用户指南

Amazon CodeCatalyst

Amazon CodeCatalyst: 用户指南

Table of Contents

什么是亚马逊 CodeCatalyst？

Amazon CodeCatalyst 是一项综合服务，适用于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 将你需要的工具全部放在一个地方。您可以使用持续集成/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。您还可以通过将您的空间连接到您的 CodeCatalyst 空间来将AWS资源与您的AWS 账户项目集成。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。

在中 CodeCatalyst，您可以创建一个空间来代表您的公司、部门或小组，然后创建包含支持您的开发团队和任务所需的资源的项目。CodeCatalyst资源结构在位于空间内的项目中。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。当您根据项目蓝图创建项目时，该项目会附带资源，例如包含示例代码的源存储库、生成脚本、部署操作、虚拟服务器或无服务器资源等。

我能用什么做 CodeCatalyst呢？

您和您的开发团队可以使用 CodeCatalyst 来执行软件开发的各个方面，从规划工作到部署应用程序。您可以使用 CodeCatalyst 来：

- 对代码进行迭代和协作 — 在源代码存储库中使用分支、合并、拉取请求和注释与您的团队协作处理代码。创建开发环境以快速处理代码，无需克隆或建立与存储库的连接。
- 使用工作流程构建、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程，以处理应用程序的持续集成和交付。您可以手动启动工作流程，也可以将其配置为根据事件（例如代码推送或创建或关闭拉取请求）自动启动。
- 通过问题跟踪确定团队工作的优先顺序 — 使用议题创建待办事项并通过看板监控正在进行的任务的状态。创建和维护一个健康的待办事项供您的团队处理是开发软件的重要组成部分。
- 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。

我该如何开始 CodeCatalyst？

如果您没有空间，或者想学习如何设置和管理空间，我们建议您开始阅读《Amazon CodeCatalyst 管理员指南》。

如果您不熟悉在项目或空间中工作，我们建议您通过以下方式开始工作：

- 回顾 CodeCatalyst 概念
- 创建空间

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

了解更多关于 CodeCatalyst

您可以在本用户指南以及以下资源 CodeCatalyst 中详细了解的功能：

- AWS DevOps 关于亚马逊的博客文章 CodeCatalyst
- 亚马逊 CodeCatalyst API 参考指南
- Amaz CodeCatalyst on Action 开发套件开发者指南
- CodeCatalyst 常见问题
- 感言

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。这些概念包括源代码控制、持续集成和持续交付 (CI/CD) 以及建模和配置自动发布流程中使用的术语。

有关其他概念信息，请参阅以下主题：

- 源存储库概念
- 工作流程概念

主题

- AWS 中的建筑商 ID 空间 CodeCatalyst
- 支持身份联合的空间 CodeCatalyst
- 项目
- 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
- 问题
- 个人访问令牌 (PAT)
- 人际关系
- 角色

AWS 中的建筑商 ID 空间 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

已添加到 IAM Identity Center 实例的 SSO 用户和群组并在身份存储中管理并通过 IAM Identity Center 被邀请进入您的空间的用户。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 登录门户登录。支持身份联合的空间通过 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。

项目

项目代表一种为开发团队和任务提供支持的协作努力。 CodeCatalyst 创建项目后，您可以添加、更新或删除用户和资源，自定义项目仪表板，并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，该蓝图会更新您的项目文件，例如软件依赖关系，并重新生成资源。项目使用名为 Projen 的工具通过同步最新的项目更新和生成支持文件来合成项目。根据您的应用程序类型和语言 `package.json` `Makefile` `eslint`，这些文件可能包括、、等。项目蓝图可以生成支持 CDK 构造、AWS 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 CodeCatalyst 工作流程操作。

有关账户关联的更多信息，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程操作并创建符合网络规则并可以访问关联 VPC 中资源的开发环境。

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 建造者 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，请参阅更新你的个人资料。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 (PAT)，但只能使用查看或删除 PAT。 AWS CLI您的用户名是您在注册时选择的别名。您无法更改您的用户名。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 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。您在注册时输入了此信息。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，请参阅更新你的个人资料。

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

源存储库是您安全地存储项目代码和文件的地方。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，则将创建一个源存储库，其中存储了项目配置信息。如果您创建了一个空项目，则必须先创建源存储库，然后才能创建需要配置信息（例如工作流程）的资源。

有关可以帮助您使用源代码库和源代码管理的更多概念，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。提交包含有关变更的信息，包括进行更改的用户的身份、更改的时间和日期、提交标题以及包含的有关变更的任何消息。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。您还可以在提交中添加 Git 标签，以帮助识别特定的提交。

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源存储库中的开发文件定义。如果您的源存储库中没有开发文件，则会自动应用默认的开发文件。默认的 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，开发环境配置为具有 2 核处理器、4 GB RAM 和 16 GiB 永久存储空间。

工作流

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

有关工作流的更多信息，请参阅使用中的工作流程构建、测试和部署 CodeCatalyst。

操作

操作是工作流程的主要组成部分，它定义了工作流程运行期间要执行的逻辑工作单元或任务。通常，一个工作流程包括多个按顺序运行或并行运行的操作，具体取决于您的配置方式。

有关操作的更多信息，请参阅配置工作流程执行的操作。

问题

问题是跟踪与您的项目相关工作的记录。您可以为功能、任务、错误或与项目相关的任何其他工作创建议题。如果你使用的是敏捷开发，问题也可以描述一个长篇故事或用户故事。

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

个人访问令牌 (PAT)

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 PAT 访问 CodeCatalyst 资源，包括集成开发环境 (IDE) 和基于 Git 的源存储库。PAT 代表你 CodeCatalyst ，你可以在用户设置中对其进行管理。一个用户可以拥有多个 PAT。个人访问令牌仅显示一次。作为最佳实践，请务必将其安全地存储在本地计算机上。默认情况下，PAT 将在一年后过期。

有关 PAT 的更多信息，请参阅使用个人访问令牌向用户授予存储库访问权限。

人际关系

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 相关联，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。当你邀请用户加入项目时，你可以为他们选择角色。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

您可以设置两种类型的空间 CodeCatalyst：支持 AWS Builder ID 用户的空间，以及创建支持身份联合的空间（在 IAM Identity Center 中管理 SSO 用户和群组）。 AWS 建筑商 ID 空间中的用户使用其 AWS 建造者 ID 登录，企业空间中的用户 CodeCatalyst 使用 CodeCatalyst 与该空间关联的公司的 SSO 门户登录。

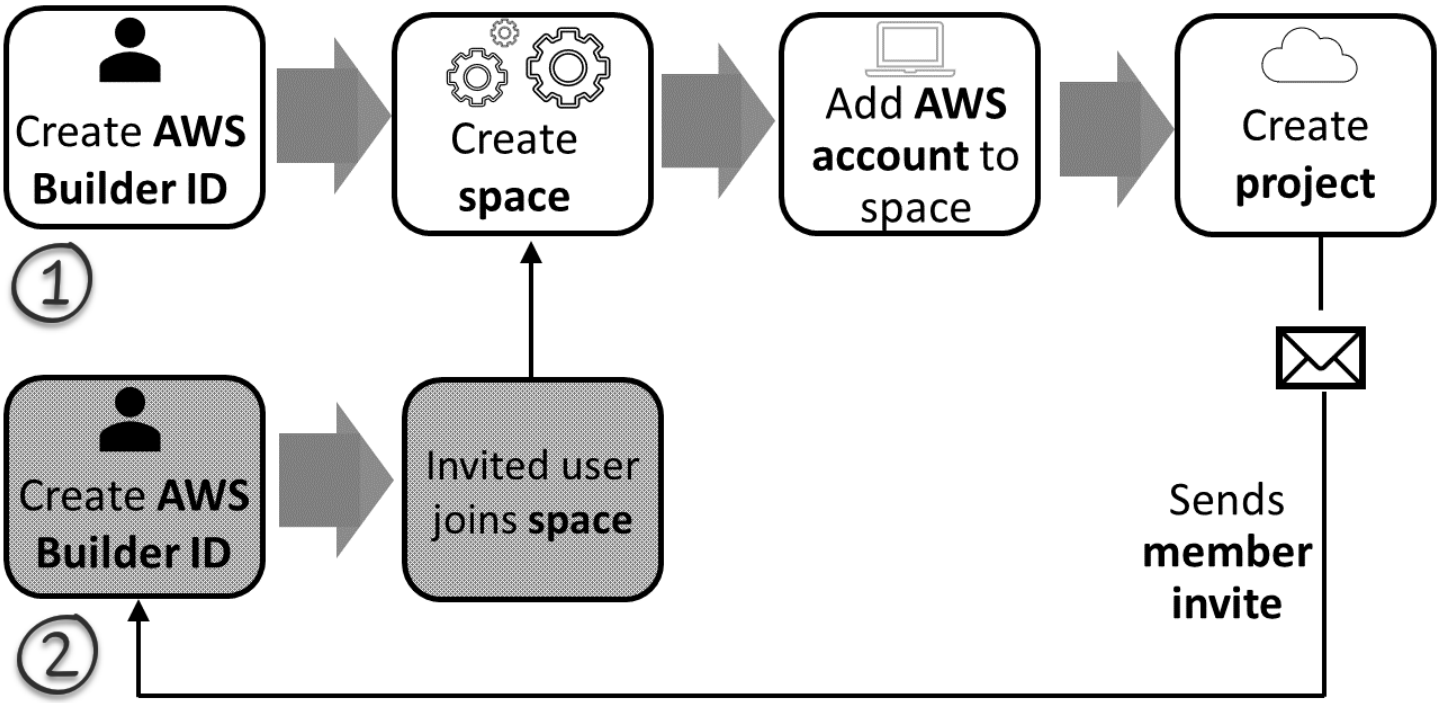
本指南提供了设置和管理 AWS 生成器 ID 空间的步骤。要使用 CodeCatalyst AWS 生成器 ID 空间，您需要 CodeCatalyst 使用登录时使用的用户设置和 AWS 生成器 ID 进行设置 CodeCatalyst。

《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。

本节提供了两种设置在亚马逊 CodeCatalyst 使用 AWS 建筑商 ID 空间的常用途径：以第一个用户身份创建空间和项目，以及接受现有空间或项目的邀请。这些设置工作流程必然大不相同。下图显示了两个注册过程，如下所示：

设置在 Amazon 工作有两种常见途径 CodeCatalyst：以第一个用户身份创建空间和项目，以及接受现有空间或项目的邀请。这些设置工作流程必然大不相同。下图显示了两个注册过程，如下所示：

1. 在第一种情况下，您为公司、团队或小组创建和设置空间，并在邀请其他人使用这些资源之前创建一个项目。出于计费目的， AWS 账户 必须提供，您仍然可以默认使用免费套餐。
2. 在第二种情况下，如果你 CodeCatalyst 通过接受项目邀请加入，则其他人已经为你创建了空间和项目。但是，您仍然需要配置自己的个人资料，以便可以开始与他人合作。



> **Tip**
>
> CodeCatalyst 使用空间对项目和资源进行分组。当你首次注册时 CodeCatalyst，系统会提示你创建空间和项目。

无论您是注册创建空间和项目，还是作为接受邀请的一部分进行注册，您都可以创建用于登录的 AWS 建筑商 ID CodeCatalyst。要创建 AWS Builder 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 CodeCatalyst 后，您可以使用电子邮件和密码登录。您也可以使用此 AWS 生成器 ID 登录其他使用 AWS 生成器 ID 凭据的应用程序。

在 CodeCatalyst AWS Builder ID 中，根据您的登录信息生成个人资料。您的个人资料包含您对 CodeCatalyst 项目中语言和通知设置的 CodeCatalyst 偏好。

> **Tip**
>
> 如果您在注册 Amazon CodeCatalyst 个人资料时遇到任何问题，请按照该页面提供的步骤操作。如果您需要其他帮助，请参阅注册时遇到问题。

主题

创建您的第一个空间和开发角色（无需邀请即可开始）

CodeCatalyst 无需邀请您加入现有空间或项目，即可注册 Amazon。完成后，您将在创建 AWS 建筑商 ID 后创建空间和项目。作为创建空间的一部分，您需要添加一个 AWS 账户 用于计费目的。

> **Tip**
>
> 如果您在注册 Amazon CodeCatalyst 个人资料时遇到任何问题，请按照该页面提供的步骤操作。如果您需要其他帮助，请参阅注册时遇到问题。

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

Mary Major 是一位对它感兴趣 CodeCatalyst 并决定试一试的开发者。她导航到 CodeCatalyst 控制台，选择注册并创建 AWS 建筑商 ID 的选项。Mary 提供了一个电子邮件地址和密码来创建她的 AWS Builder ID。她将能够使用自己的 AWS Builder ID 登录 CodeCatalyst 和其他应用程序。当被要求选择别名时，她将指定`MaryMajor`为将在中 CodeCatalyst 显示的 CodeCatalyst 用户名，其他项目成员也将使用该用户名来 @mention Mary。

接下来，系统会自动指示 Mary 创建空间。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，这样她就可以在自己的第一个项目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，在那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，然后她查看项目的蓝图列表。在查看了可用蓝图的信息后，她决定在自己的第一个项目中尝试使用现代三层 Web 应用程序蓝图。她填写必填字段并创建项目。项目准备就绪后，她就会被带到一个项目摘要页面，其中包含最近的活动以及指向项目代码的链接以及自动生成和部署该代码的工作流程。她探索了代码和工作流程，包括查看已部署的示例 Web 应用程序。喜欢她所看到的，她决定邀请一些同事参加这个项目，开始探索。 CodeCatalyst

当她有时间时，Mary 会配置她的 AWS Builder ID，使其 CodeCatalyst 使用多重身份验证 (MFA) 登录。配置 MFA 后，Mary 可以使用 CodeCatalyst 密码和经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。 CodeCatalyst

创建您的第一个空间和 IAM 角色

按照以下步骤注册您的 Amazon CodeCatalyst 个人资料，创建空间，为您的空间添加账户、支持角色和开发者角色。

最后一个过程是创建和添加开发者角色。开发人员角色是一 AWS 个 IAM 角色，可让您的 CodeCatalyst 工作流程访问 AWS 资源。开发者角色是用于管理的服务角色 AWS 服务 ，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。该角色将有一个名字`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，仅向需要管理空间中 AWS 资源访问权限的管理用户和开发人员分配管理权限。

在开始之前，您必须准备好为拥有管理权限的账户提供一个 AWS 账户 ID。请准备好 12 位数的 AWS 账户 身份证。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。

以新用户身份注册

1. 在开始使用 CodeCatalyst 控制台之前，请先打开 AWS Management Console，然后确保您登录时使用的账号与创建空间时使用的相同 AWS 账户 。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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 建筑商 ID 是您为登录而创建的身份。它和... 不一样 AWS 账户。
4. 在您的电子邮件地址中，输入您要关联的电子邮件地址 CodeCatalyst。然后选择下一步。
5. 在 “您的姓名” 中，提供您希望在使用 AWS 构建器 ID 的应用程序中显示的名字和姓氏。允许有空格。这将是你的 AWS Builder ID 个人资料名称，例如 Mary Major。您可以稍后更改名称。

   选择下一步。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箱。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，也无法在垃圾邮件或垃圾邮件文件夹中找到该验证码，请选择 “重新发送验证码”。
7. 一旦我们验证了您的验证码，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 Builder ID。
8. 在创建您的 CodeCatalyst 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。您以后不能更改您的 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，但项目成员将使用您的别名来表示 @mention you。

   选择下一步。页面将更新为显示创建您的 CodeCatalyst 空间部分。
9. 在命名您的空间中，输入您的空间名称。您以后无法更改此设置。

   > **Note**
   >
   > 空间名称在各处必须是唯一的 CodeCatalyst。您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。您以后无法更改此设置。
11. 选择下一步。页面将更新以显示用于添加的页面 AWS 账户。此账户将用作该空间的账单账户。
12. 在 AWS 账户 ID 中，输入您要连接到空间的账户的十二位数 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。
13. 选择前往 AWS 控制台进行验证。
14. “验证 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。这是 Amazon CodeCatalyst 空间页面。您可能需要登录才能访问该页面。

    在中 AWS Management Console，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，请登录亚马逊 CodeCatalyst 空间，网址为 AWS Management Console https://console.aws.amazon.com/codecatalyst/home/。

    中的验证令牌字段会自动填充中生成的令牌 CodeCatalyst。 AWS Management Console

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），为您的账单账户开启付费套餐。

> **Note**
>
> 这不会将计费等级升级到付费等级。但是，这将进行配置， AWS 账户 以便您可以随时更改空间的计费等级。 CodeCatalyst您可以随时开启付费等级。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。

    系统将显示账户验证成功消息，表明该账户已被添加到空间。

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。

    包含CodeCatalyst 空间详细信息的连接页面将在中打开 AWS Management Console。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，会显示 CodeCatalyst 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：

    - 立即选择 “跳过”。
    - 选择为您的空间创建您的第一个项目。有关向您展示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。

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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。

5. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建了一个服务角色，其中包含开发角色的权限策略和信任策略。该角色将有一个名字`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。

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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。
4. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建的服务角色包含预览版开发角色的权限策略和信任策略。该角色的AWSRoleForCodeCatalystSupport名称将附加唯一标识符。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 AWS Builder ID、创建第一个空间并添加账户后，即可创建项目。有关更多信息，请参阅 创建项目。如果这是您第一次使用 CodeCatalyst，我们建议从开始教程：使用现代三层 Web 应用程序蓝图创建项目。

接受邀请并创建您的 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，请参阅注册时遇到问题。

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 建筑商 ID。Saanvi 将能够使用她的 AWS Builder ID 登录 CodeCatalyst 和其他应用程序。稍后，她可以编辑自己的个人资料以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 的别名与她分配的项目角色一起显示在成员列表中。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 在项目上工作时，她的别名将在该项目的工作活动日志中列出。Saanvi 的问题和评论将显示她的别名，其他项目成员可以在回复中用 @mention 回复她。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 使用您的AWS建筑商 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。

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。它和... 不一样 AWS 账户。

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。允许有空格。这将是你的 AWS Builder ID 个人资料名称，例如 Mary Major。您可以稍后更改名称。

   选择下一步。

   验证码将发送到您指定的电子邮箱。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，也无法在垃圾邮件或垃圾邮件文件夹中找到该验证码，请选择 “重新发送验证码”。
4. 验证代码后，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，但项目成员将使用您的别名来表示 @mention you。

选择创建别名。您将被带到受邀进入的项目或空间。

使用您的AWS建筑商 ID 登录

请按照以下步骤登录您的Amazon CodeCatalyst 个人资料。

> **Note**
>
> 您是否已经注册了用于多重身份验证 (MFA) 的设备？ 我们强烈建议您在 Amazon 中配置 MFA CodeCatalyst 以提高安全性。有关更多信息，请参阅如何注册设备以使用多因素身份验证：

使用您的 AWS Builder ID 登录

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 输入您的电子邮件地址。或者，如果您想保存电子邮件地址以备将来登录，请选择 “保存我的电子邮件地址”。选择 Continue（继续）。
3. 输入您的 Password（密码）。选择 Sign in (登录)。如果您忘记了密码，请按照中的步骤操作我忘记密码了。

受信任设备

在您从登录页面选择 “这是一台可信设备” 选项后，Amazon 会将该设备的所有未来登录 CodeCatalyst 视为已授权。只要您使用可信设备，Amazon 就 CodeCatalyst 不会提供输入 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 登录亚马逊 CodeCatalyst。

要改用您的AWS生成器 ID 登录，请参阅使用您的AWS建筑商 ID 登录。

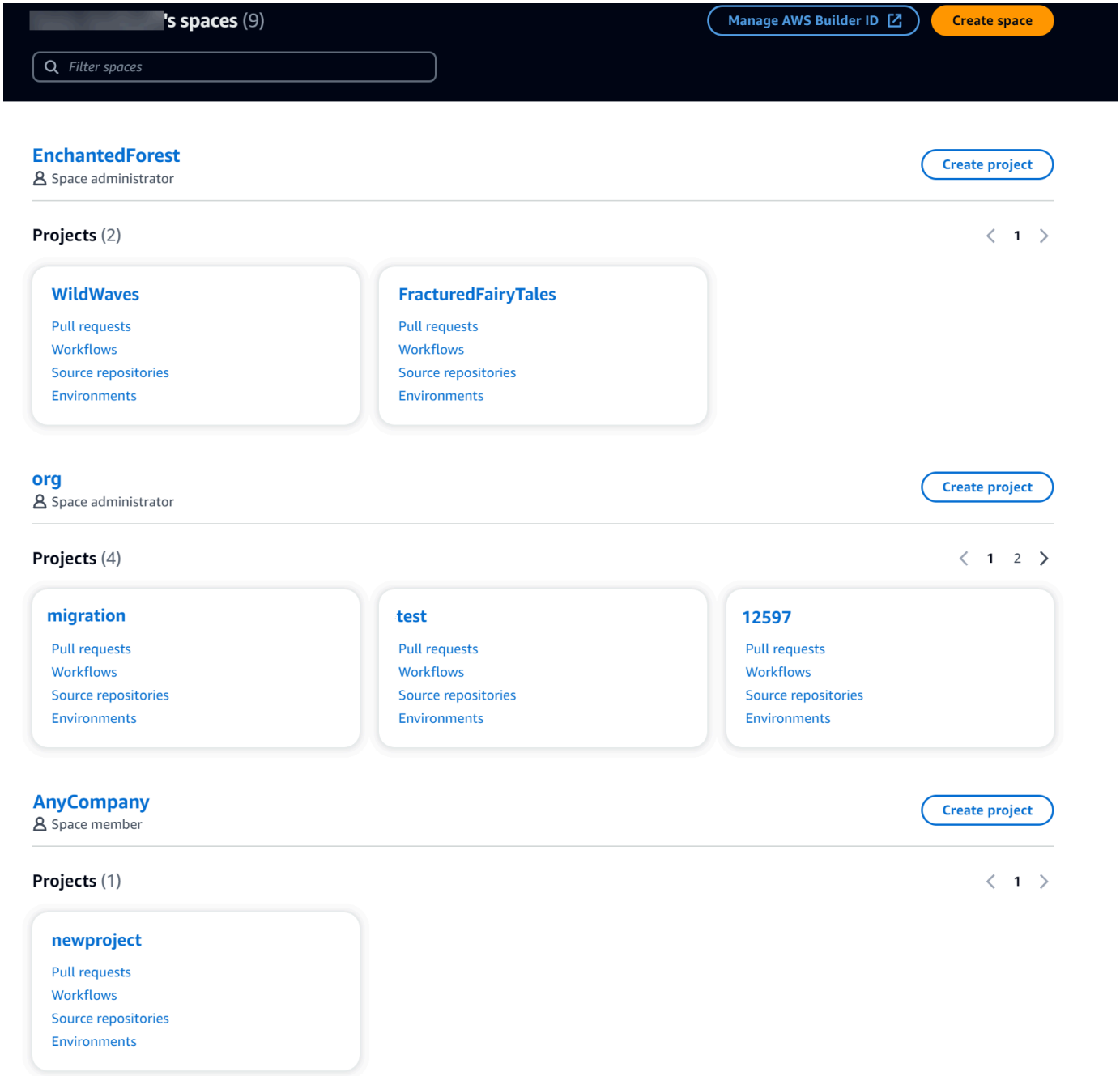
使用 SSO 登录

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 “选择登录选项” 下，选择 “使用单点登录 (SSO)”。
3. 在 AWSIden tity Center 应用程序名称中，输入您的联合身份验证管理员提供的应用程序名称。
4. 选择继续进入 IAM 身份中心。

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列表。用户主页显示用户所属的每个空间、该空间中用户的角色（例如空间管理员）以及该用户拥有成员资格的每个空间中的项目的列表。

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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看到的空间或项目，则可能需要以其他用户身份登录。

查看和管理 CodeCatalyst 个人资料

您可以在 Amazon 中查看用户资料 CodeCatalyst 以获取诸如电子邮件地址和 CodeCatalyst 别名之类的信息。您也可以更新您的个人资料和AWS建造者 ID。如果您忘记了密码，可以申请重置密码。

查看您的 CodeCatalyst 个人资料

您在注册时提供的信息将用作登录亚马逊 CodeCatalyst 的凭证，这些信息将在您的个人资料中进行管理。这包括您的姓名、昵称和用于登录的电子邮件地址 CodeCatalyst。

> **Note**
>
> AWSBuilder ID 昵称不是您的 CodeCatalyst 别名。您在注册时选择了您的 CodeCatalyst 别名。

查看您的 CodeCatalyst 个人资料

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在右上角，选择带有第一个首字母缩写的图标旁边的箭头，然后选择我的设置。 CodeCatalyst 我的设置页面打开。
3. 要更新您的AWS建筑商 ID 电子邮件地址或密码，或者要设置 MFA，请选择管理AWS建筑商 ID。将打开 “AWS生成器 ID” 页面。

查看其他用户的 CodeCatalyst个人资料

查看其他用户的 CodeCatalyst 个人资料

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在侧面导航栏中，选择项目设置。选择 “成员” 选项卡。查看您 CodeCatalyst项目的成员列表。
3. 选择你要查找的成员名或 @mention。我的设置页面显示用户的别名、电子邮件地址和全名。使用 @mention 项目成员的 CodeCatalyst 别名。

   > **Note**
   >
   > 用户的 AWS Builder 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个人资料，请在列表中选择他们的姓名。

更新你的个人资料

在中 CodeCatalyst，您的个人资料包含由 AWSBuilder ID 管理的个人信息和中管理的设置 CodeCatalyst。

- 您的个人资料的全名、电子邮件地址和密码由 AWSBuilder ID 管理。您在注册时输入了此信息。当您将 MFA 设置为使用身份验证器应用程序登录时，会将您 CodeCatalyst 带到生成器 ID 页面。AWS
- CodeCatalyst 您的个人访问令牌 (PAT)、 CodeCatalyst 通知和语言首选项的设置可在的 “我的设置” 页面中进行管理 CodeCatalyst。有关更多信息，请参阅使用个人访问令牌向用户授予存储库访问权限：

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

更新您的AWS建筑商 ID 或电子邮件地址

更新您的地址AWS 构建者 ID或电子邮件地址

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在右上角，选择带有第一个首字母缩写的图标旁边的箭头，然后选择我的设置。 CodeCatalyst我的设置页面打开。
3. 在个人资料页面上，选择管理AWS生成器 ID。将打开 “AWS生成器 ID” 页面。
4. 在页面的左侧，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” 字段将反映全名中的名字。这不是你的 CodeCatalyst 别名。

   > **Note**
   >
   > 这会更新AWS生成器 ID 的全名和名字。这不会更新您的 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**
>
> 这将更新您将用于登录的电子邮件地址 CodeCatalyst。

