Identifier: aws/build@v1
    Inputs:
```

```
      Artifacts:
        - ARTIFACT4
    Configuration:
```

示例：引用多个构件中的文件

以下示例向您展示了如何在 BuildActionC 中输出名为 ART5 和 ART6 的两个构件，然后在 BuildActionD 中（位于 Steps 下）引用名为 file5.txt（在构件 ART5 中）和 file6.txt（在构件 ART6 中）的两个文件。

> **Note**
>
> 有关引用文件的更多信息，请参阅在构件中引用文件。

> **Note**
>
> 尽管该示例显示使用了 $CATALYST_SOURCE_DIR_ART5 前缀，但您可以省略它。这是因为 ART5 是主输入。要了解有关主输入的更多信息，请参阅在构件中引用文件。

```
Actions:
  BuildActionC:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART5
          Files:
            - build-output/file5.txt
        - Name: ART6
          Files:
            - build-output/file6.txt
  BuildActionD:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART5
        - ART6
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART5/build-output && cat file5.txt
```

```
      - run: cd $CATALYST_SOURCE_DIR_ART6/build-output && cat file6.txt
```

示例：在单个构件中引用文件

以下示例向您展示了如何在 BuildActionE 中输出名为 ART7 的构件，然后在 BuildActionF 中（位于 Steps 下）引用 file7.txt（在构件 ART7 中）文件。

请注意，该引用并不需要像在 示例：引用多个构件中的文件 中那样，在 build-output 目录前面添加 $CATALYST_SOURCE_DIR_*artifact-name* 前缀。这是因为 Inputs 下仅指定了一项。

> **Note**
>
> 有关引用文件的更多信息，请参阅在构件中引用文件。

```
Actions:
  BuildActionE:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART7
          Files:
            - build-output/file7.txt
  BuildActionF:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART7
    Configuration:
      Steps:
        - run: cd build-output && cat file7.txt
```

示例：存在 WorkflowSource 时引用构件中的文件

以下示例向您展示了如何在 BuildActionG 中输出名为 ART8 的构件，然后在 BuildActionH 中（位于 Steps 下）引用 file8.txt（在构件 ART8 中）文件。

请注意，引用需要 $CATALYST_SOURCE_DIR_*artifact-name* 前缀，就像在 示例：引用多个构件中的文件 中一样。这是因为 Inputs 下指定了多项（一个源和一个构件），所以您需要使用前缀来指示在哪里查找文件。

> **Note**
>
> 有关引用文件的更多信息，请参阅在构件中引用文件。

```
Actions:
  BuildActionG:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART8
          Files:
            - build-output/file8.txt
  BuildActionH:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - ART8
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART8/build-output && cat file8.txt
```

示例：存在操作组时引用构件中的文件

以下示例向您展示了如何在 ActionGroup1、ActionI 中输出名为 ART9 的构件，然后在 ActionJ 中引用 file9.txt（在构件 ART9 中）。

有关引用文件的更多信息，请参阅在构件中引用文件。

```
Actions:
  ActionGroup1:
    Actions:
      ActionI:
        Identifier: aws/build@v1
        Outputs:
          Artifacts:
            - Name: ART9
              Files:
                - build-output/file9.yml
      ActionJ:
```

```
      Identifier: aws/cfn-deploy@v1
      Inputs:
        Sources:
          - WorkflowSource
        Artifacts:
          - ART9
      Configuration:
        template: /artifacts/ActionGroup1@ActionJ/ART9/build-output/file9.yml
```

定义输出构件

按照以下说明定义您希望 Amazon CodeCatalyst 操作输出的对象。该构件随后可供其他操作使用。

> **Note**
>
> 并非所有操作都支持输出构件。要确定您的操作是否支持输出构件，请仔细阅读随后的可视化编辑器说明，并查看操作的输出选项卡上是否包含输出构件按钮。如果是，则支持输出构件。

Visual

使用可视化编辑器定义输出构件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择将生成构件的操作。
8. 选择输出选项卡。
9. 在构件下，选择添加构件。
10. 选择添加构件，然后在字段中输入信息，如下所示。

    生成构件名称

指定操作生成的构件的名称。构件名称在工作流内必须是唯一的，并且仅限于字母数字字符（a-z、A-Z、0-9）和下划线（_）。不允许使用空格、连字符（-）和特殊字符。不能使用引号以使输出构件名称包含空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

通过构建生成的文件

指定操作输出的对象中 CodeCatalyst 包含的文件。这些文件由工作流操作在运行时生成，也可在您的源存储库中找到。文件路径可以位于源存储库或先前操作的构件中，并且相对于源存储库或构件根目录。您可以使用 glob 模式来指定路径。示例：

- 要指定位于构建位置或源存储库位置根目录中的单个文件，请使用 `my-file.jar`。
- 要在子目录中指定单个文件，请使用 `directory/my-file.jar` 或 `directory/subdirectory/my-file.jar`。
- 要指定所有文件，请使用 `"**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件和目录，请使用 `"directory/**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件，而非其任意子目录，请使用 `"directory/*"`。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（*）或其他特殊字符，请用双引号（""）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个构件或源中查找它。有关更多信息，请参阅引用源存储库文件和在构件中引用文件。

11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器定义输出构件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流操作中，添加类似于下文的代码：

```
action-name:
  Outputs:
    Artifacts:
      - Name: artifact-name
        Files:
          - file-path-1
          - file-path-2
```

有关更多示例，请参阅构件示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

定义输入构件

如果您想使用由其他 Amazon CodeCatalyst 操作生成的项目，则必须将其指定为当前操作的输入。您可以指定多个构件作为输入，具体取决于相应操作。有关更多信息，请参阅相应操作的工作流 YAML 定义。

> **Note**
>
> 您不能引用其他工作流中的构件。

按照以下说明，将来自另一个操作的构件指定为当前操作的输入。

先决条件

在开始之前，请确保您已从其他操作输出了构件。有关更多信息，请参阅定义输出构件。输出构件，以使其可供其他操作使用。

Visual

将构件指定为操作的输入（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要将构件指定为输入的操作。
8. 选择输入。
9. 在构件 – 可选中，执行以下操作：

   指定以前操作中的一些构件，您希望将这些构件用作此操作的输入。这些构件必须已在以前的操作中定义为输出构件。

   如果未指定任何输入构件，则必须在 *action-name*/Inputs/Sources 下指定至少一个源存储库。

   有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

   > **Note**
   >
   > 如果构件 – 可选下拉列表不可用（可视化编辑器），或者在验证 YAML 时出现错误（YAML 编辑器），这可能是因该操作仅支持一个输入导致的。在这种情况下，请尝试移除源输入。

10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

将构件指定为操作的输入（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要将构件指定作为输入的操作中，添加类似于以下内容的代码：

```
action-name:
  Inputs:
    Artifacts:
      - artifact-name
```

有关更多示例，请参阅构件示例。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

在构件中引用文件

如果您的文件位于构件中，并且有一个 Amazon CodeCatalyst 工作流操作需要引用此文件，请完成以下过程。

> **Note**
>
> 另请参阅引用源存储库文件。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

引用构件中的文件（YAML 编辑器）

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要引用文件的操作中，添加类似于以下内容的代码：

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - artifact-name
    Configuration:
      template: artifact-path/path/to/file.yml
```

在前面的代码中，将：

- *artifact-name* 替换为构件的名称。
- *artifact-path* 替换为下表中的值。

如果您添加以下引用...	将 ***artifact-path*** 替换为...
构件操作或测试操作	$CATALYST_SOURCE_DIR_ *artifact-name* /
所有其他操作	$CATALYST_SOURCE_DIR_ *artifact-name* / 或

如果您添加以下引用...	将 ***artifact-path*** 替换为...
	/artifacts/ *current-action-name* /*artifact-name* / 或 如果当前操作在操作组中： /artifacts/ *current-action-group@current-action-name* /*artifact-name* /

有关示例，请参阅 构件示例。

> **Note**
>
> 在以下情况下，您可以省略 *artifact-path*，只需指定相对于构件根目录的文件路径：
>
> - 包含引用的操作仅包含 Inputs 下的一项（例如，它包含一个输入构件而不包含任何源）。
> - 您要引用的文件位于主输入中。主输入要么是 WorkflowSource，要么是列出的第一个输入构件（如果没有 WorkflowSource）。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

下载构件

您可以下载由您的 Amazon CodeCatalyst 工作流操作生成的构件，并进行检查以排除故障。您可以下载两种类型的构件：

- 源构件 – 包含运行开始时存在的源存储库内容的快照的构件。
- 工作流构件 – 在工作流配置文件的 Outputs 属性中定义的构件。

下载工作流输出的构件

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 在工作流的名称下，选择运行。
6. 在运行历史记录的运行 ID 列中，选择一个运行。例如 Run-95a4d。
7. 在运行的名称下，选择构件。
8. 在构件旁边，选择下载。此时会下载存档文件。其文件名由七个随机字符组成。
9. 使用您选择的档案提取实用程序来提取存档。

指定要使用的操作版本

默认情况下，当您向工作流程添加操作时，Amazon CodeCatalyst 会使用以下格式将完整版本添加到工作流程定义文件中：

v*major*.*minor*.*patch*

例如：

```
My-Build-Action:
  Identifier: aws/build@v1.0.0
```

可以在 Identifier 属性中缩短完整版本，以便工作流始终使用操作的最新次要版本或修补版本。

例如，如果您指定：

```
My-CloudFormation-Action:
  Identifier: aws/cfn-deploy@v1.0
```

...并且最新修补版本是 1.0.4，那么操作将使用 1.0.4。如果发布了更高版本，如 1.0.5，那么操作将使用 1.0.5。如果发布了次要版本，如 1.1.0，那么操作将继续使用 1.0.5。

有关指定版本的详细说明，请参阅以下主题之一。

按照以下说明来指明您希望工作流使用哪个版本的操作。您可以指定最新的主要版本或次要版本，也可以指定特定的修补版本。

建议使用操作的最新次要版本或修补版本。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

将工作流配置为使用操作的最新版本或特定的修补版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 找到要编辑其版本的操作。
8. 找到操作的 `Identifier` 属性，然后将版本设置为下列项之一：

   - actionidentifier @v *major* — 使用此语法让工作流程使用特定的主要版本，并允许自动选择最新的次要版本和补丁版本。
   - 操作标识符 @v *major*。 *minor*— 使用此语法让工作流程使用特定的次要版本，并允许自动选择最新的补丁版本。
   - 操作标识符 @v *major*。 *minor*。 *patch*— 使用此语法让工作流程使用特定的补丁版本。

   > **Note**
   >
   > 如果您不确定可用的版本，请参阅列出可用的操作版本。

   > **Note**
   >
   > 不能省略主要版本。

9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

列出可用的操作版本

按照以下说明来确定哪些版本的操作可供您在工作流中使用。

Visual

确定哪些操作版本可用

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 找到要查看其版本的操作：
   a. 在导航窗格中，选择 CI/CD，然后选择工作流。
   b. 选择任意工作流的名称，或创建一个工作流。有关创建工作流的信息，请参阅创建工作流。
   c. 选择编辑。
   d. 在左上角，选择 + 操作打开操作目录。
   e. 在下拉列表中，选择 CodeCatalyst要查看的 Amazon CodeCatalyst、 CodeCatalyst Labs 和第三方操作，或者选择GitHub查看精选 GitHub操作。
   f. 搜索操作，然后选择其名称。请不要选择加号（+）。

   有关该操作的详细信息将显示。
4. 在右上角附近的操作详细信息对话框中，选择版本下拉列表以查看该操作的可用版本列表。

YAML

不可用。选择“可视化”以查看可视化编辑器说明。

查看操作的源代码

您可以查看操作的源代码，以确保其中不包含危险代码、安全漏洞或其他缺陷。

按照以下说明查看CodeCatalyst、CodeCatalyst Labs 或第三方操作的源代码。

> **Note**
>
> 要查看GitHub操作的源代码，请前往 GitHub Marketplace 中该操作的页面。该页面包含操作存储库的链接，您可以在其中找到操作的源代码。

> **Note**
>
> 您无法查看以下 CodeCatalyst 操作的源代码：构建、测试、GitHub 操作。

> **Note**
>
> AWS 不支持或保证操作代码或第三方操作。 GitHub

查看操作的源代码

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 找到要查看其代码的操作：
   a. 在导航窗格中，选择 CI/CD，然后选择工作流。
   b. 选择任意工作流的名称，或创建一个工作流。有关创建工作流的信息，请参阅创建工作流。
   c. 选择编辑。
   d. 在左上角，选择 + 操作打开操作目录。
   e. 在下拉列表中，选择 CodeCatalyst要查看的 Amazon CodeCatalyst、 CodeCatalyst Labs 和第三方操作。
   f. 搜索操作，然后选择其名称。请不要选择加号（+）。

      有关该操作的详细信息将显示。
4. 在靠近底部的操作详细信息对话框中，选择下载。

   出现的页面中显示操作源代码所在的 Amazon S3 存储桶。有关 Amazon S3 的信息，请参阅《Amazon Simple Storage Service 用户指南》中的什么是 Amazon S3？
5. 检查代码，确保代码符合您对质量和安全性的要求。

与 GitHub 操作集成

Acti GitHub on 很像一个CodeCatalyst 动作，不同之处在于它是为与 GitHub 工作流程一起使用而开发的。有关 GitHub 操作的详细信息，请参阅GitHub 操作文档。

在 CodeCatalyst 工作流程中，您可以将 GitHub CodeCatalyst 操作与原生操作一起使用。

有两种方法可以向 CodeCatalyst 工作流程中添加 GitHub 操作：

- 您可以从 CodeCatalyst 控制台的精选列表中选择 “ GitHub 操作”。有几种流行的 GitHub 操作可供选择。有关更多信息，请参阅 添加精心策划 GitHub 的动作。
- 如果您要使用的 GitHub 操作在 CodeCatalyst 控制台中不可用，则可以使用 “GitHub 操作” 操作将其添加。

  GitHub 操作操作是一种封装CodeCatalyst 动作并使其与 CodeCatalyst 工作流程兼容的 GitHub 动作。

  以下是封装 Su per-Linter GitHub 动作的GitHub 操作操作示例：

  ```
  Actions:
    GitHubAction:
      Identifier: aws/github-actions-runner@v1
      Configuration:
        Steps:
          - name: Lint Code Base
            uses: github/super-linter@v4
            env:
              VALIDATE_ALL_CODEBASE: "true"
              DEFAULT_BRANCH: main
  ```

  在前面的代码中， CodeCatalyst GitHub 操作操作（由标识`aws/github-actions-runner@v1`）封装了 Super-Linter 操作（由标识`github/super-linter@v4`），使其在工作流程中起作用。CodeCatalyst

  有关更多信息，请参阅 添加 “GitHub 操作” 操作。

所有 GitHub 动作（无论是否精选）都必须封装在 Actions 动GitHub 作 (`aws/github-actions-runner@v1`) 中，如前面的示例所示。此操作需要包装程序才能正常运行。

主题

GitHub 动作与动作有何不同 CodeCatalyst ？

GitHub 在 CodeCatalyst 工作流程中使用的操作与 CodeCatalyst 操作的访问权限和集成级别 AWS 以及 CodeCatalyst 功能（例如环境和问题）不同。

GitHub 操作能否与工作流程中的其他 CodeCatalyst 操作进行交互？

可以。例如，GitHub Actions 可以使用其他 CodeCatalyst 操作生成的变量作为输入，也可以将输出参数和构件与 CodeCatalyst 操作共享。有关更多信息，请参阅导出 GitHub 输出参数和引用 GitHub 输出参数。

我可以使用哪些 GitHub 操作？

您可以使用 CodeCatalyst 控制台提供的任何 GitHub 操作以及 GitHubMarketplace 中可用的任何 GitHub 操作。如果您决定使用 Marketplace 中的 GitHub 操作，请记住以下限制。

GitHub 操作的局限性 CodeCatalyst

- GitHub 操作不能与 CodeCatalyst Lambda 计算类型一起使用。

- GitHub 操作在 2022 年 11 月的运行时环境 Docker 镜像上运行，其中包括较旧的工具。有关映像和工具的更多信息，请参阅指定运行时环境映像。

- GitHub 内部依赖于github上下文或引用 GitHub特定资源的操作在中不起作用。 CodeCatalyst例如，以下操作在以下情况下不起作用 CodeCatalyst：
  - 尝试添加、更改或更新 GitHub 资源的操作。示例包括更新拉取请求或在中创建问题的操作 GitHub。
  - 几乎所有操作都列在 https://github.com/actions 中。
- GitHub 作为 Docker 容器操作的操作可以运行，但必须由默认 Docker 用户（root）运行。请不要以用户 1001 的身份运行此操作。（在撰写本文时，用户 1001 在中工作 GitHub，但不在中 CodeCatalyst。）有关更多信息，请参阅 Dockerfile 操作支持 GitHub 中的用户主题。

有关可通过 CodeCatalyst 控制台 GitHub 执行的操作的列表，请参阅添加精心策划 GitHub 的动作。

如何添加 GitHub 操作（高级步骤）？

向 CodeCatalyst 工作流程添加 GitHub 操作的高级步骤如下：

1. 在您的 CodeCatalyst 项目中，您可以创建工作流程。在工作流中，您可以定义如何构建、测试和部署应用程序。有关更多信息，请参阅 入门工作流。
2. 在工作流程中，您可以添加精心策划的 GitHub 操作或添加 GitHub 操作操作。
3. 您可以执行下列操作之一：
   - 如果您选择添加一个经策管操作，请配置该操作。有关更多信息，请参阅 添加精心策划 GitHub 的动作。
   - 如果您选择添加非精选动作，则在GitHub操作操作中粘贴该 GitHub 操作的 YAM L 代码。您可以在 M GitHubarketplac e 中选择的 GitHub操作的详情页面上找到此代码。你可能需要稍微修改一下代码才能让它发挥作用 CodeCatalyst。有关更多信息，请参阅 添加 “GitHub 操作” 操作。
4. （可选）在工作流中，您可以添加其他操作，例如构建和测试操作。有关更多信息，请参阅 使用工作流进行构建、测试和部署。
5. 您可以手动启动工作流，也可以通过触发器自动启动工作流。工作流程运行 GitHub 操作和工作流程中的任何其他操作。有关更多信息，请参阅 手动启动工作流运行。

有关详细步骤，请参阅：

- 添加精心策划 GitHub 的动作.

- 添加 “GitHub 操作” 操作.

GitHub 动作会运行 GitHub吗？

不是。 Acti GitHub on 在中运行 CodeCatalyst，使用 CodeCatalyst运行时环境镜像。

我也可以使用 GitHub 工作流程吗？

否。

“GitHub 动作” 操作使用的运行时镜像

CodeCatalyst GitHub 操作操作在 2022 年 11 月的图像上运行。有关更多信息，请参阅 活动映像。

教程：使用 GitHub Action 对代码进行 lint 检查

在本教程中，您将 Super-Linter GitHub Action 添加到 Amazon CodeCatalyst 工作流。Super-Linter 操作会检查代码，找到代码存在错误、格式问题和可疑构造的区域，然后将结果输出到 CodeCatalyst 控制台。将 linter 添加到工作流后，您可以运行工作流来对示例 Node.js 应用程序（`app.js`）进行 lint 检查。之后，您可以修复报告的问题，并再次运行工作流以查看是否已成功修复问题。

> **Tip**
>
> 考虑使用 Super-Linter 来对 YAML 文件（例如 CloudFormation 模板）进行 lint 检查。

主题

先决条件

在开始之前，您需要：

- 一个带已连接的 AWS 账户的 CodeCatalyst 空间。有关更多信息，请参阅 创建空间。

- 您的 CodeCatalyst 空间中的一个名为 codecatalyst-linter-project 的空项目。选择从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。

步骤 1：创建源存储库

在此步骤中，您将在 CodeCatalyst 中创建源存储库。您将使用此存储库来存储示例应用程序源文件（本教程中为 app.js）。

有关源存储库的更多信息，请参阅创建源存储库。

创建源存储库

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的项目 codecatalyst-linter-project。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。
5. 在存储库名称中，输入：

   ```
   codecatalyst-linter-source-repository
   ```

6. 选择创建。

步骤 2：添加 app.js 文件

在此步骤中，您将 app.js 文件添加到源存储库。app.js 包含的函数代码存在几处错误，linter 将发现这些错误。

添加 app.js 文件

1. 在 CodeCatalyst 控制台中，选择您的项目 codecatalyst-linter-project。
2. 在导航窗格中，选择代码，然后选择源存储库。
3. 从源存储库列表中，选择您的存储库 codecatalyst-linter-source-repository。
4. 在文件中，选择创建文件。
5. 在文本框中，输入以下代码：

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;
/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
  try {
    // const ret = await axios(url);
    response = {
      statusCode: 200,
      'body': JSON.stringify({
        message: 'hello world'
        // location: ret.data.trim()
      })
    }
  } catch (err) {
    console.log(err)
    return err
  }

    return response
}
```

6. 对于文件名，输入 app.js。保留其他默认选项。
7. 选择提交。

现在，您已创建一个名为 app.js 的文件。

步骤 3：创建运行 Super-Linter 操作的工作流

在此步骤中，您将创建一个工作流，当您将代码推送到源存储库时，该工作流将运行 Super-Linter 操作。该工作流包含您在 YAML 文件中定义的以下构建基块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- “GitHub Actions”操作 – 触发后，GitHub Actions 操作将运行 Super-Linter 操作，后者反过来会检查源存储库中的所有文件。如果 linter 发现问题，则工作流操作将失败。

创建运行 Super-Linter 操作的工作流

1. 在 CodeCatalyst 控制台中，选择您的项目 `codecatalyst-linter-project`。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择创建工作流。
4. 对于源存储库，选择 `codecatalyst-linter-source-repository`。
5. 对于分支，选择 `main`。
6. 选择创建。
7. 删除 YAML 示例代码。
8. 添加以下 YAML：

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        github-action-code
```

在上述代码中，将 *`github-action-code`* 替换为 Super-Linter 操作代码，如本过程的以下步骤中所示。

9. 转到 GitHub Marketplace 中的 Super-Linter 页面。

10. 在 `steps:`（小写形式）下，找到代码并将其粘贴到 CodeCatalyst 工作流中的 `Steps:`（大写形式）下。

    调整 GitHub Action 代码以符合 CodeCatalyst 标准，如以下代码所示。

    现在，您的 CodeCatalyst 工作流如下所示：

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Lint Code Base
          uses: github/super-linter@v4
          env:
            VALIDATE_ALL_CODEBASE: "true"
            DEFAULT_BRANCH: main
```

11. （可选）选择验证，确保 YAML 代码在提交之前有效。
12. 选择提交，输入提交消息，选择您的 `codecatalyst-linter-source-repository` 存储库，然后再次选择提交。

    现在，您已创建工作流。由于在工作流顶部定义了触发器，因此工作流运行会自动启动。

查看正在运行的工作流

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择您刚刚创建的工作流：`codecatalyst-linter-workflow`。
3. 在工作流图中，选择 SuperLinterAction。
4. 等待操作失败。预计会失败，因为 linter 在代码中发现了问题。
5. 将 CodeCatalyst 控制台保持打开状态并转至步骤 4：修复 Super-Linter 发现的问题。

步骤 4：修复 Super-Linter 发现的问题

Super-Linter 应已在 `app.js` 代码以及源代码库中包含的 `README.md` 文件中发现了问题。

解决 linter 发现的问题

1. 在 CodeCatalyst 控制台中，选择日志选项卡，然后选择 Lint 代码库。

   这将显示 Super-Linter 操作生成的日志。
2. 在 Super-Linter 日志中，向下滚动到第 90 行左右，您可以从该行开始查看问题。其内容与以下内容类似：

   ```
   /github/workspace/hello-world/app.js:3:13: Extra semicolon.
   /github/workspace/hello-world/app.js:9:92: Trailing spaces not allowed.
   /github/workspace/hello-world/app.js:21:7: Unnecessarily quoted property 'body'
    found.
   /github/workspace/hello-world/app.js:31:1: Expected indentation of 2 spaces but
    found 4.
   /github/workspace/hello-world/app.js:32:2: Newline required at end of file but not
    found.
   ```
3. 修复源存储库中的 `app.js` 和 `README.md`，然后提交您的更改。

   > **Tip**
   >
   > 要修复 `README.md`，请在代码块中添加 `markdown`，如下所示：
   >
   > ````
   > ```markdown
   > Setup examples:
   > ...
   > ```
   > ````

   您的更改将自动启动另一个工作流。等待该工作流完成。如果您已修复所有问题，则该工作流将成功。

清理

在 CodeCatalyst 中进行清理以从环境中移除本教程的跟踪。

在 CodeCatalyst 中进行清理

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 删除 `codecatalyst-linter-source-repository`。
3. 删除 `codecatalyst-linter-workflow`。

在本教程中，您已了解如何将 Super-Linter GitHub Action 添加到 CodeCatalyst 工作流中以便对一些代码进行 lint 检查。

添加 “GitHub 操作” 操作

GitHub 操作操作是一种封装CodeCatalyst 动作并使其与 CodeCatalyst 工作流程兼容的 GitHub 动作。

有关更多信息，请参阅 与 GitHub 操作集成。

要将 “GitHub 操作” 操作添加到工作流程，请执行以下步骤。

> **Tip**
>
> 有关向您展示如何使用 “GitHub 操作” 操作的教程，请参阅教程：使用 GitHub Action 对代码进行 lint 检查。

Visual

使用可视化编辑器添加 “GitHub 操作” 动作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择GitHub。

9. 搜索 “GitHub 操作” 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择GitHub 操作。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅'GitHub Actions'动作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加 “GitHub 操作” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择GitHub。
9. 搜索 “GitHub 操作” 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择GitHub 操作。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 根据需求修改 YAML 代码中的属性。'GitHub Actions'动作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

'GitHub 操作'动作定义

GitHub 操作操作定义为工作流程定义文件中的一组 YAML 属性。有关这些属性的信息，请参阅工作流 YAML 定义中的'GitHub Actions'动作 YAML。

添加精心策划 GitHub 的动作

策划的 GitHub 操作是在 CodeCatalyst 控制台中可用的 GitHub 操作，可作为如何在 CodeCatalyst工作流程中使用 GitHub 操作的示例。

策划的 GitHub 操作被封装在由标识符标识的 CodeCatalyst-authored Actions GitHub 操作中。`aws/github-actions-runner@v1`例如，以下是精心策划的 Acti GitHub on TruffleHog OSS 的样子：

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ' ' # Required; description: Repository path
            base: ' ' # Required; description: Start scanning from here (usually main
 branch).
            head: ' ' # Optional; description: Scan commits until here (usually dev
 branch).
            extra_args: ' ' # Optional; description: Extra args to be passed to the
 trufflehog cli.
```

在前面的代码中，Actions CodeCatalyst GitHub 操作（由标识`aws/github-actions-runner@v1`）封装 TruffleHog OSS 操作（由标识`trufflesecurity/trufflehog@v3.16.0`），使其在工作 CodeCatalyst 流程中运行。

要配置此操作，您需要将 with: 下的空字符串替换为您自己的值。例如：

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ./
            base: main # Required; description: Start scanning from here (usually main
 branch).
            head: HEAD # Optional; description: Scan commits until here (usually dev
 branch).
            extra_args: '--debug --only-verified' # Optional; description: Extra args
 to be passed to the trufflehog cli.
```

要将精心策划的 GitHub 操作添加到工作流程，请按以下步骤操作。有关在 CodeCatalyst 工作流程中使用 GitHub 操作的一般信息，请参阅与 GitHub 操作集成。

> **Note**
>
> 如果您在精选操作列表中看不到您的 GitHub 操作，您仍然可以使用 “操作” GitHub 操作将其添加到工作流程中。有关更多信息，请参阅 添加 “GitHub 操作” 操作。

Visual

使用可视化编辑器添加精选 GitHub 动作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。

6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择GitHub。
9. 浏览或搜索某项 GitHub 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 GitHub 操作的名称。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入、配置和输出选项卡中，根据需要填写字段。有关每个字段的描述，请参阅'GitHub Actions'动作 YAML。本参考提供了有关 “GitHub操作” 操作中可用的每个字段（以及相应的 YAML 属性值）的详细信息，因为该字段同时出现在 YAML 和可视编辑器中。

    有关策划的 Acti GitHub on 可用的配置选项的信息，请参阅其文档。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加精选 GitHub 动作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择GitHub。
9. 浏览或搜索某项 GitHub 操作，然后执行以下任一操作：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

或

- 选择 GitHub 操作的名称。此时会显示操作详细信息对话框。在此对话框中：
  - （可选）选择查看源以查看操作的源代码。
  - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 根据需求修改 YAML 代码中的属性。中提供了对 “GitHub 操作” 操作可用的每个属性的解释'GitHub Actions'动作 YAML。

    有关策划的 Acti GitHub on 可用的配置选项的信息，请参阅其文档。

11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

导出 GitHub 输出参数

您可以在 CodeCatalyst 工作流中使用 GitHub 输出参数。

> **Note**
>
> 输出参数的另一个词是变量。由于 GitHub 在其文档中使用了术语输出参数，因此我们也将使用此术语。

按照以下说明操作，从 GitHub Action 中导出 GitHub 输出参数，以便其他 CodeCatalyst 工作流操作可以使用该参数。

导出 GitHub 输出参数

1. 打开工作流并选择编辑。有关更多信息，请参阅 创建工作流。
2. 在生成要导出的输出参数的 GitHub Actions 操作中，添加一个包含基础 `Variables` 属性的 `Outputs` 部分，如下所示：

```
Actions:
  MyGitHubAction:
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'step-id_output-name'
```

进行如下替换：

- 将 *step-id* 替换为 GitHub 操作的 steps 部分中的 id: 属性的值。
- 将 *output-name* 替换为 GitHub 输出参数的名称。

示例

以下示例演示如何导出名为 SELECTEDCOLOR 的 GitHub 输出参数。

```
Actions:
  MyGitHubAction:
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'random-color-generator_SELECTEDCOLOR'
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
```

引用 GitHub 输出参数

按照以下说明操作以引用 GitHub 输出参数。

引用 GitHub 输出参数

1. 完成导出 GitHub 输出参数中的步骤。

   GitHub 输出参数现在可用于其他操作。
2. 记下输出参数的 Variables 值。它包含下划线（_）。
3. 使用以下语法引用输出参数：

   ```
   ${action-name.output-name}
   ```

   进行如下替换：

- 将 *action-name* 替换为生成输出参数的 CodeCatalyst GitHub Action 的名称（请勿使用 GitHub 操作的 name 或 id）。
- 将 *output-name* 替换为您之前记下的输出参数的 Variables 值。

示例：

```
BuildActionB:
  Identifier: aws/build@v1
  Configuration:
    Steps:
      - Run: echo ${MyGitHubAction.random-color-generator_SELECTEDCOLOR}
```

带上下文的示例

以下示例说明如何在 GitHubActionA 中设置 SELECTEDCOLOR 变量、输出该变量，然后在 BuildActionB 中引用该变量。

```
Actions:
  GitHubActionA:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
    Outputs:
      Variables:
      - 'random-color-generator_SELECTEDCOLOR'

   BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: echo ${GitHubActionA.random-color-generator_SELECTEDCOLOR}
```

'GitHub Actions'动作 YAML

以下是 “GitHub操作” 操作的 YAML 定义。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

在以下代码中选择一个 YAML 属性可查看其描述。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier:  aws/github-actions-runner@v1
    DependsOn:
      - dependent-action-name-1
    Compute:
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
```

```
    Outputs:
      Artifacts:
        - Name: output-artifact-1
          Files:
            - github-output/artifact-1.jar
            - "github-output/build*"
        - Name: output-artifact-2
          Files:
            - github-output/artifact-2.1.jar
            - github-output/artifact-2.2.jar
      Variables:
        - variable-name-1
        - variable-name-2
      AutoDiscoverReports:
        Enabled: true | false
        ReportNamePrefix: AutoDiscovered
        IncludePaths:
          - "**/*"
        ExcludePaths:
          - node_modules/cdk/junit.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
            Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
            Number: whole-number
      Reports:
        report-name-1:
          Format: format
          IncludePaths:
            - "*.xml"
          ExcludePaths:
            - report2.xml
            - report3.xml
          SuccessCriteria:
            PassRate: percent
            LineCoverage: percent
            BranchCoverage: percent
            Vulnerabilities:
              Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
              Number: whole-number
    Configuration
      Steps:
```

```
        - github-actions-code
```

action-name

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

对应的 UI：“配置” 选项卡/ *action-name*

Identifier

(*action-name*/Identifier)

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅 指定要使用的操作版本。

aws/github-actions-runner@v1用于GitHub操作操作。

对应的用户界面：工作流程图// aws***action-name***/@v1 标签 github-actions-runner

DependsOn

(*action-name*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*action-name*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Fleet

(*action-name*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

对应的 UI：“配置”选项卡/计算实例集 – 可选

Timeout

(*action-name*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Environment

(*action-name*/Environment)

（可选）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*action-name*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*action-name*/Environment/Connections)

（可选）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台中环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Name

(*action-name*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Role

(*action-name*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定 IAM 角色的名称，该操作使用此角色来访问 Amazon S3 和 Amazon ECR 等 AWS 服务并在其中进行处理。确保将此角色添加到空间中的 AWS 账户 连接。要向账户连接添加 IAM 角色，请参阅将 IAM 角色添加到账户连接。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

> **Note**
>
> 您可以将 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色用于此操作。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

> **Warning**
>
> 将权限限制为 “GitHub 操作” 操作所需的权限。使用具有更广泛权限的角色可能会带来安全风险。

对应的用户界面：配置在tab/Environment/What里面*my-environment*吗？ /三点菜单/ 切换角色

Inputs

(*action-name*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间操作所需的数据。

> **Note**
>
> 每个 “GitHub 操作” 操作最多允许四个输入（一个源和三个构件）。变量不计入此总数。

如果您需要引用驻留在不同输入（例如源和构件）中的文件，则源输入是主输入，构件是辅助输入。辅助输入中对文件的引用采用特殊前缀，以与主输入中的文件区分开来。有关更多信息，请参阅 示例：引用多个构件中的文件。

对应的 UI：输入选项卡

Sources

(*action-name*/Inputs/Sources)

（可选）

指定表示操作所需的源存储库的标签。当前，支持的唯一标签是 `WorkflowSource`，它表示存储工作流定义文件的源存储库。

如果省略源，则必须在 *action-name*`/Inputs/Artifacts` 下至少指定一个输入构件。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*action-name*/Inputs/Artifacts)

（可选）

指定以前操作中的一些构件，您希望将这些构件用作此操作的输入。这些构件必须已在以前的操作中定义为输出构件。

如果未指定任何输入构件，则必须在 *action-name*/Inputs/Sources 下指定至少一个源存储库。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

> **Note**
>
> 如果构件 – 可选下拉列表不可用（可视化编辑器），或者在验证 YAML 时出现错误（YAML 编辑器），这可能是因该操作仅支持一个输入导致的。在这种情况下，请尝试移除源输入。

对应的 UI：“输入”选项卡/构件 – 可选

Variables - input

(*action-name*/Inputs/Variables)

（可选）

指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输入”选项卡/变量 – 可选

Outputs

(*action-name*/Outputs)

（可选）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts - output

(*action-name*/Outputs/Artifacts)

（可选）

指定操作生成的构件的名称。构件名称在工作流内必须是唯一的，并且仅限于字母数字字符（a-z、A-Z、0-9）和下划线（_）。不允许使用空格、连字符（-）和特殊字符。不能使用引号以使输出构件名称包含空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件

Name

(*action-name*/Outputs/Artifacts/Name)

（如果包含 Artifacts - output，则为必需）

指定操作生成的构件的名称。构件名称在工作流内必须是唯一的，并且仅限于字母数字字符（a-z、A-Z、0-9）和下划线（_）。不允许使用空格、连字符（-）和特殊字符。不能使用引号以使输出构件名称包含空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的用户界面：输出tab/Artifacts/Add构件/构建构件名称

Files

(*action-name*/Outputs/Artifacts/Files)

（如果包含 Artifacts - output，则为必需）

指定由操作输出的构件中 CodeCatalyst 包含的文件。这些文件由工作流操作在运行时生成，也可在您的源存储库中找到。文件路径可以位于源存储库或先前操作的构件中，并且相对于源存储库或构件根目录。您可以使用 glob 模式来指定路径。示例：

- 要指定位于构建位置或源存储库位置根目录中的单个文件，请使用 `my-file.jar`。
- 要在子目录中指定单个文件，请使用 `directory/my-file.jar` 或 `directory/subdirectory/my-file.jar`。
- 要指定所有文件，请使用 `"**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件和目录，请使用 `"directory/**/*"`。** glob 模式表示匹配任意数量的子目录。
- 要指定名为 `directory` 的目录中的所有文件，而非其任意子目录，请使用 `"directory/*"`。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（*）或其他特殊字符，请用双引号（""）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个构件或源中查找它。有关更多信息，请参阅引用源存储库文件和在构件中引用文件。

相应的 UI：输出tab/Artifacts/Add构件/编译生成的文件

Variables - output

(*action-name*/Outputs/Variables)

（可选）

指定希望操作导出的变量，以便后续操作可以使用这些变量。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输出”选项卡/变量/添加变量

variable-name-1

（*action-name*/Outputs/Variables变量名称-1）

（可选）

指定希望操作导出的变量的名称。此变量必须已在同一操作的 `Inputs` 或 `Steps` 部分中定义。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的用户界面：输出tab/Variables/Add变量/名称

AutoDiscoverReports

(*action-name*/Outputs/AutoDiscoverReports)

（可选）

定义自动发现功能的配置。

启用自动发现后，会 CodeCatalyst 搜索操作中`Inputs`传递的所有文件以及操作本身生成的所有文件，以查找测试、代码覆盖率和软件组合分析 (SCA) 报告。对于找到的每个报告，将其 CodeCatalyst 转换为 CodeCatalyst 报告。CodeCatalyst 报告是完全集成到 CodeCatalyst 服务中的报告，可以通过 CodeCatalyst 控制台查看和操作。

> **Note**
>
> 默认情况下，自动发现功能会检查所有文件。您可以使用 IncludePaths 或 ExcludePaths 属性限制检查哪些文件。

对应的 UI：无

Enabled

(*action-name*/Outputs/AutoDiscoverReports/Enabled)

（可选）

启用或禁用自动发现功能。

有效值为 `true` 或 `false`。

如果省略 `Enabled`，则默认值为 `true`。

对应的 UI：“输出”选项卡/报告/自动发现报告

ReportNamePrefix

(*action-name*/Outputs/AutoDiscoverReports/ReportNamePrefix)

（如果包含并启用 AutoDiscoverReports，则为必需）

为其找到的所有报告指定一个前缀，以便命名其关联 CodeCatalyst 的报告。 CodeCatalyst 例如，如果您将前缀指定为`AutoDiscovered`，并 CodeCatalyst自动发现两个测试报告`TestSuiteTwo.xml`，`TestSuiteOne.xml`则关联 CodeCatalyst 的报告将命名为 `AutoDiscoveredTestSuiteOne` and。`AutoDiscoveredTestSuiteTwo`

相应的 UI：输出tab/Reports/Automatically发现报告/ 报告前缀

IncludePaths

(*action-name*/Outputs/AutoDiscoverReports/IncludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/IncludePaths)

（如果包含并启用 AutoDiscoverReports，或者包含 Reports，则为必需）

指定搜索原始报告时 CodeCatalyst 包含的文件和文件路径。例如，如果您指定"/test/report/*"，则会在操作使用的整个构建映像中 CodeCatalyst 搜索该/test/report/*目录。当它找到该目录时， CodeCatalyst 然后在该目录中查找报告。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（*）或其他特殊字符，请用双引号（""）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

如果省略此属性，则默认值为 "**/*"，这意味着搜索范围包括所有路径的所有文件。

> **Note**
>
> 对于手动配置的报告，IncludePaths 必须是与单个文件匹配的 glob 模式。

对应的 UI：

- 输出tab/Reports/Automatically discover reports/'Include/exclude路径'/包括路径
- 输出tab/Reports/Manually配置报告/ ***report-name-1*** /'包含/排除路径'/包含路径

ExcludePaths

(*action-name*/Outputs/AutoDiscoverReports/ExcludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/ExcludePaths)

（可选）

指定搜索原始报告时 CodeCatalyst 排除的文件和文件路径。例如，如果您指定"/test/my-reports/**/*"，则 CodeCatalyst 不会在/test/my-reports/目录中搜索文件。要忽略某个目录中的所有文件，请使用 **/* glob 模式。

> **Note**
>
> 如果您的文件路径包含一个或多个星号（*）或其他特殊字符，请用双引号（""）将路径括起来。有关特殊字符的更多信息，请参阅语法准则和惯例。

对应的 UI：

- 输出tab/Reports/Automatically discover reports/'Include/exclude路径'/排除路径
- 输出tab/Reports/Manually配置报告/ ***report-name-1*** /'包含/排除路径'/排除路径

SuccessCriteria

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria)

（可选）

为测试、代码覆盖率、软件组成分析（SCA）和静态分析（SA）报告指定成功标准。

有关更多信息，请参阅 配置报告的成功标准。

对应的 UI：

- 产出tab/Reports/Automatically发现报告/ 成功标准
- 输出tab/Reports/Manually配置报告/*report-name-1*/成功标准

PassRate

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/PassRate)

（可选）

指定测试报告中必须通过测试的百分比，关联 CodeCatalyst的报告才会被标记为通过。有效值包括十进制数字。例如：50、60.5。通过率标准仅适用于测试报告。有关测试报告的更多信息，请参阅测试报告。

对应的 UI：

- 输出tab/Reports/Automatically discover reports/Success标准/通过率
- 输出tab/Reports/Manually配置报告 ***report-name-1*** //成功标准/通过率

LineCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/LineCoverage)

（可选）

指定代码覆盖率报告中必须覆盖的行数百分比，关联 CodeCatalyst 的报告才会被标记为通过。有效值包括十进制数字。例如：50、60.5。行覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

对应的 UI：

- 输出tab/Reports/Automatically discover reports/Success标准/线路覆盖范围
- 输出tab/Reports/Manually配置报告 ***report-name-1*** //成功标准/线路覆盖率

BranchCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/BranchCoverage)

（可选）

指定代码覆盖率报告中必须覆盖的分支百分比才能将关联 CodeCatalyst 报告标记为已通过。有效值包括十进制数字。例如：50、60.5。分支覆盖率标准仅适用于代码覆盖率报告。有关代码覆盖率报告的更多信息，请参阅代码覆盖率报告。

对应的 UI：

- 产出tab/Reports/Automatically discover reports/Success标准/分支机构覆盖范围
- 输出tab/Reports/Manually配置报告 ***report-name-1*** //成功标准/分支覆盖范围

Vulnerabilities

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/Vulnerabilities)

（可选）

指定 SCA 报告中允许将关联 CodeCatalyst 报告标记为已通过的最大漏洞数量和严重性。要指定漏洞，您必须指定：

- 要计入的漏洞的最低严重性。有效值（按严重程度从高到低）为 CRITICAL、HIGH、MEDIUM、LOW 和 INFORMATIONAL。

  例如，如果您选择 HIGH，则将计算 HIGH 和 CRITICAL 漏洞的总数。
- 您希望允许的具有指定严重性的漏洞的最大数量。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

漏洞标准仅适用于 SCA 报告。有关 SCA 报告的更多信息，请参阅软件组成分析报告。

要指定最低严重性，请使用 Severity 属性。要指定最大漏洞数，请使用 Number 属性。

有关 SCA 报告的更多信息，请参阅质量报告类型。

对应的 UI：

- 输出tab/Reports/Automatically discover reports/Success标准/漏洞
- 输出tab/Reports/Manually配置报告 ***report-name-1*** //成功标准/漏洞

Reports

(*action-name*/Outputs/Reports)

（可选）

一个部分，用于指定测试报告的配置。

对应的 UI：“输出”选项卡/报告

report-name-1

（*action-name*/Outputs/Reports/报告名称 1）

（如果包含 Reports，则为必需）

您要为将从原始 CodeCatalyst 报告生成的报告命名。

相应的 UI：输出tab/Reports/Manually配置报告/ 报告名称

Format

(*action-name*/Outputs/Reports/*report-name-1*/Format)

（如果包含 Reports，则为必需）

指定您用于报告的文件格式。可能值如下所示。

- 对于测试报告：
  - 对于 Cucumber JSON，请指定 Cucumber（可视化编辑器）或 CUCUMBERJSON（YAML 编辑器）。
  - 对于 JUnit XML，请指定 JUnit（可视化编辑器）或JUNITXML（YAML 编辑器）。
  - 对于 NUnit XML，请指定 NUnit（可视化编辑器）或NUNITXML（YAML 编辑器）。
  - 对于 NUnit 3 XML，请指定 NUnit3（可视化编辑器）或NUNIT3XML（YAML 编辑器）。
  - 对于 Visual Studio TRX，请指定 Visual Studio TRX（可视化编辑器）或 VISUALSTUDIOTRX（YAML 编辑器）。
  - 对于 TestNG XML，请指定 TestNG（可视化编辑器）或 TESTNGXML（YAML 编辑器）。
- 对于代码覆盖率报告：
  - 对于 Clover XML，请指定 Clover（可视化编辑器）或 CLOVERXML（YAML 编辑器）。

- 对于 Cobertura XML，请指定 Cobertura（可视化编辑器）或 COBERTURAXML（YAML 编辑器）。
- 对于 JaCoCo XML，请指定 JaCoCo（可视化编辑器）或JACOCOXML（YAML 编辑器）。
- 对于由 s implecov 生成的 SimpleCov JSON，而不是 s implecov-json，请指定 S implecov（可视化编辑器）或（YAML 编辑器）。SIMPLECOV

- 对于软件组成分析（SCA）报告：
  - 对于 SARIF，请指定 SARIF（可视化编辑器）或 SARIFSCA（YAML 编辑器）。

对应的用户界面：输出tab/Reports/Manually configure reports/Add报告*report-name-1*//报告类型和报告格式

Configuration

(*action-name*/Configuration)

（必需）可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

Steps

(*action-name*/Configuration/Steps)

（必需）

在 GitHub M GitHub arketplac e 中指定操作详情页面上显示的操作代码。按照以下指南来添加代码：

1. 将 “ GitHub 操作steps:” 部分中的代码粘贴到 CodeCatalyst 工作流程的Steps:部分中。该代码以短划线（-）开头，与以下内容类似。

   GitHub 要粘贴的代码：

   ```
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: master
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

2. 查看您刚刚粘贴的代码，并在必要时对其进行修改，使其符合标准。 CodeCatalyst例如，在前面的代码块中，您可以删除中的代码*red italics*，然后添加粗体代码。

CodeCatalyst 工作流程 yaml：

```
Steps:
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: mastermain
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

3. 对于 GitHub 操作中包含但该`steps:`部分中不存在的其他代码，请使用 CodeCatalyst等效代码将其添加到 CodeCatalyst 工作流程中。您可以查看，工作流 YAML 定义以深入了解如何将 GitHub 代码移植到 CodeCatalyst。详细的迁移步骤不在本指南的讨论范围内。

以下是如何在 “GitHub 操作” 操作中指定文件路径的示例：

```
Steps:
  - name: Lint Code Base
    uses: github/super-linter@v4
    ...
  - run: cd /sources/WorkflowSource/MyFolder/  && cat file.txt
  - run: cd /artifacts/MyGitHubAction/MyArtifact/MyFolder/  && cat file2.txt
```

有关指定文件路径的更多信息，请参阅引用源存储库文件和在构件中引用文件。

对应的用户界面：“配置” 选项卡/ GitHub 操作 YAML

配置计算和运行时映像

在 CodeCatalyst 工作流程中，您可以指定用于运行工作流程操作的 CodeCatalyst 计算和运行时环境映像。

计算是指为运行工作流操作而管理和维护的计算引擎（CPU、内存和操作系统）。 CodeCatalyst

> **Note**
>
> 如果计算被定义为工作流的属性，则不能将其定义为该工作流中任何操作的属性。同样，如果计算被定义为任何操作的属性，则无法将其定义为工作流中的属性。

运行时环境镜像是一个 Docker 容器，在其中 CodeCatalyst运行工作流程操作。Docker 容器在您选择的计算平台上运行，包括操作系统和工作流程操作可能需要的额外工具，例如 AWS CLI、Node.js 和.tar。

主题

计算类型

CodeCatalyst 提供以下计算类型：

- Amazon EC2
- AWS Lambda

Amazon 在操作运行期间 EC2 提供了优化的灵活性，而 Lambda 则提供了优化的操作启动速度。由于启动延迟较短，Lambda 支持更快的工作流操作运行。Lambda 可以运行基本工作流，这些工作流可以构建、测试和部署具有常见运行时系统的无服务器应用程序。这些运行时包括 Node.js、Python、Java、.NET 和 Go。但是，有些用例是 Lambda 不支持的，如果它们对您造成影响，请使用 Amazon 计算类型： EC2

- Lambda 不支持来自指定注册表的运行时环境映像。
- Lambda 不支持需要 root 权限的工具。对于`yum`或之类的工具`rpm`，请使用 Amazon EC2 计算类型或其他不需要根权限的工具。
- Lambda 不支持 Docker 构建或运行。不支持以下使用 Docker 镜像的操作：部署 AWS CloudFormation 堆栈、部署到 Amazon ECS、Amazon S3 发布、 AWS CDK 引导、 AWS CDK 部

署、 AWS Lambda 调用和 GitHub 操作。Lambda GitHub 计算也不支持在 “ CodeCatalyst GitHub 操作” 操作中运行的基于 Docker 的操作。您可以使用不需要 root 权限的替代方案，例如 Podman。

- Lambda 不支持写入到 `/tmp` 外部的文件。配置工作流操作时，您可以重新配置工具来安装或写入到 `/tmp`。如果您的构件操作要安装 npm，请务必将其配置为安装到 `/tmp`。
- Lambda 不支持超过 15 分钟的运行时。

计算实例集

CodeCatalyst 提供以下计算队列：

- 按需车队
- 预置实例集

对于按需实例集，当工作流操作启动时，工作流会预置所需的资源。操作完成后，计算机就会被销毁。您只需为运行操作的分钟数付费。按需实例集是完全托管式的，并包括自动扩展功能以应对需求激增。

CodeCatalyst 还提供预配置的队列，其中包含由 Amazon 提供支持并由其维护 EC2 的机器。CodeCatalyst使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。使用预置实例集，您的计算机将始终处于运行状态，并且只要预置完毕，它们就会产生成本。

要创建、更新或删除实例集，您必须拥有空间管理员角色或项目管理员角色。

按需实例集属性

CodeCatalyst 提供以下按需队列：

Name	操作系统	架构	v CPUs	内存（GiB）	磁盘空间	支持的计算类型
`Linux.Arm64.Large`	Amazon Linux 2	Arm64	2	4	64 GB	Amazon EC2
					10 GB	Lambda
`Linux.Arm64.XLarge`	Amazon Linux 2	Arm64	4	8	128 GB	Amazon EC2

Name	操作系统	架构	v CPUs	内存（GiB）	磁盘空间	支持的计算类型
					10 GB	Lambda
`Linux.Arm64.2XLarge`	Amazon Linux 2	Arm64	8	16	128 GB	Amazon EC2
`Linux.x86-64.Large`	Amazon Linux 2	x86-64	2	4	64 GB	Amazon EC2
					10 GB	Lambda
`Linux.x86-64.XLarge`	Amazon Linux 2	x86-64	4	8	128 GB	Amazon EC2
					10 GB	Lambda
`Linux.x86-64.2XLarge`	Amazon Linux 2	x86-64	8	16	128 GB	Amazon EC2

> **Note**
>
> 按需实例集的规格将根据您的计费等级而有所不同。有关更多信息，请参阅定价。

如果未选择任何舰队，则 CodeCatalyst 使用`Linux.x86-64.Large`。

预置实例集属性

预置实例集包含以下属性：

操作系统

操作系统 以下操作系统可用：

- Amazon Linux 2
- Windows Server 2022

> **Note**
>
> 只有在构建操作中才支持 Windows 实例集。其他操作目前不支持 Windows。

架构

处理器架构。以下架构可用：

- x86_64
- Arm64

计算机类型

每个实例的计算机类型。以下计算机类型可用：

v CPUs	内存（GiB）	磁盘空间	操作系统
2	4	64 GB	Amazon Linux 2
4	8	128 GB	Amazon Linux 2
			Windows Server 2022
8	16	128 GB	Amazon Linux 2
			Windows Server 2022

Capacity

分配给实例集的计算机的初始数量，它定义了可以并行运行的操作数量。

扩展模式

定义操作数量超过实例集容量时的行为。

按需预置额外容量

按需设置其它计算机，这些计算机根据正在运行的新操作自动纵向扩展，然后在操作完成时缩减到基本容量。这可能会产生额外的成本，因为您需要为每台运行的计算机按分钟付费。

等到有额外的实例集容量可用

操作运行将放在队列中，直到有计算机可用。这限制了额外成本，因为没有分配额外的计算机。

创建预置的实例集

按照以下说明操作来创建预置的实例集。

> **Note**
>
> 预置的实例集将在处于不活动状态 2 周后被停用。如果重新使用这些实例集，它们将自动重新激活，但这种重新激活可能会导致延迟。

创建预置的实例集

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 选择创建预置实例集。
3. 在预置机群名称文本字段中，输入实例集的名称。
4. 从操作系统下拉菜单中，选择操作系统。
5. 从计算机类型下拉菜单中，选择您的计算机的计算机类型。
6. 在容量文本字段中，输入实例集中的最大机器数。
7. 从扩展模式下拉菜单中，选择所需的溢出行为。有关这些字段的更多信息，请参阅预置实例集属性。
8. 选择创建。

创建预置实例集后，便可将其分配给操作。有关更多信息，请参阅 向操作分配实例集或计算。

编辑预置的实例集

按照以下说明操作来编辑预置的实例集。

> **Note**
>
> 预置的实例集将在处于不活动状态 2 周后被停用。如果重新使用这些实例集，它们将自动重新激活，但这种重新激活可能会导致延迟。

编辑预置的实例集

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 在预置实例集列表中，选择要编辑的实例集。
3. 选择编辑。
4. 在容量文本字段中，输入实例集中的最大机器数。
5. 从扩展模式下拉菜单中，选择所需的溢出行为。有关这些字段的更多信息，请参阅预置实例集属性。
6. 选择保存。

删除预置的实例集

按照以下说明操作来删除预置的实例集。

删除预置的实例集

> **Warning**
>
> 在删除某个预置的实例集之前，请从操作的 YAML 代码中删除 `Fleet` 属性，以从所有操作中移除该实例集。任何在删除预置实例集后继续引用该实例集的操作都将在下次运行时失败。

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 在预置实例集列表中，选择要删除的实例集。
3. 选择删除。
4. 输入 **`delete`** 以确认删除。
5. 选择删除。

向操作分配实例集或计算

默认情况下，工作流程操作使用具有 Amazon EC2 计算类型的`Linux.x86-64.Large`按需队列。要改为使用预置的实例集，或者使用其他按需实例集（例如 `Linux.x86-64.2XLarge`），请按照以下说明操作。

Visual

开始前的准备工作

- 如果您要分配预置实例集，则必须先创建预置实例集。有关更多信息，请参阅创建预置的实例集。

向操作分配预置实例集或其他实例集类型

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要向其分配预置实例集或新实例集类型的操作。
8. 选择配置选项卡。
9. 在计算实例集中，执行以下操作：

   指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

   使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

   如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

开始前的准备工作

- 如果您要分配预置实例集，则必须先创建预置实例集。有关更多信息，请参阅创建预置的实例集。

向操作分配预置实例集或其他实例集类型

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 找到您要向其分配预置实例集或新实例集类型的操作。
8. 在操作中，添加一个 `Compute` 属性并将 `Fleet` 设置为您的实例集名称或按需实例集类型。有关更多信息，请参阅构建和测试操作 YAML 中相应操作 `Fleet` 属性的说明。
9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

跨操作共享计算

默认情况下，工作流中的操作在实例集中的不同实例上运行。这种行为为操作提供了隔离性以及输入状态的可预测性。默认行为需要明确配置，以便在操作之间共享上下文，例如文件和变量。

计算共享是一种功能，让您能够在同一个实例上运行工作流中的所有操作。使用计算共享可以缩短工作流运行时间，因为预置实例所花费的时间更少。您也可以在操作之间共享文件（构件），而无需执行额外的工作流配置。

使用计算共享运行工作流时，默认实例集或指定实例集中的实例将在该工作流中的所有操作持续时间内予以预留。工作流运行完成后，就会释放实例预留。

主题

- 在共享计算上运行多个操作

- 计算共享注意事项
- 启用计算共享
- 示例

在共享计算上运行多个操作

您可以在工作流级别，使用定义 YAML 中的 Compute 属性来指定操作的实例集和计算共享属性。您还可以使用 CodeCatalyst 中的可视化编辑器配置计算属性。要指定实例集，请设置现有实例集的名称，将计算类型设置为 EC2，然后启用计算共享。

> **Note**
>
> 只有当计算类型设置为 EC2 时，才支持计算共享，并且 Windows Server 2022 操作系统不支持计算共享。有关计算实例集、计算类型和属性的更多信息，请参阅配置计算和运行时映像。

> **Note**
>
> 如果您使用的是免费套餐，并且在工作流定义 YAML 中手动指定 Linux.x86-64.XLarge 或 Linux.x86-64.2XLarge 实例集，则该操作仍将在默认实例集（Linux.x86-64.Large）上运行。有关计算可用性和定价的更多信息，请参阅套餐选项表。

在启用计算共享后，将自动跨操作复制包含工作流源的文件夹。您不需要配置输出构件，也无需在整个工作流定义（YAML 文件）中将它们作为输入工件引用。作为工作流的作者，您需要使用输入和输出连接环境变量，与未使用计算共享时一样。如果要在工作流源之外的操作之间共享文件夹，请考虑使用文件缓存。有关更多信息，请参阅在操作之间共享构件和文件和在工作流运行之间缓存文件。

您的工作流定义文件所在的源存储库由标签 WorkflowSource 来标识。使用计算共享时，将在引用该工作流源的第一个操作中下载工作流源，并自动提供给工作流运行中的后续操作使用。如果某个操作（例如添加、修改或删除文件）对包含工作流源的文件夹执行任何更改，这些更改也将在工作流的后续操作中可见。您可以在任何工作流操作中引用工作流源文件夹中的文件，与未使用计算共享时一样。有关更多信息，请参阅 引用源存储库文件。

> **Note**
>
> 计算共享工作流需要指定严格的操作顺序，因此无法设置并行操作。虽然可以在序列中的任何操作中配置输出构件，但不支持输入构件。

计算共享注意事项

您可以利用计算共享功能运行工作流，以便加快工作流运行，并且可以在使用相同实例的工作流中的操作之间共享上下文。要确定使用计算共享是否适合您的场景，请考虑以下因素：

	计算共享	不使用计算共享
计算类型	Amazon EC2	Amazon EC2、AWS Lambda
实例预置	操作在同一实例上运行	操作在单独的实例上运行
操作系统	Amazon Linux 2	Amazon Linux 2、Windows Server 2022（仅限构建操作）
引用文件	`$CATALYST_SOURCE_DIR_WorkflowSource`，`/sources/WorkflowSource/`	`$CATALYST_SOURCE_DIR_WorkflowSource`，`/sources/WorkflowSource/`
工作流结构	操作只能按顺序运行	操作可以并行运行
跨工作流操作访问数据	访问缓存的工作流来源（`WorkflowSource`）	访问共享构件的输出（需要额外配置）

启用计算共享

请按照以下说明，为工作流启用计算共享。

Visual

使用可视化编辑器启用计算共享

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。
5. 选择编辑。
6. 选择可视化。
7. 选择工作流属性。
8. 从计算类型下拉菜单中，选择 EC2。
9. （可选）从计算实例集 – 可选下拉菜单中，选择要用于运行工作流操作的实例集。您可以选择按需实例集，也可以创建并选择预置实例集。有关更多信息，请参阅 创建预置的实例集 和 向操作分配实例集或计算
10. 切换开关以启用计算共享，让工作流中的操作在同一个实例集上运行。
11. （可选）选择工作流的运行模式。有关更多信息，请参阅 配置运行的排队行为。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器启用计算共享

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。
5. 选择编辑。
6. 选择 YAML。
7. 将 `SharedInstance` 字段设置为 `TRUE` 并将 `Type` 设置为 `EC2`，从而启用计算共享。将 `Fleet` 设置为要用于运行工作流操作的计算实例集。您可以选择按需实例集，也可以创建并选择预置实例集。有关更多信息，请参阅 创建预置的实例集 和 向操作分配实例集或计算

   在工作流 YAML 中，添加类似于以下代码的代码：

```
Name: MyWorkflow
SchemaVersion: "1.0"
Compute: # Define compute configuration.
  Type: EC2
```

```
    Fleet: MyFleet # Optionally, choose an on-demand or provisioned fleet.
    SharedInstance: true # Turn on compute sharing. Default is False.
  Actions:
    BuildFirst:
      Identifier: aws/build@v1
      Inputs:
        Sources:
          - WorkflowSource
      Configuration:
        Steps:
          - Run: ...
          ...
```

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

示例

主题

- 示例：Amazon S3 发布

示例：Amazon S3 发布

以下工作流示例说明了如何通过两种方式执行 Amazon S3 发布操作：首先使用输入构件，然后使用计算共享。使用计算共享时，不需要输入构件，因为您可以访问缓存的 `WorkflowSource`。此外，不再需要“构建”操作中的输出构件。S3 发布操作配置为使用显式 `DependsOn` 属性来保持连续操作；“构建”操作必须成功运行才能运行 S3 发布操作。

- 如果未使用计算共享，则需要使用输入构件并将输出与后续操作共享：

```
Name: S3PublishUsingInputArtifact
SchemaVersion: "1.0"
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ArtifactToPublish
          Files: [output.zip]
    Inputs:
```

```
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    Inputs:
      Artifacts:
      - ArtifactToPublish
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
      DestinationBucketName: amzn-s3-demo-bucket
```

- 通过将 SharedInstance 设置为 TRUE 来使用计算共享，您可以对同一个实例运行多个操作并通过指定单个工作流源来共享构件。输入构件不是必需的，也无法指定：

```
Name: S3PublishUsingComputeSharing
SchemaVersion: "1.0"
Compute:
  Type: EC2
  Fleet: dev-fleet
  SharedInstance: TRUE
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - Build
```

```
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
      DestinationBucketName: amzn-s3-demo-bucket
```

指定运行时环境映像

运行时环境镜像是一个 Docker 容器，在其中 CodeCatalyst运行工作流程操作。Docker 容器在您选择的计算平台上运行，包括操作系统和工作流程操作可能需要的额外工具，例如 AWS CLI、Node.js 和.tar。

默认情况下，工作流程操作将在由提供和维护的其中一个活动图像上运行 CodeCatalyst。仅构建操作和测试操作支持自定义映像。有关更多信息，请参阅 为操作分配自定义运行时环境 Docker 映像。

主题

- 活动映像
- 如果活动映像未包括我需要的工具，该怎么办？
- 为操作分配自定义运行时环境 Docker 映像
- 示例

活动映像

活动镜像是运行时环境镜像，完全支持 CodeCatalyst 并包含预安装的工具。目前有两组活动映像：一组于 2024 年 3 月发布，另一组于 2022 年 11 月发布。

操作是使用 2024 年 3 月版还是 2022 年 11 月版映像取决于：

- 在 2024 年 3 月 26 日当天或之后添加到工作流中的构建操作和测试操作将在其 YAML 定义中包含一个 `Container` 部分，用于明确指定 2024 年 3 月版映像。您可以选择移除 `Container` 部分以恢复到 2022 年 11 月版映像。
- 在 2024 年 3 月 26 日之前添加到工作流中的构建操作和测试操作将不会在其 YAML 定义中包含 `Container` 部分，因此将使用 2022 年 11 月版映像。您可以保留 2022 年 11 月版映像，也

可以升级该映像。要升级映像，请在可视化编辑器中打开操作，选择配置选项卡，然后从运行时环境 Docker 映像下拉列表中选择 2024 年 3 月版映像。此选择将在操作的 YAML 定义中添加 `Container` 部分，而该部分中将填入相应的 2024 年 3 月版映像。

- 其他所有操作将使用 2022 年 11 月版映像或 2024 年 3 月版映像。有关更多信息，请参阅操作的文档。

主题

2024 年 3 月版映像

2024 年 3 月的图片是提供的最新图片。 CodeCatalyst每个计算 type/fleet 组合有一张 2024 年 3 月的图像。

下表显示在每个 2024 年 3 月版映像上安装的工具。

2024 年 3 月版映像工具

工具	CodeCatalyst EC2 适用于 Linux 的亚马逊 x86_64-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst EC2 适用于 Linux 的亚马逊 Arm64-**CodeCatalystLinux_Arm64:2024_03**	CodeCatal 于 Linux 的 Arm64-**CodCatalystLinuxLambda_Ar64:2024_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS Copilot CLI	1.32.1	1.32.1	1.32.1	1.32.1
Docker	24.0.9	不适用	24.0.9	不适用
Docker Compose	2.23.3	不适用	2.23.3	不适用
Git	2.43.0	2.43.0	2.43.0	2.43.0
Go	1.21.5	1.21.5	1.21.5	1.21.5