更改您的 CodeCatalyst 密码

要更改您的 CodeCatalyst 密码

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在右上角，选择带有第一个首字母缩写的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst 我的设置页面打开。
3. 在个人资料页面上，选择管理AWS生成器 ID。将打开 “AWS生成器 ID” 页面。
4. 在页面的左侧，选择 “安全”。
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codecatalyst用您的个人资料名称替换。

```
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存储库访问权限
- 使用事件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来帮助您开始项目。您也可以选择从一个空项目开始，然后向其添加资源。按照这些教程中的步骤学习一些工作方式 CodeCatalyst。

如果这是您第一次使用 CodeCatalyst，我们建议从开始教程：使用现代三层 Web 应用程序蓝图创建项目。

> **Note**
>
> 要学习这些教程，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
- 教程：使用可组合的 PDK 蓝图创建全栈应用程序

有关侧重于中特定功能区域的其他教程 CodeCatalyst，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
- 工作流程入门
- 自定义蓝图入门
- 开始阅读 Amazon CodeCatalyst 操作开发者指南

有关深入的教程，请参阅：

- 教程：将构件上传到 Amazon S3
- 教程：使用部署无服务器应用程序 AWS CloudFormation
- 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：在工作流程中使用 GitHub操作的 Lint 代码

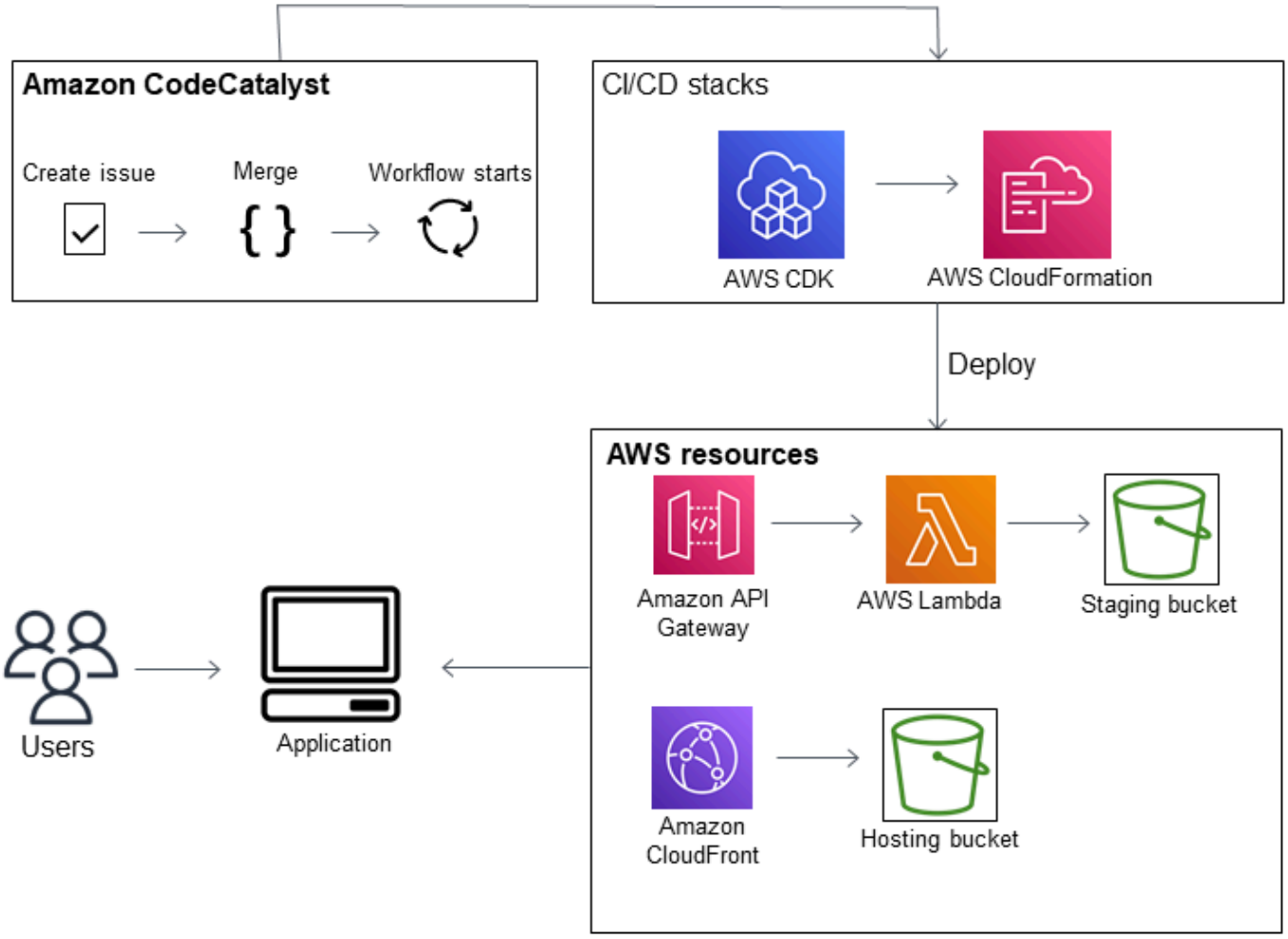
- 教程：创建和更新 React 应用程序

教程：使用现代三层 Web 应用程序蓝图创建项目

通过创建带有蓝图的项目，可以更快地开始开发软件。使用蓝图创建的项目包括您需要的资源，包括用于管理代码的源存储库以及用于生成和部署应用程序的工作流程。在本教程中，我们将引导您使用现代三层 Web 应用程序蓝图在 Amazon CodeCatalyst 中创建项目。本教程还包括查看已部署的示例、邀请其他用户使用该示例，以及使用拉取请求对代码进行更改，拉取请求在合并拉取请求 AWS 账户 时会自动生成并部署到连接中的资源。在使用报告、活动源和其他工具 CodeCatalyst 创建项目的地方，蓝图在与您的项目 AWS 账户 关联的中创建 AWS 资源。蓝图文件允许您构建和测试样本现代应用程序，并将其部署到中的基础架构 AWS Cloud。

下图显示了如何使用中的 CodeCatalyst 工具来创建用于跟踪、合并和自动生成变更的问题，然后在CodeCatalyst项目中启动工作流程，该工作流运行操作以允许 AWS CDK 和 AWS CloudFormation 配置您的基础架构。

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



使用现代三层 Web 应用程序蓝图创建项目时，将使用以下资源创建项目：

在 CodeCatalyst 项目中：

- 包含示例代码和工作流程 YAML 的源代码库
- 一种工作流程，每当对默认分支进行更改时都会生成和部署示例代码
- 议题板和待办事项列表，可用于计划和跟踪工作
- 示例代码中包含自动报告的测试报告套件

在相关的 AWS 账户：

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

有关将在本教程中创建的资源 AWS 以及 CodeCatalyst 作为本教程一部分创建的资源的更多详细信息，请参阅参考。

> **Note**
>
> 项目中包含的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应用程序项目，必须设置并登录 CodeCatalyst按以下方式完成中的任务：

- 拥有用于登录的 AWS 生成器 ID CodeCatalyst。
- 属于某个空间，并在该空间中为您分配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。
- 与您的空间 AWS 账户 相关联，并拥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角色策略的角色策略创建服务CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。该角色的`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*名

称将附加唯一标识符。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：创建现代三层 Web 应用程序项目

创建完项目后，您可以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。您的项目还包含您的开发工具和资源。

在本教程中，您将使用现代三层 Web 应用程序蓝图来创建交互式应用程序。作为项目的一部分自动创建和运行的工作流程将生成和部署应用程序。只有在为您的空间配置了所有角色和账户信息后，工作流程才会成功运行。工作流程成功运行后，您可以访问终端节点 URL 以查看应用程序。

使用蓝图创建项目

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. 选择现代三层 Web 应用程序蓝图，然后选择下一步。
7. 在为项目命名中，输入项目名称。例如：

   MyExampleProject.

   > **Note**
   >
   > 该名称在您的空间中必须是唯一的。

8. 在“帐户”中，选择 AWS 账户 您在注册时添加的。蓝图会将资源安装到此账户中。
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 角色并将该角色添加到您的 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 托管。有关信息 AWS Amplify，请参阅什么是 AWS Amplify 托管？ 在《AWS Amplify 用户指南》中。
12. 在部署区域中，输入您希望蓝图在 AWS 区域 哪里部署 Mysfits 应用程序和支持资源的区域代码。有关区域代码的列表，请参阅中的区域终端节点AWS 一般参考。
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。选择查看工作流程以预览蓝图将安装的 CI/CD 工作流定义文件。预览会根据您的选择动态更新。
15. 选择创建项目。

项目工作流程将在您创建项目后立即启动。完成代码的构建和部署需要一点时间。同时，继续邀请其他人加入你的项目。

第 2 步：邀请他人加入您的项目

既然您已经设置了项目，请邀请其他人与您合作。

邀请他人加入您的项目

1. 导航到您要邀请用户加入的项目。
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. 键入您想要邀请成为项目用户的人员的电子邮件地址。您可以键入多个电子邮件地址，用空格或逗号分隔。您也可以从空间中不是项目成员的成员中进行选择。
5. 为用户选择角色。

   添加完用户后，选择邀请。

第 3 步：创建要协作处理的议题并跟踪工作

CodeCatalyst 帮助您跟踪项目中涉及的功能、任务、错误以及任何其他有问题的工作。您可以创建议题来跟踪所需的工作和想法。默认情况下，当您创建议题时，它会添加到待办事项中。您可以将问题移至图板，在那里您可以跟踪正在进行的工作。您也可以将议题分配给特定的项目成员。

为项目创建议题

1. 在导航窗格中，选择 “问题”。

2. 选择创建问题。
3. 在问题标题中，提供问题的名称。（可选）提供问题描述。在此示例中，使用 **make a change in the src/mysfit_data.json file.**
4. 选择优先级、估计值、状态和标签。在 assign ee 下，选择 +Add me 将问题分配给自己。
5. 选择创建问题。该问题现已显示在黑板上。选择卡片将问题移至 “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。

第 4 步：查看您的源代码库

您的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。源存储库中一些值得注意的目录和文件是：

- .cloud9 目录 — 包含 AWS Cloud9 开发环境的支持文件。
- .codecatalyst 目录-包含YAML蓝图中包含的每个工作流程的工作流程定义文件。
- .ide a 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 — 包含在中定义基础架构的 AWS CDK 堆栈文件。 AWS Cloud
- 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，这些文件是在您生成和测试应用程序时运行的自动 CI/CD 工作流程的一部分运行的。
- 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文件、网站`index.html`页面、用于整理代码`.eslintrc.cjs`的文件以及用于指定根`tsconfig.json`文件和编译器选项的文件。`package.json`
- `Dockerfile`file-描述应用程序的容器。
- `README.md`file-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，然后执行以下任一操作：
   - 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来筛选存储库列表。

2. 在存储库的主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求的数量和工作流程。默认情况下，会显示默认分支的内容。您可以通过从下拉列表中选择其他分支来更改视图。

第 5 步：创建带有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来快速处理源存储库中的代码。在本教程中，我们假设你会：

- 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。
- 使用这个新分支`test`的名称。

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. 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。或者，在导航窗格中，选择代码，选择源存储库，然后选择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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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。

在本节中，您将 CodeCatalyst 通过更改包含拉取请求的代码来使用生成的示例应用程序，拉取请求在合并拉取请求 AWS 账户 时会自动生成并部署到连接中的资源。

要更改您的**src/mysfit_data.json**文件

1. 导航到您的项目开发环境。在中 AWS Cloud9，展开侧面导航菜单以浏览文件。展开mysfitssrc、和打开src/mysfit_data.json。
2. 在文件中，将该"Age":字段的值从 6 更改为 12。您的行应如下所示：

```
    {
        "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 comm it 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：查看构建现代应用程序的工作流程

创建现代应用程序项目后，代表您 CodeCatalyst 生成多个资源，包括工作流程。工作流程是在.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 创建了一个工作流程，并在您创建项目时自动启动了该工作流程。（工作流程可能仍在运行，具体取决于您创建项目的时间。）使用以下过程检查工作流程的进度，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。

查看工作流程进度

1. 在 CodeCatalyst 控制台的导航窗格中，选择 C I/CD，然后选择 Workflows。

   此时将显示工作流程列表。这些是 CodeCatalyst 蓝图在您创建项目时生成和启动的工作流程。

2. 查看工作流程列表。你应该看到四个：

   - 顶部的两个工作流程对应于您之前在中创建的`test`分支[第 5 步：创建带有测试分支的开发环境并快速更改代码](#)。这些工作流程是`main`分支上的工作流程的克隆。之所以ApplicationDeploymentPipeline处于非活动状态，是因为它已配置为与`main`分支一起使用。由于未发出拉取请求，OnPullRequest工作流程未运行。
   - 底部的两个工作流程对应于你之前运行蓝图时创建的`main`分支。工作ApplicationDeploymentPipeline流程处于活动状态，正在运行（或已完成）。

   > **Note**
   >
   > 如果ApplicationDeploymentPipeline运行失败并出现 Bu ild @cdk_bootstrap 或DeployBackend错误，那可能是因为你之前运行过现代三层 Web 应用程序，但它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程。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ApplicationDeploymentPipeline的工作流程。此工作流程是使用`main`分支上的源代码运行的。

   将出现工作流程图。该图显示了几个方块，每个方块代表一项任务或操作。大多数动作都是垂直排列的，顶部的动作先于下面的动作。并排排列的动作并行运行。分组在一起的操作必须全部成功运行，然后才能开始它们下面的操作。

   主要区块是：

   - WorkflowSource— 此区块代表您的源存储库。除其他信息外，它还显示源存储库名称 (mysfit s) 和自动启动工作流程运行的提交。 CodeCatalyst 在你创建项目时生成了这个提交。
   - 构建 — 此方块代表两个操作的分组，这两个操作都必须成功完成才能开始下一个操作。
   - DeployBackend— 此块表示将应用程序的后端组件部署到 AWS 云中的操作。

- 测试 — 此块代表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— 此块表示将应用程序的前端组件部署到云中的 AWS 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。该文件包含以下值得注意的部分：

   - 一个`Triggers`部分，在顶部。本节指出，每当将代码推送到源存储库的`main`分支时，工作流程都必须启动。推送到其他分支（例如`test`）不会启动此工作流程。工作流程使用`main`分支上的文件运行。
   - 下方有一`Actions`节`Triggers`。本节定义了您在工作流程图中看到的操作。

5. 选择 “最新状态” 选项卡（靠近顶部），然后在工作流程图中选择任意操作。
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。每个配置设置在工作流定义文件中都有一个匹配的属性。
7. 让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. 在工作流程图中，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，然后展开几个部分以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。 CodeCatalyst 显示关联的报告。该报告显示了已运行的代码覆盖率测试，并指出了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，请参阅使用工作流程进行测试。

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

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

确认现代应用程序已成功部署

1. 返回ApplicationDeploymentPipeline工作流程，然后选择最新运行的 “运行###” 链接。
2. 在工作流程图中，找到DeployFrontend操作并选择 “查看应用程序” 链接。Mysfit 网站出现了。

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

3. 搜索名为 Twilight Glitter 的飞马 Mysfit。记下年龄的值。确实如此6。您将更改代码以更新年龄。

第 7 步：让其他人查看您的更改

现在，您已在名为的分支中进行了更改`test`，您可以通过创建拉取请求来要求其他人对其进行审查。执行以下步骤创建拉取请求，以将分支中的更改合并到分`testmain`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，选择代码，选择拉取请求，然后选择创建拉取请求。
   - 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已提交代码的存储库。只有在您没有从存储库的主页创建拉取请求时，才会显示此选项。
4. 在 D est ination 分支中，在查看代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含已提交代码的分支。
6. 在 Pull req uest 标题中，输入一个标题，以帮助其他用户了解需要审阅的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如问题链接或更改描述之类的信息。

> **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。将自动生成的描述添加到拉取请求后，您可以对其进行更改。
> 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

8. （可选）在“问题”中，选择“关联问题”，然后从列表中选择议题或输入其 ID。要取消问题链接，请选择取消链接图标。
9. （可选）在必填审稿人中，选择添加所需的审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，必需的审阅者必须批准更改。

> **Note**
>
> 您不能将审阅者同时添加为必填审阅者和可选审阅者。您不能将自己添加为审阅者。

10. （可选）在可选审阅者中，选择添加可选审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，可选的审阅者不必将更改作为一项要求进行批准。
11. 查看分支之间的差异。拉取请求中显示的区别在于源分支中的修订版本和合并基础之间的变化，合并基础是创建拉取请求时目标分支的头部提交。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都选择了相同的分支。
12. 如果您对拉取请求中包含要查看的代码和更改感到满意，请选择“创建”。

> **Note**
>
> 创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用 @ 符号和文件名来添加指向资源（例如文件）的链接。

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

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

合并来自 CodeCatalyst 控制台的拉取请求

1. 导航到您的现代应用程序项目。
2. 在项目页面的“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”，然后从列表中进行选择。选择 Merge (合并)。
3. 从拉取请求的可用合并策略中进行选择。（可选）选择或取消选择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，或者无法在控制台中合并拉取请求。 CodeCatalyst 拉取请求详情区域概述中的时钟图标会显示尚未批准拉取请求的审阅者。如果所有必需的审阅者都批准了拉取请求，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`test`分支中的更改合并到分**`main`**支后，更改会自动启动构建和部署更改ApplicationDeploymentPipeline的工作流程。

要查看合并的提交，请执行 ApplicationDeploymentPipeline 工作流程

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 在 “工作流程” 中 ApplicationDeploymentPipeline，展开 “最近运行”。你可以看到工作流程的运行是由合并提交启动的。（可选）选择它来观看运行进度。
3. 运行完成后，重新加载您之前访问过的 URL。查看飞马以验证年龄是否发生了变化。

第 8 步：关闭问题

问题解决后，可以在 CodeCatalyst 控制台上将其关闭。

关闭项目的议题

1. 导航到您的项目。
2. 在导航窗格中，选择 “问题”。
3. rag-and-drop 将问题发送到 “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

清理 CodeCatalyst AWS 并从您的环境中删除本教程的痕迹。

您可以选择继续使用您在本教程中使用的项目，也可以删除该项目及其关联资源。

> **Note**
>
> 删除此项目将删除项目中所有成员的所有存储库、议题和项目。

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 AWS CloudFormation 和 Amazon S3 中的资源

1. AWS Management Console 使用您在 CodeCatalyst空间中添加的相同帐户登录。
2. 前往AWS CloudFormation服务。
3. 删除 mysfits ***###***堆栈。
4. *## developm* ***ent-Mysfits #####***
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，则在重新运行现代三层 Web 应用程序蓝图时可能会看到错误。

6. （可选）删除 CDKToolkit 堆栈。

参考

现代三层 Web 应用程序蓝图将资源部署到您的 CodeCatalyst 空间中，将您的 AWS 帐户部署到云中。AWS 这些资源是：

- 在你的 CodeCatalyst 空间里：
  - 包含以下资源的 CodeCatalyst 项目：
    - 源代码库 — 此存储库包含 “Mysfits” Web 应用程序的示例代码。
    - 工作流程 — 每当对默认分支进行更改时，此工作流程都会构建和部署 Mysfits 应用程序代码
    - 问题板和待办事项列表 — 此看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报告。
- 在相关的 AWS 账户：
  - 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密 AWS KMS 钥。
    - 用于部署操作的 IAM 部署角色。
    - AWS 支持堆栈中资源的 IAM 角色和策略。

    > **Note**
    >
    > CDKToolkit 不会针对每次部署进行拆卸和重新创建。这是一个在每个账户中启动的堆栈，用于支持 AWS CDK。

  - dev elopment-mysfits ###BackEnd堆栈 — 此堆栈部署以下后端资源：
    - 亚马逊 API Gateway 终端节点。
    - AWS 支持堆栈中资源的 IAM 角色和策略。
    - AWS Lambda 函数和层为现代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 ###堆栈 — 此堆栈部署 AWS Amplify 前端应用程序。

另请参阅

有关在本教程中创建资源的 AWS 服务的更多信息，请参阅以下内容：

- Amazon S3 — 一种在对象存储服务上存储前端资产的服务，可提供业界领先的可扩展性、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 — 一项用于创建、发布、维护、监控和保护任何规模的 REST、HTTP 和 WebSocket API 的服务。有关更多信息，请参阅 API Gateway 开发者指南。
- 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 主机用户指南》。
- AWS Cloud Development Kit (AWS CDK)— 一个框架，用于在代码中定义云基础架构并通过它进行配置 AWS CloudFormation。 AWS CDK 包括 AWS CDK Toolkit，这是一款用于与 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 Dy namoDB — 一种用于存储数据的完全托管的 NoSQL 数据库服务。有关更多信息，请参阅 Amazon DynamoDB 开发人员指南。
- AWS Lambda— 一项无需预置或管理服务器即可在高可用性计算基础设施上调用代码的服务。有关更多信息，请参阅 AWS Lambda 开发人员指南。
- AWS IAM — 一项用于安全控制访问 AWS 及其资源的服务。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，可以创建一个内部没有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。由于在没有蓝图的情况下创建的项目在创建时为空，因此此选项需要更多有关创建和配置 CodeCatalyst 资源的知识才能开始使用。

主题

- 邀请他人加入你的项目
- 创建要协作处理的议题并跟踪工作

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。如果这是您第一次登录 CodeCatalyst，请参阅设置并登录 CodeCatalyst。

创建一个空项目

创建项目是能够协同工作的第一步。如果您想创建自己的资源，例如源存储库和工作流程，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。

现在你有一个空项目，下一步是创建一个源存储库。

创建源存储库

创建源代码库，用于存储和协作处理项目的代码。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，您可以选择链接托管在受支持服务中的存储库，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. 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，添加存储库的描述，以帮助项目中的其他用户了解存储库的用途。
7. （可选）为您计划推送的代码类型添加.gitignore文件。
8. 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，但无法更改或删除默认分支。

您可以通过创建开发环境在仓库中快速添加代码。在本教程中，我们建议您使用创建开发环境 AWS Cloud9，并在创建开发环境时选择从主分支创建分支的选项。我们使用此分支**test**的名称，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. 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。或者，在导航窗格中，选择代码，选择源存储库，然后选择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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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、测试和部署代码变更的工作流程

在中 CodeCatalyst，您可以在工作流程中组织应用程序或服务的构建、测试和部署。工作流程由操作组成，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，请参阅使用中的工作流程构建、测试和部署 CodeCatalyst。

按照中的说明创建工作流程入门您的第一个工作流程。

邀请他人加入你的项目

现在，您已经设置了自定义项目，请邀请其他人与您合作。

邀请他人加入您的项目

1. 导航到您要邀请用户加入的项目。
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. 键入您想要邀请成为项目用户的人员的电子邮件地址。您可以键入多个电子邮件地址，用空格或逗号分隔。您也可以从空间中不是项目成员的成员中进行选择。
5. 为用户选择角色。

   添加完用户后，选择邀请。

创建要协作处理的议题并跟踪工作

CodeCatalyst 帮助您跟踪项目中涉及的功能、任务、错误以及任何其他有问题的工作。您可以创建议题来跟踪所需的工作和想法。默认情况下，当您创建议题时，它会被添加到待办事项中。您可以将议题移至图板，在那里您可以跟踪正在进行的工作。您也可以将议题分配给特定的项目成员。

为项目创建议题

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

   确保您正在要创建问题的项目中导航。要查看所有项目，请在导航窗格中选择 Amazon CodeCatalyst，如果需要，选择查看所有项目。选择要在其中创建或处理议题的项目。
2. 在导航窗格中，选择 “跟踪”，然后选择 “待办事项列表”。
3. 选择 “创建问题”。

4. 在问题标题中，提供问题的名称。（可选）提供问题描述。如果需要，可以选择问题的状态、优先级和估计值。您也可以将议题分配给项目成员列表中的项目成员。

> **Tip**
>
> 您可以选择将问题分配给 Amazon Q，让 Amazon Q 尝试解决问题。如果尝试成功，将创建拉取请求，问题的状态将更改为“正在审核”，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。
> 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

5. 选择保存。

创建议题后，您可以将其分配给项目成员，对其进行估算，然后在看板上对其进行跟踪。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，他们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不言自明。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，请参阅什么是 Amazon Q 开发者？ 在 Amazon Q 开发者用户指南中。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于用于软件开发的 Amazon Q 开发者代理的“为我写描述”、“创建内容摘要”和“将问题分配给 Amazon Q”功能是在 Amazon Bedrock 上构建的，因此用户可以充分利用 Amazon Bedrock 中实施的控制措施来加强安全、保障和负责任地使用人工智能 (AI)。

在本教程中，你将学习如何使用中的生成式 AI 功能 CodeCatalyst 来帮助你在创建拉取请求时总结分支之间的变化，并总结拉取请求中留下的评论。您还将学习如何根据自己的代码更改或改进想法创建问题并将其分配给 Amazon Q。在处理分配给 Amazon Q 的问题时，您将学习如何允许 Amazon Q 提出任务建议，以及如何分配和处理它在处理问题时创建的任何任务。

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
- 您的项目位于启用生成式 AI 功能的空间中。有关更多信息，请参阅管理生成式 AI 功能。
- 在该空间的项目中，您具有参与者或项目管理员角色。
- 该项目至少为其配置了一个源存储库。不支持链接存储库。
- 在分配问题以使用生成式 AI 创建初始解决方案时，无法使用 Jira Softwar e 扩展程序配置项目。此功能不支持该扩展。

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。如果您使用使用其他蓝图创建的项目，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审阅、评论代码更改以及将代码更改从一个分支合并到另一个分支的主要方式。您可以使用拉取请求以协作方式查看代码更改，以了解已发布软件的细微更改或修复、主要功能添加或新版本。作为拉取请求描述的一部分，总结代码更改和变更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而发生的变化的历史记录。但是，开发人员通常依靠自己的代码来解释自己或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变更背后的意图。

在创建拉取请求时，您可以使用 “为我写描述” 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，它总结了这些变化的内容，以及对这些变化的意图和效果的最佳解释。

> **Note**
>
> 此功能不适用于 Git 子模块。它不会汇总作为拉取请求一部分的 Git 子模块中的任何更改。

你可以用你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 Modern 三层 Web 应用程序蓝图中创建的项目中包含的代码进行一些简单的更改来对其进行测试。

> **Tip**
>
> 如果您使用的是使用不同的蓝图或您自己的代码创建的项目，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的项目中的代码不匹配。与其使用下面建议的示例，不如在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，你将在源存储库中创建一个分支。然后，您将使用控制台中的文本编辑器对该分支中的文件进行快速代码更改。然后，您将创建一个拉取请求，并使用 “为我写描述” 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，只需进行简单的更改即可编辑该分支中的文件。在此示例中，您将编辑`test_endpoint.py`文件以将测试的重试次数从更改**3**为 5。