工具	CodeCatalyst EC2 适用于 Linux 的亚马逊 x86_64-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst EC2 适用于 Linux 的亚马逊 Arm64-**CodeCatalystLinux_Arm64:2024_03**	CodeCatal 于 Linux 的 Arm64-**CodeCatalystLinuxLambda_Arm64:2024_**
Gradle	8.5	8.5	8.5	8.5
Java	Corretto17	Corretto17	Corretto17	Corretto17
Maven	3.9.6	3.9.6	3.9.6	3.9.6
Node.js	18.19.0	18.19.0	18.19.0	18.19.0
npm	10.2.3	10.2.3	10.2.3	10.2.3
Python	3.9.18	3.9.18	3.9.18	3.9.18
Python3	3.11.6	3.11.6	3.11.6	3.11.6
pip	22.3.1	22.3.1	22.3.1	22.3.1
.NET	8.0.100	8.0.100	8.0.100	8.0.100

2022 年 11 月版映像

每个计算 type/fleet 组合都有一个 2022 年 11 月的图像。如果您配置了预置实例集，则还可将 2022 年 11 月版 Windows 映像用于构建操作。

下表显示在每个 2022 年 11 月版映像上安装的工具。

2022 年 11 月版映像工具

工具	CodeCatalyst EC2 适用于 Linux 的亚马逊 x86_64-**CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst EC2 适用于 Linux 的亚马逊 Arm64-**CodeCatalystLinux_Arm64:2022_11**	CodeCatal 于 Linux 的 Arm64-**Coc ystLinux ambda_Ar 64:2022_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS Copilot CLI	0.6.0	0.6.0	不适用	不适用
Docker	23.01	不适用	23.0.1	不适用
Docker Compose	2.16.0	不适用	2.16.0	不适用
Git	2.40.0	2.40.0	2.39.2	2.39.2
Go	1.20.2	1.20.2	1.20.1	1.20.1
Gradle	8.0.2	8.0.2	8.0.1	8.0.1
Java	Corretto17	Corretto17	Corretto17	Corretto17
Maven	3.9.4	3.9.4	3.9.0	3.9.0
Node.js	16.20.2	16.20.2	16.19.1	16.14.2
npm	8.19.4	8.19.4	8.19.3	8.5.0
Python	3.9.15	2.7.18	3.11.2	2.7.18
Python3	不适用	3.9.15	不适用	3.11.2
pip	22.2	22.2	23.0.1	23.0.1
.NET	6.0.407	6.0.407	6.0.406	6.0.406

如果活动映像未包括我需要的工具，该怎么办？

如果提供的活动图像均不 CodeCatalyst 包含您需要的工具，则有以下几种选择：

- 您可以提供包含必要工具的自定义运行时环境 Docker 映像。有关更多信息，请参阅 为操作分配自定义运行时环境 Docker 映像。

  > **Note**
  >
  > 如果要提供自定义运行时环境 Docker 映像，请确保您的自定义映像中安装了 Git。

- 您可以让工作流的构建操作或测试操作安装所需的工具。

  例如，您可以在构建操作或测试操作的 YAML 代码的 `Steps` 部分中包含以下指令：

  ```
  Configuration:
    Steps:
      - Run: ./setup-script
  ```

  然后，该*`setup-script`*指令将运行以下脚本来安装 Node 包管理器 (npm)：

  ```
  #!/usr/bin/env bash
  echo "Setting up environment"

  touch ~/.bashrc
  curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
  source ~/.bashrc
  nvm install v16.1.0
  source ~/.bashrc
  ```

  有关构建操作 YAML 的更多信息，请参阅构建和测试操作 YAML。

为操作分配自定义运行时环境 Docker 映像

如果您不想使用提供的 A ctive 镜像 CodeCatalyst，可以提供自定义运行时环境 Docker 镜像。如果要提供自定义映像，请确保其中安装了 Git。映像可以存放在 Docker Hub、Amazon Elastic Container Registry 或任何公共存储库中。

要了解如何创建自定义 Docker 映像，请参阅 Docker 文档中的 Containerize an application。

按照以下说明操作，将您的自定义运行时环境 Docker 映像分配给操作。指定映像后，在操作开始时将其 CodeCatalyst 部署到您的计算平台。

> **Note**
>
> 以下操作不支持自定义运行时环境 Docker 镜像：部署 CloudFormation 堆栈、部署到 ECS 和GitHub 操作。自定义运行时环境 Docker 镜像也不支持 Lambda 计算类型。

Visual

使用可视化编辑器分配自定义运行时环境 Docker 映像

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择可视化。
6. 在工作流图中，选择将使用您的自定义运行时环境 Docker 映像的操作。
7. 选择配置选项卡。
8. 在底部附近，请填写以下字段。

   运行时环境 Docker 映像 – 可选

   指定存储映像的注册表。有效值包括：

   - `CODECATALYST`（YAML 编辑器）

     图像存储在 CodeCatalyst 注册表中。
   - Docker Hub（可视化编辑器）或 `DockerHub`（YAML 编辑器）

     映像存储在 Docker Hub 映像注册表中。
   - 其他注册表（可视化编辑器）或 `Other`（YAML 编辑器）

     映像存储在自定义映像注册表中。可以使用任何公开可用的注册表。
   - Amazon Elastic Container Registry（可视化编辑器）或 `ECR`（YAML 编辑器）

映像存储在 Amazon Elastic Container Registry 映像存储库中。要使用 Amazon ECR 存储库中的映像，此操作需要对 Amazon ECR 的访问权限。要启用此访问权限，您必须创建包含以下权限和自定义信任策略的 IAM 角色。（如果需要，可以修改现有角色以包含这些权限和策略。）

IAM 角色必须在其角色策略中包含以下权限：

- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:GetAuthorizationToken`
- `ecr:GetDownloadUrlForLayer`

IAM 角色必须包含以下自定义信任策略：

有关如何创建 IAM 角色的更多信息，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

创建该角色后，必须通过环境将该角色分配给操作。有关更多信息，请参阅 将环境与操作关联。

ECR 映像 URL、Docker Hub 映像或映像 URL

指定下列项之一：

- 如果您使用的是 CODECATALYST 注册表，请将映像设置为以下活动映像之一：
  - `CodeCatalystLinux_x86_64:2024_03`
  - `CodeCatalystLinux_x86_64:2022_11`
  - `CodeCatalystLinux_Arm64:2024_03`
  - `CodeCatalystLinux_Arm64:2022_11`
  - `CodeCatalystLinuxLambda_x86_64:2024_03`
  - `CodeCatalystLinuxLambda_x86_64:2022_11`
  - `CodeCatalystLinuxLambda_Arm64:2024_03`
  - `CodeCatalystLinuxLambda_Arm64:2022_11`
  - `CodeCatalystWindows_x86_64:2022_11`
- 如果您使用的是 Docker Hub 注册表，请将映像设置为 Docker Hub 映像名称和可选标签。

示例：postgres:latest

- 如果您使用的是 Amazon ECR 注册表，请将映像设置为 Amazon ECR 注册表 URI。

  示例：111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo

- 如果您使用的是自定义注册表，请将映像设置为自定义注册表所期望的值。

9. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
10. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器分配自定义运行时环境 Docker 映像

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
3. 选择编辑。
4. 选择 YAML。
5. 找到要为其分配运行时环境 Docker 映像的操作。
6. 在操作中，添加一个 Container 部分以及底层的 Registry 和 Image 属性。有关更多信息，请参阅操作中针对操作的 Container、Registry 和 Image 属性的描述。
7. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
8. 选择提交，输入提交消息，然后再次选择提交。

示例

以下示例说明如何在工作流定义文件中将自定义运行时环境 Docker 映像分配给操作。

主题

- 示例：使用自定义运行时环境 Docker 映像通过 Amazon ECR 添加对 Node.js 18 的支持
- 示例：使用自定义运行时环境 Docker 映像通过 Docker Hub 添加对 Node.js 18 的支持

示例：使用自定义运行时环境 Docker 映像通过 Amazon ECR 添加对 Node.js 18 的支持

以下示例说明如何使用自定义运行时环境 Docker 映像通过 Amazon ECR 添加对 Node.js 18 的支持。

```
Configuration:
  Container:
    Registry: ECR
    Image: public.ecr.aws/amazonlinux/amazonlinux:2023
```

示例：使用自定义运行时环境 Docker 映像通过 Docker Hub 添加对 Node.js 18 的支持

以下示例说明如何使用自定义运行时环境 Docker 映像通过 Docker Hub 添加对 Node.js 18 的支持。

```
Configuration:
  Container:
    Registry: DockerHub
    Image: node:18.18.2
```

将源存储库连接到工作流

源也称为输入源，是一个源存储库，工作流操作连接到该存储库以获取执行操作所需的文件。例如，工作流操作可能连接到源存储库来获取应用程序源文件，以便构建应用程序。

CodeCatalyst 工作流支持以下来源：

- CodeCatalyst 源存储库-有关更多信息，请参阅使用源存储库存储代码并协作处理代码 CodeCatalyst。
- GitHub 存储库、Bitbucket 存储库和 GitLab 项目存储库 — 有关更多信息，请参阅为带有扩展程序的项目添加功能 CodeCatalyst。

主题

- 指定工作流文件的源存储库
- 指定工作流操作的源存储库
- 引用源存储库文件
- BranchName'和' CommitId '变量

指定工作流文件的源存储库

按照以下说明指定要存储工作流程定义文件的 CodeCatalyst 源存储库。如果您想指定 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，请参阅。为带有扩展程序的项目添加功能 CodeCatalyst

您的工作流定义文件所在的源存储库由标签 `WorkflowSource` 来标识。

> **Note**
>
> 首次提交工作流定义文件时，您可以指定工作流定义文件所在的源存储库。提交后，存储库和工作流定义文件将永久链接在一起。在初始提交后，要想更改存储库，唯一方法是在不同的存储库中重新创建工作流。

指定用于存储工作流定义文件的源存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择创建工作流以创建工作流。有关更多信息，请参阅 创建工作流。

   在工作流程创建过程中，您可以指定要 CodeCatalyst存储工作流程定义文件的存储库、分支和文件夹。

指定工作流操作的源存储库

按照以下说明指定用于工作流操作的源存储库。在启动时，操作将配置的源存储库中的文件捆绑到构件中，将该构件下载到运行该操作的运行时环境 Docker 映像，然后使用下载的文件完成其处理。

> **Note**
>
> 目前，在一个工作流操作中，您只能指定一个源存储库，即工作流定义文件所在的源存储库（位于 `.codecatalyst/workflows/` 目录或其子目录中）。此源存储库由标签 `WorkflowSource` 表示。

Visual

指定操作将使用的源存储库（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要为其指定源的操作。
8. 选择输入。
9. 在源 – 可选中，执行以下操作：

   指定表示操作所需的源存储库的标签。当前，支持的唯一标签是 `WorkflowSource`，它表示存储工作流定义文件的源存储库。

   如果省略源，则必须在 *`action-name`*`/Inputs/Artifacts` 下至少指定一个输入构件。

   有关来源的更多信息，请参阅将源存储库连接到工作流。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

指定操作将使用的源存储库（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。

7. 在操作中，添加类似于下文的代码：

```
action-name:
 Inputs:
   Sources:
     - WorkflowSource
```

有关更多信息，请参阅工作流 YAML 定义 中对相应操作的 Sources 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

引用源存储库文件

如果您的文件位于源存储库中，并且需要在工作流操作中引用这些文件，请完成以下过程。

> **Note**
>
> 另请参阅在构件中引用文件。

引用存储在源存储库中的文件

- 在要引用文件的操作中，添加类似于以下内容的代码：

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Configuration:
        Steps:
        - run: cd my-app && cat file1.jar
```

在上面的代码中，操作查看 WorkflowSource 源存储库 my-app 根目录下的目录，以查找并显示 file1.jar 文件。

BranchName'和' CommitId '变量

当您的工作流程运行时， CodeCatalyst 源代码会生成BranchName并设置和CommitId变量。这些变量被称为预定义变量。有关这些变量的信息，请参见下表。

有关在工作流中引用这些变量的信息，请参阅使用预定义变量。

键	值
CommitId	提交 ID 代表了启动工作流运行时存储库的状态。 示例：example3819261db00a3ab59468c8b 另请参阅：示例：引用“CommitId”预定义变量
BranchName	在其上启动工作流运行的分支的名称。 示例：main、feature/branch 、test-LiJuan 。 另请参阅：示例：引用“BranchName”预定义变量

将程序包存储库连接到工作流

软件包是一个包含安装软件和解决任何依赖关系所需的软件和元数据的捆绑包。 CodeCatalyst 支持 npm 包格式。

程序包中包括：

- 一个名称（例如，webpack 是一个流行的 npm 程序包的名称）
- 一个可选的命名空间（例如，@types/node 中的 @types）
- 一组版本（例如，1.0.0、1.0.1、1.0.2）
- 程序包级别的元数据（例如 npm dist 标签）

在中 CodeCatalyst，您可以将包发布到工作流程中的软件包存储库并使用来自软件 CodeCatalyst 包存储库的包。您可以使用 CodeCatalyst包存储库配置构建或测试操作，以自动配置操作的 npm 客户端，使其从指定的存储库中推送和拉取软件包。

有关程序包的更多信息，请参阅在中发布和共享软件包 CodeCatalyst。

> **Note**
>
> 目前，生成和测试操作支持 CodeCatalyst 软件包存储库。

主题

教程：从程序包存储库中拉取

在本教程中，您将了解如何创建工作流，该工作流运行的应用程序的依赖项是从 CodeCatalyst 程序包存储库中拉取的。该应用程序是一个简单的 Node.js 应用程序，它向 CodeCatalyst 日志打印 “Hello World” 消息。该应用程序具有一个依赖项：lodash npm 程序包。`lodash` 程序包用于将 `hello-world` 字符串转换为 `Hello World`。您将使用此程序包的 4.17.20 版本。

设置应用程序和工作流后，您可以将 CodeCatalyst 配置为阻止其他版本的 `lodash` 从公共外部注册表（npmjs.com）导入 CodeCatalyst 程序包存储库中。之后，您可以测试是否已成功阻止其他版本的 `lodash`。

在本教程结束时，您应已清楚地了解工作流如何与 CodeCatalyst 内部和外部的程序包存储库进行交互来检索程序包。您还应了解 npm、程序包存储库、工作流和应用程序的 `package.json` 文件之间进行的后台交互。

主题

先决条件

开始前的准备工作：

- 您需要一个 CodeCatalyst 空间。有关更多信息，请参阅创建空间。
- 在您的 CodeCatalyst 空间中，您需要一个空项目，其名称为：

  ```
  codecatalyst-package-project
  ```

  使用从头开始选项来创建此项目。

  有关更多信息，请参阅 在 Amazon 中创建一个空项目 CodeCatalyst。

步骤 1：创建源存储库

在此步骤中，您将在 CodeCatalyst 中创建源存储库。此存储库将存储教程的源文件，例如 `index.js` 和 `package.json` 文件。

有关源存储库的更多信息，请参阅创建源存储库。

创建源存储库

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的项目 `codecatalyst-package-project`。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择创建存储库。
5. 在存储库名称中，输入：

   ```
   hello-world-app
   ```

6. 选择创建。

步骤 2：创建 CodeCatalyst 和网关程序包存储库

在此步骤中，您将在 CodeCatalyst 项目中创建一个程序包存储库，然后将它连接到该 CodeCatalyst 项目中的网关存储库。稍后，您将本教程中的依赖项 `lodash` 从 npmjs.com 导入到这两个存储库中。

网关存储库是将 CodeCatalyst 中的程序包存储库连接到公共 npmjs.com 的“胶粘剂”。

有关程序包存储库的更多信息，请参阅在中发布和共享软件包 CodeCatalyst。

> **Note**
>
> 在本教程中，CodeCatalyst 程序包存储库和网关存储库这两个词是指您在以下过程中在 CodeCatalyst 中创建的两个存储库。

创建 CodeCatalyst 程序包和网关存储库

1. 在导航窗格中，选择程序包。
2. 选择创建程序包存储库。
3. 在存储库名称中，输入：

   ```
   codecatalyst-package-repository
   ```

4. 选择 + 选择上游存储库。
5. 选择网关存储库。
6. 在 npm-public-registry-gateway 框中，选择创建。
7. 选定选择。
8. 选择创建。

   CodeCatalyst 创建一个名为 `codecatalyst-package-repository` 的软件包存储库，该存储库连接到网关存储库。网关存储库将连接到 npmjs.com 注册表。

步骤 3：创建“Hello World”应用程序

在此步骤中，您创建一个“Hello World”应用程序并将其依赖项（`lodash`）导入网关和 CodeCatalyst 程序包存储库。

要创建该应用程序，您需要一台安装了 Node.js 和相关 npm 客户端的开发机器。

本教程假定您将 CodeCatalyst 开发环境用作开发机器。虽然您不必使用 CodeCatalyst 开发环境，但建议您使用该开发环境，因为它提供了一个干净的工作环境，预安装了 Node.js 和 npm，并且可让您在完成本教程后轻松删除。有关 CodeCatalyst 开发环境的更多信息，请参阅创建开发环境。

按照以下说明操作，启动 CodeCatalyst 开发环境并使用它创建“Hello World”应用程序。

启动 CodeCatalyst 开发环境

1. 在导航窗格中，选择代码，然后选择开发环境。
2. 在顶部附近，选择创建开发环境，然后选择 AWS Cloud9 (在浏览器中)。
3. 确保存储库设置为 `hello-world-app`，并且现有分支设置为 `main`。选择创建。

   您的开发环境将在新的浏览器标签页中启动，并且您的存储库（`hello-world-app`）将克隆到其中。
4. 将 CodeCatalyst 浏览器标签页保持打开状态，然后转到下一个过程。

创建“Hello World”Node.js 应用程序

1. 转到您的开发环境。
2. 在终端提示符下，更改为 `hello-world-app` 源存储库根目录：

   ```
   cd hello-world-app
   ```
3. 初始化一个 Node.js 项目：

   ```
   npm init -y
   ```

   初始化过程将在 `hello-world-app` 的根目录中创建一个 `package.json` 文件。
4. 将开发环境中的 npm 客户端连接到 CodeCatalyst 程序包存储库：

   1. 切换到 CodeCatalyst 控制台。
   2. 在导航窗格中，选择程序包。
   3. 选择 `codecatalyst-package-repository`。
   4. 选择连接到存储库。
   5. 选择创建令牌。这将为您创建个人访问令牌（PAT）。

6. 选择复制以复制命令。
7. 切换到您的开发环境。
8. 确保您位于 hello-world-app 目录中。
9. 粘贴命令。其内容与以下内容类似：

```
npm set registry=https://packages.us-west-2.codecatalyst.aws/npm/ExampleCompany/
codecatalyst-package-project/codecatalyst-package-repository/ --location project
npm set //packages.us-west-2.codecatalyst.aws/npm/ExampleCompany/codecatalyst-
package-project/hello-world-app/:_authToken=username:token-secret
```

5. 导入 lodash 版本 4.17.20：

```
npm install lodash@v4.17.20 --save --save-exact
```

npm 按以下顺序在以下位置查找 lodash 版本 4.17.20：

- 在开发环境中。它在该位置找不到此版本。
- 在 CodeCatalyst 程序包存储库中。它在该位置找不到此版本。
- 在网关存储库中。它在该位置找不到此版本。
- 在 npmjs.com 中。它在该位置找到此版本。

npm 将 lodash 导入网关存储库、CodeCatalyst 程序包存储库和开发环境中。

> **Note**
>
> 如果您在步骤 4 中未将 npm 客户端连接到 CodeCatalyst 程序包存储库，则 npm 应已直接从 npmjs.com 中拉取 lodash，并且不会将程序包导入任一存储库。

npm 还使用 lodash 依赖项更新 package.json 文件，并创建一个包含 lodash 其所有依赖项的 node_modules 目录。

6. 测试是否已将 lodash 成功导入开发环境中。输入：

```
npm list
```

这将显示以下消息，表示已成功导入：

```
`-- lodash@4.17.20
```

7. （可选）打开 hello-world-app/package.json 并验证是否添加了####行：

```
{
  "name": "hello-world-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  dependencies": {
    "lodash": "4.17.20"
  }
}
```

8. 在 /hello-world-app 中，创建一个名为 index.js 的包含以下内容的文件：

> **Tip**
>
> 您可以使用开发环境中的侧面导航来创建此文件。

```
// Importing lodash library
const _ = require('lodash');

// Input string
const inputString = 'hello-world';

// Transforming the string using lodash
const transformedString = _.startCase(inputString.replace('-', ' '));

// Outputting the transformed string to the console
console.log(transformedString);
```

测试“lodash”是否已导入您的网关和 CodeCatalyst 程序包存储库中

1. 切换到 CodeCatalyst 控制台。
2. 在导航窗格中，选择程序包。
3. 选择 npm-public-registry-gateway。
4. 确保已显示 `lodash`。最新版本列指示 `4.17.20`。
5. 对 `codecatalyst-package-repository` 重复此过程。您可能需要刷新浏览器窗口才能看到已导入的程序包。

在开发环境中测试“Hello World”

1. 切换到您的开发环境。
2. 确保您仍在 `hello-world-app` 目录中，然后运行应用程序：

   ```
   node index.js
   ```

   这将显示 `Hello World` 消息。Node.js 使用您在上一步中下载到开发环境的 `lodash` 程序包运行应用程序。

忽略“node_modules”目录并提交“Hello World”

1. 忽略 `node_modules` 目录。输入：

   ```
   echo "node_modules/" >> .gitignore
   ```

   最佳实践是避免提交此目录。此外，提交此目录会干扰本教程中的后续步骤。
2. 添加、提交和推送：

   ```
   git add .
   git commit -m "add the Hello World application"
   git push
   ```

   “Hello World”应用程序和项目文件将添加到源存储库中。

步骤 4：创建运行“Hello World”的工作流

在此步骤中，您将创建一个使用 lodash 依赖项运行“Hello World”应用程序的工作流。该工作流包含一个名为 RunHelloWorldApp 的操作或任务。RunHelloWorldApp 操作包括以下值得注意的命令和部分：

- **Packages**

  此部分指出该操作在运行 npm install 时必须连接到的 CodeCatalyst 程序包存储库的名称。

- **- Run: npm install**

  此命令告知 npm 安装 package.json 文件中指定的依赖项。package.json 文件中指定的唯一依赖项是 lodash。npm 会在以下位置查找 lodash：

  - 在运行该操作的 Docker 映像中。它在该位置找不到此版本。
  - 在 CodeCatalyst 程序包存储库中。它在该位置找到此版本。

  npm 在找到 lodash 后，会将它导入到运行该操作的 Docker 映像中。

- **- Run: npm list**

  此命令会打印出已下载到运行该操作的 Docker 映像的 lodash 的版本。

- **- Run: node index.js**

  此命令使用 package.json 文件中指定的依赖项运行“Hello World”应用程序。

请注意，RunHelloWorldApp 操作是一个构建操作，如工作流顶部附近的 aws/build@v1 标识符所示。有关构建操作的更多信息，请参阅使用工作流进行构建。

按照以下说明操作来创建一个工作流，该工作流从 CodeCatalyst 程序包存储库中拉取 lodash 依赖项，然后运行“Hello World”应用程序。

创建工作流

1. 切换到 CodeCatalyst 控制台。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择创建工作流。
4. 对于源存储库，选择 hello-world-app。
5. 对于分支，选择 main。

这将在选定的源存储库和分支中创建工作流定义文件。

6. 选择创建。
7. 在顶部附近选择 YAML。
8. 删除 YAML 示例代码。
9. 添加以下 YAML 代码：

```
Name: codecatalyst-package-workflow
SchemaVersion: "1.0"

# Required - Define action configurations.
Actions:
  RunHelloWorldApp:
    # Identifies the action. Do not modify this value.
    Identifier: aws/build@v1
    Compute:
      Type: Lambda
    Inputs:
      Sources:
        - WorkflowSource # This specifies your source repository.
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm list
        - Run: node index.js
      Container: # This specifies the Docker image that runs the action.
        Registry: CODECATALYST
        Image: CodeCatalystLinuxLambda_x86_64:2024_03
    Packages:
      NpmConfiguration:
        PackageRegistries:
          - PackagesRepository: codecatalyst-package-repository
```

在前面的代码中，将 *codecatalyst-package-repository* 替换为您在步骤 2：创建 CodeCatalyst 和网关程序包存储库中创建的 CodeCatalyst 程序包存储库的名称。

有关此文件中的属性的信息，请参阅构建和测试操作 YAML。

10. （可选）选择验证，确保 YAML 代码在提交之前有效。
11. 选择提交。
12. 在提交工作流对话框中，输入以下内容：

a. 对于工作流文件名，保留默认值 codecatalyst-package-workflow。
b. 对于提交消息，输入：

```
add initial workflow file
```

c. 对于存储库，选择 hello-world-app。
d. 对于分支名称，选择主。
e. 选择提交。

现在，您已创建工作流。

运行工作流

1. 在您刚刚创建的工作流（codecatalyst-package-workflow）旁边，选择操作，然后选择运行。

   工作流运行将开始。
2. 在右侧顶部的绿色通知中，选择运行的链接。该链接看起来类似于 View Run-1234。

   这将显示一个工作流图表，其中显示谁启动了运行和 RunHelloWorldApp 操作。
3. 选中 RunHelloWorldApp 操作框以查看操作的进度。
4. 运行完成后，转至步骤 5：验证工作流。

步骤 5：验证工作流

在此步骤中，您将验证工作流是否已使用其 lodash 依赖项成功运行“Hello World”应用程序。

验证“Hello World”应用程序是否已使用其依赖项运行

1. 在工作流图表中，选中 RunHelloWorldApp 框。

   这将显示日志消息列表。
2. 展开 node index.js 日志消息。

   这将显示以下消息：

```
[Container] 2024/04/24 21:15:41.545650 Running command node index.js
```

```
Hello World
```

显示 Hello Word（而不是 hello-world）指明已成功使用 lodash 依赖项。

3. 展开 npm list 日志。

   这将显示一条与以下内容类似的消息：

```
### lodash@4.17.20
```

   此消息指示 lodash 版本 4.17.20 已下载到运行工作流操作的 Docker 映像。

步骤 6：阻止来自 npmjs.com 的导入

现在，lodash 版本 4.17.20 已存在于网关和 CodeCatalyst 程序包存储库中，您可以阻止其他版本的导入。阻止可防止您意外导入可能包含恶意代码的更高（或更早）版本的 lodash。有关更多信息，请参阅编辑程序包来源控制和依赖项替换攻击。

按照以下说明操作，阻止将 lodash 导入到网关存储库中。当您在网关上阻止程序包时，它们也会在下游位置被阻止。

阻止导入到网关存储库

1. 在导航窗格中，选择程序包。
2. 选择 npm-publish-registry-gateway。
3. 选择 lodash。
4. 在顶部附近，选择源控制。
5. 在上游下，选择阻止。
6. 选择保存。

   现在，您已阻止从 npmjs.com 导入网关存储库（以及下游存储库和计算机）中。

步骤 7：测试阻止功能

在此部分中，您将验证您在 步骤 6：阻止来自 npmjs.com 的导入 中设置的阻止是否有效。首先，将“Hello World”配置为 lodash 的请求版本 4.17.21，而不是网关存储库中可用的版本，即 4.17.20。然后，检查应用程序是否无法从 nmpjs.com 中拉取版本 4.17.21，这表示已成功阻止。作为最终测试，您需要解除阻止导入到网关存储库，并检查应用程序是否能成功拉取 lodash 的版本 4.17.21。

使用以下一组过程来测试阻止功能。

开始前的准备工作

1. 切换到您的开发环境。
2. 拉取您之前使用 CodeCatalyst 控制台创建的 `codecatalyst-package-workflow.yaml` 文件：

```
git pull
```

将“Hello World”配置为“lodash”的请求版本 4.17.21

1. 打开 `/hello-world-app/package.json`。
2. 将 `lodash` 版本更改为 4.17.21，如####所示：

```
{
  "name": "hello-world-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "lodash": "4.17.21"
  }
}
```

现在，`package.json` 文件中的版本（4.17.21）与网关和 CodeCatalyst 程序包存储库中的版本（4.17.20）不匹配。

3. 添加、提交和推送：

```
git add .
git commit -m "update package.json to use lodash 4.17.21"
git push
```

测试“Hello World”是否无法拉取“lodash”的版本 4.17.21

1. 在版本不匹配的情况下运行工作流：

   1. 切换到 CodeCatalyst 控制台。
   2. 在导航窗格中，选择 CI/CD，然后选择工作流。
   3. 在 `codecatalyst-package-workflow` 旁边，选择操作，然后选择运行。

      npm 会在 `package.json` 中查找依赖项，并看到“Hello World”需要 `lodash` 的版本 4.17.21。npm 按以下顺序在以下位置查找依赖项：

      - 在运行该操作的 Docker 映像中。它在该位置找不到此版本。
      - 在 CodeCatalyst 程序包存储库中。它在该位置找不到此版本。
      - 在网关存储库中。它在该位置找不到此版本。
      - 在 npmjs.com 中。它在该位置找到此版本。

      npm 在 npmjs.com 中找到版本 4.17.21 后，会尝试将它导入网关存储库中，但由于您将网关设置为阻止导入 `lodash`，因此无法执行导入。

      由于无法执行导入，因此工作流将失败。

2. 验证工作流是否已失败：

   1. 在右侧顶部的绿色通知中，选择运行的链接。该链接看起来类似于 `View Run-2345`。
   2. 在工作流图表中，选中 RunHelloWorldApp 框。
   3. 展开 `npm install` 日志消息。

      这将显示以下消息：

      ```
      [Container] 2024/04/25 17:20:34.995591 Running command npm install
      npm ERR! code ETARGET
      npm ERR! notarget No matching version found for lodash@4.17.21.
      npm ERR! notarget In most cases you or one of your dependencies are requesting
      npm ERR! notarget a package version that doesn't exist.

      npm ERR! A complete log of this run can be found in: /tmp/.npm/
      _logs/2024-05-08T22_03_26_493Z-debug-0.log
      ```

      该错误表明找不到版本 4.17.21。这是预期结果，因为您已阻止导入。

解除阻止来自 npmjs.com 的导入

1. 在导航窗格中，选择程序包。
2. 选择 npm-publish-registry-gateway。
3. 选择 lodash。
4. 在顶部附近，选择源控制。
5. 在上游下，选择允许。
6. 选择保存。

   您现在已解除阻止导入 lodash。

   您的工作流现在可以导入 lodash 的版本 4.17.21。

测试是否已解除阻止来自 npmjs.com 的导入

1. 再次运行工作流。这次工作流应成功，因为现在应已能够导入版本 4.17.21。要再次运行工作流，请执行以下操作：

   1. 选择 CI/CD，然后选择工作流。
   2. 在 codecatalyst-package-workflow 旁边，选择操作，然后选择运行。
   3. 在右侧顶部的绿色通知中，选择运行的链接。该链接看起来类似于 View Run-3456。

      这将显示一个工作流图表，其中显示谁启动了运行和 RunHelloWorldApp 操作。
   4. 选中 RunHelloWorldApp 操作框以查看操作的进度。
   5. 展开 npm list 日志消息，验证是否显示与以下内容类似的消息：

      ```
      ### lodash@4.17.21
      ```

      此消息指示已下载 lodash 版本 4.17.21。
2. 验证版本 4.17.21 是否已导入 CodeCatalyst 和网关存储库中：

   1. 在导航窗格中，选择程序包。
   2. 选择 npm-public-registry-gateway。
   3. 查找 lodash 并确保版本为 4.17.21。

> **Note**
>
> 虽然此页面上未列出版本 4.17.20，但您可以通过选择 `lodash`，然后选择顶部附近的版本来找到该版本。

4. 重复这些步骤以检查是否已将版本 4.17.21 导入 `codecatalyst-package-repository`。

清理

清理本教程中使用的文件和服务以免被收取费用。

清理程序包教程

1. 删除 `codecatalyst-package-project`：
   a. 在 CodeCatalyst 控制台中，导航到 `codecatalyst-package-project` 项目（如果您尚未这样做）。
   b. 在导航窗格中，选择项目设置。
   c. 选择删除项目，输入 **`delete`**，然后选择删除项目。

      CodeCatalyst 会删除所有项目资源，包括源、网关和 CodeCatalyst 程序包存储库。也将删除开发环境。
2. 删除 PAT 令牌：
   a. 在右侧选择您的用户名，然后选择我的设置。
   b. 在个人访问令牌下，选择您在本教程中创建的令牌，然后选择删除。

在本教程中，您已了解如何创建工作流，该工作流运行的应用程序将从 CodeCatalyst 程序包存储库中拉取其依赖项。您还了解如何阻止和解除阻止程序包进入网关和 CodeCatalyst 程序包存储库。

在工作流程中指定 CodeCatalyst 软件包存储库

在中 CodeCatalyst，您可以将 CodeCatalyst 包存储库添加到工作流程中的生成和测试操作中。您的程序包存储库必须配置为程序包格式，例如 npm。您也可以为选定的程序包存储库包括一系列范围。

按照以下说明指定要用于工作流操作的程序包配置。

Visual

指定操作将使用的程序包配置（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，对要配置的程序包存储库选择构建或测试操作。
8. 选择程序包。
9. 从添加配置下拉菜单中，选择要用于工作流操作的程序包配置。
10. 选择添加程序包存储库。
11. 在 P ackage 存储库下拉菜单中，指定您希望该操作使用的 CodeCatalyst 包存储库的名称。

    有关程序包存储库的更多信息，请参阅程序包存储库。
12. （可选）在范围 – 可选中，指定您要在程序包注册表中定义的范围序列。

    定义范围时，将指定的程序包存储库配置为所有列出范围的注册表。如果通过 npm 客户端请求具有范围的程序包，此程序包将使用该存储库而不是默认存储库。每个范围名称必须以“@”为前缀。

    如果省略 `Scopes`，则将指定的程序包存储库配置为该操作使用的所有程序包的默认注册表。

    有关范围的更多信息，请参阅程序包命名空间和范围限定的程序包。
13. 选择添加。
14. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
15. 选择提交，输入提交消息，然后再次选择提交。

YAML

指定操作将使用的程序包配置（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在构建或测试操作中，添加类似于以下内容的代码：

```
action-name:
 Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: package-repository
              Scopes:
                - "@scope"
```

有关更多信息，请参阅构建和测试操作 YAML 中对相应操作的 Packages 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

在工作流操作中使用授权令牌

您可以使用工作流操作提供的令牌手动配置包管理器以使用 CodeCatalyst 包存储库进行身份验证。CodeCatalyst 使此令牌可用作环境变量，供您在操作中引用。

环境变量	值
CATALYST_MACHINE_RESOURCE_NAME	授权令牌的用户身份。
CATALYST_PACKAGES_AUTHORIZATION_TOKEN	授权令牌的值。

> **Note**
>
> 请注意，只有在您将操作配置为导出授权令牌后，才会填充这些环境变量。

按照以下说明在工作流操作中使用授权令牌。

Visual

将导出的授权令牌与操作配合使用（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，对要配置的程序包存储库选择构建或测试操作。
8. 选择程序包。
9. 开启导出授权令牌。

YAML

将导出的授权令牌与操作配合使用（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在构建或测试操作中，添加类似于以下内容的代码：

```
Actions:
  action-name:
    Packages:
      ExportAuthorizationToken: true
```

您可以在 YAML 的 Steps 部分中引用 $CATALYST_MACHINE_RESOURCE_NAME 和 $CATALYST_PACKAGES_AUTHORIZATION_TOKEN 环境变量。有关更多信息，请参阅示例：手动配置pip为使用进行身份验证 CodeCatalyst。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

示例：工作流中的程序包存储库

以下示例演示了如何在工作流定义文件中引用程序包。

主题

- 示例：使用 NpmConfiguration 定义程序包
- 示例：覆盖默认注册表
- 示例：覆盖程序包注册表中的范围
- 示例：手动配置pip为使用进行身份验证 CodeCatalyst

示例：使用 **NpmConfiguration** 定义程序包

以下示例展示了如何使用 NpmConfiguration 在工作流定义文件中定义程序包。

```
Actions:
  Build:
  Identifier: aws/build-beta@v1
  Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: main-repo
            - PackagesRepository: scoped-repo
              Scopes:
                - "@scope1"
```

此示例如下所示配置 npm 客户端：

```
default: main-repo
@scope1: scoped-repo
```

在此示例中，定义了两个存储库。默认注册表设置为 main-repo，在定义时没有范围。范围 @scope1 在 PackageRegistries 中为 scoped-repo 配置。

示例：覆盖默认注册表

以下示例演示了如何覆盖默认注册表。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-repo-1
    - PackagesRepository: my-repo-2
    - PackagesRepository: my-repo-3
```

此示例如下所示配置 npm 客户端：

```
default: my-repo-3
```

如果您指定多个默认存储库，则优先使用最后一个存储库。在此示例中，列出的最后一个存储库是 my-repo-3，这意味着 npm 将连接到 my-repo-3。这会覆盖存储库 my-repo-1 和 my-repo-2。

示例：覆盖程序包注册表中的范围

以下示例展示了如何覆盖程序包注册表中的范围。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-default-repo
    - PackagesRepository: my-repo-1
      Scopes:
        - "@scope1"
        - "@scope2"
    - PackagesRepository: my-repo-2
      Scopes:
        - "@scope2"
```

此示例如下所示配置 npm 客户端：

```
default: my-default-repo
@scope1: my-repo-1
@scope2: my-repo-2
```

如果您包含覆盖范围，则优先使用最后一个存储库。在本示例中，在 PackageRegistries 中最后一次配置范围 @scope2 是为 my-repo-2 配置的。这会覆盖为 my-repo-1 配置的范围 @scope2。

示例：手动配置**pip**为使用进行身份验证 CodeCatalyst

以下示例向您展示了如何在生成操作中引用 CodeCatalyst 授权环境变量。

```
Actions:
  Build:
    Identifier: aws/build@v1.0.0
    Configuration:
      Steps:
        - Run: pip config set global.index-url https://$CATALYST_MACHINE_RESOURCE_NAME:
$CATALYST_PACKAGES_AUTHORIZATION_TOKEN@codecatalyst.aws/pypi/my-space/my-project/my-
repo/simple/
    Packages:
      ExportAuthorizationToken: true
```

使用工作流调用 Lambda 函数

本节介绍如何使用 CodeCatalyst 工作流程调用 AWS Lambda 函数。为此，您必须将 AWS Lambda 调用操作添加到工作流中。AWS Lambda 调用操作会调用您指定的 Lambda 函数。

除了调用您的函数之外，AWS Lambda 调用操作还会将从 Lambda 函数收到的响应有效载荷中的每个顶级密钥转换为工作流输出变量。之后，可以在后续工作流操作中引用这些变量。如果您不想将所有顶级密钥都转换为变量，则可以使用筛选条件来指定确切的密钥。有关更多信息，请参阅“AWS Lambda 调用”操作 YAML 中的 ResponseFilters 属性描述。

主题

- 何时使用此操作
- “AWS Lambda 调用” 操作使用的运行时镜像
- 示例：调用 Lambda 函数
- 添加 “AWS Lambda 调用” 操作

- “AWS Lambda 调用”变量
- “AWS Lambda 调用”操作 YAML

何时使用此操作

如果您想向工作流中添加封装在 Lambda 函数中并由 Lambda 函数执行的功能，请使用此操作。

例如，您可能希望工作流在开始构建应用程序之前向 Slack 频道发送 `Build started` 通知。在此情况下，您的工作流将包括一个 AWS Lambda 调用操作（用于调用 Lambda 以发出 Slack 通知）和一个构建操作（用于构建应用程序）。

再举一个例子，您可能希望工作流在部署应用程序之前对其进行漏洞扫描。在此情况下，您将使用构建操作来构建应用程序，使用 AWS Lambda 调用操作来调用 Lambda 以扫描漏洞，并使用部署操作来部署已扫描的应用程序。

“AWS Lambda 调用” 操作使用的运行时镜像

AWS Lambda 调用操作在 2022 年 11 月版映像上运行。有关更多信息，请参阅 活动映像。

示例：调用 Lambda 函数

以下示例工作流包括 AWS Lambda 调用操作以及部署操作。该工作流程会发出 Slack 通知，表明部署已开始，然后 AWS 使用模板部署应用程序。 CloudFormation 工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- AWS Lambda 调用操作（`LambdaNotify`）– 此操作在触发后将调用指定的 AWS 账户和区域（`my-aws-account` 和 `us-west-2`）中的 `Notify-Start` Lambda 函数。此 Lambda 函数一经调用，就会发送一条 Slack 通知，表明部署已开始。
- 部署 CloudFormation 堆栈操作 (`Deploy`)-AWS Lambda 调用操作完成后，部署 CloudFormation 堆栈操作将运行模板 (`cfn-template.yml`) 来部署您的应用程序堆栈。有关 “部署 CloudFormation 堆栈” 操作的更多信息，请参阅部署 CloudFormation 堆栈。

> **Note**
>
> 以下工作流示例仅用于说明目的，如果不执行附加配置，则无法运行。

> **Note**
>
> 在接下来的 YAML 代码中，如果需要，可以省略 Connections: 部分。如果您省略这些部分，则必须确保在您的环境中 Defa ult IAM 角色字段中指定的角色包含AWS Lambda 调用和部署 CloudFormation 堆栈操作所需的权限和信任策略。有关使用默认 IAM 角色设置环境的更多信息，请参阅创建环境。有关AWS Lambda 调用和部署 CloudFormation 堆栈操作所需的权限和信任策略的更多信息，请参阅“AWS Lambda 调用”操作 YAML和中对Role属性的描述'部署 CloudFormation 堆栈'动作 YAML。

```
Name: codecatalyst-lamda-invoke-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  LambdaNotify:
    Identifier: aws/lambda-invoke@v1
    Environment:
      Name: my-production-environment
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-lambda-invoke-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Function: Notify-Start
      AWSRegion: us-west-2

  Deploy:
    Identifier: aws/cfn-deploy@v1
    Environment:
      Name: my-production-environment
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-deploy-role
    Inputs:
      Sources:
```

```
      - WorkflowSource
  Configuration:
    name: my-application-stack
    region: us-west-2
    role-arn: arn:aws:iam::111122223333:role/StackRole
    template: ./cfn-template.yml
    capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

添加 “AWS Lambda 调用” 操作

按照以下说明，将 AWS Lambda 调用操作添加到您的工作流中。

先决条件

在开始之前，请确保您的 AWS Lambda 函数和关联的 Lambda 执行角色已准备就绪，并且可以在中使用。 AWS有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 执行角色主题。

Visual

使用可视化编辑器添加AWS Lambda “调用” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS Lambda 调用操作，然后执行以下操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 AWS Lambda 调用。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 在输入、配置和输出选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“AWS Lambda 调用”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添AWS Lambda 加 “调用” 操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 AWS Lambda 调用操作，然后执行以下操作之一：
   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择 AWS Lambda 调用。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 根据需求修改 YAML 代码中的属性。“AWS Lambda 调用”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

“AWS Lambda 调用”变量

默认情况下，AWS Lambda 调用操作会在 Lambda 响应有效载荷中为每个顶级密钥生成一个变量。

例如，如果响应有效载荷与以下内容类似：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

...则该操作会生成以下变量。

键	值
name	Saanvi
地点	Seattle
department	{"company": "Amazon", "team": "AWS"}

> **Note**
>
> 您可以使用 `ResponseFilters` YAML 属性更改生成的变量。有关更多信息，请参阅 “AWS Lambda 调用”操作 YAML 中的 ResponseFilters。

运行时由 “in AWS Lambda voke” 操作生成和设置的变量称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅使用预定义变量。

“AWS Lambda 调用”操作 YAML

下面是 AWS Lambda 调用操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流调用 Lambda 函数。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```yaml
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  LambdaInvoke_nn:
    Identifier: aws/lambda-invoke@v1
    DependsOn:
      - dependent-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - request-payload
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
```

```
      Function: my-function|function-arn
      AWSRegion: us-west-2
      # Specify RequestPayload or RequestPayloadFile, but not both.
      RequestPayload: '{"firstname": "Li", lastname: "Jean", "company": "ExampleCo",
"team": "Development"}'
      RequestPayloadFile: my/request-payload.json
      ContinueOnError: true|false
      LogType: Tail|None
      ResponseFilters: '{"name": ".name", "company": ".department.company"}'
    Outputs:
      Artifacts:
        - Name: lambda_artifacts
          Files:
            - "lambda-response.json"
```

LambdaInvoke

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：Lambda_Invoke_Action_Workflow_nn。

对应的 UI：“配置”选项卡/操作名称

Identifier

(*LambdaInvoke*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/lambda-invoke@v1。

对应的 UI：工作流图表/LambdaInvoke_nn/aws/lambda-invoke@v1 标签

DependsOn

(*LambdaInvoke*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*LambdaInvoke*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*LambdaInvoke*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

对应的 UI：“配置”选项卡/计算类型

Fleet

(*LambdaInvoke*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

对应的 UI：“配置”选项卡/计算实例集

Timeout

(*LambdaInvoke*/Timeout)

（必需）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Inputs

(*LambdaInvoke*/Inputs)

（必需）

`Inputs` 部分中定义了工作流运行期间 AWS Lambda 调用操作所需的数据。

> **Note**
>
> 每个 AWS Lambda 调用操作只能有一个输入（可以是源或构件）。变量不计入此总数。

对应的 UI：输入选项卡

Sources

(*LambdaInvoke*/Inputs/Sources)

（如果RequestPayloadFile已提供，则为必填项）

如果要将一个请求有效载荷 JSON 文件传递到 AWS Lambda 调用操作，并且此有效载荷文件存储在一个源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 `WorkflowSource`。

如果您的请求有效载荷文件不包含在源存储库中，则必须位于另一个操作生成的构件中。

有关有效载荷文件的更多信息，请参阅 RequestPayloadFile。

> **Note**
>
> 您可以使用 `RequestPayload` 属性直接将有效载荷的 JSON 代码添加到操作中，而不必指定有效载荷文件。有关更多信息，请参阅 RequestPayload。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*LambdaInvoke*/Inputs/Artifacts)

（如果RequestPayloadFile已提供，则为必填项）

如果要将请求有效载荷 JSON 文件传递到 AWS Lambda 调用操作，并且此有效载荷文件包含在上一操作的输出构件中，请在此处指定该构件。

有关有效载荷文件的更多信息，请参阅 RequestPayloadFile。

> **Note**
>
> 您可以使用 `RequestPayload` 属性直接将有效载荷的 JSON 代码添加到操作中，而不必指定有效载荷文件。有关更多信息，请参阅 RequestPayload。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Variables - input

(*LambdaInvoke*/Inputs/Variables)

（可选）

指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输入”选项卡/变量 – 可选

Environment

(*LambdaInvoke*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。该操作连接到在所选环境中指定的 AWS 账户 和可选的 Amazon VPC。该操作使用环境中指定的默认 IAM 角色连接到 AWS 账户，并使用在 A mazon VPC 连接中指定的 IAM 角色连接到亚马逊 VPC。

> **Note**
>
> 如果默认 IAM 角色不具有操作所需的权限，则可以将操作配置为使用其他角色。有关更多信息，请参阅更改操作的 IAM 角色。

有关环境的更多信息，请参阅部署到 AWS 账户 和 VPCs和创建环境。

对应的 UI：“配置”选项卡/环境

Name

(*LambdaInvoke*/Environment/Name)

（如果包含 Environment，则为必需）

指定要与操作关联的现有环境的名称。

对应的 UI：“配置”选项卡/环境

Connections

(*LambdaInvoke*/Environment/Connections)

（在新版本的操作中为可选；在旧版本中为必需）

指定要与操作关联的账户连接。您在 `Environment` 下最多只能指定一个账户连接。

如果您不指定账户连接：

- 该操作使用 CodeCatalyst 控制台环境中指定的 AWS 账户 连接和默认 IAM 角色。有关向环境添加账户连接和默认 IAM 角色的信息，请参阅创建环境。
- 默认 IAM 角色必须包含操作所需的策略和权限。要具体确定这些策略和权限，请参阅操作的 YAML 定义文档中 Role 属性的描述。

有关账户连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关向环境添加账户连接的信息，请参阅创建环境。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Name

(*LambdaInvoke*/Environment/Connections/Name)

（如果包含 Connections，则为必需）

指定账户连接的名称。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*my-environment*吗？ /三点菜单/ 切换角色
- （旧版本）配置选项卡/ Environment/account/role "/账户连接AWS

Role

(*LambdaInvoke*/Environment/Connections/Role)

（如果包含 Connections，则为必需）

指定AWS Lambda 调用操作用于访问 AWS 和调用您的 Lambda 函数的 IAM 角色的名称。请确保您已将该角色添加到您的 CodeCatalyst 空间，并且该角色包含以下策略。

如果您未指定 IAM 角色，则该操作将使用 CodeCatalyst 控制台中环境中列出的默认 IAM 角色。如果您使用此环境中的默认角色，请确保该角色具有以下策略。

- 以下权限策略：

> **Warning**
>
> 将权限限制在以下策略所示的范围内。使用具有更广泛权限的角色可能会带来安全风险。

- 以下自定义信任策略：

> **Note**
>
> 如果需要，可以在此操作中使用 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色。有关该角色的更多信息，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***`spaceName`*** 角色。了解 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色具有完全访问权限可能会带来安全风险。我们建议您仅在教程和安全要求较低的场景中使用此角色。

对应的 UI：根据操作版本的不同，为下列项之一：

- （新版本）配置tab/Environment/What在*`my-environment`*吗？ /三点菜单/ 切换角色
- （旧版本）“配置” 选项卡/' '/ 角色 Environment/account/role

Configuration

(*`LambdaInvoke`*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

Function

(*`LambdaInvoke`*/Configuration/Function)

（必需）

指定此操作将调用的 AWS Lambda 函数。您可以指定函数的名称或其 Amazon 资源名称（ARN）。您可以在 Lambda 控制台中查找名称或 ARN。

> **Note**
>
> Lambda 函数所在的 AWS 账户可以与下指定的账户不同。`Connections:`

对应的 UI：“配置”选项卡/函数

AWSRegion

(*LambdaInvoke*/Configuration/AWSRegion)

（必需）

指定您的 AWS Lambda 函数所在的 AWS 区域。有关区域代码的列表，请参阅《AWS 一般参考》中的 Regional endpoints。

对应的 UI：“配置”选项卡/目标存储桶 – 可选

RequestPayload

(*LambdaInvoke*/Configuration/RequestPayload)

（可选）

如果要将请求有效载荷传递到 AWS Lambda 调用操作，请在此处以 JSON 格式指定该请求有效载荷。

示例请求有效载荷：

```
'{ "key": "value" }'
```

如果您不想将请求有效载荷传递到 Lambda 函数，请忽略此属性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同时指定两者。

有关请求有效载荷的更多信息，请参阅《AWS Lambda API 参考》中的 Invoke 主题。

对应的 UI：“配置”选项卡/请求有效载荷 – 可选

RequestPayloadFile

(*LambdaInvoke*/Configuration/RequestPayloadFile)

（可选）

如果要将请求有效载荷传递到 AWS Lambda 调用操作，请在此处指定该请求有效载荷文件的路径。该文件必须采用 JSON 格式。

请求有效载荷文件可以驻留在源存储库中，也可以驻留在上一操作的构件中。文件路径相对于源存储库或构件根目录。

如果您不想将请求有效载荷传递到 Lambda 函数，请忽略此属性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同时指定两者。

有关请求有效载荷文件的更多信息，请参阅《AWS Lambda API 参考》中的 Invoke 主题。

对应的 UI：“配置”选项卡/请求有效载荷文件 – 可选

ContinueOnError

(*LambdaInvoke*/Configuration/RequestPayloadFile)

（可选）

指定即使调用的 AWS Lambda 函数失败，是否也将 AWS Lambda 调用操作标记为成功。考虑将此属性设置为 `true` 以允许在 Lambda 失败的情况下启动工作流中的后续操作。

默认情况下，如果 Lambda 函数失败（在可视化编辑器中为“off”，在 YAML 编辑器中为 `false`），则操作将失败。

对应的 UI：“配置”选项卡/出错时继续

LogType

(*LambdaInvoke*/Configuration/LogType)

（可选）

指定是否要在 Lambda 函数被调用后提供的响应中包含错误日志。您可以在控制台的 Lambda 调用操作的“日志”选项卡中查看这些日志。 CodeCatalyst 可能的值有：

- `Tail` – 返回日志
- `None` – 不返回日志

默认值为 Tail。

有关日志类型的更多信息，请参阅《AWS Lambda API 参考》中的 Invoke 主题。

有关查看日志的更多信息，请参阅查看工作流运行状态和详细信息。

对应的 UI：“配置”选项卡/日志类型

ResponseFilters

(*LambdaInvoke*/Configuration/ResponseFilters)

（可选）

指定 Lambda 响应有效载荷中要转换为输出变量的密钥。然后，您可以在工作流的后续操作中引用输出变量。有关中变量的更多信息 CodeCatalyst，请参阅在工作流中使用变量。

例如，如果您的响应有效载荷与以下内容类似：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

...并且您的响应筛选条件与以下内容类似：

```
Configuration:
  ...
  ResponseFilters: '{"name": ".name", "company": ".department.company"}'
```

... 则该操作会生成以下流输出变量：

键	值
name	Saanvi
company	Amazon

然后，您可以在后续操作中引用 name 和 company 变量。

如果您未在 ResponseFilters 中指定任何密钥，则该操作会将 Lambda 响应中的每个顶级密钥转换为输出变量。有关更多信息，请参阅 “AWS Lambda 调用”变量。

考虑使用响应筛选条件将生成的输出变量限定为仅实际要使用的变量。

对应的 UI：“配置”选项卡/响应筛选条件 – 可选

Outputs

(*LambdaInvoke*/Outputs)

（可选）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts

(*LambdaInvoke*/Outputs/Artifacts)

（可选）

指定操作生成的构件。您可以在其他操作中将这些构件作为输入来引用。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件/构建构件名称

Name

(*LambdaInvoke*/Outputs/Artifacts/Name)

（可选）

指定将包含由 Lambda 函数返回的 Lambda 响应有效载荷的构件的名称。默认值为 `lambda_artifacts`。如果您未指定项目，则可以在操作的日志中查看 Lambda 响应有效负载，这些日志位于控制台中操作的“日志”选项卡上。 CodeCatalyst 有关查看日志的更多信息，请参阅查看工作流运行状态和详细信息。

对应的 UI：“输出”选项卡/构件/构建构件名称

Files

(*LambdaInvoke*/Outputs/Artifacts/Files)

（可选）

指定要包含在构件中的文件。您必须指定 `lambda-response.json` 才能包含 Lambda 响应有效载荷文件。

对应的 UI：“输出”选项卡/构件/构建生成的文件

修改 Amazon ECS 任务定义

本节介绍如何使用 CodeCatalyst 工作流程更新亚马逊弹性容器服务 (Amazon ECS) 任务定义文件中的`image`字段。为此，您必须将渲染 Amazon ECS 任务定义操作添加到工作流。此操作使用工作流在运行时提供的 Docker 映像名称更新任务定义文件中的映像字段。

> **Note**
>
> 您还可以通过此操作使用环境变量更新任务定义的 `environment` 字段。

主题

- “渲染 Amazon ECS 任务定义”变量
- “渲染 Amazon ECS 任务定义”操作 YAML

何时使用此操作

如果您的工作流使用动态内容（例如提交 ID 或时间戳）来构建和标记 Docker 映像，请使用此操作。

如果您的任务定义文件包含始终保持不变的映像值，请不要使用此操作。在这种情况下，您可以手动将映像名称输入任务定义文件中。

“渲染 Amazon ECS 任务定义”操作的工作原理

您必须在工作流中将渲染 Amazon ECS 任务定义操作与构建和部署到 Amazon ECS 操作结合使用。通过结合使用这些操作可实现以下目的：

1. 构建操作会构建 Docker 映像，并使用名称、提交 ID、时间戳或其他动态内容对其进行标记。例如，您的构建操作可能与以下内容类似：

```
MyECSWorkflow
  Actions:
    BuildAction:
      Identifier: aws/build@v1
      ...
      Configuration:
        Steps:
        # Build, tag, and push the Docker image...
          - Run: docker build -t MyDockerImage:${WorkflowSource.CommitId} .
          ...
```

   在前面的代码中，`docker build -t` 指令指示构建 Docker 映像，并在操作运行时使用提交 ID 对其进行标记。生成的映像名称可能与以下内容类似：

   `MyDockerImage:a37bd7e`

2. 渲染 Amazon ECS 任务定义操作会将动态生成的映像名称 `MyDockerImage:a37bd7e` 添加到您的任务定义文件中，如下所示：

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
```

```
        {
            "name": "codecatalyst-ecs-container",
            "image":  MyDockerImage:a37bd7e,
            "essential": true,
            ...
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
...
}
```

（可选）您也可以让渲染 Amazon ECS 任务定义操作将环境变量添加到任务定义中，如下所示：

```
{
  "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-
role",
  "containerDefinitions": [
    {
      "name": "codecatalyst-ecs-container",
      "image":  MyDockerImage:a37bd7e,
      ...
      "environment": [
        {
          name": "ECS_LOGLEVEL",
          value": "info"
        }
      ]
    }
  ],
...
}
```

有关环境变量的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的指定环境变量。

3. 部署到 Amazon ECS 操作会将更新后的任务定义文件注册到 Amazon ECS。注册更新后的任务定义文件会将新映像 `MyDockerImage:a37bd7e` 部署到 Amazon ECS 中。

“渲染 Amazon ECS 任务定义”操作使用的运行时映像

渲染 Amazon ECS 任务定义操作在 2022 年 11 月版映像上运行。有关更多信息，请参阅 活动映像。

示例：修改 Amazon ECS taskdef

以下是包含渲染 Amazon ECS 任务定义操作以及构建和部署操作的完整工作流示例。该工作流旨在构建 Docker 映像并将其部署到 Amazon ECS 集群中。工作流包含以下按顺序运行的构造块：

- 触发器 – 当您将更改推送到源存储库时，此触发器会自动启动工作流运行。有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。
- 构建操作（`BuildDocker`）– 此操作在触发后会使用 Dockerfile 构建 Docker 映像，使用提交 ID 标记该映像并将其推送到 Amazon ECR。有关构建操作的更多信息，请参阅使用工作流进行构建。
- 渲染 Amazon ECS 任务定义操作（`RenderTaskDef`）– 构建操作完成后，此操作将使用包含正确提交 ID 的 `image` 字段值更新位于源存储库根目录中的现有 `taskdef.json`。它使用新的文件名（`task-definition-`*`random-string`*`.json`）保存更新后的文件，然后创建包含此文件的输出构件。渲染操作还会生成一个名为 `task-definition` 的变量，并将该变量设置为新任务定义文件的名称。构件和变量将用于接下来的部署操作。
- 部署到 Amazon ECS 操作（`DeployToECS`）– 渲染 Amazon ECS 任务定义操作完成后，部署到 Amazon ECS 操作将查找渲染操作（`TaskDefArtifact`）所生成的输出构件，查找其中包含的 `task-definition-`*`random-string`*`.json` 文件，并将该文件注册到 Amazon ECS 服务。之后，Amazon ECS 服务将按照 `task-definition-`*`random-string`*`.json` 文件中的说明操作，在 Amazon ECS 集群中运行 Amazon ECS 任务以及关联的 Docker 映像容器。

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildDocker:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
```

```
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-east-2 | docker login --username
 AWS --password-stdin 111122223333.dkr.ecr.us-east-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG

  RenderTaskDef:
    DependsOn:
      - BuildDocker
    Identifier: aws/ecs-render-task-definition@v1
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      task-definition: taskdef.json
      container-definition-name: codecatalyst-ecs-container
      image: $REPOSITORY_URI:$IMAGE_TAG
    # The output artifact contains the updated task definition file.
    # The new file is prefixed with 'task-definition'.
    # The output variable is set to the name of the updated task definition file.
    Outputs:
```

```
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - "task-definition*"
      Variables:
        - task-definition

  DeployToECS:
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-deploy-role
    #Input artifact contains the updated task definition file.
    Inputs:
      Sources: []
      Artifacts:
        - TaskDefArtifact
    Configuration:
      region: us-east-2
      cluster: codecatalyst-ecs-cluster
      service: codecatalyst-ecs-service
      task-definition: ${RenderTaskDef.task-definition}
```

添加“渲染 Amazon ECS 任务定义”操作

按照以下说明操作，将渲染 Amazon ECS 任务定义操作添加到工作流中。

先决条件

开始之前，请确保您的工作流包含一个动态生成 Docker 映像的构建操作。有关详细信息，请参阅前面的示例工作流。

Visual

使用可视化编辑器添加“渲染 Amazon ECS 任务定义”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。

4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索渲染 Amazon ECS 任务定义操作，然后执行下列操作之一：

   - 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

     或

   - 选择渲染 Amazon ECS 任务定义。此时会显示操作详细信息对话框。在此对话框中：
     - （可选）选择查看源以查看操作的源代码。
     - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。
10. 在输入和配置选项卡中，根据需要填写字段。有关每个字段的描述，请参阅“渲染 Amazon ECS 任务定义”操作 YAML。本参考提供了有关在 YAML 编辑器和可视化编辑器中显示的每个字段（以及对应的 YAML 属性值）的详细信息。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

YAML

使用 YAML 编辑器添加“渲染 Amazon ECS 任务定义”操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索渲染 Amazon ECS 任务定义操作，然后执行下列操作之一：

- 选择加号（+），将操作添加到工作流图表中并打开其配置窗格。

  或

- 选择渲染 Amazon ECS 任务定义。此时会显示操作详细信息对话框。在此对话框中：
  - （可选）选择查看源以查看操作的源代码。
  - 选择添加到工作流，将操作添加到工作流图表中并打开其配置窗格。

10. 根据需求修改 YAML 代码中的属性。“渲染 Amazon ECS 任务定义”操作 YAML中提供了每个可用属性的解释。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。

后续步骤

在添加渲染操作后，按照使用工作流部署到 Amazon ECS 中的说明操作，将部署到 Amazon ECS 操作添加到工作流中。添加部署操作时，请执行以下操作：

1. 在部署操作的输入选项卡中，在构件 – 可选中，选择由渲染操作生成的构件。它包含更新后的任务定义文件。

   有关构件的更多信息，请参阅在操作之间共享构件和文件。

2. 在部署操作的“配置”选项卡的“任务定义”字段中，指定以下操作变量：${*action-name*.task-definition}其中*action-name*是渲染操作的名称，例如RenderTaskDef。渲染操作将此变量设置为任务定义文件的新名称。

   有关变量的更多信息，请参阅在工作流中使用变量。

   有关如何配置部署操作的更多信息，请参阅前面的示例工作流。

查看更新后的任务定义文件

您可以查看更新后的任务定义文件的名称和内容。

要查看更新后的任务定义文件的名称，请在渲染 Amazon ECS 任务定义操作处理完该文件后进行查看。

1. 查找包含已完成的渲染操作的运行：

a. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
b. 选择您的项目。
c. 在导航窗格中，选择 CI/CD，然后选择工作流。
d. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
e. 选择包含已完成的渲染操作的运行。

2. 在工作流图表中，选择渲染操作。
3. 选择输出。
4. 选择变量。
5. 这将显示任务定义文件的名称。该名称类似于 `task-definition--259-0a2r7gxlTF5X-.json`。

查看更新后的任务定义文件的内容

1. 查找包含已完成的渲染操作的运行：
   a. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
   b. 选择您的项目。
   c. 在导航窗格中，选择 CI/CD，然后选择工作流。
   d. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
   e. 选择包含已完成的渲染操作的运行。
2. 在工作流运行中，在顶部的可视化和 YAML 旁边，选择工作流输出。
3. 在构件部分中，选择包含更新后的任务定义文件的构件旁边的下载。此构件的生成者列将设置为渲染操作的名称。
4. 打开 .zip 文件以查看任务定义 .json 文件。

“渲染 Amazon ECS 任务定义”变量

渲染 Amazon ECS 任务定义操作在运行时会生成并设置以下变量。这些变量被称为预定义变量。

有关在工作流中引用这些变量的信息，请参阅 使用预定义变量。

键	值
task-definition	为已由渲染 Amazon ECS 任务定义操作更新的任务定义文件指定的名称。此名称遵循以下格式：`task-definition- random-string .json`。 示例：`task-definition--259-0a2r7gxlTF5Xr.json`

“渲染 Amazon ECS 任务定义”操作 YAML

下面是渲染 Amazon ECS 任务定义操作的 YAML 定义。要了解如何使用此操作，请参阅修改 Amazon ECS 任务定义。

此操作定义部分包含在更广泛的工作流定义文件中。有关此文件的更多信息，请参阅工作流 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSRenderTaskDefinition_nn:
    Identifier: aws/ecs-render-task-definition@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
```

```
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - task-definition-artifact
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Configuration
      task-definition: task-definition-path
      container-definition-name: container-definition-name
      image: docker-image-name
      environment-variables:
        - variable-name-1=variable-value-1
        - variable-name-2=variable-value-2
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files: "task-definition*"
      Variables:
        - task-definition
```