> **Tip**
>
> 您也可以选择创建或使用开发环境来更改此代码。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更改为：

```
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，您已经有一个包含更改的分支，您可以创建拉取请求。

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 D est ination 分支中，在查看代码后，选择要将代码合并到的分支。

> **Tip**
>
> 选择您在前面的过程中创建分支的分支，以最简单地演示此功能。例如，如果您从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 S ourc e 分支中，选择包含您刚刚提交给该`test_endpoint.py`文件的更改的分支。
4. 在 Pull req uest 标题中，输入一个标题，以帮助其他用户了解需要审阅的内容及其原因。
5. 在拉取请求描述中，选择为我写描述，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中。
7. （可选）修改摘要以更好地反映您对代码所做的更改。您也可以选择添加审阅者或将议题链接到此拉取请求。完成所需的任何其他更改后，选择 “创建”。

创建拉取请求中对代码更改留下的评论摘要

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评论。如果有很多审稿人发表了很多评论，则可能很难在反馈中挑出共同的主题，甚至很难确保你已经审阅了所有修订版中的所有评论。您可以使用 “创建评论摘要” 功能让 Amazon Q 分析拉取请求中代码更改时留下的所有评论，并创建这些评论的摘要。

> **Note**
>
> 评论摘要是短暂的。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评论，只包括对拉取请求修订版中代码差异的评论。
> 此功能不适用于在 Git 子模块中对代码更改留下的任何评论。

在拉取请求中创建评论摘要

1. 导航到您在上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。在导航栏中，选择代码，选择拉取请求，然后选择任何已打开的拉取请求。

2. 如果拉取请求还没有评论，请在 “更改” 中向拉取请求添加一些评论。
3. 在概述中，选择创建评论摘要。完成后，评论摘要部分将展开。
4. 查看拉取请求修订版中对代码变更留下的评论摘要，并将其与拉取请求中的评论进行比较。

创建议题并将其分配给 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。您可以将问题分配给名为 A mazon Q 的生成式 AI 助手，该助手可以根据问题的标题和描述来分析问题。如果您将问题分配给 Amazon Q，它将尝试创建解决方案草案供您评估。这可以帮助您和您的团队将工作重点放在需要您注意的问题上，并对其进行优化，而 Amazon Q 则为您没有资源可以立即解决的问题提供解决方案。

> **Tip**
>
> Amazon Q 在简单问题和直截了当的问题上表现最好。为了获得最佳结果，请使用通俗易懂的语言清楚地解释你想做什么。

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。它需要你回答三个问题才能继续：

- 无论您是想确认它所采取的每一个步骤，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 还可以查看用户在其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，但它不会审查您在问题或它创建的任何拉取请求中给出的任何反馈。
- 是否要允许它更新工作流程文件作为其工作的一部分。您的项目可能已将工作流程配置为在拉取请求事件上开始运行。如果是这样，Amazon Q 创建的任何拉取请求（包括创建或更新工作流程 YAML）都可能启动拉取请求中包含的那些工作流程。作为最佳实践，不要选择允许 Amazon Q 处理工作流程文件，除非您确定您的项目中没有可以自动运行这些工作流程的工作流程，然后再审核和批准它创建的拉取请求。
- 是否允许它建议创建任务，将问题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。允许 Amazon Q 建议和创建任务可以让多人处理问题的不同部分，从而有助于加快复杂问题的开发。它还可以帮助降低理解整个工作的复杂性，因为理想情况下，完成每项任务所需的工作比它所属的问题更简单。
- 你想让它在哪个源代码库中工作。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。不支持链接存储库。

在您做出并确认选择后，Amazon Q 会将问题移至 “进行中” 状态，同时它会尝试根据问题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。它将创建一个固定评论，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通过更新其固定评论并在每个阶段评论该问题的进展来记录其行动。与固定评论和回复不同，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的信息放在置顶评论的顶部。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，该请求将该分支与默认分支合并。当 Amazon Q 完成其工作后，它会将问题移至 “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）区域的议题提供。如果您已将项目配置为使用带有 Jira Software 扩展的 Jira，则该扩展不可用。此外，如果您自定义了看板的布局，则问题可能不会改变状态。为获得最佳效果，请仅在采用标准电路板布局的项目中使用此功能。
>
> 此功能不适用于 Git 子模块。它无法对仓库中包含的任何 Git 子模块进行更改。
>
> 将问题分配给 Amazon Q 后，您就无法更改问题的标题或描述，也不能将其分配给其他任何人。如果您从问题中取消分配 Amazon Q，它将完成当前步骤，然后停止工作。一旦取消指定，它就无法恢复工作或重新分配给问题。

如果用户选择允许问题创建任务，则如果将问题分配给 Amazon Q，则该问题可以自动移至“正在审核”列。但是，“审阅中”中的问题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 应用程序蓝图创建的项目中包含的代码的潜在功能创建三个问题：一个用于添加一个用于创建新的 mysfit 生物，一个用于添加排序功能，另一个用于更新工作流程以包含名为的分支。**`test`**

> **Note**
>
> 如果您在使用不同代码的项目中工作，请使用与该代码库相关的标题和描述来创建问题。

创建问题并生成解决方案供您评估

1. 在导航窗格中，选择“问题”，并确保您处于“看板”视图。
2. 选择“创建问题”。
3. 给问题起一个标题，用通俗易懂的语言解释你想做什么。例如，对于本期，请输入标题**`Create another mysfit named Quokkapus`**。在描述中，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）在问题上附加一张用作 mysfit 缩略图和个人资料图片的图片。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，将本期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。您可以自己将图片添加到网站，然后在创建拉取请求后留言，让 Amazon Q 更新其代码以指向您希望它使用的图片。

在继续下一步之前，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“受托人”中，选择“分配给 Amazon Q”。
6. 在源存储库中，选择包含项目代码的源存储库。
7. 如有必要，请将“要求 Amazon Q 停止”滑动到活动状态，等待对其工作选择器的审核。

> **Note**
>
> 选择在每个步骤后让 Amazon Q 停止，这样您就可以对问题或任何已创建的任务发表评论，从而可以选择让 Amazon Q 根据您的评论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，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 不在等待您的反馈，但您将无法通过发表评论来影响 Amazon Q 所走的方向。如果您选择该选项，Amazon Q 也不会回复拉取请求中留下的评论。

8. 让“允许 Amazon Q 修改工作流程文件”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“创建问题”。您的视图将更改为“问题”面板。
11. 选择创建议题以创建另一个议题，这次是标题为的议题**`Change the get_all_mysfits() API to return mysfits sorted by the Age attribute`**。将此问题分配给 Amazon Q 并创建问题。
12. 选择创建议题以创建另一个议题，这次是标题为的议题**`Update the OnPullRequest workflow to include a branch named test in its triggers`**。（可选）在描述中

链接到工作流程。将此问题分配给 Amazon Q，但这次请确保将 “允许 Amazon Q 修改工作流程文件” 选择器设置为活动状态。创建问题以返回问题板。

> **Tip**
>
> 您可以通过输入 at 符号 (@) 并输入文件名来搜索文件，包括工作流程文件。

创建并分配议题后，问题将移至 “处理中”。Amazon Q 将在固定评论中添加评论，跟踪其在问题中的进展。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” 部分和详细介绍其创建解决方案的建议方法的 “方法” 部分来更新问题的描述。如果 Amazon Q 成功提出问题中描述的问题的解决方案，它将创建一个分支并在该分支中更改代码以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，它将提供一个包含该代码链接的文件，以便您可以查看。代码准备就绪后，它会创建一个拉取请求，以便您可以查看建议的代码更改，添加指向该议题的拉取请求的链接，然后将问题移至 In revie w 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。与任何其他代码更改一样，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 所做的代码更改可能会对您的代码库和基础设施代码产生负面影响。

查看问题和包含 Amazon 所做更改的关联拉取请求 Q

1. 在 “问题” 中，选择分配给 Amazon Q 且正在处理的问题。查看评论以监控 Amazon Q 的进展情况。如果有，请查看背景并与其在问题描述中记录的方法进行比较。如果您选择允许 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，而您想更改顺序或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。查看完问题后，选择 X 关闭问题窗格。

   > **Tip**
   >
   > 要查看任务的进度，请从问题中的任务列表中选择任务。任务不会作为单独的项目显示在图板上，只能通过议题进行访问。如果任务分配给 Amazon Q，则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链接的拉取请求，因为它们不会

作为链接出现在议题中，而只出现在任务中。要从任务中返回议题，请选择指向该议题的链接。

2. 现在，选择分配给 Amazon Q 且正在审核中的问题。查看问题描述中记录的背景和方法。查看评论以了解其执行的操作。查看为与此问题相关的工作创建的所有任务，包括其进度、您可能需要采取的任何操作以及任何评论。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链接拉取请求的形式出现。它们不会以问题的链接拉取请求的形式出现。

3. 在拉取请求中，查看代码更改。有关更多信息，请参阅 查看拉取请求。如果您希望 Amazon Q 更改其任何建议的代码，请在拉取请求上留下评论。为了获得最佳效果，请在 Amazon Q 上发表评论时务必具体说明。

   例如，在查看为创建的拉取请求时**`Create another mysfit named Quokkapus`**，您可能会注意到描述中有一个错字。你可以在 Amazon Q 上留言，上面写着 “通过在 “需求” 和 “a” 之间添加一个空格来更改描述以修复 “needsa” 的错字。” 或者，您可以发表评论，告诉 Amazon Q 更新描述并提供修改后的完整描述以供其纳入。

   如果你将新 mysfit 的图片上传到网站，你可以留言，让 Amazon Q 更新 mysfit，并附上用于新 mysfit 的图像指针和缩略图。

> **Note**
>
> Amazon Q 不会回复个人评论。只有当您在创建问题时选择了在每个步骤后停止审批的默认选项时，Amazon Q 才会将评论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想要更改代码的所有评论后，选择 “创建修订版”，让 Amazon Q 创建拉取请求的修订版，其中包含您在注释中请求的更改。Amazon Q 将在 “概览” 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 关于创建修订版的最新更新。

> **Note**
>
> 只有创建议题的用户才能创建拉取请求的修订版。您只能请求拉取请求的一个修订版。在选择 Create revision 之前，请确保您已经解决了所有评论问题，并且对评论的内容感到满意。

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

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。

- 取消分配 Amazon Q 以解决任何不再处理的问题。如果 Amazon Q 已完成对某个问题的处理或找不到解决方案，请务必取消分配 Amazon Q，以免达到生成人工智能功能的最大配额。有关更多信息，请参阅管理生成式 AI 功能和定价。
- 将所有工作已完成的议题移至 “完成”。
- 如果不再需要该项目，请删除该项目。

教程：使用可组合的 PDK 蓝图创建全栈应用程序

Amazon CodeCatalyst 提供了许多不同的蓝图，以帮助您快速开始项目。使用蓝图创建的项目包括您需要的资源，包括源存储库、示例源代码、CI/CD 工作流程、构建和测试报告以及集成的问题跟踪工具。但是，有时您可能希望逐步构建项目，或者为蓝图创建的现有项目添加功能。你也可以用蓝图来做到这一点。本教程演示了如何开始使用单个蓝图，该蓝图可以奠定基础，并允许您将所有项目代码存储在单个存储库中。然后，您可以方便地在初始蓝图之上应用其他蓝图，从而灵活地整合其他资源和基础架构。通过这种构造块方法，您可以满足多个项目的特定需求。

本教程展示了如何将多个 AWS 项目开发套件 (AWS PDK) 蓝图组合在一起，创建一个包含 React 网站、Smithy API 和支持 CDK 基础设施的应用程序，将其部署到 AWS。AWS PDK 提供了常用模式的构建块以及用于管理和构建项目的开发工具。有关更多信息，请参阅 AWS PDK GitHub 源存储库。

以下 PDK 蓝图旨在相互配合使用，以可组合的方式构建应用程序：

- Monorepo-创建一个根级项目，用于管理 monorepo 中项目之间的相互依赖关系。该项目还提供构建缓存和依赖关系可视化。
- 类型安全 API-创建可以在 Smithy 或 OpenAPI v3 中定义的 API，并管理构建时代码的生成，以允许您以类型安全的方式实现 API 并与之交互。提供一个 CDK 构造，用于管理将 API 部署到 API Gateway 并配置自动输入验证。
- Cloudscape React 网站 ——创建一个基于 React 的网站，使用 Cloudscape 构建，该网站预先集成了 Cognito Auth 和（可选）你创建的 API，这使你能够安全地调用 API。
- 基础架构-创建一个项目，用于设置部署应用程序所需的所有与 CDK 相关的基础架构。它还经过预配置，可以在每次构建时根据您的 CDK 代码生成图表。
- DevOps-创建与 AWS 项目开发套件 (AWS PDK) 中的结构兼容 DevOps 的工作流程。

本教程还包括有关如何查看已部署的应用程序、邀请其他用户使用该应用程序以及如何使用拉取请求对代码进行更改的步骤，拉取请求在合并拉取请求时会自动生成并部署到连接的 AWS 账户中的资源。

当您创建包含 PDK 蓝图的项目时，您的项目是使用项目中的以下资源创建的： CodeCatalyst

- 配置为 monorepo 的源存储库。
- 一种工作流程，它运行静态代码分析和许可证检查，并在对默认分支进行更改时生成和部署示例代码。每次对代码进行更改时，都会生成架构图。
- 议题板和待办事项列表，可用于计划和跟踪工作。
- 带有自动报告的测试报告套件。

主题

先决条件

要创建和更新项目，您必须设置并登录 CodeCatalyst按以下方式完成中的任务：

- 拥有用于登录的 AWS 生成器 ID CodeCatalyst。
- 属于某个空间，并在该空间中为您分配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。
- 拥有与您的空间关联的 AWS 账户，并拥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*** 角色策略的角色策略来创建服务角色。该角色的`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*名称将附加唯一标识符。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。

第 1 步：创建 monorepo 项目

从 PDK-Monorepo 蓝图开始，创建作为基础的 monorepo 代码库，允许你添加额外的 PDK 蓝图。

使用 PDK-Monorepo 蓝图创建项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要在其中创建项目的空间。
3. 在空间控制面板上，选择创建项目。
4. 选择 “从蓝图开始”。
5. 选择 PDK-Monorepo 蓝图，然后选择 “下一步”。
6. 在 “为项目命名” 下，输入要分配给项目的名称及其关联的资源名称。该名称在空间内必须是唯一的。
7. 在 “项目资源” 下，执行以下操作：

    a. 在 “主要编程语言” 下，选择要用来开发项目代码的语言。你可以从 TypeScript Java 或 Python 中进行选择。

    b. 选择代码配置

    c. 在源存储库文本输入字段中，输入将创建新存储库的源存储库的名称，或者从现有的链接存储库中进行选择。现有存储库必须为空。有关更多信息，请参阅 链接源存储库。

    d. （可选）从 Pack age Manager 下拉菜单中，选择软件包管理器。只有在您选择 TypeScript 作为主要编程语言时，才需要这样做。

8. （可选）要预览将根据您选择的项目参数生成的代码，请从 “生成项目预览” 中选择 “查看代码”。
9. （可选）从蓝图的卡片中选择查看详细信息以查看有关蓝图的特定详细信息，例如蓝图架构概述、所需的连接和权限以及蓝图创建的资源类型。
10. 选择 “创建项目” 来创建您的 monorepo 项目。创建的根级项目管理 monorepo 中项目之间的相互依赖关系，并提供构建缓存和依赖项管理。

    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。

PDK-Monorepo 蓝图仅为项目奠定基础。要使用蓝图创建可行的应用程序，你需要添加其他 PDK 蓝图，例如 Type Safe API、Cloudscape React 网站、基础设施或。 DevOps在下一步中，您将对项目应用类型安全 API。

第 2 步：向项目添加类型安全 API

PDK-Type Safe API 蓝图允许你使用 Smithy 或 OpenAI v3 定义 API。它根据您的 API 定义生成运行时包，其中包括用于与您的 API 交互的客户端和用于实现 API 的服务器端代码。该蓝图还会为每个 API 操作生成一个具有类型安全性的 CDK 构造。您可以将蓝图应用于现有的 PDK monorepo 项目，为该项目添加 API 功能。

要应用 PDK-Type Safe API 蓝图

1. 在 monorepo 项目的导航窗格中，选择蓝图，然后选择应用蓝图。
2. 选择 PDK-Type Safe API 蓝图，然后选择 “下一步”。
3. 在配置蓝图下，配置蓝图参数：
   - 在模型语言下，选择定义 API 模型所用的语言。
   - 在命名空间文本输入字段中，输入您的 API 的命名空间。
   - 在 API 名称文本输入字段中，输入 API 的名称。
   - 在 CDK 语言下，选择您首选的语言来编写 CDK 基础架构以部署 API。
   - 选择 Handler 语言下拉菜单，然后选择要实现用于 API 操作的处理程序的语言。
   - 选择文档格式下拉菜单，然后选择生成 API 文档所需的格式。
4. 在 “代码更改” 选项卡中，查看建议的更改。拉取请求中显示的差异显示了创建拉取请求时您的项目所做的更改。
5. 如果您对应用蓝图时将要进行的拟议更改感到满意，请选择应用蓝图。

创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用@符号和文件名来添加指向资源（例如文件）的链接。

> **Note**
>
> 在拉取请求获得批准并合并之前，蓝图才会被应用。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

6. 从 “状态” 列中，选择 PDK-Type Safe API 蓝图行的待处理拉取请求，然后选择已打开的拉取请求的链接。
7. 选择 “合并”，选择您的首选合并策略，然后选择 “合并” 以合并应用蓝图中的更改。

   合并拉取请求后，将在您的 monorepo 项目中生成一个新`packages/apis/mypdkapi`文件夹，其中包含您配置的 Type Safe API 的所有与 API 相关的源代码。
8. 在导航窗格中，选择 Bluepr ints 以确认 PDK 的状态-Type Safe API 显示为最新。

第 3 步：为项目添加 Cloudscape React 网站

PDK-Cloudscape React 网站蓝图生成一个网站。您可以关联可选参数（Type Safe API）来自动配置您的网站以设置经过身份验证的类型安全客户端，还可以关联交互式 API 资源管理器来测试您的各种 API。

应用 PDK-Cloudscape React 网站蓝图

1. 在 monorepo 项目的导航窗格中，选择蓝图，然后选择应用蓝图。
2. 选择 PDK-Cloudscape React 网站蓝图，然后选择 “下一步”。
3. 在配置蓝图下，配置蓝图参数：

   - 在网站名称文本输入字段中，输入您的网站名称。
   - 选择 Type Safe API 下拉菜单，然后选择要集成到网站中的 API 蓝图。传入 API 会设置经过身份验证的客户端，并添加所需的依赖项、API 资源管理器和其他功能。
4. 在 “代码更改” 选项卡中，查看建议的更改。拉取请求中显示的差异显示了创建拉取请求时您的项目所做的更改。
5. 如果您对应用蓝图时将要进行的拟议更改感到满意，请选择应用蓝图。

创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用@符号和文件名来添加指向资源（例如文件）的链接。

> **Note**
>
> 在拉取请求获得批准并合并之前，蓝图才会被应用。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

6. 从 “状态” 列中，选择 PDK-Cloudscape React Webs ite 蓝图行的待处理拉取请求，然后选择已打开的拉取请求的链接。
7. 选择 “合并”，选择您的首选合并策略，然后选择 “合并” 以合并应用蓝图中的更改。

   合并拉取请求后，将在您的 monorepo 项目中生成一个新`packages/websites/`*`my-website-name`*文件夹，其中包含新网站的所有源代码。
8. 在导航窗格中，选择蓝图以确认 PDK 的状态 ——Cloudscape React 网站显示为最新。

接下来，您将应用 PDK-基础设施蓝图来生成基础设施，以便将您的网站部署到 AWS 云中。

步骤 4：生成基础设施以将应用程序部署到 AWS 云中

PDK-基础架构蓝图会设置一个包含所有 CDK 代码的软件包，用于部署您的网站和 API。默认情况下，它还提供图表生成和与原型设计工具包的一致性。

应用 PDK-基础设施蓝图

1. 在 monorepo 项目的导航窗格中，选择蓝图，然后选择应用蓝图。
2. 选择 PDK-基础架构蓝图，然后选择 “下一步”。
3. 在配置蓝图下，配置蓝图参数：

   - 在 CDK 语言下，选择要用来开发基础架构的语言。
   - 在堆栈名称文本输入字段中，输入为蓝图生成的 CloudFormation 堆栈的名称。

     > **Note**
     >
     > 请记下此堆栈名称，以便在下一步中设置 DevOps 工作流程。

   - 选择 Type Safe API 下拉菜单，然后选择要集成到网站中的 API 蓝图。

- 选择 Cloudscape React TS 网站下拉菜单，然后选择要在基础架构中部署的网站蓝图（例如，PDK-Clou dscape React 网站）。

4. 在 “代码更改” 选项卡中，查看建议的更改。拉取请求中显示的差异显示了创建拉取请求时您的项目所做的更改。
5. 如果您对应用蓝图时将要进行的拟议更改感到满意，请选择应用蓝图。

   创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用@符号和文件名来添加指向资源（例如文件）的链接。

   > **Note**
   >
   > 在拉取请求获得批准并合并之前，蓝图才会被应用。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

6. 从 “状态” 列中，选择 PDK-基础架构蓝图行的待处理拉取请求，然后选择已打开的拉取请求的链接。
7. 选择 “合并”，选择您的首选合并策略，然后选择 “合并” 以合并应用蓝图中的更改。

   合并拉取请求后，将在您的 monorepo 项目中生成一个新`packages/infra`文件夹，其中包含将您的项目部署到 AWS 云中的基础设施。

8. 在导航窗格中，选择 Blue prints 以确认 PDK 的状态-基础架构显示为最新。

接下来，您将应用 PDK- DevOps 蓝图来部署您的应用程序。

第 5 步：设置部署项目 DevOps 的工作流程

PDK- bluep DevOps rint 使用配置中指定的 AWS 账户和角色生成构建和部署项目所需 DevOps 的工作流程。

要应用 PDK-蓝图 DevOps

1. 在 monorepo 项目的导航窗格中，选择蓝图，然后选择应用蓝图。
2. 选择 PDK- DevOps 蓝图，然后选择 “下一步”。
3. 在配置蓝图下，配置蓝图参数：

   - 在当前环境中选择 Bootstrap CDK。

- 在堆栈名称文本输入字段中，输入要部署的 CloudFormation 堆栈的名称。这应与在 PDK-基础架构蓝图中配置步骤 4：生成基础设施以将应用程序部署到 AWS 云中的堆栈名称相匹配。
- 选择 AWS 账户连接下拉菜单，然后选择要用于资源的 AWS 账户。有关更多信息，请参阅 向 AWS 账户 空间添加。
- 选择用于部署应用程序的角色下拉菜单，然后选择要用于部署项目应用程序的 IAM 角色。

> **Note**
>
> 创建 IAM 角色时，`SourceArn`请将限制为在项目设置中`ProjectID`找到的当前角色。有关更多信息，请参阅 了解CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色。

- 选择 “区域” 下拉菜单，然后选择要部署 monorepo 项目的区域。部署仅适用于存在所需 AWS 服务的区域。有关更多信息，请参阅按区域划分的 AWS 服务。

4. 在 “代码更改” 选项卡中，查看建议的更改。拉取请求中显示的差异显示了创建拉取请求时您的项目所做的更改。
5. 如果您对应用蓝图时将要进行的拟议更改感到满意，请选择应用蓝图。

   创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用@符号和文件名来添加指向资源（例如文件）的链接。

   > **Note**
   >
   > 在拉取请求获得批准并合并之前，蓝图才会被应用。有关更多信息，请参阅 查看拉取请求和 合并拉取请求。

6. 从 “状态” 列中，选择 PDK-基础架构蓝图行的待处理拉取请求，然后选择已打开的拉取请求的链接。
7. 选择 “合并”，选择您的首选合并策略，然后选择 “合并” 以合并应用蓝图中的更改。

   合并拉取请求后，将在您的 monorepo 项目中生成一个新`.codecatalyst/workflows`文件夹。
8. 在导航窗格中，选择 Blue prints 以确认 PDK 的状态- DevOps 显示为最新。

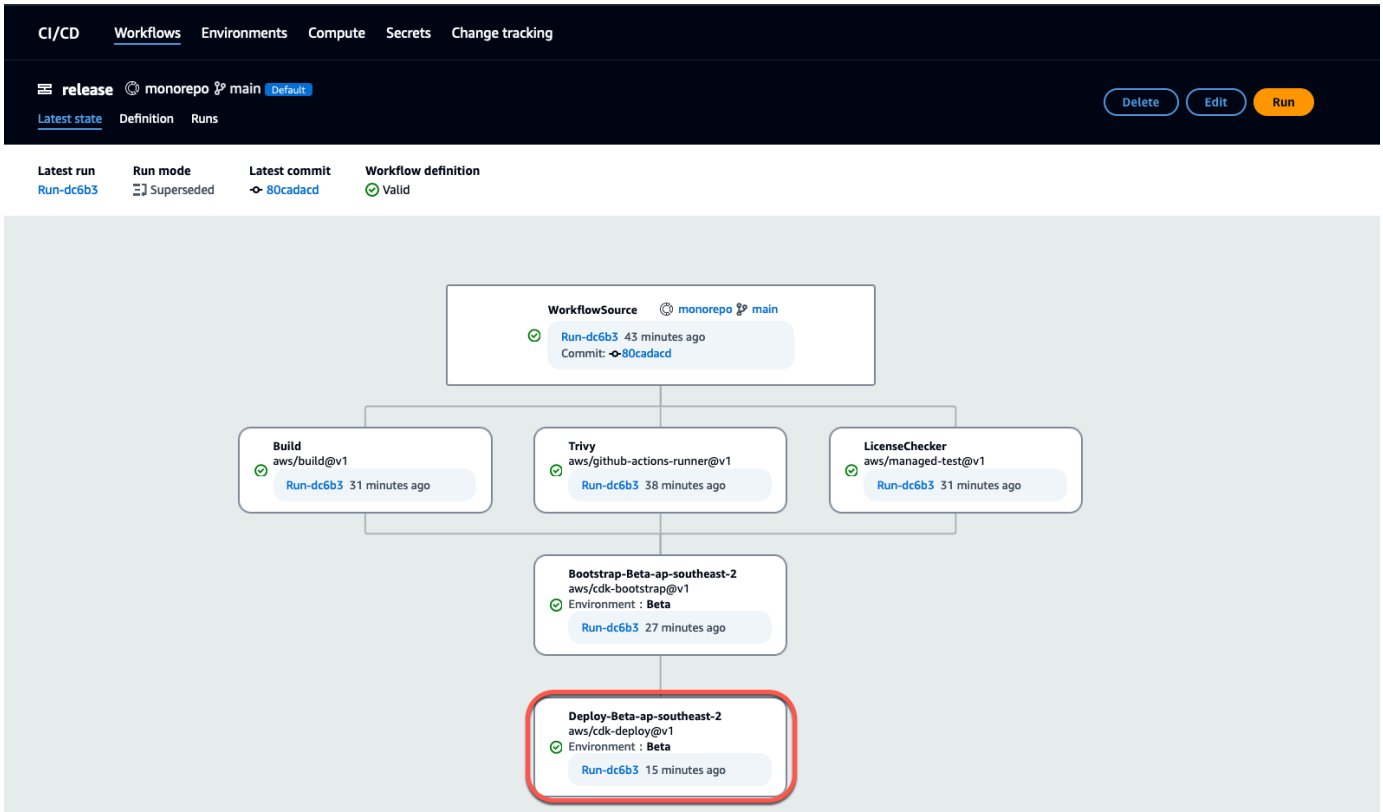
> **Note**
>
> 从那时起，PDK- DevOps 蓝图和所有后续对PDK蓝图的更改都将显著减慢，因为在幕后生成锁定文件是为了确保将来可以重复构建和部署。它将为所有支持语言的软件包生成锁定文件。

第 6 步：确认发布工作流程并查看您的网站

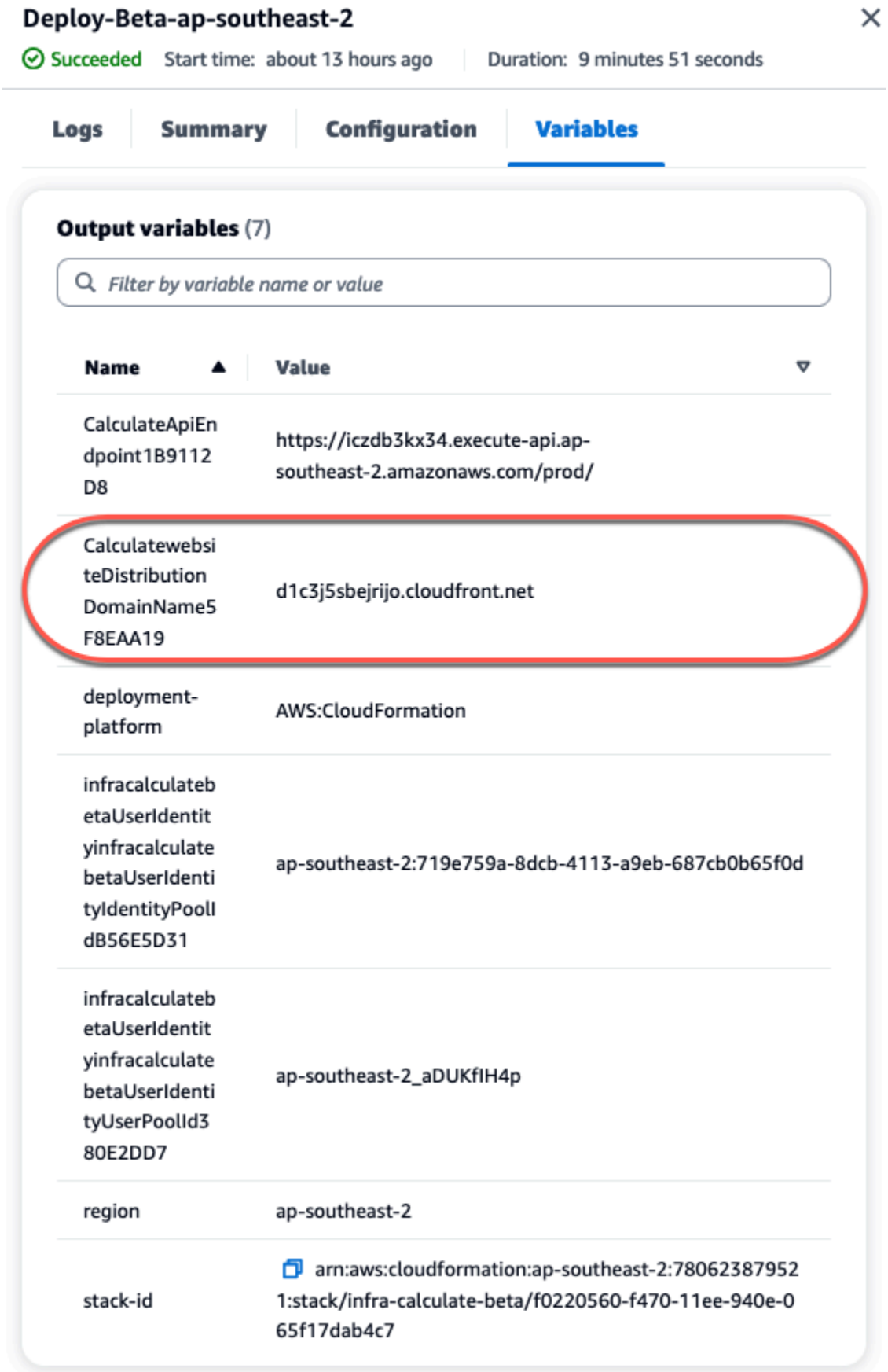
完成前面的步骤后，您可以确认发布工作流程以确保项目正在构建中。

确认发布工作流程并查看您的网站

1. 在 monorepo 项目的导航窗格中，选择 C I/CD，然后选择工作流程。
2. 对于发布工作流程，请选择最新运行的工作流程以查看详细信息。有关更多信息，请参阅 查看单次运行的状态和详细信息。
3. 成功完成工作流程运行后，选择工作流程中的最后一个操作（例如 Deplo y-B-eta-ap-souteast 2），然后选择变量。



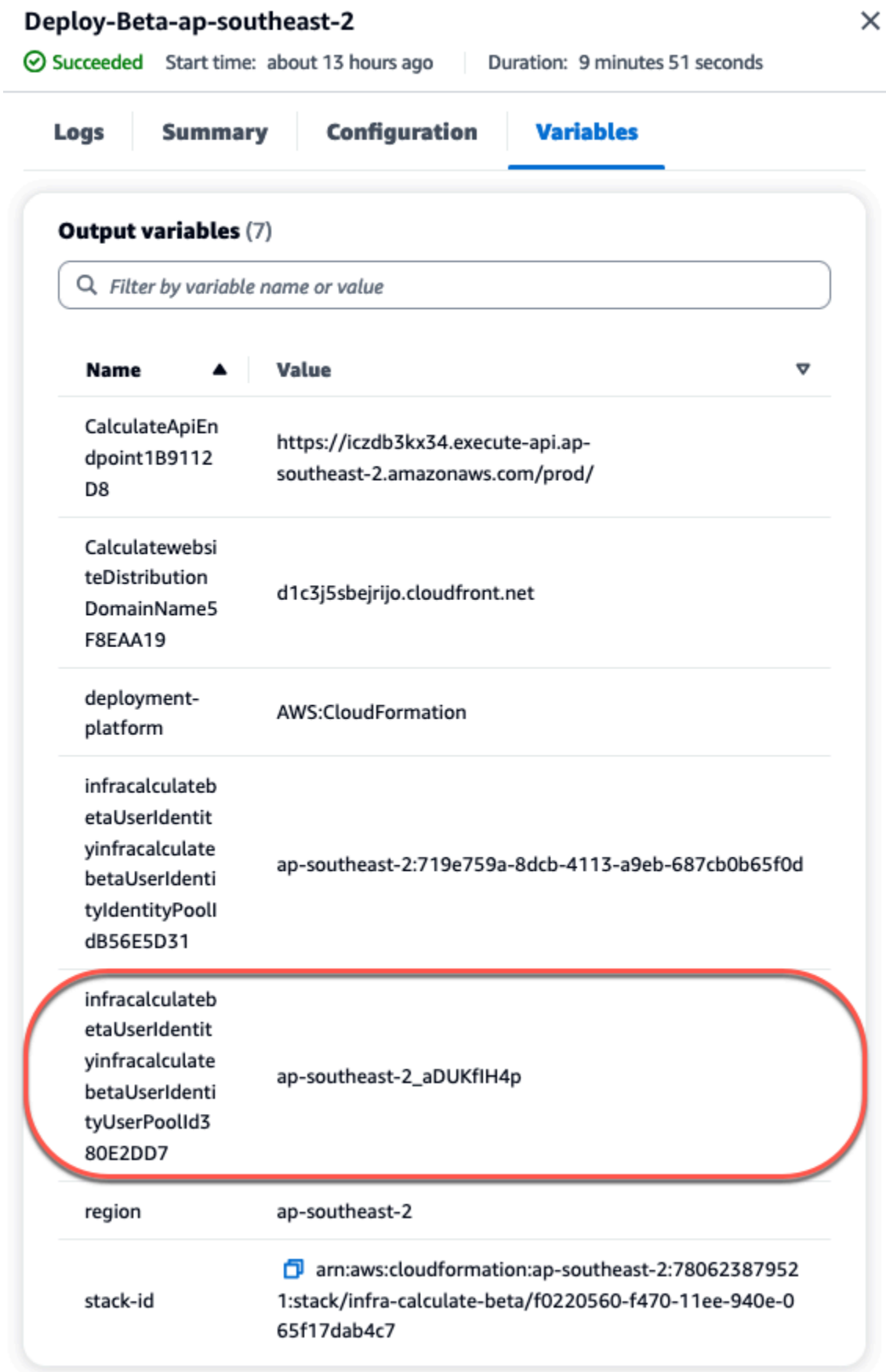
4. 将变量表中的链接（例如 ***mypdkApi websiteDistributionDomain namexXXXX########## ######***，即可查看已部署的网站。


Deploy-Beta-ap-southeast-2
Succeeded
Start time: about 13 hours ago
Duration: 9 minutes 51 seconds
Logs
Summary
Configuration
Variables
Output variables (7)
Filter by variable name or value
Name
Value
CalculateApiEndpoint1B9112D8
https://iczdb3kx34.execute-api.ap-southeast-2.amazonaws.com/prod/
CalculatewebsiteDistributionDomainName5F8EAA19
d1c3j5sbejrijo.cloudfront.net
deployment-platform
AWS:CloudFormation
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityIdentityPooldB56E5D31
ap-southeast-2:719e759a-8dcb-4113-a9eb-687cb0b65f0d
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityUserPoolId380E2DD7
ap-southeast-2_aDUKfIH4p
region
ap-southeast-2
stack-id
arn:aws:cloudformation:ap-southeast-2:780623879521:stack/infra-calculate-beta/f0220560-f470-11ee-940e-065f17dab4c7

您需要一个 Amazon Cognito 账户才能登录您的网站。默认情况下，用户池未设置为允许自行注册。

a. 导航到 AWS Cognito 控制台。

b. 从 “用户池” 表中，选择与 PDK 创建的用户池相匹配的用户池名称 —— DevOps 蓝图，该名称可在变量表中找到（例如，in fra calc ***ul*** ate calculate XXXXX）betaUserIdentityinfra***#*** betaUserIdentity IdentityPoolId有关更多信息，请参阅用户池入门。


Deploy-Beta-ap-southeast-2
Succeeded
Start time: about 13 hours ago
Duration: 9 minutes 51 seconds
Logs
Summary
Configuration
Variables
Output variables (7)
Filter by variable name or value
Name
Value
CalculateApiEndpoint1B9112D8
https://iczdb3kx34.execute-api.ap-southeast-2.amazonaws.com/prod/
CalculatewebsiteDistributionDomainName5F8EAA19
d1c3j5sbejrijo.cloudfront.net
deployment-platform
AWS:CloudFormation
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityIdentityPoolIdB56E5D31
ap-southeast-2:719e759a-8dcb-4113-a9eb-687cb0b65f0d
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityUserPoolId380E2DD7
ap-southeast-2_aDUKfIH4p
region
ap-southeast-2
stack-id
arn:aws:cloudformation:ap-southeast-2:780623879521:stack/infra-calculate-beta/f0220560-f470-11ee-940e-065f17dab4c7

c. 选择 创建用户。

d. 配置用户信息参数：

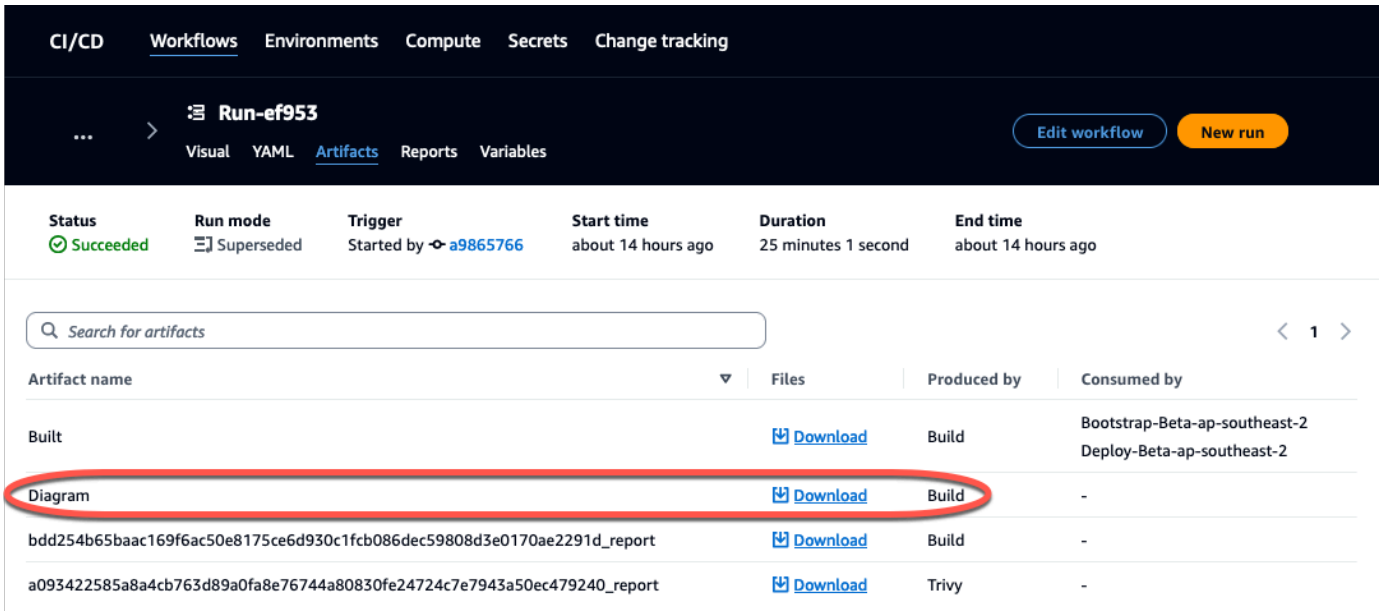
- 在 “邀请消息” 下，选择 “发送电子邮件邀请”。
- 在用户名文本输入字段中，输入用户名。
- 在电子邮件地址文本输入字段中，输入用户名。
- 在 “临时密码” 下，选择 “生成密码”。

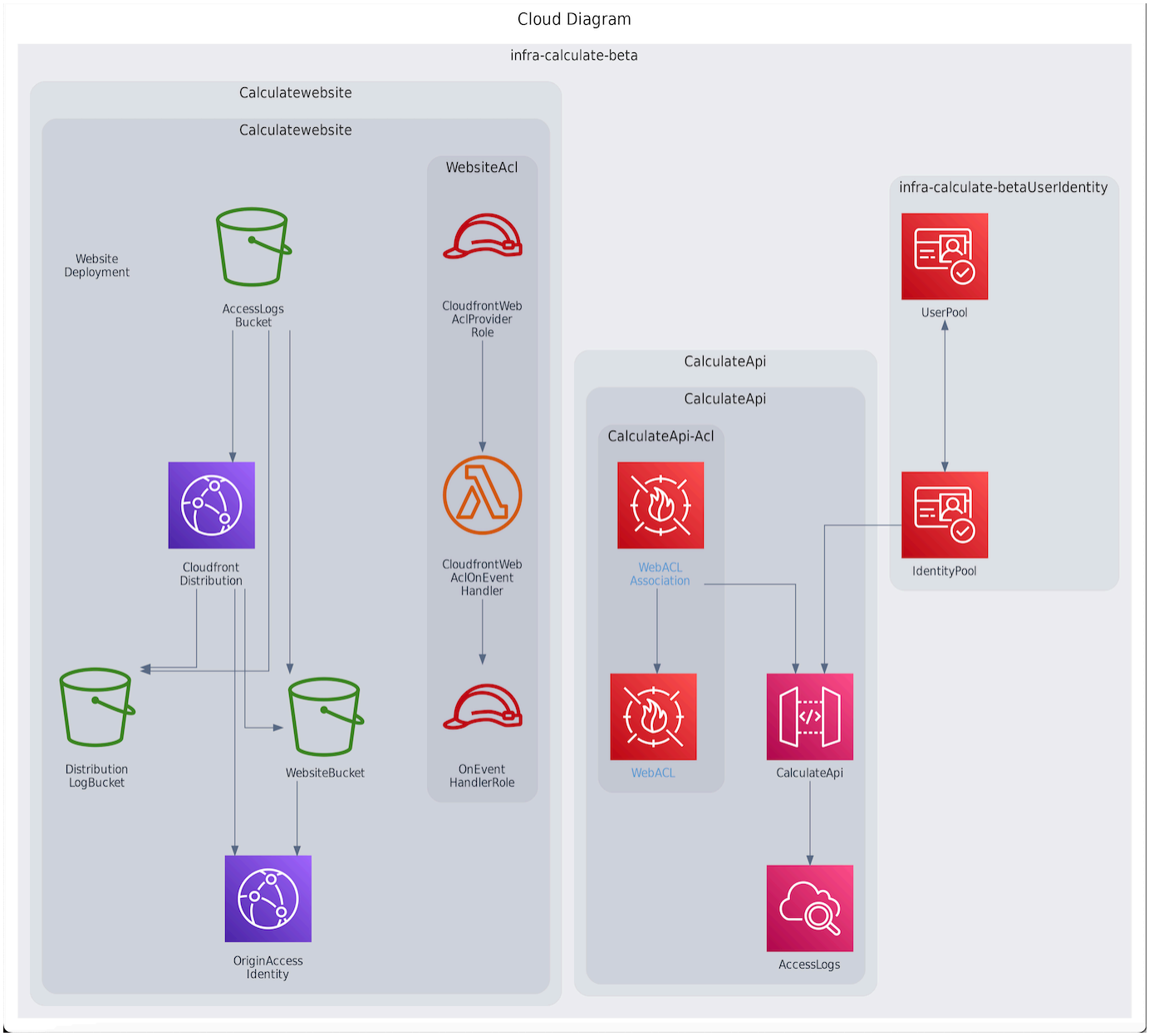
e. 选择 创建用户。

f. 导航到您为用户信息参数输入的电子邮件帐户，使用临时密码打开一封电子邮件。记下密码。

g. 返回已部署的网站，输入您创建的用户名和收到的临时密码，然后选择登录。

5. （可选）成功完成工作流程运行后，您还可以查看生成的逻辑示意图。选择中的 “构件” 选项卡 CodeCatalyst，在 “关系图” 行中选择 “下载”，然后打开下载的文件。





在 PDK 项目上进行协作和迭代

项目设置完成后，您可以对源代码进行更改。您也可以邀请其他空间成员参与该项目。PDK 蓝图允许您以迭代方式构建应用程序，仅在需要时添加所需的内容，同时保留对每个蓝图配置的完全控制。

主题

- 第 1 步：邀请成员加入您的项目
- 第 2 步：创建议题以进行协作和跟踪工作
- 第 3 步：查看您的源代码库

- 步骤 4：创建开发环境并更改代码
- 第 5 步：推送和合并代码更改

第 1 步：邀请成员加入您的项目

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在空间之外添加姓名。

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，因为他们已经隐式访问空间中的所有项目。

当您邀请用户加入您的项目（未分配空间管理员角色）时，该用户将显示在项目成员表格中的项目下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。

第 2 步：创建议题以进行协作和跟踪工作

CodeCatalyst 帮助您跟踪项目中涉及的功能、任务、错误以及任何其他有问题的工作。您可以创建议题来跟踪所需的工作和想法。默认情况下，当您创建议题时，它会添加到待办事项中。您可以将问题移至图板，在那里您可以跟踪正在进行的工作。您也可以将议题分配给特定的项目成员。在此步骤中，创建一个议题以更改您的 PDK 项目。

创建问题

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到你要在其中创建议题的 monorepo 项目。
3. 在项目主页上，选择创建问题。或者，在导航窗格中选择 “问题”。
4. 选择创建问题。

> **Note**
>
> 使用网格视图时，您也可以内联添加问题。

5. 输入问题的标题。
6. （可选）输入描述。对于此问题，请输入以下描述：`a change in the src/mysfit_data.json file.`。你可以使用 Markdown 来添加格式。
7. （可选）选择问题的状态、优先级和估计。
8. （可选）添加现有标签或创建新标签并通过选择 + 添加标签进行添加。
   a. 要添加现有标签，请从列表中选择标签。您可以在字段中输入搜索词，以搜索项目中包含该术语的所有标签。
   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。
9. （可选）通过选择 + 添加工作负责人来添加工作负责人。您可以通过选择 + 添加我来快速将自己添加为受托人。

> **Tip**
>
> 您可以选择将问题分配给 Amazon Q，让 Amazon Q 尝试解决问题。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。

此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

10. （可选）添加现有的自定义字段或创建新的自定义字段。议题可能有多个自定义字段。

    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搜索词，以搜索项目中包含该术语的所有自定义字段。

    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。

11. 选择创建问题。右下角会显示一条通知：如果问题已成功创建，则会显示一条确认消息，说明问题已成功创建。如果问题未成功创建，则会显示一条包含失败原因的错误消息。然后，您可以选择“重试”进行编辑并重试创建问题，或者选择“丢弃”放弃问题。这两个选项都将关闭通知。

> **Note**
>
> 在创建议题时，您无法将拉取请求与议题相关联。但是，您可以在创建拉取请求后对其进行编辑，以添加拉取请求的链接。

有关更多信息，请参阅 跟踪和组织处理问题的工作 CodeCatalyst。

第 3 步：查看您的源代码库

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
- 仓库的未处理拉取请求数量
- 存储库的相关工作流程数量
- 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
- 在 Markdown 中呈现的 README.md 文件的内容（如果包含任何 README.md 文件）

该页面还提供指向仓库提交、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。要查看有关链接存储库的信息，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，然后执行以下任一操作：
   - 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存储库名称的一部分来筛选存储库列表。
2. 在存储库的主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求的数量和工作流程。默认情况下，会显示默认分支的内容。您可以通过从下拉列表中选择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。

步骤 4：创建开发环境并更改代码

在此步骤中，创建一个开发环境并进行代码更改，然后将其合并到主分支中。虽然本教程将引导您完成一个简单的 AWS PDK 项目，但您也可以参照 A WS PDK GitHub 存储库中提供的更复杂的示例进行操作。

使用新分支创建开发环境

1. 在 monorepo 项目的导航窗格中，执行以下任一操作：
   - 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”，选择 “源存储库”，然后选择要为其创建开发环境的 monorepo 存储库。
2. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成开发环境了解更多信息。
3. 选择克隆存储库。
4. 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。开发环境将从您从源存储库页面中选择的源存储库创建。

5. （可选）在“别名-可选”中，输入开发环境的别名。
6. （可选）选择开发环境配置编辑按钮以编辑开发环境的计算、存储或超时配置。
7. （可选）在 Amazon Virtual Private Cloud（亚马逊 VPC）（可选）中，从下拉菜单中选择要与开发环境关联的 VPC 连接。

   如果您的空间设置了默认 VPC，则您的开发环境将连接到该 VPC 运行。您可以通过关联不同的 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。

   > **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。

8. 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。

开发环境运行后，您可以在中使用生成的示例应用程序， CodeCatalyst 方法是使用拉取请求对代码进行更改，拉取请求将在合并拉取请求时自动生成并部署到连接的 AWS 账户中的资源。monorepo 出售开发文件，因此所有必需的全局依赖项和运行时都会自动出现。

更改项目中的代码

1. 在开发环境的工作终端中，导航到您的 monorepo 项目，然后通过运行以下命令安装项目依赖项：

   ```
   npx projen install
   ```

2. 导航到`packages/apis/mypdkapi/model/src/main/smithy/operations/say-hello.smithy`，其中定义了 API 操作示例。在本教程中，您将构建一个将两个数字相加在一起的简单`Calculate`操作。更改代码以定义此操作，包括其输入和输出。

   示例：

```
$version: "2"
namespace com.aws

@http(method: "POST", uri: "/calculate")
@handler(language: "typescript")
operation Calculate {
    input := {
        @required
        numberA: Integer
        @required
        numberB: Integer
    }
    output := {
        @required
        result: Integer
    }
}
```

@handler该特征告诉类型安全 API，您将以写入的 AWS Lambda 处理程序的形式实现此操作。TypeScriptType Safe API 将为此操作生成一个存根供你在中实现。 TypeScript该@required特征已添加，这意味着它将在运行时由部署的 API 网关强制执行。有关更多信息，请参阅 Smithy 文档。

3. 使用与您的代码更改一致/say-hello.smithy的文件名来重命名（例如，calculate.smithy）。
4. 导航到packages/apis/*mypdkapi*/model/src/main/smithy/main.smithy，然后对代码进行更改以连接操作。您可以/calculate.smithy通过在此文件的operations字段中列出中定义的Calculate操作来公开该操作。

   示例：

```
$version: "2"
namespace com.aws

use aws.protocols#restJson1

/// A sample smithy api
@restJson1
service MyPDKApi {
    version: "1.0"
    operations: [Calculate]
```

```
    errors: [
      BadRequestError
      NotAuthorizedError
      InternalFailureError
    ]
}
```

5. 通过运行以下命令来生成更改：

```
npx projen build
```

> **Note**
>
> 或者，您可以传入--parallel X标志，它将在X内核之间分配构建。

由于添加了@handler特征，因此将在构建完成后生成以下文件：

- /packages/apis/*mypdkapi*/handlers/typescript/src/*calculate.ts*
- /packages/apis/*mypdkapi*/handlers/typescript/test/*calculate.test.ts*

6. 导航到packages/apis/*mypdkapi*/handlers/typescript/src/*calculate.ts*，然后对代码进行更改。此文件是为 API 调用的服务器处理程序。

```
import {
  calculateHandler,
  CalculateChainedHandlerFunction,
  INTERCEPTORS,
  Response,
  LoggingInterceptor,
} from 'mypdkapi-typescript-runtime';