ECSRenderTaskDefinition

（必需）

指定操作的名称。工作流中的所有操作名称都必须是唯一的。操作名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在操作名称中包含特殊字符和空格。

默认值：ECSRenderTaskDefinition_nn。

对应的 UI：“配置”选项卡/操作名称

Identifier

(*ECSRenderTaskDefinition*/Identifier)

（必需）

标识操作。除非您要更改版本，否则不要更改此属性。有关更多信息，请参阅指定要使用的操作版本。

默认值：aws/ecs-render-task-definition@v1。

对应的用户界面：工作流程图/ ECSRenderTaskDefinition _nn/ aws/ @v1 标签 ecs-render-task-definition

DependsOn

(*ECSRenderTaskDefinition*/DependsOn)

（可选）

指定必须成功运行才能使该操作运行的操作、操作组或阶段门。

有关“依赖于”功能的更多信息，请参阅顺序操作。

对应的 UI：“输入”选项卡/依赖于 – 可选

Compute

(*ECSRenderTaskDefinition*/Compute)

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅跨操作共享计算。

对应的 UI：无

Type

(*ECSRenderTaskDefinition*/Compute/Type)

（如果包含 Compute，则为必需）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

对应的 UI：“配置”选项卡/计算类型

Fleet

(*ECSRenderTaskDefinition*/Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

对应的 UI：“配置”选项卡/计算实例集

Timeout

(*ECSRenderTaskDefinition*/Timeout)

（可选）

指定操作在 CodeCatalyst 结束操作之前可以运行的时间（以分钟（YAML 编辑器）或小时和分钟（可视化编辑器）为单位。最小值为 5 分钟，最大值如 中的工作流程配额 CodeCatalyst 中描述。默认超时值与最大超时值相同。

对应的 UI：“配置”选项卡/超时 – 可选

Inputs

(*ECSRenderTaskDefinition*/Inputs)

（可选）

`Inputs` 部分中定义了工作流运行期间 `ECSRenderTaskDefinition` 所需的数据。

> **Note**
>
> 每个渲染 Amazon ECS 任务定义操作只能有一个输入（可以是源或构件）。变量不计入此总数。

对应的 UI：输入选项卡

Sources

(*ECSRenderTaskDefinition*/Inputs/Sources)

（如果您的任务定义文件存储在源存储库中，则为必需项）

如果您的任务定义文件存储在源存储库中，请指定该源存储库的标签。目前，唯一支持的标签是 `WorkflowSource`。

如果您的任务定义文件不包含在源存储库中，则必须位于另一个操作生成的构件中。

有关来源的更多信息，请参阅将源存储库连接到工作流。

对应的 UI：“输入”选项卡/来源 – 可选

Artifacts - input

(*ECSRenderTaskDefinition*/Inputs/Artifacts)

（如果您的任务定义文件存储在上一操作生成的输出构件中，则为必需项）

如果要部署的任务定义文件包含在上一操作生成的构件中，请在此处指定该构件。如果您的任务定义文件不包含在构件中，则必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“配置”选项卡/构件 – 可选

Variables - input

(*ECSRenderTaskDefinition*/Inputs/Variables)

（必需）

指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流中使用变量。

对应的 UI：“输入”选项卡/变量 – 可选

Configuration

(*ECSRenderTaskDefinition*/Configuration)

（必需）

可在其中定义操作的配置属性的部分。

对应的 UI：配置选项卡

task-definition

(*ECSRenderTaskDefinition*/Configuration/task-definition)

（必需）

指定现有任务定义文件的路径。如果文件位于源存储库中，则该路径相对于源存储库根文件夹。如果文件位于上一工作流操作生成的构件中，则该路径相对于构件根文件夹。有关任务定义文件的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的任务定义。

对应的 UI：“配置”选项卡/任务定义

container-definition-name

(*ECSRenderTaskDefinition*/Configuration/container-definition-name)

（必需）

指定将在其中运行 Docker 映像的容器的名称。您可以在任务定义文件的“`containerDefinitions`, `name`”字段中找到此名称。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的名称。

对应的 UI：“配置”选项卡/容器名称

image

(*ECSRenderTaskDefinition*/Configuration/image)

（必需）

指定您希望渲染 Amazon ECS 任务定义操作添加到任务定义文件中的 Docker 映像的名称。此操作会将此名称添加到任务定义文件中的“containerDefinitions, image”字段。如果 image 字段中已有一个值，此操作会将其覆盖。可以在映像名称中包含变量。

示例：

如果您指定MyDockerImage:${WorkflowSource.CommitId}，则该操作会MyDockerImage:*commit-id*添加到任务定义文件中，其中*commit-id*是工作流程在运行时生成的提交 ID。

如果您指定my-ecr-repo/image-repo:$(date +%m-%d-%y-%H-%m-%s)，则该操作会将 *my-ecr-repo* /image-repo: *date +%m-%d-%y-%H-%m-%s* 添加到任务定义文件中，其中*my-ecr-repo*是 Amazon 弹性容器注册表 (ECR) 的 URI，*date +%m-%d-%y-%H-%m-%s*是工作流程在运行时month-day-year-hour-minute-second生成的格式的时间戳。

有关 image 字段的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的映像。有关变量的更多信息，请参阅在工作流中使用变量。

对应的 UI：“配置”选项卡/映像名称

environment-variables

(*ECSRenderTaskDefinition*/Configuration/environment-variables)

（必需）

指定您希望渲染 Amazon ECS 任务定义操作添加到任务定义文件中的环境变量。此操作会将变量添加到任务定义文件中的“containerDefinitions, environment”字段。如果文件中已有变量，则此操作将覆盖现有变量的值并添加任意新变量。有关 Amazon ECS 环境变量的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的指定环境变量。

对应的 UI：“配置”选项卡/环境变量 – 可选

Outputs

(*ECSRenderTaskDefinition*/Outputs)

（必需）

定义在工作流运行期间操作输出的数据。

对应的 UI：输出选项卡

Artifacts

(*ECSRenderTaskDefinition*/Outputs/Artifacts)

（必需）

指定操作生成的构件。您可以在其他操作中将这些构件作为输入来引用。

有关构件的更多信息（包括示例），请参阅在操作之间共享构件和文件。

对应的 UI：“输出”选项卡/构件

Name

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Name)

（必需）

指定将包含更新后的任务定义文件的构件的名称。默认值为 `MyTaskDefinitionArtifact`。之后，您必须将此构件指定为部署到 Amazon ECS 操作的输入。要了解如何将此构件添加为部署到 Amazon ECS 操作的输入，请参阅示例：修改 Amazon ECS taskdef。

对应的 UI：“输出”选项卡/构件/名称

Files

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Files)

（必需）

指定要包含在构件中的文件。您必须指定 `task-definition-*` 以包括更新后的任务定义文件（以 `task-definition-` 开头）。

对应的 UI：“输出”选项卡/构件/文件

Variables

(*ECSRenderTaskDefinition*/Outputs/Variables)

（必需）

指定要由渲染操作设置的变量的名称。渲染操作会将此变量的值设置为更新后的任务定义文件的名称（例如 `task-definition-random-string.json`）。之后，您必须在部署到 Amazon ECS 操作的任务定义（可视化编辑器）或 `task-definition`（YAML 编辑器）属性中指定此变量。要了解如何将此变量添加到部署到 Amazon ECS 操作，请参阅示例：修改 Amazon ECS taskdef。

默认值：`task-definition`

对应的 UI：“输出”选项卡/变量/名称字段

在工作流中使用变量

变量是一个键值对，其中包含您可以在 Ama CodeCatalyst zon 工作流程中引用的信息。工作流运行时，变量的值部分将替换为实际值。

在工作流中，您可以使用两种类型的变量：

- 用户定义的变量 – 这些是您定义的键-值对。
- 预定义变量 – 这些是工作流自动发出的键-值对。您无需对其进行定义。

有关工作流的更多信息，请参阅使用工作流进行构建、测试和部署。

> **Note**
>
> CodeCatalyst 还支持GitHub 输出参数，这些参数的行为类似于变量，可以在其他操作中引用。有关更多信息，请参阅 导出 GitHub 输出参数 和 引用 GitHub 输出参数

主题

使用用户定义的变量

用户定义的变量是您定义的键-值对。这些变量分为两种类型：

- 纯文本变量，简称为变量 – 这些是您在工作流定义文件中以纯文本形式定义的键-值对。
- 密钥 — 这些是您在 Ama CodeCatalyst zon 控制台的单独密钥页面上定义的键值对。键（名称）是一个公共标签，值包含您要保密的信息。您只能在工作流定义文件中指定键。在工作流定义文件中，使用密钥代替密码和其他敏感信息。

> **Note**
>
> 为简洁起见，本指南使用术语变量来表示纯文本变量。

有关变量的更多信息，请参阅在工作流中使用变量。

主题

变量示例

以下示例演示了如何在工作流定义文件中定义和引用变量。

有关变量的更多信息，请参阅在工作流中使用变量。

示例

- 示例：引用密钥

示例：使用 Inputs 属性定义变量

以下示例向您演示了如何在 Inputs 部分中定义两个变量 VAR1 和 VAR2。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
      - Name: VAR1
        Value: "My variable 1"
      - Name: VAR2
        Value: "My variable 2"
```

示例：使用 Steps 属性定义变量

以下示例向您演示了如何在 Steps 部分中显式定义 DATE 变量。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: DATE=$(date +%m-%d-%y)
```

示例：使用 Outputs 属性导出变量

以下示例向您演示了如何使用 Outputs 部分定义两个变量 REPOSITORY-URI 和 TIMESTAMP 并导出它们。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: REPOSITORY-URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
    Configuration:
```

```
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - REPOSITORY-URI
        - TIMESTAMP
```

示例：引用在同一操作中定义的变量

以下示例向您演示了如何在 MyBuildAction 中指定变量 VAR1，然后使用 $VAR1 在同一个操作中引用该变量。

```
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: VAR1
          Value: my-value
    Configuration:
      Steps:
        - Run: $VAR1
```

示例：引用在其他操作中定义的变量

以下示例向您演示了如何在 BuildActionA 中指定变量 TIMESTAMP，使用 Outputs 属性导出变量，然后使用 ${BuildActionA.TIMESTAMP} 在 BuildActionB 中进行引用。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - TIMESTAMP
  BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: docker build -t my-ecr-repo/image-repo:latest .
```

```
        - Run: docker tag my-ecr-repo/image-repo:${BuildActionA.TIMESTAMP}

        # Specifying just '$TIMESTAMP' here will not work
        # because TIMESTAMP is not a variable
        # in the BuildActionB action.
```

示例：引用密钥

以下示例演示如何引用 my-password 密钥。my-password 是密钥的键。此密钥的密钥和相应的密码值必须先在 CodeCatalyst 控制台的 S ec rets 页面上指定，然后才能在工作流程定义文件中使用。有关更多信息，请参阅 使用密钥遮蔽数据。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: curl -u LiJuan:${Secrets.my-password} https://example.com
```

定义变量

您可以通过两种方式定义变量：

- 在工作流操作的 Inputs 部分 – 请参阅在“输入”部分中定义变量
- 在工作流操作的 Steps 部分 – 请参阅在“步骤”部分中定义变量

> **Note**
>
> 该Steps方法仅适用于 CodeCatalyst 构建、测试和GitHub 操作操作，因为这些是唯一包含Steps部分的操作。

有关示例，请参阅 变量示例。

有关变量的更多信息，请参阅在工作流中使用变量。

Visual

在“输入”部分中定义变量（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要设置变量的操作。
8. 选择输入。
9. 在变量 – 可选中，选择添加变量，然后执行以下操作：

   指定一系列 name/value 对，用于定义要提供给操作的输入变量。变量名称仅限字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号以使变量名能够包含特殊字符和空格。

   有关变量的更多信息（包括示例），请参阅在工作流中使用变量。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

在“输入”部分中定义变量（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流操作中，添加类似于下文的代码：

```
action-name:
  Inputs:
    Variables:
```

```
      - Name: variable-name
        Value: variable-value
```

有关更多示例，请参阅变量示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

Visual

在“步骤”部分中定义变量（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要设置变量的操作。
8. 选择配置。
9. 在 Shell 命令或 GitHub Actions YAML（无论哪个可用）中，在操作中显式或隐`Steps`式地定义一个变量。

   - 要显式定义变量，请将其直接包含在 `Steps` 部分的 bash 命令中。
   - 要隐式定义变量，请在操作的 `Steps` 部分引用的文件中指定该变量。

   有关示例，请参阅 变量示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

在“步骤”部分中定义变量（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流操作中，在操作的 Steps 部分中显式或隐式定义变量。

   - 要显式定义变量，请将其直接包含在 Steps 部分的 bash 命令中。
   - 要隐式定义变量，请在操作的 Steps 部分引用的文件中指定该变量。

   有关示例，请参阅 变量示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

定义密钥

您可以在 CodeCatalyst控制台的 “密钥” 页面上定义密钥。有关更多信息，请参阅 使用密钥遮蔽数据。

例如，您可以定义一个如下所示的密钥：

- 名称（键）：**my-password**
- 值：**^*H3#!b9**

定义密钥后，您可以在工作流定义文件中指定密钥的键（**my-password**）。有关如何执行此操作的示例，请参阅示例：引用密钥。

导出变量以便其他操作使用

按照以下说明从操作中导出变量，以便在其他操作中引用该变量。

在导出变量之前，请注意以下事项：

- 如果您只需要在定义变量的操作中引用该变量，则无需将其导出。
- 并非所有操作都支持导出变量。要确定您的操作是否支持此功能，请仔细阅读后文的可视化编辑器说明，并查看操作的输出选项卡上是否包含变量按钮。如果是，则支持导出变量。

- 要从 GitHub 操作中导出变量，请参阅导出 GitHub 输出参数。

有关变量的更多信息，请参阅在工作流中使用变量。

先决条件

确保您已定义要导出的变量。有关更多信息，请参阅定义变量。

Visual

导出变量（可视化编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。
7. 在工作流图表中，选择要从中导出变量的操作。
8. 选择输出。
9. 在变量 – 可选中，选择添加变量，然后执行以下操作：

   指定希望操作导出的变量的名称。此变量必须已在同一操作的 `Inputs` 或 `Steps` 部分中定义。
10. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
11. 选择提交，输入提交消息，然后再次选择提交。

YAML

导出变量（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。

4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要从中导出变量的操作中，添加类似于下文的代码：

```
action-name:
  Outputs:
    Variables:
      - Name: variable-name
```

有关更多示例，请参阅变量示例。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

在定义变量的操作中引用该变量

按照以下说明在定义变量的操作中引用该变量。

> **Note**
>
> 要引用 GitHub 操作生成的变量，请参见引用 GitHub 输出参数。

有关变量的更多信息，请参阅在工作流中使用变量。

先决条件

确保您已定义了要引用的变量。有关更多信息，请参阅 定义变量。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

在定义变量的操作中引用该变量

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在定义要引用的变量的 CodeCatalyst 操作中，使用以下 bash 语法添加变量：

```
$variable-name
```

例如：

```
MyAction:
    Configuration:
      Steps:
        - Run: $variable-name
```

有关更多示例，请参阅变量示例。有关更多信息，请参阅工作流 YAML 定义中相应操作的参考信息。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

引用其他操作输出的变量

按照以下说明引用其他操作输出的变量。

> **Note**
>
> 要引用 GitHub 操作的变量输出，请参见引用 GitHub 输出参数。

有关变量的更多信息，请参阅在工作流中使用变量。

先决条件

确保您已导出了要引用的变量。有关更多信息，请参阅导出变量以便其他操作使用。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

引用其他操作输出的变量（YAML 编辑器）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 CodeCatalyst 操作中，使用以下语法添加对变量的引用：

```
${action-group-name.action-name.variable-name}
```

进行如下替换：

- *action-group-name*使用包含输出变量的操作的操作组的名称。

  > **Note**
  >
  > *action-group-name*如果没有操作组，或者变量是由同一操作组中的操作生成的，则可以省略。

- *action-name*使用输出变量的操作的名称。
- *variable-name*使用变量的名称。

例如：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: ${MyFirstAction.TIMESTAMP}
```

有关更多示例，请参阅变量示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

引用密钥

有关在工作流定义文件中引用密钥的说明，请参阅使用密钥。

有关示例，请参阅示例：引用密钥。

使用预定义变量

预定义变量是键-值对，由工作流自动发出，可供您在工作流操作中使用。

有关变量的更多信息，请参阅在工作流中使用变量。

主题

- 引用预定义变量的示例
- 引用预定义变量
- 确定您的工作流会发出哪些预定义变量
- 预定义变量列表

引用预定义变量的示例

以下示例演示了如何在工作流定义文件中引用预定义变量。

有关预定义变量的更多信息，请参阅使用预定义变量。

示例

- 示例：引用“CommitId”预定义变量
- 示例：引用“BranchName”预定义变量

示例：引用“CommitId”预定义变量

以下示例向您演示了如何在 `MyBuildAction` 操作中引用 `CommitId` 预定义变量。`CommitId` 变量由 CodeCatalyst 自动输出。有关更多信息，请参阅预定义变量列表。

尽管示例显示的是构建操作中使用的变量，不过您可以在任何操作中使用 CommitId。

```
MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
      #Build Docker image and tag it with a commit ID
        - Run: docker build -t image-repo/my-docker-image:latest .
        - Run: docker tag image-repo/my-docker-image:${WorkflowSource.CommitId}
```

示例：引用“BranchName”预定义变量

以下示例向您演示了如何在 CDKDeploy 操作中引用 BranchName 预定义变量。BranchName 变量由 CodeCatalyst 自动输出。有关更多信息，请参阅 预定义变量列表。

尽管该示例显示的是 AWS CDK 部署操作中使用的变量，不过您可以在任何操作中使用 BranchName。

```
CDKDeploy:
    Identifier: aws/cdk-deploy@v2
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: app-stack-${WorkflowSource.BranchName}
```

引用预定义变量

您可以在 Amazon CodeCatalyst 工作流的任何操作中引用预定义的变量。

按照以下说明在工作流中引用预定义变量。

有关预定义变量的更多信息，请参阅使用预定义变量。

先决条件

确定要引用的预定义变量的名称，例如 CommitId。有关更多信息，请参阅确定您的工作流会发出哪些预定义变量。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

引用预定义变量（YAML 编辑器）

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 CodeCatalyst 操作中，使用以下语法添加对预定义变量的引用：

```
${action-group-name.action-name-or-WorkflowSource.variable-name}
```

进行如下替换：

- 将 *action-group-name* 替换为操作组的名称。

  > **Note**
  >
  > 如果没有操作组，或者变量是由相同操作组中的操作生成的，则可以省略 *action-group-name*。

- 将 *action-name-or-WorkflowSource* 替换为：

  输出变量的操作的名称。

  或

  `WorkflowSource`，如果变量是 `BranchName` 或 `CommitId` 变量。

- 将 *variable-name* 替换为变量的名称。

例如：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: echo ${MyFirstECSAction.cluster}
```

另一个示例是：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: echo ${WorkflowSource.CommitId}
```

有关更多示例，请参阅引用预定义变量的示例。有关更多信息，请参阅相应操作的工作流 YAML 定义。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

确定您的工作流会发出哪些预定义变量

使用以下过程来确定工作流在运行时会发出哪些预定义变量。然后，您可以在同一个工作流中引用这些变量。

有关预定义变量的更多信息，请参阅使用预定义变量。

确定工作流发出的预定义变量

- 请执行以下操作之一：
  - 运行一次工作流。运行完成后，工作流发出的变量将显示在运行详细信息页面的变量选项卡上。有关更多信息，请参阅查看工作流运行状态和详细信息。
  - 请参考预定义变量列表。此参考资料列出了每个预定义变量的变量名（键）和值。

> **Note**
>
> 工作流变量的最大总大小在 中的工作流程配额 CodeCatalyst中列出。如果总大小超过最大值，则在达到最大值之后执行的操作可能会失败。

预定义变量列表

请参阅以下各节，查看 CodeCatalyst 操作在工作流运行过程中自动生成的预定义变量。

有关预定义变量的更多信息，请参阅使用预定义变量。

> **Note**
>
> 此列表只包括由 CodeCatalyst 源和 CodeCatalyst 操作发布的预定义变量。如果您使用的是其它类型的操作，例如 GitHub Actions 或 CodeCatalyst Labs 操作，请改为参阅确定您的工作流会发出哪些预定义变量。

列表

> **Note**
>
> 并非所有 CodeCatalyst 操作都会产生预定义变量。如果操作不在列表中，则不会产生变量。

- BranchName'和' CommitId '变量
- '部署 CloudFormation 堆栈'变量
- “部署到 Amazon ECS”变量
- “部署到 Kubernetes 集群”变量
- “AWS CDK 部署”变量
- “AWS CDK 引导”变量
- “AWS Lambda 调用”变量
- “渲染 Amazon ECS 任务定义”变量

使用密钥遮蔽数据

有时您可能需要在工作流中使用敏感数据，例如身份验证凭证。应避免将这些值以纯文本形式存储在存储库中的任何地方，因为任何有权访问包含密钥的存储库的人都可以看到密钥。同样，不应在任何工作流定义中直接使用这些值，因为它们将作为文件在存储库中可见。使用 CodeCatalyst，您可以通过向项目添加密钥，然后在工作流程定义文件中引用该密钥来保护这些值。请注意，每个操作最多可有 5 个密钥。

> **Note**
>
> 密钥只能用于替换工作流定义文件中的密码和敏感信息。

主题

创建密钥

使用以下过程创建密钥。密钥包含您想要隐藏以防其他人查看的敏感信息。

> **Note**
>
> 密钥对操作可见，在写入文件时不会被屏蔽。

创建密钥

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择密钥。
3. 选择创建密钥。
4. 输入以下信息：

   名称

   输入密钥的名称。

   值

   输入密钥的值。这是您想要隐藏以防其他人查看的敏感信息。默认情况下，该值不显示。要显示该值，请选择显示值。

   描述

   （可选）输入密钥的描述。

5. 选择创建。

编辑密钥

使用以下过程编辑步骤。

编辑密钥

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择密钥。
3. 在密钥列表中，选择要编辑的密钥。
4. 选择编辑。
5. 编辑以下属性：

   值

   输入密钥的值。这是您要隐藏以防其他人查看的值。默认情况下，该值不显示。

   描述

   （可选）输入密钥的描述。
6. 选择保存。

使用密钥

要在工作流操作中使用密钥，您必须获取密钥的引用标识符并在工作流操作中使用该标识符。

主题

- 获取密钥的标识符
- 在工作流中引用密钥

获取密钥的标识符

使用以下过程获取密钥的引用标识符。您将此标识符添加到工作流中。

获取密钥的引用标识符

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 在导航窗格中，选择 CI/CD，然后选择密钥。
3. 在密钥列表中，选择要使用的密钥。
4. 在引用 ID 列中，复制密钥的标识符。以下是引用 ID 的语法：

```
${Secrets.<name>}
```

在工作流中引用密钥

使用以下过程在工作流中引用密钥。

引用密钥

1. 在导航窗格中，选择 CI/CD，然后选择工作流。
2. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
3. 选择编辑。
4. 选择 YAML。
5. 修改 YAML 以使用密钥的标识符。例如，要在 `curl` 命令中使用存储为密钥的用户名和密码，应使用类似于以下内容的 Run 命令：

```
- Run: curl -u <username-secret-identifier>:<password-secret-identifier> https://
example.com
```

6. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
7. 选择提交，输入提交消息，然后再次选择提交。

删除密钥

使用以下步骤来删除密钥和密钥引用标识符。

> **Note**
>
> 在删除密钥之前，我们建议您从所有工作流操作中移除该密钥的引用标识符。如果您删除密钥而不删除引用标识符，则该操作在下次运行时将失败。

从工作流中删除密钥的引用标识符

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择工作流。
3. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
4. 选择编辑。
5. 选择 YAML。
6. 在工作流中搜索以下字符串：

   ```
   ${Secrets.
   ```

   这将找到所有密钥的所有引用标识符。
7. 删除所选密钥的参考标识符，或将其替换为纯文本值。
8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。

删除密钥

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 CI/CD，然后选择密钥。
3. 在密钥列表中，选择要删除的密钥。
4. 选择删除。
5. 输入 **`delete`** 以确认删除。
6. 选择删除。

查看工作流的状态

您可能需要查看工作流程的状态，以查看是否存在需要解决的工作流程配置问题，或者对无法启动的运行进行故障排除。 CodeCatalyst每次创建或更新工作流程的基础工作流定义文件时，都会评估工作流程状态。

> **Note**
>
> 您还可以查看工作流的运行状态，该状态不同于工作流状态。有关更多信息，请参阅查看工作流运行状态和详细信息。

有关可能的工作流状态列表，请参阅 工作流程状态为 CodeCatalyst。

查看工作流的状态

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

   状态与工作流一起显示在列表中。
5. （可选）选择工作流的名称，然后找到工作流定义字段。其中显示工作流的状态。

中的工作流程配额 CodeCatalyst

下表描述了 Amazon 中工作流程的配额和限制 CodeCatalyst。

有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

每个空间的最大工作流数量	800
工作流定义文件的最大大小	256 KB
在单个源事件中处理的工作流文件的最大数量	50
在单个源事件中处理的文件的最大数量	4,000
每个空间的最大活动实例集数量	10
每个实例集的最大活动计算实例数量	20
每个操作的最大输入构件数	10

每个操作的最大输出构件数	10
单个操作输出变量的最大总大小	120 KB
输出变量值的最大长度	500 个字符或更多，具体取决于发出该值的操作。 **Note** 如果值超过操作的限制，则可能会被截断。
工作流运行期间保留生成构件的最大天数	30
每个操作的最大报告数	50
每份测试报告的最大测试用例数	20000
每份代码覆盖率报告的最大文件数	20000
每份报告的软件组成分析调查发现最大数量	20000
每份静态分析报告的文件最大数	20000
每个空间的最大并发工作流运行数	100
每个工作流的最大操作数量	50
每个工作流并发运行的操作的最大数	50
每个空间并发运行的操作的最大数	200
操作可以运行的最长时间	对于构件和测试操作，超时时间为 8 小时。 对于所有其他操作，超时时间为 1 小时。
每个空间与 AWS 账户 关联的环境最大数	5000
每个操作的最大密钥数	5
每个空间的最大密钥数	500,000

工作流运行状态

工作流运行可以处于以下几种状态之一：

- 成功 – 工作流运行已成功处理。
- 失败 – 工作流运行中的一个或多个操作失败。
- 进行中 – 工作流运行当前正在处理中。
- 已停止 – 在工作流运行期间，有人停止了工作流的运行。
- 正在停止 – 当前正在停止工作流运行。
- 已取消 — 工作流程运行已被取消， CodeCatalyst 因为在运行过程中关联的工作流程已被删除或更新。
- 已取代 – 仅在配置了取代运行模式时才会发生。工作流程运行已被取消， CodeCatalyst 因为稍后的工作流程运行取代了该运行。

工作流程状态为 CodeCatalyst

工作流可能具有以下状态之一：

- 有效 – 工作流可运行，可通过触发器激活。

  标记为有效的工作流必须满足以下两个条件：

  - 工作流定义文件必须有效。
  - 工作流必须没有触发器，没有推送触发器，也没有使用当前分支上的文件运行的推送触发器。有关更多信息，请参阅触发器和分支的使用准则。
- 无效 – 工作流的定义文件无效。工作流无法手动运行，也无法通过触发器自动运行。无效的工作流程与工作流定义一起出现在 CodeCatalyst 控制台中出现***n***错误消息（或类似消息）。

  标记为无效的工作流必须满足以下条件：

  - 工作流定义文件必须有配置错误。

    要修复配置错误的工作流定义文件，请参阅如何修复“工作流定义有 *n* 个错误”错误？。
- 非活动 – 工作流定义有效，但无法手动运行，也无法通过触发器自动运行。

  标记为非活动的工作流必须满足以下两个条件：

  - 工作流定义文件必须有效。

- 工作流定义文件必须包含推送触发器，该触发器指定的分支不同于工作流定义文件当前所在的分支。有关更多信息，请参阅触发器和分支的使用准则。

  要将工作流从非活动切换为活动，请参阅如何修复“工作流非活动”消息？。

  > **Note**
  >
  > 如果有多个工作流处于非活动状态，则可以将其从视图中筛选出来。要筛选出非活动的工作流，请选择工作流页面顶部的筛选工作流字段，选择状态，选择状态 != 不等于，然后选择非活动。

> **Note**
>
> 如果工作流程指定了您稍后移除的资源（例如包存储库），则 CodeCatalyst 不会检测到此更改，并将继续将该工作流程标记为有效。这些类型的问题将在工作流运行时发现。

工作流 YAML 定义

以下是工作流定义文件的参考文档。

工作流定义文件是描述您的工作流的 YAML 文件。默认情况下，该文件存储在源存储库根目录下的 `~/.codecatalyst/workflows/` 文件夹中。该文件的扩展名可以为 .yml 或 .yaml，并且扩展名必须小写。

要创建和编辑工作流程定义文件，您可以使用编辑器（例如 vim），也可以使用 CodeCatalyst 控制台的可视化编辑器或 YAML 编辑器。有关更多信息，请参阅 使用 CodeCatalyst 控制台的视觉编辑器和 YAML 编辑器。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有对应的 UI 元素。要查找 UI 元素，请使用 Ctrl+F。该元素将与其关联的 YAML 属性一起列出。

主题

工作流定义文件示例

下面是一个简单的工作流定义文件示例。该示例包括几个顶级属性、一个 Triggers 部分和一个包含 Build 和 Test 这两个操作的 Actions 部分。有关更多信息，请参阅 关于工作流定义文件。

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest .
  Test:
    Identifier: aws/managed-test@v1
    DependsOn:
      - Build
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
```

语法准则和惯例

本节介绍工作流定义文件的语法规则，以及本参考文档中使用的命名约定。

YAML 语法准则

工作流定义文件是用 YAML 编写的，并遵循 YAML 1.1 规范，因此该规范中允许的内容在工作流 YAML 中也是允许的。如果您不熟悉 YAML，这里有一些快速指南，可确保您提供有效的 YAML 代码。

- 区分大小写：工作流定义文件区分大小写，因此请确保使用本文档中显示的大小写。
- 特殊字符：我们建议使用引号或双引号将包含以下任意特殊字符的属性值引起来：{、}、[、]、&、*、#、?、|、-、<、>、=、!、%、@、:、` 和 ,

  如果不加引号，前面列出的特殊字符可能会以意想不到的方式解释。
- 属性名称：属性名称（相对于属性值）仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号或双引号在属性名称中启用特殊字符和空格。

  不允许：

  ```
  'My#Build@action'
  ```

  ```
  My#Build@action
  ```

  ```
  My Build Action
  ```

  允许：

  ```
  My-Build-Action_1
  ```
- 转义码：如果您的属性值包含转义码（如 \n 或 \t），请遵循以下准则：
  - 使用单引号以字符串形式返回转义码。例如，'my string \n my string' 返回 my string \n my string 字符串。
  - 使用双引号解析转义码。例如，"my string \n my new line" 返回：

    ```
    my string
    my new line
    ```
- 注释：在注释前加上 #。

  示例：

  ```
  Name: MyWorkflow
  # This is a comment.
  ```

```
SchemaVersion: 1.0
```

- Triple dash (`---`)：不要`---`在你的 YAML 代码中使用。 CodeCatalyst 忽略后面的所有内容。`---`

命名约定

在本指南中，我们使用属性和部分两词来指代工作流定义文件中的主要项目。

- 属性是任何包含冒号（`:`）的项目。例如，在下面的代码片段中，以下所有项都是属性：`Name`、`SchemaVersion`、`RunMode`、`Triggers`、`Type` 和 `Branches`。
- 部分是任何具有子属性的属性。在下面的代码片段中，有一个 `Triggers` 部分。

> **Note**
>
> 在本指南中，“部分”有时被称为“属性”，反之亦然，视上下文而定。

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
```

顶级属性

以下是工作流定义文件中顶级属性的参考文档。

```
# Name
Name: workflow-name

# Schema version
SchemaVersion: 1.0

# Run mode
RunMode: QUEUED|SUPERSEDED|PARALLEL

# Compute
```

```
Compute:
...

# Triggers
Triggers:
...

# Actions
Actions:
...
```

Name

（必需）

工作流的名称。工作流名称显示在工作流列表中，并在通知和日志中提及。工作流名称和工作流定义文件名可以一致，也可以不同。工作流名称不必是唯一的。工作流名称仅限于字母数字字符（a-z、A-Z、0-9）、连字符（-）和下划线（_）。不允许使用空格。不能使用引号在工作流名称中启用特殊字符和空格。

对应的用户界面：视觉 editor/Workflow 属性/工作流程名称

SchemaVersion

（必需）

工作流定义的架构版本。目前唯一有效的值是 `1.0`。

对应的 UI：无

RunMode

（可选）

如何 CodeCatalyst 处理多次运行。可以使用下列值之一：

- QUEUED – 多个运行排队并一个接一个地运行。一个队列中最多可包含 50 个运行。
- SUPERSEDED – 多个运行排队并一个接一个地运行。一个队列只能有一个运行，因此如果两个运行同时出现在同一个队列中，则后一个运行将取代（接替）前一个运行，而前一个运行将被取消。
- PARALLEL – 同时进行多个运行。

如果省略该属性，则默认值为 QUEUED。

有关更多信息，请参阅 配置运行的排队行为。

对应的用户界面：视觉 editor/Workflow 属性/高级/运行模式

Compute

（可选）

用于运行工作流操作的计算引擎。您可以在工作流级别或操作级别指定计算，但不能同时在这两个级别指定计算。在工作流级别指定计算时，计算配置将应用于工作流中定义的所有操作。在工作流级别，您还可以在同一个实例上运行多个操作。有关更多信息，请参阅 跨操作共享计算。

有关计算的更多信息，请参阅配置计算和运行时映像。

对应的 UI：无

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Compute:
  Type: EC2 | Lambda
  Fleet: fleet-name
  SharedInstance: true | false
```

Type

(Compute/Type)

（设置了 Compute 时是必需的）

计算引擎的类型。可以使用下列值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  已经过优化，提高了操作运行期间的灵活性。
- Lambda（可视化编辑器）或 Lambda（YAML 编辑器）

  优化了操作启动速度。

有关计算类型的更多信息，请参阅计算类型。

对应的用户界面：视觉 editor/Workflow 属性/高级/计算类型

Fleet

(Compute/Fleet)

（可选）

指定将运行您的工作流或工作流操作的计算机或实例集。对于按需实例集，当操作开始时，工作流会预置操作所需的资源，操作完成后计算机就会被销毁。按需实例集的示例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。有关按需实例集的更多信息，请参阅按需实例集属性。

使用预置的实例集，您可以配置一组专用计算机来运行工作流操作。这些计算机保持空闲状态，可随时开始立即处理操作。有关预置实例集的更多信息，请参阅预置实例集属性。

如果省略 `Fleet`，则默认值为 `Linux.x86-64.Large`。

有关计算实例集的更多信息，请参阅计算实例集。

对应的用户界面：视觉 editor/Workflow 属性/高级/计算舰队

SharedInstance

(Compute/SharedInstance)

（可选）

为您的操作指定计算共享功能。通过计算共享，工作流中的操作会在同一个实例（运行时环境映像）上运行。可以使用下列值之一：

- `TRUE` 表示工作流操作之间共享运行时环境映像。
- `FALSE` 表示为工作流中的每个操作启动和使用单独的运行时环境映像，因此如果没有额外的配置，就无法共享构件和变量等资源。

有关计算共享的更多信息，请参阅跨操作共享计算。

对应的 UI：无

Triggers

（可选）

该工作流的一个或多个触发器序列。如果未指定触发器，则必须手动启动工作流。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

对应的用户界面：可视化 editor/workflow 图表/触发器

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Triggers:
  - Type: PUSH
    Branches:
      - branch-name
    FilesChanged:
      - folder1/file
      - folder2/

  - Type: PULLREQUEST
    Events:
      - OPEN
      - CLOSED
      - REVISION
    Branches:
      - branch-name
    FilesChanged:
      - file1.txt

  - Type: SCHEDULE
    # Run the workflow at 10:15 am (UTC+0) every Saturday
    Expression: "15 10 ? * 7 *"
    Branches:
      - branch-name
```

Type

(Triggers/Type)

（设置了 Triggers 时是必需的）

指定触发器的类型。可以使用下列值之一：

- 推送（可视化编辑器）或 PUSH（YAML 编辑器）

  当更改推送到源存储库时，推送触发器会启动工作流运行。工作流运行将使用您推送到的分支（即目标分支）中的文件。

- 拉取请求（可视化编辑器）或 PULLREQUEST（YAML 编辑器）

  在源存储库中打开、更新或关闭拉取请求时，拉取请求触发器会启动工作流运行。工作流运行将使用您拉取自的分支（即源分支）中的文件。

- 计划（可视化编辑器）或 SCHEDULE（YAML 编辑器）

  计划触发器按您指定的 cron 表达式定义的计划来启动工作流运行。对于源存储库中的每个分支，将使用分支的文件启动单独的工作流运行。[要限制在其上激活触发器的分支，请使用分支字段（可视化编辑器）或 Branches 属性（YAML 编辑器）。]

  配置计划触发器时，请遵循以下指南：

  - 每个工作流只能使用一个计划触发器。
  - 如果您在自己的 CodeCatalyst 空间中定义了多个工作流程，我们建议您安排的同时启动不超过 10 个工作流程。
  - 确保将触发器的 cron 表达式配置为在两次运行之间留出足够的时间。有关更多信息，请参阅Expression。

有关示例，请参阅 示例：工作流中的触发器。

对应的用户界面：可视化 editor/workflow 图/触发器/触发器类型

Events

(Triggers/Events)

（如果触发器 Type 设置为 PULLREQUEST，则是必需的）

指定将启动工作流运行的拉取请求事件的类型。有效值如下所示：

- 拉取请求已创建（可视化编辑器）或 OPEN（YAML 编辑器）

  工作流运行将在拉取请求创建后启动。

- 拉取请求已关闭（可视化编辑器）或 CLOSED（YAML 编辑器）

  工作流运行将在拉取请求关闭后启动。CLOSED 事件的行为不太好说明，最好通过一个例子来理解。请参阅示例：带有拉取、分支和“CLOSED”事件的触发器了解更多信息。

- 对拉取请求进行了新的修订（可视化编辑器）或 REVISION（YAML 编辑器）

工作流运行将在对拉取请求的修订创建后启动。在创建拉取请求时，将创建第一个修订。之后，每当有人将新的提交推送到拉取请求中指定的源分支时，就会创建一个新的修订。如果您在拉取请求触发器中包含 REVISION 事件，则可以忽略 OPEN 事件，因为 REVISION 是 OPEN 的超集。

您可以在同一个拉取请求触发器中指定多个事件。

有关示例，请参阅 示例：工作流中的触发器。

相应的用户界面： editor/workflow 拉取请求的可视化图表/触发器/事件

Branches

(Triggers/Branches)

（可选）

指定触发器监控的源存储库中的分支，以便知道何时启动工作流运行。您可以使用正则表达式模式来定义分支名称。例如，使用 `main.*` 来匹配以 `main` 开头的所有分支。

根据触发器的类型，要指定的分支会有所不同：

- 对于推送触发器，请指定要推送至的分支，即目标分支。对于每个匹配的分支，将使用匹配分支中的文件，启动一个工作流运行。

  示例：`main.*`、`mainline`
- 对于拉取请求触发器，请指定要推送至的分支，即目标分支。对于每个匹配的分支，将使用工作流定义文件和源分支（不是匹配的分支）中的文件，启动一个工作流运行。

  示例：`main.*`、`mainline`、`v1\-.*`（匹配以 `v1-` 开头的分支）
- 对于计划触发器，请指定包含您希望计划运行使用的文件的分支。对于每个匹配的分支，将使用工作流定义文件和匹配分支中的源文件，启动一个工作流运行。

  示例：`main.*`、`version\-1\.0`

> **Note**
>
> 如果您未指定分支，则触发器会监控源存储库中的所有分支，并将使用以下位置中的工作流定义文件和源文件启动工作流运行：

- 您要推送至的分支（对于推送触发器）。有关更多信息，请参阅示例：一个简单的代码推送触发器。
- 您要拉取自的分支（对于拉取请求触发器）。有关更多信息，请参阅示例：一个简单的拉取请求触发器。
- 所有分支（对于计划触发器）。源存储库中的每个分支将启动一个工作流运行。有关更多信息，请参阅示例：一个简单的计划触发器。

有关分支和触发器的更多信息，请参阅触发器和分支的使用准则。

有关更多示例，请参阅示例：工作流中的触发器。

对应的用户界面：视觉editor/workflow diagram/Triggers/分支

FilesChanged

(Triggers/FilesChanged)

（如果触发器 Type 设置为 PUSH 或 PULLREQUEST，则为可选的。如果触发器 Type 设置为 SCHEDULE，则不受支持。）

指定触发器监控的源存储库中的文件或文件夹，以便知道何时启动工作流运行。您可以使用正则表达式来匹配文件名或路径。

有关示例，请参阅 示例：工作流中的触发器。

相应的用户界面：可视化 editor/workflow 图表/触发器/文件已更改

Expression

(Triggers/Expression)

（如果触发器 Type 设置为 SCHEDULE，则是必需的）

指定 cron 表达式，用于描述您希望何时执行计划的工作流运行。

中的 Cron 表达式 CodeCatalyst 使用以下六字段语法，其中每个字段用空格分隔：

minutes hours days-of-month month days-of-week year

cron 表达式示例

分钟	小时	一个月中的第几天	Month	星期几	年	意义
0	0	?	*	MON-FRI	*	每个星期一到星期五的午夜（UTC +0）运行工作流。
0	2	*	*	?	*	每天凌晨 2:00（UTC +0）运行工作流。
15	22	*	*	?	*	每天晚上 10:15（UTC +0）运行工作流。
0/30	22-2	?	*	SAT-SUN	*	星期六至星期日，从起始日晚上 10:00 至次日凌晨 2:00（UTC +0）每隔 30 分钟运行一次工作流。
45	13	L	*	?	2023-2027	2023 年至 2027 年（含）之间，在每个月的最后一天下午

分钟	小时	一个月中的第几天	Month	星期几	年	意义
						1:45（UTC +0）运行工作流。

在中指定 cron 表达式时 CodeCatalyst，请务必遵循以下准则：

- 为每个 SCHEDULE 触发器指定一个 cron 表达式。
- 在 YAML 编辑器中，将 cron 表达式用双引号（"）括起来。
- 使用协调世界时（UTC）格式指定时间。不支持其他时区。
- 两次运行之间应至少配置 30 分钟的运行间隔。不支持更快的节奏。
- 指定*days-of-month*或字*days-of-week*段，但不能同时指定两者。如果您在其中一个字段中指定值或星号（*），则必须在另一个字段中使用问号（？）。星号表示“全部”，问号表示“任何”。

有关 cron 表达式的更多示例以及有关通配符（如?、和）的信息 *L，请参阅《亚马逊 EventBridge 用户指南》中的 Cron 表达式参考。 EventBridge 和中的 Cron 表达式 CodeCatalyst 的工作方式完全相同。

有关计划触发器的示例，请参阅示例：工作流中的触发器。

对应的用户界面：视觉editor/workflow diagram/Triggers/日程安排

操作

此工作流程的一个或多个操作序列。 CodeCatalyst 支持多种操作类型，例如生成和测试操作，它们提供不同类型的功能。每种操作类型都有：

- 表示操作的唯一硬编码 ID 的 Identifier 属性。例如，aws/build@v1 标识构建操作。
- 包含特定于操作的属性的 Configuration 部分。

有关每种操作类型的更多信息，请参阅操作类型。操作类型主题有每个操作的文档链接。

以下是工作流定义文件中操作和操作组的 YAML 参考。

```
Name: MyWorkflow
```

```
SchemaVersion: 1.0
...
Actions:
  action-or-gate-name:
    Identifier: identifier
    Configuration:
    ...
  #Action groups
  action-group-name:
    Actions:
      ...
```

action-or-gate-name

(Actions/*action-or-gate-name*)

（必需）

*action-name*替换为要执行操作的名称。操作名称在工作流中必须是唯一的，并且只能包含字母数字字符、连字符和下划线。有关语法规则的更多信息，请参阅 YAML 语法准则。

有关操作命名惯例（包括限制）的更多信息，请参阅 action-or-gate-name。

对应的用户界面：可视化编辑器/ ***action-name*** /配置选项卡/ 操作名称或操作显示名称

action-group-name

(Actions/*action-group-name*)

（可选）

一个操作组包含一个或多个操作。将操作分组为操作组有助于将工作流保持得井井有条，还可以配置不同组之间的依赖关系。

*action-group-name*替换为你要给操作组起的名字。操作组名称在工作流中必须是唯一的，并且只能包含字母数字字符、连字符和下划线。有关语法规则的更多信息，请参阅 YAML 语法准则。

有关操作组的更多信息，请参阅将操作分组为操作组。

对应的 UI：无

跟踪和组织处理问题的工作 CodeCatalyst

在中 CodeCatalyst，您可以监视项目中涉及的功能、错误和任何其他工作。每项工作均保存在称为事务的不同记录中。您可以通过向事务添加任务清单来将事务分解为多个较小目标。每个事务都可以具有描述、被分派人、状态和其他属性，您可以对这些属性进行搜索、分组和筛选。您可以使用默认视图查看自己的事务，也可以使用自定义筛选、排序或分组来创建自己的视图。有关与事务相关的概念的更多信息，请参阅事务概念。要了解如何创建您的首个事务，请参阅在中创建问题 CodeCatalyst。

对于使用事务的团队来说，可能会使用以下工作流：

Jorge Souza 是一名参与项目的开发人员。他和他的项目成员同事 Li Juan、Mateo Jackson 和 Wang Xiulan 合作确定需要完成哪些工作。每天，他和他的开发人员同事们都会参加由 Wang Xiulan 主持的同步会议。他们通过导航到面板的某个团队视图来调出面板。通过创建视图，用户和团队可以保存事务的筛选条件、分组和排序，以便轻松查看符合其指定条件的事务。他们的视图包含按被分派人分组并按优先级排序的事务，以显示每个开发人员的最重要事务及其状态。在向 Jorge 分配要完成的任务时，他通过为每个任务创建一个事务来规划自己的工作。在创建事务时，Jorge 可以选择相应的状态、优先级和工作估算工作量。对于较大的事务，Jorge 会向事务添加任务，将工作细分成较小目标。Jorge 创建具有草稿状态的事务，例如待办事项，因为他不打算立即开始处理这些事务。具有草稿状态的事务将显示在草稿视图中，将在该视图中规划这些事务并确定其优先级。在 Jorge 准备好开始工作后，他会将相应事务的状态更新为另一个类别（未开始、已开始或已完成）中的状态，从而将该事务移至面板。在处理每个任务时，团队可以按标题、状态、被分派人、标签、优先级和估算进行筛选，以找到与指定参数匹配的特定事务或类似事务。使用看板，Jorge 和他的团队可以看到每个问题已完成的任务数量，并通过将每个问题从一种状态拖到下一个状态直到任务完成来跟踪 day-to-day进度。随着项目的进行，已完成的事务将累积并处于已完成状态。Wang Xiulan 决定使用快速存档按钮对事务进行归档，从而将其从视图中移除，这样开发人员就能专注于与当前和即将开展的工作相关的事务。

在规划期工作时，参与项目的开发人员会选择排序依据和分组依据，以查找他们要从待办事项移至面板的事务。他们可能会选择根据优先级最高的客户请求向面板添加事务，因此他们按客户请求标签对面板进行分组，并按优先级对其进行排序。他们还可以按估算进行排序，以确保自己有能力完成所分配的工作量。项目经理 Saanvi Sarkar 会定期审查和整理待办事项，以帮助确保优先级准确反映每个事务对项目取得成功的重要性。

主题

- 事务概念
- 跟踪有关事务的工作
- 使用待办事项、标签和面板整理工作

- 中的问题配额 CodeCatalyst

事务概念

创建事务可以快速有效地跟踪项目中正在完成的工作。您可以使用事务来帮助您在每日同步会议中讨论工作、确定工作的优先顺序等。

本页包含一个概念列表，可帮助您有效地使用中的问题 CodeCatalyst。

活跃事务

活跃事务是指任何未处于草稿状态或已存档状态的事务。换而言之，活跃事务是指处于以下任意状态类别中的状态的事务：未开始、已开始和已完成。有关状态和状态类别的更多信息，请参阅状态和状态类别。

您可以从默认的活跃事务视图中查看项目中的所有活跃事务。

已存档事务

已存档事务是指与您的项目不再相关的事务。例如，在以下情况下，您可以存档事务：事务已完成且不再需要显示在完成列中，或者错误地创建了事务。如果需要，可以取消存档已存档的事务。

被分派人

被分派人是指将事务分配给的人员。如果您在搜索被分派人时，他们未显示在列表中，则表示尚未将他们添加到项目中。要添加他们，请参阅邀请用户加入项目。要允许将一个事务分配给多个被分派人，请参阅启用或禁用多个被分派人。具有多个被分派人的事务将显示在面板上，并将显示不同的彩色头像，每个头像代表一个被分派人。

自定义字段

自定义字段允许您根据需求自定义事务的不同属性，以便跟踪和管理项目中的事务。例如，您可以添加用于路线图的字段、特定的到期日期或请求者字段。

Estimate

在敏捷开发中，估算被称为故事点。除了事务的模糊性和复杂性之外，您还可以使用事务的估算来表示所需的工作量。对于具有更大的风险、难度和未知性的事务，可以考虑使用更高的估算。

有关估算类型及其配置方法的更多信息，请参阅配置事务工作量估算。

事务

事务是跟踪与项目相关的工作的记录。您可以为功能、任务、错误或任何其他与项目相关的工作主体创建事务。如果您使用的是敏捷开发，一个事务还可以描述一个长篇故事或用户故事。

标签

标签用于对事务进行分组、排序和筛选。您可以输入新的标签名，也可以从填充的列表中选择一个标签。此列表包含项目中最近使用的标签。一个事务可以有多个标签，并且可以从事务中移除标签。要自定义标签，请参阅使用标签对工作进行分类。

优先级

优先级是指事务的重要程度。有四个选项：低、中、高和无优先级。

状态和状态类别

状态是事务的当前状态，可用于快速检查事务从开始到完成的整个生命周期内的进度。所有事务都必须具有一个状态，并且每个状态都属于一个状态类别。状态类别用于帮助整理状态和填充默认事务视图。

中有五种默认状态和四种状态类别。 CodeCatalyst您可以创建其他状态，但无法创建其他状态类别。以下列表包含了默认状态及其状态类别（括号中的内容）：待办事项(草稿)、待办事项(未开始)、进行中(已开始)、审核中(已开始) 和完成(已完成)。

有关处理状态的更多信息，请参阅使用自定义状态跟踪工作。

任务

可以向事务添加任务，以进一步细分和整理事务的工作。您可以在创建事务时向事务添加任务，也可以向现有事务添加任务。查看事务时，您可以重新排序、移除其任务或将其任务标记为已完成。

观点

CodeCatalyst 项目中的问题显示在视图中。视图可以是网格视图或面板视图，前者以列表格式显示事务，后者将事务显示为按事务状态整理的列中的磁贴。有四个默认视图，您可以利用自定义分组、筛选和排序来创建自己的视图。以下列表包含有关四个默认视图的详细信息。

- 草稿视图是一个网格视图，将显示当前未处理的事务。任何创建的具有草稿状态类别中的状态的事务都会显示在此视图中。团队可以使用此视图来查看哪些事务仍在定义中，或者哪些事务正在等待分配和处理。

- 活跃事务视图是当前正在处理的所有事务的面板视图。任何具有未开始、已开始或已完成状态类别中的状态的事务都将显示在此视图中。
- 所有事务视图是一个网格视图，将显示项目中的所有事务，包括草稿和活跃事务。
- 已存档视图显示所有已存档的事务。

跟踪有关事务的工作

您可以使用事务来计划和跟踪您的项目工作。每个事务均为保存在不同的记录中的一项工作。可以将事务细分成多个任务，以进一步整理和跟踪事务的工作。您还可以在事务之间创建链接以帮助跟踪相关工作，创建标签以帮助对工作进行整理和分类，对事务进行分组，为工作分配优先级，并指明工作是否被阻止。

当您准备好处理一个或一组事务时，可以估算工作，将其分配给用户，并添加评论以帮助其他人了解工作及其进度。您还可以导出事务以帮助您将事务包含的信息转换为其他格式。

在中创建问题 CodeCatalyst

开发团队创建事务来帮助跟踪和管理他们的工作。您可以根据您的需求在项目中创建事务。例如，您可以创建事务来更新代码中的变量。您可以将事务分配给项目中的其他用户，使用标签来帮助您跟踪工作等。

按照以下说明在中创建问题 CodeCatalyst。

创建事务

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建事务的项目。
3. 在项目主页上，选择创建事务。或者，在导航窗格中，选择事务。
4. 选择创建事务。

> **Note**
>
> 在使用网格视图时，也可以内联方式添加事务。

5. 输入事务的标题。
6. （可选）输入描述。您可以使用 Markdown 添加格式。
7. （可选）选择事务的状态、优先级和估算。

> **Note**
>
> 如果项目的估算设置设为隐藏估算，则不会显示估算字段。

8. （可选）向事务添加任务。任务可用于将事务的工作分成多个小目标。要添加任务，请选择 + 添加任务。然后，在文本字段中输入任务名称并按 Enter。添加任务后，您可以通过选中复选框将任务标记为已完成，或者通过从复选框左侧选择并拖动任务来重新为任务排序。
9. （可选）添加现有标签或创建一个新标签并通过选择 + 添加标签来添加该标签。
   a. 要添加现有标签，请从列表中选择该标签。您可以在字段中输入一个搜索词来搜索项目中所有包含该搜索词的标签。
   b. 要创建并添加新标签，请在搜索字段中输入要创建的标签的名称，然后按 Enter。
10. （可选）通过选择 + 添加被分派人来添加被分派人。您可以通过选择 + 添加我来快速将自己添加为被分派人。

> **Tip**
>
> 您可以选择将事务分配给 Amazon Q，让 Amazon Q 尝试解决该事务。有关更多信息，请参阅教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作。此功能仅在美国西部（俄勒冈州）区域中提供。
> 此功能要求为空间启用生成式人工智能功能。有关更多信息，请参阅 Managing generative AI features。

11. （可选）添加现有的自定义字段或创建新的自定义字段。事务可具有多个自定义字段。
    a. 要添加现有的自定义字段，请从列表中选择该自定义字段。您可以在字段中输入一个搜索词来搜索项目中所有包含该搜索词的自定义字段。
    b. 要创建并添加新的自定义字段，请在搜索字段中输入要创建的自定义字段的名称，然后按 Enter。然后选择要创建的自定义字段的类型并设置一个值。
12. 选择创建事务。右下角将显示一条通知：如果已成功创建事务，则会显示一条确认消息，指明已成功创建事务。如果事务创建失败，则会显示一条错误消息，指明失败的原因。之后，您可以选择重试以进行编辑并重试创建事务，或选择放弃以放弃事务。选择这两个选项都将关闭通知。

> **Note**
>
> 在创建一个事务时，无法将拉取请求链接到该事务。不过，您可以在创建事务后编辑它，以添加拉取请求的链接。

创建和处理分配给 Amazon Q 的事务时的最佳实践

在创建事务时，有时一些事务会持续存在。导致出现这种情况的原因可能是复杂而多变的。有时是因为不知道谁将处理事务。其他时候，事务需要对代码库的特定部分进行研究或有专门的了解，而这项工作的最佳候选人却忙于处理其他事务。通常还需优先处理其他紧急工作。这些原因中的任何一个或全部都可能导致无法解决的问题。 CodeCatalyst 包括与名为 Amazon Q 的生成式 AI 助手的集成，该助手可以根据问题的标题和描述来分析问题。如果您将事务分配给 Amazon Q，它将尝试创建解决方案草稿以供您评估。这有助于您和您的团队专注于并优化处理需要关注的事务，而 Amazon Q 则专注于解决您没有资源来立即解决的问题。

> **Note**
>
> > **Note**
> >
> > 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于为我编写描述、创建内容摘要、推荐任务、使用 Amazon Q 创建功能或将功添加到项目以及将事务分配给 Amazon Q 功能与用于软件开发的 Amazon Q 开发者版代理程序的功能都是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施来强制实施安全性并负责任地使用人工智能（AI）。

Amazon Q 能够出色地处理简单的事务和问题。为了获得最佳结果，请使用简明易懂的语言来清楚地说明您要执行的操作。以下是一些最佳实践，可帮助您创建优化的事务以供 Amazon Q 处理。

> **Important**
>
> 生成式人工智能功能仅在美国西部（俄勒冈州）区域中可用。

- 保持简单。Amazon Q 能够出色地处理简单的代码更改和修复，可以在事务的标题和描述中说明这些更改和修复。请勿分配具有模糊的标题或过于华丽或相互矛盾的描述的事务。
- 具体化。您提供的有关解决事务所需的确切更改的信息越多，Amazon Q 创建出解决该事务的解决方案的可能性就越高。如果可能，请包括具体的细节，例如要更改的 APIs 名称、要更新的方法、需要更改的测试以及您能想到的任何其他细节。
- 在将事务分配给 Amazon Q 之前，请确保事务的标题和描述中包含所有详细信息。在将事务分配给 Amazon Q 之后，无法更改其标题或描述，因此，在将事务分配给 Amazon Q 之前，请确保您拥有事务中所需的所有信息。
- 仅分配需要在单一源存储库中更改代码的事务。Amazon Q 只能在中处理单源存储库中的代码 CodeCatalyst。不支持链接的存储库。在将事务分配给 Amazon Q 之前，请确保该事务只需要在单一源存储库中进行更改。
- 使用 Amazon Q 建议的默认值来批准每个步骤。默认情况下，您需要审批 Amazon Q 执行的每个步骤。这使您能够针对事务评论以及 Amazon Q 创建的任何拉取请求与之进行交互。这将提供与 Amazon Q 的交互性更高的体验，帮助您调整其方法并优化它为解决事务而创建的代码。

> **Note**
>
> Amazon Q 不会响应事务或拉取请求中的单个评论，但它在需要重新考虑其方法或创建修订时将审查这些评论。

- 请始终仔细审查 Amazon Q 建议的方法。在您批准其方法后，Amazon Q 将开始根据该方法生成代码。在告知 Amazon Q 继续操作之前，请确保该方法看起来正确并包含您期望的所有详细信息。
- 请确保仅在以下情况下允许 Amazon Q 使用工作流：您没有可在审核方法前部署方法的现有工作流。您的项目可能已将工作流配置为在拉取请求事件上开始运行。如果是这样的话，Amazon Q 创建的任何拉取请求（包括创建或更新工作流 YAML）都可能启动拉取请求中包含的那些工作流的运行。作为最佳实践，请不要选择让 Amazon Q 能够使用工作流文件，除非您确定项目中没有将自动运行这些工作流文件的工作流，然后再审核和批准它创建的拉取请求。

有关更多信息，请参阅教程：使用 CodeCatalyst 生成式 AI 功能加快开发工作和 Managing generative AI features。

估算事务

在敏捷开发中，估算被称为故事点。除了事务的模糊性和复杂性之外，您还可以使用事务的估算来表示所需的工作量。对于具有更大的风险、难度和未知性的事务，可以考虑使用更高的估算。

您必须先选择要用于项目的估算的类型，之后才能开始估算事务。默认情况下，提供了两种类型以供选择。要有效地使用 T 恤尺寸或斐波纳契数列，您的团队必须根据每种尺寸所代表的项目做出调整。共同决定每个估算的含义，然后开始将这些估算应用于每个事务。考虑定期审查

按照以下步骤配置中问题的工作量估算设置。 CodeCatalyst

为事务配置工作量估算

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在基本设置部分中的估算中，选择估算值的显示方式。可用的估算类型有 T 恤尺寸、斐波纳契数列或隐藏估算。如果项目的估算设置设为隐藏估算，则项目的事务中不会显示估算字段。

   更新估算类型后，不会丢失任何数据，并且将自动转换所有事务的估算值。转换映射如下表所示。

T 恤尺寸	斐波纳契数列
XS	1
XS	2
S	3
M	5
L	8
XL	13

要添加或更改事务的估算，您可以编辑事务。

编辑和协作处理中的问题 CodeCatalyst

目录

- 将一个事务链接到另一个事务
- 将事务标记为已屏蔽或已取消屏蔽
- 添加、编辑或删除评论
  - 在评论中使用提及

编辑事务

按照以下步骤操作，编辑事务的标题、描述、状态、被分派人、优先级、估算或标签。

编辑事务

1. 选择要编辑的事务以查看事务详细信息。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 要编辑事务标题，请选择该标题，输入新的标题，然后按 Enter。
3. 要编辑描述，请选择描述，输入新的描述，然后按 Enter。您可以使用 Markdown 添加格式。
4. 在任务中，您可以查看和管理事务的任务。如果没有任务，您可以让 Amazon Q 分析事务并推荐一些任务，这些任务可以将事务中的工作细分成单独的项目，每个项目均可分配给一个用户。有关更多信息，请参阅管理有关事务的任务。
5. 要编辑状态、估算或优先级，请从相应的下拉菜单中选择一个选项。
6. 在标签中，您可以添加现有标签、创建新标签或移除标签。
   a. 要添加现有标签，请选择 + 添加标签，然后从列表中选择该标签。您可以在字段中输入一个搜索词来搜索项目中所有包含该搜索词的标签。
   b. 要创建并添加新标签，请选择 + 添加标签，在搜索字段中输入要创建的标签的名称，然后按 Enter。
   c. 要移除标签，请选择要移除的标签旁边的 X 图标。如果您从所有事务中移除一个标签，该标签将显示在事务设置的标签部分的未使用标签部分中。在使用筛选条件或向事务添加标签时，未使用的标签将显示在标签列表的末尾。您可以在事务设置中找到所有标签（已使用和未使用）以及包含这些标签的事务的概述。
7. 要分配事务，请在被分派人部分中选择 + 添加被分派人，然后从列表中搜索并选择被分派人。您可以选择 + 添加我来快速将自己添加为被分派人。
8. 在附件中，您可以添加、下载或移除附件。有关更多信息，请参阅 处理附件。
9. 要链接拉取请求，请选择链接拉取请求，然后从列表中选择拉取请求或输入拉取请求的 URL 或 ID。要取消链接拉取请求，请选择取消链接图标。

> **Tip**
>
> 在向事务添加拉取请求的链接后，您可以通过在已链接的拉取请求列表中选择事务 ID 来快速导航到事务。您可以使用拉取请求的 URL 来链接与事务面板不同的项目中的拉取请求，但只有作为该项目成员的用户才能查看或导航到该拉取请求。

10. （可选）添加和设置现有的自定义字段、创建新的自定义字段或移除自定义字段。事务可具有多个自定义字段。

    a. 要添加现有的自定义字段，请从列表中选择该自定义字段。您可以在字段中输入一个搜索词来搜索项目中所有包含该搜索词的自定义字段。

    b. 要创建并添加新的自定义字段，请在搜索字段中输入要创建的自定义字段的名称，然后按 Enter。然后选择要创建的自定义字段的类型并设置一个值。

    c. 要移除自定义字段，请选择要移除的自定义字段旁边的 X 图标。如果从所有事务中移除一个自定义字段，则该自定义字段将被删除且不再在筛选时显示。

处理附件

您可以在中为议题添加附件 CodeCatalyst ，以便于访问相关文件。使用以下过程管理事务的附件。

添加到事务的附件大小将计入您空间的存储配额。有关查看和管理项目附件的信息，请参阅查看和管理附件。

> **Important**
>
> Amazon 不会扫描或分析问题的附件 CodeCatalyst。任何用户都可以向可能包含恶意代码或内容的事务添加附件。确保用户了解有关管理附件和防范恶意代码、内容或病毒的最佳实践。

添加、下载或移除附件

1. 选择要管理其附件的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 要添加附件，请选择上传文件。在操作系统的文件资源管理器中浏览找到该文件并将其选中。选择打开以将其添加为附件。有关配额信息（例如最大附件大小），请参阅 中的问题配额 CodeCatalyst。

   请记下以下针对附件文件名和内容类型的限制：

- 文件名不得包含以下字符：
  - 控制字符：0x00-0x1f 和 0x80-0x9f
  - 保留字符：/、?、<、>、\、:、*、| 和 "
  - Unix 保留文件名：. 和 ..
  - 尾随句点和空格
  - Windows 保留文件名：CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9
- 附件的内容类型必须符合以下媒体类型模式：

```
media-type = type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

例如 text/html; charset=UTF-8。

3. 要下载附件，请选择要下载的附件旁边的省略号菜单，然后选择下载。
4. 要复制附件的 URL，请选择要复制其 URL 的附件旁边的省略号菜单，然后选择复制 URL。
5. 要移除附件，请选择要移除的附件旁边的省略号菜单，然后选择删除。

查看和管理附件

您可以在事务设置中查看一个表，其中包含已添加到项目中的事务的每个附件。此表包含每个附件的详细信息，包括内容类型、添加时间、添加到的事务及其状态以及文件大小等信息。

此表可用于轻松识别已完成或已存档的事务的大型附件，以将其移除来释放空间存储。

> **Important**
>
> Amazon 不会扫描或分析问题的附件 CodeCatalyst。任何用户都可以向可能包含恶意代码或内容的事务添加附件。确保用户了解有关管理附件和防范恶意代码、内容或病毒的最佳实践。

查看和管理项目中的所有事务附件

1. 在导航窗格中，选择事务。
2. 选择省略号图标并选择设置。
3. 选择分配选项卡。

管理有关事务的任务

可以向事务添加任务，以进一步细分、组织和跟踪该事务的处理情况。您可以自行创建任务，也可以使用 Amazon Q 根据对事务的分析以及事务的复杂性来推荐任务。

Amazon Q Developer 是一款基于人工智能的生成式对话助手，可以帮助您理解、构建、扩展和操作 AWS 应用程序。为了加快您的构建 AWS，为Amazon Q提供支持的模型增加了高质量的 AWS 内容，以生成更完整、更具可操作性和参考性的答案。有关更多信息，请参阅《Amazon Q Developer User Guide》中的 What is Amazon Q Developer?。

管理有关事务的任务

1. 选择要管理其任务的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 在任务中，您可以查看和管理事务的任务。

    1. 要添加任务，请在文本字段中输入任务名称并按 Enter。
    2. 如果事务没有任务，您可以选择让 Amazon Q 分析事务并根据事务标题、描述及其对事务复杂性的分析和存储库代码来创建任务，然后选择推荐任务。您将需要指定包含事务代码的源存储库。选择开始推荐任务以开始任务推荐分析。该对话框将关闭。推荐完成后，在选择创建任务之前，选择查看推荐的任务以查看任务并执行任何所需操作，例如删除任务或将任务添加到列表或重新对推荐的任务进行排序。

        在系统为您创建任务后，您可以将任务分配给用户，并像处理手动创建的任务一样使用这些任务。
    3. 要将某个任务标记为已完成，请选中该任务的复选框。
    4. 要查看或更新某个任务的详细信息，请从列表中选择该任务。
    5. 要重新对任务进行排序，请选择任务并从复选框的左侧拖动它。
    6. 要移除某个任务，请选择该任务的省略号菜单，然后选择移除。

将一个事务链接到另一个事务

如果某个事务与另一个事务中的工作相关，则可以将这两个事务链接起来。这是一种可帮助您快速了解和跟踪工作项之间的依赖关系的方法。

在链接事务时，有四种状态可供选择：

- 相关：使用此状态可表示与链接的事务相关的工作。
- 被屏蔽：使用此状态可表示事务已被链接的事务屏蔽。

- 屏蔽：使用此状态可表示事务屏蔽了已链接事务的进度。
- 重复：使用此状态可表示事务与在另一个事务中捕获的工作重复。

可以在创建链接后更改其状态。对于在其中创建链接的事务和已链接的事务，链接及其链接状态显示在已链接的事务中。您也可以在链接创建后将其移除。

将一个事务链接到另一个事务或任务

1. 打开要链接到另一个事务的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 选择链接事务。
3. 在标记为中，选择链接的状态。

   在事务编号中，输入事务的编号，或从下拉列表中选择该编号。
4. 要添加其他链接，请选择添加链接的事务。
5. 完成后，选择更新以更新该事务以及具有链接关系的所有已链接事务。

将事务标记为已屏蔽或已取消屏蔽

如果您在处理事务时受阻，则可能需要将事务标记为已屏蔽。例如，如果您的事务依赖于对代码库中尚未合并的另一部分所做的更改，则事务可能会被屏蔽。

当您将议题标记为已屏蔽时， CodeCatalyst 会在议题上添加红色的 “已屏蔽” 标签，使其在待办事项列表或存档中或图板上高度可见。

在解决外部情况后，可以取消对事务的屏蔽。

将事务标记为已屏蔽

1. 打开要标记为已屏蔽的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 选择操作，然后选择标记为已屏蔽。

取消对事务的屏蔽

1. 打开要取消屏蔽的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 选择操作，然后选择标记为已取消屏蔽。

添加、编辑或删除评论

您可以发表对事务的评论。在评论中，您可以标记其他空间成员、空间中的其他项目、相关事务和代码。

向事务添加评论

1. 导航到您的项目。
2. 在导航栏中，选择事务。
3. 选择要向其添加评论的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
4. 在评论字段中输入评论。您可以使用 Markdown 添加格式。
5. 选择发送。

编辑评论

您可以编辑对事务发表的评论。您只能编辑自己撰写的评论。

1. 导航到您的项目。
2. 在导航栏中，选择事务。
3. 选择要在其中编辑评论的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
4. 要编辑评论，请找到要编辑的评论。

> **Tip**
>
> 您可以按从最旧到最新或从最新到最旧的顺序对评论排序。一次加载 10 条评论。

5. 选择省略号图标，然后选择编辑。
6. 编辑评论。您可以使用 Markdown 添加格式。
7. 选择保存。此时已更新评论。

删除评论

您可以删除对事务发表的评论。您只能删除自己撰写的评论。删除评论后，将显示您的用户名，但此评论已被删除将替代原始评论文本。

1. 导航到您的项目。

2. 在导航栏中，选择事务。
3. 选择要从中删除评论的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
4. 选择省略号图标，再选择删除，然后选择确认。

在评论中使用提及

您可以在评论中提及空间成员、空间中的其他项目、相关事务和代码。执行此操作将创建一个指向提及的用户或资源的快速链接。

在评论中使用 @mention

1. 导航到您的项目。
2. 在导航栏中，选择事务。
3. 选择要编辑的事务以查看事务详细信息。要获取有关查找事务的帮助，请参阅查找和查看事务。
4. 选择添加评论文本框。
5. 键入 @*user_name* 以提及其他用户。
6. 键入 @*project_name* 以提及项目。
7. 键入 @*issue_name* 或 @*issue_number* 以提及其他事务。
8. 键入 @*file_name* 以提及源存储库中的特定文件或代码。

> **Note**
>
> 在您键入时，将填充包含与您的 @mention 匹配的词的前 5 个项目（用户、源存储库、项目等）的列表。

9. 选择要提及的所需项目。将在评论文本框中填充显示项目所在位置的路径。
10. 完成评论并选择发送。

查找和查看事务

以下各节介绍如何有效地搜索和查看 CodeCatalyst 项目中的问题。

搜索事务

您可以通过搜索特定参数来查找事务。有关优化搜索的更多信息，请参阅在 CodeCatalyst 中搜索代码、事务、项目和用户。

搜索事务

1. 导航到您的项目。
2. 使用搜索栏搜索事务或与事务相关的信息。您可以使用查询参数来优化搜索。有关更多信息，请参阅 在 CodeCatalyst 中搜索代码、事务、项目和用户。

对事务进行排序

默认情况下，中的问题 CodeCatalyst 按手动顺序排序。手动顺序将按用户移动事务的顺序显示事务。按手动顺序排序时，您可以拖放事务来更改其顺序。此排序选项在整理事务待办事项和确定事务的优先级时非常有用。

下表说明了如何在网格视图和面板视图中对事务进行排序。

网格视图排序选项	面板视图排序选项
手动顺序	手动顺序
上次更新时间	上次更新时间
优先级	优先级
Estimate	Estimate
标题	标题
ID	
状态	
阻止	
自定义字段	

使用以下过程可更改事务的排序方式。

对事务进行排序

1. 导航到您的项目。

2. 在导航窗格中，选择事务。默认视图为面板。
3. （可选）选择活跃事务以打开事务视图切换器下拉菜单，导航到其他事务视图。
4. 要对网格视图进行排序，可使用以下两个选项：

   a. 选择要作为排序依据的字段的标题。选择标题将在升序顺序和降序顺序之间循环。

   b. 选择排序依据下拉菜单并选择要作为排序依据的参数。事务将按升序顺序进行排序。
5. 要对面板视图进行排序，请选择排序依据下拉菜单并选择要作为排序依据的参数。事务将按升序顺序进行排序。

对事务进行分组

分组用于按多个参数（例如被分派人、标签和优先级）来整理面板上的事务。

对事务进行分组

1. 导航到您的项目。
2. 在导航窗格中，选择事务。默认视图为面板。
3. （可选）选择活跃事务以打开事务视图切换器下拉菜单，导航到其他事务视图。
4. 选择分组。
5. 在分组依据中，选择作为分组依据的参数：

   - 如果您选择被分派人或优先级，请选择分组顺序。
   - 如果选择标签，请选择标签，然后选择分组顺序。
6. （可选）选择显示空组开关以显示或隐藏当前未分配任何事务的组。
7. 在您做出选择时，视图将更新。事务仅显示在与配置的参数匹配的组中。

筛选事务

使用筛选功能查找包含指定名称、优先级、标签、自定义字段或被分派人的事务。

筛选事务

1. 导航到您的项目。
2. 在导航窗格中，选择事务。
3. （可选）选择活跃事务以打开事务视图切换器下拉菜单，导航到其他事务视图。

> **Note**
>
> 要根据事务名称或描述中的字符串进行筛选，请在事务搜索栏中输入该字符串。

4. 选择筛选条件，然后选择 + 添加筛选条件。
5. 选择用于筛选的参数。您可以选择多个筛选条件和参数。您可以通过选择 and 或 or 来配置筛选条件，以显示符合每个筛选条件或任一筛选条件的事务。视图将更新以显示符合筛选条件的事务。

使事务取得进展

每个事务都有生命周期。在中 CodeCatalyst，问题通常以待办事项列表中的草稿形式开始。在要开始处理事务时，事务会被移到另一个状态类别中，并在各种状态之间移动直到事务完成，之后将对事务进行存档。您可以通过以下方式在事务的整个生命周期中移动事务或使事务取得进展：

- 您可以在待办事项和面板之间移动事务。
- 您可以将进行中的事务移至不同的完成阶段。
- 您可以对已完成的事务进行存档。

在待办事项和面板之间移动事务

在开始处理事务后，您可以将事务从待办事项移至面板。如果工作被推迟，您也可以将事务移回待办事项中。

在待办事项和面板之间移动事务

1. 导航到您的项目。
2. 在导航窗格中，选择事务。默认视图为面板。
3. 将事务从面板移至待办事项：
   a. 选择要移动的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
   b. 从状态下拉菜单中选择待办事项。
4. 将事务从待办事项移至面板：
   a. 要导航到待办事项，请选择面板，然后选择待办事项。
   b. 选择要移动的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
   c. 选择添加到面板，或选择待办事项以外的状态。

在面板上使事务在各个生命周期阶段取得进展。

您可以将面板中的事务移至不同的状态，直到事务完成。

在面板中移动事务

1. 在导航窗格中，选择事务。默认视图为面板。
2. 请执行以下操作之一：
   - 将事务拖放到其他状态。
   - 选择一个事务，然后从状态下拉菜单中选择一个状态。
   - 选择一个问题，然后选择移至：***next-status***。

有关对事务进行存档的信息，请参阅对事务进行存档。

在组之间移动事务

您可以按各种参数在所有事务和面板视图中对事务进行分组。如果事务已分组，则可以将事务从一个组移至另一个组。将事务从一个组移至另一个组将自动编辑作为事务分组依据的字段，使其与目标相匹配。

举个例子，假设有一家公司在使用 CodeCatalyst ，将问题分配给了两个人，即王秀兰和萨恩维·萨卡。面板会按 `Assignee` 分成两组，一个组对应于一个被分派人。在将事务从 Wang Xiulan 组移至 Saanvi Sarkar 组后，事务的最新被分派人将更新为 Saanvi Sarkar。

对事务进行存档

> **Note**
>
> 不会在项目中删除事务，而是对事务进行存档。要删除事务，您必须删除项目。

如果项目中不再需要某个事务，则可以对该事务进行存档。存档议题时，将其从所有筛选出已存档议题的视图中 CodeCatalyst 移除。可以在已存档的事务默认视图中查看已存档的事务，如果需要，可以在该视图中取消存档事务。

在以下情况下，可以对事务进行存档：

- 您已完成事务，并且不再需要它显示在完成列中。
- 您未计划处理事务。
- 您错误地创建了事务。
- 您已达到最大活跃事务数。

对事务进行存档

1. 打开要存档的事务。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 选择操作，然后选择移动到存档。
3. （可选）要快速对多个具有已完成状态的事务进行存档，请选择面板上任何已完成状态顶部的垂直省略号，然后选择存档事务。

取消存档事务

1. 打开要取消存档的事务。您可以通过从事务视图切换器下拉菜单中打开已存档事务视图来查看已存档事务列表。要获取有关查找事务的帮助，请参阅查找和查看事务。
2. 选择取消存档。

导出事务

您可以将当前视图中的事务导出到一个 .xlsx 文件中。要导出事务，请执行以下步骤。

导出事务

1. 导航到您的项目。
2. 在导航栏中，选择事务。
3. 选择活跃事务以打开事务视图切换器下拉菜单，然后导航到包含要导出的事务的视图。仅导出视图中显示的事务。
4. 选择省略号菜单，然后选择导出到 Excel。
5. 这将下载该 .xlsx 文件。默认情况下，该文件的标题包含项目名称和导出的完成日期。

使用待办事项、标签和面板整理工作

并非所有团队的工作方式都相同。您可以配置问题的显示方式以及在 Amazon 中分配问题的方式，CodeCatalyst 以帮助您准确了解正在处理的内容和工作状态。您可以选择可对事务使用的估算方法，

以便您的用户都使用相同的估算方法。您可以创建自定义标签和状态，这些标签和状态也可用于筛选工作视图。根据团队的工作方式，您可以配置是允许一个事务有多个被分派人，还是只允许将一个事务分配给一个用户。您还可以创建事务的自定义视图，以便通过向您或您的团队显示最相关信息的方式来显示工作。

使用标签对工作进行分类

您可以为事务自定义标签。这包括编辑标签和更改颜色。标签可帮助您对工作进行分类和整理。例如，您可以为软件的特定方面创建标签，也可以为不同的组或团队创建标签。

主题

创建标签

在中 CodeCatalyst，您可以通过在创建新议题或编辑现有议题时添加标签来创建标签。有关更多信息，请参阅在中创建问题 CodeCatalyst和编辑和协作处理中的问题 CodeCatalyst。

编辑标签

使用以下过程可更改现有标签的名称或颜色。

编辑标签

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 标签磁贴上有一个项目中使用的标签的列表。选择要编辑的标签旁边的编辑图标。执行以下一个或多个操作：

    a. 编辑标签的名称。

    b. 要更改颜色，请选择色轮。使用选取器选择一种新颜色。
4. 要保存对标签所做的更改，请选中复选标记图标。
5. 更改后的标签现在会显示在可用标签列表中。您还可以查看正在使用该标签的事务的数量。

> **Note**
>
> 您可以选择每个标签旁边显示的数字，以导航到所有事务页面并查看所有包含该标签的事务。

删除标签

您目前无法在中删除问题标签 CodeCatalyst。如果您从所有事务中移除一个标签，该标签将显示在事务设置的标签部分的未使用标签部分中。在使用筛选条件或向事务添加标签时，未使用的标签将显示在标签列表的末尾。您可以在事务设置中找到所有标签（已使用和未使用）以及包含这些标签的事务的概述。

使用自定义字段整理工作

您可以创建自定义字段来帮助整理和查看项目的工作。自定义字段将添加到筛选条件中的可用筛选条件列表中，因此您可以按自定义字段筛选事务。自定义字段是名称/值对。您可以按自定义字段的名称进行筛选，然后按自定义字段的值进行筛选。

一个事务可具有多个自定义字段。

创建自定义字段

在中 CodeCatalyst，您可以通过在创建议题或编辑现有议题时添加自定义字段来创建自定义字段。有关更多信息，请参阅在中创建问题 CodeCatalyst和编辑和协作处理中的问题 CodeCatalyst。

删除自定义字段

要删除某个自定义字段，您必须从该自定义字段已添加到的每个事务中移除该自定义字段。删除自定义字段后，筛选条件中将不再显示该自定义字段。您可以使用筛选条件查看带自定义字段的所有事务，并通过编辑事务来将其移除。有关更多信息，请参阅 查找和查看事务 和 编辑事务

使用自定义状态跟踪工作

您可以在面板上添加自定义状态。每个自定义状态都必须属于下列类别之一：草稿、未开始、已开始或已完成。状态类别用于帮助整理状态和填充默认视图。有关状态和状态类别的更多信息，请参阅状态和状态类别；有关视图的更多信息，请参阅查找和查看事务。

创建状态

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在状态中，选择您希望状态所处的类别旁边的加号图标。
4. 为状态命名，然后选择复选标记图标。

> **Note**
>
> 选择 X 图标可取消添加状态。

现在，自定义状态将显示在您的面板上，并在创建事务时显示为一个选项。

编辑状态

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在状态中，选择要编辑或更改的状态旁边的编辑图标。
4. 编辑状态，然后选中复选标记图标。

编辑后的状态现在会显示在面板上。

移动状态

1. 在导航窗格中，选择事务。
2. 选择省略号图标并选择设置。
3. 在状态中，选择要移动的状态。
4. 将该状态拖放到所需位置。

> **Note**
>
> 您只能在状态的指定类别中移动状态。

现在，已在面板上重新对状态排序。

停用状态

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在状态中，选择要停用的状态。
4. 在要停用的状态上，选择该状态的开关。此时状态将灰显。

> **Note**
>
> 已停用的状态将显示在面板上，直到从中移除所有事务。不能将事务添加到已停用的状态。

5. 要重新激活已停用的状态，请选择该状态的开关。该状态将不再灰显。

> **Note**
>
> 每个类别必须具有至少一种活跃的状态。如果类别中只有一种状态，则无法停用该状态。

配置事务工作量估算

按照以下步骤配置中问题的工作量估算设置。 CodeCatalyst

为事务配置工作量估算

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在基本设置部分中的估算中，选择估算值的显示方式。可用的估算类型有 T 恤尺寸、斐波纳契数列或隐藏估算。更新估算类型后，不会丢失任何数据，并且将自动转换所有事务的估算值。转换映射如下表所示。

T 恤尺寸	斐波纳契数列
XS	1
XS	2

T 恤尺寸	斐波纳契数列
S	3
M	5
L	8
XL	13

启用或禁用多个被分派人

按照以下步骤为中问题的多个受托人配置设置。 CodeCatalyst

启用或禁用多个被分派人

1. 在导航窗格中，选择事务。
2. 选择活跃事务以打开事务视图切换器下拉菜单，然后选择设置。
3. 在基本设置部分中的被分派人中，切换指示器以允许将多个被分派人分配给同一事务。一个事务最多可以有 10 个被分派人。如果您不启用此选项，则只能向一个事务分配一个被分派人。

创建事务视图

您可以创建视图来快速查看符合一组特定的筛选条件的事务。这有助于您节省时间并快速查看之前已筛选、分组或排序的事务。

创建事务视图

1. 在导航窗格中，选择事务。
2. （可选）根据您的应用场景，您可能需要从现有视图创建视图。要导航到其他视图，请选择活跃事务以打开事务视图切换器下拉菜单并选择视图。
3. （可选）在创建视图之前，请配置筛选条件、分组和排序。您可以在创建视图时添加这些项，但如果您提前这样做，则可以在创建视图之前预览视图中显示的内容。
4. 从标题栏中打开事务视图切换器下拉菜单。要创建面板视图（其中根据状态在列中查看事务），请在面板列中选择 +。要创建在列表中查看事务的网格视图，请在网格列中选择 +。如果您改变主意，可以在创建视图前更改其类型。
5. 在创建视图对话框中，输入视图的名称。

6. 这将根据当前视图的设置填充筛选条件、事务分组依据和事务排序依据字段。如果需要，可以更新这些字段。
7. 选择创建视图以创建视图并切换到该视图。

中的问题配额 CodeCatalyst

下表描述了 Amazon 中问题的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额...

资源	默认配额
活跃事务	每个项目最多 1000 个。
附件大小	每个附件的最多 500MB。 最大总附件存储空间受空间的总体存储限制的影响。有关更多信息，请参阅定价。
事务总数（活跃和已存档）	每个项目最多 100000 个。
保存的视图	每个项目最多保存 50 个保存的事务视图。
可链接到一个事务的拉取请求的数量	每个事务最多 50 个拉取请求。
状态（每个项目）	每个项目最多 50 个。
状态（每个事务）	每个事务最多 50 个。
标签（每个项目）	每个项目最多 200 个。
标签（每个事务）	每个事务最多 50 个。
自定义字段（每个事务）	每个事务最多 50 个。
被分派人	每个事务最多 10 个。
评论	每个事务最多 1000 个。
任务	每个事务最多 100 个。

在中配置身份、权限和访问权限 CodeCatalyst

首次登录 Amazon CodeCatalyst 时，您就创建了一个 AWS 建筑商 ID。 AWS 中 IDs 不存在生成器 AWS Identity and Access Management。您在首次登录时选择的用户名将成为您身份的唯一用户 ID。

在中 CodeCatalyst，您可以通过以下两种方式之一进行首次登录：

- 在创建空间的过程中。
- 作为接受项目或空间邀请的一部分 CodeCatalyst.

与您的身份关联的一个或多个角色决定了您可以执行的操作 CodeCatalyst。项目角色（例如项目管理员和贡献者）是特定于项目的，因此您可以在一个项目中拥有一个角色，并在另一个项目中拥有其他角色。如果您创建了空间，则 CodeCatalyst会自动为您分配空间管理员角色。当用户接受项目邀请时，CodeCatalyst 会将这些身份添加到空间并向他们分配受限访问角色。当您邀请用户加入项目时，可以选择您希望用户在项目中拥有的角色，这将决定用户可以和不可以在项目中执行的操作。参与项目的大多数用户只需要贡献者角色即可执行其任务。有关更多信息，请参阅 使用用户角色授予访问权限。

除了项目角色外，项目中的用户在使用 Git 客户端或集成开发环境时，还需要个人访问令牌 (PAT) 才能访问项目的源存储库 (IDEs)。项目成员可以在第三方应用程序中使用此 PAT 作为与其 CodeCatalyst 身份关联的应用程序专用密码。例如，当您将源存储库克隆到本地计算机时，必须提供 PAT 和 CodeCatalyst 用户名。

在工作流中部署操作时，您可以使用服务角色执行诸如访问 AWS CloudFormation 堆栈和资源之类的操作，从而配置和资源之间的 CodeCatalyst 访问权限。 AWS 您必须在 CodeCatalyst 和 AWS 资源之间配置访问权限，以便项目模板中包含的工作流操作才能运行。

主题

使用用户角色授予访问权限

在 Amazon 中 CodeCatalyst，您可以在项目级别和空间级别为用户分配角色。在项目中，角色指定让用户能够使用项目的资源在该项目中执行的操作。用户加入项目后即可获得空间中的成员资格。您可以添加或移除作为空间管理员的用户。空间管理员角色的权限是中所有角色中最广泛的。 CodeCatalyst 最佳实践是向用户分配执行任务所需的最低权限。

您可以在空间中向用户分配角色。您也可以在用户所属的项目中向用户分配角色。每个用户在一个项目或空间中只能具有一个角色，但用户可在每个项目和空间中具有不同的角色。例如，一个用户可在一个项目中具有项目管理员角色，并在另一个项目中具有贡献者角色。

主题

了解空间和项目的用户角色

空间有三个可用的角色：

- 空间管理员
- 高级用户
- 受限访问

接受项目邀请的用户将在包含项目的空间中自动获得受限访问角色。

项目中的成员有四个可用角色：

- 项目管理员
- 贡献者
- 审阅者
- 只读

当您将用户添加到项目时， CodeCatalyst 会自动为其授予受限访问权限。如果您从所有项目中移除某个用户，则 CodeCatalyst会自动删除该用户的受限访问角色。

空间管理员角色

空间管理员角色是中最强大的角色 CodeCatalyst。仅将空间管理员角色分配给需要管理空间各个方面的用户，因为该角色拥有中的所有权限 CodeCatalyst。具有空间管理员角色的用户是唯一可以在空间管理员角色中添加或移除其他用户以及删除空间的用户。

创建空间时， CodeCatalyst 会自动为您分配空间管理员角色。作为最佳实践，我们建议您将此角色添加给至少一个其他用户，该用户可以在原始空间创建者不可用时代入此角色。

高级用户角色

P ower user 角色是 CodeCatalyst空间中第二强大的角色，但它无法访问空间中的项目。该角色专为需要能够在空间中创建项目并帮助管理空间的用户和资源的用户而设计。将高级用户角色分配给作为团队领导或经理的用户，他们需要能够在工作中在空间中创建项目和管理用户。

受限的访问角色

“受限访问权限” 角色是大多数用户在 CodeCatalyst 空间中将扮演的角色。当用户接受空间中的项目邀请时，系统会自动将该角色分配给用户。该角色提供用户在包含项目的空间内工作所需的有限权限。将受限访问角色分配给您直接邀请加入空间的用户，除非他们的工作要求他们管理空间的某些方面。

项目管理员角色

项目管理员角色是 CodeCatalyst项目中最强大的角色。仅将此角色分配给需要管理项目的各个方面（包括编辑项目设置、管理项目权限和删除项目）的用户。

项目角色不具有任何空间级权限。因此，具有项目管理员角色的用户无法创建其他项目。仅拥有空间管理员或高级用户角色的用户能够创建项目。

> **Note**
>
> 空间管理员角色拥有中的所有权限 CodeCatalyst。

贡献者角色

“贡献者” 角色适用于 CodeCatalyst 项目中的大多数成员。将此角色分配给需要能够在项目中处理代码、工作流、事务和操作的用户。

审阅者角色

审阅者角色适用于需要能够与项目中的资源（例如拉取请求和议题）进行交互，但不能在项目中创建和合并代码、创建工作流或启动或停止工作流程运行的 CodeCatalyst 用户。将审阅者角色分配给需要能够批准和评论拉取请求、创建、更新、解决和评论事务，以及查看项目中的代码和工作流的用户。

只读角色

只读角色将分配给需要查看资源和资源状态，但不与之交互或直接参与项目的用户。具有此角色的用户无法在中创建资源 CodeCatalyst，但他们可以查看和复制资源，例如克隆存储库和将议题的附件下载到本地计算机。将只读角色分配给需要查看资源和项目状态但不直接与之交互的用户。

查看每个角色可用的权限

下表显示了每个 CodeCatalyst 角色的可用权限。使用下面的链接可跳转到相应的权限集。

- Space permissions
- Extensions permissions
- Project permissions
- Source repository permissions
- Dev Environment permissions
- Package repository and package permissions
- Workflow permissions
- Issues permissions
- Blueprint permissions
- Notifications permissions
- Search permissions

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
空间权限							
创建空间							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
编辑空间计费详细信息							
设置并启用单点登录							
移除单点登录							
为空间启用生成式人工智能功能							
为空间禁用生成式人工智能功能							
删除空间							
将其他用户添加到空间管理员角色							
从空间管理员角色中移除其他用户							
创建团队							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
删除团队							
更新团队							
为空间禁用机器资源							
为空间启用机器资源							
创建项目							
将 AWS 账户连接与空间关联							
更新 AWS 账户连接							
取消 AWS 账户连接与空间的关联							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
删除 AWS 账户连接并从空间中将其移除	✓	✓	✗	✗	✗	✗	✗
在空间中启用受项目限制的账户连接[1]	✓	✓	✗	✗	✗	✗	✗
在空间中禁用受项目限制的账户连接[2]	✓	✓	✗	✗	✗	✗	✗
邀请其他人加入空间	✓	✓	✗	✗	✗	✗	✗
创建 VPC 连接	✓	✓	✗	✗	✗	✗	✗
编辑 VPC 连接	✓	✓	✗	✗	✗	✗	✗
删除 VPC 连接	✓	✓	✗	✗	✗	✗	✗
查看空间中的活动日志	✓	✓	✗	✗	✗	✗	✗

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看 AWS 账户连接							
查看以下事件 CodeCatalyst							
查看空间							
查看团队							
查看 VPC 连接							

[1] 利用高级用户角色，您可以为账户启用项目限制，但只能为您所属的项目配置访问权限。

[2] 利用高级用户角色，您可以为账户禁用项目限制，但只能为您所属的项目配置访问权限。

扩展权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
安装扩展							
更新扩展							
删除扩展							
Connect 一个 GitHub 账户							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
断开 GitHub 账号							
连接 Jira 站点							
断开 Jira 站点的连接							
查看已安装扩展的配置详细信息							
查看扩展							

项目权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
编辑项目设置							
为项目禁用机器资源							
为项目启用机器资源							
删除项目							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
邀请用户加入项目							
更改项目中的用户角色							
从项目中移除用户							
向项目添加团队							
从项目中移除团队							
更改团队的项目角色							
查看项目							
查看项目活动							
查看项目中的团队							
查看蓝图							

源存储库权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建存储库							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
链接存储库							
取消链接存储库							
删除存储库							
编辑存储库设置							
查看存储库							
查看存储库设置							
克隆存储库							
创建分支							
创建分支规则							
更改默认分支							
删除分支							
合并分支							
更新分支规则							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看分支							
查看分支规则							
创建文件夹							
删除文件夹							
编辑文件夹							
查看文件夹							
创建文件							
删除文件							
编辑文件							
查看文件							
创建和推送提交							
查看提交							
创建拉取请求							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建拉取请求的审批规则							
覆盖拉取请求的合并要求							
更新拉取请求							
更新拉取请求的审批规则							
查看拉取请求							
查看拉取请求的审批规则							
关闭拉取请求							
批准拉取请求							
关于拉取请求的评论							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
针对拉取请求的评论与 Amazon Q 进行交互							
为 Amazon Q 创建的拉取请求创建修订							
将事务链接到拉取请求							
取消事务与拉取请求的链接							
开发环境权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建您自己的开发环境							
停止您自己的开发环境							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
停止其他用户创建的开发环境							
恢复您自己的开发环境							
查看您自己的开发环境							
查看其他用户创建的开发环境							
编辑您自己的开发环境							
编辑其他用户创建的开发环境							
删除您自己的开发环境							
删除其他用户创建的开发环境							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建开发环境的 devfile							
编辑开发环境的 devfile							
删除开发环境的 devfile							
查看开发环境的 devfile							
程序包存储库和程序包权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建程序包存储库							
查看程序包存储库							
编辑程序包存储库							
删除程序包存储库							
创建网关程序包存储库							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看网关程序包存储库							
删除网关程序包存储库							
添加上游程序包存储库							
编辑上游存储库的搜索顺序							
移除上游程序包存储库							
连接到程序包存储库							
从程序包存储库中读取程序包							
将程序包发布到程序包存储库							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
读取和保留上游存储库中的程序包							
查看程序包							
查看程序包版本							
查看程序包版本资产							
列出程序包版本依赖项							
更新程序包版本状态							
更新程序包源配置							
删除程序包版本							

工作流权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建工作流							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
更新工作流	✓	✗	✗	✓	✓	✗	✗
删除工作流	✓	✗	✗	✓	✓	✗	✗
启动工作流	✓	✗	✗	✓	✓	✗	✗
停止工作流	✓	✗	✗	✓	✓	✗	✗
创建工作流密钥	✓	✗	✗	✓	✓	✗	✗
更新工作流密钥	✓	✗	✗	✓	✗	✗	✗
删除工作流密钥	✓	✗	✗	✓	✗	✗	✗
创建环境	✓	✗	✗	✓	✓	✗	✗
删除环境	✓	✗	✗	✓	✗	✗	✗
创建实例集	✓	✗	✗	✓	✗	✗	✗
更新实例集	✓	✗	✗	✓	✗	✗	✗
删除实例集	✓	✗	✗	✓	✗	✗	✗

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
管理其他账户中的工作流资源	✓	✗	✗	✓	✗	✗	✗
将 AWS 账户 连接与环境关联	✓	✗	✗	✓	✗	✗	✗
将默认 IAM 角色与环境关联	✓	✗	✗	✓	✗	✗	✗
将 VPC 连接与环境关联	✓	✗	✗	✓	✗	✗	✗
取消 VPC 连接与环境的关联	✓	✗	✗	✓	✗	✗	✗
将 VPC 连接的环境与工作流关联	✓	✗	✗	✓	✓	✗	✗
取消 VPC 连接的环境与工作流的关联	✓	✗	✗	✓	✓	✗	✗
批准工作流运行	✓	✗	✗	✓	✓	✗	✗

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
在工作流中跟踪提交	✓	✗	✗	✓	✓	✓	✓
查看环境	✓	✗	✗	✓	✓	✓	✓
查看构建操作日志	✓	✗	✗	✓	✓	✓	✓
查看实例集	✓	✗	✗	✓	✓	✓	✓
查看测试操作日志	✓	✗	✗	✓	✓	✓	✓
查看工作流	✓	✗	✗	✓	✓	✓	✓
查看工作流运行	✓	✗	✗	✓	✓	✓	✓
查看工作流运行结果	✓	✗	✗	✓	✓	✓	✓
查看工作流密钥	✓	✗	✗	✓	✓	✓	✓

事务权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建事务	✓	✗	✗	✓	✓	✓	✗
更新事务	✓	✗	✗	✓	✓	✓	✗

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看事务							
创建任务							
更新任务							
查看任务							
存档事务							
将事务分配给 Amazon Q							
针对事务的评论与 Amazon Q 进行交互							
取消向 Amazon Q 分配事务							
使用 Amazon Q 推荐针对事务的任务							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建 Amazon Q 推荐的任务							
更新其他用户创建的事务							
查看事务评论							
创建事务评论							
更新事务评论							
创建标签							
更新标签							
查看标签							
向事务添加标签							
从事务中移除标签							
创建事务的自定义状态							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
更新自定义状态							
查看自定义状态							
移动自定义状态							
停用自定义状态							
向事务添加附件							
查看事务附件							
从事务中移除附件							
将一个事务链接到另一个事务							
取消一个事务与另一个事务的链接							
更新事务链接							
查看事务链接							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
将拉取请求链接到事务							
取消拉取请求与事务的链接							
链接 Jira 项目							
取消链接 Jira 项目							

蓝图权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建自定义蓝图项目							
发布预览自定义蓝图							
发布自定义蓝图							
将自定义蓝图添加到空间蓝图目录中							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
从空间蓝图目录中移除自定义蓝图							
管理自定义蓝图的发布权限							
管理自定义蓝图的目录版本							
更新自定义蓝图							
删除自定义蓝图版本							
删除自定义蓝图							
将源存储库转换为自定义蓝图							
向项目添加自定义蓝图							
取消自定义蓝图与项目的链接							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
更新已应用的自定义蓝图的版本							
编辑自定义蓝图的设置							
查看已发布的自定义蓝图							

通知权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
配置通知渠道							
移除通知渠道							
编辑通知设置							
查看通知设置							
自动接收有关 CodeCatalyst 事件的通知							

权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
为关联的电子邮件账户配置电子邮件通知	✓	✓	✓	✓	✓	✓	✓

搜索权限	空间管理员角色	高级用户角色	受限的访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
在项目内搜索	✓	✗	✗	✓	✓	✓	✓
在空间中搜索	✓	✓	✗	✓	✓	✓	✓

查看和更改用户角色

您可以查看分配给用户的角色。这有助于您了解他们可在项目中执行的操作。如果他们需要其他权限，您也可以更改其角色。

查看用户在项目中的角色

1. 导航到要在其中查看与每个项目成员关联的角色的项目。

   > **Tip**
   >
   > 您可以在顶部导航栏中选择要查看的项目。

2. 在导航窗格中，选择项目设置。
3. 在成员选项卡上，每个项目成员的角色显示在角色中。

更改用户在项目中的角色

1. 导航到要在其中更改与每个项目成员关联的角色的项目。

> **Tip**
>
> 您可以在顶部导航栏中选择要查看的项目。

2. 在导航窗格中，选择项目设置。
3. 在成员选项卡上的项目成员中，选择要更改其角色的用户。选择操作，然后选择编辑角色。
4. 在角色中，选择项目角色，然后选择确认。

查看和更改空间中的角色

所有接受项目邀请的用户都将 CodeCatalyst 成为该项目空间的成员。您可以查看空间成员列表。您可以将用户的角色从受限访问更改为空间管理员，从而更好地管理您的空间及其资源。Space 管理员角色是唯一允许用户在中创建项目的角色 CodeCatalyst。

> **Warning**
>
> 空间管理员角色是中最强大的角色 CodeCatalyst。具有此角色的用户可以在中执行任何操作 CodeCatalyst，包括删除空间。仅将此角色分配给需要此级别的空间访问权限的用户。有关更多信息，请参阅 空间管理员角色。

更改用户在空间中的角色

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择要查看的空间。

3. 选择成员选项卡。
4. 选择要更改其角色的用户，然后选择更改角色。
5. 在更改角色中，选择要分配的角色，然后选择确认。

使用个人访问令牌向用户授予对存储库的访问权限

要在装有 Git 客户端或集成开发环境 (IDE) 的本地计算机上访问某些 CodeCatalyst资源，例如源存储库，必须输入应用程序特定的密码。您可以创建用于此目的的个人访问令牌 (PAT)。 PATs 您在中的所有空间和项目中创建的都与您的用户身份相关联 CodeCatalyst。您可以为自己的 CodeCatalyst身份创建多个 PAT。

您可以查看已创建的 PATs 名称和到期日期，也可以删除不再需要的名称。您只能在创建 PAT 密钥时对其进行复制。

> **Note**
>
> 默认情况下， PATs 将在 1 年后过期。

正在创建 PATs

PATs 与您的用户身份相关联 CodeCatalyst。您只能在创建 PAT 密钥时对其进行复制。

创建 PATs （控制台）

您可以使用控制台在 PATs 中创建 CodeCatalyst。

创建个人访问令牌（控制台）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。 CodeCatalyst我的设置页面打开。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在个人访问令牌下，选择创建。

   这将显示创建 PAT 页面。
4. 在 PAT 名称中，为您的 PAT 输入描述性名称。

5. 在到期日期中，保留默认日期，或者选择日历图标以自定义日期。到期日期默认为自当前日期起之后的 1 年时间。
6. 选择创建。

> **Tip**
>
> 当为源存储库选择克隆存储库时，也可以创建此令牌。

7. 要复制 PAT 密钥，请选择复制。将 PAT 密钥存储到可检索它的位置。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后将无法再检索该密钥。如果您未将 PAT 密钥保存在安全位置，则可以创建另一个 PAT 密钥。

.

正在创建 PATs (CLI)

您可以使用 CLI 在 PATs 中创建 CodeCatalyst。

创建个人访问令牌（AWS CLI）

1. 在终端或命令行中，运行 create-access-token 命令，如下所示。

```
aws codecatalyst create-access-token
```

如果成功，该命令将返回有关创建的 PAT 的信息，如以下示例所示。

```
{
    "secret": "value",
    "name": "marymajor-22222EXAMPLE",
    "expiresTime": "2024-02-04T01:56:04.402000+00:00"
}
```

2.

在创建 PAT 时，您只能查看 PAT 密钥一次。如果您错放了 PAT 密钥或担心它未安全存储，则可以创建另一个密钥。

您可以使用查看与您的用户帐户 PATs 关联的 AWS CLI。您只能查看有关 PAT 的信息，而无法查看 PAT 密钥本身的值。

> **Note**
>
> 请确保使用的是最新版本 AWS CLI 的 CodeCatalyst。早期版本可能不包含这些 CodeCatalyst 命令。必须先配置您的 AWS CLI 配置文件，然后才能将其与一起使用 CodeCatalyst。有关更多信息，请参阅 设置为 AWS CLI 与一起使用 CodeCatalyst。

正在查看 PATs

你可以在 PATs 中查看 CodeCatalyst。该列表显示了您与用户身份关联的所有内容。 PATs 您的 PAT 与您在所有空间和项目中的用户个人资料相关联 CodeCatalyst。已过期 PATs 不会显示，因为它们是在过期后被删除的。

查看 PATs （控制台）

您可以使用控制台在中查看与您的用户身份 PATs 相关联的内容 CodeCatalyst。

查看您的个人访问令牌（控制台）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。 CodeCatalyst 我的设置页面打开。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在 “个人访问令牌” 下，查看当前访问令牌的名称和到期日期 PATs。

查看 PATs (CLI)

您可以使用 CLI 在中查看 PATs 与您的用户身份相关联的内容 CodeCatalyst。

查看您的个人访问令牌（AWS CLI）

- 在终端或命令行中，运行 list-access-tokens 命令，如下所示。

```
aws codecatalyst list-access-tokens
```

如果成功，该命令将返回与您的用户账户 PATs 关联的相关信息，如下例所示。

```
{
    "items": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "marymajor-22222EXAMPLE",
            "expiresTime": "2024-02-04T01:56:04.402000+00:00"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "marymajor-11111EXAMPLE",
            "expiresTime": "2023-03-12T01:58:40.694000+00:00"
        }
    ]
}
```

正在删除 PATs

您可以在中删除 PATs 与您的用户身份关联的内容 CodeCatalyst。

正在删除 PATs（控制台）

您可以使用控制台在 PATs 中删除 CodeCatalyst。

删除个人访问令牌（控制台）

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。 CodeCatalyst 我的设置页面打开。

> **Tip**
>
> 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在个人访问令牌下，选择要删除的 PAT 旁边的选择器，然后选择删除。

   在删除 PAT：<name>? 页面上，要确认删除，请在文本字段中键入 delete。选择删除。

正在删除 PATs (CLI)

您可以使用 AWS CLI删除与用户身份关联的 PAT。为此，您必须提供 PAT 的 ID，可使用 delete-access-token 命令查看该 ID。

> **Note**
>
> 请确保使用的是最新版本 AWS CLI 的 CodeCatalyst。早期版本可能不包含这些 CodeCatalyst 命令。有关与 with AWS CLI 一起使用的更多信息 CodeCatalyst，请参阅设置为 AWS CLI 与一起使用 CodeCatalyst。

删除个人访问令牌（AWS CLI）

- 在终端或命令行中，运行 delete-access-token 命令，并提供要删除的 PAT 的 ID。例如，运行以下命令删除 ID 为的 PAT *123EXAMPLE*。

```
aws codecatalyst delete-access-token --id a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb
```

  如果成功，该命令不返回任何响应。

通过人际关系访问 GitHub 资源

您可以使用个人关系来授权您的第三方 GitHub 资源并将其与之关联 CodeCatalyst。例如，使用个人连接授权 CodeCatalyst 访问您的 GitHub帐户，并创建存储库作为项目或蓝图的来源。该连接将映射到您的 CodeCatalyst 身份，可用于连接到一个或多个源存储库。您创建的连接与您在所有空间和项目中的用户身份相关联 CodeCatalyst。

> **Note**
>
> 在您有权访问的 GitHub 组织中，您可以管理与蓝图的个人联系。

您可以为每种提供商类型的所有空间中的一个用户身份（CodeCatalyst 别名）创建一个个人连接。

您可以使用中的个人关系 CodeCatalyst 为项目创建 GitHub 存储库、为蓝图选择 GitHub 源存储库以及管理存储 GitHub 库中的 CodeCatalyst 拉取请求。

> **Note**
>
> 使用个人关系将蓝图与 GitHub 存储库关联与使用中的 CodeCatalyst 扩展链接存储库不同。GitHub 有关扩展的更多信息，请参阅为带有扩展程序的项目添加功能 CodeCatalyst。

创建个人连接

您可以使用控制台在中创建与您的用户身份关联的个人连接 CodeCatalyst。

创建个人连接

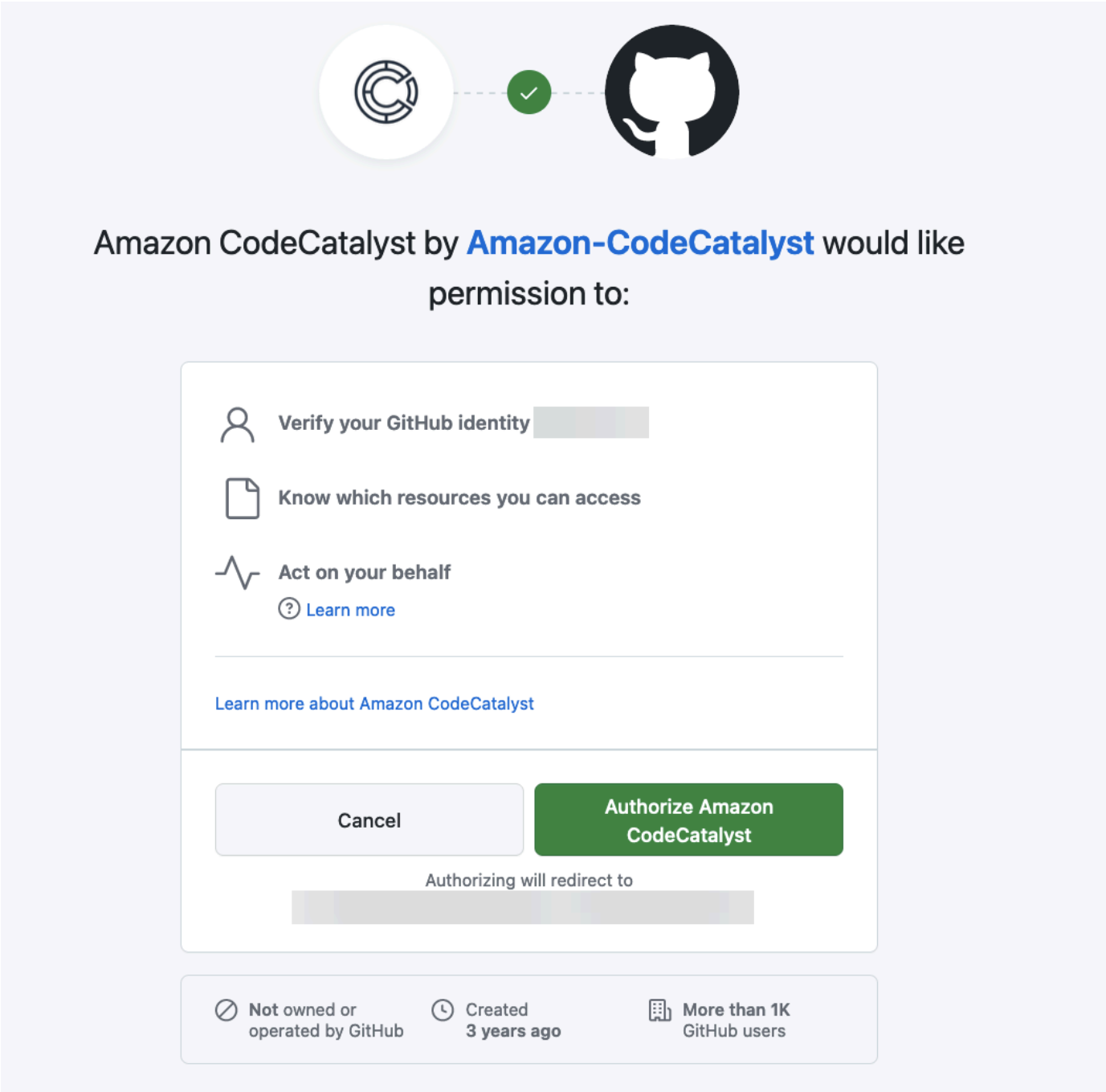
1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。 CodeCatalyst 我的设置页面打开。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在个人连接下，选择创建。

   创建连接页面随即显示。
4. 选择创建。创建连接页面随即显示。
5. 在 “创建连接” 页面的 “提供者” 中，选择GitHub。在连接名称中，键入您的连接的名称。选择创建。
6. 如果出现提示，请登录您的 GitHub 账户。

7. 在连接确认页面上，选择接受。
8. 在安装确认页面上，选择授权按钮以确认要安装连接器应用程序。



删除个人连接

您可以在中删除与您的用户身份关联的个人连接 CodeCatalyst。

> **Note**
>
> 删除中的个人连接 CodeCatalyst 并不会卸载您 GitHub 账户中的应用程序。如果您创建一个新的个人连接，则可以使用应用程序安装。要在中卸载应用程序 GitHub，可以撤消该应用程序，然后稍后重新安装。

要在中删除个人连接 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在顶部菜单栏中，选择您的个人资料徽章，然后选择我的设置。 CodeCatalyst 我的设置页面打开。

   > **Tip**
   >
   > 您还可以通过转到项目或空间的成员页面，然后从成员列表中选择您的姓名，找到您的用户个人资料。

3. 在个人连接下，选择要删除的连接旁边的选择器，然后选择删除。

   在删除连接：<name>? 页面上，要确认删除，请在文本字段中键入 delete。选择删除。

1. 登录 GitHub 并导航到已安装应用程序的帐户设置。选择您的个人资料图标，再选择设置，然后选择应用程序。
2. 在授权 GitHub 应用程序选项卡的授权应用程序列表中，查看已安装的应用程序 CodeCatalyst。要撤销安装，请选择撤销。

将您的 AWS 生成器 ID 配置为使用多重身份验证 (MFA) 登录

无论您创建的 AWS Builder ID 个人资料是供个人使用还是专业用途，我们都建议将多因素身份验证 (MFA) 配置为另一层安全保护。具体而言，如果您是空间成员并与其他人员协作处理项目，我们建议您配置 MFA。由于多个人员可以访问一个项目，因此出现安全漏洞的几率会更高。

启用 MFA 后，您必须 CodeCatalyst 使用电子邮件和密码登录亚马逊。登录的此部分是第一个因素，即您使用自己知道的信息。之后，您使用代码或安全密钥进行登录。这是第二个因素，即您拥有的信息。第二个因素可能是由您的移动设备生成的验证码，或通过点按已连接到计算机的安全密钥生成的验证码。总之，上述多个因素可以防止未经授权的访问，从而提高安全性。

如何注册设备以便在多重身份验证中使用

在我的配置文件 > 多重身份验证中使用以下过程来注册新设备以进行多重身份验证（MFA）。

> **Note**
>
> 我们建议您先将适当的身份验证器应用程序下载到您的设备上，然后再开始执行此过程中的步骤。有关可以在 MFA 设备上使用的应用程序的列表，请参阅 身份验证器应用程序。

注册您的设备，以便在 MFA 上使用

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在配置文件页面上，选择管理配置文件和安全性。这将打开 AWS 构建者 ID 配置文件页面。
4. 在该页面的左侧，选择安全性。
5. 在多重身份验证页面上，选择注册设备。
6. 在注册 MFA 设备页面上，选择以下 MFA 设备类型之一，然后按照说明进行操作：

   - 安全密钥或内置身份验证器

     1. 在注册用户的安全密钥页面上，按照浏览器或平台提供的说明进行操作。

        > **Note**
        >
        > 体验因操作系统和浏览器而异，请按照浏览器或平台显示的说明进行操作。成功注册设备后，您可以选择将友好显示名称与新注册的设备相关联。如果要更改此设置，请选择重命名，输入新名称，然后选择保存。

   - 身份验证器应用程序

     1. 在设置身份验证器应用程序页面上，您可能会注意到新 MFA 设备的配置信息，包括 QR 代码图形。该图表示可在不支持 QR 码的设备上手动输入的密钥。
     2. 使用物理 MFA 设备，执行以下操作：

a. 打开兼容的 MFA 身份验证器应用程序。有关可以在 MFA 设备上使用的经过测试的应用程序的列表，请参阅经过测试的身份验证器应用程序。如果 MFA 应用程序支持多个设备，请选择相应的选项以创建新的 MFA 设备。

b. 确定 MFA 应用程序是否支持 QR 码，然后在设置身份验证器应用程序页面上执行以下操作之一：

   i. 选择显示 QR 代码，然后使用该应用程序扫描 QR 代码。例如，您可选择摄像头图标或选择类似于 Scan code（扫描代码）的选项，然后使用装置的摄像头扫描此代码。

   ii. 选择显示密钥，然后将该密钥输入到您的 MFA 应用程序中。

> **Important**
>
> 当您为 AWS 构建者 ID 配置 MFA 设备时，请将 QR 代码或密钥的副本保存在安全的位置。如果您丢失手机或必须重新安装 MFA 身份验证器应用程序，这会有所帮助。如果出现上述任一情况，您可以快速重新配置应用程序，使其使用相同的 MFA 配置。

3. 在设置身份验证器应用程序页面的身份验证器代码下，输入当前显示在物理 MFA 设备上的一次性密码。

> **Important**
>
> 生成代码之后立即提交您的请求。如果您生成代码后等待时间过长才提交请求，则表示 MFA 设备已成功与您的 AWS Builder ID 配置文件关联，但是 MFA 设备不同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

4. 选择分配 MFA。MFA 设备现在可以开始生成一次性密码，而且可供使用。

身份验证器应用程序

身份验证器应用程序是基于一次性密码（OTP）的第三方身份验证器。用户可将安装在移动设备或平板电脑上的身份验证器应用程序用作授权的 MFA 设备。第三方身份验证器应用程序必须符合 RFC 6238，这是一种基于标准的 TOTP（基于时间的一次性密码）算法，且能够生成六位数身份验证码。

提示进行多重身份验证时，用户必须在显示的输入框中输入来自其身份验证器应用程序的有效代码。分配给用户的每台 MFA 设备必须是唯一的。可为任意给定用户注册两个身份验证器应用程序。

经过测试的身份验证器应用程序

任何符合 TOTP 的应用程序都可以使用 IAM Identity Center MFA，下表列出了已知的第三方身份验证器应用程序以供选择。

操作系统	经过测试的身份验证器应用程序
Android	Authy、Duo Mobile、LastPass 身份验证器、微软身份验证器、谷歌身份验证器
iOS	Authy、Duo Mobile、LastPass 身份验证器、微软身份验证器、谷歌身份验证器

更改 MFA 设备

注册 MFA 设备后，您可以更改其名称或将其删除。建议您始终启用至少一台 MFA 设备以增加一层安全性。您可以注册最多五台设备。要了解如何添加更多设备，请参阅如何注册设备以便在多重身份验证中使用。

重命名 MFA 设备

重命名 MFA 设备

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在配置文件页面上，选择管理配置文件和安全性。这将打开 AWS 构建者 ID 配置文件页面。
4. 选择页面左侧的多重身份验证。在进入此页面后，您将看到重命名已灰显。
5. 选择要更改的 MFA 设备。选择重命名。这将弹出一个模式。
6. 在打开的提示中，在 MFA 设备名称中输入新名称，然后选择重命名。重命名的设备将显示在多重身份验证 (MFA) 设备下。

删除 MFA 设备

删除 MFA 设备

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 在右上角，选择带有您的名字首字母的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在配置文件页面上，选择管理配置文件和安全性。这将打开 AWS 构建者 ID 配置文件页面。
4. 选择页面左侧的多重身份验证。在进入此页面后，您将看到删除已灰显。
5. 选择要更改的 MFA 设备。选择删除。这将显示一个删除 MFA 设备？模式。按照说明操作以删除设备。
6. 选择删除。已删除设备将不再显示在多重身份验证 (MFA) 设备下。

Amazon 的安全 CodeCatalyst

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足最安全敏感的空间要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 CodeCatalyst，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Amazon 时如何应用分担责任模型 CodeCatalyst。它向您展示了如何进行配置 CodeCatalyst 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 CodeCatalyst 资源。

内容

- Amazon CodeCatalyst 中的数据和隐私
- Amazon CodeCatalyst 中的工作流操作的最佳实践
- 了解 CodeCatalyst 信任模型

Amazon CodeCatalyst 中的数据隐私

CodeCatalyst 会在 IAM Identity Center 中收集聚合信息。CodeCatalyst 使用该聚合数据来提供对 CodeCatalyst 中空间和项目的访问权限。收集的聚合信息如下：

- 用户的全名
- 用户的电子邮件
- 用户的用户名

CodeCatalyst 数据会自动被静态加密。客户无需执行任何操作。用户、身份中心或空间被删除后，CodeCatalyst 会在 24 小时内删除客户数据。

CodeCatalyst 使用 AWS 拥有的 AWS KMS 密钥。CodeCatalyst 不支持使用客户自主管理型 KMS 密钥对从 IAM Identity Center 检索到的身份属性进行静态加密。

有关 AWS KMS 密钥的更多信息，请参阅 AWS KMS 密钥中的用户文档。

Amazon 的数据保护 CodeCatalyst

安全与合规是 Amazon CodeCatalyst 和买家的 AWS 共同责任，就像分担适用于您对工作流程中使用的 AWS 资源的使用一样。如本模型所述 CodeCatalyst ，负责保护服务的全球基础架构。您负责维护对托管在此基础结构上的内容的控制。这种责任共担模式适用于中的数据保护 CodeCatalyst。

出于数据保护目的，建议您保护自己的账户凭证并在登录时设置多重身份验证。有关更多信息，请参阅将您的 AWS 生成器 ID 配置为使用多重身份验证 (MFA) 登录。

请勿在标签或自由格式字段（例如名称字段）中输入机密信息或敏感信息（例如您客户的电子邮件地址）。这包括资源名称和您输入的任何其他标识符，以及任何已连接的标识符 AWS 账户。CodeCatalyst例如，请勿在空间、项目或部署实例集名称中输入机密信息或敏感信息。您在标签、名称或用于名称的自由格式字段中输入的任何数据都可能用于计费或诊断日志，或包含在 URL 路径中。这适用于使用控制台、API AWS CLI、 CodeCatalyst 操作开发套件或任何工具 AWS SDKs。

如果您向外部服务器提供 URL，强烈建议您不要在 URL 中包含任何安全凭证信息来验证对该服务器的请求。

CodeCatalyst 源存储库会自动进行静态加密。无需客户采取任何行动。 CodeCatalyst 还使用 HTTPS 协议对传输中的存储库数据进行加密。

CodeCatalyst 不支持使用客户托管的 KMS 密钥对从 IAM Identity Center 检索的身份属性进行静态加密。

CodeCatalyst 支持 MFA。有关更多信息，请参阅 将您的 AWS 生成器 ID 配置为使用多重身份验证 (MFA) 登录。

数据加密

CodeCatalyst 在服务内安全地存储和传输数据。静态和传输中的所有数据均加密。由服务创建或存储的任何数据（包括服务的任何元数据）都原生存储在服务中并进行加密。

> **Note**
>
> 有关事务的信息安全地存储在服务中，而有关未解决事务的信息存储在浏览器的本地缓存中，您已在该浏览器中查看事务面板、待办事项和单个事务。要实现最高安全性，请务必清除浏览器缓存以移除此信息。

如果您使用链接到的资源 CodeCatalyst，例如与某个存储库的账户连接 AWS 账户 或中的链接存储库 GitHub，则从 CodeCatalyst 该链接资源传输的数据会被加密，但该链接资源中的数据处理由该关联服务管理。有关更多信息，请参阅关联服务的文档和 Amazon CodeCatalyst 中的工作流操作的最佳实践。

密钥管理

CodeCatalyst使用 AWS自有的 KMS 密钥。这意味着密钥由管理 AWS ，客户无需直接创建或管理密钥。这简化了 AWS中常见加密场景的密钥管理。

互联网络流量隐私

在中创建空间时 CodeCatalyst，您可以选择该 AWS 区域 空间的数据和资源存储位置。项目数据和元数据绝不会离开该 AWS 区域。但是，为了支持在分区内导航 CodeCatalyst，将在分区 AWS 区域中的所有空间中复制有限的一组空间、项目和用户元数据。它不会被复制到该分区 AWS 区域 之外。例如，如果您在创建空间时选择美国西部（俄勒冈州）作为 AWS 区域 ，则您的数据将不会复制到中国地区或 AWS GovCloud (US)中的区域。有关更多信息，请参阅管理 AWS 区域、AWS 全球基础设施和AWS 服务端点。

在分区 AWS 区域 内复制的数据包括：

- 一种加密的哈希值，它表示空间的名称以确保空间名称的唯一性。此值不是用户可读的，也不会暴露空间的实际名称
- 空间的唯一 ID
- 空间的元数据，有助于跨空间导航
- 空间 AWS 区域 所在的位置
- 该 IDs 领域所有项目的独特之处
- 表示用户在空间或项目中角色的角色 ID
- 注册时 CodeCatalyst，有关注册过程的数据和元数据，包括：
  - 的唯一 ID AWS 构建者 ID
  - 用户在其中的显示名称 AWS 构建者 ID
  - 用户在其中的别名 AWS 构建者 ID
  - 用户注册时使用的电子邮件地址 AWS 构建者 ID
  - 注册过程的进度
  - 如果在注册过程中创建空间，则使用作为空间账单账户的 AWS 账户 ID

空间名称在各处都是唯一的 CodeCatalyst。请勿在空间名称中包含敏感数据。

在使用关联的资源和关联的帐户（例如与 AWS 账户 或 GitHub 存储库的连接）时，我们建议将源位置和目标位置配置为各自支持的最高安全级别。 CodeCatalyst 使用传输层安全 (TLS) 1.2 保护 AWS 账户 AWS 区域、和可用区之间的连接。

Identity and Access Management 和亚马逊 CodeCatalyst

在 Amazon 中 CodeCatalyst，您可以创建并使用 AWS 建筑商 ID 来登录和访问您的空间和项目。AWS 生成器 ID 不是 AWS Identity and Access Management (IAM) 中的身份，也不存在于 AWS 账户。但是，在验证用于计费目的的空间时，以及在连接到空间以在其中创建和使用资源时，CodeCatalyst 确实会与 IAM 集成 AWS 账户。 AWS 账户

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制可以通过身份验证（登录）和授权（具有权限）使用资源的人员。您可以使用 IAM AWS 服务 ，无需支付额外费用。

在 Amazon 中创建空间时 CodeCatalyst，必须将一个 AWS 账户 作为空间的账单账户进行关联。您必须拥有管理员权限 AWS 账户 才能验证 CodeCatalyst 空间，或者拥有该权限。您还可以选择为自己的空间添加一个 IAM 角色，该角色 CodeCatalyst 可用于在连接的空间中创建和访问资源 AWS 账户。该

角色称作服务角色。您可以选择创建与多个账户的连接， AWS 账户 并在每个账户 CodeCatalyst 中为其创建服务角色。

> **Note**
>
> 的账单 CodeCatalyst 是在 AWS 账户 指定的账单账户中进行的。但是，如果您在该 AWS 账户 或任何其他连接中创建 CodeCatalyst 服务角色 AWS 账户，则该 CodeCatalyst 服务角色创建和使用的资源将按该连接 AWS 账户的资源计费。有关更多信息，请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

主题

IAM 中的基于身份的策略

基于身份的策略是可附加到身份的 JSON 权限策略文档。该身份可以是用户、用户组或角色。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

CodeCatalyst 基于身份的策略示例

要查看 CodeCatalyst 基于身份的策略的示例，请参阅。基于身份的连接策略示例 CodeCatalyst

IAM 中的策略操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "prefix:action1",
      "prefix:action2"
         ]
```

IAM 中的策略资源

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

IAM 中的策略条件键

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

基于身份的连接策略示例 CodeCatalyst

中 CodeCatalyst AWS 账户 ，需要管理空间的账单和访问项目工作流程中的资源。账户连接用于授权向空间添加 AWS 账户 。在已连接的 AWS 账户中使用基于身份的策略。

默认情况下，用户和角色没有创建或修改 CodeCatalyst 资源的权限。他们也无法使用 AWS 管理控制台、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限，以对所需资源执行操作。然后，管理员必须为需要这些策略的用户附加这些策略。

以下 IAM 策略示例授予与账户连接相关的操作的权限。使用它们来限制连接账户的访问权限 CodeCatalyst。

示例 1：允许用户单次接受连接请求 AWS 区域

以下权限策略仅允许用户查看和接受 CodeCatalyst 和之间的连接请求 AWS 账户。此外，该策略使用一个条件来仅允许在 us-west-2 区域执行操作，而不允许来自其他区域的操作。 AWS 区域要查看和批准请求，用户需要使用 AWS 管理控制台 与请求中指定的帐户相同的帐户登录。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:GetPendingConnection"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-west-2"
        }
      }
```

```
    }
  ]
}
```

示例 2：允许在控制台中管理单个的连接 AWS 区域

以下权限策略允许用户管理单个区域之间 CodeCatalyst和区域 AWS 账户 内的连接。该策略使用一个条件来仅允许在 us-west-2 区域执行操作，而不允许来自其他区域的操作。 AWS 区域创建连接后，您可以通过选择 AWS 管理控制台中的选项来创建 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。在示例策略中，`iam:PassRole`操作的条件包括的服务委托人。 CodeCatalyst仅具有该访问权限的角色才会在 AWS 管理控制台中创建。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecatalyst:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-west-2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

示例 3：禁止管理连接

以下权限策略不允许用户管理 CodeCatalyst 和之间的连接 AWS 账户。

JSON

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

使用标签控制对账户连接资源的访问

标签可以附加到资源，也可以从请求传入支持标签的服务。策略中的资源可以具有标签，而且策略中的某些操作可以包括标签。标记条件键包括 `aws:RequestTag` 和 `aws:ResourceTag` 条件键。在创建 IAM 策略时，您可以使用标签条件键来控制：

- 哪些用户可以对连接资源执行操作（基于该资源已有的标签）。
- 哪些标签可以在操作的请求中传递。
- 特定的标签键是否能在请求中使用。

以下示例演示了如何在策略中为 CodeCatalyst账户连接用户指定标签条件。有关条件键的更多信息，请参阅 IAM 中的策略条件键。

示例 1：基于请求中的标签允许操作

以下策略授予用户批准账户连接的权限。

为此，如果请求指定一个名为 `Project` 的带有值 `ProjectA` 的标签，则它允许 `AcceptConnection` 和 `TagResource` 操作。（`aws:RequestTag` 条件键用于控制可以通过 IAM 请求传递哪些标签。）`aws:TagKeys` 条件确保标签键区分大小写。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

示例 2：基于资源标签允许操作

以下策略授予用户对账户连接资源执行操作以及获取其相关信息的权限。

为此，如果该连接具有名为 Project 的带值 ProjectA 的标签，则它允许特定操作。（aws:ResourceTag 条件键用于控制可以通过 IAM 请求传递哪些标签。）

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:GetConnection",
        "codecatalyst:DeleteConnection",
        "codecatalyst:AssociateIamRoleToConnection",
        "codecatalyst:DisassociateIamRoleFromConnection",
        "codecatalyst:ListIamRolesForConnection",
        "codecatalyst:PutBillingAuthorization"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "ProjectA"
        }
      }
    }
  ]
}
```