/**
 * Type-safe handler for the Calculate operation
 */
export const calculate: CalculateChainedHandlerFunction = async (request) => {
  LoggingInterceptor.getLogger(request).info('Start Calculate Operation');

  const { input } = request;

  return Response.success({
    result: input.body.numberA + input.body.numberB,
```

```
  });
};

/**
 * Entry point for the AWS Lambda handler for the Calculate operation.
 * The calculateHandler method wraps the type-safe handler and manages marshalling
 inputs and outputs
 */
export const handler = calculateHandler(...INTERCEPTORS, calculate);
```

7. 导航到该/packages/apis/*mypdkapi*/handlers/typescript/test/*calculate.test.ts*文件，然后对代码进行更改以更新单元测试。

   示例：

```
import {
  CalculateChainedRequestInput,
  CalculateResponseContent,
} from 'mypdkapi-typescript-runtime';
import {
  calculate,
} from '../src/calculate';

// Common request arguments
const requestArguments = {
  chain: undefined as never,
  event: {} as any,
  context: {} as any,
  interceptorContext: {
    logger: {
      info: jest.fn(),
    },
  },
} satisfies Omit<CalculateChainedRequestInput, 'input'>;

describe('Calculate', () => {

  it('should return correct sum', async () => {
    const response = await calculate({
      ...requestArguments,
      input: {
        requestParameters: {},
        body: {
```

```
          numberA: 1,
          numberB: 2
        }
      },
    });

    expect(response.statusCode).toBe(200);
    expect((response.body as CalculateResponseContent).result).toEqual(3);
  });

});
```

8. 导航到该/packages/infra/main/src/constructs/apis/*mypdkapi.ts*文件，然后对代码进行更改，以便在 CDK 基础架构中为该Calculate操作添加集成。API 构造具有集成属性，您可以在其中传入之前添加的实现。由于您使用的是 Smithy 模型中的@handler特征进行Calculate操作，因此您可以使用预先配置的生成的 CalculateFunction CDK 构造来指向您的处理程序实现。

   示例：

```
import { UserIdentity } from "@aws/pdk/identity";
import { Authorizers, Integrations } from "@aws/pdk/type-safe-api";
import { Stack } from "aws-cdk-lib";
import { Cors } from "aws-cdk-lib/aws-apigateway";
import {
  AccountPrincipal,
  AnyPrincipal,
  Effect,
  PolicyDocument,
  PolicyStatement,
} from "aws-cdk-lib/aws-iam";
import { Construct } from "constructs";
import { Api, CalculateFunction } from "calculateapi-typescript-infra";

/**
 * Api construct props.
 */
export interface CalculateApiProps {
  /**
   * Instance of the UserIdentity.
   */
  readonly userIdentity: UserIdentity;
}
```

```
/**
 * Infrastructure construct to deploy a Type Safe API.
 */
export class CalculateApi extends Construct {
  /**
   * API instance
   */
  public readonly api: Api;

  constructor(scope: Construct, id: string, props?: CalculateApiProps) {
    super(scope, id);

    this.api = new Api(this, id, {
      defaultAuthorizer: Authorizers.iam(),
      corsOptions: {
        allowOrigins: Cors.ALL_ORIGINS,
        allowMethods: Cors.ALL_METHODS,
      },
      integrations: {
        calculate: {
          integration: Integrations.lambda(new CalculateFunction(this,
 "CalculateFunction"))
        }
      },
      policy: new PolicyDocument({
        statements: [
          // Here we grant any AWS credentials from the account that the prototype
 is deployed in to call the api.
          // Machine to machine fine-grained access can be defined here using more
 specific principals (eg roles or
          // users) and resources (ie which api paths may be invoked by which
 principal) if required.
          // If doing so, the cognito identity pool authenticated role must still
 be granted access for cognito users to
          // still be granted access to the API.
          new PolicyStatement({
            effect: Effect.ALLOW,
            principals: [new AccountPrincipal(Stack.of(this).account)],
            actions: ["execute-api:Invoke"],
            resources: ["execute-api:/*"],
          }),
          // Open up OPTIONS to allow browsers to make unauthenticated preflight
 requests
```

```
          new PolicyStatement({
            effect: Effect.ALLOW,
            principals: [new AnyPrincipal()],
            actions: ["execute-api:Invoke"],
            resources: ["execute-api:/*/OPTIONS/*"],
          }),
        ],
      }),
    });

    // Grant authenticated users access to invoke the api
    props?.userIdentity.identityPool.authenticatedRole.addToPrincipalPolicy(
      new PolicyStatement({
        effect: Effect.ALLOW,
        actions: ["execute-api:Invoke"],
        resources: [this.api.api.arnForExecuteApi("*", "/*", "*")],
      }),
    );
  }
}
```

9. 通过运行以下命令来生成更改：

```
npx projen build
```

项目完成构建后，您可以查看更新的生成的逻辑示意图，该图可在中找到/packages/infra/main/cdk.out/cdkgraph/diagram.png。该图显示了如何添加该函数并将其连接到创建的 API。随着 CDK 代码的修改，此图表也随之更新。

现在，您可以通过将更改推送并合并到仓库的主分支来部署更改。

第 5 步：推送和合并代码更改

提交并推送您的代码更改，然后可以将其合并到源存储库的主分支中。

将更改推送到您的功能分支

- 通过运行以下命令将更改提交并推送到您的功能分支：