CodeCatalyst 权限参考

本节提供与账户连接资源相关联的操作 AWS 账户 的权限参考 CodeCatalyst。以下部分描述了与连接账户相关的仅限授权执行的操作。

所需的账户连接权限

需要以下权限才能使用账户连接。

CodeCatalyst 账户连接权限	所需的权限	资源
AcceptConnection	需要接受将此账户关联到CodeCatalyst空间的请求。这只是一种 IAM 策略权限，不是 API 操作。	仅支持在策略的 Resource 元素中使用通配符 (*)。
AssociateIamRoleToConnection	需要此权限才能将 IAM 角色与账户连接关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DeleteConnection	需要此权限才能删除账户连接。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DisassociateIamRoleFromConnection	需要此权限才能解除 IAM 角色与账户连接的关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetBillingAuthorization	需要此权限才能描述账户连接的计费授权。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetConnection	需要此权限才能获取账户连接。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetPendingConnection	需要获取将此账户关联到CodeCatalyst空间的待处理请求。这只是一种 IAM 策略权限，不是 API 操作。	仅支持在策略的 Resource 元素中使用通配符 (*)。

CodeCatalyst 账户连接权限	所需的权限	资源
ListConnections	需要此权限才能列出未处于待处理状态的账户连接。这只是一种 IAM 策略权限，不是 API 操作。	仅支持在策略的 Resource 元素中使用通配符 (*)。
ListIamRolesForConnection	需要此权限才能列出与账户连接关联的 IAM 角色。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
ListTagsForResource	需要此权限才能列出与账户连接关联的标签。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
PutBillingAuthorization	需要此权限才能创建或更新账户连接的计费授权。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
RejectConnection	需要拒绝将此账户连接到 CodeCatalyst空间的请求。这只是一种 IAM 策略权限，不是 API 操作。	仅支持在策略的 Resource 元素中使用通配符 (*)。
TagResource	需要此权限才能创建或编辑与账户连接关联的标签。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
UntagResource	需要此权限才能移除与账户连接关联的标签。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*

所需的 IAM Identity Center 应用程序权限

需要以下权限才能使用 IAM Identity Center 应用程序。

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
AssociateIdentityCenterApplicationToSpace	需要将 IAM 身份中心应用程序与 CodeCatalyst 空间关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
AssociateIdentityToIdentityCenterApplication	需要将身份与 CodeCatalyst空间的 IAM 身份中心应用程序关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
BatchAssociateIdentitiesToIdentityCenterApplication	需要将多个身份与 CodeCatalyst 空间的 IAM Identity Center 应用程序关联起来。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
BatchDisassociateIdentitiesFromIdentityCenterApplication	需要为 CodeCatalyst 空间解除多个身份与 IAM 身份中心应用程序的关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
CreateIdentityCenterApplication	需要此权限才能创建 IAM Identity Center 应用程序。这	arn:aws:codecatalyst:region: *account_ID* :/identity-

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
	只是一种 IAM 策略权限，不是 API 操作。	center-applicati ons/ *identity-center-application_ID*
CreateSpaceAdminRo leAssignment	需要为给定 CodeCatalyst空间和 IAM Identity Center 应用程序创建管理员角色分配。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
DeleteIdentityCenterApplica tion	需要此权限才能删除 IAM Identity Center 应用程序。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
DisassociateIdentityCenterA pplicationFromSpace	需要解除 IAM 身份中心应用程序与 CodeCatalyst 空间的关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
DisassociateIdentityFromIde ntityCenterApplication	需要为 CodeCatalyst 空间解除身份与 IAM 身份中心应用程序的关联。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
GetIdentityCenterApplication	需要此权限才能获取有关 IAM Identity Center 应用程序的信息。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
ListIdentityCenterApplications	需要此权限才能查看账户中所有 IAM Identity Center 应用程序的列表。这只是一种 IAM 策略权限，不是 API 操作。	仅支持在策略的 Resource 元素中使用通配符 (*) 。
ListIdentityCenterApplicationsForSpace	按 CodeCatalyst 空间查看 IAM 身份中心应用程序列表所必需的。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
ListSpacesForIdentityCenterApplication	需要通过 IAM 身份中心应用程序查看 CodeCatalyst 空间列表。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
SynchronizeIdentityCenterApplication	需要此权限才能将 IAM Identity Center 应用程序与备用身份存储同步。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
UpdateIdentityCenterApplication	需要此权限才能更新 IAM Identity Center 应用程序。这只是一种 IAM 策略权限，不是 API 操作。	`arn:aws:codecatalyst:region:` *`account_ID`* `:/identity-center-applications/` *`identity-center-application_ID`*

使用 CodeCatalyst 的服务相关角色

Amazon CodeCatalyst 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 CodeCatalyst 直接相关。服务相关角色由 CodeCatalyst 预定义，并包含服务代表您调用其它 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 CodeCatalyst，因为您不必手动添加必要的权限。CodeCatalyst 定义其服务相关角色的权限，除非另外定义，否则只有 CodeCatalyst 可以代入其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这可以保护您的 CodeCatalyst 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并查找服务相关角色列中显示为是的服务。请选择是与查看该服务的服务相关角色文档的链接。

CodeCatalyst 的服务相关角色权限

CodeCatalyst 使用名为 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 的服务相关角色 – 使 Amazon CodeCatalyst 能够代表您对应用程序实例配置文件及相关目录用户和组进行只读访问。

AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 服务相关角色信任以下服务代入该角色：

- `codecatalyst.amazonaws.com`

名为 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy 的角色权限策略使 CodeCatalyst 能够对指定资源完成以下操作：

- 操作：CodeCatalyst spaces that support identity federation and SSO users and groups 的 View application instance profiles and associated directory users and groups

您必须配置使用户、组或角色能够创建、编辑或删除服务相关角色的权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

创建 CodeCatalyst 的服务相关角色

您无需手动创建服务相关角色。当您在 AWS 管理控制台、AWS CLI 或 AWS API 中创建空间时，CodeCatalyst 会为您创建服务相关角色。

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。此外，如果您在 2023 年 11 月 17 日之前使用 CodeCatalyst 服务，当它开始支持服务相关角色时，则 CodeCatalyst 会在您的账户中创建 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。要了解更多信息，请参阅我的 AWS 账户中出现新角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您创建空间时，CodeCatalyst 会再次为您创建服务相关角色。

您也可以使用 IAM 控制台，在查看应用程序实例配置文件及相关目录用户和组使用案例中创建服务相关角色。在 AWS CLI 或 AWS API 中，使用 codecatalyst.amazonaws.com 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

编辑 CodeCatalyst 的服务相关角色

您无法使用 CodeCatalyst 编辑 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除 CodeCatalyst 的服务相关角色

无需手动删除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。在 AWS 管理控制台、AWS CLI 或 AWS API 中删除空间时，CodeCatalyst 会为您清理资源并删除服务相关角色。

还可以使用 IAM 控制台、AWS CLI 或 AWS API 手动删除服务相关角色。为此，必须先手动清除服务相关角色的资源，然后才能手动删除。

> **Note**
>
> 如果在您试图删除资源时 CodeCatalyst 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 所用的 CodeCatalyst 资源

- 删除空间。

使用 IAM 手动删除 服务相关角色

使用 IAM 控制台，即 AWS CLI 或 AWS API 来删除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

CodeCatalyst 服务相关角色的受支持区域

CodeCatalyst 支持在提供该服务的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

CodeCatalyst 不支持在提供该服务的每个区域中使用服务相关角色。您可以在以下区域中使用 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。

区域名称	区域标识	CodeCatalyst 中的支持
美国东部（弗吉尼亚州北部）	us-east-1	否
美国东部（俄亥俄州）	us-east-2	否

区域名称	区域标识	CodeCatalyst 中的支持
美国西部（加利福尼亚北部）	us-west-1	否
美国西部（俄勒冈州）	us-west-2	是
非洲（开普敦）	af-south-1	否
亚太地区（香港）	ap-east-1	否
亚太地区（雅加达）	ap-southeast-3	否
亚太地区（孟买）	ap-south-1	否
亚太地区（大阪）	ap-northeast-3	否
亚太地区（首尔）	ap-northeast-2	否
亚太地区（新加坡）	ap-southeast-1	否
亚太地区（悉尼）	ap-southeast-2	否
亚太地区（东京)	ap-northeast-1	否
加拿大（中部）	ca-central-1	否
欧洲地区（法兰克福）	eu-central-1	否
欧洲地区（爱尔兰）	eu-west-1	是
欧洲地区（伦敦）	eu-west-2	否
欧洲地区（米兰）	eu-south-1	否
欧洲地区（巴黎）	eu-west-3	否
欧洲地区（斯德哥尔摩）	eu-north-1	否
中东（巴林）	me-south-1	否
中东（阿联酋）	me-central-1	否

区域名称	区域标识	CodeCatalyst 中的支持
South America（São Paulo）	sa-east-1	否
AWS GovCloud（美国东部）	us-gov-east-1	否
AWS GovCloud（美国西部）	us-gov-west-1	否

AWS 适用于亚马逊的托管政策 CodeCatalyst

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略：AmazonCodeCatalystSupportAccess

该策略向所有空间管理员和空间成员授予使用与空间计费账户关联的 Business 或 Enterprise 高级支持计划的权限。这些权限允许空间管理员和成员利用高级支持计划来获得权限策略中他们有权访问的资源。 CodeCatalyst

权限详细信息

该策略包含以下权限。

- support— 授予权限以允许用户搜索、创建和解决 Su AWS pport 案例。还授予描述通信、严重性级别、附件和相关支持案例详细信息的权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "support:DescribeAttachment",
        "support:DescribeCaseAttributes",
        "support:DescribeCases",
        "support:DescribeCommunications",
        "support:DescribeIssueTypes",
        "support:DescribeServices",
        "support:DescribeSeverityLevels",
        "support:DescribeSupportLevel",
        "support:SearchForCases",
        "support:AddAttachmentsToSet",
        "support:AddCommunicationToCase",
        "support:CreateCase",
        "support:InitiateCallForCase",
        "support:InitiateChatForCase",
        "support:PutCaseAttributes",
        "support:RateCaseCommunication",
        "support:ResolveCase"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 托管策略：AmazonCodeCatalystFullAccess

该政策授予在亚马逊 CodeCatalyst CodeCatalyst 空间页面中管理您的空间和关联账户的权限 AWS 管理控制台。此应用程序用于配置与您的空间 AWS 账户 相连的内容 CodeCatalyst。

权限详细信息

该策略包含以下权限。

- `codecatalyst`— 授予访问中 Amazon CodeCatalyst Spaces 页面的全部权限 AWS 管理控制台。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeCatalystResourceAccess",
            "Effect": "Allow",
            "Action": [
                "codecatalyst:*",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeCatalystAssociateIAMRole",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
```

```
    ]
}
```

AWS 托管策略：AmazonCodeCatalystReadOnlyAccess

该政策授予在 Amazon Spaces 页面中查看和列出 CodeCatalyst 空间和关联账户信息的权限 AWS 管理控制台。此应用程序用于配置与您的空间 AWS 账户 相连的内容 CodeCatalyst。

权限详细信息

该策略包含以下权限。

- `codecatalyst`— 向中的 Amazon CodeCatalyst Spaces 页面授予只读权限 AWS 管理控制台。

JSON

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

AWS 托管策略：AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy

您不能将 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy 附加到自己的 IAM 实体。此策略附加到允许代表您执行操作 CodeCatalyst 的服务相关角色。有关更多信息，请参阅 使用 CodeCatalyst 的服务相关角色。

此策略允许客户在管理空间时查看应用程序实例配置文件以及关联的目录用户和组 CodeCatalyst。客户在管理支持身份联合验证以及 SSO 用户和组的空间时将查看这些资源。

权限详细信息

该策略包含以下权限。

- sso— 授予权限以允许用户查看在 IAM Identity Center 中管理的相关空间的应用程序实例配置文件 CodeCatalyst。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid":
 "AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy",
   "Effect": "Allow",
   "Action": [
    "sso:ListInstances",
    "sso:ListApplications",
    "sso:ListApplicationAssignments",
    "sso:DescribeInstance",
    "sso:DescribeApplication"
   ],
   "Resource": "*"
  }
 ]
}
```

CodeCatalyst AWS 托管策略的更新

查看 CodeCatalyst 自该服务开始跟踪这些更改以来 AWS 托管策略更新的详细信息。要获得有关此页面更改的自动提醒，请订阅 “ CodeCatalyst 文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy - 新策略	CodeCatalyst 添加了策略。 授予权限以允许 CodeCatalyst 用户查看应用程序实例配置文件以及关联的目录用户和组。	2023 年 11 月 17 日
AmazonCodeCatalystSupportAccess：新策略	CodeCatalyst 添加了策略。 授予权限以允许 CodeCatalyst 用户搜索、创建和解决支持案例，以及查看相关通信和详细信息。	2023 年 4 月 20 日
AmazonCodeCatalystFullAccess：新策略	CodeCatalyst 添加了策略。 授予对的完全访问权限 CodeCatalyst。	2023 年 4 月 20 日
AmazonCodeCatalystReadOnlyAccess：新策略	CodeCatalyst 添加了策略。 授予对的只读访问权限 CodeCatalyst。	2023 年 4 月 20 日
CodeCatalyst 开始跟踪更改	CodeCatalyst 开始跟踪其 AWS 托管策略的更改。	2023 年 4 月 20 日

使用 IAM 角色授予对项目 AWS 资源的访问权限

CodeCatalyst 可以通过将您的连接到 CodeCatalyst 空间 AWS 账户 来访问 AWS 资源。之后，您可以创建以下服务角色并在连接账户时将其关联。

有关在 JSON 策略中使用的元素的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

- 要访问 CodeCatalyst 项目和工作流程中的资源，必须先授予代表您访问这些资源的权限。 AWS 账户 CodeCatalyst 为此，您必须在 connected 中创建一个服务角色 AWS 账户 ，该角色 CodeCatalyst可以代表空间中的用户和项目代替。您可以选择创建和使用

CodeCatalystWorkflowDevelopmentRole-*spaceName* 服务角色，也可以创建自定义的服务角色并手动配置这些 IAM 策略和角色。最佳实践是向这些角色分配最少的必需权限。

> **Note**
>
> 对于自定义的服务角色，需要 CodeCatalyst 服务主体。有关 CodeCatalyst 服务主体和信任模型的更多信息，请参阅了解 CodeCatalyst 信任模型。

- 要通过互联管理对空间的支持 AWS 账户，您可以选择创建和使用允许 CodeCatalyst 用户访问支持的AWSRoleForCodeCatalystSupport服务角色。有关 CodeCatalyst 空间支持的更多信息，请参阅支持 for Amazon CodeCatalyst。

了解 CodeCatalystWorkflowDevelopmentRole-*spaceName* 服务角色

您可以为空间添加一个 IAM 角色，该角色 CodeCatalyst 可用于在互联环境中创建和访问资源 AWS 账户。该角色称作服务角色。创建服务角色的最简单方法是，在创建空间时添加一个服务角色并为该角色选择 CodeCatalystWorkflowDevelopmentRole-*spaceName* 选项。这不仅创建了`AdministratorAccess`附加的服务角色，而且还创建了信任策略， CodeCatalyst 允许在空间中的项目中代表用户担任该角色。服务角色的范围限于空间而不是单个项目。如需创建此角色，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。您只能为每个账户中的每个空间创建一个角色。

> **Note**
>
> 此角色仅建议与开发账户一起使用，并且使用`AdministratorAccess` AWS 托管策略，从而赋予其在其中创建新策略和资源的完全访问权限 AWS 账户。

附加到 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色的策略旨在用于使用空间中的蓝图创建的项目。该策略使这些项目中的用户能够使用已连接的 AWS 账户中的资源来开发、构建、测试和部署代码。有关更多信息，请参阅为 AWS 服务创建角色。

附加到CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的策略是中的`AdministratorAccess`托管策略 AWS。这是一项授予对所有 AWS 操作和资源的完全访问权限的策略。要在 IAM 控制台中查看 JSON 策略文档，请参阅AdministratorAccess。

以下信任策略 CodeCatalyst 允许代入该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。有关 CodeCatalyst 信任模型的更多信息，请参阅了解 CodeCatalyst 信任模型。

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/*"
                }
            }
        }
    ]
```

为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色

按照以下步骤操作，创建将用于空间中的工作流的 CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色。对于要具有在项目中使用的 IAM 角色的每个账户，您必须向空间添加一个角色，例如开发人员角色。

在开始之前，您必须拥有自己的管理权限 AWS 账户 或能够与您的管理员合作。有关中如何使用 AWS 账户 和 IAM 角色的更多信息 CodeCatalyst，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。

要创建并添加 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-***spaceName***

1. 在开始进入 CodeCatalyst 控制台之前，请先打开 AWS 管理控制台，然后确保您的空间使用相同 AWS 账户 的方式登录。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
3. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
4. 选择要创建角色的 AWS 账户 位置的链接。此时将显示 AWS 账户 详细信息页面。
5. 从中选择管理角色 AWS 管理控制台。

“将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

6. 选择在 IAM 中创建 CodeCatalyst 开发管理员角色。此选项将创建一个服务角色，其中包含开发角色的权限策略和信任策略。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-spaceName`。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。

   > **Note**
   >
   > 此角色仅建议与开发者账户一起使用，并且使用`AdministratorAccess` AWS 托管策略，授予其在其中创建新策略和资源的完全访问权限 AWS 账户。

7. 选择创建开发角色。
8. 在连接页面的可用的 IAM 角色下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表中的角色。`CodeCatalystWorkflowDevelopmentRole-spaceName`
9. 要返回您的空间，请选择 Go to Amazon CodeCatalyst。

了解 AWSRoleForCodeCatalystSupport 服务角色

您可以为空间添加一个 IAM 角色，空间中的 CodeCatalyst 用户可以使用该角色来创建和访问支持案例。此角色称作用于支持的服务角色。创建用于支持的服务角色的最简单方法是，在创建空间时添加一个服务角色并为该角色选择 `AWSRoleForCodeCatalystSupport` 选项。这不仅可以创建策略和角色，还可以创建信任策略，该策略 CodeCatalyst 允许在空间中的项目中代表用户担任该角色。服务角色的范围限于空间而不是单个项目。如需创建此角色，请参阅为您的账户和空间创建 AWSRoleForCodeCatalystSupport 角色。

附加到 `AWSRoleForCodeCatalystSupport` 角色的策略是提供支持权限的托管式策略。有关更多信息，请参阅 AWS 托管策略：AmazonCodeCatalystSupportAccess。

策略的信任角色 CodeCatalyst 允许代入该角色。

JSON

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

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

为您的账户和空间创建 AWSRoleForCodeCatalystSupport 角色

按照以下步骤操作，创建将用于空间中的支持案例的 `AWSRoleForCodeCatalystSupport` 角色。必须将该角色添加到空间的指定计费账户中。

在开始之前，您必须拥有自己的管理权限 AWS 账户 或能够与您的管理员合作。有关中如何使用 AWS 账户 和 IAM 角色的更多信息 CodeCatalyst，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。

要创建并添加 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在开始进入 CodeCatalyst 控制台之前，请先打开 AWS 管理控制台，然后确保您的空间使用相同 AWS 账户 的方式登录。
2. 导航到您的 CodeCatalyst 空间。选择设置，然后选择 AWS 账户。
3. 选择要创建角色的 AWS 账户 位置的链接。此时将显示 AWS 账户 详细信息页面。
4. 从中选择管理角色 AWS 管理控制台。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS 管理控制台。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。
5. 在CodeCatalyst 空间详情下，选择添加 Su CodeCatalyst pport 角色。此选项将创建一个服务角色，其中包含预览开发角色的权限策略和信任策略。该角色的名称为 AWSRoleForCodeCatalystSupport，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 AWSRoleForCodeCatalystSupport 服务角色。
6. 在 “为 Su CodeCatalyst pport 添加角色” 页面上，将默认角色保留为选中状态，然后选择创建角色。

7. 在 “可用的 IAM 角色” CodeCatalystWorkflowDevelopmentRole-*spaceName* 下 CodeCatalyst，查看添加到您的账户的 IAM 角色列表中的角色。
8. 要返回您的空间，请选择 Go to Amazon CodeCatalyst。

在中为工作流程操作配置 IAM 角色 CodeCatalyst

本部分详细介绍了您可以创建的用于 CodeCatalyst账户的 IAM 角色和策略。有关创建示例角色的说明，请参阅手动为工作流操作创建角色。创建 IAM 角色后，复制角色 ARN 以将 IAM 角色添加到账户连接中，并将其与项目环境关联。要了解更多信息，请参阅将 IAM 角色添加到账户连接。

CodeCatalyst 为 Amazon S3 访问权限构建角色

对于 CodeCatalyst 工作流程构建操作，您可以使用默认CodeCatalystWorkflowDevelopmentRole-***spaceName***服务角色，也可以创建名为 CodeCatalystBuildRoleforS3 Access 的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在中的 CloudFormation 资源上运行任务。 AWS 账户

此角色授予执行以下操作所需的权限：

- 写入 Amazon S3 存储桶。
- 使用 Support 来支持资源构建 CloudFormation。这需要 Amazon S3 访问权限。

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为其构建角色 CloudFormation

对于 CodeCatalyst 工作流程构建操作，您可以使用默认CodeCatalystWorkflowDevelopmentRole-***spaceName***服务角色，也可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在中的 CloudFormation 资源上运行任务。 AWS 账户

此角色授予执行以下操作所需的权限：

- 使用 Support 来支持资源构建 CloudFormation。这与访问 Amazon S3 的 CodeCatalyst构建角色和的 CodeCatalyst 部署角色一起是必需的 CloudFormation。

应将以下 AWS 托管策略附加到此角色：

- AWSCloudFormationFullAccess
- IAMFull访问
- 亚马逊 3 FullAccess
- 亚马逊APIGateway管理员
- AWSLambdaFullAccess

CodeCatalyst 为 CDK 构建角色

对于运行 CDK 构建操作 CodeCatalyst 的工作流程，例如现代三层 Web 应用程序，您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要对您的 CloudFormation 资源进行引导和运行 CDK 构建命令。 AWS 账户

此角色授予执行以下操作所需的权限：

- 写入 Amazon S3 存储桶。
- Support 构建 CDK 构造和 CloudFormation 资源堆栈。这需要访问 Amazon S3 以存储构件，访问 Amazon ECR 以获得映像存储库支持，并访问 SSM 以进行系统治理和监控虚拟实例。

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为部署角色 CloudFormation

对于使用 CodeCatalyst 的工作流程部署操作 CloudFormation，您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以使用具有限定权限的策略，该策略 CodeCatalyst 需要对中的 CloudFormation AWS 账户资源运行任务。

此角色授予执行以下操作所需的权限：

- CodeCatalyst 允许调用 λ 函数来执行 blue/green 部署 CloudFormation。
- CodeCatalyst 允许在中创建和更新堆栈和变更集。 CloudFormation

该角色使用以下策略：

```
{"Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
    "Effect": "Allow"
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为 Amazon 部署角色 EC2

CodeCatalyst 工作流程部署操作使用具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您 AWS 账户的 Amazon EC2 资源上运行任务。该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的默认策略不包括亚马逊 EC2 或 Amazon A EC2 uto Scaling 的权限。

此角色授予执行以下操作所需的权限：

- 创建亚马逊 EC2 部署。
- 读取实例上的标签或通过 Auto Scaling 组名识别亚马逊 EC2 实例。
- 读取、创建、更新和删除 Amazon A EC2 uto Scaling 群组、生命周期挂钩和扩展策略。
- 将信息发布到 Amazon SNS 主题。
- 检索有关 CloudWatch 警报的信息。
- 读取和更新 Elastic Load Balancing。

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为 Amazon ECS 部署角色

对于 CodeCatalyst 工作流程操作，您可以创建具有必要权限的 IAM 角色。您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以创建用于 CodeCatalyst 部署操作的 IAM 角色以用于 Lambda 部署。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您 AWS 账户的 Amazon ECS 资源上运行任务。

此角色授予执行以下操作所需的权限：

- 代表 CodeCatalyst 用户使用 CodeCatalyst 连接中指定的账户启动滚动部署 Amazon ECS。
- 读取、更新和删除 Amazon ECS 任务集。
- 更新 Elastic Load Balancing 目标组、侦听器和规则。

- 调用 Lambda 函数。
- 访问 Amazon S3 存储桶中的修订文件。
- 检索有关 CloudWatch 警报的信息。
- 将信息发布到 Amazon SNS 主题。

该角色使用以下策略：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
```

```
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

Note

第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。

```
"Resource": "*"
```

CodeCatalyst 为 Lambda 部署角色

对于 CodeCatalyst 工作流程操作，您可以创建具有必要权限的 IAM 角色。您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以创建用于 CodeCatalyst部署操作的 IAM 角色以用于 Lambda 部署。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您的 Lambda 资源上运行任务。 AWS 账户

此角色授予执行以下操作所需的权限：

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

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为 Lambda 部署角色

对于 CodeCatalyst 工作流程操作，您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您的 Lambda 资源上运行任务。 AWS 账户

此角色授予执行以下操作所需的权限：

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

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为部署角色 AWS SAM

对于 CodeCatalyst 工作流程操作，您可以使用默认CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色，也可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您的 AWS 账户上运行任务 AWS SAM 和 CloudFormation 资源。

此角色授予执行以下操作所需的权限：

- CodeCatalyst 允许调用 Lambda 函数来部署无服务器和 CLI AWS SAM 应用程序。
- CodeCatalyst 允许在中创建和更新堆栈和变更集。 CloudFormation

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon 的只读角色 EC2

对于 CodeCatalyst 工作流程操作，您可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您 AWS 账户的 Amazon EC2 资源上运行任务。该CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色不包括亚马逊的权限 EC2 或所描述的亚马逊操作 CloudWatch。

此角色授予执行以下操作所需的权限：

- 获取 Amazon EC2 实例的状态。
- 获取 Amazon EC2 实例的 CloudWatch 指标。

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon ECS 的只读角色

对于 CodeCatalyst 工作流程操作，您可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您 AWS 账户的 Amazon ECS 资源上运行任务。

此角色授予执行以下操作所需的权限：

- 读取 Amazon ECS 任务集。
- 检索有关 CloudWatch 警报的信息。

该角色使用以下策略：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Lambda 的只读角色

对于 CodeCatalyst 工作流程操作，您可以创建具有必要权限的 IAM 角色。此角色使用具有限定权限的策略，该策略 CodeCatalyst 需要在您的 Lambda 资源上运行任务。 AWS 账户

此角色授予执行以下操作所需的权限：

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

该角色使用以下策略。

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

手动为工作流操作创建角色

CodeCatalyst 工作流程操作使用您创建的 IAM 角色，这些角色称为构建角色、部署角色和堆栈角色。

按以下步骤操作，在 IAM 中创建这些角色。

创建部署角色

1. 按如下步骤操作，为角色创建策略：
    a. 登录到 AWS。
    b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
    c. 在导航窗格中，选择策略。
    d. 选择创建策略。
    e. 选择 JSON 选项卡。
    f. 删除现有代码。

g. 粘贴以下代码：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-deploy-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建部署角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-deploy-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-deploy-role
```

i. 对于角色描述，输入：

```
CodeCatalyst deploy role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的部署角色。

3. 按如下步骤操作，获取部署角色 ARN：

   a. 在导航窗格中，选择角色。

   b. 在搜索框中，输入刚创建的角色的名称（`codecatalyst-deploy-role`）。

   c. 从列表中选择该角色。

      此时将显示该角色的摘要页面。

   d. 在顶部，复制 ARN 值。

现在，您已创建具有相应权限的部署角色并已获取其 ARN。

创建构建角色

1. 按如下步骤操作，为角色创建策略：

   a. 登录到 AWS。

   b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   c. 在导航窗格中，选择策略。

   d. 选择创建策略。

   e. 选择 JSON 选项卡。

   f. 删除现有代码。

   g. 粘贴以下代码：

      > **Note**
      >
      > 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
      >
      > ```
      > "Resource": "*"
      > ```

   h. 选择下一步：标签。

   i. 选择下一步：审核。

   j. 在名称中，输入：

```
codecatalyst-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 在权限策略中，搜索 codecatalyst-build-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst build role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

3. 按如下步骤操作，获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-build-role）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。

现在，您已创建具有相应权限的构建角色并已获取其 ARN。

创建堆栈角色

> **Note**
>
> 您不必创建堆栈角色，但出于安全考虑，建议您这样做。如果您不创建堆栈角色，则需要将此过程中进一步描述的权限策略添加到部署角色。

1. AWS 使用要部署堆栈的账户登录。
2. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 在导航窗格中，选择角色，然后选择创建角色。
4. 选择顶部的 AWS 服务。
5. 从服务列表中选择CloudFormation。
6. 选择下一步: 权限。
7. 在搜索框中，添加访问堆栈中的资源所需的所有策略。例如，如果您的堆栈包含一个 AWS Lambda 函数，则需要添加授予对 Lambda 的访问权限的策略。

   > **Tip**
   >
   > 如果您不确定要添加哪些策略，此时可以将其忽略。在测试操作时，如果您没有正确的权限，则 CloudFormation 会生成错误，显示您需要添加哪些权限。

8. 选择下一步：标签。
9. 选择下一步：审核。
10. 对于角色名称，输入：

    ```
    codecatalyst-stack-role
    ```

11. 选择创建角色。
12. 要获取堆栈角色的 ARN，请执行以下操作：

    a. 在导航窗格中，选择角色。

    b. 在搜索框中，输入刚创建的角色的名称（`codecatalyst-stack-role`）。

    c. 从列表中选择该角色。

    d. 在摘要页面上，复制角色 ARN 值。

AWS CloudFormation 用于在 IAM 中创建策略和角色

您可以选择创建和使用 AWS CloudFormation 模板来创建访问 CodeCatalyst 项目和工作流程中资源所需的策略和角色。 AWS 账户 CloudFormation 是一项服务，可帮助您对 AWS 资源进行建模和设置，这样您就可以花更少的时间管理这些资源，而将更多的时间集中在运行的应用程序上 AWS。如果您打算创建多个角色 AWS 账户，则创建模板可以帮助您更快地执行此任务。

以下示例模板创建了部署操作角色和策略。

```
Parameters:
  CodeCatalystAccountId:
    Type: String
    Description: Account ID from the connections page
  ExternalId:
    Type: String
    Description: External ID from the connections page
Resources:
  CrossAccountRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS:
                - !Ref CodeCatalystAccountId
            Action:
              - 'sts:AssumeRole'
            Condition:
              StringEquals:
                sts:ExternalId: !Ref ExternalId
      Path: /
      Policies:
        - PolicyName: CodeCatalyst-CloudFormation-action-policy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - 'cloudformation:CreateStack'
                  - 'cloudformation:DeleteStack'
                  - 'cloudformation:Describe*'
```

```
                - 'cloudformation:UpdateStack'
                - 'cloudformation:CreateChangeSet'
                - 'cloudformation:DeleteChangeSet'
                - 'cloudformation:ExecuteChangeSet'
                - 'cloudformation:SetStackPolicy'
                - 'cloudformation:ValidateTemplate'
                - 'cloudformation:List*'
                - 'iam:PassRole'
              Resource: '*'
```

手动为 Web 应用程序蓝图创建角色

CodeCatalyst Web 应用程序蓝图使用您创建的 IAM 角色，这些角色称为 CDK 的构建角色、部署角色和堆栈角色。

按以下步骤操作，在 IAM 中创建角色。

创建构建角色

1. 按如下步骤操作，为角色创建策略：

   a. 登录到 AWS。

   b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   c. 在导航窗格中，选择策略。

   d. 选择创建策略。

   e. 选择 JSON 选项卡。

   f. 删除现有代码。

   g. 粘贴以下代码：

      > **Note**
      >
      > 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
      >
      > ```
      > "Resource": "*"
      > ```

   h. 选择下一步：标签。

   i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-webapp-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 将权限策略附加到构建角色。在添加权限页面上的权限策略部分中，搜索 codecatalyst-webapp-build-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-webapp-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst Web app build role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

3. 将权限策略附加到构建角色，如下所示：

a. 在导航窗格中，选择角色，然后搜索 codecatalyst-webapp-build-role。

b. 选择 codecatalyst-webapp-build-role 以显示其详细信息。

c. 在权限选项卡中，选择添加权限，然后选择附加策略。

d. 搜索 codecatalyst-webapp-build-policy，选中其复选框，然后选择附加策略。

您现在已将权限策略附加到构建角色。构建角色现在具有两个策略：权限策略和信任策略。

4. 按如下步骤操作，获取构建角色 ARN：

   a. 在导航窗格中，选择角色。

   b. 在搜索框中，输入刚创建的角色的名称（`codecatalyst-webapp-build-role`）。

   c. 从列表中选择该角色。

      此时将显示该角色的摘要页面。

   d. 在顶部，复制 ARN 值。

   现在，您已创建具有相应权限的构建角色并已获取其 ARN。

手动为 SAM 蓝图创建角色

SA CodeCatalyst M 蓝图使用您创建的 IAM 角色，分别称为构建角色 CloudFormation和 SA M 的部署角色。

按以下步骤操作，在 IAM 中创建这些角色。

为创建生成角色 CloudFormation

1. 按如下步骤操作，为角色创建策略：

   a. 登录到 AWS。

   b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   c. 在导航窗格中，选择策略。

   d. 选择创建策略。

   e. 选择 JSON 选项卡。

   f. 删除现有代码。

   g. 粘贴以下代码：

      JSON

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

```
            "Action": [
                "s3:*",
                "cloudformation:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-SAM-build-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 将权限策略附加到构建角色。在添加权限页面上的权限策略部分中，搜索 codecatalyst-SAM-build-policy 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-SAM-build-role
```

i. 对于角色描述，输入：

```
CodeCatalyst SAM build role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

3. 将权限策略附加到构建角色，如下所示：

a. 在导航窗格中，选择角色，然后搜索 codecatalyst-SAM-build-role。

b. 选择 codecatalyst-SAM-build-role 以显示其详细信息。

c. 在权限选项卡中，选择添加权限，然后选择附加策略。

d. 搜索 codecatalyst-SAM-build-policy，选中其复选框，然后选择附加策略。

您现在已将权限策略附加到构建角色。构建角色现在具有两个策略：权限策略和信任策略。

4. 按如下步骤操作，获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-SAM-build-role）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。

现在，您已创建具有相应权限的构建角色并已获取其 ARN。

创建适用于 SAM 的部署角色

1. 按如下步骤操作，为角色创建策略：

a. 登录到 AWS。

b. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

c. 在导航窗格中，选择策略。

d. 选择创建策略。

e. 选择 JSON 选项卡。

f. 删除现有代码。

g. 粘贴以下代码：

> **Note**
>
> 第一次使用该角色运行工作流操作时，请在资源策略语句中使用以下通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

h. 选择下一步：标签。

i. 选择下一步：审核。

j. 在名称中，输入：

```
codecatalyst-SAM-deploy-policy
```

k. 选择创建策略。

现在，您已经创建了权限策略。

2. 按如下步骤操作，创建构建角色：

a. 在导航窗格中，选择角色，然后选择创建角色。

b. 选择自定义信任策略。

c. 删除现有的自定义信任策略。

d. 添加以下自定义信任策略：

e. 选择下一步。

f. 将权限策略附加到构建角色。在添加权限页面上的权限策略部分中，搜索 `codecatalyst-SAM-deploy-policy` 并选中其复选框。

g. 选择下一步。

h. 对于角色名称，输入：

```
codecatalyst-SAM-deploy-role
```

i. 对于角色描述，输入：

```
CodeCatalyst SAM deploy role
```

j. 选择创建角色。

现在，您已创建一个具有信任策略和权限策略的构建角色。

3. 将权限策略附加到构建角色，如下所示：

a. 在导航窗格中，选择角色，然后搜索 codecatalyst-SAM-deploy-role。

b. 选择 codecatalyst-SAM-deploy-role 以显示其详细信息。

c. 在权限选项卡中，选择添加权限，然后选择附加策略。

d. 搜索 codecatalyst-SAM-deploy-policy，选中其复选框，然后选择附加策略。

您现在已将权限策略附加到构建角色。构建角色现在具有两个策略：权限策略和信任策略。

4. 按如下步骤操作，获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入刚创建的角色的名称（codecatalyst-SAM-deploy-role）。

c. 从列表中选择该角色。

此时将显示该角色的摘要页面。

d. 在顶部，复制 ARN 值。

现在，您已创建具有相应权限的构建角色并已获取其 ARN。

Amazon CodeCatalyst 的合规性验证

要了解某个 AWS 服务是否在特定合规性计划范围内，请参阅合规性计划范围内的 AWS 服务，然后选择您感兴趣的合规性计划。有关常规信息，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 AWS 服务 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。有关您在使用 AWS 服务时的合规责任的更多信息，请参阅 AWS 安全性文档。

Amazon CodeCatalyst 中的韧性

AWS 全球基础设施围绕 AWS 区域 和可用区而构建。各区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础结构。要了解有关跨 AWS 区域复制哪些 CodeCatalyst 数据的更多信息，请参阅 Amazon 的数据保护 CodeCatalyst。

Amazon CodeCatalyst 中的基础设施安全性

作为一项托管式服务，Amazon CodeCatalyst 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全性。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 CodeCatalyst。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

Amazon CodeCatalyst 中的配置和漏洞分析

配置和 IT 控制是 AWS 和您（我们的客户）之间的共同责任。有关更多信息，请参阅 AWS 责任共担模型。

Amazon CodeCatalyst 中的数据和隐私

Amazon CodeCatalyst 非常重视您的隐私，确保您的信息安全是我们的首要任务。您可以在AWS 隐私声明中详细了解我们如何处理您的信息。

要请求和查看您的数据，请参阅《AWS 一般参考》中的 Requesting your data。

删除 AWS 构建者 ID 个人资料

删除您的个人资料是永久性操作，无法撤消。选择删除后，删除过程将立即开始。Amazon CodeCatalyst 开始删除您的个人资料和所有相关的个人信息。完成此过程需要大约 90 天。

删除了您的个人资料后，您将无法在 Amazon CodeCatalyst 中访问或恢复您的数据。这包括个人访问令牌、角色、用户成员资格以及您作为唯一成员的任何 Amazon CodeCatalyst 空间。您无法再登录 Amazon CodeCatalyst。

有关如何删除您的 AWS 构建者 ID 个人资料的信息，请参阅《AWS 一般参考》中的 Deleting your AWS Builder ID。

Amazon CodeCatalyst 中的工作流操作的最佳实践

在 CodeCatalyst 中开发工作流时，需要考虑大量安全最佳实践。以下是一般指导原则，并不代表完整安全解决方案。这些最佳实操可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

主题

敏感信息

请勿在 YAML 中嵌入敏感信息。建议您使用 CodeCatalyst 密钥，而不是在 YAML 中嵌入凭证、密钥或令牌。利用密钥，可以轻松地在 YAML 中存储和引用敏感信息。

许可条款

请务必注意您选择使用的操作的许可条款。

不受信任的代码

操作通常是独立的、单一用途的模块，可以跨项目、空间或更广泛的社区进行共享。虽然使用来自其他人的代码能够极大地提高便利性和效率，但也会引入新的威胁向量。请查看以下部分，务必遵循最佳实践以确保 CI/CD 工作流的安全。

GitHub Actions

GitHub Action 是开源的，由社区构建和维护。我们采用责任共担模式，将 GitHub Action 源代码视为客户数据，由您负责。您可以授权 GitHub Action 访问密钥、存储库令牌、源代码、账户链接和您的计算时间。请确保计划运行的 GitHub Action 安全可信。

更多适用于 GitHub Action 的具体指导和安全最佳实操，请参阅：

- 安全加固
- 阻止 pwn 请求
- 不可信的输入
- 如何信任您的构建基块

了解 CodeCatalyst 信任模型

Amazon CodeCatalyst 信任模型 CodeCatalyst 允许在互联中扮演服务角色 AWS 账户。该模型将 IAM 角色、 CodeCatalyst 服务委托人和 CodeCatalyst 空间联系起来。信任策略使用`aws:SourceArn`条件密钥向条件键中指定的 CodeCatalyst 空间授予权限。有关此条件密钥的更多信息，请参阅 I A M 用户指南SourceArn中的 aws:。

信任策略位于 JSON 策略文档中，您可以在其中定义您信任代入该角色的主体。角色信任策略是附加在 IAM 角色上的必填基于资源的策略。有关更多信息，请参阅《IAM 用户指南》中的术语和概念。有关服务主体的详细信息 CodeCatalyst，请参阅的服务负责人 CodeCatalyst。

在下面的信任策略中，`Principal` 元素中列出的服务主体从基于资源的策略中获得权限，而 `Condition` 块则用于限制对范围缩小的资源的访问。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/
project/*"
                }
            }
        }
    ]
}
```

在信任策略中， CodeCatalyst 服务委托人通过条件密钥获得访问权限，aws:SourceArn条件密钥包含空间 ID 的 Amazon 资源名称 (ARN)。 CodeCatalyst ARN 使用以下格式：

```
arn:aws:codecatalyst:::space/spaceId/project/*
```

> **Important**
>
> 仅在条件键中使用空间 ID，例如 aws:SourceArn。请勿将 IAM 策略声明中的空间 ID 用作资源 ARN。

最佳做法是在策略中尽可能缩小权限范围。

- 您可以在 aws:SourceArn 条件键中使用通配符（*），用 project/* 指定空间中的所有项目。
- 您可以使用 project/*projectId* 在 aws:SourceArn 条件键中为空间中的特定项目指定资源级权限。

的服务负责人 CodeCatalyst

您可以在基于资源的 JSON 策略中使用 Principal 元素指定允许或拒绝访问资源的主体。您可以在信任策略中指定的主体包括用户、角色、账户和服务。您不能在基于身份的策略中使用 Principal 元素；同样，您也不能在策略（如基于资源的策略）中将用户组标识为主体，因为组与权限有关，与身份验证无关，而主体是经过身份验证的 IAM 实体。

在信任策略 AWS 服务 中，您可以在基于资源的策略的Principal元素或支持委托人的条件密钥中指定。服务主体由服务定义。以下是为定义的服务主体： CodeCatalyst

- codetalyst.amazonaws.com-此服务主体用于授予访问权限的角色。 CodeCatalyst AWS
- codetalyst-runner.amazonaws.com-此服务主体用于授予 CodeCatalyst 工作流程部署中资源访问权限的角色。 AWS CodeCatalyst

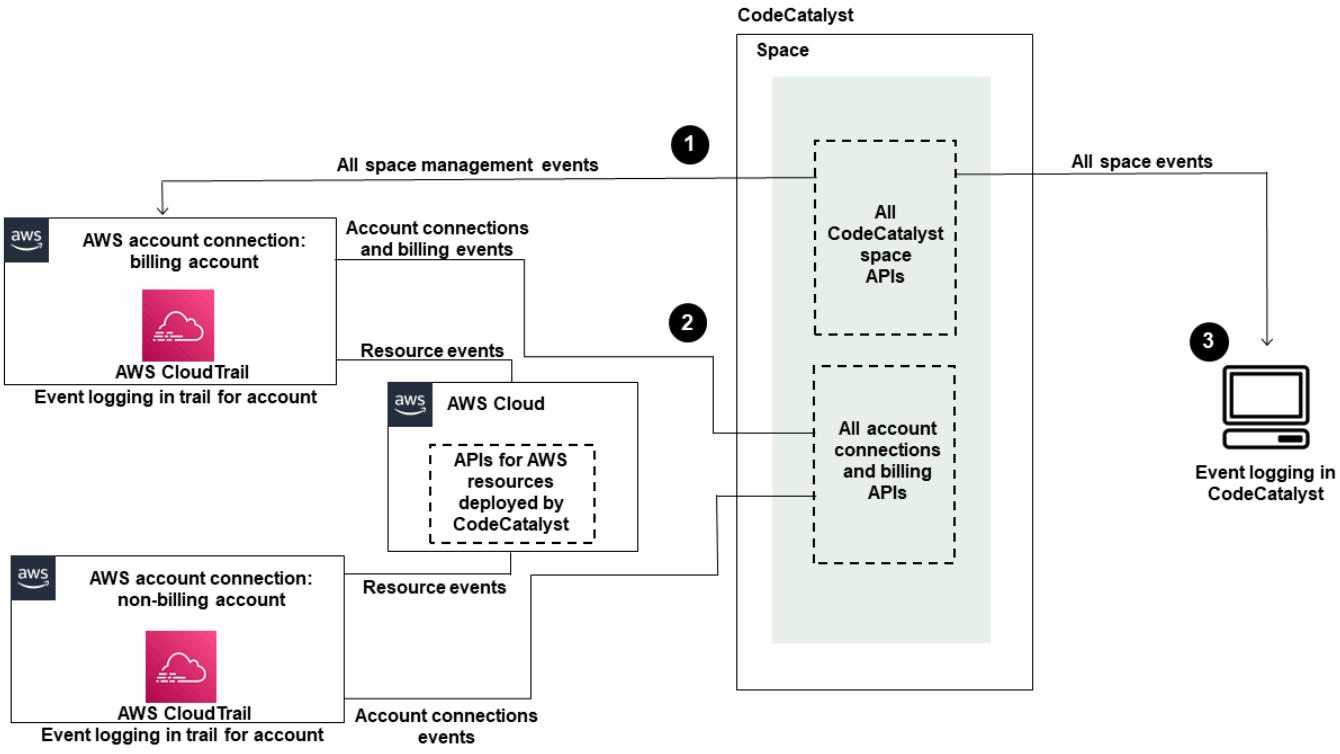
有关更多信息，请参阅《IAM 用户指南》中的 AWS JSON 策略元素：主体。

使用日志记录监控事件和 API 调用

在 Amazon 中 CodeCatalyst，空间的管理事件由该空间的账单账户收集 AWS CloudTrail 并记录在跟踪中。 CloudTrail logging 是管理 CodeCatalyst 事件日志记录的主要方法，次要方法是查看事件日志记录 CodeCatalyst。

账户中的事件使用为 AWS 账户设置的跟踪记录和指定存储桶进行记录。

下图显示了账单账户如何登录 CloudTrail 空间的所有管理事件，而相应账户的账户 connections/billing 事件和 AWS 资源事件是如何登录 CloudTrail 的。



下图说明了以下步骤：

1. 创建空间后，将连接到 AWS 账户 该空间并被指定为结算帐号。使用的跟踪是为账单账户创建的跟踪， CloudTrail 用于记录空间事件。 CloudTrail 捕获空间或代表 CodeCatalyst 空间发出的 API 调用和相关事件，并将日志文件传送到您指定的 S3 存储桶。如果计费账户更改为另一个 AWS 账户，则空间事件将记录在该账户的跟踪记录和存储桶中。有关由记录的 CodeCatalyst 管理事件的更多信息 CloudTrail，请参阅CloudTrail 中的 CodeCatalyst 信息。
2. 与空间关联的其他账户（包括账单账户）会记录账户关联和账单事件的子集。 CodeCatalyst 为为该账户部署的 AWS 资源生成账户事件的工作流程也会记录在的跟踪和存储桶中 AWS 账户。 CloudTrail 捕获空间或代表 CodeCatalyst空间发出的 API 调用和相关事件，并将日志文件传送到您指定的 S3 存储桶。有关由记录的 CodeCatalyst 管理事件的更多信息 CloudTrail，请参阅使用事件日志记录访问已记录的事件。
3. 您还可以使用list-event-logs命令监视空间中特定时间内的 CodeCatalyst 操作 AWS CLI。有关更多信息，请参阅 Amazon CodeCatalyst API 参考指南。您必须具有空间管理员角色才能调用空间中 CodeCatalyst 操作的事件列表。有关更多信息，请参阅 使用事件日志记录访问已记录的事件。

> **Note**
>
> `ListEventLogs` 保证给定空间中最近 30 天的事件。您还可以在 AWS CloudTrail 控制台 CodeCatalyst 中查看和检索过去 90 天的管理事件列表，方法是查看事件历史记录，或者创建跟踪以创建和维护超过 90 天的事件记录。有关更多信息，请参阅使用 CloudTrail 事件历史记录和使用跟 CloudTrail 踪。

> **Note**
>
> AWS 部署到 CodeCatalyst 工作流关联账户中的资源不会作为 CodeCatalyst 空间 CloudTrail 日志记录的一部分进行记录。例如， CodeCatalyst 资源包括空间或项目。 AWS 资源包括亚马逊 ECS 服务或 Lambda 函数。您必须为每个部署资源 AWS 账户 的地方单独配置 CloudTrail 日志记录。

以下是用于监控事件的一种可能流程 CodeCatalyst。

Mary Major 是 CodeCatalyst 空间的空间管理员，可以查看已登录空间中 CodeCatalyst 空间级和项目级资源的所有管理事件。 CloudTrail有关已登录事件CloudTrail 中的 CodeCatalyst 信息的示例，请参阅 CloudTrail。

对于在中创建的资源 CodeCatalyst，例如开发环境，Mary 会查看空间账单账户中的事件历史记录，并调查项目成员在中 CodeCatalyst创建开发环境的事件。该事件为创建开发环境的用户提供身份存储 IAM 身份类型和 AWS 生成器 ID 的证书。对于通过中的工作流程部署 AWS 时在中创建的资源 CodeCatalyst，例如用于无服务器部署的 Lambda 函数， AWS 账户 所有者可以查看与工作流程部署操作的单独账户 AWS 账户 （也是一个关联账户 CodeCatalyst）关联的跟踪的事件历史记录。

为了进一步调查，Mary 还可以使用 CodeCatalyst APIs 中的list-event-logs命令查看空间中所有人的事件 AWS CLI。

主题

使用 AWS CloudTrail 日志记录通过 AWS 账户监控 API 调用

Amazon CodeCatalyst 与 AWS CloudTrail 集成，后者是一项提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将代表 CodeCatalyst 在已连接的 AWS 账户中发出的 API 调用捕获为事件。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 S3 存储桶（包括 CodeCatalyst 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。

CodeCatalyst 支持将以下操作记录为 CloudTrail 日志文件中的事件：

- CodeCatalyst 空间的管理事件将记录在作为该空间的指定计费账户的 AWS 账户中。有关更多信息，请参阅 CodeCatalyst 空间事件。

  > **Note**
  >
  > 可使用 CLI 访问 CodeCatalyst 空间的数据事件，如使用事件日志记录访问已记录的事件中详述。

- 在已连接的 AWS 账户中执行的 CodeCatalyst 工作流操作中使用的资源事件将在该 AWS 账户中记录为事件。有关更多信息，请参阅 CodeCatalyst 账户连接和计费事件。

> **Important**
>
> 虽然可以将多个账户与一个空间关联，但针对 CodeCatalyst 空间和项目中事件的 CloudTrail 记录仅适用于计费账户。

空间计费账户是您的 AWS 账户，将从中扣除 AWS 免费套餐之外的 CodeCatalyst 资源使用费用。可将多个账户连接到一个空间，但只能将一个账户作为指定的计费账户。空间的计费账户或其他已连接账户可以具有 IAM 角色，这些角色用于从 CodeCatalyst 工作流部署 AWS 资源和基础设施，例如 Amazon ECS 集群或 S3 存储桶。您可以使用工作流 YAML 来识别部署到的 AWS 账户。

> **Note**
>
> 部署到 CodeCatalyst 工作流的已连接账户中的 AWS 资源不会记录到 CodeCatalyst 空间的 CloudTrail 日志记录中。例如，CodeCatalyst 资源包括空间或项目。AWS 资源包括 Amazon ECS 服务或 Lambda 函数。必须为将资源部署到其中的每个 AWS 账户单独配置 CloudTrail 日志记录。

已连接的账户中的 CodeCatalyst 日志记录包括以下注意事项：

- 对 CloudTrail 事件的访问将通过已连接账户中的 IAM 管理，而不是在 CodeCatalyst 中进行管理。
- 第三方连接（例如链接到 GitHub 存储库）将促使第三方资源名称被记录在 CloudTrail 日志中。

> **Note**
>
> CodeCatalyst 事件的 CloudTrail 日志记录是在空间级别进行的，并且不会按项目边界隔离事件。

有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html。

> **Note**
>
> 此部分介绍了 CodeCatalyst 空间以及连接到 CodeCatalyst 的 AWS 账户中记录的所有事件的 CloudTrail 日志记录。此外，要查看 CodeCatalyst 空间中记录的所有事件，也可以使用 AWS CLI 和 aws codecatalyst list-event-logs 命令。有关更多信息，请参阅使用事件日志记录访问已记录的事件。

CodeCatalyst 空间事件

CodeCatalyst 中用于管理空间级和项目级资源的操作将记录在空间的计费账户中。对于 CodeCatalyst 空间的 CloudTrail 日志记录，记录事件时要考虑以下注意事项。

- CloudTrail 事件适用于整个空间，而未限定于任一个项目。
- 在将一个 AWS 账户连接到 CodeCatalyst 空间时，账户连接的可记录事件将记录到该 AWS 账户中。此连接一经启用便无法禁用。
- 在将一个 AWS 账户连接到 CodeCatalyst 空间并将其指定为空间的计费账户后，事件将记录到该 AWS 账户中。此连接一经启用便无法禁用。

  空间级和项目级资源的事件仅记录在计费账户中。要更改 CloudTrail 目标账户，请在 CodeCatalyst 中更新计费账户。在下一个月度计费周期开始时，更改将应用于 CodeCatalyst 中的新计费账户。之后，将更新 CloudTrail 目标账户。

以下是 AWS 中的事件的示例，这些事件与 CodeCatalyst 中用于管理空间级和项目级资源的操作相关。以下 API 将通过 SDK 和 CLI 发布。事件将记录在指定为 CodeCatalyst 空间的计费账户的 AWS 账户中。

- CreateDevEnvironment
- CreateProject
- DeleteDevEnvironment
- GetDevEnvironment
- GetProject
- GetSpace
- GetSubscription
- ListDevEnvironments
- ListDevEnvironmentSessions
- ListEventLogs
- ListProjects
- ListSourceRepositories
- StartDevEnvironment
- StartDevEnvironmentSession
- StopDevEnvironment

- StopDevEnvironmentSession
- UpdateDevEnvironment

CodeCatalyst 账户连接和计费事件

以下是 AWS 中的事件的示例，这些事件与 CodeCatalyst 中用于账户连接或计费的操作相关：

- AcceptConnection
- AssociateIAMRoletoConnection
- DeleteConnection
- DissassociateIAMRolefromConnection
- GetBillingAuthorization
- GetConnection
- GetPendingConnection
- ListConnections
- ListIAMRolesforConnection
- PutBillingAuthorization
- RejectConnection

CloudTrail 中的 CodeCatalyst 信息

创建 AWS 账户时将在该账户上启用 CloudTrail。在将该 AWS 账户连接到 CodeCatalyst 空间时，该 AWS 账户中发生的此空间事件将记录在该 AWS 账户的 CloudTrail 日志中。CodeCatalyst 中的可记录事件将与已连接账户中的其他可记录的 AWS 事件一起，作为 CloudTrail 事件记录在该账户的 CloudTrail 日志中以及 CloudTrail 控制台的事件历史记录中。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是否由用户使用 AWS 构建者 ID 发出。
- 请求是使用根用户凭证还是 AWS Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

访问 CloudTrail 事件

要持续记录 AWS 账户中的事件（包括 AWS 账户中的 CodeCatalyst 活动的事件），请创建跟踪记录。通过跟踪记录，CloudTrail 可将日志文件传送至 S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录将适用于所有 AWS 区域。跟踪记录 AWS 分区所有区域的事件，将日志文件传送至指定的 S3 存储桶。此外，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行动。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- Configuring Amazon SNS notifications for CloudTrail
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 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，因此它们不会按任何特定顺序显示。

AWS 中的示例 CodeCatalyst 账户连接事件

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了 `ListConnections` 操作。对于已连接到空间的 AWS 账户，`ListConnections` 用于查看此 AWS 账户与 CodeCatalyst 的所有账户连接。事件将记录在 `accountId` 中指定的 AWS 账户中，`arn` 值将是用于操作的角色的 Amazon 资源名称（ARN）。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "role-ARN",
        "accountId": "account-ID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "arn": "role-ARN",
```

```
                "accountId": "account-ID",
                "userName": "user-name"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-09-06T15:04:31Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-09-06T15:08:43Z",
    "eventSource": "account-ID",
    "eventName": "ListConnections",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.168.0.1",
    "userAgent": "aws-cli/1.18.147 Python/2.7.18 Linux/5.4.207-126.363.amzn2int.x86_64
 botocore/1.18.6",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-ID",
    "eventCategory": "Management"
}
```


AWS 中的示例 CodeCatalyst 项目资源事件

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了 `CreateDevEnvironment` 操作。如果一个 AWS 账户已连接到空间且作为空间的指定计费账户，则可用于空间中的项目级活动，例如创建开发环境。

在 `userIdentity` 下的 `accountId` 字段中，这是托管所有 AWS 构建者 ID 身份的身份池的 IAM Identity Center 账户 ID（`432677196278`）。此账户 ID 包含有关事件的 CodeCatalyst 用户的以下信息。

- `type` 字段表示请求的 IAM 实体的类型。对于空间和项目资源的 CodeCatalyst 事件，此值为 `IdentityCenterUser`。`accountId` 字段指定拥有已用于获取凭证的实体的账户。
- `userId` 字段包含用户的 AWS 构建者 ID 标识符。

- identityStoreArn 字段包含身份存储账户和用户的角色 ARN。

recipientAccountId 字段包含空间计费账户的账户 ID，此处的示例值为 111122223333。

有关更多信息，请参阅 CloudTrail userIdentity 元素。


```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "IdentityCenterUser",
  "accountId": "432677196278",
  "onBehalfOf": {
   "userId": "user-ID",
   "identityStoreArn": "arn:aws:identitystore::432677196278:identitystore/d-9067642ac7"
  },
  "credentialId": "ABCDefGhiJKLMn11Lmn_1AbCDEFgHijk-AaBCdEFGHIjKLmnOPqrs11abEXAMPLE"
 },
 "eventTime": "2023-05-18T17:10:50Z",
 "eventSource": "codecatalyst.amazonaws.com",
 "eventName": "CreateDevEnvironment",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "192.168.0.1",
 "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101
 Firefox/102.0",
 "requestParameters": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
  "ides": [{
   "runtime": "public.ecr.aws/q6e8p2q0/cloud9-ide-runtime:2.5.1",
   "name": "Cloud9"
  }],
  "instanceType": "dev.standard1.small",
  "inactivityTimeoutMinutes": 15,
  "persistentStorage": {
   "sizeInGiB": 16
  }
 },
 "responseElements": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 "
 },
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "eventCategory": "Management"
}
```

> **Note**
>
> 在一些事件中，用户代理可能是未知的。在此情况下，CodeCatalyst 将在 CloudTrail 事件中的 `userAgent` 字段中提供 Unknown 值。

查询 CodeCatalyst 事件跟踪记录

您可以使用 Amazon Athena 中的查询表来创建和管理对 CloudTrail 日志的查询。有关创建查询的更多信息，请参阅《Amazon Athena 用户指南》中的查询 AWS CloudTrail 日志。

使用事件日志记录访问已记录的事件

当用户在 Amazon CodeCatalyst 中执行操作时，这些操作将被记录为事件。您可以使用 AWS CLI 查看指定时段内空间中的事件日志。您可以查看这些事件以审查空间中执行的操作，包括操作的日期和时间、执行操作的用户名以及用户用于发出请求的 IP 地址。

> **Note**
>
> CodeCatalyst 空间的管理事件将记录在已连接的计费账户的 CloudTrail 中。有关 CloudTrail 记录的 CodeCatalyst 管理事件的更多信息，请参阅 CloudTrail 中的 CodeCatalyst 信息。

要查看空间的事件日志，您必须已安装 AWS CLI 并为其配置 CodeCatalyst 的个人资料，并且您必须具有空间的空间管理员角色。有关更多信息，请参阅设置为 AWS CLI 与一起使用 CodeCatalyst和空间管理员角色。

> **Note**
>
> 要查看在已连接的 AWS 账户中代表 CodeCatalyst 执行的事件的日志记录，或查看已连接的计费账户中的空间或项目资源的事件日志记录，您可以使用 AWS CloudTrail。有关更多信息，请参阅 使用 AWS CloudTrail 日志记录通过 AWS 账户监控 API 调用。

1. 打开终端或命令行并运行 aws codecatalyst list-event-logs 命令，同时：

   - 使用 **--space-name** 选项指定空间的名称。
   - 使用 **--start-time** 选项指定要开始审查事件的日期和时间，采用 RFC 3339 中指定的协调世界时（UTC）时间戳格式。
   - 使用 **--end-time** 选项指定要停止审查事件的日期和时间，采用 RFC 3339 中指定的协调世界时（UTC）时间戳格式。
   - （可选）使用 **--max-results** 选项指定要在单个响应中返回的结果的最大数量。如果结果数大于您指定的数目，则响应将包含一个 nextToken 元素，可使用该元素返回后续结果。
   - （可选）使用 **--event-name** 选项将结果限制为希望返回的特定事件类型。

   此示例返回名为 *ExampleCorp* 的空间中 *2022-11-30* 到 *2022-12-01* 期间记录的事件，并且响应中最多返回 *2* 个事件。

   ```
   aws codecatalyst list-event-logs --space-name ExampleCorp --start-time 2022-11-30
    --end-time 2022-12-01 --event-name list-event-logs --max-results 2
   ```

2. 如果事件在该时段内发生，则此命令将返回类似于以下内容的结果：

   ```
   {
       "nextToken": "EXAMPLE",
       "items": [
           {
               "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
               "eventName": "listEventLogs",
               "eventType": "AwsApiCall",
               "eventCategory": "MANAGEMENT",
               "eventSource": "manage",
               "eventTime": "2022-12-01T22:47:24.605000+00:00",
               "operationType": "READONLY",
               "userIdentity": {
                   "userType": "USER",
   ```

```
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111"
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"startTime\":
\"2022-12-01T00:00:00Z\",\"endTime\":\"2022-12-10T00:00:00Z\",\"maxResults\":
\"2\"}"
            },
            "sourceIpAddress": "127.0.0.1",
            "userAgent": "aws-cli/2.9.0 Python/3.9.11 Darwin/21.3.0 exe/x86_64
 prompt/off command/codecatalyst.list-event-logs"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "eventName": "createProject",
            "eventType": "AwsApiCall",
            "eventCategory": "MANAGEMENT",
            "eventSource": "manage",
            "eventTime": "2022-12-01T09:15:32.068000+00:00",
            "operationType": "MUTATION",
            "userIdentity": {
                "userType": "USER",
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111",
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\"}"
            },
            "responsePayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\",\"id\":\"a1b2c3d4-5678-90ab-cdef-
EXAMPLE4444\"}"
            },
            "sourceIpAddress": "192.0.2.23",
            "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0)
 Gecko/20100101 Firefox/102.0"
        }
    ]
```

```
}
```

3. 使用 --next-token 选项和已返回令牌的值再次运行 list-event-logs 命令以检索与请求匹配的下一组记录的事件。

中的身份、权限和访问配额 CodeCatalyst

下表描述了 Amazon 中身份、权限和访问的配额和限制 CodeCatalyst。有关 Amazon 配额的更多信息 CodeCatalyst，请参阅CodeCatalyst 的配额。

资源	信息
中的别名 CodeCatalyst	允许的字符的任意组合，长度在 3 到 100 个字符之间，并且必须以字母开头。有效字符：A-Z、a-z 和 0-9。别名不得： • 少于 3 个字符 • 包含空格或以下任意字符：? ^ * [\ ~ :
一个用户每天发送的最大邀请数	500
每天向一个电子邮件地址发送的最大邀请数	25
每个用户的最大个人访问令牌（PAT）数	100
每个提供商类型在所有空间中每个用户身份（CodeCatalyst别名）的最大个人连接数	1
中的密码 CodeCatalyst	允许的字符的任意组合，长度在 8 到 64 个字符之间。有效字符：A-Z、a-z 和 0-9。您的密码可以包含以下非字母数字字符：(~ ! @ # $ % ^ & * _ - + = ` | \ { } [ ] : ; " ' < > , . ? /)
PAT 名称在 CodeCatalyst	允许的字符的任意组合，长度在 1 到 100 个字符之间
项目成员邀请过期前的时间	24 小时后过期

资源	信息
空间成员邀请过期前的时间	24 小时后过期
电子邮件地址验证过期前的时间	发送 10 分钟后过期

问题排查

本节可以帮助您解决在访问Amazon CodeCatalyst 个人资料时可能遇到的一些常见问题。

注册时出现的问题

您在注册时可能会遇到一些问题。我们提供了一些解决方案。

我的电子邮件地址已在使用中

如果您输入的电子邮件已在使用中，并且您确定该电子邮件是自己的，则您可能已经注册了个人资料。使用此现有身份进行登录。如果您没有现有的电子邮件，请使用另一个未使用的电子邮件地址进行注册。

我无法完成电子邮件验证

如果您尚未收到验证电子邮件

1. 检查您的垃圾邮件文件夹、广告邮件文件夹和已删除邮件文件夹。

   > **Note**
   >
   > 此验证电子邮件的发件地址为 `no-reply@signin.aws` 或 `no-reply@login.awsapps.com`。我们建议您配置自己的电子邮件系统，以便接受来自这些发件人电子邮件地址的电子邮件，而不将其视为广告邮件或垃圾邮件。

2. 等待 5 分钟，然后刷新您的收件箱。再次检查您的垃圾邮件文件夹、广告邮件文件夹和已删除邮件文件夹。
3. 如果仍未看到验证电子邮件，请选择重新发送验证码。如果您已经退出该页面，请重新启动工作流程以注册 Amazon CodeCatalyst。

我的密码未达到最低要求

为了安全起见，您的密码必须包含 8-20 个字符（大写和小写字母以及数字）。

登录时出现的问题

我忘记密码了

按照我忘记密码了中的步骤操作。

我的密码不起作用

无论何时设置或更改密码，您都必须遵循以下要求：

- 密码区分大小写。
- 密码长度必须介于 8 到 64 个字符之间，并且包含大写和小写字母、数字以及至少一个非字母数字字符。
- 您不能重复使用最近三次使用的密码。

我无法启用 MFA

要启用 MFA，请按照将您的 AWS 生成器 ID 配置为使用多重身份验证 (MFA) 登录中的步骤将一个或多个 MFA 设备添加到您的个人资料中。

我无法添加 MFA 设备

如果您发现无法添加其他 MFA 设备，则可能已达到可注册的 MFA 设备的限制。您可能需要先移除现有 MFA 设备，然后才能添加新的 MFA 设备。

我无法移除 MFA 设备

若打算禁用 MFA，请按照 删除 MFA 设备 中的步骤移除您的 MFA 设备。但是，若想保持 MFA 的已启用状态，则应在尝试删除现有 MFA 设备之前添加其他 MFA 设备。有关添加其他 MFA 设备的更多信息，请参阅如何注册设备以便在多重身份验证中使用。

注销时出现的问题

我找不到注销位置

在页面右上角，选择注销。

注销未能将我立即完全注销

系统设计的初衷是立即注销，但完全注销可能需要最多一个小时。

由于工作流失败，我收到一个指示角色不存在的错误

问题：在从 Web 应用程序或无服务器蓝图创建项目后，工作流失败并显示以下错误：

CLIENT_ERROR：角色不存在

可能的解决方案：在配置具有运行工作流的权限的 IAM 角色并将该 IAM 角色添加到工作流 YAML 后，工作流仍失败，因为可能需要将该 IAM 角色添加到账户连接中。将 IAM 角色添加到空间的账户连接中，如将 IAM 角色添加到账户连接中详述。

由于工作流失败，我收到角色错误

问题：在从 Web 应用程序或无服务器蓝图创建项目后，工作流失败并显示以下错误：

CLIENT_ERROR：角色设置不正确或不存在

可能的解决方案：创建项目的空间可能需要设置 AWS 账户 连接或可能需要完成账户连接请求。如果您的空间已具有活跃 AWS 账户 连接，请创建并添加一个具有运行工作流操作的权限的 IAM 角色。将该 IAM 角色添加到账户连接中，如将 IAM 角色添加到账户连接中详述。

可能的解决方案：如果已创建项目但未指定连接，则需要将账户连接与部署环境相关联。如果您的空间已有活动 AWS 账户 连接并添加了 IAM 角色，则必须将与 IAM 角色的账户连接添加到您的部署环境中，详情请参见将账户连接和 IAM 角色添加到部署环境。

我需要在项目工作流中更新 IAM 角色

如果 AWS 账户 连接设置完毕，并且已创建 IAM 角色并将其添加到账户连接中，则可以在项目工作流程中更新 IAM 角色。

1. 选择 CI/CD 选项并选择您的工作流。选择 YAML 按钮。
2. 选择编辑。
3. 在 `ActionRoleArn:` 字段中，将 IAM 角色 ARN 替换为更新后的 IAM 角色 ARN。选择验证。
4. 选择提交。

   如果工作流在主线分支上，则它会自动启动。否则，要重新运行工作流，请选择运行。

在建立个人联系后，我收到了对 GitHub 账户的审核请求

在您与创建个人连接后 GitHub，该 CodeCatalyst 应用程序将作为 GitHub 应用程序安装到您的 GitHub 帐户中。如果其中某些资源需要更新的读取或写入权限，则您可能需要访问自己的 GitHub 账户才能更新已安装应用程序的权限。 CodeCatalyst

1. 登录 GitHub 并导航到已安装应用程序的帐户设置。选择您的个人资料图标，再选择设置，然后选择应用程序。
2. 在 “已安装的 GitHub 应用程序” 选项卡上，在已安装的应用程序列表中，查看已安装的应用程序 CodeCatalyst。如果有审核权限，则将显示审核请求链接。
3. 选择该链接，然后在系统提示时确认您的凭证。输入您的凭证并选择验证。
4. 接受新权限，指明要将这些权限应用于的存储库，然后选择保存。

如何填写支持表单？

您可以前往 Amazon CodeCatalyst 或填写 Su pport 反馈表。在 “请求信息” 部分，在 “我们如何为您提供帮助” 下，说明您是 Amazon CodeCatalyst 客户。请尽量提供详细信息，以便我们能最有效地解决您的问题。

为带有扩展程序的项目添加功能 CodeCatalyst

Amazon CodeCatalyst 包含扩展程序，可帮助您添加功能并与之外的产品集成 CodeCatalyst。使用 CodeCatalyst 目录中的扩展，团队可以在中自定义自己的体验 CodeCatalyst。

主题

可用的第三方扩展

您可以根据您选择与之集成资源的扩展程序为 CodeCatalyst 项目添加特定功能。

将 GitHub 存储库集成到 CodeCatalyst

GitHub 是一项基于云的服务，可帮助开发人员存储和管理他们的代码。GitHub 存储库扩展允许您在 Amazon CodeCatalyst 项目中使用链接 GitHub 存储库。您还可以在创建新 CodeCatalyst 项目时链接 GitHub 存储库。有关更多信息，请参阅 使用链接的第三方存储库创建项目。

> **Note**
>
> - 您不能在 CodeCatalyst 项目中使用空仓库或已存档 GitHub 存储库。
> - GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。

安装和配置GitHub 存储库扩展后，您将能够：

- 在中的源 GitHub 存储库列表中查看您的仓库 CodeCatalyst
- 在存储库中 GitHub 存储和管理工作流程定义文件
- 在 CodeCatalyst 开发环境中创建、读取、更新和删除存储在链接 GitHub 存储库中的文件
- 将链接存储库中的文件 GitHub 存储在中并为其编制索引 CodeCatalyst
- 使用已关联 GitHub 账户的现有存储库创建 CodeCatalyst 项目
- 使用蓝图创建项目或添加蓝图时，使用蓝图生成的代码创建 GitHub 存储库
- 当代码被推送到链接 GitHub 存储库时，或者在链接存储库中创建、修改或关闭拉取请求时，启动 CodeCatalyst 工作流程会自动运行 GitHub
- 在 CodeCatalyst 工作流程中使用链接 GitHub 存储库源文件
- 在 CodeCatalyst 工作流程中读取和执行 GitHub 操作
- 将 CodeCatalyst 工作流程运行状态发送到链接 GitHub 仓库，并根据提交状态阻止 GitHub 拉取请求合并

将 Bitbucket 存储库 CodeCatalyst

Bitbucket 是一项基于云的服务，可帮助开发人员存储和管理他们的代码。Bitbucket 存储库扩展允许您在亚马逊 CodeCatalyst 项目中使用关联的 Bitbucket 存储库。您还可以在创建新 CodeCatalyst 项目时链接 Bitbucket 存储库。有关更多信息，请参阅 使用链接的第三方存储库创建项目。

> **Note**
>
> - 您不能在 CodeCatalyst 项目中使用空的或已存档的 Bitbucket 存储库。
> - Bitbucket 存储库扩展与 Bitbucket Data Center 存储库不兼容。

在安装和配置 Bitbucket 存储库扩展后，您将能够：

- 在中的源存储库列表中查看您的 Bitbucket 存储库 CodeCatalyst
- 在 Bitbucket 存储库中存储和管理工作流定义文件。
- 在 CodeCatalyst 开发环境中创建、读取、更新和删除存储在链接 Bitbucket 存储库中的文件
- 使用已连接 Bitbucket 账户的现有存储库创建 CodeCatalyst 项目
- 将链接的 Bitbucket 存储库中的文件存储在中并为其编制索引 CodeCatalyst
- 在使用蓝图创建项目或添加蓝图时，使用蓝图生成的代码创建 Bitbucket 存储库
- 当代码被推送到链接的 Bitbucket 存储库时，或者在关联的 Bitbucket 存储库中创建、修改或关闭拉取请求时，启动 CodeCatalyst 工作流程会自动运行
- 在工作流程中使用关联的 Bitbucket 存储库源文件 CodeCatalyst
- 将 CodeCatalyst 工作流程运行状态发送到关联的 Bitbucket 存储库，并根据提交状态阻止 Bitbucket 拉取请求合并

将 GitLab 存储库集成到 CodeCatalyst

GitLab 是一项基于云的服务，可帮助开发人员存储和管理他们的代码。GitLab 存储库扩展允许您在 Amazon GitLab 项目中使用链接的 CodeCatalyst 项目存储库。您还可以在创建新 GitLab 项目时链接 CodeCatalyst 项目存储库。有关更多信息，请参阅 使用链接的第三方存储库创建项目。

> **Note**
>
> - 您不能将空的或已存档的 GitLab 项目存储库用于 CodeCatalyst 项目。
> - GitLab 存储库扩展与 GitLab 自行管理的存储库不兼容。

安装和配置GitLab 存储库扩展后，您将能够：

- 在中的源存储库列表中查看您的 GitLab 项目存储库 CodeCatalyst
- 在 GitLab 项目存储库中存储和管理工作流程定义文件。
- 在 CodeCatalyst 开发环境中创建、读取、更新和删除存储在链接 GitLab 项目存储库中的文件
- 使用已连接 GitLab 用户的现有存储库创建 CodeCatalyst 项目
- 将链接的 GitLab 项目存储库中的文件存储在中并为其编制索引 CodeCatalyst
- 使用蓝图创建 GitLab 项目或添加蓝图时，使用蓝图生成的代码创建项目存储库
- 当代码被推送到链接的项目存储库时，或者在链接的 GitLab 项目存储库中创建、修改或关闭拉取请求时，启动 CodeCatalyst 工作流程会自动运行 GitLab

- 在 CodeCatalyst 工作流程中使用链接的 GitLab 项目存储库源文件
- 将 CodeCatalyst 工作流程运行状态发送到链接的 GitLab 项目存储库，并根据提交状态阻止 GitLab 合并请求

将 Jira 事务集成到 CodeCatalyst

Jira 是一款软件应用程序，可帮助敏捷开发团队计划、分配、跟踪、报告和管理工作。Jira Softwar e 扩展允许您在亚马逊 CodeCatalyst 项目中使用 Jira 项目。

> **Note**
>
> CodeCatalyst 仅与 Jira Software Cloud 兼容。

为亚马逊 CodeCatalyst 项目安装和配置 Jira Softwar e 扩展程序后，您将能够：

- 通过将 Jira 项目链接到项目 CodeCatalyst 来访问这些项目 CodeCatalyst
- 使用拉 CodeCatalyst 取请求更新 Jira 问题
- 在 Jira 事务中查看关联 CodeCatalyst 拉取请求的状态和工作流程运行情况

扩展概念

以下是在中使用扩展时需要了解的一些概念和术语 CodeCatalyst。

扩展程序

扩展程序是一种附加组件，您可以将其安装到您的 CodeCatalyst 空间中，以便为项目添加新功能并与之外的服务集成 CodeCatalyst。可以从 CodeCatalyst 目录中浏览和安装扩展。

CodeCatalyst 目录

该 CodeCatalyst 目录集中列出了中所有可用的扩展 CodeCatalyst。您可以浏览 CodeCatalyst 目录以查找可以改善团队在源代码、工作流程 CodeCatalyst 等领域的体验的扩展。

连接和链接

根据您要使用或管理的第三方资源，您需要关联 GitHub 账户、Bitbucket 工作空间或 Jira 项目。然后，你需要将你的 GitHub 存储库、Bitbucket 存储库或 Jira 项目链接到你的 CodeCatalyst 项目。

- GitHub 存储库：Connect GitHub 帐户，然后关联 GitHub 存储库。
- Bitbucket 存储库：连接 Bitbucket 工作区，然后链接 Bitbucket 存储库。
- GitLab 存储库：连接 GitLab 用户，然后链接 GitLab 项目存储库。
- Jira Software：连接 Jira 站点，然后链接 Jira 项目。

快速入门：安装扩展、连接提供商和链接资源 CodeCatalyst

本教程演练了以下三个任务：

1. 安装GitHub 存储库、Bitbucket 存储GitLab 库、存储库或 Jira Software 扩展程序。在外部站点中，系统会提示您连接并 CodeCatalyst 提供对第三方资源的访问权限，这将在下一步中完成。

> **Important**
>
> 要将GitHub 存储库、Bitbucket GitLab 存储库、存储库或 Jira Softwar e 扩展程序安装到您的 CodeCatalyst 空间，您必须使用在该空间中具有空间管理员角色的账户登录。

2. 将您的 GitHub 账户、Bitbucket 工作空间、 GitLab 用户或 Jira 网站连接到。 CodeCatalyst

> **Important**
>
> 要将您的 GitHub 账户、Bitbucket 工作空间、 GitLab 用户或 Jira 站点连接到您的 CodeCatalyst 空间，您必须同时是第三方来源的管理员和 CodeCatalyst 空间管理员。

> **Important**
>
> 安装存储库扩展后，您链接到的任何存储库都 CodeCatalyst 将对其代码进行索引和存储。CodeCatalyst这将使代码可在中 CodeCatalyst搜索。要更好地了解在中使用链接存储库时代码的数据保护 CodeCatalyst，请参阅 Amazon CodeCatalyst 用户指南中的数据保护。

> **Note**
>
> 如果您使用的是 GitHub 账户连接，则必须创建个人连接才能在您的身份和身份之间建立 CodeCatalyst 身份映射。 GitHub 有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

3. 将您的 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库或 Jira 项目链接到您的 CodeCatalyst 项目。

> **Important**
>
> - 虽然您可以作为贡献者链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目仓库，但您只能以 Space 管理员或项目管理员的身份取消第三方仓库的链接。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
> - 要将您的 Jira 项目链接到您的 CodeCatalyst 项目，您必须是 CodeCatalyst Space 管理员或 CodeCatalyst 项目管理员。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

> **Note**
>
> - GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库只能链接到空间中的一个 CodeCatalyst 项目。
> - 您不能将空仓库或已存档 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库与 CodeCatalyst 项目一起使用。

- 您不能链接与 GitLab 项目中 GitHub 仓库同名的存储库、Bitbucket 存储库或 CodeCatalyst 项目存储库。
- GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。
- Bitbucket 存储库扩展与 Bitbucket Data Center 存储库不兼容。
- GitLab 存储库扩展与 GitLab 自行管理的项目存储库不兼容。
- 您无法在已链接存储库中使用为我编写描述或汇总评论功能。这些功能仅在中的拉取请求中可用 CodeCatalyst。
- 一个 CodeCatalyst 项目只能链接到一个 Jira 项目。一个 Jira 项目可以链接到多个 CodeCatalyst 项目。

创建新 CodeCatalyst 项目时，您还可以安装GitHub 存储库、Bitbuck et GitLab 存储库、存储库扩展、连接到您的 GitHub 账户、Bitbucket 工作空间或 GitLab 用户，以及链接第三方存储库。有关更多信息，请参阅 使用链接的第三方存储库创建项目。

主题

步骤 1：从 CodeCatalyst 目录中安装第三方扩展

在中使用第三方资源的第一步 CodeCatalyst 是从目录中安装GitHub 存储库扩展。 CodeCatalyst 要安装扩展程序，请执行以下步骤，为要使用的第三方资源选择扩展程序。 GitHub 存储库、Bitbucket 存储GitLab 库和存储库允许您在中使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库。 CodeCatalystJira Softwar e 允许您在中管理 Jira 问题。 CodeCatalyst

从 CodeCatalyst 目录中安装扩展

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 通过选择顶部菜

   单 中

的目录图标来导航到目录。 CodeCatalyst CodeCatalyst 您可以搜索GitHub 存储库、Bitbucket 存储GitLab 库、存储库或 Jira So ftware。您也可以根据类别筛选扩展。

4. （可选）要查看有关扩展的更多详细信息（例如扩展将具有的权限），请选择扩展名。
5. 选择安装。查看扩展所需的权限，如果要继续，请再次选择安装。

安装扩展后，您将进入扩展详细信息页面。根据已安装的扩展，您可以查看和管理连接的提供程序和链接的资源。

第 2 步：将您的第三方提供商连接到您的 CodeCatalyst 空间

安装GitHub 存储库、Bitbucket 存储GitLab 库、存储库或 Jira Softwar e 扩展程序后，下一步是将您的 GitHub 账户、Bitbucket 工作空间、 GitLab 项目存储库或 Jira 站点连接到您的空间。 CodeCatalyst

将您的 GitHub 账户、Bitbucket 工作空间或 Jira 网站连接到 CodeCatalyst

- 根据已安装的第三方扩展，执行下列操作之一：
  - GitHub 存储库：Connect 连接到 GitHub 帐户。
    1. 在 “已连接的 GitHub 帐户” 选项卡中，选择 Connect GitHub 帐户以转到外部站点 GitHub。
    2. 使用您的 GitHub 凭证登录您的 GitHub 账户，然后选择要安装Amazon的账户 CodeCatalyst。

       > **Tip**
       >
       > 如果您之前已将 GitHub 账户关联到该空间，则系统不会提示您重新授权。相反，如果您是多个组织的成员或合作者，则会看到一个对话框，询问您要在哪里安装扩展程序；如果您只属于一个 GitHub 组织，则会看到Amazon CodeCatalyst 应用程序的配置页面。 GitHub 为要允许的存储库访问权限配置应用程序，然后选择保存。如果保存按钮未激活，请更改配置，然后重试。

    3. 选择是 CodeCatalyst 允许访问所有当前和将来的存储库，还是选择要在中使用的特定 GitHub 存储库 CodeCatalyst。默认选项是将 GitHub 账户中的所有 GitHub 存储库包括在内，包括将由访问的 future 存储库 CodeCatalyst。
    4. 查看授予的权限 CodeCatalyst，然后选择 “安装”。

将您的 GitHub 账户关联到后 CodeCatalyst，系统会将您带到GitHub 存储库扩展详情页面，您可以在其中查看和管理关联的 GitHub 账户和关联的 GitHub 存储库。

- Bitbucket 存储库：连接到 Bitbucket 工作区。

  1. 在已连接的 Bitbucket 工作区选项卡中，选择连接 Bitbucket 工作区以转到 Bitbucket 的外部站点。
  2. 使用您的 Bitbucket 凭据登录您的 Bitbucket 工作空间并查看授予的权限。 CodeCatalyst
  3. 从 “授权工作空间” 下拉菜单中，选择要提供 CodeCatalyst 访问权限的 Bitbucket 工作空间，然后选择 “授予访问权限”。

     > **Tip**
     >
     > 如果您之前已将 Bitbucket 工作区连接到空间，则系统不会提示您重新授权。相反，如果您是多个 Bitbucket 工作空间的成员或合作者，则会看到一个对话框，询问您要在哪里安装扩展程序；如果您只属于一个 Bitbucket 工作空间，则会看到亚马逊 CodeCatalyst 应用程序的配置页面。为要允许的工作区访问权限配置应用程序，然后选择授予访问权限。如果授予访问权限按钮未激活，请更改配置，然后重试。

  将 Bitbucket 工作空间连接到后 CodeCatalyst，您将被带到 Bitbucket 存储库扩展详细信息页面，您可以在其中查看和管理已连接的 Bitbucket 工作空间和关联的 Bitbucket 存储库。

- GitLab 存储库：Connect 连接到 GitLab 用户。

  1. 选择 Connect GitLab 用户以访问外部站点 GitLab。
  2. 使用您的 GitLab 证书登录您的 GitLab 用户并查看授予的权限 CodeCatalyst。

     > **Tip**
     >
     > 如果您之前已将 GitLab 用户连接到空间，则系统不会提示您重新授权。相反，您将被导航回 CodeCatalyst 控制台。

  3. 选择 “为 AWS 连接器授权” GitLab。

将 GitLab 用户连接到后 CodeCatalyst，系统会将您带到GitLab 存储库扩展详细信息页面，您可以在其中查看和管理关联的 GitLab 用户和关联的 GitLab 项目存储库。

- Jira Software：连接 Jira 站点。

  1. 在已连接的 Jira 站点选项卡中，选择连接 Jira 站点以转到 Atlassian Marketplace 的外部站点。
  2. 选择 “立即获取”，开始在 Jira 网站上 CodeCatalyst 进行安装。

     > **Note**
     >
     > 如果您之前已安装 CodeCatalyst 到 Jira 站点，则会收到通知。选择开始使用以入最后一步。

  3. 根据您的角色，请执行下列操作之一：

     1. 如果您是 Jira 站点管理员，请从站点下拉菜单中选择要安装应用程序的 Jira 站点，然后选择安装 CodeCatalyst 应用程序。

        > **Note**
        >
        > 如果您有一个 Jira 站点，则不会显示此步骤，系统会自动将您转到下一步。

     2. a. 如果您不是 Jira 管理员，请从站点下拉菜单中选择要安装应用程序的 Jira 站点，然后选择请求 CodeCatalyst 应用程序。有关安装 Jira 应用程序的更多信息，请参阅 Who can install apps?。

        b. 在输入文本字段中 CodeCatalyst 输入需要安装的原因或保留默认文本，然后选择提交请求。

  4. 查看安装应用程序 CodeCatalyst 时所执行的操作，然后选择 “立即获取”。
  5. 安装应用程序后，选择 “返回到” CodeCatalyst 以返回到 CodeCatalyst。

  将 Jira 站点连接到后 CodeCatalyst，您可以在 Jira So f tware 扩展详细信息页面的 “已连接 Jira 站点” 选项卡中查看已连接的站点。

第 3 步：将您的第三方资源链接到您的 CodeCatalyst 项目

使用您的 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库或在中管理 Jira 事务的第三步也是最后一步 CodeCatalyst 是将它们链接到您要在其中使用它的 CodeCatalyst 项目。

从扩展详细信息页面将 GitHub 存储库、Bitbucket 存储库、项目存储库或 Jira CodeCatalyst 项目链接到项目 GitLab

- 根据已安装的第三方扩展和已连接的提供程序，执行下列操作之一：
  - GitHub 存储库：链接存储 GitHub 库。
    1. 在 “链接 GitHub 存储库” 选项卡中，选择 “链接 GitHub 存储库”。
    2. 从GitHub 账户下拉列表中，选择包含您要关联的存储库的 GitHub 账户。
    3. 从GitHub 存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

       > **Tip**
       >
       > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

    4. （可选）如果您在 GitHub 存储库列表中看不到存储库，则可能未在的 Amazon CodeCatalyst 应用程序中将其配置为可以访问存储库 GitHub。您可以配置可在关联账户 CodeCatalyst 中使用哪些 GitHub 存储库。
       a. 导航到您的GitHub帐户，选择 “设置”，然后选择 “应用程序”。
       b. 在 “已安装的 GitHub 应用程序” 选项卡中，选择 Amazon CodeCatalyst 应用程序的 “配置”。
       c. 执行以下任一操作来配置要链接的 GitHub 存储库的访问权限 CodeCatalyst：
          - 要提供对所有当前和未来存储库的访问权限，请选择所有存储库。
          - 要提供对特定存储库的访问权限，请选择 “仅选择存储库”，选择 “选择存储库” 下拉列表，然后选择要允许链接的存储库 CodeCatalyst。
    5. 从CodeCatalyst 项目下拉菜单中，选择要将 GitHub 存储库链接到的 CodeCatalyst 项目。
    6. 选择链接。

如果您不想再在中使用 GitHub 存储库 CodeCatalyst，可以取消它与 CodeCatalyst 项目的链接。解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法

在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- Bitbucket 存储库：链接 Bitbucket 存储库。

  1. 在已链接的 Bitbucket 存储库选项卡中，选择链接 Bitbucket 存储库。
  2. 从 Bitbucket 工作区下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作区。
  3. 从 Bitbucket 存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

     > **Tip**
     >
     > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

  4. 从CodeCatalyst 项目下拉菜单中，选择要将 Bitbucket 存储库关联到的 CodeCatalyst 项目。
  5. 选择链接。

  如果您不想再在中使用 Bitbucket 存储库 CodeCatalyst，可以取消该存储库与项目的链接。CodeCatalyst 解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- GitLab 存储库：链接 GitLab 项目存储库。

  1. 在 “链接的 GitLab 项目存储库” 选项卡中，选择 “链接 GitLab 项目存储库”。
  2. 从GitLab 用户下拉列表中，选择包含要链接的存储库的 GitLab 用户。
  3. 从GitLab 项目存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

     > **Tip**
     >
     > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

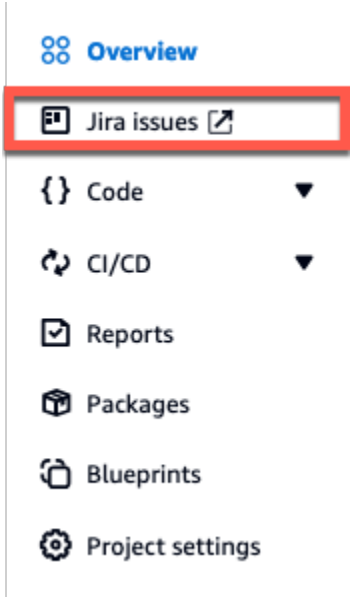
  4. 从CodeCatalyst 项目下拉菜单中，选择要将 CodeCatalyst 项目存储库链接到的 GitLab 项目。
  5. 选择链接。

如果您不想再在中使用 GitLab 项目存储库 CodeCatalyst，则可以取消其与 CodeCatalyst 项目的链接。取消链接项目存储库后，该项目存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该项目存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- Jira Software：链接 Jira 项目。

  1. 在已链接的 Jira 项目选项卡中，选择链接 Jira 项目。
  2. 从 Jira 站点下拉菜单中，选择包含要链接的项目的 Jira 站点。
  3. 从 Jira 项目下拉菜单中，选择要链接到 CodeCatalyst 项目的项目。
  4. 从CodeCatalyst 项目下拉菜单中，选择要链接到 Jira 项目的项目。 CodeCatalyst
  5. 选择链接。

  将 Jira 项目链接到 CodeCatalyst 项目后，将完全禁用对 CodeCatalyst 议题的访问权限，CodeCatalyst 导航窗格中的议题将被链接到 Jira 项目的 Jira 议题项所取代。

  如果您不想再在中使用 Jira 项目 CodeCatalyst，可以取消其与项目的关联。 CodeCatalyst当 Jira 项目取消关联后，Jira 事务将不可用于该 CodeCatalyst项目，而 CodeCatalyst 问题将再次成为议题提供者。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

您还可以在 C od e 中将您的 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库链接到源存储库中的项目。有关更多信息，请参阅 从已连接的第三方提供程序链接资源。

后续步骤

安装GitHub 存储库、Bitbuck et GitLab 存储库或存储库扩展、连接资源提供商并将第三方存储库与 CodeCatalyst 项目关联后，您可以在 CodeCatalyst 工作流程和开发环境中使用它。您还可以在关联的 GitHub 账户、Bitbucket 工作区或 GitLab 使用蓝图生成的代码的用户中创建第三方存储库。有关更多信息，请参阅在第三方存储库事件发生后自动启动工作流运行和创建开发环境。

安装 Jira Software 扩展程序、连接您的 Jira 站点、将 Jira 项目链接到您的 CodeCatalyst 项目以及关联拉取请求后，来自的更新将反映在您 CodeCatalyst 的 Jira 项目中。有关将拉取请求链接到 Jira 事务的更多信息，请参阅将 Jira 事务与拉 CodeCatalyst 取请求相关联。有关在 Jira 中查看 CodeCatalyst 事件的更多信息，请参阅在 Jira 事务中查看 CodeCatalyst 事件。

在空间中安装扩展

您可以为空间安装扩展，为该 CodeCatalyst 空间中的项目添加功能。您可以通过选择 CodeCatalyst 目录图标来查看目

录

。

要了解有关扩展及其功能的更多信息，请参阅可用的第三方扩展。

> **Important**
>
> 要安装扩展，您必须使用在空间中具有空间管理员角色的账户登录。

> **Important**
>
> 安装存储库扩展后，您链接到的任何存储库都 CodeCatalyst 将对其代码进行索引和存储。CodeCatalyst这将使代码可在中 CodeCatalyst搜索。要更好地了解在中使用链接存储库时代码的数据保护 CodeCatalyst，请参阅 Amazon CodeCatalyst 用户指南中的数据保护。

从 CodeCatalyst 目录中安装扩展

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到您的 CodeCatalyst 空间。
3. 通过选择顶部菜单  中的目录图标来导航到目录。 CodeCatalyst 您可以搜索GitHub 存储库、Bitbucket 存储GitLab 库、存储库或 Jira So ftware。您也可以根据类别筛选扩展。
4. （可选）选择扩展的名称以查看有关扩展的更多详细信息（例如扩展将具有的权限）。
5. 选择安装。查看扩展所需的权限，如果要继续，请再次选择安装。

安装扩展后，您将看到已安装扩展的详细信息页面。浏览选项卡以查看有关扩展的更多信息。如果需要，您还可以在详细信息页面上进一步配置扩展。

卸载空间中的扩展

您可以卸载以前安装在您 CodeCatalyst 空间中的扩展程序。卸载扩展程序可能会从您的 CodeCatalyst 空间或项目中移除与该扩展程序相关的资源。

> **Important**
>
> 要卸载扩展，您必须使用在空间中具有空间管理员角色的账户登录。

从您的 CodeCatalyst 空间中卸载扩展程序

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下操作之一，以查看您空间已安装的扩展列表：
   a. 选择设置，然后选择已安装的扩展。
   b. 在顶部菜单中，选择目录图标  。
4. 在要卸载的扩展上选择配置。
5. 在扩展详细信息页面上选择卸载。
6. 查看卸载扩展对话框中的信息。按照说明进行操作，然后选择卸载来卸载扩展。

关联 GitHub 账户、Bitbucket 工作空间、 GitLab 用户和 Jira 站点 CodeCatalyst

要使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库或在中管理 Jira 项目 CodeCatalyst，必须先将第三方来源连接到您的 CodeCatalyst 空间。要了解有关扩展及其功能的更多信息，请参阅可用的第三方扩展。

Important

要将您的 GitHub 账户、Bitbucket 工作空间、 GitLab 用户或 Jira 站点连接到您的 CodeCatalyst 空间，您必须同时是第三方来源的管理员和 CodeCatalyst 空间管理员。

Note

如果您使用的是 GitHub 账户连接，则必须创建个人连接才能在您的身份和身份之间建立 CodeCatalyst 身份映射。 GitHub 有关更多信息，请参阅个人连接和通过人际关系访问 GitHub 资源。

要将您的 GitHub 账户、Bitbucket 工作空间、 GitLab 用户或 Jira 网站连接到 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下操作之一，以查看您空间已安装的扩展列表：

   a. 选择设置，然后选择已安装的扩展。

   b. 在顶部菜单中，选择目录图标

   。

4. 为要配置的以下扩展程序之一选择 “配置”：GitHub 存储库、Bitbucket 存储库、存储库或 Jira Software。GitLab
5. 根据您选择配置的第三方扩展执行下列操作之一：

   - GitHub 存储库：Connect 连接到 GitHub 帐户。

     1. 在 “已连接的 GitHub 帐户” 选项卡中，选择 Connect GitHub 帐户以转到外部站点 GitHub。

2. 使用您的 GitHub 凭证登录您的 GitHub 账户，然后选择要安装Amazon的账户 CodeCatalyst。

> **Tip**
>
> 如果您之前已将 GitHub 账户关联到该空间，则系统不会提示您重新授权。相反，如果您是多个空间的成员或合作者，则会看到一个对话框询问您要在哪里安装扩展程序；如果您只属于一个 GitHub 空间，则会看到Amazon CodeCatalyst 应用程序的配置页面。 GitHub 为要允许的存储库访问权限配置应用程序，然后选择保存。如果保存按钮未激活，请更改配置，然后重试。

3. 选择是 CodeCatalyst 允许访问所有当前和将来的存储库，还是选择要在中使用的特定 GitHub 存储库 CodeCatalyst。默认选项是将 GitHub 账户中的所有 GitHub 存储库包括在内，包括将由访问的 future 存储库 CodeCatalyst。
4. 查看授予的权限 CodeCatalyst，然后选择 “安装”。

将您的 GitHub 账户关联到后 CodeCatalyst，系统会将您带到GitHub 存储库扩展详情页面，您可以在其中查看和管理关联的 GitHub 账户和关联的 GitHub 存储库。

- Bitbucket 存储库：连接到 Bitbucket 工作区。

1. 在已连接的 Bitbucket 工作区选项卡中，选择连接 Bitbucket 工作区以转到 Bitbucket 的外部站点。
2. 使用您的 Bitbucket 凭据登录您的 Bitbucket 工作空间并查看授予的权限。 CodeCatalyst
3. 从 “授权工作空间” 下拉菜单中，选择要提供 CodeCatalyst 访问权限的 Bitbucket 工作空间，然后选择 “授予访问权限”。

> **Tip**
>
> 如果您之前已将 Bitbucket 工作区连接到空间，则系统不会提示您重新授权。相反，如果您是多个 Bitbucket 工作空间的成员或合作者，则会看到一个对话框，询问您要在哪里安装扩展程序；如果您只属于一个 Bitbucket 工作空间，则会看到亚马逊 CodeCatalyst 应用程序的配置页面。为要允许的工作区访问权限配置应用程序，然后选择授予访问权限。如果授予访问权限按钮未激活，请更改配置，然后重试。

将 Bitbucket 工作空间连接到后 CodeCatalyst，您将被带到 Bitbucket 存储库扩展详细信息页面，您可以在其中查看和管理已连接的 Bitbucket 工作空间和关联的 Bitbucket 存储库。

- GitLab 存储库：Connect 连接到 GitLab 用户。

  1. 选择 Connect GitLab 用户以访问外部站点 GitLab。
  2. 使用您的 GitLab 证书登录您的 GitLab 用户并查看授予的权限 CodeCatalyst。

     > **Tip**
     >
     > 如果您之前已将 GitLab 用户连接到空间，则系统不会提示您重新授权。相反，您将被导航回 CodeCatalyst 控制台。

  3. 选择 “为 AWS 连接器授权” GitLab。

  将 GitLab 用户连接到后 CodeCatalyst，系统会将您带到GitLab 存储库扩展详细信息页面，您可以在其中查看和管理关联的 GitLab 用户和关联的 GitLab 项目存储库。

- Jira Software：连接 Jira 站点。

  1. 在已连接的 Jira 站点选项卡中，选择连接 Jira 站点以转到 Atlassian Marketplace 的外部站点。
  2. 选择 “立即获取”，开始在 Jira 网站上 CodeCatalyst 进行安装。

     > **Note**
     >
     > 如果您之前已安装 CodeCatalyst 到 Jira 站点，则会收到通知。选择开始使用以入最后一步。

  3. 根据您的角色，请执行下列操作之一：

     1. 如果您是 Jira 站点管理员，请从站点下拉菜单中选择要安装应用程序的 Jira 站点，然后选择安装 CodeCatalyst 应用程序。

        > **Note**
        >
        > 如果您有一个 Jira 站点，则不会显示此步骤，系统会自动将您转到下一步。

2. a. 如果您不是 Jira 管理员，请从站点下拉菜单中选择要安装应用程序的 Jira 站点，然后选择请求 CodeCatalyst 应用程序。有关安装 Jira 应用程序的更多信息，请参阅 Who can install apps?。

   b. 在输入文本字段中 CodeCatalyst 输入需要安装的原因或保留默认文本，然后选择提交请求。

4. 查看安装应用程序 CodeCatalyst 时所执行的操作，然后选择 “立即获取”。
5. 安装应用程序后，选择 “返回到” CodeCatalyst 以返回到 CodeCatalyst。

将 Jira 站点连接到后 CodeCatalyst，您可以在 Jira So f tware 扩展详细信息页面的 “已连接 Jira 站点” 选项卡中查看已连接的站点。

如果您不想再使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，也不想在中管理 Jira 事务 CodeCatalyst，则可以断开第三方来源的连接。当 GitHub 账户、Bitbucket 工作空间或 GitLab 用户断开连接时，第三方存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用这些存储库。当 Jira 站点断开连接时，来自该站点项目的 Jira 事务将无法在 CodeCatalyst 项目中使用，而 CodeCatalyst 问题将再次成为议题提供者。有关更多信息，请参阅 断开 GitHub 账户、Bitbucket 工作空间、 GitLab 用户和 Jira 网站的连接 CodeCatalyst。

断开 GitHub 账户、Bitbucket 工作空间、 GitLab 用户和 Jira 网站的连接 CodeCatalyst

如果您不想再使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，也不想在中管理 Jira 事务 CodeCatalyst，则可以断开第三方来源的连接。一旦 GitHub 账户、Bitbucket 工作空间或 GitLab 用户断开连接，存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用这些存储库。当 Jira 站点断开连接时，来自该站点项目的 Jira 事务将无法在 CodeCatalyst 项目中使用，而 CodeCatalyst 问题将再次成为议题提供者。

> **Note**
>
> - 要断开 GitHub 账户的连接，必须先取消所有关联 GitHub 仓库与该账户的关联。
> - 要断开某个 Bitbucket 工作区的连接，您必须先取消所有链接的 Bitbucket 存储库与该工作区的链接。
> - 要断开 GitLab 用户的连接，必须先取消所有链接的 GitLab 项目存储库与该工作区的链接。
> - 要断开某个 Jira 站点的连接，必须先取消所有链接的 Jira 项目与该账户的链接。

有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

断开 GitHub 项目、Bitbucket 工作空间、 GitLab 用户或 Jira 网站的连接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下操作之一，以查看您空间已安装的扩展列表：

   a. 选择设置，然后选择已安装的扩展。

   b. 在顶部菜单中，选择目录图标

   。

4. 为要配置的以下扩展程序之一选择 “配置”：GitHub 存储库、Bitbucket 存储库、存储库或 Jira Software。GitLab
5. 根据您选择配置的第三方扩展执行下列操作之一：

   - GitHub 存储库：断开与帐户的连接。 GitHub

     在 “已连接的 GitHub 帐户” 选项卡中，选择要断开连接的 GitHub 帐户，然后选择 “断开 GitHub 帐户”。

   - Bitbucket 存储库：断开与 Bitbucket 工作区的连接。

     在已连接的 Bitbucket 工作区选项卡中，选择要断开连接的 Bitbucket 工作区，然后选择断开 Bitbucket 工作区的连接。

   - GitLab 存储库：断开与用户的连接。 GitLab

     在 “已连接的 GitLab 用户” 选项卡中，选择要断开连接的 GitLab 用户，然后选择 “断开 GitLab 用户连接”。

   - Jira Software：断开与 Jira 站点的连接。

     在已连接的 Jira 站点选项卡中，选择要断开连接的 Jira 站点，然后选择断开 Jira 站点的连接。

6. 在断开连接对话框中，查看断开账户的连接所带来的影响。
7. 在文本输入字段中输入 disconnect，然后选择断开连接。

在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst

在使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库或管理 Jira 项目之前，必须将存储库或项目所属的第三方源与您的 CodeCatalyst 空间连接起来。有关更多信息，请参阅 关联 GitHub 账户、Bitbucket 工作空间、 GitLab 用户和 Jira 站点 CodeCatalyst。

您可以在工作流程中使用链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，其中链接存储库中的事件会启动可能构建、测试或部署代码的工作流程，具体取决于工作流程配置。使用链接存储库 GitHub 或 Bitbucket 存储库的工作流程配置文件存储在链接存储库中。此外，可以将已链接存储库与开发环境结合使用，来创建、更新和删除已链接存储库中的文件。您可以从 GitHub 存储库、Bitbucket 存储库或 GitLab 存储库扩展的详细信息页面，或者从 CodeCatalyst 项目本身的 “代码” 中的 “源存储库” 视图将GitLab 存储库、Bitbuck et 存储库或项目存储库链接到项目。GitHub

> **Important**
>
> 虽然您可以以贡献者的身份链接 GitHub 或 Bitbucket 存储库，但您只能以 Space 管理员或项目管理员的身份取消第三方仓库的链接。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

> **Important**
>
> 安装存储库扩展后，您链接到的任何存储库都 CodeCatalyst 将对其代码进行索引和存储。CodeCatalyst这将使代码可在中 CodeCatalyst搜索。要更好地了解在中使用链接存储库时代码的数据保护 CodeCatalyst，请参阅 Amazon CodeCatalyst 用户指南中的数据保护。

> **Important**
>
> CodeCatalyst 不支持检测链接仓库的默认分支中的更改。要更改链接存储库的默认分支，必须先取消其与该分支的链接 CodeCatalyst，更改默认分支，然后再次进行链接。
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

您可以使用关联的 Jira 项目来管理议题并将 CodeCatalyst 拉取请求链接到 Jira 事务。拉取请求的摘要状态和关联 CodeCatalyst 的工作流事件的状态会反映在您的 Jira 事务中。

Important

要将您的 Jira 项目链接到您的 CodeCatalyst 项目，您必须是 CodeCatalyst Space 管理员或 CodeCatalyst 项目管理员。

Note

- GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库只能链接到空间中的一个 CodeCatalyst 项目。
- 您不能将空仓库或已存档 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库与 CodeCatalyst 项目一起使用。
- 您不能链接与 CodeCatalyst 项目中 GitHub 仓库同名的存储库、Bitbucket 存储库或 GitLab 存储库。
- GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。
- Bitbucket 存储库扩展与 Bitbucket Data Center 存储库不兼容。
- GitLab 存储库扩展与 GitLab 自行管理的项目存储库不兼容。
- 您无法在已链接存储库中使用为我编写描述或汇总评论功能。这些功能仅在中的拉取请求中可用 CodeCatalyst。
- 一个 CodeCatalyst 项目只能链接到一个 Jira 项目。一个 Jira 项目可以链接到多个 CodeCatalyst 项目。

主题

从已连接的第三方提供程序链接资源

从扩展详细信息页面将 GitHub 存储库、Bitbucket 存储库、项目存储库或 Jira CodeCatalyst 项目链接到项目 GitLab

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。

3. 执行以下操作之一，以查看您空间已安装的扩展列表：

   a. 选择设置，然后选择已安装的扩展。

   b. 在顶部菜单中，选择目录图标

   。

4. 为以下扩展程序之一选择 “配置”：GitHub 存储库、Bitbucket 存储GitLab 库、存储库或 Jira Software。

5. 根据您选择配置的第三方扩展执行下列操作之一：

   - GitHub 存储库：链接存储 GitHub 库。

     1. 在 “链接 GitHub 存储库” 选项卡中，选择 “链接 GitHub 存储库”。
     2. 从GitHub 账户下拉列表中，选择包含您要关联的存储库的 GitHub 账户。
     3. 从GitHub 存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

        > **Tip**
        > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

     4. （可选）如果您在 GitHub 存储库列表中看不到存储库，则可能未在的 Amazon CodeCatalyst 应用程序中将其配置为可以访问存储库 GitHub。您可以配置可在关联账户 CodeCatalyst 中使用哪些 GitHub 存储库。

        a. 导航到您的GitHub帐户，选择 “设置”，然后选择 “应用程序”。

        b. 在 “已安装的 GitHub 应用程序” 选项卡中，选择 Amazon CodeCatalyst 应用程序的 “配置”。

        c. 执行以下任一操作来配置要链接的 GitHub 存储库的访问权限 CodeCatalyst：

           - 要提供对所有当前和未来存储库的访问权限，请选择所有存储库。
           - 要提供对特定存储库的访问权限，请选择 “仅选择存储库”，选择 “选择存储库” 下拉列表，然后选择要允许链接的存储库 CodeCatalyst。

     5. 从CodeCatalyst 项目下拉菜单中，选择要将 GitHub 存储库链接到的 CodeCatalyst 项目。
     6. 选择链接。

如果您不想再在中使用 GitHub 存储库 CodeCatalyst，可以取消它与 CodeCatalyst 项目的链接。解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- Bitbucket 存储库：链接 Bitbucket 存储库。

  1. 在已链接的 Bitbucket 存储库选项卡中，选择链接 Bitbucket 存储库。
  2. 从 Bitbucket 工作区下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作区。
  3. 从 Bitbucket 存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

     > **Tip**
     >
     > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

  4. 从CodeCatalyst 项目下拉菜单中，选择要将 Bitbucket 存储库关联到的 CodeCatalyst 项目。
  5. 选择链接。

  如果您不想再在中使用 Bitbucket 存储库 CodeCatalyst，可以取消该存储库与项目的链接。CodeCatalyst 解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- GitLab 存储库：链接 GitLab 项目存储库。

  1. 在 “链接的 GitLab 项目存储库” 选项卡中，选择 “链接 GitLab 项目存储库”。
  2. 从GitLab 用户下拉列表中，选择包含要链接的项目存储库的 GitLab 用户。
  3. 从GitLab 项目存储库下拉列表中，选择要链接到 CodeCatalyst 项目的存储库。

     > **Tip**
     >
     > 如果存储库的名称已灰显，则无法链接该存储库，因为它已经链接到空间中的另一个项目。

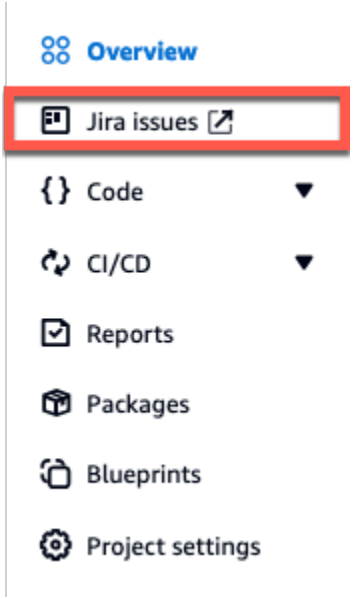
4. 从CodeCatalyst 项目下拉菜单中，选择要将 CodeCatalyst 项目存储库链接到的 GitLab 项目。
5. 选择链接。

如果您不想再在中使用 GitLab 项目存储库 CodeCatalyst，则可以取消其与 CodeCatalyst 项目的链接。取消链接项目存储库后，该项目存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该项目存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

- Jira Software：链接 Jira 项目。

1. 在已链接的 Jira 项目选项卡中，选择链接 Jira 项目。
2. 从 Jira 站点下拉菜单中，选择包含要链接的项目的 Jira 站点。
3. 从 Jira 项目下拉菜单中，选择要链接到 CodeCatalyst 项目的项目。
4. 从CodeCatalyst 项目下拉菜单中，选择要链接到 Jira 项目的项目。 CodeCatalyst
5. 选择链接。

将 Jira 项目链接到 CodeCatalyst 项目后，将完全禁用对 CodeCatalyst 议题的访问权限，CodeCatalyst 导航窗格中的议题将被链接到 Jira 项目的 Jira 议题项所取代。



如果您不想再在中使用 Jira 项目 CodeCatalyst，可以取消其与项目的关联。 CodeCatalyst当 Jira 项目取消关联后，Jira 事务将不可用于该 CodeCatalyst项目，而 CodeCatalyst 问题将再次

成为议题提供者。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

从项目的源 GitHub 存储库页面将存储库、Bitbucket 存储库或 GitLab CodeCatalyst 项目存储库链接到项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择添加存储库，然后选择链接存储库。
5. 从存储库提供商下拉菜单中，选择以下第三方存储库提供商之一：GitHub、Bitbucket、GitLab。
6. 根据您已选择链接的第三方存储库提供程序，执行下列操作之一：

   - GitHub 存储库：链接存储 GitHub 库。

     1. 从GitHub 账户下拉菜单中，选择包含您要关联的存储库的 GitHub 账户。
     2. 从GitHub 存储库下拉菜单中，选择要链接 CodeCatalyst 项目的 GitHub 存储库。

        > **Tip**
        > 如果存储库的名称显示为灰色，则无法链接该存储库，因为它已经链接到 Amazon CodeCatalyst 中的另一个项目。

     3. （可选）如果您在 GitHub 存储库列表中看不到存储库，则可能未在的 Amazon CodeCatalyst 应用程序中将其配置为可以访问存储库 GitHub。您可以配置可在关联账户 CodeCatalyst 中使用哪些 GitHub 存储库。

        a. 导航到您的GitHub帐户，选择 “设置”，然后选择 “应用程序”。

        b. 在 “已安装的 GitHub 应用程序” 选项卡中，选择 Amazon CodeCatalyst 应用程序的 “配置”。

        c. 执行以下任一操作来配置要链接的 GitHub 存储库的访问权限 CodeCatalyst：

           - 要提供对所有当前和未来存储库的访问权限，请选择所有存储库。
           - 要提供对特定存储库的访问权限，请选择 “仅选择存储库”，选择 “选择存储库” 下拉列表，然后选择要允许链接的存储库 CodeCatalyst。

   - Bitbucket 存储库：链接 Bitbucket 存储库。

1. 从 Bitbucket 工作区下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作区。
2. 从 Bitbucket 存储库下拉菜单中，选择要关联项目的 Bitbucket 存储库。 CodeCatalyst

> **Tip**
>
> 如果存储库的名称显示为灰色，则无法链接该存储库，因为它已经链接到 Amazon CodeCatalyst 中的另一个项目。

- GitLab 存储库：链接 GitLab 项目存储库。

1. 从GitLab 用户下拉菜单中，选择包含要链接的项目存储库的 GitLab 用户。
2. 从GitLab 项目存储库下拉菜单中，选择要链接 CodeCatalyst 项目的项目存储库。 GitLab

> **Tip**
>
> 如果项目存储库的名称显示为灰色，则无法链接该项目存储库，因为它已经链接到 Amazon CodeCatalyst 中的另一个项目。

7. 选择链接。

如果您不想再在中使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库 CodeCatalyst，则可以取消其与项目的链接。 CodeCatalyst 解除存储库的链接后，该存储库中的事件将无法启动工作流程运行，并且您将无法在 CodeCatalyst 开发环境中使用该存储库。有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

将您的 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库链接到您的 CodeCatalyst 项目后，您可以在 CodeCatalyst 工作流程和开发环境中使用它。您还可以将已链接存储库与 Amazon Q 开发者版、蓝图等结合使用。有关更多信息，请参阅在第三方存储库事件发生后自动启动工作流运行和创建开发环境。

将 Jira 项目与项目关联并关联拉取请求后，来自的更新将反映在您 CodeCatalyst 的 Jira CodeCatalyst 项目中。有关将拉取请求链接到 Jira 事务的更多信息，请参阅将 Jira 事务与拉 CodeCatalyst 取请求相关联。有关在 Jira 中查看 CodeCatalyst 事件的更多信息，请参阅在 Jira 事务中查看 CodeCatalyst 事件。

在 CodeCatalyst 项目创建过程中将第三方存储库链接到

在创建新项目时，您可以将存储 GitHub 库、Bitbucket 存储库或 GitLab 项目存储库链接到新 CodeCatalyst 项目。 CodeCatalyst 有关更多信息，请参阅 使用链接的第三方存储库创建项目。

取消关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst

如果您不想再使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，也不想在中管理 Jira 项目 CodeCatalyst，则可以取消该存储库或项目与项目的链接。 CodeCatalyst

取消关联 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库不会删除存储库或对其进行任何更改。这不会删除存储在该已链接的存储库中的任何工作流配置文件。但是，一旦您取消了 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库的链接，该存储库中的事件将不再启动工作流程运行，并且您也无法在开发环境中使用该存储库。您可以从 GitHub 存储库、Bitbucket 存储库或 GitLab 存储库扩展的详细信息页面，或者从 CodeCatalyst 项目本身的 “代码” 中的 “源GitHub 存储库” 视图中取消项目GitLab 仓库、Bitbuck et 存储库或项目存储库的链接。

取消链接 Jira 项目不会删除该项目，包括计划项目或开发信息，也不会对其进行任何更改。但是，解除与 Jira 项目的关联后，该项目的 Jira 事务将无法再链接到该 CodeCatalyst 项目， CodeCatalyst 问题将再次成为议题提供者。

> **Important**
>
> 要取消存储 GitHub 库、Bitbucket 存储库或 Gitlab 项目存储库与 CodeCatalyst 项目的关联，您必须是 Space 管理员或项目管理员。

取消项目中 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库或 Jira 项目与扩展详细信息 CodeCatalyst 页面的关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下操作之一，以查看您空间已安装的扩展列表：

    a. 选择设置，然后选择已安装的扩展。

b. 在顶部菜单中，选择目录图标

。

4. 为要配置的以下扩展程序之一选择 “配置”：GitHub 存储库、Bitbucket 存储库、存储库或 Jira Software。GitLab
5. 根据您选择配置的第三方扩展执行下列操作之一：

   - GitHub 存储库：取消存储 GitHub 库的链接。

     在 “GitHub 存储库” 选项卡中，选择要取消链接的 GitHub 存储库，然后选择 “取消链接 GitHub 存储库”。

   - Bitbucket 存储库：取消链接 Bitbucket 存储库。

     在 Bitbucket 存储库选项卡中，选择要取消链接的 Bitbucket 存储库，然后选择取消链接 Bitbucket 存储库。

   - GitLab 存储库：取消 GitLab 项目存储库的链接。

     在 “GitLab 项目存储库” 选项卡中，选择要取消链接的 GitLab 项目存储库，然后选择 “取消链接 GitLab 项目存储库”。

   - Jira Software：取消链接 Jira 项目。

     在 Jira 项目选项卡中，选择要取消链接的 Jira 项目，然后选择取消链接 Jira 项目。

6. 在取消链接对话框中，查看取消链接存储库所带来的影响。
7. 在文本输入字段中输入 unlink，然后选择取消链接。

从源存储库页面取消项目中存储 GitHub 库、Bitbucket 存储库或 GitLab CodeCatalyst 项目存储库的链接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在导航窗格中，选择代码，然后选择源存储库。
4. 选择要取消链接的存储库的单选按钮，然后选择取消链接存储库。
5. 查看对话框中的信息。按照说明进行操作，然后选择取消链接以取消存储库的链接。

在中查看第三方存储库并搜索 Jira 事务 CodeCatalyst

关联 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库后，您可以在中查看它们 CodeCatalyst 以确认和配置资源。您也可以在中搜索关联的 Jira 事务。 CodeCatalyst

主题

在中查看第三方存储库 CodeCatalyst

您可以在 GitLab 项目的源存储库列表中或从存储 GitHub 库、Bitbucket 存储库或存储库扩展详细信息页面查看链接GitHub 存储库、Bitbucket 存储库或项目GitLab 存储库。从存储库列表中选择它们并不能在中打开它们 CodeCatalyst。相反，它们在第三方存储库提供程序中打开，您可以在其中查看和处理已链接的存储库中的代码。

要在中查看链接的 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在导航窗格中，选择代码，然后选择源存储库。

从扩展详情页面查看链接的存储 GitHub 库、Bitbucket 存储库或 GitLab 项目存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间，然后选择 “已安装的扩展” 选项卡。
3. 根据要查看的第三方存储库，执行下列操作之一：

    - 在GitHub 存储库中，选择 “配置”，然后选择 “链接 GitHub 存储库”，查看与 CodeCatalyst 空间中 CodeCatalyst 项目关联的所有 GitHub 存储库。
    - 在 Bitbucket 存储库中，选择 “配置”，然后选择 “关联的 Bitbucket 存储库”，查看与您空间中 CodeCatalyst 项目关联的所有 Bitbucket 存储库。 CodeCatalyst
    - 在GitLab 存储库中，选择 “配置”，然后选择 “链接的 GitLab 项目存储库”，以查看与 CodeCatalyst 空间中 CodeCatalyst 项目关联的所有 GitLab 项目存储库。

列表中显示了链接到您的 GitLab 项目的存储 GitHub 库、Bitbucket 存储库或 CodeCatalyst 项目存储库。选择 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库以查看和编辑第三方存储库提供商中的文件。

> **Note**
>
> 如果工作流程在源操作中使用 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库，则您在可视化编辑器或 YAML 编辑器中对工作流程 YAML 所做的更改 CodeCatalyst 将自动提交并推送到第三方存储库。

在中搜索 Jira 问题 CodeCatalyst

关联 Jira 项目后，您可以使用 CodeCatalyst 全局搜索栏在链接的 Jira 项目中搜索问题。您还可以在中搜索 Jira 事务， CodeCatalyst 同时链接到拉取请求中的事务。有关将 Jira 事务与拉 CodeCatalyst 取请求关联的更多信息，请参阅将 Jira 事务与拉 CodeCatalyst 取请求相关联。

在已链接的 Jira 项目中搜索 Jira 事务

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在全局搜索栏中，在已链接的 Jira 项目中搜索要链接到拉取请求的事务或 Jira 事务。

在第三方存储库事件发生后自动启动工作流运行

您可以使用链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库作为工作流程的来源，其中对链接存储库、Bitbucket GitHub 存储库或 GitLab 项目存储库中指定分支的更改会自动启动工作流程运行。

工作流是一个自动化过程，它描述了如何在持续集成和持续交付（CI/CD）系统中构建、测试和部署代码。工作流定义了在工作流运行期间要执行的一系列步骤，也称为操作。工作流还定义了促使工作流启动的事件或触发器。要设置工作流程，您可以使用 CodeCatalyst 控制台的可视化或 YAML 编辑器创建工作流程定义文件。

> **Tip**
>
> 要快速了解如何在项目中使用工作流，请使用蓝图创建项目。每个蓝图都部署了一个可以正常运行的工作流，您可以对工作流进行查看、运行和试验。

将工作流程配置为使用链接 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库时，工作流程配置文件存储在该存储 GitHub 库、Bitbucket 存储库或 GitLab 项目存储库中。工作流配置是一个 YAML 文件，用于定义工作流名称、触发器、资源、构件和操作。有关工作流配置文件的更多信息，请参阅工作流 YAML 定义。

工作流程配置文件必须位于 GitHub 存储库、Bitbucket 存储库或 GitLab 项目存储库的`./codecatalyst/workflows/`目录中。

您可以使用工作流编辑器来创建和配置工作流。有关更多信息，请参阅入门工作流和将源存储库连接到工作流。

添加触发器以启动工作流运行

您可以将 CodeCatalyst 工作流程配置为在将代码推送到您 GitHub 或 Bitbucket 存储库的指定分支时自动开始运行。要自动启动工作流运行，请在工作流配置文件的 `Triggers` 部分中添加触发器。

示例：一个简单的代码推送触发器

以下示例显示了一个触发器，在代码被推送到源存储库中的任何分支时，该触发器将启动工作流运行。

```
Triggers:
  - Type: PUSH
```

示例：一个简单的拉取请求触发器

以下示例显示了一个触发器，在针对源存储库中的任何分支创建拉取请求时，该触发器将启动工作流运行。

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
```

有关更多信息，请参阅 使用触发器自动启动工作流运行。

限制第三方存储库提供程序的 IP 访问权限

您可以通过设置规则或配置，根据 IP 地址限制对存储 GitHub 库、Bitbucket 存储库或 GitLab 项目存储库的访问权限。您可以通过第三方提供程序的设置或访问控制功能来执行此操作。

根据您使用的第三方存储库提供程序，请参阅下列内容之一：

- Amazon CodeCatalyst GitHub 存储库扩展与 GitHub 企业云 IP 访问限制兼容。在将 E GitHub nterprise Cloud 组织配置为限制对特定 IP 地址的访问时，您还可以允许 GitHub 应用程序配置允许列表，允许自动 CodeCatalyst 注册其 IP 地址 GitHub。或者，您可以手动添加 CodeCatalyst IP 地址。
- 亚马逊 CodeCatalyst Bitbucket 存储库扩展与 Bitbucket 云高级版访问限制兼容。在配置 Bitbucket Cloud Premium 工作区以限制对特定 IP 地址的访问时，您还可以 add IP addresses or network blocks for a set of IP addresses to an allowlist。
- Amazon CodeCatalyst GitLab 存储库扩展与 GitLab IP 地址限制兼容。在配置 GitLab 高级或旗舰版组以限制对特定 IP 地址的访问时，您还可以将一组 IP 地址的 IP 地址或网络块添加到许可名单中。

如果 CodeCatalyst IP 地址不在第三方存储库的许可名单中，Amazon CodeCatalyst 应用程序将无法访问您的第三方存储库。有关更多信息，请参阅 第三方存储库扩展使用的 IP 地址。

第三方存储库扩展使用的 IP 地址

第三方扩展使用以下 IP 地址访问您的第三方资源：

- GitHub 存储库：

```
us-west-2
  52.32.242.246
  54.148.176.49
  35.164.118.94
eu-west-1
  34.241.64.10
  34.246.255.80
  3.248.38.7
```

- 比特存储库和存储GitLab 库：