```
git add .
```

```
git commit -m "my commit message"
```

```
git push
```

推送更改会触发功能分支的新工作流程运行，您可以在 CodeCatalyst 控制台中查看该工作流程。然后，您可以创建一个拉取请求，将更改合并到源存储库的主分支中。将功能分支合并到主分支会触发发布工作流程。您也可以将拉取请求链接到您的问题。

创建拉取请求并将其链接到您的议题

1. 在你的 monorepo 项目中，执行以下任一操作：
   - 在导航窗格中，选择代码，选择拉取请求，然后选择创建拉取请求。
   - 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。
2. 在源代码库中，确保指定的源存储库是包含已提交代码的存储库。只有当你没有从仓库的主页创建拉取请求时，此选项才会出现。
3. 在 D est ination 分支中，在查看代码后，选择要将代码合并到的主分支。
4. 在源分支中，选择包含已提交代码的功能分支。
5. 在 Pull req uest 标题中，输入一个标题，以帮助其他用户了解需要审阅的内容及其原因。
6. （可选）在拉取请求描述中，提供诸如问题链接或更改描述之类的信息。

> **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。将自动生成的描述添加到拉取请求后，您可以对其进行更改。
> 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅在 A mazon 中管理生成式 AI 功能 CodeCatalyst。

7. 在 “问题” 中，选择 “关联问题”，然后选择您在中创建的议题第 2 步：创建议题以进行协作和跟踪工作。要取消议题的链接，请选择取消链接图标。
8. （可选）在必填审稿人中，选择添加所需的审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，必需的审阅者必须批准更改。

> **Note**
>
> 您不能将审阅者同时添加为必填审阅者和可选审阅者。您无法将自己添加为审阅者。

9. （可选）在可选审阅者中，选择添加可选审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，可选的审阅者不必将更改作为一项要求进行批准。
10. 您的拉取请求必须由审阅者或您自己审阅并合并到主分支中。有关更多信息，请参阅 合并拉取请求。

    当您的更改合并到源存储库的主分支时，会自动触发新的工作流程。
11. 合并完成后，您可以将问题移至 “完成”。

    a. 在导航窗格中，选择 “问题”。

    b. 选择在中创建的问题第 2 步：创建议题以进行协作和跟踪工作，选择 “状态” 下拉列表，然后选择 “完成”。

发布工作流程将在成功运行后部署您的应用程序，因此您可以查看更改。

确认发布工作流程并查看您的网站

1. 在 monorepo 项目的导航窗格中，选择 C I/CD，然后选择工作流程。
2. 对于发布工作流程，请选择最新运行的工作流程以查看详细信息。有关更多信息，请参阅 查看单次运行的状态和详细信息。
3. 成功完成工作流程运行后，选择工作流程中的最后一个操作（Deplo y-B-eta-ap-souteast 2），然后选择变量。
4. 通过将链接从 ***mypdkApi*** websiteDistributionDomain namexxxxx 行复制并粘贴到新的浏览器窗口来查看已部署的网站。
5. 输入您在中创建的用户名和密码第 6 步：确认发布工作流程并查看您的网站，然后选择 “登录”。
6. （可选）测试应用程序中的更改。

    a. 选择 P OST 下拉菜单。

    b. 为`numberA`和输入两个值`number B`，然后选择执行。

    c. 在响应正文中确认结果。

随着时间的推移，PDK 蓝图的目录版本可能会发生变化。您可以将项目的蓝图更改为目录版本，以了解最新的更改。在更改项目的蓝图版本之前，您可以查看代码更改和受影响的环境。有关更多信息，请参阅 更改项目中的蓝图版本。

使用空格整理资源 CodeCatalyst

您可以创建一个代表您、您的公司、部门或群体的空间，并提供开发团队可以管理项目的地方。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。您不能重复使用已删除空间的名称。

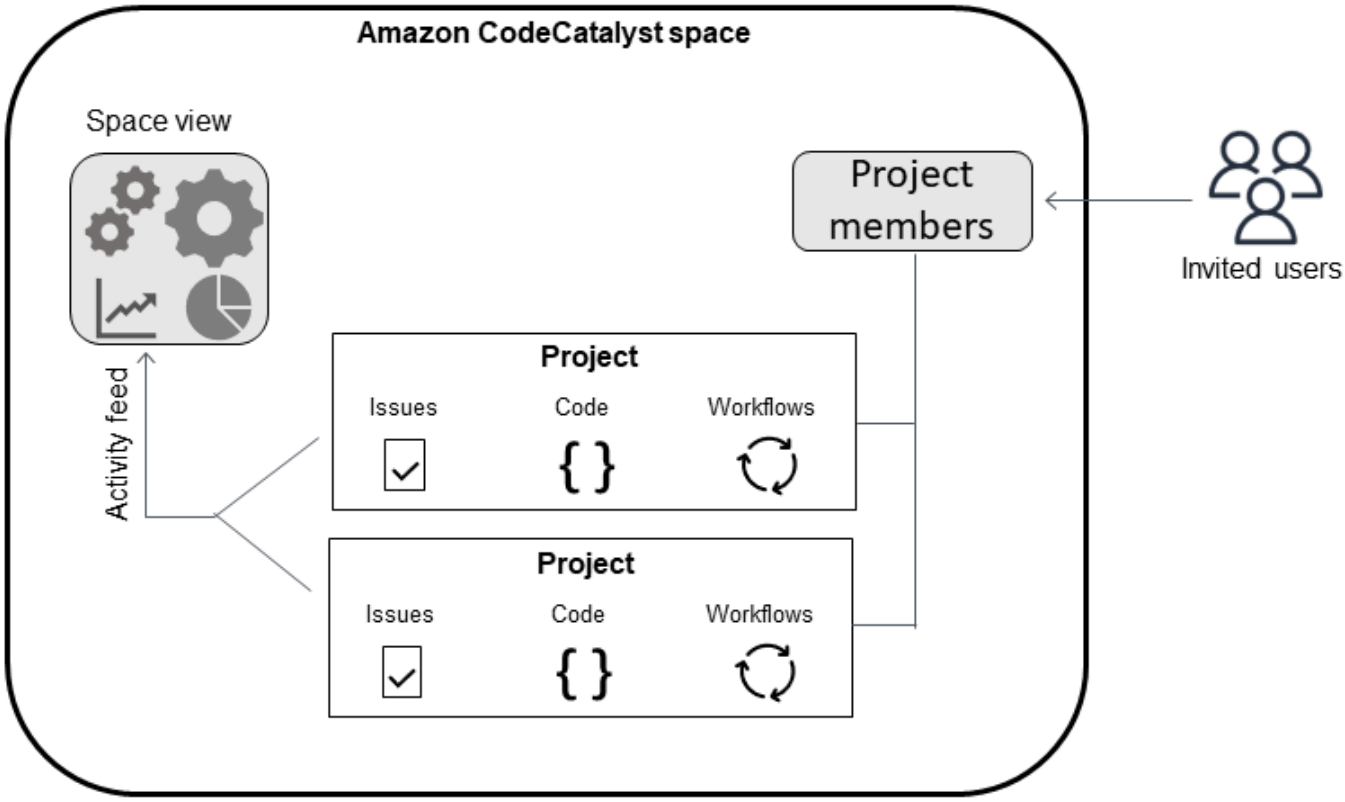
创建空间时，系统会自动为您分配空间管理员角色。您可以将此角色添加到空间中的其他用户。

使用空间管理员角色，您可以按如下方式管理空间：

- 向空间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提要

创建空间时，系统会自动将您添加到空间中，其中包含两个角色：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，他们会自动添加为空间的成员。该空间的成员资格不授予空间中的任何权限。用户在空间中可以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。



以下是添加账户的其他注意事项：

- 有一个空间的账户连接 one-to-one 映射。 AWS 账户 AWS 账户 可以将单个空间添加到多个不同的空间。您部署到的 AWS 账户不必是唯一的，并且可以由多个空间使用。
- 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。中只有一个 AWS 账户 可用作空间的结算账号 CodeCatalyst。如果某个账户已用于某个空间，则必须使用其他账单账户来获得额外的空间。
- 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，请参阅部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

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
- 允许计算机资源访问空间
- 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以满足您的业务需求。

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。您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**
>
> 当您创建其他空间时，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 Management Console，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，请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。您以后无法更改此设置。

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

6. 在中 AWS 区域，选择要存储空间和项目数据的区域。您以后无法更改此设置。
7. 在 AWS 账户 ID 中，输入您要连接到空间的账户的十二位数 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 Management Console。这是 Amazon CodeCatalyst Spaces 页面。您可能需要登录才能访问该页面。

   在中 AWS Management Console，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，请登录亚马逊 CodeCatalyst 空间，网址为 AWS Management Console https://console.aws.amazon.com/codecatalyst/home/。

   验证令牌会自动输入到验证令牌中。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。

    系统将显示账户验证成功消息，表明该账户已被添加到空间。
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 Management Console。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。此选项创建的服务角色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。 AWS CloudFormation该角色的`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*名称将附加唯一标识符。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，以帮助用户更好地了解空间的用途。

您必须具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，请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

您可以删除空间以移除对该空间所有资源的访问权限。您必须具有空间管理员角色才能删除空间。

> **Note**
>
> 您无法撤消对空间的删除。

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

> **Note**
>
> 空间名称在各处必须是唯一的 CodeCatalyst。您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。

要删除空间

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，请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，则会被重定向到空间概述页面。如果您属于一个空间，则会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，您可以查看工作流程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，请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
- 向账户连接添加 IAM 角色
- 将账户连接和 IAM 角色添加到您的部署环境中
- 查看账户连接
- 从空间中删除帐户（在 CodeCatalyst）

- 为空间配置结算账号

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。
- 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 允许在已连接的情况下访问 AWS 资源 AWS 账户。

账户连接是通过完成授权来创建的 AWS 账户。创建连接后，您可以通过添加 IAM 角色进一步配置要使用的工作流程和项目的连接。

向 AWS 账户 空间添加

您可以使用 CodeCatalyst 控制台和 AWS Management Console 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

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。您还必须对要添加的 AWS 账户 具有管理权限。

创建连接

1. 在中 AWS Management Console，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 Settings (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 Conn ec tion 描述中，输入账户描述，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. 该页面返回到AWS 账户 详细信息页面，其中显示成功横幅。

第 2 步：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 Management Console 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 Management Console，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 Settings (设置)，然后选择 AWS 账户。
4. 在AWS 账户 详细信息页面上，选择中的完成设置 AWS Management Console。
5. “验证 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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，请登录亚马逊 CodeCatalyst 空间，网址为 AWS Management Console https://console.aws.amazon.com/codecatalyst/home/。

验证令牌会自动输入到验证令牌中。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），为您的账单账户开启付费套餐。

> **Note**
>
> 这不会将计费等级升级到付费等级。但是，这将进行配置， AWS 账户 以便您可以随时更改空间的计费等级。 CodeCatalyst您可以随时开启付费等级。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。

系统将显示账户验证成功消息，表明该账户已被添加到空间。

步骤 3：查看已批准的连接

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。

查看已批准的连接

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

步骤 4：向您的连接添加 IAM 角色

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 向账户连接添加 IAM 角色。

后续步骤：为您的账户连接创建其他 IAM 角色

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

要关联您的账户，您需要为创建的角色提供亚马逊资源名称 (ARN)。复制您的一个或多个角色的 ARN，详见此处。有关使用 IAM 角色的 ARN 的更多信息，请参阅亚马逊资源名称 (ARN)。

要访问您的 IAM 角色，请执行以下操作：ARN

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
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。

向账户连接添加 IAM 角色

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 Management Console，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 Settings (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 Management Console。

   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`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*名称将附加唯一标识符。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，从下拉列表中选择角色。

选择 Add role (添加角色)。

该页面将在中打开 AWS Management Console。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，请登录亚马逊 CodeCatalyst 空间，网址为 AWS Management Console https://console.aws.amazon.com/codecatalyst/home/。

7. 选择为您的 CodeCatalyst 空间添加的账户。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“目的”下，选择一个角色用途，描述您希望如何在账户关联中使用该角色。指定`RUNNER`用于在工作流程中运行操作的角色。指定`SERVICE`用于访问其他服务的角色。

   您可以指定多个目的。

   > **Note**
   >
   > 必须为角色选择目的 ARN。

10. 选择关联 IAM 角色。为其他 IAM 角色重复这些步骤。

将账户连接和 IAM 角色添加到您的部署环境中

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。然后，您可以将 IAM 角色添加到您的账户连接中。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 角色添加到您的部署环境中，请确保按中所述创建账户连接和角色步骤 4：向您的连接添加 IAM 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

将您的账户连接和 IAM 角色添加到您的部署环境中

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 使用部署环境导航到要在其中添加账户连接和 IAM 角色的项目。
3. 展开 CI/CD，然后选择环境。
4. 选择您的环境，然后显示其他选项卡。
5. 选择 “AWS 账户 连接” 选项卡。在 “连接名称” 下，列出了已添加到环境中的帐户（如果有）。
6. 选择关联 AWS 账户。将<environment_name>显示 “ AWS 账户 与关联” 页面。
7. 在 “连接” 下，选择与要添加的 IAM 角色关联的账户名称。选择关联。

查看账户连接

您可以查看您的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. 导航到包含您要查看的账户关联的空间。
3. 选择 “AWS 帐户” 选项卡。
4. 在 “AWS 帐户” 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。

查看账户连接详情

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。选择 Settings (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 角色列表以及其他详细信息。

从空间中删除帐户（在 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。这会从您的空间中删除账户关联，前提是该账户不是该空间的结算账号。

> **Important**
>
> 删除帐户连接后，您将无法重新连接。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。在为指定账单账户的账户删除空间之前，您需要为自己的空间添加另一个账户。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删除帐户，但不建议这样做。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，您必须拥有空间管理员或高级用户角色。

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 Settings (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。在字段中输入姓名以确认删除，然后选择 “删除”。

   将显示成功横幅，并且该账户连接将从连接列表中删除。

为空间配置结算账号

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。您可以使用 CodeCatalyst 中的页面 AWS 来删除已添加到空间的帐户。在此过程中，使用您所管理的特定账户的管理权限，登录的

Amazon CodeCatalyst Spaces 页面， AWS 账户 从您的空间中移除。 AWS Management Console 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。

已删除的账户可以稍后重新添加，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 角色重新关联到已添加的账户。

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。如果您要添加结算账号，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 参考和示例策略，请参阅身份与访问管理与亚马逊 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将角色添加到您的账户关联中。

- 要创建包含该角色的权限策略和信任策略的服务CodeCatalystWorkflowDevelopmentRole-***spaceName***角色，请参阅CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。
- 有关创建角色和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程教程，如下所示：
  - 教程：将构件上传到 Amazon S3
  - 教程：使用部署无服务器应用程序 AWS CloudFormation
  - 教程：将应用程序部署到 Amazon ECS
  - 教程：在工作流程中使用 GitHub操作的 Lint 代码

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

您可以在 IAM 中将开发人员角色创建为一键式角色。您必须在要添加帐户的空间中拥有 Space 管理员或 P ower 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 Management Console 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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。
5. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建了一个服务角色，其中包含开发角色的权限策略和信任策略。该角色将有一个名字`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

您可以在 IAM 中将支持角色创建为一键式角色。您必须在要添加帐户的空间中拥有 Space 管理员或 Power 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 Management Console 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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。
4. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建的服务角色包含预览版开发角色的权限策略和信任策略。该角色的AWSRoleForCodeCatalystSupport名称将附加唯一标识符。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

或者，您可以创建一个包含该角色的权限策略和信任策略的服务`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色。有关更多信息，请参阅 向账户连接添加 IAM 角色。

1. 登录 AWS Management Console 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” 窗体下，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. 在角色名称中，输入角色的名称，例如：codecatalyst-project-role
9. 选择 创建角色。
10. 复制角色亚马逊资源名称 (ARN)。在向账户关联或环境中添加角色时，您需要提供此信息。

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，包括有关他们的显示名称、别名以及他们在空间中扮演的角色的信息。空间中的成员有三个角色：

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

  如果不先移除用户，则以后无法更改此角色。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。它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-默认情况下，此角色是为通过接受空间项目邀请加入空间的用户分配的。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。

空间成员表格显示了空间中在项目中扮演角色但不具有空间管理员角色的所有成员。

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且稍后接受项目邀请和角色的用户将不会显示在空间成员表格中的空间下或项目下的 “项目成员” 表格中。它们将继续显示在空间管理员表中的两个地方。在每个项目中，所有具有空间管理员角色的用户都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使用受限访问角色添加到空间中。如果用户的角色稍后更改为空间管理员角色，但也将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” 表格移至 “空间管理员” 表格。

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，请在顶部导航栏中选择一个空间。

3. 选择 “设置”，然后选择 “成员”。

   空间成员的用户将显示在空间成员表中。

   > **Tip**
   >
   > 如果您拥有 Space 管理员角色，则可以查看哪些项目是直接受邀的。导航到项目的项目设置，然后选择我的项目。

   在 “状态” 列中，以下是有效值：

   - 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。当您想邀请该用户通过为其分配空间管理员或超级用户角色来帮助您管理空间时，这非常有用。将其中一个角色分配给其他用户可以帮助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” 表格显示空间中在项目中扮演角色的所有成员。

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

邀请用户访问您的空间

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的空间的人的电子邮件地址。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送的加入空间的邀请，但该邀请尚未被接受，则可以将其取消。

要管理空间邀请，您必须拥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，请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. 验证该成员的状态是否为 “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. 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，因为他们是自动（隐式）分配给空间中的所有项目的。

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，请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，您可以将其移除。您必须具有空间管理员角色才能将成员从空间中移除。

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。在此表格中，您只能直接移除空间的成员。

从“项目成员”表格中移除用户

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，请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”表格中选择用户。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，以及与这些项目中的资源相关的权限。

移除或更改具有 Space 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际上会将该用户从 “空间管理员” 表中删除。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删除。

> **Note**
>
> 作为拥有 Space 管理员角色的用户，您无法将自己移除。联系其他具有空间管理员角色的用户。

从 “空间成员” 表格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（项目管理员或参与者）。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，并确保列表中没有未经授权的待处理空间邀请（“已邀请” 状态）。

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

5. 选择成员选项卡。在空间管理员表中，选择用户，然后选择删除。

   在“移除成员”对话框中，执行以下任一操作。

   - 选择仅删除用户的 Space 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删除。

   - 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 Space 管理员角色才能管理团队。

团队也在项目/空间层面进行管理。 CodeCatalyst要了解有关空间/项目中团队的更多信息，请参阅。允许使用团队访问空间

主题

创建团队

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相关联，每个项目的角色各不相同。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 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，并首先显示权限最多的角色。

> **Note**
>
> 如果您的空间支持身份联合，则必须已在 IAM Identity Center 中设置了 SSO 用户或 SSO 群组。

如何管理团队成员取决于添加和删除用户的方式。管理团队成员有两种选择：

- 直接添加用户-您可以单独添加或删除用户。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 群组 — 您可以通过已在 IAM Identity Center 中设置的 SSO 群组来管理团队成员。当您选择使用 SSO 群组管理团队成员时，直接添加用户的选项将不再可用。

您必须拥有 Space 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，输入您的团队的描述性名称。

   > **Note**
   >
   > 团队名称在您的空间中必须是唯一的。

   （可选）在团队描述中，输入您的团队的描述。

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。该角色将由团队的所有成员继承。

   - 空间管理员-有关详细信息，请参阅空间管理员角色。
   - 访问受限-有关详细信息，请参阅有限访问角色。
   - 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，选择向团队添加成员的方法。

   - 选择 “直接添加成员” 可单独管理用户。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 群组”，选择已在 IAM Identity Center 中设置的 SSO 群组。

     在 SSO 群组中，选中要添加的群组旁边的复选框。您最多可以添加五个 SSO 组。

   > **Note**
   >
   > 您以后无法更改此设置。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。当您选择使用 SSO 群组管理团队成员时，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 群组时，请注意，创建团队时不会拉出 SSO 群组中的用户。用户需要先登录 CodeCatalyst 才能显示在列表中。

查看球队

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 小组类型的团队，您还可以看到与该团队关联的 SSO 群组列表。

查看球队

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，查看分配给该空间团队的角色。
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列表。
6. 在 SSO 群组选项卡上，查看分配给团队的 SSO 群组列表（仅适用于支持身份联合的空间）。

为团队授予太空角色

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，您可以让团队能够为用户管理空间，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 Space 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。这会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。
   - 访问受限-有关详细信息，请参阅有限访问角色。
   - 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。角色变更将应用于团队，团队的所有成员都将继承这些权限。您可以为每个项目选择一个角色，该角色将自动授予团队。

您必须拥有 Space 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**
> 从团队中移除角色会移除团队中所有用户的相关权限。

5. 选择保存。

直接向团队添加用户

您可以将团队成员添加到您的团队中。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 群组管理团队成员时，直接使用添加用户的选项不可用。要使用 SSO 群组，请参阅向团队添加 SSO 群组。

您必须拥有 Space 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。您不能添加或邀请不是空间成员的团队成员。

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

您可以将团队成员从团队中移除。用户将不再继承所有权限。您可以稍后将该用户重新添加到团队中。

> **Note**
>
> 当您移除团队成员时，该用户的相关权限将从空间中的所有项目和资源中移除。

您必须拥有 Space 管理员角色才能管理团队。

删除团队成员

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. 在输入字段中输入 “移除”，然后选择 “移除”。

向团队添加 SSO 群组

如果您的空间配置为在 IAM Identity Center 中管理 SSO 用户和群组的空间，则可以添加一个 SSO 群组，该群组将作为一个单独的团队加入该空间。

> **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。

您必须具有空间管理员角色才能管理团队。

将 SSO 群组添加为一个小组

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 群组。您最多可以添加五个 SSO 组。

删除球队

您可以删除不再需要的团队。

> **Note**
>
> 删除团队时，将移除该空间中所有项目和资源中所有团队成员的相关权限。

您必须拥有 Space 管理员角色才能管理团队。

删除球队

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。这会改变整个团队的角色。
4. 选择 删除。

允许计算机资源访问空间

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” 一词并不指云基础设施，例如 Amazon EC2 实例，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。计算机资源用于向空间中的资源（例如蓝图和工作流程）授予权限。您可以查看空间中的计算机资源，也可以选择为空间启用或禁用计算机资源。例如，您可能需要禁用计算机资源来管理访问权限，然后稍后再将其重新启用。

在需要撤消或禁用计算机资源的情况下，这些操作可用于计算机资源。例如，如果您怀疑凭据可能已被泄露，则可以禁用计算机资源。通常，不需要使用这些操作。

您必须具有空间管理员角色才能查看此页面并在空间级别管理计算机资源。

计算机资源也在中的项目级别进行管理 CodeCatalyst。要了解有关项目中团队的更多信息，请参阅允许计算机资源访问空间 。

主题

查看计算机资源的空间访问权限

您可以查看您的空间中正在使用的计算机资源的列表。

您必须具有 Space 管理员角色才能管理计算机资源。

查看计算机资源

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”。选择计算机资源。
3. 在下拉列表中，选择工作流操作以仅查看工作流的计算机资源。选择 Blue p rint 可仅查看蓝图的计算机资源。

   您也可以使用 “筛选器” 字段对名称进行筛选。

禁用计算机资源的空间访问权限

您可以选择禁用空间中正在使用的计算机资源。

> **Important**
> 禁用计算机资源将移除对空间中所有关联蓝图或工作流程的所有权限。

您必须具有 Space 管理员角色才能管理计算机资源。

禁用计算机资源

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”。选择计算机资源。
3. 选择以下选项之一。

   > **Important**
   > 禁用计算机资源将移除对空间中所有关联蓝图或工作流程的所有权限。

   - 要单独禁用，请选择要禁用的一个或多个计算机资源旁边的选择器。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。
   - 要禁用所有工作流程操作，请选择 “禁用”，然后选择 “所有工作流程操作”。
   - 要禁用所有蓝图，请选择禁用，然后选择所有蓝图。

为计算机资源启用空间访问权限

您可以选择启用空间中正在使用且已被禁用的计算机资源。

您必须具有 Space 管理员角色才能管理计算机资源。

启用计算机资源

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”。选择计算机资源。
3. 选择以下选项之一。
   - 要单独启用，请选择要启用的一个或多个计算机资源旁边的选择器。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。
   - 要启用所有工作流程操作，请选择 “启用”，然后选择 “所有工作流程操作”。
   - 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内项目的一部分创建的。空间成员可以在源代码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，您代表空间成员维护这些资源。
- 开发环境默认为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。

有关开发环境的其他注意事项，包括停止运行实例、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

查看您所在空间的开发环境

您可以查看您所在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。

您必须具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。

查看您所在空间中的开发环境

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，请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”。

   该页面列出了您所在空间中的所有开发环境。您可以查看资源名称、资源别名（如果适用）、IDE 类型、默认或配置的计算和存储，以及为每个开发环境配置的超时。

为你的空间编辑开发环境

您可以编辑开发环境的配置，例如为空闲的开发环境停止运行配置的超时长度（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。

您必须具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。

编辑您所在空间中的开发环境

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，请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. 根据需要对开发环境的计算或非活动状态超时进行更改。
6. 选择保存。

为你的空间停止开发环境

如果开发环境配置为超时，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，超时已过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。

您必须具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。

在你的空间中停止开发环境

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，请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。

为你的空间删除开发环境

您可以删除不再需要或不再拥有所有者的开发环境。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。

您必须具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。

删除您空间中的开发环境

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，请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。要进行确认，请键入`delete`，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，请参阅的配额 CodeCatalyst。

一个空间的最大 Slack 通道数	500
一个电子邮件地址的最大邀请数量	25
每位用户的最大邀请数量	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空间的最大数量	5
一支队伍的最大 SSO 群组数	5
一个空间的最大队伍数量	100
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，则其长度必须介于 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、破折号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。您不能重复使用已删除空间的名称。 空间名称的长度必须介于 3 到 63 个字符之间。它们还必须以字母数字字符开头。空间名称可以包含字母、数字、句点、下划线和破折号的任意组合。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通过共享的持续集成/持续交付 (CI/CD) 工作流程和存储库来执行开发任务。创建项目时，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，但只能使用您所参与的项目。创建项目时，将生成项目的默认角色，然后将其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 “参与者” 角色）的任何人都可以访问项目资源，例如源存储库。
- 任何拥有 Space 管理员项目管理员或角色的人都可以发送加入项目的邀请。
- 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。
- 作为 CI/CD 工作流程的一部分，具有受限访问权限角色的用户可以管理功能、代码修复和测试的项目分配。

  工作流用于构建、测试、发布或更新作为 CI/CD 管道的应用程序。您可以通过添加用于传输和处理源构件的操作来组装工作流程。当您运行操作时，您的项目云资源用于为您的工作流程操作提供按需计算能力。您可以根据要设置的活动和输出配置更多 CI/CD 工作流程。例如，您可以创建一个仅用于生成和测试操作的工作流程，在修复错误的同时，无需部署即可查看测试结果并完成工作流程。然后，您可以创建另一个工作流程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。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，您可以手动添加或移除资源。

每个项目都以用户事件列表的形式跟踪项目活动，例如创建项目或修改资源的时间。项目活动在空间层面进行监控和汇总。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，您可以将源存储库、议题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，您可以使用共享的持续集成/持续交付 (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，您必须拥有 Space 管理员或 Power 用户角色。

主题

使用蓝图创建项目

您可以使用项目蓝图来配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”。
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 “为项目命名” 下，输入要分配给项目的名称及其关联的资源名称。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。或者，您可以选择将蓝图的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。

   根据您要使用的第三方存储库提供商，执行以下任一操作：

   - 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 人际关系 和 通过人际关系访问 GitHub 资源。

- Bitbucket 存储库：连接 Bitbucket 工作空间。

  选择 “高级” 下拉菜单，选择 Bitbucket 作为存储库提供程序，然后选择要存储蓝图创建的源代码的 Bitbucket 工作空间。

7. 在项目资源下，配置蓝图参数。根据蓝图，您可以选择命名源存储库的名称。
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 “查看工作流程”。
9. （可选）从蓝图的卡片中选择查看详细信息以查看有关蓝图的特定详细信息，例如蓝图架构概述、所需的连接和权限以及蓝图创建的资源类型。
10. 选择创建项目。

在 Amazon 中创建一个空项目 CodeCatalyst

您可以创建一个没有资源的空项目，然后在以后手动添加所需的资源。

在创建项目之前，您必须拥有 Space 管理员或 Power 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 或 Bitbucket 存储库的新 CodeCatalyst 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 CodeCatalyst 链接到您的 GitHub 账户或 Bitbucket 工作空间中的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 存储库或 Bitbucket 扩展程序。在外部站点中，系统会提示您连接并 CodeCatalyst 提供对第三方存储库的访问权限，这将在下一步中完成。

   > **Important**
   >
   > 要将GitHub 存储库或 Bitbucket 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 Space 管理员角色的账户登录。

2. 将您的 GitHub 账户或 Bitbucket 工作空间连接到。 CodeCatalyst

   > **Important**
   >
   > 要将您的 GitHub 账户或 Bitbucket 工作 CodeCatalyst 空间连接到您的空间，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。

> **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 人际关系 和 通过人际关系访问 GitHub 资源。

3. 创建一个链接到您的 GitHub 存储库或 Bitbucket 存储库的 CodeCatalyst 项目。

> **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 存储库和 Jira 项目 CodeCatalyst。

> **Note**
>
> - GitHub 只能将一个空间中的一个 CodeCatalyst 项目链接到一个项目。
> - 您不能在 CodeCatalyst 项目中使用空仓库或已存档 GitHub 存储库。
> - 您不能链接与 GitHub 项目中仓库同名的存储库。 CodeCatalyst
> - GitHub 存储库扩展与 GitHub 企业服务器存储库不兼容。

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 存储库或 Bitbu cket 存储库。系统会提示你关联 GitHub 账户或 Bitbucket 工作空间。如果尚未安装第三方扩展程序，则会显示安装提示。
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 账户或 Bitbucket 工作空间连接到您的 CodeCatalyst 空间。

将您的 GitHub 或 Bitbucket 存储库提供商连接到 CodeCatalyst

根据您选择配置的第三方扩展，执行以下任一操作：

- 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 工作空间。

  1. 选择 Connect Bitbucket 工作空间可前往 Bitbucket 的外部站点。
  2. 使用您的 Bitbucket 凭据登录到您的 Bitbucket
  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 工作空间连接到该空间，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。为要允许的工作空间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。

将第三方存储库提供商连接到后 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，选择您连接的 GitHub 账户或 Bitbucket 工作空间。
2. 根据您连接的第三方存储库提供商，选择GitHub 存储库或 Bitbucket 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“命名您的项目”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 存储库或 Bitbucket 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。您还可以使用蓝图生成的代码在关联的 GitHub 账户或 Bitbucket 工作空间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程 和 创建开发环境。

向已创建的项目添加资源和任务

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的 CI/CD 工作流程，请参阅。工作流程入门
- 要使用类似于新项目中将构建项目部署到 Amazon S3 存储桶的构建操作，请参阅使用工作流程进行构建和教程：将构件上传到 Amazon S3。
- 要从一个空项目开始，然后使用 AWS CloudFormation 堆栈部署来部署类似的无服务器应用程序，请参阅教程：使用部署无服务器应用程序 AWS CloudFormation。
- 要添加问题计划板，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，请参阅获取项目清单。
- 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。

- 要设置发送工作流程运行成功或失败状态警报的通知，请参阅在 Amazon 中管理通知 CodeCatalyst。
- 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。

获取项目清单

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。

要查看项目，您必须是该项目的成员，或者拥有该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。对于要在其中创建项目的空间，您必须具有空间管理员角色。

- 在项目概述中，您可以查看项目成员、源存储库、工作流程运行、打开的拉取请求、项目开发环境和问题。
- 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

要查看项目任务的摘要，例如分配给你或由你创建的未解决议题和拉取请求，以及项目的关联开发环境，请使用控制台。

要查看项目，您必须是该项目的成员，或者拥有该空间的空间管理员角色。

查看您的源代码库、工作流程运行、议题、拉取请求、开发环境和问题

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含您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. 查看分配给您并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”卡片以查看与项目关联的源存储库。
    - 查看“工作流程运行”卡片以查看与项目关联的工作流程。
    - 查看“打开拉取请求”卡片，查看代码存储库状态摘要，以及分配给您并由您创建的拉取请求。
    - 查看“我的开发环境”卡片，查看与项目关联的开发环境摘要。

- 查看 “问题” 卡片，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。

要查看项目任务的摘要，例如分配给你或由你创建的未解决议题和拉取请求，以及项目的关联开发环境，请使用控制台。

要查看项目，您必须是该项目的成员，或者拥有该空间的空间管理员角色。

查看源代码库、工作流程运行、议题、拉取请求、开发环境和问题

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含您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、工作流程运行、打开的拉取请求、项目开发环境和问题。

要查看项目任务的摘要，例如分配给你或由你创建的未解决议题和拉取请求，以及项目的关联开发环境，请使用控制台。

查看您的源代码库、工作流程运行、议题、拉取请求、开发环境和问题

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含您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

要切换到其他项目，请使用控制台从您有权访问的项目列表中进行选择。

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. 展开下拉列表并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资源的所有访问权限。您必须具有空间管理员或项目管理员角色才能删除项目。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提示的任何工作流程都将停止。

删除您的项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含您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。您可以添加或删除用户、管理当前成员的角色、发送加入项目的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，并首先显示权限最多的角色。

获取成员及其项目角色的列表

在向项目中添加用户时，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”角色具有在项目中工作所需的权限。将此角色分配给需要在项目中处理代码、工作流程、问题和操作的用户。有关更多信息，请参阅 贡献者角色。

- 审阅者角色具有审阅权限。有关更多信息，请参阅 审阅者角色。
- 只读角色具有读取权限。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，因为他们已经隐式访问空间中的所有项目。

当您邀请用户加入您的项目（未分配空间管理员角色）时，该用户将显示在项目成员表格中的项目下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. 导航到包含您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” 表格显示了在项目中具有角色的所有成员。

   > **Tip**
   >
   > 如果您拥有 Space 管理员角色，则可以查看哪些项目是直接受邀的。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。这些用户会自动（隐式）分配给空间中的所有项目，并且不在项目中扮演任何角色。

   在 “状态” 列中，以下是有效值：

   - 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
- 取消邀请
- 从项目中移除用户
- 接受或拒绝项目邀请

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在空间之外添加姓名。

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，因为他们已经隐式访问空间中的所有项目。

当您邀请用户加入您的项目（未分配空间管理员角色）时，该用户将显示在项目成员表格中的项目下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

如果您最近发送了邀请，则只要邀请尚未被接受，就可以取消邀请。

要管理项目邀请，您必须具有项目管理员或空间管理员角色。

取消项目成员邀请

1. 导航到您已向其发送要取消的邀请的项目。
2. 在导航窗格中，选择项目设置。
3. 查看 “成员” 选项卡，确认该成员的状态是否为 “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. 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

您可以使用控制台将用户从项目中移除。

要将用户从项目中移除，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该空间中移除。

从项目中移除用户

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含您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. 选择要删除的个人资料旁边的选择器，然后选择 “删除”。
6. 确认您要删除该用户，然后选择删除。

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，通知他们您拒绝了邀请。

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。

您必须具有项目管理员角色才能管理项目的团队。

团队也在空间层面进行管理 CodeCatalyst。要了解有关空间中团队的更多信息，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，让团队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，并首先显示权限最多的角色。

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 “项目设置”，然后选择 “团队”。
3. 选择添加团队。
4. 在 Tea m 中，从可用队伍列表中选择一支球队。
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 “项目设置”，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 “项目设置”，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**
> 从团队中移除角色会移除团队中所有用户的相关权限。

5. 选择保存。

允许对计算机资源进行项目访问

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” 一词并不指云基础设施，例如 EC2 实例，而是指具有空间或项目权限的蓝图或工作流程资源。

在项目中使用计算机资源的一个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。计算机资源用于向项目中的资源（例如蓝图和工作流程）授予权限。您可以查看项目中的计算机资源，也可以选择为项目启用或禁用计算机资源。例如，您可能需要禁用计算机资源来管理访问权限，然后稍后再将其重新启用。

在需要撤消或禁用计算机资源的情况下，这些操作可用于计算机资源。例如，如果您怀疑凭据可能已被泄露，则可以禁用计算机资源。通常，不需要使用这些操作。

您必须具有 Space 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计算机资源。

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/项目中团队的更多信息，请参阅。允许计算机资源访问空间

主题

查看计算机资源的项目访问权限

您可以查看项目中正在使用的计算机资源的列表。

您必须具有空间管理员角色或项目管理员角色。

查看计算机资源

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。选择计算机资源。
3. 在下拉列表中，选择工作流操作以仅查看工作流的计算机资源。选择蓝图可仅查看蓝图的计算机资源。

   您也可以使用 “筛选器” 字段筛选姓名。

禁用计算机资源的项目访问权限

您可以选择禁用项目中正在使用的计算机资源。

> **Important**
>
> 禁用计算机资源将移除对空间中所有关联蓝图或工作流程的所有权限。

您必须具有空间管理员角色或项目管理员角色。

禁用计算机资源

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。选择计算机资源。
3. 选择以下选项之一。

> **Important**
>
> 禁用计算机资源将移除对空间中所有关联蓝图或工作流程的所有权限。

- 要单独禁用，请选择要禁用的一个或多个计算机资源旁边的选择器。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。
- 要禁用所有工作流程操作，请选择“禁用”，然后选择“所有工作流程操作”。
- 要禁用所有蓝图，请选择禁用，然后选择所有蓝图。

为计算机资源启用项目访问权限

您可以选择启用项目中正在使用且已被禁用的计算机资源。

您必须具有空间管理员角色或项目管理员角色。

启用计算机资源

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。选择计算机资源。
3. 选择以下选项之一。

   - 要单独启用，请选择要启用的一个或多个计算机资源旁边的选择器。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。
   - 要启用所有工作流程操作，请选择“启用”，然后选择“所有工作流程操作”。
   - 要启用所有蓝图，请选择启用，然后选择所有蓝图。

项目配额

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，请参阅的配额 CodeCatalyst。

每个空间的最大项目数	100

用户可以加入的最大项目数	1000
可以属于一个项目的最大成员数。	10000
项目名称	空间内的项目名称必须是唯一的。名称必须介于 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、空格和.、 _（下划线）-（连字符） 项目名称不能包含以下任何字符：！ ？ @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
产品描述	项目描述最多可包含 200 个字符。有效字符：A-Z、a-z、0-9、空格和.、 _（下划线）-（连字符）。产品描述是可选的。

在中处理通知 CodeCatalyst

您可以在中设置通知来监控您的项目和资源 CodeCatalyst。在他们所属的任何项目中，用户可以选择要接收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 ID，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传递应用程序（例如 Slack）提供通知。

通知需要哪些权限？

任何项目成员都可以在中配置、查看、更新或删除频道的通知设置 CodeCatalyst。但是，只有空间管理员角色的用户才能添加或删除 Slack 工作空间。所有用户都可以为他们所属的项目配置他们希望接收的有关哪些项目事件的电子邮件 CodeCatalyst。

我可以配置哪些 CodeCatalyst 事件的通知？

您可以配置为 CodeCatalyst 向一个或多个 Slack 频道发送有关工作流程事件的通知。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 频道中直接接收有关事件的消息。 CodeCatalyst 添加 Slack 成员 ID 的用户将在为其项目配置的 Slack 频道中直接被提及其 ID，这有助于提高人们对他们关心的事件的认识。

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

通知是如何浮出水面的？

您可以配置为 CodeCatalyst 向一个或多个 Slack 频道发送通知。您需要授权 CodeCatalyst 才能授予访问您的 Slack 工作空间的权限。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，他们可以在为该项目配置的 Slack 频道中收到有关 CodeCatalyst事件的提及。

如何设置通知？

电子邮件通知配置为的一部分 CodeCatalyst。项目用户可以在 “我的设置” 页面中选择他们希望接收的电子邮件中的事件。

要为您的项目资源设置 Slack 通知，您必须完成以下高级任务。

设置通知（高级任务）

1. 在中 CodeCatalyst，您可以在 CodeCatalyst 和消息客户端（例如 Slack）之间建立连接。连接 Slack 工作空间后，该工作区将可供该空间中的所有项目使用。

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

2. 在中的项目中 CodeCatalyst，添加您希望团队接收通知的频道。
3. 在中 CodeCatalyst，您可以打开各种事件（例如工作流程运行失败）的通知，并指定要将通知发送到哪个渠道。

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，以直接接收有关为其项目配置的 Slack 频道中 CodeCatalyst 事件的消息，

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，请参阅在 Amazon 中管理通知 CodeCatalyst。

> **Note**
>
> 可以发送到通知渠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，请参阅在 Amazon 中管理通知 CodeCatalyst：

主题

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 该频道可以是公开的，也可以是私人的。

第 1 步： CodeCatalyst 连接到你的 Slack 工作空间

只有拥有 Space 管理员角色的用户才能添加或删除 Slack 工作空间。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 工作空间 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**
>
> 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 工作空间”。
7. 阅读对话框内容，然后选择 “连接到 Slack 工作区”。
8. 在 AWSChatbot 消息中：
   a. 在右上角，选择包含您的频道的 Slack 工作区。
   b. 选择 Allow。

您将返回到 CodeCatalyst 控制台。

9. 继续浏览 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

要获取你的 Slack 频道 ID

1. 登录 Slack。有关更多信息，请参阅登录 Slack。
2. 前往包含您要发送通知的频道的 Slack 工作区。有关更多信息，请参阅在 Slack 工作空间之间切换或登录其他 Sl ack 工作空间。
3. 在导航窗格中，打开要发送通知的频道的上下文（右键单击）菜单，然后选择打开频道详情。

   频道 ID 显示在对话框的底部。
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 频道是私密频道，请按如下方式AWS将 Chatbot 应用添加到该频道：

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

   b. 按 Enter 键。

      此时会出现 Slackbot 消息，表明AWS聊天机器人不在私人频道中。

   c. 选择 “邀请他们” 以邀请 AWS Chatbot 加入频道。
2. 在 CodeCatalyst 控制台中，选择 “下一步”。
3. 在频道 ID 中，粘贴你之前获得的 Slack 频道 ID。
4. 在频道名称中，输入一个名称。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，选择要接收通知的事件类型。
7. 选择 Finish (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程状态通知后，您可以在 Slack 中查看通知。

在 Slack 中查看你的通知

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，查看你为通知设置的频道。您的通知会显示每次工作流程运行的最新状态以及工作流程是失败还是成功。

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 ID 配置个人 Slack 通知，并配置他们将接收电子邮件的事件。有关更多信息，请参阅 在 Amazon 中管理通知 CodeCatalyst。

在 Amazon 中管理通知 CodeCatalyst

您可以配置 CodeCatalyst 为发送有关项目中事件的通知。您可以向消息客户端（例如 Slack 频道）发送通知。项目用户可以选择通过向为其个人资料配置的电子邮件地址发送电子邮件来通知他们哪些项目事件。

> **Note**
>
> 可以发送到通知渠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。

主题

- 管理直接发送给您的通知
- 管理发送到频道的通知

管理直接发送给您的通知

您可以选择向您发送有关您所属的任何项目中发生的事件的电子邮件通知。这些电子邮件将发送到您的 AWS Builder 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以发送电子邮件的项目事件的电子邮件。

如果已将项目配置为向 Slack 频道发送通知，则可以添加您的 Slack 成员 ID，以接收有关该 Slack 频道中 CodeCatalyst 事件的直接提及。这可以帮助你提高对你所参与的项目中发生的事件的认识。

为项目事件配置电子邮件通知

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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而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. 选择您要接收电子邮件的事件，然后选择 “保存”。

配置个人 Slack 通知

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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而找到您的用户个人资料。

3. 在 “个人 Slack 通知” 中，选择 Connect Slack ID，然后选择 “连接到 Sl ack 工作空间”。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，否则此选项不可配置。有关更多信息，请参阅 开始使用 Slack 通知 和 管理发送到频道的通知。

4. 在权限请求窗口中，确保工作空间的名称与为该空间配置的 Slack 工作 CodeCatalyst 空间相匹配。选择 “允许” 以允许AWS Chatbot访问工作区。窗口将关闭，Slack 工作区将显示连接状态为 “已连接”。

> **Tip**
>
> 如果连接状态没有改变，请检查连接 Slack 工作区时是否出现错误。您可能需要向上滚动才能看到错误。

5. 要停止接收个人 Slack 通知，请选择连接的 Slack 工作区，然后选择断开连接 Sl ack ID。

管理发送到频道的通知

您可以选择在 CodeCatalyst 发送到团队资源（例如团队 Slack 频道）中添加和管理有关项目事件的通知。通过这样做，您可以帮助确保整个团队都知道重要事件，例如工作流程运行失败的情况。

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

为项目添加通知渠道

您可以添加要在其中接收通知的频道，例如团队的 Slack 频道。

为通知添加 Slack 频道

1. 如果您要添加第一个 Slack 频道，请改开始使用 Slack 通知为查看。

   设置第一个频道后，请返回此过程以设置其他频道。
2. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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 Add channel（添加通道）。
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，则可以按照中的开始使用 Slack 通知说明进行添加。
8. 在输入频道 ID 之前，如果您要添加的 Slack 频道是私密频道，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 a ws 应用程序。
   b. 按 Enter 键。

      此时会出现 Slackbot 消息，表明AWS聊天机器人不在私人频道中。
   c. 选择 “邀请他们” 以邀请 AWS Chatbot 加入频道。
9. 在 CodeCatalyst “频道 ID” 字段中，输入 Slack 频道 ID。要查找 ID，请前往 Slack，然后在导航窗格中右键单击该频道，然后选择 “打开频道详情”。

   频道 ID 显示在对话框的底部。
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，选择要接收通知的事件类型。
12. 选择 Add（添加）。

编辑通知渠道的通知

您可以更改通知发送到哪些频道，也可以完全关闭特定的通知。

编辑通知

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. 选择编辑通知。
6. 请执行下列操作之一：

   - 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

移除频道

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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. 选择要删除的频道旁边的指示器，然后选择移除频道。出现提示时，在确认窗口中选择 Ok。

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。该组件可以由任何内容组成，从单个文件中的工作流程到包含示例代码的整个项目。蓝图采用任意一组选项，并使用这些选项生成任意一组输出代码，然后将其转发到项目中。当蓝图使用最新的最佳实践或新选项进行更新时，它可以在包含该蓝图的项目中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应用于现有 CodeCatalyst 项目。你可以在项目中应用多个蓝图，每个蓝图都可以作为一个独立的组件来应用。例如，您可以创建使用 Web 应用程序蓝图创建的项目，然后稍后再应用安全蓝图。当其中一个蓝图更新后，您可以通过生命周期管理将更改或修复纳入您的项目中。有关更多信息，请参阅 使用 CodeCatalyst 蓝图创建综合项目 和 以蓝图用户身份使用生命周期管理。

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。可以开发自定义蓝图，以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进行更新，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 标准化项目自定义蓝图 CodeCatalyst。要查看蓝图 SDK 和示例蓝图，请参阅开源 GitHub 存储库。

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。根据蓝图，您的项目是使用特定资源创建的。有关更多信息，请参阅 使用蓝图创建项目 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将 CodeCatalyst 目录或空间目录中的其他蓝图应用于您的 CodeCatalyst 项目。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，以整合团队的最佳实践。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，请参阅以蓝图用户身份使用生命周期管理。

在项目中应用蓝图以添加资源

您可以在一个项目中应用多个蓝图来整合功能组件、资源和治理。您的项目可以支持在单独的蓝图中独立管理的各种元素。将蓝图应用于项目可以减少手动创建资源和使软件组件正常运行的需求。随着需求的变化，您的项目也可以保持最新状态。要了解有关在项目中应用蓝图的更多信息，请参阅以蓝图用户身份使用生命周期管理。

在配置蓝图的详细信息时，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，您仍然可以在该存储库中管理蓝图并利用生命周期管理功能使您的项目保持最新状态。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 以蓝图用户身份使用生命周期管理。

将蓝图应用于您的项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到空间，然后选择要应用蓝图的项目。
3. 在导航窗格中，选择蓝图，然后选择应用蓝图。
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。或者，您可以选择将蓝图的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。

   根据您要使用的第三方存储库提供商，执行以下任一操作：

   - 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 人际关系 和 通过人际关系访问 GitHub 资源。

- Bitbucke t：连接 Bitbucket 工作空间。

  选择 “高级” 下拉菜单，选择 Bitbucket 作为存储库提供程序，然后选择要存储蓝图创建的源代码的 Bitbucket 工作空间。

7. 在配置蓝图下，配置蓝图参数。根据蓝图，您可以选择命名源存储库。
8. 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显示了当前版本和最新版本之间的更改，最新版本是创建拉取请求时的所需版本。如果未显示任何更改，则版本可能相同，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要查看的代码和更改感到满意，请选择 A pply blueprint。创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用@符号和文件名来添加指向诸如文件之类的资源的链接。

> **Note**
>
> 在拉取请求获得批准并合并之前，蓝图才会被应用。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

蓝图作者还可以将自定义蓝图应用于指定空间中没有蓝图可用来创建新项目或应用于现有项目的项目。有关更多信息，请参阅 在指定空间和项目中发布和应用自定义蓝图。

如果您不想再接收蓝图的更新，可以取消该蓝图与项目的关联。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

如果您不想对蓝图进行新的更新，则可以取消该蓝图与项目的关联。从蓝图中添加到项目中的资源和功能软件组件将保留在您的项目中。

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. 选择包含要取消关联的资源的蓝图，选择操作下拉菜单，然后选择取消关联蓝图。
5. 输入`confirm`以确认解除关联。
6. 选择确认。

更改项目中的蓝图版本

如果您使用蓝图创建项目或将蓝图应用于现有项目，则会收到蓝图新版本的通知。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。生命周期管理允许您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他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 以蓝图用户身份使用生命周期管理。

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. 在空间仪表板上，选择包含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。拉取请求中显示的区别在于当前版本和最新版本之间的变化，最新版本是创建拉取请求时的所需版本。如果未显示任何更改，则版本可能相同，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要查看的代码和更改感到满意，请选择 A pply update。创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求中，也可以添加到文件中的各个行以及整个拉取请求中。您可以使用@符号和文件名来添加指向诸如文件之类的资源的链接。

> **Note**
>
> 在拉取请求获得批准并合并之前，蓝图不会更新。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

> **Note**
>
> 如果您已打开用于更新蓝图版本的现有拉取请求，请在创建新的拉取请求之前关闭之前的拉取请求。当您选择更新版本时，您将被定向到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. 在空间仪表板上，选择包含要更新的蓝图的项目。
4. 在导航窗格中，选择 Blueprin ts，然后选择要更新的蓝图的单选按钮。
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。拉取请求中显示的区别在于当前版本和最新版本之间的变化，最新版本是创建拉取请求时的所需版本。如果未显示任何更改，则版本可能相同，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要查看的代码和更改感到满意，请选择 A pply update。创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求中，也可以添加到文件中的各个行以及整个拉取请求中。您可以使用@符号和文件名来添加指向诸如文件之类的资源的链接。

> **Note**
>
> 在拉取请求获得批准并合并之前，蓝图不会更新。有关更多信息，请参阅 查看拉取请求 和 合并拉取请求。

> **Note**
>
> 如果您已打开用于更新蓝图版本的现有拉取请求，请在创建新的拉取请求之前关闭之前的拉取请求。当您选择更新版本时，您将被定向到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。

在项目中编辑蓝图的描述

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。一个蓝图可以在一个项目中多次使用。为了区分项目中蓝图的用途，您可以使用这些蓝图的描述。描述还可用于识别您正在从特定蓝图中应用的组件。

在项目中编辑蓝图的描述

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. 选择包含要更新的描述的蓝图，选择操作下拉菜单，然后选择编辑描述。
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。

以蓝图用户身份使用生命周期管理

生命周期管理是指能够根据蓝图的更新选项或版本重新生成代码库。这样，蓝图作者就可以集中管理包含特定蓝图的每个项目的软件开发生命周期。例如，将安全修复推送到 Web 应用程序蓝图将允许每个包含 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程序。同样的管理框架还允许您作为蓝图用户在选择蓝图选项后对其进行更改。

主题

- 对现有项目使用生命周期管理
- 对项目中的多个蓝图使用生命周期管理
- 处理生命周期拉取请求中的冲突

对现有项目使用生命周期管理

您可以对根据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。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。蓝图生成安全扫描工作流程和其他相关代码。现在，每当对蓝图进行更改时，Java 应用程序中的那部分代码库都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

由于蓝图代表建筑组件，因此在同一个项目中通常可以同时使用多个蓝图。例如，一个项目可以由公司平台工程师构建的中央 Web API 蓝图以及应用程序安全团队构建的发布检查蓝图组成。这些蓝图中的每一个都可以独立更新，并且会记住过去应用于它们的合并分辨率。

> **Note**
>
> 作为任意架构组件，并非所有蓝图在一起都有意义或逻辑上可以协同工作，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。这些问题可以手动解决。后续的蓝图更新中会记住分辨率。

选择退出生命周期管理变更

用户可以从项目中移除蓝图，以取消对蓝图的所有引用并选择退出生命周期更新。出于安全考虑，这不会删除或影响项目的任何代码或资源，包括从蓝图中添加的内容。有关更多信息，请参阅 取消蓝图与项目的关联以停止更新。

在项目中重写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。蓝图始终尊重代码所有者文件而不是其他所有文件，并且可以生成如下所示的示例：

```
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.ownership-file包含以下内容的：

```
[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架构、资源和示例源工件。

有关更多信息，请参阅 创建项目。只有空间管理员才能创建项目。

主题

- 可用的蓝图
- 查找项目蓝图信息

可用的蓝图

蓝图名称	蓝图描述
ASP.NET 核心网页 API	该蓝图创建了一个.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，将逗号分隔值 (CSV) 转换为 Apache Parquet。
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	此蓝图创建了一个容器化 Web 服务项目。该项目使用 C AWS opilot CLI 在亚马逊 ECS 上构建和部署由亚马逊 DynamoDB 支持的容器化 Spring Boot Java 网络服务。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。该应用程序将数据存储在 DynamoDB 表中。工作流程成功运行后，示例 Web 服务将通过 Application Load Balancer 公开提供。
现代三层 Web 应用程序	该蓝图使用 Python 为应用层和 Vue 前端框架生成代码，用于构建和部署架构良好的三层现代 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#。

蓝图名称	蓝图描述
Node.js API 带有 AWS Fargate	此蓝图创建了一个容器化 Web 服务项目。该项目使用 C AWS opilot CLI 在亚马逊弹性容器服务上构建和部署容器化 Express/Node.js 网络服务。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。工作流程成功运行后，示例 Web 服务将通过 Application Load Balancer 公开提供。
无服务器应用程序模型 (SAM)	此蓝图创建了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
无服务器图像处理器	该蓝图创建了一个在不降低图像质量的情况下进行高速图像处理的应用程序。
无服务器 RESTful 微服务	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	该蓝图创建了一个使用 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。
静态网站	此蓝图使用 Hugo 或 Jekyll 静态网站生成器创建静态网站。静态网站生成器使用文本输入文件（例如 Markdown）生成静态网页。它们非常适合存放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。

蓝图名称	蓝图描述
待办事项 Web 应用程序	该蓝图创建了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
V ideo-on-demand 网络服务	该蓝图创建了一项 video-on-demand 服务，该服务提供了接收、转码和交付内容的能力。蓝图使用 AWS Lambda Amazon S3 和 AWS Elemental MediaConvert。 Amazon CloudWatch
订阅外部蓝图	该蓝图为每个导入的包创建一个工作流程。这些工作流程每天运行一次，以检查 NPM 中是否有新版本的软件包。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。如果找不到包或不是蓝图，则操作将失败。目标包必须在 NPM 上，并且包必须是蓝图。必须按支持自定义蓝图的等级订阅空间。
Bedrock GenAI 聊天机器人	该蓝图使用亚马逊 Bedrock 和 Anthropic 的 Claude 创建了一个生成式人工智能聊天机器人。使用此蓝图，您可以构建和部署自己的安全、受登录保护的 LLM Playground，并可以根据您的数据进行自定义。有关更多信息，请参阅 B edrock GenAI Chatbot 文档。
AWS 项目开发套件 (AWS PDK) 蓝图	这些 PDK 蓝图可以组合在一起，创建一个包含 React 网站、Smithy API 和支持 CDK 基础设施的应用程序，用于将其部署到 AWS。AWS PDK 提供了常用模式的构建块以及用于管理和构建项目的开发工具。有关更多信息，请参阅 AWS PDK GitHub 源存储库和。教程：使用可组合的 PDK 蓝图创建全栈应用程序

查找项目蓝图信息

中 CodeCatalyst提供了多个项目蓝图。每份蓝图都附有一份摘要和自述文件。摘要描述了蓝图安装的资源，而自述文件则详细解释了蓝图并提供了如何使用蓝图的说明。

标准化项目自定义蓝图 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，这样您就可以看到标准在各个项目中的应用情况。蓝图的生命周期管理使您可以集中管理每个项目的软件开发生命周期，从而确保您所在领域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 以蓝图作者的身份参与生命周期管理。

自定义蓝图提供了通过重新合成对照先前项目更新蓝图版本的功能。Resynthesis 是使用更新的版本重新运行蓝图合成的过程，或者能够将修复和更改合并到现有项目中。有关更多信息，请参阅 自定义蓝图概念。

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
- 自定义蓝图入门
- 教程：创建和更新 React 应用程序
- 以蓝图作者的身份参与生命周期管理
- 开发自定义蓝图以满足项目要求
- 将自定义蓝图发布到空间
- 查看自定义蓝图的详细信息、版本和项目
- 向太空目录添加自定义蓝图
- 从太空目录中移除自定义蓝图
- 为自定义蓝图设置发布权限
- 更改自定义蓝图的目录版本
- 删除已发布的自定义蓝图或版本
- 处理依赖关系、不匹配和工具

- 投稿

自定义蓝图概念

以下是您在中使用自定义蓝图时应了解的一些概念和术语。 CodeCatalyst

主题

- 蓝图项目
- 太空蓝图
- 太空蓝图目录
- 合成
- 重新合成
- 部分期权
- Projen

蓝图项目

蓝图项目使您能够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程版本，则会在蓝图中使用蓝图生成器蓝图创建发布工作流程。该工作流程会自动发布您的最新版本。

太空蓝图

导航到空间的蓝图部分时，您可以查看和管理太空蓝图表中的所有蓝图。蓝图发布到您的空间后，它们将作为空间蓝图提供，供您从空间的蓝图目录中添加和删除。您还可以在空间的 “蓝图” 部分管理发布权限并删除蓝图。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。

太空蓝图目录

您可以从空间的蓝图目录中查看所有添加的自定义蓝图。在这里，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，后者已经为所有空间成员提供了可用的蓝图。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 项目包的过程，该包代表项目中的源代码、配置和资源。然后，CodeCatalyst 部署 API 操作使用该捆绑包部署到项目中。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，而无需在中 CodeCatalyst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
```

蓝图首先在合并了该选项的情况下调用主`blueprint.ts`类`defaults.json`。将在该`synth/synth.`*`[options-name]`*`/proposed-bundle/`文件夹下生成一个新的项目包。输出包括自定义蓝图根据您设置的选项生成的项目包，包括您可能已配置的部分选项。

重新合成

Resynthesis 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。您还可以在中定义所有权边界，`.ownership-file`以指定允许在代码库的哪些部分更新蓝图。虽然自定义蓝图可以提出更新建议`.ownership-file`，但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 resynthesis，并在发布自定义蓝图之前进行测试和更新。使用以下命令执行重新合成：

```
yarn blueprint:resynth             # fast mode
yarn blueprint:resynth --cache     # wizard emulation mode
```

蓝图首先在合并了该选项的情况下调用主`blueprint.ts`类`defaults.json`。将在该`synth/resynth.`*`[options-name]`*`/`文件夹下生成一个新的项目包。输出包括自定义蓝图根据您设置的选项生成的项目包，包括您可能已配置的部分选项。

以下内容是在合成和再合成过程之后创建的：

- proposed-bundle-使用目标蓝图版本的新选项运行合成时的输出。
- e@@ xisting-bundle-对你现有项目的模拟。如果此文件夹中没有任何内容，则生成该文件夹的输出与他相同`proposed-bundle`。
- ancestor-bundle-模拟使用先前版本、先前选项或组合运行蓝图时会生成的内容。如果此文件夹中没有任何内容，则会使用与相同的输出生成`proposed-bundle`。
- r@@ esolved-bund le-始终会重新生成捆绑包，并且默认为、和之间的三向合并。`proposed-bundle` `existing-bundle` `ancestor-bundle`此捆绑包提供了重新合成将在本地输出内容的模拟。

要了解有关蓝图输出包的更多信息，请参阅生成带有重新合成功能的文件。

部分期权

你可以在下方添加选项变体src/wizard-configuration/，这些变体不必枚举整个Options界面，而且这些选项会合并到defaults.json文件顶部。这使您可以跨特定选项定制测试用例。

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 是一个开源工具，自定义蓝图使用它来保持更新和一致性。蓝图以 Projen 包的形式出现，因为此框架使您能够构建、捆绑和发布项目，并且您可以使用该界面来管理项目的配置和设置。

即使蓝图已创建，您也可以使用 Projen 大规模更新蓝图。Projen 工具是生成项目包的蓝图合成背后的底层技术。Projen 拥有项目的配置，它不应该影响你作为蓝图作者。添加依赖关系后，您可以运行`yarn projen`重新生成项目的配置，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 页面。要了解有关使用 Projen 的更多信息，请参阅 Projen 文档和如何使用 Projen 简化项目设置。

自定义蓝图入门

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，该项目默认包含一个用于发布到空间蓝图目录的工作流程。

在配置自定义蓝图的详细信息时，您还可以选择将蓝图的源代码存储在第三方存储库中，您仍然可以在其中管理自定义蓝图，并利用生命周期管理功能在修改自定义蓝图时保持空间的项目同步。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 以蓝图作者的身份参与生命周期管理。

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 管理员指南》中的管理账单。
- 要创建自定义蓝图，您需要拥有 Space 管理员或 Power 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：在中创建自定义蓝图 CodeCatalyst

当您根据空间的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. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要创建自定义蓝图的空间。
3. 在太空仪表板上，选择设置选项卡，然后选择蓝图。
4. 选择创建蓝图。
5. 在“为蓝图命名”下，输入要分配给项目的名称及其关联的资源名称。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。或者，您可以选择将蓝图的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。

   根据您要使用的第三方存储库提供商，执行以下任一操作：

   - 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 人际关系 和 通过人际关系访问 GitHub 资源。

   - Bitbucke t：连接 Bitbucket 工作空间。

     选择“高级”下拉菜单，选择 Bitbucket 作为存储库提供程序，然后选择要存储蓝图创建的源代码的 Bitbucket 工作空间。
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，输入一个将出现在空间蓝图目录中的名称。
   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，发布工作流程是使用项目中名为 Blueprint Builder 的发布蓝图生成的。由于发布工作流程启用了发布权限，因此当您推送更改时，工作流程会将最新的蓝图版本发布到您的空间。要关闭工作流程生成，请取消选中 “发布工作流程” 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目录。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 “查看工作流程”。

9. 选择创建蓝图。

如果您没有关闭自定义蓝图的工作流程生成，则在创建蓝图时，工作流程会自动开始运行。工作流程运行完成后，您的自定义蓝图可以默认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 为自定义蓝图设置发布权限 和 运行工作流。

由于名为的发布工作流程`blueprint-release`是使用蓝图创建的，因此可以在项目中将该蓝图作为已应用的蓝图找到。有关更多信息，请参阅 在项目中应用蓝图以添加资源 和 取消蓝图与项目的关联以停止更新。

第 2 步：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，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，请考虑这些包可能带来的风险。您对添加到空间中的自定义蓝图及其生成的代码负责。

在配置蓝图代码之前，使用支持的集成开发环境 (IDE) 在 CodeCatalyst 项目中创建开发环境。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，执行以下任一操作：

   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. 选择 “代码”，选择 “源存储库”，然后选择您在创建蓝图时创建的存储库。

2. 选择创建开发环境。
3. 从下拉菜单中选择支持的 IDE。有关更多信息，请参阅开发环境支持的集成开发环境。
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，创建开发环境后，“开发环境”状态列会显示“启动”，状态列显示为“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 的安装文档。
2. 开发您的自定义蓝图，使其根据您的喜好进行配置。您可以通过添加组件来修改蓝图的向导。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。预览版使您能够在蓝图用于创建新项目或将其应用于现有项目之前检查其是否符合您的需求。

预览自定义蓝图

1. 在工作终端中，使用以下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，则可以将其发布到您的空间。这允许您在将非预览版本添加到目录之前以用户身份查看蓝图。预览版允许您在不占用实际版本的情况下进行发布。例如，如果您正在处理某个`0.0.1`版本，则可以发布和添加预览版本，这样就可以将第二个版本的新更新发布并添加为`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。此链接是在中运行`yarn`命令时提供的步骤 3：预览自定义蓝图。

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 向太空目录添加自定义蓝图。

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其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将其应用于现有项目。您可以继续对蓝图进行更改，然后通过拉取请求将其作为更新提供。

本教程从蓝图作者的视角和蓝图用户的角度提供了演练。本教程展示了如何创建 React 单页 Web 应用程序蓝图。然后使用该蓝图来创建新项目。当使用更改更新蓝图时，根据蓝图创建的项目会通过拉取请求合并这些更改。

主题

先决条件

要创建和更新自定义蓝图，必须设置并登录 CodeCatalyst按以下方式完成中的任务：

- 拥有用于登录的 AWS 生成器 ID CodeCatalyst。
- 属于某个空间，并在该空间中为您分配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

创建自定义蓝图时，会创建一个包含蓝图源代码以及开发工具和资源的 CodeCatalyst 项目。您的项目是开发、测试和发布蓝图的地方。

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. 选择 “设置” 以导航到空间设置。
4. 在空间设置选项卡中，选择蓝图并选择创建蓝图。
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. 或者，选择查看代码以预览蓝图的蓝图源代码。同样，选择查看工作流程以预览将在构建和发布蓝图的项目中创建的工作流程。
7. 选择创建蓝图。
8. 蓝图创建完成后，您将被带到蓝图的项目。该项目包含蓝图源代码以及开发、测试和发布蓝图所需的工具和资源。发布工作流程已生成，它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对其进行配置。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. 保留默认设置并选择 “创建”。
12. 在 AWS Cloud9 终端中，通过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 应用程序蓝图生成一个新文件夹。

    通过运行以下命令删除静态资产文件夹：

    ```
    rm -r static-assets
    ```

    AWS Cloud9 是在基于 Linux 的平台上构建的。如果你使用的是 Windows 操作系统，则可以改用以下命令：

    ```
    rmdir /s /q static-assets
    ```

14. 既然默认文件夹已删除，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
    ```

    如果出现提示，请输入`y`以继续。

    在该`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。需要将更改推送到您的远程 CodeCatalyst 源代码库。

15. 确保您有最新的更改，然后通过运行以下命令提交更改并将其推送到蓝图的 CodeCatalyst 源存储库：

    ```
    git pull
    ```

    ```
    git add .
    ```

    ```
    git commit -m "Add React app to static-assets"
    ```

    ```
    git push
    ```

将更改推送到蓝图的源存储库时，发布工作流程将自动启动。此工作流程会增加蓝图版本，构建蓝图并将其发布到您的空间。在下一步中，您将导航到发布工作流程运行以查看其运行情况。

第 2 步：查看发布工作流程

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. 选择蓝图发布工作流程。
3. 您可以看到工作流程包含构建和发布蓝图的操作。
4. 在“最新运行”下，选择工作流程运行链接以查看您所做的代码更改中的运行情况。
5. 运行完成后，您的新蓝图版本即会发布。已发布的蓝图版本可以在空间的“设置”中查看，但在将其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，您将把蓝图添加到目录中。

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，即可在空间中的所有项目中使用该蓝图。空间成员可以使用蓝图创建新项目或将其应用于现有项目。

1. 在 CodeCatalyst 控制台中，导航回空间。
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。

第 4 步：使用蓝图创建项目

现在，蓝图已添加到目录中，可以在项目中使用。在此步骤中，您将使用刚刚创建的蓝图创建一个项目。在后面的步骤中，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“太空蓝图”，然后选择react-app-blueprint。

> **Note**
>
> 选择蓝图后，您可以看到蓝图`README.md`文件的内容。

3. 选择下一步。
4.

> **Note**
>
> 可以在蓝图中配置此项目创建向导的内容。

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，您将用它来更新此项目。

步骤 5：更新蓝图

在使用蓝图创建新项目或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。然后可以将新版本添加为目录版本。

1. 导航到中创建的react-app-blueprint项目教程：创建和更新 React 应用程序。
2. 打开在中创建的开发环境教程：创建和更新 React 应用程序。
   a. 在导航窗格中，选择 “代码”，然后选择 “开发环境”。
   b. 从表格中找到 “开发环境”，然后选择 “打开方式” AWS Cloud9 （在浏览器中）。
3. 运行蓝图发布工作流程时，它通过更新`package.json`文件来增加蓝图版本。通过在 AWS Cloud9 终端中运行以下命令来提取更改：

```
git pull
```

4. 通过运行以下命令导航到该`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。如果你使用的是 Windows 操作系统，则可以改用以下命令：

```
echo > hello-world.text
```

6. 在左侧导航栏中，双击`hello-world.txt`文件以在编辑器中将其打开，然后添加以下内容：

```
Hello, world!
```

保存该文件。

7. 确保您有最新的更改，然后通过运行以下命令提交更改并将其推送到蓝图的 CodeCatalyst 源存储库：

```
git pull
```

```
git add .
```

```
git commit -m "prettier setup"
```

```
git push
```

推送更改启动了发布工作流程，该工作流程将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将其应用于现有项目后，您仍然可以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”。

第 7 步：使用新的蓝图版本更新项目

该空间的蓝图目录中现已推出新版本。作为蓝图用户，您可以更新在中创建的项目的版本第 4 步：使用蓝图创建项目。这样可以确保您获得符合最佳实践所需的最新更改和修复。

1. 在 CodeCatalyst 控制台中，导航到中创建的react-app-project项目第 4 步：使用蓝图创建项目。
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 A pply up date 将在项目中创建拉取请求，其中包含更新后的蓝图版本中的更改。要对项目进行更新，必须合并拉取请求。有关更多信息，请参阅 查看拉取请求 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”。

第 8 步：查看项目中的更改

之后，对蓝图的更改现在可以在您的项目中使用第 7 步：使用新的蓝图版本更新项目。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，您可以查看在中创建的`hello-world.txt`文件步骤 5：更新蓝图。
3. 选择`hello-world.txt`可查看文件内容。

生命周期管理使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，您可以推送蓝图的更新，然后使用该蓝图创建新项目或将其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 以蓝图作者的身份参与生命周期管理。

以蓝图作者的身份参与生命周期管理

生命周期管理允许您通过单一通用的最佳实践来源使大量项目保持同步。这可以扩展修复程序的传播以及任意数量项目在整个软件开发生命周期中的维护。生命周期管理简化了内部活动、安全修复、审计、运行时升级、最佳实践的变更以及其他维护实践，因为这些标准是在一个地方定义的，并且在新标准发布时会自动集中保持最新状态。

发布蓝图的新版本时，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出于合规性目的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创建新资源。用户可以自由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版本（包括选项）。要了解如何以蓝图用户身份使用生命周期管理，请参阅对现有项目使用生命周期管理和对项目中的多个蓝图使用生命周期管理。

主题

- 测试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。将在synth/目录下生成一系列代表生命周期更新的各个阶段的捆绑包。要测试生命周期管理，你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，请参阅和。重新合成 生成带有重新合成功能的文件

使用合并策略生成捆绑包并指定文件

主题

- 生成带有重新合成功能的文件
- 使用合并策略
- 为生命周期管理更新指定文件
- 编写合并策略

生成带有重新合成功能的文件

Resynthesis 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，从而允许将对蓝图的更改传播到现有项目。通过resynth()函数跨蓝图输出包进行合并。Resynthesis 首先生成三个包，分别代表蓝图和项目状态的不同方面。可以使用yarn blueprint:resynth命令在本地手动运行它，如果捆绑包尚不存在，则该命令将创建捆绑包。手动使用捆绑包将允许您在本地模拟和测试重新合成行为。默认情况下，蓝图仅在下的存储库中运行重新合成，src/*因为通常只有捆绑包的该部分受源代码控制。

- existing-bundle-此捆绑包代表现有项目状态。这是由合成计算人为构造的，目的是为其部署到的项目中的内容（如果有的话）提供蓝图上下文。如果在本地运行 resynthess 时此位置已经存在某些内容，则它将被重置并视为模拟。否则，它将设置为的内容ancestor-bundle。
- ancestor-bundle-如果蓝图输出是用一些以前的选项和/或版本合成的，则此捆绑包代表蓝图输出。如果这是首次将此蓝图添加到项目中，则其祖先不存在，因此将其设置为与相同的existing-bundle内容。在本地，如果此分发包已存在于此位置，则会将其视为模拟。
- proposed-bundle-如果蓝图是用一些新选项和/或版本合成的，则这个捆绑包会模拟蓝图。这与该synth()函数生成的捆绑包相同。在本地，此捆绑包始终被覆盖。

每个捆绑包都是在重新合成阶段创建的，可以从下的蓝图类中访问该包。`this.context.resynthesisPhase`

- `resolved-bundle`-这是最后一个捆绑包，它代表了打包并部署到 CodeCatalyst 项目中的内容。您可以查看哪些文件和差异已发送到部署机制。这是解析其他三个捆绑包之间合并的`resynth()`函数的输出。

应用三向合并的方法是取`ancestor-bundle`和之间的差异，`proposed-bundle`然后将其应用于`existing-bundle`以生成。`resolved-bundle`所有合并策略都将文件解析为`resolved-bundle`。Resynthesis 使用蓝图的合并策略解析这些捆绑包的覆盖范围，`resynth()`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。这些策略提供了解决本生成带有重新合成功能的文件节中提到的文件输出和文件冲突的方法。

- `alwaysUpdate`-一种始终解析为建议文件的策略。
- `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文件的策略。否则，解析为现有文件。
- `threeWayMerge`-一种在现有的、提议的和共同的祖先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。提供的文件内容必须采用 UTF-8 编码，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，则始终返回建议的文件。
- `preferProposed`-一种在现有的、提议的和共同的祖先文件之间进行三向合并的策略。此策略通过选择每个冲突中建议文件的一方来解决冲突。
- `preferExisting`-一种在现有的、提议的和共同的祖先文件之间进行三向合并的策略。此策略通过选择每个冲突的现有文件一方来解决冲突。