```
us-west-2
  35.160.210.199
  54.71.206.108
  54.71.36.205
eu-west-1
  34.242.64.82
  52.18.37.201
  54.77.75.62
```

在工作流失败时阻止第三方合并

将 GitHub 或 Bitbucket 存储库关联到后 CodeCatalyst，您可以为拉取请求添加 CodeCatalyst 工作流程。同样，在将 GitLab 项目存储库链接到之后， CodeCatalyst 您可以为合并请求添加 CodeCatalyst 工作流程。在一次特定的提交中可以运行一个或多个工作流程，并且中每个工作流程的运行状态 CodeCatalyst 也反映在 GitHub、Bitbucket 或 GitLab中的提交状态中。推送新提交时，新的工作流程运行状态将反映在 GitHub、Bitbucket 或 GitLab 该新提交中。如果您再次为提交运行工作流，则新的工作流运行状态将覆盖该提交和工作流的上一个状态。

您可以在 GitHub 或 Bitbucket 中设置分支保护规则以阻止拉取请求合并，或者在 GitLab 最新提交的工作流程运行状态为失败时阻止合并请求。使用分支保护规则，最新提交的状态会影响在、Bitbucket 或 GitLab中 GitHub合并拉取请求的能力。要了解有关工作流的更多信息，请参阅运行工作流和使用触发器自动启动工作流运行。

根据您使用的第三方存储库提供商，请参阅以下内容：

- GitHub 存储库： GitHub的文档关于状态检查和关于受保护的分支。
- Bitbucket 存储库：Bitbucket 的 Using branch permissions 和 Take control with branch permissions in Bitbucket Cloud 文档。
- GitLab 存储库： GitLab自动合并和受保护分支的文档。

将 Jira 事务与拉 CodeCatalyst 取请求相关联

您可以将 CodeCatalyst 源存储库中创建的拉取请求链接到 Jira 议题。链接 Jira 事务后，该事务将显示为拉取请求的属性。因此，拉取请求事件、工作流事件和部署事件将发送到 Jira 并添加到 Jira 事务中。拉取请求可以链接到一个或多个 Jira 事务。您只能链接 CodeCatalyst 源存储库中的拉取请求，不能链接第三方存储库中的拉取请求 GitHub。在将 Jira 事务与拉取请求关联之前，必须先将您的 Jira 项

目链接到该 CodeCatalyst 项目。有关将 Jira 项目链接到项目的更多信息，请参阅在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。 CodeCatalyst

> **Note**
>
> 如果 CodeCatalyst 项目中没有包含两个分支的源存储库，则无法创建拉取请求。有关拉取请求的更多信息，请参阅中的处理拉取请求 CodeCatalyst。

将 Jira 事务与拉 CodeCatalyst 取请求相关联

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在导航窗格中，选择代码，然后选择拉取请求。
4. 选择创建拉取请求以输入拉取请求详细信息。
5. 从源存储库下拉菜单中，选择要在其中链接拉取请求的源存储库。
6. 从源分支下拉菜单中，选择包含要审查的更改的分支。
7. 从目标分支下拉菜单中，选择要在其中合并已审查的更改的分支。
8. 在拉取请求标题文本输入字段中，输入拉取请求的标题。
9. 为 Jira 事务 - 可选字段选择链接事务，选择下拉菜单，然后从链接的 Jira 项目中搜索要添加的 Jira 事务。
10. 选择要添加到拉取请求的 Jira 事务。
11. 选择创建以创建拉取请求。

将 Jira 事务与 CodeCatalyst 拉取请求关联后，即可获得拉取请求的摘要。摘要包括工作流运行、链接的事务、所需的审阅者、可选的审阅者和作者。

> **Note**
>
> 与 Jira 事务相关的@@ 受让人和创建者信息在中不可用。 CodeCatalyst

关联拉取请求后，同步的 CodeCatalyst 项目和 Jira 项目允许来自 CodeCatalyst 的更新反映在你的 Jira 项目中。在 Jira 中查看已链接的拉取请求以及任何与该拉取请求相关的工作流事件的状态时，该状态将显示在 Jira 事务中。有关在 Jira 中查看 CodeCatalyst 事件的更多信息，请参阅在 Jira 事务中查看 CodeCatalyst 事件。

在 Jira 事务中查看 CodeCatalyst 事件

如果您的 CodeCatalyst 项目和 Jira 项目已关联，则拉取请求的摘要状态和关联 CodeCatalyst 的工作流事件的状态将反映在您的 Jira 事务中。例如，如果您关闭或合并拉取请求 CodeCatalyst，则状态更新将反映在 Jira 问题中。 CodeCatalyst 与 CodeCatalyst 拉取请求相关的工作流程 CI/CD 事件是同步的，因此成功运行的工作流程也会发送到 Jira 事务。

查看 Jira 事务中的 CodeCatalyst 事件

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 项目。
3. 在 CodeCatalyst 导航窗格中，选择 “代码”，选择 “拉取请求”，然后选择要在 Jira 项目中查看的 Jira 事务的拉取请求。
4. 在其它信息窗格中，选择要在 Jira 项目中查看的 Jira 事务。
5. 从 Jira 项目的详细信息窗格中，选择为开发列出的拉取请求以查看拉取请求的详细信息。
6. （可选）要查看最新构建，请选择构建选项卡。
7. （可选）要查看开发状态，请选择部署选项卡。

在 CodeCatalyst 中搜索代码、事务、项目和用户

在 CodeCatalyst 中使用搜索栏或专用搜索结果窗口来搜索代码、事务、项目和用户。

您可以通过在搜索栏中输入名称、描述和状态等查询来在空间和项目中查找资源。您也可以使用搜索查询语言来细化搜索查询。

主题

搜索

1. 在顶部导航栏中的搜索栏中，输入搜索查询。
2. （可选）使用 CodeCatalyst 的搜索查询语言来细化您的搜索查询。有关更多信息，请参阅细化搜索查询。
3. 请执行以下操作之一：
   - 要在您当前所在的项目中搜索资源，请选择此项目。
   - 要在您当前所在的空间中的所有项目中搜索资源，请选择此空间。
4. 通过执行下列操作之一，在专用的搜索结果窗口中查看搜索结果：
   - 在快速搜索结果窗口的底部，选择在 project-name | space-name 中查看所有结果以查看所有搜索结果。
   - 按 Enter 以查看所有搜索结果。

> **Tip**
>
> 在拉取请求评论或描述中或在事务评论或描述中提及其他项目用户，方法是使用 @ 符号后跟这些用户的显示名称或用户名。您还可以链接到事务或代码文件等资源，方法是使用 @ 符号后跟事务或代码文件的名称。

细化搜索查询

如果通过搜索找不到要查找的内容，则可以使用 CodeCatalyst 的专用查询语言来细化搜索。单个字段没有字符限制，但整个查询的长度不得超过 1024 个字符。

主题

按类型细化

要将搜索范围细化到特定类型的信息，请在搜索中包含 *type:result-type*，其中 *result-type* 为 code、issue、project 或 user。

示例：

- type:code AND java – 在包含“java”的与代码相关的字段中显示代码结果。

  有关更多信息，请参阅 代码字段。
- type:issue AND Bug – 在包含“Bug”的与事务相关的字段中显示事务结果。

  有关更多信息，请参阅 事务字段。
- type:user AND MaryMajor – 在包含“MaryMajor”的与用户相关的字段中显示用户结果。

  有关更多信息，请参阅 用户字段。
- type:project AND Datafeeder – 显示包含“Datafeeder”的项目结果。

  有关更多信息，请参阅 项目字段。

按字段细化

要将搜索范围细化到特定字段，请在搜索中包含 *field-name:query*，其中 *field-name* 为 title、username、project、description 等，*query* 为要搜索的文本。有关字段的列表，请参阅可搜索字段参考。您可以使用圆括号搜索多个查询。

示例：

- `title:bug` – 显示标题包含“bug”的结果。
- `username:John` – 显示用户名包含“John”的结果。
- `project:DataFeeder` – 显示项目包含“DataFeeder”的结果。查询不区分大小写。
- `description:overview` – 显示描述包含“overview”的结果。

使用布尔运算符进行细化

要指定针对搜索短语的限制，可以使用布尔运算符 AND、OR 和 NOT。如果您列出了多个短语，默认情况下，CodeCatalyst 会使用 OR 联接它们。您可以使用圆括号对搜索短语进行分组。

- `exception AND type:code` – 仅显示“exception”的代码结果。
- `path:README.md AND repo:ServerlessAPI` – 显示带“README.md”的路径的结果，其中存储库名为“ServerlessAPI”。
- `buildspec.yml AND (repo:ServerlessAPI OR ServerlessWebApp)` – 显示“buildspec.yml”的结果，其中存储库为“ServerlessAPI”或“ServerlessWebApp”。
- `path:java NOT (path:py OR path:ts)` – 显示路径包含“java”但不包含“py”或“ts”的结果。

按项目细化

要将搜索范围细化到特定项目，请在搜索中包含 *`project:name AND query`*，其中 *`name`* 为要在其中搜索的项目，*`query`* 为要搜索的内容。

- `project:name AND query` – 显示路径包含查询和项目名称的结果。

使用搜索时的注意事项

延迟的内容更新 – 内容更新（例如，名称更改或事务重新分配）可能需要几分钟才能反映在搜索结果中。大型更新（例如代码库迁移）可能需要更长的时间才能显示在搜索结果中。

对特殊字符进行转义 – 在搜索查询中，需要特别注意以下特殊字符：`+ - & & || ! ( ) { } [ ] ^ " ~ * ? : \`。特殊字符不会影响查询，您必须将其移除或对其进行转义。要对字符进行转义，请在字符前面添加反斜杠（`\`）。例如，搜索查询 [Feature] 应为 Feature 或 \[Feature\]。

缩小搜索范围 – 搜索不区分大小写。采用全小写字母形式进行搜索可防止查询在大小写变化时拆分单词。例如，要查询 `MyService` 和仅查询 `MyService`，请考虑查询 `myservice` 以避免出现仅包含 `my` 或 `service` 的结果。

默认情况下，搜索使用连词 OR 来联接单词和单词的某个部分。例如，`new function` 可以返回同时包含 `new` 和 `function` 的结果，也可以返回仅包含 `new` 或 `function` 的结果。要避免出现后一种情况，可使用 `AND` 组合多个单词。例如，您可以搜索 `new AND function`。

默认分支 – 搜索将仅返回源存储库默认分支上的最新提交中的代码结果。要在其他分支或提交上查找代码，可以考虑本地克隆存储库、在开发环境中打开分支或在 CodeCatalyst UI 中查看分支和详细信息。如果更改默认分支，则会更新可通过搜索发现的文件。有关更多信息，请参阅 管理存储库的默认分支。

> **Important**
>
> CodeCatalyst 不支持检测已链接存储库的默认分支中的更改。要更改已链接存储库的默认分支，您必须先取消该存储库与 CodeCatalyst 的链接，更改默认分支，然后重新链接该存储库。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。
> 作为最佳实践，在链接存储库之前，请始终确保您拥有最新版本的扩展。

可搜索字段参考

在您输入搜索查询时，CodeCatalyst 会搜索以下字段。别名是另一个可用于在高级查询语言中引用字段的名称。

代码字段

字段	别名	描述
branchName	branch	代码文件所在分支的名称。
code	不适用	以代码片段的形式表示的有关代码内容的信息，指示符合搜索条件的源代码部分。
commitId	不适用	上次在其中更新返回的代码文件的提交的提交 ID。可能是或可能不是 `branchName` 中指定的分支名称最新块的提交 ID。

字段	别名	描述
commitMessage	不适用	上次在其中更新代码文件的提交的提交消息。可能是或可能不是 branchName 中指定的分支名称最新块的提交消息。如果未提供提交消息，则此值将为空字符串。
filePath	path	此代码文件的文件路径。
lastUpdatedBy	不适用	上次更新代码文件的 CodeCatalyst 用户。如果用户名不可用，则此值将是 Git 配置文件中配置的用户的电子邮件地址。
lastUpdatedById	不适用	系统为上次更新代码文件的用户生成的唯一 ID。如果用户 ID 不可用，则此值可能是用户的电子邮件地址。
lastUpdatedTime	不适用	上次使用包含代码文件的提交更新搜索数据的时间（采用世界协调时间（UTC）时间戳）。
projectId	不适用	系统生成的项目的唯一 ID。
projectName	projectNames、project	包含已在其中提交代码文件的源存储库的项目的显示名称。
repositoryId	repoId	系统生成的源存储库的唯一 ID。
repositoryName	repository、repo	已在其中提交代码文件的源存储库的显示名称。

事务字段

字段	别名	描述
assigneeIds	assigneeId	系统为分配给事务的用户生成的唯一 ID。
assignees	assignee	分配给事务的用户的用户名。
createdBy	不适用	创建了事务的用户的显示名称。
createdById	不适用	系统为创建了事务的用户生成的唯一 ID。
createdTime	不适用	事务的创建时间（采用世界协调时间（UTC）时间戳）。
description	不适用	事务的描述。
isArchived	archived	布尔值，指示是否创建处于已存档状态的事务。
isBlocked	blocked	布尔值，指示是否已将事务标记为已阻止。
labelIds	labelId	系统为事务标签生成的唯一 ID。
lastUpdatedBy	不适用	上次更新了事务的用户的显示名称。
lastUpdatedById	不适用	系统为上次更新了事务的用户生成的唯一 ID。
lastUpdatedTime	不适用	事务的上次更新时间（采用世界协调时间（UTC）时间戳）。

字段	别名	描述
priority	不适用	事务的优先级（如果已分配）。
projectId	不适用	系统生成的项目的唯一 ID。
projectName	projectNames、project	可在其中找到此事务的项目。
shortId	不适用	事务的缩短的自动递增标识符。
状态	不适用	事务的状态，指示事务是在待办事项中还是在面板上的列中。
statusId	不适用	状态的系统标识符。
title	不适用	事务的标题。

项目字段

字段	别名	描述
description	不适用	项目的描述。
lastUpdatedTime	不适用	项目元数据的上次更新时间（采用世界协调时间（UTC）时间戳）。
projectName	project	空间中的项目的名称。
projectPath	不适用	项目的 URL 可路由名称（在项目创建过程中定义）。用于需要项目名称的 URL 中。

用户字段

字段	别名	描述
displayName	不适用	在 CodeCatalyst 中对用户使用的名称。显示名称不是唯一的。
电子邮件	不适用	用户的电子邮件地址。
lastUpdatedTime	不适用	用户元数据的上次更新时间（采用世界协调时间（UTC）时间戳）。
userName	username	用户在注册 CodeCatalyst 时选择的用户名。与显示名称不同，用户名无法更改。

排查 Amazon CodeCatalyst 的问题

以下信息有助于您排查 CodeCatalyst 中的常见问题。您还可以使用 Amazon CodeCatalyst 运行状况报告来确定是否存在可能影响体验的服务问题。

主题

排查一般访问问题

我忘记密码了

问题：我忘记了用于 AWS 构建者 ID 和 Amazon CodeCatalyst 的密码。

可能的修复方法：解决此问题的最简单方法是重置密码。

1. 打开 Amazon CodeCatalyst 并输入您的电子邮件地址。然后，选择继续。

2. 选择忘记密码？。
3. 我们将向您发送一封电子邮件，其中包含更改密码的链接。如果您在收件箱中未看到这封电子邮件，请检查垃圾邮件文件夹。

Amazon CodeCatalyst 的部分或全部内容不可用

问题：我导航到 CodeCatalyst 控制台，或点击链接进入控制台，但却看到一个错误。

可能的修复方法：出现这种问题的最常见原因是，您点击了一个链接，进入了一个您没有被邀请加入的项目或空间，或者是服务的总体可用性出现了问题。查看运行状况报告，了解服务是否存在任何已知问题。如果不存在问题，请联系邀请您加入项目或空间的人员，并要求再次发出邀请。如果您尚未被邀请加入任何项目或空间，则可以注册并创建自己的空间和项目。

我无法在 CodeCatalyst 中创建项目

问题：我想创建一个项目，但创建项目按钮显示为不可用，或者我收到了错误消息。

可能的修复方法：出现此问题的最常见原因是，您使用的是不具有空间管理员角色的 AWS 构建者 ID 登录控制台。您必须拥有此角色才能在空间中创建项目。

如果您确实拥有此角色，但该按钮显示不可用，则可能是服务出现了暂时性问题。刷新您的浏览器，然后重试。

排查支持问题

我在访问 支持 for Amazon CodeCatalyst 时出现错误

问题：当我选择 支持 for Amazon CodeCatalyst 选项时，我收到了以下错误消息：

```
Unable to assume role

To access support cases, you must add the role
AWSRoleForCodeCatalystSupport to the AWS ## that is the billing account for
the space.
```

可能的修复方法：将所需角色添加到作为空间计费账户的 AWS 账户。指定为空间计费账户的账户使用 `AWSRoleForCodeCatalystSupport` 角色和 `AmazonCodeCatalystSupportAccess` 托管式策略。有关更多信息，请参阅 为您的账户和空间创建 AWSRoleForCodeCatalystSupport 角色。

> **Note**
>
> AWS 构建者 ID 只能为其通过身份验证的别名获得支持，并且只能为基于 CodeCatalyst 中权限的资源获得支持。为空间中的所有用户提供账户和计费支持。不过，构建者只能获得他们在 CodeCatalyst 中拥有权限的资源和信息的支持。

我无法为自己的空间创建技术支持案例

问题：我无法为自己的空间创建技术支持案例。

修复方法：需要在空间计费账户中添加 Business Support 或 Enterprise Support 计划，空间中的用户才能创建技术支持案例。请您的空间管理员为您的空间计费账户添加 支持 计划，或访问 https://repost.aws/ 向 AWS 社区提问。

我的支持案例账户不再与我在 CodeCatalyst 中的空间连接

问题：我的支持案例账户不再与我在 CodeCatalyst 中的空间连接。

修复方法：如果具有空间管理员角色的用户切换空间计费账户，则会断开 支持 计划和所有相关案例与空间的连接。在 支持 for Amazon CodeCatalyst 中将不再显示与旧空间计费账户相关的 支持 案例。该计费账户的根用户可以从 AWS 管理控制台查看和解决旧案例，并可以为 支持 设置 IAM 权限，以便其他用户查看和解决旧案例。您将无法继续通过 AWS 管理控制台从旧的空间计费账户获得 CodeCatalyst 的技术支持，但您可以获得其它服务的技术支持，直到您的 支持 计划被取消。

有关更多信息，请参阅《支持 User Guide》中的 Updating, resolving, and reopening your case。

我无法在 支持 for Amazon CodeCatalyst 中为另一项 AWS 服务打开支持案例

问题：我无法在 支持 for CodeCatalyst 中为另一项 AWS 服务打开支持案例。

可能的修复方法：您只能从 支持 for CodeCatalyst 打开 CodeCatalyst 支持案例。如果您需要为从 CodeCatalyst 部署到其它 AWS、Amazon 或其它第三方服务的服务或资源提供支持，您需要通过 AWS 管理控制台或第三方服务支持渠道创建一个案例。有关更多信息，请参阅《支持 User Guide》中的 Creating support cases and case management。

Amazon CodeCatalyst 的部分或全部内容不可用

问题：我导航到 CodeCatalyst 控制台，或点击链接进入控制台，但却看到一个错误。

可能的修复方法：出现这种问题的最常见原因是，您点击了一个链接，进入了一个您没有被邀请加入的项目或空间，或者是服务的总体可用性出现了问题。查看运行状况报告，了解服务是否存在任何已知问题。如果不存在问题，请联系邀请您加入项目或空间的人员，并要求再次发出邀请。如果您尚未被邀请加入任何项目或空间，则可以注册并创建自己的空间和项目。

我无法在 CodeCatalyst 中创建项目

问题：我想创建一个项目，但创建项目按钮显示为不可用，或者我收到了错误消息。

可能的修复方法：出现此问题的最常见原因是，您使用的是不具有空间管理员角色的 AWS 构建者 ID 登录控制台。您必须拥有此角色才能在空间中创建项目。

如果您确实拥有此角色，但该按钮显示不可用，则可能是服务出现了暂时性问题。刷新您的浏览器，然后重试。

由于用户名被截断，我无法以新用户身份访问我的 BID 空间，或无法将其添加为新的 SSO 用户

问题：CodeCatalyst 会在 100 个字符后截断用户名，这可能导致某些用户名看起来相同。作为访问 CodeCatalyst 空间的新用户，我遇到的问题取决于空间的类型，具体如下：

- 我有一个 AWS 构建者 ID，我想用它登录 CodeCatalyst。当我尝试登录该空间时，我收到一条信息，提示我的用户名无效。
- 我是支持身份联合验证的 CodeCatalyst 空间的联合身份管理员。在 IAM Identity Center 中向 SSO 用户和组添加新用户时，我收到一条用户无效的消息。

可能的修复方法：第一个使用给定的截断用户名登录 CodeCatalyst 或作为 SSO 用户添加到空间的用户将获得成功。任何使用 AWS 构建者 ID 注册的用户或在此之后在 IAM Identity Center 中添加的用户都将无法登录，因为名称会显示为重复。根据空间类型，执行以下操作之一：

- 要能够登录 AWS 构建者 ID 空间，请使用不同的用户名注册。
- 要能够在 IAM Identity Center 中添加新用户，请使用不同的用户名添加用户。

> **Note**
>
> 即使用户名看起来被截断，CodeCatalyst 与身份的映射方式也不会受到截断用户名的影响。但是，如果创建的用户名与截断的用户名相同，那么如果（与同一空间或 IAM Identity Center 应

用程序相关联的）另一个用户已经使用该截断的用户名加入 CodeCatalyst，则该用户名将不可用。

将一个 SSO 用户作为新用户添加到我的联合空间后，创建了重复的用户

问题：如果将 CodeCatalyst SSO 用户添加到 CodeCatalyst 空间又将其删除，这些用户可能会遇到用户名尝试重复使用的问题。这可能会导致类似于这样的错误：即使该用户是重新创建的，系统仍不允许重复使用它。

`Unable to assume role`

```
To access support cases, you must add the role
AWSRoleForCodeCatalystSupport to the AWS ## that is the billing account for
the space.
```

可能的修复方法：如果删除现有的 IDC 用户，然后使用相同的用户名重新创建新用户，则新用户无法登录，因为用户名与旧用户冲突。将用户名作为 SSO 用户添加到空间后，该名称将无法再次使用。任何使用 AWS 构建者 ID 注册的用户或在此之后在 IAM Identity Center 中添加的用户都将无法登录，因为名称会显示为重复。

根据空间类型，执行以下操作之一：

- 要能够登录 AWS 构建者 ID 空间，请使用不同的用户名注册。
- 要能够在 IAM Identity Center 中添加新用户，请使用不同的用户名添加用户。

我想在 CodeCatalyst 中提交反馈

问题：我在 CodeCatalyst 中发现了一个错误，我想提交反馈。

可能的修复方法：您可以直接在 CodeCatalyst 中提交反馈。

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 在导航窗格中，选择给出反馈。
3. 从下拉菜单中选择反馈类型，然后输入您的反馈。

排查源存储库的问题

以下信息有助于您排查 CodeCatalyst 中源存储库的常见问题。

主题

我已达到空间的最大存储量并看到警告或错误

问题：我想将代码提交到 CodeCatalyst 中的一个或多个源存储库，但看到了一个错误。控制台中的源存储库页面上显示一条消息，指示我已达到空间的存储限制。

可能的修复方法：根据您在项目或空间中的角色，您可以缩小一个或多个源存储库的大小，删除未使用的源存储库，或者将计费套餐更改为具有更多存储量的计费套餐。

- 要缩小项目中源存储库的大小，可以删除未使用的分支。有关更多信息，请参阅删除分支和贡献者角色。
- 要减少空间的总体存储量，可以删除未使用的源存储库。有关更多信息，请参阅删除源存储库和项目管理员角色。
- 要增加空间的可用存储量，您可以将计费套餐更改为具有更多存储量的计费套餐。有关更多信息，请参阅《Amazon CodeCatalyst Administrator Guide》中的 Changing your CodeCatalyst billing tier。

我在尝试克隆或推送到 Amazon CodeCatalyst 源存储库时收到错误

问题：当我尝试将源存储库克隆到本地计算机或集成式开发环境（IDE）时，出现权限错误。

可能的修复方法：您的 AWS 构建者 ID 可能没有个人访问令牌（PAT），可能未使用 PAT 配置凭证管理系统，或者您的 PAT 可能已过期。尝试以下一种或多种解决方案：

- 创建个人访问令牌（PAT）。有关更多信息，请参阅 使用个人访问令牌向用户授予对存储库的访问权限。
- 确保您已接受加入包含源存储库的项目的邀请，并且您仍然是该项目的成员。如果您不是该项目的活跃成员，则无法克隆源存储库。登录控制台，并尝试导航到您试图克隆源存储库的空间和项目。如果您在空间的项目列表中看不到该项目，则说明您不是该项目的成员，或者您尚未接受加入该项目的邀请。有关更多信息，请参阅 接受邀请并创建 AWS 建筑商 ID。
- 确保克隆命令的格式正确，并包含您的 AWS 构建者 ID。例如：

```
https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

- 使用 AWS CLI 确保您有一个与您的 AWS 构建者 ID 关联的 PAT，并且该 PAT 未过期。如果您没有 PAT 或 PAT 已过期，请创建一个。有关更多信息，请参阅 使用个人访问令牌向用户授予对存储库的访问权限。
- 尝试创建一个开发环境来处理源存储库中的代码，而不是将其克隆到本地存储库或 IDE。有关更多信息，请参阅 创建开发环境。

我在尝试提交或推送到 Amazon CodeCatalyst 源存储库时收到错误

问题：当我尝试推送到源存储库时，出现权限错误。

可能的修复方法：您在项目中可能没有可让您向项目提交和推送代码更改的角色。查看您在试图将更改推送到源存储库的项目中的角色。有关更多信息，请参阅获取成员及其项目角色的列表和使用用户角色授予访问权限。

如果您的角色可让您提交和推送更改，那么您尝试提交更改的分支可能配置有一个分支规则，阻止您将代码更改推送到该分支。尝试创建一个分支，然后将代码推送到该分支。有关更多信息，请参阅 使用分支规则管理分支允许的操作。

我的项目需要一个源存储库

问题：我的项目要么没有源存储库，要么我的项目需要另一个源存储库。

可能的修复方法：有些项目是在没有任何资源的情况下创建的。如果您是该项目的成员，则可以在 CodeCatalyst 中为该项目创建源存储库。如果具有空间管理员角色的人员安装 GitHub 存储库并将其连

接到 GitHub 账户，那么，如果您具有项目管理员角色，则可以链接到可用的 GitHub 存储库以将其添加到您的项目中。有关更多信息，请参阅创建源存储库和链接源存储库。

我的源存储库是全新的，但包含一个提交

问题：我刚刚创建了一个源存储库。该源存储库应该是空的，但其中包含一个提交、一个分支和一个 README.md 文件。

可能的修复方法：这是预期的行为。CodeCatalyst 中的所有源存储库都包含一个初始提交，该提交将默认分支设置为 main 并包含示例代码（如果存储库是使用包含示例代码的蓝图为项目创建的）或存储库 README 文件的模板 Markdown 文件。您可以在控制台和 Git 客户端中创建其它分支。您可以在控制台中创建和编辑文件，也可以在开发环境和 Git 客户端中删除文件。

我想要一个不同的分支作为默认分支

问题：我的源存储库有一个名为 main 的默认分支，但我想要一个不同的分支作为默认分支。

可能的修复方法：您无法在 CodeCatalyst 中更改或删除源存储库中的默认分支。您可以创建其它分支，并在工作流的源操作中使用这些分支。您也可以选择链接 GitHub 存储库并将其用作项目的存储库。

我收到关于拉取请求中活动的电子邮件

问题：我没有注册或配置关于拉请求活动的电子邮件通知，但我还是收到了相关通知。

可能的修复方法：自动发送关于拉取请求活动的电子邮件通知。有关更多信息，请参阅 在 Amazon 中使用拉取请求查看代码 CodeCatalyst。

我忘记了我的个人访问令牌（PAT）

问题：我一直在使用 PAT 来克隆、推送和拉取源存储库的代码，但我丢失了令牌的值，而且我无法在 CodeCatalyst 控制台中找到该值。

可能的修复方法：解决此问题的最快方法是创建另一个 PAT，并将您的凭证管理器或 IDE 配置为使用这个新的 PAT。我们仅在您创建 PAT 时显示其值。如果您丢失该值，则无法检索它。有关更多信息，请参阅 使用个人访问令牌向用户授予对存储库的访问权限。

拉取请求不显示我期望的更改

问题：我创建了一个拉取请求，但我看不到源分支和目标分支之间预期的变化。

可能的修复方法：这可能是由许多问题引起的。尝试以下一种或多种解决方案：

- 您可能正在查看旧修订版之间的更改，也可能没有查看最新更改。刷新浏览器，确保选择了要查看的修订版之间的比较。
- 并非拉取请求中的所有更改都可以在控制台中显示。例如，您无法在控制台中查看 Git 子模块，因此无法在拉取请求中查看子模块中的差异。有些差异可能太大而无法显示。有关更多信息，请参阅中的源存储库配额 CodeCatalyst和查看文件。
- 拉取请求会显示合并基准与您选择的任何修订版之间的差异。在创建拉取请求时，为您显示的差异是源分支最新块与目标分支最新块之间的差异。创建拉取请求后，显示的差异是修订版与其合并基准之间的差异。合并基准是在创建修订版时作为目标分支最新块的提交。合并基准可在不同修订版之间发生变化。有关 Git 中差异和合并基准的更多信息，请参阅 Git 文档中的 git-merge-base。

拉取请求的状态显示为“不可合并”

问题：我想合并拉取请求，但其状态显示为不可合并。

可能的修复方法：这可能是由一个或多个问题引起的：

- 拉取请求的所有必需的审阅者都必须批准拉取请求，然后才能合并请求。查看必需的审阅者列表中是否有在姓名旁边显示时钟图标的任何审阅者。时钟图标表示该审阅者尚未批准拉取请求。

  > **Note**
  >
  > 如果在批准拉取请求之前从您的项目中移除了必需的审阅者，则无法合并拉取请求。关闭该拉取请求并创建新的拉取请求。

- 源分支和目标分支之间可能存在合并冲突。CodeCatalyst 不支持所有可能的 Git 合并策略和选项。您可以在开发环境中评估分支是否存在合并冲突，也可以克隆存储库，并使用 IDE 或 Git 工具来查找和解决合并冲突。有关更多信息，请参阅 合并拉取请求。

排查项目和蓝图的问题

此部分可帮助您解决在 Amazon CodeCatalyst 中使用项目和蓝图时可能遇到的一些常见问题。

带 AWS Fargate 的 Java API 蓝图缺少 apache-maven-3.8.6 的依赖项

问题：对于通过带 AWS Fargate 的 Java API 蓝图创建的项目，工作流失败并显示一个错误，指示缺少 `apache-maven-3.8.6` 依赖项。工作流失败，并显示与以下示例类似的输出：

```
Step 8/25 : RUN wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-
maven-3.8.6-bin.tar.gz -P /tmp
---> Running in 1851ce6f4d1b
[91m--2023-03-10 01:24:55--  https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz
[0m[91mResolving dlcdn.apache.org (dlcdn.apache.org)...
[0m[91m151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443...
[0m[91mconnected.
[0m[91mHTTP request sent, awaiting response... [0m[91m404 Not Found
2023-03-10 01:24:55 ERROR 404: Not Found.
[0mThe command '/bin/sh -c wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz -P /tmp' returned a non-zero code: 8
[Container] 2023/03/10 01:24:55 Command failed with exit status 8
```

解决方案：执行以下步骤来更新蓝图 Dockerfile。

1. 在搜索栏中，输入 apache-maven-3.8.6 以在使用带 AWS Fargate 的 Java API 蓝图创建的项目中找到 dockerfile。
2. 更新 Dockerfile（/static-assets/app/Dockerfile）以将 maven:3.9.0-amazoncorretto-11 用作基础映像，并移除 apache-maven-3.8.6 程序包的依赖项。
3. （推荐）我们还建议您将 Maven 堆大小更新为 6 GB。

以下是示例 Dockerfile。

```
FROM maven:3.9.0-amazoncorretto-11 AS builder

COPY ./pom.xml ./pom.xml
COPY src ./src/

ENV MAVEN_OPTS='-Xmx6g'

RUN mvn -Dmaven.test.skip=true clean package

FROM amazoncorretto:11-alpine

COPY —from=builder target/CustomerService-0.0.1.jar CustomerService-0.0.1.jar
EXPOSE 80
```

```
CMD ["java","-jar","-Dspring.profiles.active=prod","/CustomerService-0.0.1.jar", "—
server.port=80"]
```

现代三层 Web 应用程序蓝图工作流程OnPullRequest失败，出现 Amazon 权限错误 CodeGuru

问题：当我尝试为我的项目运行一个工作流时，此工作流无法运行，并显示以下消息：

```
Failed at codeguru_codereview: The action failed during runtime. View the action's logs
 for more details.
```

解决方案：此操作失败的一个可能原因可能是由于 IAM 角色策略中缺少权限，即您在连接 CodeCatalyst AWS 账户 中使用的服务角色版本缺少 code guru_codereview 操作成功运行所需的权限。要解决此问题，要么必须使用所需权限更新服务角色，要么必须将用于工作流程的服务角色更改为具有 Amazon CodeGuru 和 Amazon CodeGuru Reviewer 所需权限的服务角色。使用以下步骤，找到您的角色并更新角色策略权限，以使工作流能够成功运行。

> **Note**
>
> 这些步骤适用于中的以下工作流程 CodeCatalyst：
>
> - 为使用现代三层 Web 应用程序蓝图创建的项目提供OnPullRequest的工作流程。CodeCatalyst
> - 通过访问 Amazon CodeGuru 或 Amazon CodeGuru Reviewer 的操作将工作流程添加到项目中 CodeCatalyst 。

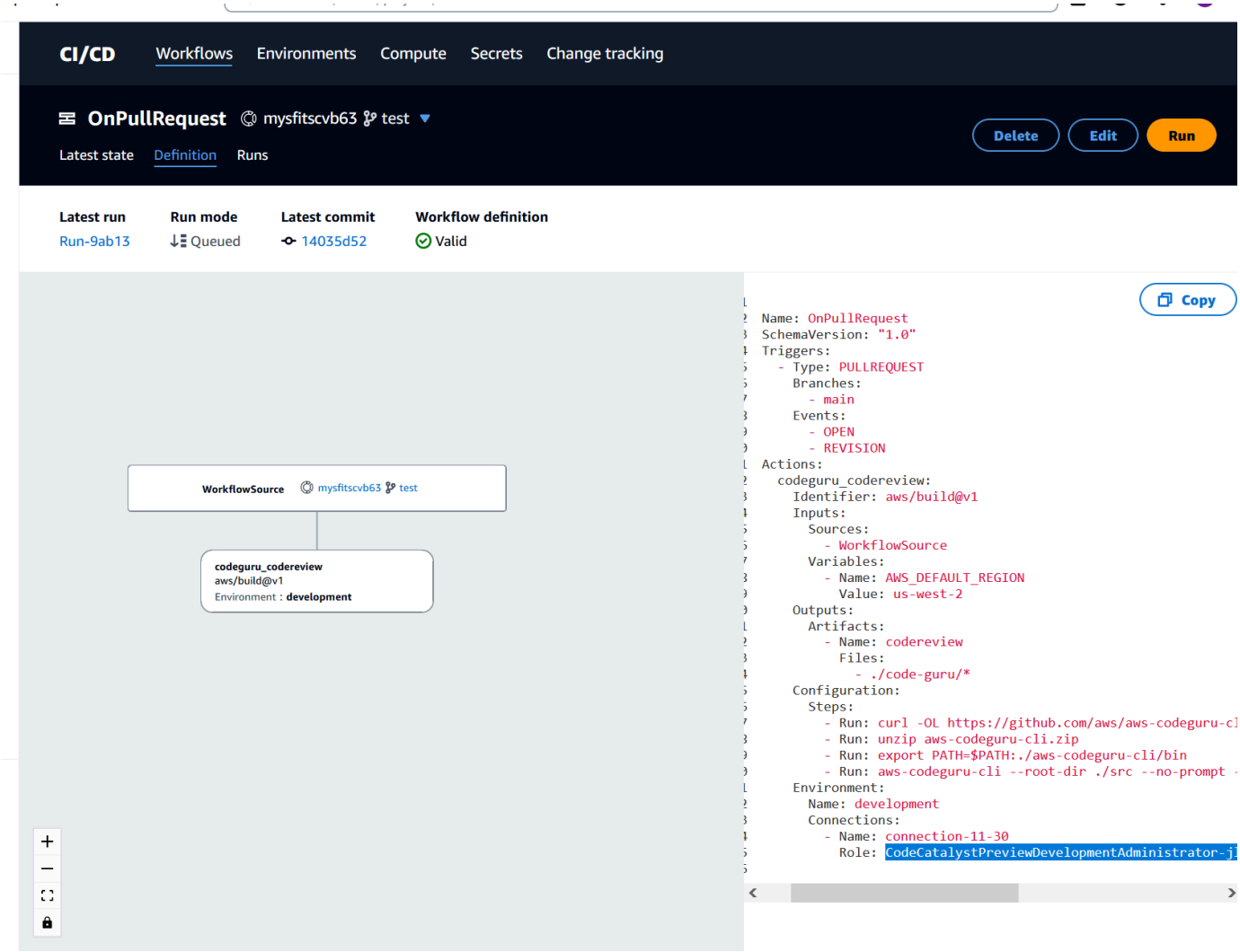
每个项目都包含工作流程，其操作使用的是与您的项目 AWS 账户 关联的用户提供的角色和环境 CodeCatalyst。包含操作及其指定策略的工作流存储在源存储库中的 /.codecatalyst/workflows 目录中。除非您要向现有工作流添加新的角色 ID，否则无需修改工作流 YAML。有关 YAML 模板元素和格式的信息，请参阅工作流 YAML 定义。

以下是编辑角色策略和验证工作流 YAML 时要遵循的高级步骤。

在工作流 YAML 中引用您的角色名称并更新策略

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。

2. 导航到您的 CodeCatalyst 空间。导航到您的项目。
3. 选择 CI/CD，然后选择工作流。
4. 选择标题为的工作流程OnPullRequest。选择定义选项卡。
5. 在工作流 YAML 中，在 codeguru_codereview 操作下的 `Role:` 字段中，记下角色名称。这是具有要在 IAM 中修改的策略的角色。以下示例显示了角色名称。



6. 请执行以下操作之一：

   - （推荐）将与您的项目关联的服务角色更新为亚马逊 CodeGuru 和亚马逊 CodeGuru 评论者所需的权限。该角色的名称为 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***`spaceName`*** 服务角色。继续执行后续步骤以在 IAM 中更新策略。

> **Note**
>
> 您必须拥有具有角色和策略的 AWS 管理员访问权限。 AWS 账户

- 将用于工作流程的服务角色更改为具有 Amazon CodeGuru 和 Amazon CodeGuru Reviewer 所需权限的服务角色，或者创建具有所需权限的新角色。

7. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。

   在 IAM 控制台中，找到步骤 5 中的角色，例如 `CodeCatalystPreviewDevelopmentRole`。
8. 在步骤 5 中的角色中，将权限策略更改为包含 `codeguru-reviewer:*` 和 `codeguru:*` 权限。添加这两项权限后，权限策略应类似于以下内容：
9. 修改政策后，返回 CodeCatalyst 并重新开始运行工作流程。

还在寻求如何解决您的问题吗？

您可以转到 Amazon CodeCatalyst 或填写支持反馈表单。在请求信息部分中的我们如何帮助您下，说明您是 Amazon CodeCatalyst 客户。请尽量提供详细信息，以便我们能最有效地解决您的问题。

排查开发环境的问题

要排查与开发环境相关的问题，请参阅以下部分。有关开发环境的更多信息，请参阅使用开发环境编写和修改代码 CodeCatalyst。

主题

- 排查 IDE 的问题
- 排查 devfile 的问题

由于出现配额问题，我无法创建开发环境

问题：我要在 CodeCatalyst 中创建一个开发环境，但显示了一个错误。控制台中的开发环境页面上显示一条消息，指示我已达到空间的存储限制。

可能的修复方法：根据您在项目或空间中的角色，您可以删除自己的一个或多个开发环境，或者如果您具有空间管理员角色，则可以删除其他用户创建的未使用的开发环境。您也可以决定将计费等级更改为一个包含更多存储空间的计费等级。

- 要查看存储限制，请查看 Amazon CodeCatalyst 空间的账单选项卡，以确定用量配额是否已达到允许的最大值。如果配额已达到最大值，请联系具有空间管理员角色的人员以移除不需要的开发环境或考虑更改计费等级。
- 要移除您创建的不再需要的任何开发环境，请参阅删除开发环境。

如果问题仍然存在，并且您的 IDE 中出现错误，请检查您是否具有让您能够创建开发环境的 CodeCatalyst 角色。空间管理员角色、项目管理员角色和贡献者角色都有创建开发环境所需的权限。有关更多信息，请参阅使用用户角色授予访问权限。

我无法将更改从开发环境推送到存储库中的特定分支

问题：我要将开发环境中的代码更改提交并推送到源存储库中的分支，但显示一个错误。

可能的修复方法：根据您在项目或空间中的角色，您可能无权将代码推送到项目中的源存储库。空间管理员角色、项目管理员角色和贡献者角色都有权将代码推送到项目中的存储库。

如果您具有贡献者角色但无法将代码推送到特定分支，则可能为该特定分支配置了一个分支规则，该规则禁止具有此角色的用户将代码推送到该特定分支。尝试将更改推送到其他分支，或者创建一个分支，然后将代码推送到该分支。有关更多信息，请参阅 使用分支规则管理分支允许的操作。

我的开发环境未恢复

问题：我的开发环境在停止后未恢复。

可能的修复方法：要修复此问题，请查看 Amazon CodeCatalyst 空间的账单选项卡，以确定用量配额是否已达到最大限制。如果配额已达到最大限制，请联系您的空间管理员来提高计费等级。

我的开发环境已断开连接

问题：我的开发环境在使用期间断开连接。

可能的修复方法：要修复此问题，请检查您的 Internet 连接。如果您未连接到 Internet，请连接开发环境并继续工作。

我的与 VPC 连接的开发环境出现问题

问题：我将 VPC 连接关联到我的开发环境，但出现错误。

可能的修复方法：Docker 使用了一种名为网桥网络的链路层设备，它使连接到同一网桥网络的容器能够进行通信。默认网桥通常使用 `172.17.0.0/16` 子网进行容器联网。如果环境实例的 VPC 子网使用的地址范围与 Docker 使用的相同，则可能会出现 IP 地址冲突。要解决由 Amazon VPC 和使用相同的 IPv4 CIDR 地址块的 Docker 导致的 IP 地址冲突，请配置一个与 `172.17.0.0/16` 中的 CIDR 块不同的 CIDR 块。

> **Note**
>
> 您不能更改现有 VPC 或子网的 IP 地址范围。

我找不到我的项目位于哪个目录

问题：我找不到我的项目位于哪个目录

可能的修复方法：要找到您的项目，请将目录更改为 `/projects`。您可以在此目录中找到您的项目。

我无法通过 SSH 连接到我的开发环境

要排查通过 SSH 与开发环境建立的连接的问题，您可以执行带 `-vvv` 选项的 `ssh` 命令，以显示有关如何解决问题的更多信息：

```
ssh -vvv codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
```

我无法通过 SSH 连接到我的开发环境，因为我的本地 SSH 配置缺失

如果本地 SSH 配置（`~/.ssh/config`）缺失或 `Host codecatalyst-dev-env*` 部分的内容已过期，则您将无法通过 SSH 连接到您的开发环境。要解决此问题，请删除 `Host codecatalyst-`

dev-env* 部分并重新从 SSH 访问权限模式执行第一条命令。有关更多信息，请参阅 使用 SSH 连接到开发环境。

我无法通过 SSH 连接到我的开发环境，因为我的 codecatalyst 个人资料的 AWS Config 有问题

确保您的 codecatalyst 个人资料的 AWS Config（~/.aws/config）与设置为 AWS CLI 与一起使用 CodeCatalyst中描述的项相匹配。如果不是这样，请删除 codecatalyst 的个人资料并重新从 SSH 访问权限模式执行第一条命令。有关更多信息，请参阅 使用 SSH 连接到开发环境。

当我使用单点登录账户登录 CodeCatalyst 时，无法创建开发环境

问题：当我以 SSO 用户身份登录 CodeCatalyst 控制台时，我在选择在空间中创建开发环境时收到一个未知的异常错误。当我选择创建开发环境并选择 IDE 以进行访问（例如 AWS Cloud9）时，我遇到了与以下内容类似的问题：

- CodeCatalyst 控制台中的开发环境页面在列表中显示带 FAILED 状态的开发环境。
- 这将显示一条与以下内容类似的错误消息：

```
An unknown exception happened

We encountered an unknown exception when launching your Dev Environment.
Mention your Dev Environment id error_message_ID if you want to report or
need any help.
```

可能的修复方法：

开发环境不适用于将 Active Directory 用作身份提供商的空间中的用户。该空间的管理员可以使用其他身份提供商来访问开发环境，例如 IAM Identity Center。有关规划支持身份联合验证的空间的更多信息，请参阅《CodeCatalyst Administrator Guide》中的 Planning your space that supports identity federation。

排查 IDE 的问题

要在 CodeCatalyst 中排查与 IDE 相关的问题，请参阅以下部分。有关 IDE 的更多信息，请参阅在 IDE 中创建开发环境。

主题

- 我在 AWS Cloud9 中的运行时映像版本不匹配

我在 AWS Cloud9 中的运行时映像版本不匹配

AWS Cloud9 使用了不同版本的前端资产和后端运行时映像。使用不同的版本可能会导致 Git 扩展和 AWS Toolkit 无法正常运行。要修复此问题，请导航到开发环境控制面板，停止开发环境，然后重新启动它。要使用 API 修复此问题，请使用 `UpdateDevEnvironment` API 更新运行时。有关更多信息，请参阅《Amazon CodeCatalyst API reference》中的 UpdateDevEnvironment。

我无法在 AWS Cloud9 中访问 **`/projects/projects`** 中的文件

AWS Cloud9 编辑器无法访问目录 `/projects/projects` 中的文件。要修复此问题，请使用 AWS Cloud9 终端访问您的文件或将其移至其他目录。

我无法使用自定义 devfile 在 AWS Cloud9 中启动我的开发环境

您的 devfile 映像可能与 AWS Cloud9 不兼容。要修复此问题，请查看存储库中的 devfile 和相应的开发环境，然后创建一个新的开发环境以继续。

我在 AWS Cloud9 中遇到了问题

有关其他问题，请查看 AWS Cloud9 User Guide 中的问题排查部分。

在 JetBrains 中，我无法通过 CodeCatalyst 连接到我的开发环境

要修复此问题，请检查您是否只安装了最新版本的 JetBrains。如果您有多个版本，请卸载旧版本，然后通过关闭 IDE 和浏览器来重新注册协议处理程序。然后，打开 JetBrains 并重新注册协议处理程序。

我无法为我的 IDE 安装 AWS Toolkit

要为 VS Code 修复此问题，请从 GitHub 手动安装 AWS Toolkit for Visual Studio Code。

要为 JetBrains 修复此问题，请从 GitHub 手动安装 AWS Toolkit for JetBrains。

在我的 IDE 中，我无法启动我的开发环境

要为 VS Code 修复此问题，请检查是否已安装最新版本的 VS Code 和 AWS Toolkit for Visual Studio Code。如果您没有最新版本，请更新并启动您的开发环境。有关更多信息，请参阅 Amazon CodeCatalyst for VS Code。

要为 JetBrains 修复此问题，请检查是否已安装最新版本的 JetBrains 和 AWS Toolkit for JetBrains。如果您没有最新版本，请更新并启动您的开发环境。有关更多信息，请参阅适用于 JetBrains 的 Amazon CodeCatalyst。

排查 devfile 的问题

要在 CodeCatalyst 中排查与 devfile 相关的问题，请参阅以下部分。有关 devfile 的更多信息，请参阅配置开发环境的 devfile。

主题

即使我在自定义 devfile 中实现了自定义映像，我的开发环境还是使用默认通用 devfile

如果 CodeCatalyst 在启动使用自定义 devfile 的开发环境时遇到错误，则开发环境使用默认通用 devfile。要解决此问题，您可以在 `/aws/mde/logs/devfile.log` 下的日志中查看确切的错误。您还可以在日志中检查 `postStart` 执行是否成功：`/aws/mde/logs/devfileCommand.log`。

我的项目未使用默认通用 devfile 在我的开发环境中进行构建

要解决此问题，请检查您是否未使用自定义 devfile。如果您未使用自定义 devfile，请在项目的源存储库中查看 `devfile.yaml` 文件以查找并纠正所有错误。

我需要移动开发环境的存储库 devfile

您可以将 /projects/devfile.yaml 中的默认 devfile 移至您的源代码存储库。要更新 devfile 的位置，请使用以下命令：/aws/mde/mde start --location *repository-name*/devfile.yaml。

我在启动 devfile 时遇到了问题

如果启动 devfile 时出现问题，它将进入恢复模式，这样您仍能连接到您的环境并修复 devfile。在恢复模式中，运行 /aws/mde/mde status 将不会包含 devfile 的位置。

```
{
    "status": "STABLE"
}
```

您可以查看 /aws/mde/logs 下的日志中的错误，修复 devfile，然后重试运行 /aws/mde/mde start。

我无法确定如何查看我的 devfile 状态

可以通过运行 /aws/mde/mde status 来查看 devfile 状态。运行此命令后，您可能会看到以下内容之一：

- {"status": "STABLE", "location": "devfile.yaml" }

  这表明 devfile 是正确的。
- {"status": "STABLE" }

  这表示 devfile 无法启动并且已进入恢复模式。

您可以在 /aws/mde/logs/devfile.log 下的日志中查看确切的错误。

您还可以在日志中检查 postStart 执行是否成功：/aws/mde/logs/devfileCommand.log。

有关更多信息，请参阅为开发环境指定通用 devfile 映像。

我的 devfile 与最新映像中提供的工具不兼容

在您的开发环境中，如果 latest 工具没有特定项目所需的工具，则 devfile 或 devfile postStart 可能会失败。要修复此问题，请执行以下操作：

1. 导航到您的 devfile。
2. 在您的 devfile 中，更新至精细映像版本，而不是 `latest`。其内容可能与以下内容类似：

```
components:
  - container:
      image: public.ecr.aws/amazonlinux/universal-image:1.0
```

3. 使用更新后的 devfile 创建新的开发环境。

排查工作流问题

要在 Amazon CodeCatalyst 中排查与工作流相关的问题，请参阅以下部分。有关工作流的更多信息，请参阅使用工作流进行构建、测试和部署。

主题

- 如何修复“工作流非活动”消息？
- 如何修复“工作流定义有 n 个错误”错误？
- 如何修复“找不到凭证”和“ExpiredToken”错误？
- 如何修复“无法连接到服务器”错误？
- 为什么可视化编辑器中缺少 CodeDeploy 字段？
- 如何修复 IAM 功能错误？
- 如何修复“npm install”错误？
- 为什么多个工作流具有相同的名称？
- 能否将我的工作流定义文件存储在另一个文件夹中？
- 如何按顺序向我的工作流中添加操作？
- 为什么我的工作流成功验证但在运行时失败？
- 自动发现功能未发现我的操作的任何报告
- 配置成功标准后，我对自动发现的报告执行操作失败
- 自动发现功能生成我不想要的报告
- 自动发现功能为单个测试框架生成许多小报告
- CI/CD 下列出的工作流与源存储库中的工作流不匹配
- 我无法创建或更新工作流

如何修复“工作流非活动”消息？

问题：在 CodeCatalyst 控制台的 CI/CD 下，在工作流中，您的工作流显示以下消息：

```
Workflow is inactive.
```

此消息表示工作流定义文件包含一个不适用于您当前所在分支的触发器。例如，您的工作流定义文件可能包含引用 main 分支的 PUSH 触发器，但您位于功能分支上。由于您在功能分支中所做的更改不适用于 main，并且不会启动 main 中的工作流运行，因此 CodeCatalyst 会停用该分支上的工作流并将其标记为 Inactive。

可能的修复方法：

如果要在功能分支上启动工作流，可以执行以下操作：

- 在您的功能分支中，在工作流定义文件中，从 Triggers 部分中移除 Branches 属性，使其如下所示：

  ```
  Triggers:
    - Type: PUSH
  ```

  此配置会使触发器在推送到任何分支（包括功能分支）时激活。如果触发器被激活，CodeCatalyst 将使用您要推送到的任何分支中的工作流定义文件和源文件启动工作流运行。
- 在您的功能分支中，在工作流定义文件中，移除 Triggers 部分并手动运行工作流。
- 在您的功能分支中，在工作流定义文件中，更改 PUSH 部分，使其引用您的功能分支而不是另一个分支（例如 main）。

> **Important**
>
> 如果您不打算将这些更改合并回 main 分支，请注意不要提交这些更改。

有关编辑工作流定义文件的更多信息，请参阅创建工作流。

有关触发器的更多信息，请参阅使用触发器自动启动工作流运行。

如何修复“工作流定义有 *n* 个错误”错误？

问题：您看到以下任何错误消息：

错误 1：

在 CI/CD 的工作流页面中，在您的工作流名称下，您看到：

```
Workflow definition has n errors
```

错误 2：

编辑工作流时，选择验证按钮，CodeCatalyst 控制台顶部显示以下消息：

```
The workflow definition has errors. Fix the errors and choose Validate to
verify your changes.
```

错误 3：

导航到工作流的详细信息页面后，您在工作流定义字段中看到以下错误：

```
n errors
```

可能的修复方法：

- 依次选择 CI/CD、工作流和出错的工作流的名称。在靠近顶部的工作流定义字段中，选择错误链接。错误的详细信息出现在页面底部。按照错误中的问题排查提示修复问题。
- 确保工作流定义文件是 YAML 文件。
- 确保工作流定义文件中的 YAML 属性嵌套在正确的级别。要了解应如何将属性嵌套在工作流定义文件中，请参阅工作流 YAML 定义，或查阅您的操作文档，该文档链接自添加操作到工作流。
- 确保正确转义星号（*）和其它特殊字符。要转义这些字符，请添加单引号或双引号。例如：

  ```
  Outputs:
    Artifacts:
      - Name: myartifact
        Files:
          - "**/*"
  ```

  有关工作流定义文件中特殊字符的更多信息，请参阅语法准则和惯例。
- 确保工作流定义文件中的 YAML 属性使用正确的大小写。有关大小写规则的更多信息，请参阅语法准则和惯例。要确定每个属性的正确大小写，请参阅工作流 YAML 定义，或查阅您的操作文档，该文档链接自添加操作到工作流。
- 确保 SchemaVersion 属性存在并在工作流定义文件中设置为正确的版本。有关更多信息，请参阅 SchemaVersion。

- 确保工作流定义文件中的 Triggers 部分包含所有必需的属性。要确定所需的属性，请在可视化编辑器中选择触发器并查找缺少信息的字段，或者查阅Triggers处的触发器参考文档。
- 确保工作流定义文件中的 DependsOn 属性配置正确，并且不会引入循环依赖关系。有关更多信息，请参阅 顺序操作。
- 确保工作流定义文件中的 Actions 部分至少包含一个操作。有关更多信息，请参阅 操作。
- 确保每个操作都包含所有必需的属性。要确定所需的属性，请在可视化编辑器中选择操作并查找缺少信息的字段，或者查阅您的操作文档，该文档链接自添加操作到工作流。
- 确保所有输入构件都有相应的输出构件。有关更多信息，请参阅 定义输出构件。
- 确保导出在一个操作中定义的变量，以便可以在其它操作中使用它们。有关更多信息，请参阅 导出变量以便其他操作使用。

如何修复“找不到凭证”和“ExpiredToken”错误？

问题：在完成教程：将应用程序部署到 Amazon EKS 时，您在开发计算机的终端窗口中看到以下一条或两条错误消息：

```
Unable to locate credentials. You can configure credentials by running "aws configure".
```

```
ExpiredToken: The security token included in the request is expired
```

可能的修复方法：

这些错误表明您用于访问 AWS 服务的凭证已过期。在此情况下，请不要运行 aws configure 命令。相反，请按照以下说明刷新您的 AWS 访问密钥和会话令牌。

刷新您的 AWS 访问密钥和会话令牌

1. 确保您拥有在完成 Amazon EKS 教程时所使用的用户（codecatalyst-eks-user）的 AWS 访问门户 URL、用户名和密码。完成该教程的步骤 1：设置开发机器后，您应该已经配置了这些项。

   > **Note**
   >
   > 如果您没有这些信息，请转至 IAM Identity Center 中的 codecatalyst-eks-user 详细信息页面，选择重置密码和生成一次性密码 [...]，然后再次选择重置密码以在屏幕上显示信息。

2. 请执行以下操作之一：

   - 将 AWS 访问门户 URL 粘贴到浏览器的地址栏中。

     或

   - 如果 AWS 访问门户页面已加载，请刷新该页面。

3. 如果您尚未登录，请使用 codecatalyst-eks-user 的用户名和密码登录。
4. 选择 AWS 账户，然后选择您为其分配了 codecatalyst-eks-user 用户和权限集的 AWS 账户的名称。
5. 在权限集名称（codecatalyst-eks-permission-set）旁边，选择命令行访问或以编程方式访问。
6. 复制页面中间的命令。其内容与以下内容类似：

   ```
   export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
   export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
   export AWS_SESSION_TOKEN="session-token"
   ```

   ...其中 *session-token* 是一个长随机字符串。
7. 将命令粘贴到开发计算机上的终端提示中，然后按 Enter。

   新密钥和会话令牌已加载。

   现在，您已经刷新了凭证。AWS CLI、eksctl 和 kubectl 命令现在应该可以正常运行了。

如何修复“无法连接到服务器”错误？

问题：在完成教程：将应用程序部署到 Amazon EKS 中描述的教程时，您在开发计算机的终端窗口中看到类似以下内容的错误消息：

Unable to connect to the server: dial tcp: lookup *long-string*.gr7.us-west-2.eks.amazonaws.com on *1.2.3.4:5*: no such host

可能的修复方法：

此错误通常表示 kubectl 实用程序用于连接到 Amazon EKS 集群的凭证已过期。要解决此问题，请在终端提示符处输入以下命令来刷新凭证：

```
aws eks update-kubeconfig --name codecatalyst-eks-cluster --region us-west-2
```

其中：

- *codecatalyst-eks-cluster* 替换为 Amazon EKS 集群的名称。
- *us-west-2* 替换为部署集群的 AWS 区域。

为什么可视化编辑器中缺少 CodeDeploy 字段？

问题：您正在使用部署到 Amazon ECS 操作，但在工作流的可视化编辑器中看不到诸如 CodeDeploy AppSpec 之类的 CodeDeploy 字段。之所以出现此问题，可能是因为您在服务字段中指定的 Amazon ECS 服务未配置为执行蓝绿部署。

可能的修复方法：

- 在部署到 Amazon ECS 操作的配置选项卡上选择其它 Amazon ECS 服务。有关更多信息，请参阅使用工作流部署到 Amazon ECS。
- 将选定的 Amazon ECS 服务配置为执行蓝绿部署。有关配置蓝绿部署的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的使用 CodeDeploy 进行蓝绿部署。

如何修复 IAM 功能错误？

问题：您正在使用部署 CloudFormation 堆栈操作，您在部署 CloudFormation 堆栈操作的日志中看到 `##[error] requires capabilities: [`*capability-name*`]`。

可能的修复方法：完成以下步骤，将该功能添加到工作流定义文件中。有关 IAM 功能的更多信息，请参阅《IAM 用户指南》中的确认 CloudFormation 模板中的 IAM 资源。

Visual

使用可视化编辑器添加 IAM 功能

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择可视化。

7. 在工作流图表中，选择部署 CloudFormation 堆栈操作。
8. 选择配置选项卡。
9. 在底部，选择高级 - 可选。
10. 在功能下拉列表中，选中错误消息中提到的功能旁边的复选框。如果列表中没有该功能，请使用 YAML 编辑器添加该功能。
11. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
12. 选择提交，输入提交消息，然后再次选择提交。
13. 如果新的工作流运行未自动启动，请手动运行工作流以查看更改是否修复了错误。有关手动运行工作流的更多信息，请参阅手动启动工作流运行。

YAML

使用 YAML 编辑器添加 IAM 功能

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 选择您的项目。
3. 在导航窗格中，选择 CI/CD，然后选择工作流。
4. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在部署 CloudFormation 堆栈操作中，添加一个 `capabilities` 属性，如下所示：

```
DeployCloudFormationStack:
  Configuration:
    capabilities: capability-name
```

将 *capability-name* 替换为错误消息中显示的 IAM 功能的名称。使用逗号来列出多个功能（不含空格）。有关更多信息，请参阅'部署 CloudFormation 堆栈'动作 YAML中对 `capabilities` 属性的说明。

8. （可选）选择验证，在提交之前验证工作流的 YAML 代码。
9. 选择提交，输入提交消息，然后再次选择提交。
10. 如果新的工作流运行未自动启动，请手动运行工作流以查看更改是否修复了错误。有关手动运行工作流的更多信息，请参阅手动启动工作流运行。

如何修复“npm install”错误？

问题：AWS CDK 部署操作或 AWS CDK 引导操作因 npm install 错误而失败。之所以发生此错误，可能是因为您将 AWS CDK 应用程序依赖项存储在操作无法访问的私有节点程序包管理器（npm）注册表中。

可能的修复方法：按照以下说明使用其它注册表和身份验证信息更新 AWS CDK 应用程序的 cdk.json 文件。

开始前的准备工作

1. 为您的身份验证信息创建密钥。您将在 cdk.json 文件中引用这些密钥，而不是提供等效的明文信息。要创建密钥，请执行以下操作：

   a. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。

   b. 选择您的项目。

   c. 在导航窗格中，选择 CI/CD，然后选择密钥。

   d. 创建两个具有以下属性的密钥：

第一个密钥	第二个密钥
名称：npmUsername 值：*npm-username*，其中 *npm-username* 是用于向私有 npm 注册表进行身份验证的用户名。 （可选）描述：The username used to authenticate to the private npm registry.	名称：npmAuthToken 值：*npm-auth-token*，其中 *npm-auth-token* 是用于向私有 npm 注册表进行身份验证的访问令牌。有关 npm 访问令牌的更多信息，请参阅 npm 文档中的 About access tokens。 （可选）描述：The access token used to authenticate to the private npm registry.

   有关密钥的更多信息，请参阅使用密钥遮蔽数据。

2. 将密钥作为环境变量添加到 AWS CDK 操作中。该操作将在运行时用实际值替换变量。要添加密钥，请执行以下操作：

a. 在导航窗格中，选择 CI/CD，然后选择工作流。

b. 选择工作流的名称。您可以按定义工作流的源存储库或分支名称筛选，也可以按工作流名称或状态筛选。

c. 选择编辑。

d. 选择可视化。

e. 在工作流图表中，选择 AWS CDK 操作。

f. 选择输入选项卡。

g. 添加两个具有以下属性的变量：

第一个变量	第二个变量
名称：NPMUSER	名称：NPMTOKEN
值：${Secrets.npmUsername}	值：${Secrets.npmAuthToken}

现在，您有两个包含对密钥的引用的变量。

工作流定义文件 YAML 代码应该类似于以下内容：

> **Note**
>
> 下面的代码示例来自 AWS CDK 引导操作；AWS CDK 部署操作与此类似。

```
Name: CDK_Bootstrap_Action
SchemaVersion: 1.0
Actions:
  CDKBootstrapAction:
    Identifier: aws/cdk-bootstrap@v2
    Inputs:
      Variables:
        - Name: NPMUSER
          Value: ${Secrets.npmUsername}
        - Name: NPMTOKEN
          Value: ${Secrets.npmAuthToken}
      Sources:
```

```
        - WorkflowSource
    Environment:
      Name: Dev2
      Connections:
        - Name: account-connection
          Role: codecatalystAdmin
    Configuration:
      Parameters:
        Region: "us-east-2"
```

现在，您已准备好在 cdk.json 文件中使用 NPMUSER 和 NPMTOKEN 变量了。继续下一过程。

更新 cdk.json 文件

1. 转到 AWS CDK 项目的根目录，然后打开 cdk.json 文件。
2. 找到 "app": 属性，并将其更改为包含以####显示的代码：

> **Note**
>
> 下面的示例代码来自 TypeScript 项目。如果您使用的是 JavaScript 项目，则代码看起来很相似，但并不相同。

```
{
  "app": "npm set registry=https://your-registry/folder/CDK-package/ --
userconfig .npmrc && npm set //your-registry/folder/CDK-package/:always-auth=true
 --userconfig .npmrc && npm set //your-registry/folder/CDK-package/:_authToken=
\"${NPMUSER}\":\"${NPMTOKEN}\" && npm install  && npx ts-node --prefer-ts-exts bin/
hello-cdk.ts|js",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
```

```
...
```

3. 在以####突出显示的代码中，进行如下替换：

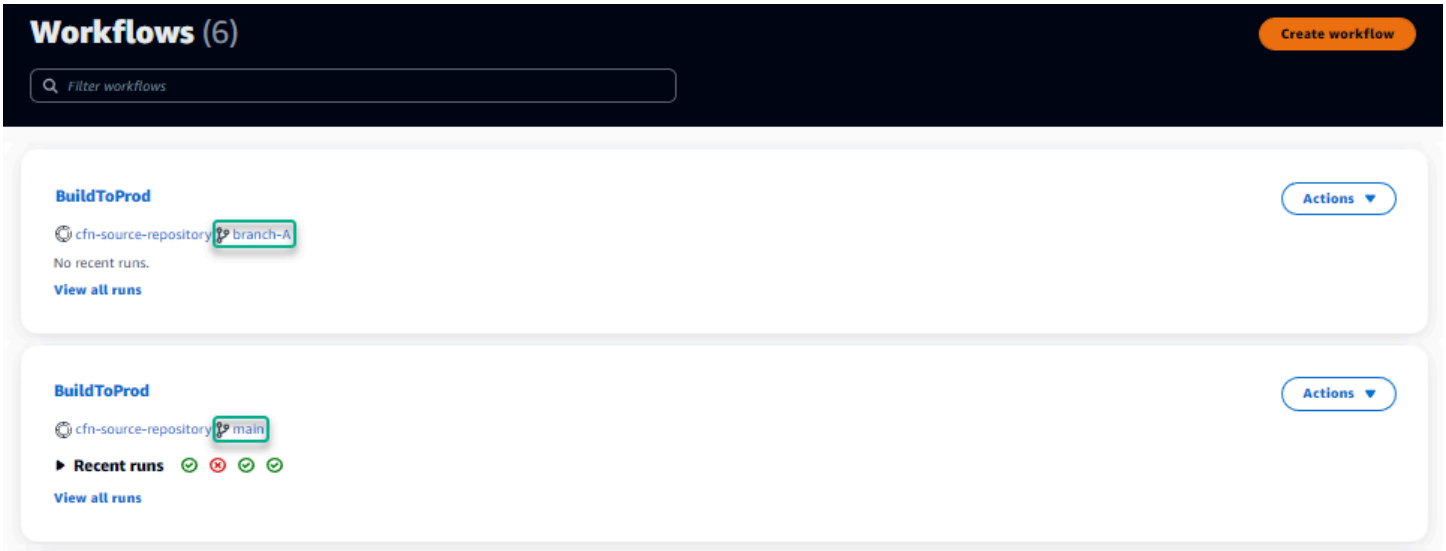
   - 将 *your-registry/folder/CDK-package/* 替换为私有注册表中 AWS CDK 项目依赖项的路径。
   - 将 *hello-cdk.ts|.js* 替换为入口点文件的名称。这可能是 `.ts`（TypeScript）或 `.js`（JavaScript）文件，具体取决于您使用的语言。

   > **Note**
   >
   > 该操作将使用您在密钥中指定的 npm 用户名和访问令牌替换 *NPMUSER* 和 *NPMTOKEN* 变量。

4. 保存 `cdk.json` 文件。
5. 手动重新运行该操作以查看更改是否修复了错误。有关手动运行操作的更多信息，请参阅手动启动工作流运行。

为什么多个工作流具有相同的名称？

工作流存储在每个存储库的每个分支上。如果两个不同的工作流存在于不同的分支中，则它们可以具有相同的名称。在“工作流”页面中，您可以通过查看分支名称来区分同名工作流。有关更多信息，请参阅使用 Amazon 中的分支来整理源代码 CodeCatalyst。



能否将我的工作流定义文件存储在另一个文件夹中？

否，必须将所有工作流定义文件存储在 `.codecatalyst/workflows` 文件夹或该文件夹的子文件夹中。如果您使用的是包含多个逻辑项目的单一存储库，请将所有工作流定义文件放在 `.codecatalyst/workflows` 文件夹或其子文件夹中，然后使用触发器中的文件已更改字段（可视化编辑器）或 `FilesChanged` 属性（YAML 编辑器）在指定的项目路径上自动触发工作流。有关更多信息，请参阅添加触发器到工作流和示例：带有推送、分支和文件的触发器。

如何按顺序向我的工作流中添加操作？

默认情况下，当您向工作流中添加操作时，该操作没有依赖关系，将与其它操作并行运行。

如果要按顺序排列操作，可以通过设置 `DependsOn` 字段来设置对另一个操作的依赖关系。您还可以将操作配置为使用作为其它操作输出的构件或变量。有关更多信息，请参阅 顺序操作。

为什么我的工作流成功验证但在运行时失败？

如果您使用 `Validate` 按钮验证了工作流，但工作流还是失败了，这可能是因为验证器存在限制。

在提交过程中，工作流配置中引用 CodeCatalyst 资源（如密钥、环境或实例集）的任何错误都不会注册。如果使用了任何无效的引用，则只有在运行工作流时才会发现错误。同样，如果操作配置中存在任何错误，例如缺少必需字段或操作属性拼写错误，则只有在运行工作流时才会发现错误。有关更多信息，请参阅 创建工作流。

自动发现功能未发现我的操作的任何报告

问题：我为运行测试的操作配置了自动发现功能，但 CodeCatalyst 未发现任何报告。

可能的修复方法：这可能是由许多问题引起的。尝试以下一种或多种解决方案：

- 确保用于运行测试的工具以 CodeCatalyst 可以理解的格式之一生成输出。例如，如果您希望 `pytest` 可让 CodeCatalyst 发现测试和代码覆盖率报告，请包含以下参数：

  ```
  --junitxml=test_results.xml --cov-report xml:test_coverage.xml
  ```

  有关更多信息，请参阅 质量报告类型。
- 确保输出的文件扩展名与所选格式一致。例如，当配置 `pytest` 以 `JUnitXML` 格式生成结果时，请检查文件扩展名是否为 `.xml`。有关更多信息，请参阅 质量报告类型。

- 确保 IncludePaths 配置为包含整个文件系统（**/*），除非您有意排除某些文件夹。同样，确保 ExcludePaths 不排除您希望报告所在的目录。
- 如果您手动将报告配置为使用特定的输出文件，则该报告将排除在自动发现之外。有关更多信息，请参阅 质量报告 YAML 示例。
- 自动发现功能可能找不到报告，因为操作在生成任何输出之前就失败了。例如，构建操作可能在运行任何单元测试之前就失败了。

配置成功标准后，我对自动发现的报告执行操作失败

问题：当我启用自动发现并配置成功标准时，有些报告不符合成功标准，因此操作失败。

可能的修复方法：要解决此问题，请尝试以下一种或多种解决方案：

- 修改 IncludePaths 或 ExcludePaths 以排除您不感兴趣的报告。
- 更新成功标准以使所有报告能够通过。例如，如果发现两份报告，其中一份的行覆盖率为 50%，另一份的行覆盖率为 70%，则将最小行覆盖率调整为 50%。有关更多信息，请参阅 成功标准。
- 将失败的报告转换为手动配置的报告。这使您能够为该特定报告配置不同的成功标准。有关更多信息，请参阅 配置报告的成功标准。

自动发现功能生成我不想要的报告

问题：当我启用自动发现功能时，该功能生成我不想要的报告。例如，CodeCatalyst 为存储在 node_modules 中的应用程序依赖关系中包含的文件生成代码覆盖率报告。

可能的修复方法：您可以调整 ExcludePaths 配置以排除不需要的文件。例如，要排除 node_modules，请添加 node_modules/**/*。有关更多信息，请参阅 包含/排除路径。

自动发现功能为单个测试框架生成许多小报告

问题：当我使用某些测试和代码覆盖率报告框架时，我注意到自动发现功能生成大量报告。例如，在使用 Maven Surefire Plugin 时，自动发现功能为每个测试类生成不同的报告。

可能的修复方法：您的框架也许能够将输出聚合到单个文件中。例如，如果您使用的是 Maven Surefire Plugin，则可以使用 npx junit-merge 手动聚合文件。完整的表达式可能如下所示：

```
mvn test; cd test-package-path/surefire-reports && npx junit-merge -d ./ && rm
 *Test.xml
```

CI/CD 下列出的工作流与源存储库中的工作流不匹配

问题：CI/CD 的工作流页面上显示的工作流与源存储库的 ~/.codecatalyst/workflows/ 文件夹中的工作流不匹配。您可能会看到以下不匹配情况：

- 一个工作流显示在工作流页面上，但源存储库中不存在相应的工作流定义文件。
- 源存储库中存在工作流定义文件，但相应的工作流未显示在工作流页面上。
- 源存储库和工作流页面中都存在一个工作流，但两者不同。

如果工作流页面没有时间刷新，或者超过了工作流配额，则可能会出现此问题。

可能的修复方法：

- 等待。在提交到源后，您通常需要等待两三秒钟才能在工作流页面上看到更改。
- 如果您已超过工作流配额，请执行以下操作之一：

> **Note**
>
> 要确定是否超过了工作流配额，请查看 中的工作流程配额 CodeCatalyst，并将记录的配额与源存储库中或工作流页面上的工作流进行交叉检查。没有错误消息表明已超出配额，因此您必须自行调查。

  - 如果您已超过每个空间的最大工作流数配额，请删除一些工作流，然后对工作流定义文件执行测试提交。测试提交的一个例子可能是在文件中添加一个空格。
  - 如果您已超过最大工作流定义文件大小配额，请更改工作流定义文件以缩短其长度。
  - 如果您已超过单个源事件中处理的最大工作流文件数配额，请执行多次测试提交。修改为少于每次提交的最大工作流数。

我无法创建或更新工作流

问题：我想创建或更新工作流，但当我尝试提交更改时出现错误。

可能的修复方法：根据您在项目或空间中的角色，您可能无权将代码推送到项目中的源存储库。工作流的 YAML 文件存储在存储库中。有关更多信息，请参阅 工作流定义文件。空间管理员角色、项目管理员角色和贡献者角色都有权将代码提交和推送到项目中的存储库。

如果您具有贡献者角色，但无法在特定分支中创建或提交对工作流 YAML 的更改，则可能是为该分支配置了分支规则，阻止拥有该角色的用户向该特定分支推送代码。尝试在不同的分支中创建工作流，或者将您的更改提交到其它分支。有关更多信息，请参阅 使用分支规则管理分支允许的操作。

排查事务问题

以下信息有助于您排查 CodeCatalyst 中事务的常见问题。

主题

- 我无法为我的事务选择受让人

我无法为我的事务选择受让人

问题：创建事务时，受让人列表为空。

可能的修复方法：受让人列表直接链接到列为项目成员的 CodeCatalyst 用户。要验证用户个人资料访问是否正常运行，请选择配置文件图标，然后选择用户个人资料。如果用户个人资料信息未填充，请检查运行状况报告是否存在任何事件。如果这些信息已填充，请提交服务单。

排查 CodeCatalyst 中的搜索问题

要在 CodeCatalyst 中排查与搜索相关的问题，请参阅以下部分。有关工作流的更多信息，请参阅在 CodeCatalyst 中搜索代码、事务、项目和用户。

主题

- 我在项目中找不到用户
- 我在项目或空间中找不到想要的内容
- 当我浏览页面时，搜索结果的数量不断变化
- 我的搜索查询未完成

我在项目中找不到用户

问题：当我尝试查看用户的详细信息时，我在项目中看不到用户信息。

可能的修复方法：搜索功能目前不支持在项目中搜索用户。要搜索有权访问您的空间的用户，请在 QuickSearch 中切换到此空间，或者移除您可能使用高级查询语言指定的任何项目筛选条件。

我在项目或空间中找不到想要的内容

问题：当我尝试搜索特定信息时，结果不出现。

可能的修复方法：内容更新可能需要几秒钟才能在搜索结果中更新。大型更新可能需要几分钟时间。

对于最近未更新的资源，您可能需要优化搜索。您可以通过添加更多关键字或使用高级查询语言进行优化。有关优化查询的更多信息，请参阅细化搜索查询。

当我浏览页面时，搜索结果的数量不断变化

问题：当我转到下一页时，搜索结果的数量似乎发生了变化，因此不清楚总共有多少结果。

可能的修复方法：在浏览搜索结果页面时，您可能会看到与您的查询匹配的搜索结果数量发生了变化。结果数量可能会更新，以反映您在浏览页面时发现的更准确的匹配项数量。

当您浏览结果时，您可能会看到以下消息：No results for "test"。如果您无权访问其余结果，则会收到此消息。

我的搜索查询未完成

问题：我的搜索查询结果没有显示，而且似乎花了太长时间。

可能的修复方法：当空间中同时进行许多搜索时，无论是程序性的还是由于团队活动频繁，您的搜索可能无法完成。如果您正在运行程序化搜索，请暂停或减少搜索。否则，几秒钟后再试一次。

排查扩展问题

要在 CodeCatalyst 中排查与扩展相关的问题，请参阅以下部分。有关扩展的更多信息，请参阅为带有扩展程序的项目添加功能 CodeCatalyst。

主题

- 我看不到对链接的第三方存储库的更改，也无法搜索这些更改的结果

我看不到对链接的第三方存储库的更改，也无法搜索这些更改的结果

问题：我的第三方存储库中的更改未显示在 CodeCatalyst 中。

可能的修复方法：CodeCatalyst 目前不支持检测已链接存储库的默认分支中的更改。要更改已链接存储库的默认分支，您必须先取消该存储库与 CodeCatalyst 的链接，更改默认分支，然后重新链接该存

储库。有关更多信息，请参阅 在中关联 GitHub 存储库、Bitbucket 存储库、 GitLab 项目存储库和 Jira 项目 CodeCatalyst。

排查与您的空间关联的账户问题

在 CodeCatalyst 中，您可以为空间添加 AWS 账户，以便授予对资源的权限和进行计费。以下信息有助于您排查 CodeCatalyst 中关联账户的常见问题。

主题

我的 AWS 账户连接请求收到无效令牌错误

问题：使用连接令牌创建连接请求时，页面不接受令牌并显示错误，说明令牌无效。

可能的修复方法：确保提供您要添加到空间的账户 ID。您必须拥有 AWS 账户的管理权限，或者能够与管理员合作添加账户。

选择验证账户后，AWS 管理控制台中将打开一个新的浏览器窗口。在控制台端登录时需要使用同一账户。验证以下内容后再试一次：

- 您使用要添加到空间的同一 AWS 账户登录到 AWS 管理控制台。
- 您已登录 AWS 管理控制台，且区域设置为您空间的正确区域。
- 如果您从计费页面进入，并希望将 AWS 账户添加为空间的指定计费账户，请确保该账户作为其它一个或多个空间的计费账户没有达到配额。

我的 Amazon CodeCatalyst 项目工作流失败，且配置的账户、环境或 IAM 角色出错

问题：如果工作流运行时没有找到与空间相关联的已配置账户或 IAM 角色，则必须在工作流 YAML 中手动填写角色、连接和环境字段。查看失败的工作流操作，并注意错误消息是否如下：

- 角色不适用于与环境关联的连接。
- 操作失败。状态：“失败”；为账户连接或环境提供的值无效。确认连接与您的空间关联，并且环境与您的项目关联。
- 操作失败。状态：“失败”；为 IAM 角色提供的值无效。确认名称存在，IAM 角色已添加到账户连接中，并且该连接已与您的 Amazon CodeCatalyst 空间关联

可能的修复方法：确保工作流 YAML 字段中的环境、连接和角色值准确无误。需要环境的 CodeCatalyst 工作流操作是运行 AWS 资源或生成 AWS 资源堆栈的构建或部署操作。

选择失败的工作流操作块，然后选择可视化。选择配置选项卡。如果未填写环境、连接名称和角色名称字段，则需要手动更新工作流。使用以下步骤编辑工作流 YAML：

- 展开 `/.codecatalyst` 目录，然后展开 `/workflows` 目录。打开工作流 YAML 文件。确保在您为工作流配置的 YAML 中指定了 IAM 角色和账户信息。示例：

```
Actions:
  cdk_bootstrap:
    Identifier: action-@v1
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: Staging
      Connections:
        - Name: account-connection
          Role: build-role
```

  使用 AWS 资源运行 CodeCatalyst 工作流构建和部署操作需要 Environment、Connection 和 Role 属性。有关示例，请参阅 CodeCatalyst 构建操作参考 YAML 参数中的 Environment、Connections 和 Role。
- 确保您的空间中添加了一个账户，并确保该账户中添加了一个或多个适当的 IAM 角色。如果您拥有空间管理员角色，则可以调整或添加账户。有关更多信息，请参阅 允许在已连接的情况下访问 AWS 资源 AWS 账户。

我需要一个关联的账户、角色和环境来创建项目

问题：在项目创建选项中，我的项目在空间中没有添加可用的账户，或者我需要在空间中添加另一个账户以供项目使用。

可能的修复方法：对于您的空间，如果您具有空间管理员角色，则可以添加已授权的 AWS 账户，以便将其添加到您的项目中。您还必须拥有一个具有管理权限或可以与 AWS 管理员合作处理的 AWS 账户。

要确保账户和角色在项目创建屏幕中可用，必须首先添加账户和角色。有关更多信息，请参阅 允许在已连接的情况下访问 AWS 资源 AWS 账户。

您可以选择使用名为 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色策略的角色策略来创建服务角色。该角色的名称为 CodeCatalystWorkflowDevelopmentRole-*spaceName*，并且将附加唯一标识符。有关角色和角色策略的更多信息，请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色。有关创建角色的步骤，请参阅为您的账户和空间创建 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。该角色添加到您的账户中，并在 CodeCatalyst 的项目创建页面中可用。

我在 AWS 管理控制台中无法访问 Amazon CodeCatalyst 空间页面。

问题：当我尝试在 AWS 管理控制台中访问 Amazon CodeCatalyst 页面以向我的 CodeCatalyst 空间添加账户或向 AWS 中的账户添加角色时，我收到了权限错误。

可能的修复方法：

对于您的空间，如果您具有空间管理员角色，则可以添加已授权的 AWS 账户，以便将其添加到您的项目中。您还必须拥有一个具有管理权限或可以与 AWS 管理员合作处理的 AWS 账户。您必须首先确保使用您想要管理的同一账户登录到 AWS 管理控制台。登录 AWS 管理控制台后，您可以打开控制台并重试。

在 AWS 管理控制台中打开 Amazon CodeCatalyst 页面，网址为：https://us-west-2.console.aws.amazon.com/codecatalyst/home?region=us-west-2#/。

我想要一个不同的账户作为计费账户

问题：当我设置 CodeCatalyst 登录信息时，我完成了几个步骤来设置我的空间并关联一个授权的 AWS 账户。现在，我想授权另一个账户进行计费。

可能的修复方法：对于您的空间，如果您具有空间管理员角色，则可以授权计费账户。您还必须拥有一个具有管理权限或可以与 AWS 管理员合作处理的 AWS 账户。

有关更多信息，请参阅《Amazon CodeCatalyst Administrator Guide》中的 Managing billing。

我的项目工作流因连接名称错误而失败

问题：创建项目然后运行项目工作流时，工作流失败并显示错误，指出连接名称无效，如下所示：

<action_name> 失败：连接名称无效。

可能的修复方法：确保提供要添加到空间的账户 ID，并确保该账户未启用受项目限制的账户连接。如果该账户启用了受项目限制的账户连接，则您可能需要通过启用对新项目的访问权限来更新账户连接。有关更多信息，请参阅 Configuring project-restricted account connections。

排查 Amazon CodeCatalyst 与 AWS SDK 或 AWS CLI 之间的问题

以下信息有助于您排查在使用 CodeCatalyst 和 AWS CLI 或 AWS SDK 时遇到的常见问题。

主题

在命令行或终端输入 aws codecatalyst 时，我收到一个错误，提示选择无效。

问题：当我尝试将 AWS CLI 与 CodeCatalyst 一起使用时，一个或多个 aws codecatalyst 命令未被识别为有效。

解决方案：导致此问题的最常见原因是您使用的 AWS CLI 版本不包含最新服务和命令的最新更新。更新 AWS CLI 的安装，然后重试。有关更多信息，请参阅 设置为 AWS CLI 与一起使用 CodeCatalyst。

运行 aws codecatalyst 命令时，我收到一个凭证错误

问题：当我尝试将 AWS CLI 与 CodeCatalyst 一起使用时，我收到一条消息，提示 `You can configure credentials by running "aws configure".` 或 `Unable to locate authorization token`。

解决方案：您必须配置 AWS CLI 配置文件才能使用 CodeCatalyst 命令。有关更多信息，请参阅 设置为 AWS CLI 与一起使用 CodeCatalyst。

通过 CodeCatalyst 运行状况报告了解当前服务状态

Amazon CodeCatalyst 运行状况报告是一个公共控制面板，它为用户提供有关 CodeCatalyst 中具有广泛影响的资源性能和服务可用性的最新通知的汇总列表。您可以查看 CodeCatalyst 中的哪些资源存在问题并且可能影响应用程序。这使您能够跟踪系统范围内的中断和其他资源停机时间。在事件发生时，运行状况报告图标上会显示一个蓝色指示器。此外，CodeCatalyst 会自动向项目中具有空间管理员角色的所有用户发送警报和电子邮件通知，并近实时地提供事件的详细信息和历史记录。

此控制面板提供所有活动事件的列表以及过去 30 天内发生的最多 100 个事件的记录。您可以根据事件的更新日期组织事件列表。您也可以刷新事件列表以获取最新更新。

以下是使用 CodeCatalyst 运行状况报告的可能工作流：

Mateo Jackson 是 Budding Space 的一名拥有空间管理员权限的开发人员。在尝试创建拉取请求时，他不断收到一条错误消息。他查看了自己的电子邮件，发现收到了一封来自 CodeCatalyst 的自动生成的系统事件电子邮件，其中提供了有关影响其空间的系统问题的详细历史记录。他选择查看更新，之后被转到 CodeCatalyst 运行状况报告，他可在其中查看所有系统报告的事件。他从列表中选择该事件以了解更多信息。这将打开一个分屏，其中显示了事件的上次更新时间的时间戳、历史记录、受影响的功能、开始时间和当前状态。他还会看到问题仍然存在，但服务团队已开始解决此问题。每当事件的历史记录或状态获得更新时，他都会收到一封电子邮件。如果他无法访问自己的电子邮件，他可以选择顶部面板中的铃铛图标来转至 CodeCatalyst 运行状况报告。

CodeCatalyst 运行状况报告概念

了解以下概念将有助于您了解 CodeCatalyst 运行状况报告，以及它们如何使您能够跟踪应用程序、服务和资源的运行状况。

事件

事件是指影响 CodeCatalyst 中的应用程序和资源的系统事件。您可以选择事件来查看事件的详细历史记录，包括事件的开始时间以及服务团队是否正在努力处理该事件。

状态

状态是指事件的实时状态。它将显示为正在进行或已解决。

受影响的功能

受影响的功能是指受事件影响的资源或应用程序。单个事件可能会影响系统中的多个方面，包括拉取请求、事务、工作流、测试、部署和来源。

更新时间

更新时间提供事件的上次更新时间的时间戳。

支持 for Amazon CodeCatalyst

创建空间时，必须关联一个 AWS 账户，并将其指定为空间的计费账户。您指定作为计费账户的 AWS 账户也是您访问 Amazon CodeCatalyst 的 支持 计划的位置。如果您需要支持，可以通过此指定的 AWS 账户创建支持案例。

空间中的 CodeCatalyst 用户使用 CodeCatalyst 中的 支持 for Amazon CodeCatalyst 页面管理支持案例。您可以升级到 支持 计划（如 Business Support 或 Enterprise Support），以便在 CodeCatalyst 中创建和管理 CodeCatalyst 技术支持案例。对于支持案例，可通过电话、Web 或聊天提供支持。

只有特定于 CodeCatalyst 服务和资源的案例才能通过适用于 Amazon CodeCatalyst 的 支持 获得支持。CodeCatalyst 资源包括在 CodeCatalyst 中部署的资源，还包括 CodeCatalyst 中的用户所部署的资源，但不包括为其他 AWS 服务或第三方服务部署的资源。如果您需要对任何其他 AWS 服务的支持，则必须通过 AWS 管理控制台打开相应服务。

要更改您的支持计划，请参阅更改支持计划。

> **Note**
>
> Developer Support 计划不适用于在生产环境内使用。如果空间计费账户有 Developer Support 计划，该计划不会在 CodeCatalyst 中的 支持 内级联到所有空间管理员和空间成员。

支持 for Amazon CodeCatalyst 计费

在 CodeCatalyst 中创建空间时，空间中的用户可以从 支持 for Amazon CodeCatalyst 创建和管理支持案例。您可以创建两种类型的客户案例：

- 账户和计费支持案例可供空间中的所有 CodeCatalyst 用户使用。根据您在 CodeCatalyst 中的权限，您可以获得有关计费和账户问题的帮助。
- 技术支持案例会将您与技术支持工程师联系起来，帮助您解决与服务相关的技术问题，以及对第三方应用程序的扩展。如果您拥有“基本”支持计划，则无法创建技术支持案例。

  指定为空间计费账户的 AWS 账户必须拥有该空间的 Business Support 或 Enterprise Support 计划，才能使用 支持 for CodeCatalyst 处理技术案例。

> **Note**
>
> 如果您的空间使用的 支持 for Amazon CodeCatalyst 来自没有 Business Support 或 Enterprise Support 计划的账户，您仍然可以使用 支持 for Amazon CodeCatalyst 处理账户和计费案例。

如需技术支持，您必须通过 CodeCatalyst 控制台打开所有案例。您无法从 AWS 管理控制台中的 支持 为 CodeCatalyst 创建技术支持案例。

> **Note**
>
> 支持 for Amazon CodeCatalyst 不提供提高服务限制的请求。这些请求只能由空间计费账户的根用户在 AWS Support Center Console 中提交。

支持 for Amazon CodeCatalyst 与 支持 具有相同的支持协议，但有以下注意事项：

- 支持 中的严重性列表、响应时间和 SLA 适用于 支持 for CodeCatalyst 中的支持案例，详见选择严重性。
- 空间管理员和空间成员不能在 Slack 中使用 支持 API 或 AWS SDK 或 支持 应用程序为 CodeCatalyst 创建案例。CodeCatalyst 支持案例只能从 CodeCatalyst 提交。

> **Note**
>
> CodeCatalyst 未与 AWS Trusted Advisor 或 AWS 事件检测及响应服务完全集成。验证 CodeCatalyst 的集成方式，确保您的业务实践与当前集成保持一致。

您必须是要请求支持的空间中的用户。

> **Note**
>
> 如果您的空间中有多个构建者，建议您购买 Business Support 或 Enterprise Support 计划。这些计划为最多可容纳 5000 个构建者的空间提供技术支持。

指定为空间计费账户的 AWS 账户使用 AWSRoleForCodeCatalystSupport 角色和 AmazonCodeCatalystSupportAccess 托管式策略。这使得空间中的 CodeCatalyst 用户可以访问 支持 for Amazon CodeCatalyst 页面。有关此角色和策略的更多信息，请参阅 AmazonCodeCatalystSupportAccess。有关计费的其他注意事项，请参阅《Amazon CodeCatalyst Administrator Guide》中的 Managing billing。

以下是构建者在 CodeCatalyst 中创建支持案例的可能流程：

Mateo Jackson 是 CodeCatalyst 项目的开发人员。通过 支持 for Amazon CodeCatalyst 注册用于管理计费的 AWS 账户并升级到 Business Support 计划后，空间中的所有构建者都可以创建技术支持案例。Mateo 就其项目中的失败工作流提交了一个技术支持案例。Mateo 使用 支持 for Amazon CodeCatalyst 页面填写表单并创建案例，在请求中提供工作流 ID 和其他详细信息。创建的案例具有案例 ID，其中包括指定为计费账户并与空间支持计划相关联的 AWS 账户的账户 ID。

虽然所有构建者都可以在 支持 for CodeCatalyst 中创建支持案例，但不会对创建的每个案例收取费用。根据您在空间计费账户上购买的 支持 Premium 计划，您几乎可以无限量打开案例和联系人。

> **Note**
>
> 空间计费账户是向您收取 CodeCatalyst 用户和资源费用的 AWS 账户。如果您已部署到其他 AWS 账户，请通过 AWS 管理控制台联系 支持，以获取有关部署到其他服务的资源的帮助。您可以从工作流中识别您部署到的 AWS 账户。

为 支持 for Amazon CodeCatalyst 设置空间

支持 for Amazon CodeCatalyst 管理支持案例，作为 支持 API 与 CodeCatalyst 集成的一部分。

AWSRoleForCodeCatalystSupport 角色是一个服务角色，用于您空间中的支持案例。必须将该角色添加到空间的指定计费账户中。有关创建角色的更多信息，请参阅为您的账户和空间创建 AWSRoleForCodeCatalystSupport 角色。

> **Note**
>
> 对于 2023 年 4 月 20 日之前创建的空间，您必须创建该角色，以便为您的空间提供 CodeCatalyst 支持。如果在 2023 年 4 月 20 日之后创建空间，您可以在创建空间时、在 CodeCatalyst 中的计费详细信息页面上或单击 CodeCatalyst 中的支持横幅链接来创建角色。

为您的空间设置支持

1. 创建 CodeCatalyst 空间时，系统会指示您关联一个计费账户。空间的指定计费账户将由 AWS 计费。有关创建空间的更多信息，请参阅创建新的空间和开发角色（无需邀请即可开始）。
2. 创建 CodeCatalyst 空间时，可选择创建 `AWSRoleForCodeCatalystSupport` 服务角色，允许 CodeCatalyst 用户访问支持服务。该角色使用托管式策略 `AmazonCodeCatalystSupportAccess`。该角色必须添加到指定为空间计费账户的 AWS 账户中。有关创建此角色的更多信息，请参阅 为您的账户和空间创建 AWSRoleForCodeCatalystSupport 角色。
3. 对于空间的指定计费账户，建议空间管理员为 AWS 账户购买 Business Support 或 Enterprise Support 计划。空间中的所有成员都将能够管理来自 支持 for Amazon CodeCatalyst 的支持案例，支持渠道将与您购买的 支持 计划保持一致，其中集成已完成。
4. 要在 CodeCatalyst 中创建和管理支持案例，请参阅在 CodeCatalyst 中创建 CodeCatalyst 支持案例。

在 AWS 管理控制台中访问对 CodeCatalyst 的支持

如果空间的已启用支持的计费账户断开连接，则在 支持 for Amazon CodeCatalyst 中将不再显示与之前空间计费账户和相关支持计划相关联的 支持 案例。该计费账户的根用户可以从 AWS 管理控制台查看和解决旧案例，并可以为 支持 设置 IAM 权限，以便其他用户查看和解决旧案例。您仍然可以通过 AWS 管理控制台享受支持计划为所有其他 AWS 服务带来的好处，并完成之前未解决的任何 CodeCatalyst 支持案例。

有关更多信息，请参阅《支持 User Guide》中的 Updating, resolving, and reopening your case。

有关 CodeCatalyst 一般操作信息的支持案例也可在 AWS 管理控制台中打开，但无法通过此渠道获得有关 CodeCatalyst 的技术支持。有关更多信息，请参阅《支持 User Guide》中的 Creating support cases and case management。

以下是用户在 AWS 管理控制台中解决 CodeCatalyst 支持案例的可能流程：

虽然所有构建者都可以通过 支持 for Amazon CodeCatalyst 创建支持案例，但支持请求将从指定为空间计费账户的账户中计费。Mateo Jackson 是 CodeCatalyst 中的项目开发人员，他就项目中的失败工作流开设了一个技术支持案例。但是，已向 支持 for Amazon CodeCatalyst 注册并购买了 Business Support 计划的空间的计费账户已与空间断开连接。Mateo 查看最新通信并解决待处理的 CodeCatalyst 的案例的唯一方法，是从 AWS 管理控制台中的 支持 Center 管理案例 ID。为此，Mateo 从附加到其支持案例的前一个空间计费账户的根用户处获得 IAM 权限，并通过控制台中的 支持 解决案例。

> **Important**
>
> 如果您更改了空间的指定计费账户，您的 支持 计划在月底之前仍只能通过 AWS 管理控制台访问。您需要在更新后的计费账户上重新购买 支持，以便继续访问 CodeCatalyst 中以前创建的支持案例。我们建议您等到解决了所有支持案例后再更改空间计费账户，以免影响通过 支持 for Amazon CodeCatalyst 访问支持案例。

在 CodeCatalyst 中创建 CodeCatalyst 支持案例

您可以在适用于 Amazon CodeCatalyst 的 支持 页面中创建支持案例。

AWS 构建者 ID 只能获得对经过身份验证的别名的支持，并且只能获得对基于其权限的资源的支持。所有空间管理员和空间成员均可使用账户和计费选项。但是，用户只能获得对他们有权在 CodeCatalyst 中访问的资源的支持，而不能获得与账户计费管理相关的支持。

您可以使用空间的适用于 CodeCatalyst 的 支持 页面，为您的 CodeCatalyst 资源创建账户和计费案例或技术支持案例。

> **Note**
>
> 只有特定于 CodeCatalyst 服务和资源的案例才能通过适用于 Amazon CodeCatalyst 的 支持获得支持。CodeCatalyst 资源包括在 CodeCatalyst 中部署的资源，还包括 CodeCatalyst 中的用户所部署的资源，但不包括为其他 AWS 服务或第三方服务部署的资源。如果您需要对任何其他 AWS 服务的支持，则必须通过 AWS 管理控制台打开相应服务。

在 CodeCatalyst 中创建支持案例

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的 CodeCatalyst 空间。

   > **Tip**
   >
   > 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 在页面顶部，选择 ？图标，然后选择支持。
4. 选择创建案例。

5. 请选择以下选项之一：

   - 账户和计费
   - 技术

   > **Note**
   >
   > 在适用于 Amazon CodeCatalyst 的 支持 中，如果将 Business Support 或 Enterprise Support 计划添加到空间计费账户，则所有空间管理员和空间成员都将获得 CodeCatalyst 技术案例支持。有关问题排查信息，请参阅我无法为自己的空间创建技术支持案例。
   >
   > 支持 计划不跨越空间。如果您是多个空间的成员，则您的空间管理员需要为每个空间购买 支持 Premium 计划，才能在所有空间中获得技术支持。

6. 选择服务、类别和严重性。有关选择严重性的信息，请参阅 Choosing a severity。

   - 一般指导
   - 系统受损
   - 生产系统受损
   - 生产系统停机
   - 业务关键系统停机

7. 选择下一步：其他信息。
8. 在其他信息页面上，对于主题，输入有关您的事务的标题。
9. 对于描述，请按照提示操作以描述您的案例，例如：

   - 特定于 CodeCatalyst 的问题排查信息，例如工作流 ID、日志或屏幕截图
   - 您收到的错误消息
   - 您遵循的问题排查步骤

   > **Note**
   >
   > 不要在案例通信中分享任何敏感信息，例如凭证、信用卡、签名 URL 或个人身份信息。

10. （可选）选择附加文件以向案例添加任何相关文件，例如错误日志或屏幕截图。您最多可以附加三个文件。每个文件最大可为 5 MB。
11. 在空间名称中，将显示您的空间名称。

12. 在构建者名称中，将自动填充与您的 AWS 构建者 ID 关联的全名。
13. （可选）在项目名称（如果适用）中选择项目。

> **Note**
>
> 您只会看到您具有其权限的项目。如果您需要访问其他项目，请在创建支持案例之前，请求您的项目管理员为您提供访问权限。

14. 选择下一步：联系我们。
15. 在首选联系语言中，选择默认语言。目前只有英语可用。
16. 选择 Web、电话或聊天选项作为联系方式。
17. 查看您的案例详细信息，然后选择提交。此时将显示您的案例 ID 号和摘要。

    支持案例是在空间级别创建的，并且可供所有有权访问您的支持案例中定义的空间和项目（如果已选定）的成员查看。目前无法省略单个用户的支持案例。

在 CodeCatalyst 中解决支持案例

您可以从 支持 for Amazon CodeCatalyst 页面解决待处理的支持案例。

在要在其中解决支持案例的空间中，您必须具有空间管理员或空间成员角色。如果您没有空间管理员角色，或者已在创建案例时选择项目，则您还需要拥有项目的成员资格才能查看和解决案例。

在 CodeCatalyst 中解决待处理的支持案例

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的 CodeCatalyst 空间。

> **Tip**
>
> 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 在页面顶部，选择 ? 图标，然后选择 支持 for Amazon CodeCatalyst。
4. 选择要管理的支持案例的链接。选择解决案例。

在 CodeCatalyst 中重新打开支持案例

您可以从 支持 for Amazon CodeCatalyst 页面重新打开已解决的支持案例。

> **Note**
>
> 从问题得到解决后的 14 天内，您可以重新打开支持案例。但是，您不能重新打开已处于非活动状态超过 14 天的案例。如果您无法重新打开案例，请打开一个新案例并包含之前的案例 ID 作为参考。
> 如果您重新打开具有与当前问题不同的信息的现有案例，则客服可能会要求您创建新案例。

在 CodeCatalyst 中打开支持案例

1. 通过访问 https://codecatalyst.aws/ 打开 CodeCatalyst 控制台。
2. 导航到您的 CodeCatalyst 空间。

   > **Tip**
   >
   > 如果您属于多个空间，则可以在顶部导航栏中选择一个空间。

3. 在页面顶部，选择 ? 图标，然后选择 AWS Support for CodeCatalyst。
4. 选择要管理的支持案例的链接。选择重新打开。在确认屏幕上选择确定，然后选择提交。
5. 在描述中填入有关同一事务的最新信息。不要在案例通信中分享任何敏感信息，例如凭证、信用卡、签名 URL 或个人身份信息。

CodeCatalyst 的配额

下表描述 Amazon CodeCatalyst 的配额和限制。您可以在以下主题中找到 CodeCatalyst 的特定方面的其他信息：

- 中的源存储库配额 CodeCatalyst
- 中的身份、权限和访问配额 CodeCatalyst
- 中的工作流程配额 CodeCatalyst
- 中开发环境的配额 CodeCatalyst
- 项目的配额
- 中的蓝图配额 CodeCatalyst
- 空间配额
- 中的问题配额 CodeCatalyst

一个账户中的最大空间数	5
用户在一个日历月内可创建的最大空间数	5
一个空间的最小 AWS 账户数	1
一个空间的最大账户连接数	5000
作为空间的计费账户的 AWS 账户的最大数量	1
免费套餐中可将单个 AWS 账户指定为其计费账户的空间的最大数量	3
付费套餐中可将单个 AWS 账户指定为其计费账户的空间的最大数量	15
一个空间的最大 VPC 连接数	100
一个空间的最大项目数	100
一个用户可以属于的项目的最大数量	1000

空间描述	空间描述是可选的。如果指定，长度必须在 0 到 200 个字符之间。它们可以包含字母、数字、空格、句点、下划线、逗号、短划线和以下特殊字符的任意组合： `? & $ % + = / \ ; : \n \t \r`
空间名称	空间名称在 CodeCatalyst 中必须是唯一的。不能重复使用已删除空间的名称。 空间名称长度必须在 3 至 63 个字符之间。它们还必须以字母数字字符开头。空间名称可包含字母、数字、句点、下划线和短划线的任意组合。它们不能包含以下任何字符： `! ? @ # $ % ^ & * ( ) + = { } [ ] | /\ > < ~ ` ' " ; :`

Amazon 的文档历史记录 CodeCatalyst

下表描述了的文档历史记录和整个文档的更新 CodeCatalyst。

变更	说明	日期
亚马逊 CodeCatalyst 不再向新买家开放	亚马逊 CodeCatalyst 不再向新买家开放。有关更多信息，请参阅如何从 Amazon 迁移 CodeCatalyst。	2025 年 11 月 7 日
新增内容：新的通用 devfile 映像	添加了有关新的通用 devfile 映像（`Universal image 5.0`）的文档。	2025 年 7 月 30 日
更新内容：数据隐私	更新了文档以包含有关与 IAM Identity Center 相关的数据隐私的其他信息。请参阅中的数据隐私 CodeCatalyst	2025 年 7 月 28 日
更新内容： CodeCatalyst 蓝图弃用	已从 “可用蓝图” 表中移除 AWS 项目开发套件 (AWS PDK) 内容。	2024 年 12 月 6 日
新增内容：新的通用 devfile 映像	添加了有关新的通用 devfile 映像（`Universal image 4.0`）的文档。	2024 年 8 月 26 日
新增内容：“AWS CDK 部署” 操作使用的运行时镜像、“AWS CDK bootstrap” 操作使用的运行时镜像	添加了两个主题：“AWS CDK 部署” 操作使用的运行时镜像和“AWS CDK bootstrap” 操作使用的运行时镜像。	2024 年 8 月 20 日
新增内容：弃用的通用 devfile 映像	添加了有关弃用的通用 devfile 映像（`Universal image 1.0` 和 `Universal image 2.0`）的文档。	2024 年 8 月 16 日

更新内容： CodeCatalyst 蓝图弃用	已从 “可用蓝图” 表中移除无服务器应用程序模型 (SAM) 和 Video-on-demand Web 服务内容。	2024 年 8 月 14 日
更新的内容：活动映像	更新了活动映像主题以指出某些操作可能使用 2022 年 11 月版或 2024 年 3 月版映像。还更新了每个操作的文档以指出使用的是哪个映像。	2024 年 8 月 9 日
更新的内容：排查用户名被截断的问题	更新了问题排查主题，以包含有关排查某些情况下用户名被截断的问题的信息。	2024 年 7 月 31 日
更新的内容：创建环境	更新了该创建环境部分以提及 CodeCatalyst 角色。	2024 年 7 月 25 日
更新的内容：在操作之间共享构件和文件	更新了在构件中引用文件子主题以包含有关操作组的信息。还添加了示例：存在操作组时引用构件中的文件子主题。	2024 年 7 月 18 日
更新的内容：使用工作流进行构建、测试和部署	更新了文档以指出工作流定义的 `.yml` 或 `.yaml` 文件扩展名必须为小写。	2024 年 7 月 15 日
更新内容：增加了针对 SSO 用户在空间中创建开发环境时的疑难解答 CodeCatalyst	添加了问题排查主题，以包含有关在创建开发环境时，使用身份联合验证登录空间的 SSO 用户遇到的问题的信息。	2024 年 7 月 12 日
更新的内容：创建工作流	更新了主题以指出工作流定义文件可存储在 `~/.codecatalyst/workflows/` 的子目录中。	2024 年 7 月 12 日

更新的内容：工作流程状态为 CodeCatalyst	添加了有关如何在 CodeCatalyst控制台中隐藏非活动工作流程的说明。	2024 年 7 月 12 日
新增内容：配置仅限手动触发的触发器	添加了配置仅限手动触发的触发器主题。	2024 年 6 月 26 日
更新的内容：创建环境	更新了创建环境部分和其他几个部分以反映一项新功能，该功能可让您在创建 CI/CD 环境时指定默认 IAM 角色。此角色将分配给已与环境关联的所有工作流操作。您不再需要直接将 IAM 角色分配给操作。	2024 年 6 月 21 日
新增内容：教程：从程序包存储库中拉取	添加了介绍如何配置工作流程以从 CodeCatalyst 包存储库提取依赖项的教程。	2024 年 6 月 20 日
更新内容：使用带有蓝图的 GitLab 项目存储库	更新了文档，使其能够在创建自定义蓝图时使用蓝图创建项目创建 GitLab 项目存储库或创建自定义蓝图。在项目中添加蓝图以整合资源	2024 年 6 月 19 日
更新的内容：教程：使用生成式人工智能功能	更新了教程，以指出可在创建项目或向现有项目添加蓝图时使用 Amazon Q 选择蓝图。	2024 年 6 月 18 日
更新的内容：创建源存储库	更新了文档以包含有关创建空存储库的信息。	2024 年 6 月 18 日

新内容： CodeCatalyst 使用 Amazon Q 创建项目	在创建项目中添加了标题为使用 Amazon Q 通过蓝图创建项目或添加功能时的最佳实践和将蓝图与项目结合使用的最佳实践的新部分，其中包含有关使用 Amazon Q 创建项目和添加蓝图的说明。	2024 年 6 月 18 日
新增内容：将现有 Git 存储库克隆到源存储库中	在中添加了一个关于如何使用 Git 将镜像克隆或本地存储库推送到空源存储库的新主题。CodeCatalyst	2024 年 6 月 18 日
新内容：支持 Maven NuGet、和 Python 包类型	在中添加了有关使用 Maven NuGet、和 Python 包的 CodeCatalyst文档。	2024 年 6 月 17 日
空间和账户连接的更新	现在，您可以将 AWS 账户 指定为结算账号的账户连接到多个 CodeCatalyst 空间。对于为身份联合验证设置的空间，可以将一个 Identity Center 实例连接到多个空间。请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户、排查与您的空间关联的账户问题和 CodeCatalyst 的配额。	2024 年 6 月 13 日
更新的内容：使用角色	更新了角色文档，以包含使用 Amazon Q 针对事务建议和创建任务的权限。	2024 年 6 月 13 日
更新的内容：使用角色	更新了角色文档，以包含在事务之间创建、更新和移除链接的权限。	2024 年 6 月 13 日

更新的内容：教程：使用生成式人工智能功能	更新了教程，以包含有关使用 Amazon Q 针对事务建议任务的信息。	2024 年 6 月 13 日
更新的内容：管理有关事务的任务	添加了有关使用 Amazon Q 为问题推荐任务的信息 CodeCatalyst。	2024 年 6 月 13 日
新增内容：将一个事务链接到另一个事务	在中添加了一个关于将议题与其他议题关联的新主题 CodeCatalyst。	2024 年 6 月 13 日
更新的内容：“AWS CDK 部署”操作 YAML	修复了 `Region` 属性的描述。	2024 年 6 月 12 日
更新的内容：使用带蓝图的第三方存储库	更新了文档，使其能够在创建自定义蓝图时使用蓝图创建项目创建 GitHub 存储库或创建自定义蓝图。在项目中添加蓝图以整合资源	2024 年 6 月 6 日
新增内容：添加了有关使用个人连接的步骤	添加了创建和删除个人连接的步骤。个人关系允许您在 Amazon CodeCatalyst 中管理项目和蓝图的 GitHub 资源。	2024 年 6 月 6 日
新增内容：Bitbucket 存储库扩展	在中添加了有关使用 Bitbucket 存储库扩展的新内容。 CodeCatalyst	2024 年 6 月 5 日
新增内容：操作类型	更新了第三方操作主题以提及 “SonarCloud 扫描” 操作。	2024 年 5 月 29 日
更新的内容：“AWS CDK 部署”操作 YAML	修复了 `CdkRootRootPath` 示例。	2024 年 5 月 28 日

更新了内容	更新了主题标题并重新组织了内容，以提高可读性和可发现性。如果您想提供关于这些更改的反馈，请使用此提供反馈链接。	2024 年 5 月 17 日
新增内容：查看文件更改历史记录	更新了文档，以反映用于查看源存储库中的文件更改历史记录的新功能。	2024 年 5 月 1 日
更新的内容：教程：使用生成式人工智能功能	更新了教程，以反映与 Amazon Q 开发者版的集成。	2024 年 4 月 29 日
更新的内容：教程：使用生成式人工智能功能	更新了教程，以反映让 Amazon Q 能够分析事务的复杂性、建议和创建任务以及处理事务中的任务。	2024 年 4 月 22 日
新增内容：用阶段门控制工作流运行	添加了用阶段门控制工作流运行、要求批准工作流运行和其他几个与工作流审批相关的主题。	2024 年 4 月 22 日
新增内容：使用任务将事务分成多个较小目标	添加了内容以支持启动事务中的任务。可以向事务添加任务，以进一步细分、组织和跟踪该事务的处理情况。	2024 年 4 月 4 日
更新的内容：在操作之间共享构件和文件	更新了在操作之间共享构件和文件主题以包含两个新的子主题：我能否在不将构件指定为输出和输入的情况下共享它们？和我能否在工作流之间共享构件？	2024 年 4 月 2 日

更新的内容： GitHub 操作的局限性 CodeCatalyst	更新了 GitHub 操作的局限性 CodeCatalyst主题以表明GitHub操作在较旧的运行时环境 Docker 镜像上运行。	2024 年 4 月 2 日
新增内容：“AWS CDK 部署”操作 YAML	向“AWS CDK 部署”操作 YAML 添加了一个新的 `CloudAssemblyRootPath` 属性。	2024 年 4 月 1 日
更新的内容：指定运行时环境映像	更新了指定运行时环境映像主题以包含有关新的 2024 年 3 月版运行时环境映像的信息。	2024 年 3 月 26 日
更新的内容：使用角色	将角色权限信息整合到单个表中。该表位于新的查看每个角色可用的权限主题中。	2024 年 3 月 18 日
新增内容：查看用户的所有空间和项目	添加了有关在用户主页上查看列表的信息，该列表显示了登录用户的每个 CodeCatalyst 空间或项目。 CodeCatalyst请参阅查看用户的所有空间和项目。	2024 年 3 月 11 日
新增内容：示例：带有拉取和分支的触发器	添加了拉取请求触发器的示例。在整个使用触发器自动启动工作流运行主题中进行了少量纠正。	2024 年 3 月 11 日
更新的内容：使用角色	更新了角色文档，以包含用于创建、删除和查看环境的权限。	2024 年 3 月 4 日
更新的内容：教程：使用生成式人工智能功能	更新了教程，以反映在创建事务并将其分配给 Amazon Q 时所做的更改。	2024 年 3 月 4 日

新增内容：事务组件	添加了新内容，说明如何以自定义蓝图开发人员的身份使用事务组件。	2024 年 2 月 27 日
更新的内容：操作类型	更新了CodeCatalyst 实验室行动主题，增加了 CodeCatalyst 实验室操作列表。	2024 年 2 月 21 日
更新的内容：处理拉取请求	更新了文档以反映新功能，包括审批规则和覆盖要求以合并拉取请求。	2024 年 2 月 15 日
更新的内容：合并拉取请求	添加了拉取请求的文档，以包含有关覆盖合并要求以合并尚未获得所需审阅者的批准或符合审批规则的拉取请求的信息。	2024 年 2 月 15 日
新增内容：管理审批规则	添加了拉取请求的文档，以包含有关创建和管理审批规则的信息。	2024 年 2 月 15 日
更新的内容：使用角色	更新了角色的文档，以包含处理审批规则和拉取请求的权限。	2024 年 2 月 14 日
更新的内容：如何修复“工作流定义有 *n* 个错误”错误？	更新了如何修复“工作流定义有 *n* 个错误”错误？部分以包含更多问题排查提示。	2024 年 2 月 9 日
新增内容：查看工作流的状态	添加了一个介绍工作流状态的部分。	2024 年 2 月 9 日
新增内容：查看工作流的状态	添加了一个介绍工作流状态的部分。	2024 年 2 月 9 日

更新内容：中的工作流程配额 CodeCatalyst	更新了 中的工作流程配额 CodeCatalyst主题的每个工作流的最大操作数和每个空间的与一个 AWS 账户 关联的最大环境数配额。	2024 年 2 月 7 日
更新的内容：创建环境	更新了创建环境部分以指明每个环境最多只能使用一个账户连接。	2024 年 1 月 31 日
新内容：自定义蓝图存储库 GitHub	为已公开的 GitHub 存储库添加了新内容。	2024 年 1 月 10 日
更新内容：使用 npm 配置 CodeCatalyst	更新了使用带有 npm 的常规配置说明 CodeCatalyst，并增加了`always-auth=true` 选项的清晰度。	2024 年 1 月 5 日
更新的内容：处理拉取请求	更新了文档，以反映中包含生成式 AI 功能的新功能 CodeCatalyst。	2023 年 11 月 28 日
更新的内容：创建事务	更新了文档，以反映中包含生成式 AI 功能的新功能 CodeCatalyst。	2023 年 11 月 28 日
新增内容：教程：使用生成式人工智能功能	添加了在 Amazon 中使用生成人工智能功能的教程 CodeCatalyst。	2023 年 11 月 28 日
新增内容：自定义蓝图和生命周期管理	添加了有关在 Amazon 中使用自定义蓝图和生命周期管理功能的新内容 CodeCatalyst。	2023 年 11 月 27 日
更新的内容：教程：使用现代三层 Web 应用程序蓝图创建项目	使用修复程序和问题排查信息更新了教程。	2023 年 11 月 22 日

更新的内容：使用触发器自动启动工作流运行	修复了几个与拉取请求触发器相关的示例和描述。添加了触发器和分支的使用准则部分。	2023 年 11 月 22 日
新增内容：使用 SSO 登录	添加了有关使用单点登录 (SSO) 登录的信息以及有关设置和管理支持身份联合的 CodeCatalyst 空间的信息的链接。请参阅 设置并登录 CodeCatalyst 和 使用 SSO 登录。	2023 年 11 月 17 日
更新的内容：使用角色	更新了角色的文档，以包含使用团队、VPC 连接、单点登录和机器资源的权限。	2023 年 11 月 16 日
更新的内容：处理拉取请求	更新了文档，以反映拉取请求更改的显示方式的变化。	2023 年 11 月 16 日
更新内容：配额 CodeCatalyst	使用一个空间的最大 VPC 连接数配额更新了 CodeCatalyst 的配额主题。	2023 年 11 月 16 日
的新迁移主题 CodeCatalyst	新的迁移主题。	2023 年 11 月 16 日
新内容：管理空间和 CodeCatalyst 项目的团队	添加了有关将团队用于空间的信息。请参阅 允许使用团队访问空间 和 允许使用团队访问项目。	2023 年 11 月 16 日
新增内容：管理空间中的蓝图和工作流的机器资源	添加了有关将机器资源用于空间的信息。请参阅允许机器资源的空间访问权限 。	2023 年 11 月 16 日
新内容：管理项目中蓝图和工作流程的 CodeCatalyst 计算机资源	添加了有关在 CodeCatalyst 项目中使用计算机资源的信息。请参阅允许机器资源的项目访问权限。	2023 年 11 月 16 日

新增内容：将 VPC 连接与环境关联	添加了有关将 VPC 连接与工作流中可使用的环境关联的文档。	2023 年 11 月 16 日
新增内容：将 VPC 连接关联到开发环境	添加了说明如何使用具有 VPC 连接的开发环境的文档。	2023 年 11 月 16 日
新增内容	《亚马逊 CodeCatalyst 管理员指南》的首次发布。	2023 年 11 月 16 日
新增内容：“AWS CDK 部署”操作 YAML	向 “AWS CDK 部署”操作 YAML 和 “AWS CDK 引导”操作 YAML 添加了一个新的 `CdkCliVersion` 属性。	2023 年 11 月 14 日
更新的内容：使用角色	更新了角色的文档，以包含处理分支规则的权限。	2023 年 11 月 13 日
更新的内容：排查源存储库、工作流和开发环境的问题	更新了问题排查主题以包含有关使用分支规则的信息。	2023 年 11 月 13 日
更新的内容：构建和测试操作 YAML	更新了 `Environment` 属性的文档。现在，它已成为构建和测试操作的可选字段。	2023 年 11 月 13 日
新增内容：管理分支规则	添加了分支的文档，以包含有关查看源存储库中分支的任何规则以及创建和管理分支规则的信息。	2023 年 11 月 13 日
更新的内容：处理拉取请求	更新了文档，以反映拉取请求的相关信息的显示方式的变化。	2023 年 11 月 10 日
更新的内容：在工作流运行之间缓存文件	更新了文档以包含文件缓存限制。	2023 年 11 月 10 日
更新的内容：教程：将应用程序部署到 Amazon EKS	更新了文档以提及 EKS 应用程序部署蓝图。	2023 年 11 月 9 日

新内容：包含 CodeCatalyst	添加了有关在中使用软件包的文档 CodeCatalyst。	2023 年 11 月 1 日
新增内容和更新的内容：使用角色	更新了四个新角色的文档 CodeCatalyst：高级用户、受限访问权限、审阅者和只读。	2023 年 11 月 1 日
更新的内容：导出 GitHub 输出参数	更新了示例以使用 `GITHUB_OUTPUT` 环境文件而不是 `set-output` 命令。建议使用环境文件来设置输出参数。 GitHub	2023 年 10 月 24 日
新增内容：使用触发器自动启动工作流运行	添加了有关计划触发器的文档。	2023 年 10 月 16 日
更新的内容：“部署到 Kubernetes 集群”操作 YAML	在“部署到 Kubernetes 集群”操作 YAML 和教程：将应用程序部署到 Amazon EKS 主题中添加了有关使用 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色的信息。	2023 年 9 月 22 日
更新的内容：CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色的新角色名称和策略	更新了将开发人员角色名称更改为 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 的步骤和角色描述。开发者角色现在使用 `AdministratorAccess` AWS 托管策略。请参阅了解 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色 和 创建新的空间和开发角色（无需邀请即可开始）。	2023 年 9 月 20 日

更新的内容：在工作流中使用变量	引入了两个新概念：用户定义的变量和预定义变量。这两个概念可使在工作流中使用变量部分更易阅读和理解。	2023 年 9 月 19 日
更新的内容：使用提交	更新了文档，以反映显示的信息的变化，并提供了有关查看具有多个父项的提交的详细信息。	2023 年 9 月 7 日
新增内容：使用触发器自动启动工作流运行	在使用触发器自动启动工作流运行主题中添加了以下示例：示例：一个简单的“push to main”触发器	2023 年 9 月 6 日
更新的内容：处理拉取请求	更新了文档，以反映创建拉取请求时源分支和目标分支的显示顺序的变化。	2023 年 8 月 30 日
新增内容：查看和更改默认分支	添加了分支的文档，以包含有关查看和更改源存储库的默认分支的信息。	2023 年 8 月 30 日
更新的内容：“部署到 Kubernetes 集群”操作 YAML	在“部署到 Kubernetes 集群”操作 YAML 中的 Manifests 属性描述中添加了有关 Helm 和 Kustomize 的说明。	2023 年 8 月 15 日
新增内容：管理事务附件	添加了有关使用和管理事务附件的文档。	2023 年 8 月 15 日
更新的内容：使用触发器自动启动工作流运行	改进并扩展了与工作流触发器相关的文档。	2023 年 8 月 11 日

新增内容：排查角色权限的问题	添加了有关更新角色权限以运行需要访问 Amazon 的工作流程的信息 CodeGuru。请参阅现代三层 Web 应用程序蓝图工作流程OnPullRequest失败，出现 Amazon 权限错误 CodeGuru。	2023 年 8 月 11 日
新增内容：如何修复“工作流非活动”消息？	添加了以下问题排查主题：如何修复“工作流非活动”消息？	2023 年 8 月 11 日
新增内容：使用工作流部署到 Amazon EKS	添加了有关部署到 Kubernetes 集群操作的文档。有关更多信息，请参阅使用工作流部署到 Amazon EKS和教程：将应用程序部署到 Amazon EKS。	2023 年 7 月 27 日
更新了如何记录 CodeCatalyst 空间的管理事件	添加了有关如何记录 CodeCatalyst空间中特定操作的管理事件的信息 AWS CloudTrail。添加了有关如何使用该 list-event-logs命令查看空间中所有事件的信息。请参阅使用日志记录监控事件和 API 调用。	2023 年 7 月 20 日
更新的内容：使用触发器自动启动工作流运行	更新了文档，表明 GitHub 源存储库现在支持拉取请求触发器。以前，只有 CodeCatalyst 源存储库支持拉取请求触发器。	2023 年 7 月 14 日
更新的内容：中的工作流程配额 CodeCatalyst	更新了 中的工作流程配额 CodeCatalyst主题的操作可以运行的最长时间配额。	2023 年 6 月 27 日
更新的内容：工作流 YAML 定义	修复了 `Compute` 代码块中的格式错误。	2023 年 6 月 27 日

更新了内容：数据保护	更新了文档以包含有关数据复制的其他信息。	2023 年 6 月 26 日
新增内容：指定要使用的操作版本	添加了指定要使用的操作版本主题。	2023 年 6 月 21 日
更新的内容：部署到 AWS 账户 和 VPCs	阐明了哪些操作支持在中显示其部署信息 CodeCatalyst？部分。	2023 年 6 月 14 日
更新的内容：重新整理了事务文档	重新整理了事务文档的大部分内容，以更好地与整个文档集和用户流程保持一致。	2023 年 5 月 31 日
更新的内容：事务视图切换器	更新了各种用户流程，使其与更新的事务视图切换器保持一致。	2023 年 5 月 31 日
更新的内容：管理通知	更新了通知的文档，以包含有关配置个人 Slack 通知的信息。	2023 年 5 月 30 日
更新的内容：管理通知	更新了通知的文档，以包含有关配置个人 Slack 通知的信息。	2023 年 5 月 30 日
新内容： CodeCatalyst 信任模型	添加了一个新主题，其中包含有关信任模型的信息，该模型 CodeCatalyst 允许在连接中扮演服务角色 AWS 账户。添加了有关为定义的服务主体的新章节。 CodeCatalyst请参阅了解 CodeCatalyst 信任模型。	2023 年 5 月 20 日
更新的内容：将源存储库连接到工作流	简化了引用源存储库文件中的说明。	2023 年 5 月 10 日

更新的内容：中的工作流程配额 CodeCatalyst	更新了 中的工作流程配额 CodeCatalyst主题的输出变量值的最大长度配额。	2023 年 5 月 10 日
新增内容：在操作之间共享构件和文件	添加了两个示例：示例：在单个构件中引用文件和示例：存在 WorkflowSource 时引用构件中的文件。	2023 年 5 月 10 日
更新的内容：处理拉取请求	更新了拉取请求的文档，以包含有关为拉取请求事件配置电子邮件首选项的信息。	2023 年 4 月 21 日
更新的内容：管理通知	更新了通知的文档，以包含有关为拉取请求事件配置电子邮件首选项的信息。	2023 年 4 月 21 日
托管式策略的更新	添加了 AWS 托管策略：AmazonCodeCatalystFullAccess、AWS 托管策略：AmazonCodeCatalystReadOnlyAccess 和 AWS 托管策略：AmazonCodeCatalystSupportAccess 托管式策略。请参阅CodeCatalyst AWS 托管策略的更新。	2023 年 4 月 20 日
新增内容：移除部署目标	添加了移除部署目标主题。	2023 年 4 月 20 日
新增内容：操作类型	添加了CodeCatalyst 行动主题。	2023 年 4 月 20 日
有关管理空间中具有空间管理员角色的用户的更新	添加了有关在空间中移除或更改具有空间管理员角色的用户的角色的信息。请参阅移除或更改具有空间管理员角色的用户的角色。	2023 年 4 月 19 日

有关管理开发环境的更新	添加了有关以空间管理员身份管理开发环境的信息。请参阅管理空间的开发环境。	2023 年 4 月 19 日
更新的内容：查找和查看事务	重新整理了查找和查看事务主题和子主题。	2023 年 4 月 19 日
更新内容： CodeCatalyst 使用链接的项目存储库在中创建 GitLab 项目	更新创建项目为在使用链接的第三方存储库创建项目章节中加入了 GitLab 集成。	2023 年 4 月 19 日
更新的内容：配置计算和运行时映像	添加了对 Amazon Linux 2 上的 Arm64 架构的支持。	2023 年 4 月 19 日
新增内容：在组内移动事务	在面板和所有事务视图中添加了有关在组内移动事务的文档。	2023 年 4 月 19 日
更新的内容：中的工作流程配额 CodeCatalyst	更新了 中的工作流程配额 CodeCatalyst主题的缺少配额信息，并将单个操作的输出变量的最大总大小配额更新为 120 KB（从 2 KB 开始）。	2023 年 4 月 18 日
新增内容：查看操作的源代码	添加了查看操作的源代码主题。	2023 年 4 月 18 日
新增内容：停止工作流运行	添加了停止工作流运行主题。	2023 年 4 月 10 日
新内容：添加了用于标记资源以用于与 Amazon AWS 之间的账户关联的部分 CodeCatalyst	添加了有关标记账户连接资源和管理连接资源的 IAM 策略的信息。请参阅 使用标签控制对账户连接资源的访问 和 CodeCatalyst 权限参考。	2023 年 4 月 6 日
新增内容：操作类型	添加了操作类型主题。	2023 年 4 月 6 日

更新的内容：'部署 CloudFormation 堆栈'动作 YAML	更新了 `parameter-overrides` 属性描述。它现在支持 JSON 文件。	2023 年 4 月 5 日
新内容： CodeCatalyst 使用链接 GitHub 存储库创建项目	在中添加了一个创建项目标题使用链接的第三方存储库创建项目为创建链接到存储 GitHub 库的项目的说明的新部分。	2023 年 4 月 5 日
更新的内容：使用通知	更新了通知的文档，以包含有关为项目事件配置电子邮件的信息。	2023 年 3 月 31 日
新增内容	Amazon CodeCatalyst 行动开发套件指南的首次发布。	2023 年 3 月 31 日
更新内容：重组了 Amazon 中的 “空间” 部分 CodeCatalyst	通过移除登录页面并合并主题来更新了“空间”部分。	2023 年 3 月 29 日
更新的内容：教程：将应用程序部署到 Amazon ECS	已更改步骤 1：设置 AWS 用户和 AWS CloudShell为描述如何在中创建用户， AWS IAM Identity Center 而不是 AWS Identity and Access Management。不再建议创建 IAM 用户。	2023 年 3 月 23 日
更新的内容：使用角色	更新了空间管理员、项目管理员和贡献者角色的文档，以包含将事务关联到拉取请求的权限。	2023 年 3 月 13 日
更新的内容：处理拉取请求	更新了拉取请求的文档，以包含有关将事务关联到拉取请求的信息。	2023 年 3 月 13 日

更新的内容：处理事务	更新了事务的文档，以包含有关将事务关联到拉取请求的信息。	2023 年 3 月 13 日
新增内容：查看项目中所有运行的状态和详细信息	添加了一个介绍新的聚合工作流运行页面的部分。	2023 年 3 月 8 日
新增内容：如何修复“工作流定义有 *n* 个错误”错误？	添加了一个说明如何纠正“工作流定义出错”错误的部分。	2023 年 3 月 7 日
更新的内容：创建工作流	更新了说明以反映新 UI。	2023 年 3 月 3 日
新增内容：与集成 universal-test-runner	添加了与集成 universal-test-runner主题。	2023 年 3 月 3 日
更新的内容：使用工作流进行构建、测试和部署	更新了多个部分，以反映工作流摘要页面上的新源存储库、分支和工作流名称筛选条件。	2023 年 3 月 2 日
新增内容：通过提交跟踪部署状态	添加了一个有关通过提交查看代码质量和部署状态的部分。	2023 年 2 月 27 日
新增内容：BranchName'和' CommitId '变量	添加了一个新的 `BranchName` 预定义变量。	2023 年 2 月 16 日
更新内容：在 Amazon 中管理空间成员 CodeCatalyst	更新了有关根据用户在中分配的角色在两个新表格中更改成员角色、邀请成员和移除成员的信息 CodeCatalyst。	2023 年 2 月 15 日
更新内容：增加了在 Amazon CodeCatalyst 控制台中管理 PAT 的步骤	增加了在控制台中查看、创建和删除 PATs 的步骤。	2023 年 2 月 15 日
更新的内容：指定运行时环境映像	在默认映像工具版本表中添加了更多工具。	2023 年 1 月 10 日
更新的内容：在操作之间共享构件和文件	修复了构件路径。	2023 年 1 月 3 日

更新的内容：'GitHub Actions'动作 YAML	修复了 Steps 部分中的代码片段。	2023 年 1 月 3 日
更新的内容：将源存储库连接到工作流	修复了源路径。	2023 年 1 月 3 日
更新的内容：更新拉取请求	更新了文档，以包含有关更新拉取请求的必需的或可选的审阅者的信息。	2022 年 12 月 23 日
新增内容：在工作流运行之间缓存文件	添加了一个有关在工作流中缓存文件的页面。	2022 年 12 月 20 日
更新的内容：处理拉取请求	更新了拉取请求的文档，以包含有关通知的信息。	2022 年 12 月 16 日
新增内容：“AWS CDK 部署”操作 YAML	添加了一个新的 CdkRootPath 属性。	2022 年 12 月 16 日
新增内容：跨操作共享计算	添加了跨操作共享计算主题。	2022 年 12 月 14 日
更新的内容：在操作之间共享构件和文件	修复了说明如何指定输入构件的示例。	2022 年 12 月 13 日
新增内容：'GitHub Actions'动作 YAML	为 “GitHub 操作” 操作添加了专门的参考页面。	2022 年 12 月 13 日
更新内容：项目配额 CodeCatalyst	更新了文档内容：一个空间中最多包含 100 个项目。	2022 年 12 月 2 日
新增内容	《亚马逊 CodeCatalyst 用户指南》的首次发布。	2022 年 12 月 1 日
新增内容：排查扩展问题	添加了一个问题排查主题，介绍如何排查用户在使用第三方扩展功能时可能遇到的问题。	2022 年 6 月 18 日

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。