要查看合并策略的源代码，请参阅开源 GitHub 存储库。

为生命周期管理更新指定文件

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。但是，您可能不想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旨在针对特定项目。如果您未指定其他策略，则三向合并策略是默认选项。蓝图可以通过在存储库构造本身上指定合并策略来指定他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，并且可以在重新合成期间使用最新的策略。

```
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，最后一个策略优先。未发现的文件默认为与 Git three-way-merge 类似。通过MergeStrategies构造提供了几种合并策略，但你可以自己编写。提供的策略符合 git 合并策略驱动程序。

编写合并策略

除了使用提供的构建合并策略之一外，您还可以编写自己的策略。策略必须遵循标准的策略接口。您必须编写一个策略函数，用于从、和existing-bundle、中获取文件版本 proposed-bundleancestor-bundle，并将它们合并到单个已解析文件中。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，以获取空间和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了解您是要重新同步以生成祖先捆绑包还是建议的捆绑包。然后，可以使用现有的代码上下文来转换仓库中的代码。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`blueprint.ts`的文件中，也可以为策略创建单独的文件。

以下示例说明如何在项目上下文中查找文件、设置工作流程生成器以及如何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、工作流程、议题或可在中创建的任何其他组件 CodeCatalyst。

> **Important**
>
> 如果要使用来自外部来源的蓝图包，请考虑这些包可能带来的风险。您对添加到空间中的自定义蓝图及其生成的代码负责。

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，则必须先创建一个开发环境。有关更多信息，请参阅 创建开发环境。

2. 在你的开发环境中打开一个可以正常工作的终端。
3. 如果您在创建蓝图时选择了发布工作流程，则会自动发布最新的蓝图版本。提取更改以确保package.json文件具有增量版本。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界面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、向蓝图添加工作流程组件 和 向蓝图添加开发环境组件。

   在开发自定义蓝图时，您还可以查看蓝图 SDK 和示例蓝图，以获得更多支持。有关更多信息，请参阅开源 GitHub 存储库。

成功合成后，自定义蓝图会提供预览包。项目包表示项目中的源代码、配置和资源， CodeCatalyst 部署 API 操作使用它来部署到项目中。如果要继续开发自定义蓝图，请重新运行蓝图合成过程。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

blueprint.ts文件中的Options界面会自动生成开启的蓝图选择向导。 CodeCatalyst 前端向导支持Options使用 JSDOC 风格的注释和标签对蓝图进行修改和功能。你可以使用 JSDOC 风格的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，或者使选项可折叠。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)。该向导会自动将自己配置为所描述的最佳类型。并非所有类型都受支持。其他支持的类型包括区域选择器和环境选择器。

以下是使用带有蓝图的 JSDOC 注释和标签的Options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界面每个选项的显示名称都显示camelCase在中。JSDOC 样式注释中的纯文本显示为向导中选项上方的文本。

主题

- 支持的标签
- 支持的 TypeScript 类型
- 在合成过程中与用户沟通

支持的标签

前端向导Options中的自定义蓝图支持以下 JSDOC 标签。

@inlinePolicy。 /路径/to/policy/file.json

- 需要-成为类型的选项Role。
- 用法-允许您传达角色所需的内联策略。该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。
- 依赖关系-blueprint-cli 0.1.12 及以上
- 示例-@inlinePolicy ./deployment-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @inlinePolicy ./path/to/deployment-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@trustPolicy。 /路径/to/policy/file.json

- 需要-成为类型的选项Role。
- 用法-允许您传达角色所需的信任策略。该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。
- 依赖关系-blueprint-cli 0.1.12 及以上
- 示例-@trustPolicy ./trust-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @trustPolicy ./path/to/trust-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@validationRegex 正则表达式

- 需要-选项为字符串。
- 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-搭配使用@validationMessage。默认情况下，验证消息为空。

@validationMessage 字符串

- 需要-@validationRegex 或其他错误才能查看使用情况。
- 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-搭配使用@validationMessage。默认情况下，验证消息为空。

@collapsed 布尔值（可选）

- 需要-不适用

- 用法-允许子选项可折叠的布尔值。如果存在折叠的注释，则其默认值为 true。将该值设置为可`@collapsed false`创建最初处于打开状态的可折叠部分。
- 示例-`@collapsed true`

@displayName 字符串

- 需要-不适用
- 用法-更改选项的显示名称。允许使用除 camelCase 以外的格式作为显示名称。
- 示例-`@displayName Blueprint Name`

@displayName 字符串

- 需要-不适用
- 用法-更改选项的显示名称。允许使用除 c amelCase 以外的格式作为显示名称。
- 示例-`@displayName Blueprint Name`

@defaultEntropy 数字

- 需要-选项为字符串。
- 用法-将指定长度的随机字母数字字符串附加到选项中。
- 示例-`@defaultEntropy 5`

@placeholder 字符串（可选）

- 需要-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）

- 需要-不适用
- 用法-将字符串输入转换为文本区域组件，用于较大的文本部分。添加数字定义行数。默认值为五行。
- 示例-`@textArea 10`

@hidden 布尔值（可选）

- 需要-不适用
- 用法-除非验证检查失败，否则不向用户隐藏文件。默认值为 true。
- 示例-@hidden

@button 布尔值（可选）

- 需要-不适用
- 用法-注释必须位于布尔属性上。添加一个按钮，选择后该按钮将合成为 true。不是切换。
- 示例-buttonExample: boolean;

```
/**
  * @button
  */
buttonExample: boolean;
```

@showName 布尔值（可选）

- 需要-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）

- 需要-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。所有连接默认为production。选项为 “非生产” 或 “生产”。
- 示例-@showEnvironmentType true

```
/**
  * @showEnvironmentType true
  */
accountConnection: AccountConnection<{
    ...
}>;
```

@forceDefault 布尔值（可选）

- 需要-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 蓝图名称

- 需要-为界面`Options`添加注释
- 用法-警告用户`blueprintName`将指定的项目作为当前蓝图的要求进行应用。
- 示例-`@requires '@amazon-codecatalyst/blueprints.blueprint-builder'`

```
/*
 * @requires '@amazon-codecatalyst/blueprints.blueprint-builder'
 */
export interface Options extends ParentOptions {
...
```

支持的 TypeScript 类型

前端向导`Options`中的自定义蓝图支持以下 TypeScript 类型。

数字

- 需要-成为类型的选项`number`。

- 用法-生成数字输入字段。
- 示例-`age: number`

```
{
  age: number
  ...
}
```

String

- 需要-成为类型的选项`string`。
- 用法-生成字符串输入字段。
- 示例-`name: string`

```
{
  age: string
  ...
}
```

字符串列表

- 需要-成为类型的选项`boolean`。
- 用法-生成一个复选框。
- 示例-`isProduction: boolean`

```
{
  isProduction: boolean
  ...
}
```

电台

- 需要-选项是三个或更少字符串的并集。
- 用法-生成选定的电台。

> **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

- 需要-选项是四个或更多字符串的并集。
- 用法-生成下拉列表。
- 示例-`runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'`

```
{
  runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'
  ...
}
```

可扩展部分

- 需要-成为对象的选项。
- 用法-生成可扩展部分。对象中的选项将嵌套在向导的可扩展部分中。
- 示例-

```
{
     expandableSectionTitle: {
         nestedString: string;
         nestedNumber: number;
     }
}
```

元组

- 需要-选择为类型Tuple。
- 用法-生成键值付费输入。
- 示例-tuple: Tuple[string, string]>

```
{
    tuple: Tuple[string, string]>;
    ...
}
```

元组列表

- 需要-选项是类型的数组Tuple。
- 用法-生成元组列表输入。
- 示例-tupleList: Tuple[string, string]>[]

```
{
  tupleList: Tuple[string, string]>[];
  ...
}
```

Selector

- 需要-选择为类型Selector。
- 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>;
  ...
}
```

多选

- 需要-选择为类型Selector。
- 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沟通

作为蓝图作者，您可以与用户进行反馈，而不仅仅是验证消息。例如，空间成员可能会查看生成不清楚的蓝图的选项组合。自定义蓝图支持通过调用合成向用户传达错误消息的功能。基础蓝图实现了一个需要明确错误消息的throwSynthesisError(...)函数。您可以使用以下方法调用消息：

```
//blueprint.ts
this.throwSynthesisError({
   name: BlueprintSynthesisErrorTypes.BlueprintSynthesisError,
   message: 'hello from the blueprint! This is a custom error communicated to the
 user.'
})
```

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
- 模拟接口示例

创建开发环境

以下示例显示了如何将应用程序部署到云端：

```
export interface Options extends ParentOptions {
        ...
        myNewEnvironment:  EnvironmentDefinition{
            thisIsMyFirstAccountConnection: AccountConnection{
                thisIsARole: Role['lambda', 's3', 'dynamo'];
             };
        };
    }
```

该接口生成一个 UI 组件，该组件要求使用单一账户连接的新环境 (myNewEnvironment) (thisIsMyFirstAccountConnection. 还会在账户连接 (thisIsARole) 上生成一个角色，其['lambda', 's3', 'dynamo']角色能力是最低要求的角色。并非所有用户都有账户关联，因此您应该检查用户未关联账户或未将账户与角色关联的情况。也可以用注释角色。@inlinePolicies有关更多信息，请参阅 @inlinePolicy。 /路径/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形状：

```
{
  ...
  "myNewEnvironment": {
    "name": "myProductionEnvironment",
    "environmentType": "PRODUCTION"
  },
}
```

然后，用户界面组件会提示您输入各种字段。填写字段时，蓝图的形状会完全展开。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。

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"
        ]
      }
    }
  },
}
```

向蓝图添加机密组件

密钥可用于存储可在工作流程中引用的敏感数据。 CodeCatalyst 您可以向自定义蓝图添加密钥并在工作流程中引用该密钥。有关更多信息，请参阅 在工作流程中配置和使用密钥。

要导入 Amazon CodeCatalyst 蓝图，请键入区域类型

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
- 在工作流程中引用密钥

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);
}
```

秘密组件需要name。以下代码是所需的最低默认形状：

```
{
    ...
    "secret": {
        "name": "secretName"
    },

}
```

在工作流程中引用密钥

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程。有关更多信息，请参阅 在工作流程中引用密钥。

```
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(),
    );

}
```

要了解有关在中使用密钥的更多信息 CodeCatalyst，请参阅在工作流程中配置和使用密钥。

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界面中，以便在蓝图向导中生成组件，您可以输入一个或多个 AWS gion。gion 类型可以从blueprint.ts文件中的基础蓝图中导入。有关更多信息，请参阅 AWS 区域。

要导入 Amazon CodeCatalyst 蓝图，请键入区域类型

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，您也可以使用它*来包含所有支持的 AWS 区域。

主题

- 注释
- 区域组件示例

注释

可以向Options界面的每个字段添加 JSDoc 标签，以自定义字段在向导中的显示和行为。对于区域类型，支持以下标签：

- 该@displayName注释可用于在向导中更改字段的标签。

  例如：@displayName AWS Region
- @placeholder注释可用于更改选择/多选组件的占位符。

  例如：@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 使用存储库 CodeCatalyst 来存储代码。存储库以名称作为输入。大多数组件都存储在存储库中，例如源代码文件、工作流程和其他组件，例如托管开发环境 (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存储库。该操作是最常见的用例之一，它需要存储库、文件路径和文本内容。如果存储库中不存在文件路径，则该组件将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同一个存储库中的相同位置，则最新的实现会覆盖前一个实现方案。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特别有用。

添加通用文件

您可以向存储库中写入任意位。您可以从缓冲区读取并使用该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());
```

正在复制文件

您可以通过复制并粘贴起始代码，然后在该基础上生成更多代码来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'
```

的子类StaticAsset是。SubstitutionAsset子类的功能完全相同，但你可以改为对文件进行胡须替换。它可用于执行 copy-and-replace 样式生成。

静态资产替换使用 mustache 模板引擎来渲染植入生成的源存储库的静态文件。Mustache 模板规则是在渲染期间应用的，这意味着默认情况下，所有值都是 HTML 编码的。要呈现未转义的 HTML，请使用三胡子语法。{{{name}}}有关更多信息，请参阅胡子模板规则。

> **Note**
>
> 对无法用文本解释的文件运行替换可能会产生错误。

```
const starterCodeText = new SubstitionAsset('path/to/file/file.txt').subsitite({
  'my_variable': 'subbed value1',
  'another_variable': 'subbed value2'
})
```

定位多个文件

静态资源支持通过静态函数进行全局定位，StaticAsset并调用其子类findAll(...)，该函数返回预加载了路径、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程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中添加文件和工作流程：

```
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myRepo一起生成一个名为的存储库，其根目录为源文件README.md和CodeCatalyst工作流程。

向蓝图添加工作流程组件

Amazon CodeCatalyst 项目使用工作流程根据触发器运行操作。您可以使用工作流程组件来构建和整理工作流程 YAML 文件。有关更多信息，请参阅 工作流程 YAML 定义。

导入 Amazon CodeCatalyst 蓝图工作流程组件

在您的blueprint.ts文件中，添加以下内容：

```
import { WorkflowBuilder, Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
```

主题

- 工作流程组件示例
- 连接到环境

工作流程组件示例

WorkflowBuilder 组件

您可以使用类来构建工作流程定义。可以为工作流组件提供定义，以便在存储库中进行渲染。

```
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),
});
```

工作流程 Projen 组件

以下示例显示了如何使用 Projen 组件将工作流程 YAML 写入存储库：

```
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

许多工作流程需要在 AWS 账户连接中运行。工作流通过允许操作连接到具有帐户和角色名称规范的环境来处理此问题。

```
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

托管开发环境 (MDE) 用于在中创建和建立 MDE 工作空间。 CodeCatalyst该组件生成一个`devfile.yaml`文件。有关更多信息，请参阅 Devfile 简介和。编辑开发环境的存储库 devfile

```
new Workspace(this, repository, SampleWorkspaces.default);
```

导入 Amazon CodeCatalyst 蓝图工作空间组件

在您的blueprint.ts文件中，添加以下内容：

```
import {...} from '@amazon-codecatalyst/codecatalyst-workspaces'
```

向蓝图添加议题组件

在中 CodeCatalyst，您可以监视项目中涉及的功能、任务、错误和任何其他工作。每件作品都保存在一个不同的记录中，称为问题。每个问题都可以有描述、受托人、状态和其他属性，您可以对其进行搜索、分组和筛选。您可以使用默认视图查看问题，也可以使用自定义筛选、排序或分组创建自己的视图。有关与议题相关的概念的更多信息，请参阅问题概念和中的问题配额 CodeCatalyst。

议题组件生成议题的 JSON 表示形式。该组件将 ID 字段和问题定义作为输入。

导入 Amazon CodeCatalyst 蓝图问题组件

在您的blueprint.ts文件中，添加以下内容：

```
import {...} from '@amazon-codecatalyst/blueprint-component.issues'
```

主题

- 问题组件示例

问题组件示例

创建议题

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myFirstIssue', {
  title: 'myFirstIssue',
  content: 'This is an example issue.',
});
```

创建高优先级问题

```
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问题

```
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 提供了用于管理和使用您的自定义蓝图的工具。

主题

- 使用蓝图工具
- 图片上传工具

使用蓝图工具

使用蓝图工具

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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，则必须先创建一个开发环境。有关更多信息，请参阅创建开发环境。
3. 在正常运行的终端中，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CodeCatalyst blueprints GitHub repository前往查找要使用的工具的名称。

如果要使用图片上传工具，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';
```

- 如果要使用图片上传工具，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. 调用该函数。

示例：

- 如果要使用发布功能，请在脚本中添加以下内容：

```
await publish(logger, config.publishEndpoint, {<your publishing options>});
```

- 如果要使用图片上传工具，请在脚本中添加以下内容：

```
const {imageUrl, imageName} = await uploadImagePublicly(logger, 'path/to/
image'));
```

图片上传工具

图像上传工具使您能够将自己的图像上传到您的 AWS 账户中的 S3 存储桶，然后在后面公开分发该图片 CloudFront。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，并返回公开可用的图像的网址。有关更多信息，请参阅 Amazon CloudFront 是什么？ 还有什么是 Amazon S3？

使用图片上传工具

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。在工作终端中，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. 运行以下命令安装依赖项：

```
yarn && yarn build
```

4. 运行以下命令以确保安装了最新的蓝图 CLI 版本：

```
yarn upgrade @amazon-codecatalyst/blueprint-util.cli
```

5. 使用您要将图像上传到的 S3 存储桶登录 AWS 账户。有关更多信息，请参阅配置 AWS CLI 和通过 AWS 命令行界面登录。
6. 从 CodeCatalyst 仓库的根目录运行以下命令，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>
```

您的图片的网址已生成。由于部署 CloudFront 发行版需要一些时间，因此该网址不会立即可用。检查分发状态以获取最新的部署状态。有关更多信息，请参阅使用发行版。

通过快照测试评估接口变化

支持在蓝图的多种配置中生成的快照测试。

蓝图支持对您作为蓝图作者提供的配置进行快照测试。这些配置是部分替换，它们合并到蓝图根目录下的 defaults.json 文件之上。启用和配置快照测试后，生成和测试过程将合成给定的配置，并验证合成后的输出是否与参考快照相比没有变化。要查看快照测试代码，请参阅CodeCatalyst 蓝图 GitHub 存储库。

启用快照测试

1. 在.projenrc.ts文件中， ProjenBlueprint 使用要快照的文件将输入对象更新为。例如：

```
{
  ....
  blueprintSnapshotConfiguration: {
    snapshotGlobs: ['**', '!environments/**', '!aws-account-to-environment/**'],
  },
}
```

2. 重新同步蓝图以在蓝图项目中创建 TypeScript 文件。不要编辑源文件，因为它们由 Projen 维护和重新生成。使用以下命令：

```
yarn projen
```

3. 导航到该src/snapshot-configurations目录以查看包含空对象的default-config.json文件。使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defaults.json文件合并，进行合成，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，会显示以下消息：Snapshot Summary › NN snapshots written from 1 test suite。随后的测试运行验证合成的输出是否与快照相比没有变化，并显示以下消息:Snapshots: NN passed, NN total.

如果您故意更改蓝图以生成不同的输出，请运行以下命令来更新参考快照：

```
yarn test:update
```

快照期望合成输出在每次运行之间保持不变。如果您的蓝图生成的文件各不相同，则必须将这些文件排除在快照测试之外。更新`ProjenBluerpint`输入`blueprintSnapshotConfiguration`对象的对象以添加该`snapshotGlobs`属性。该`snapshotGlobs`属性是一组 glob，用于确定快照中包含或排除哪些文件。

> **Note**
>
> 有一个默认的 glob 列表。如果您指定自己的列表，则可能需要显式恢复默认条目。

将自定义蓝图发布到空间

在将自定义蓝图发布到空间的蓝图目录之前，必须将其发布到空间。您也可以在发布之前在 CodeCatalyst 控制台中查看蓝图。您可以发布蓝图的预览版或普通版。

> **Important**
>
> 如果要使用来自外部来源的蓝图包，请考虑这些包可能带来的风险。您对添加到空间中的自定义蓝图及其生成的代码负责。

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普通版本
- 在指定空间和项目中发布和应用自定义蓝图

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，则可以将其发布到您的空间。这允许您在将非预览版本添加到目录之前以用户身份查看蓝图。预览版允许您在不占用实际版本的情况下进行发布。例如，如果您正在处理某个`0.0.1`版本，则可以发布和添加预览版本，这样就可以将第二个版本的新更新发布并添加为`0.0.2`。

进行更改后，通过运行`package.json`文件来重建自定义蓝图的软件包，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. 在你的开发环境中打开一个可以正常工作的终端。
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，然后将其添加到空间的蓝图目录中。导航到Enable version *[preview version number]* at:链接，将预览版发布到您的空间。

无需在中创建项目即可模拟项目创建。 CodeCatalyst要合成您的项目，请使用以下命令：

```
yarn blueprint:synth
```

蓝图将在synth/synth.*[options-name]*/proposed-bundle/文件夹中生成。有关更多信息，请参阅 合成。

如果您要更新自定义蓝图，请使用以下命令重新同步您的项目：

```
yarn blueprint:resynth
```

蓝图将在synth/synth.*[options-name]*/proposed-bundle/文件夹中生成。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，以便空间成员可以使用它来创建新项目或在现有项目中应用。有关更多信息，请参阅 向太空目录添加自定义蓝图。

查看和发布自定义蓝图的普通版本

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蓝图目录的新版本。创建项目时生成的发布工作流程会自动发布已推送的更改。如果您在创建蓝图时关闭了工作流程生成，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，您仍然可以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. 在你的开发环境中打开一个可以正常工作的终端。
3. • 如果您在创建蓝图时选择退出发布工作流程生成，请使用以下命令：

   ```
   yarn blueprint:release
   ```

   您仍然可以导航到提供的`See this blueprint at:`链接以查看您的自定义蓝图。

   发布自定义蓝图的更新版本，然后将其添加到空间的蓝图目录中。导航到`Enable version [release version number] at:`链接，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程，则在推送更改时会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
   ```

发布普通版本后，您可以添加蓝图，以便空间成员可以使用它来创建新项目或在现有项目中应用。有关更多信息，请参阅 向太空目录添加自定义蓝图。

在指定空间和项目中发布和应用自定义蓝图

默认情况下，`blueprint:preview`和`blueprint:release`命令会发布到您创建蓝图的 CodeCatalyst 空间中。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。您也可以将蓝图应用于另一个空间的现有项目。

在指定空间中发布或应用自定义蓝图

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. 在你的开发环境中打开一个可以正常工作的终端。
3. （可选）如果您尚未安装项目所需的依赖项，请安装它们。使用以下命令：

```
yarn
```

4. 使用--space标签将预览版或普通版发布到指定空间。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本应用于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
```

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或应用该蓝图的项目。

主题

查看空间的自定义蓝图

查看空间的自定义蓝图

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要查看自定义蓝图的空间。
3. 在太空仪表板上，选择设置选项卡，然后选择蓝图以查看太空蓝图。表格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。
   - 最新修改时间-空间蓝图上次更新的日期。

查看使用自定义蓝图创建或应用自定义蓝图的项目

查看使用自定义蓝图创建或应用自定义蓝图的项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要查看自定义蓝图的空间。
3. 在太空仪表板上，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表的项目。

向太空目录添加自定义蓝图

将自定义蓝图发布到空间后，可以将其添加到空间的蓝图目录中。如果您将自定义蓝图添加到 CodeCatalyst 空间的蓝图目录中，则该蓝图可供所有空间成员在创建项目或将其应用于现有项目时使用。在将自定义蓝图添加到空间的蓝图目录之前，必须启用蓝图的发布权限。如果您选择了生成工作流程版本，则默认情况下会启用发布权限。有关更多信息，请参阅 为自定义蓝图设置发布权限 和 将自定义蓝图发布到空间。

将蓝图添加到空间的蓝图目录中

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 蓝图只能从源存储库的默认分支中添加。如果您在功能分支上开发蓝图，请将您的功能分支与对默认分支的更改合并。创建拉取请求以合并对默认分支的所有更改。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，使用您的自定义蓝图导航到太空仪表板。
4. 在太空仪表板上，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。

从太空目录中移除自定义蓝图

如果您不想再使用自定义蓝图来创建新项目或将其应用于现有项目，则可以将其从空间的蓝图目录中删除。

> **Note**
>
> 如果您从太空目录中移除自定义蓝图，则不会影响根据该蓝图创建的项目或应用该蓝图的项目。蓝图的资源不会从项目中移除。

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，使用您的自定义蓝图导航到太空仪表板
3. 在太空仪表板上，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。

为自定义蓝图设置发布权限

默认情况下，如果在项目创建期间生成了工作流程版本，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，这样蓝图就无法发布。禁用权限后，蓝图创建期间生成的发布工作流程将无法运行。除非启用蓝图的权限，否则无法发布对蓝图的新更改。

> **Important**
>
> 要启用或禁用蓝图项目的发布权限，您必须具有 Space 管理员角色。

设置蓝图项目的发布权限

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要管理自定义蓝图发布权限的空间。
3. 在太空仪表板上，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，查看所有空间蓝图的发布权限。
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。如果您要启用权限，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。

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蓝图目录的版本。更改蓝图的目录版本不会影响使用不同蓝图版本的项目。

管理自定义蓝图版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要更改自定义蓝图版本的空间。
3. 在太空仪表板上，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。

删除已发布的自定义蓝图或版本

当您从 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资源的所有访问权限都将被移除。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提示的任何工作流程都将停止。

> **Note**
>
> 如果您删除蓝图，则它不会影响应用该蓝图的项目。蓝图的资源不会从项目中移除。

如果蓝图版本已发布到空间的蓝图目录，请在删除已发布版本之前为目录选择一个新版本。

删除自定义蓝图的目录版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要删除自定义蓝图目录版本的空间。
3. 在太空仪表板上，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`以确认选择蓝图目录版本。
8. 选择删除。

如果某个蓝图版本不在空间的蓝图目录中，则可以在不选择新版本的情况下删除该版本。

删除自定义蓝图版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要删除自定义蓝图版本的空间。
3. 在太空仪表板上，选择设置选项卡，然后选择蓝图。
4. 选择带有您要删除的版本的蓝图的名称。
5. 选择要删除的版本对应的单选按钮，然后选择删除版本。
6. 输入`delete`确认删除蓝图版本。
7. 选择删除。

从空间的蓝图目录中删除蓝图会删除该蓝图的所有版本。该空间中使用蓝图的项目不受删除的影响。

删除自定义蓝图版本

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在 CodeCatalyst 控制台中，导航到要删除自定义蓝图的空间。
3. 在太空仪表板上，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（删除）。

处理依赖关系、不匹配和工具

主题

- 添加依赖关系
- 处理依赖类型不匹配的问题
- 使用 yarn 和 npm
- 升级工具和组件

添加依赖关系

作为蓝图作者，您可能需要向蓝图中添加包，例如`@amazon-codecatalyst/blueprint-component.environments`。您需要使用该软件包更新`projen.ts`文件，然后使用 P rojen 重新生成项目的配置。Projen 充当每个蓝图代码库的项目模型，它允许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。该`package.json`文件是由 Projen 模型部分拥有的文件。Projen 承认 package.json 文件中包含的依赖版本，但其他选项需要源自模型。

添加依赖关系和更新**`projenrc.ts`**文件

1. 在`projen.ts`文件中，导航到 deps 部分。
2. 在蓝图中添加要使用的依赖项。
3. 使用以下命令重新生成项目的配置：

```
yarn projen && yarn
```

处理依赖类型不匹配的问题

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
```

该错误是由于一个组件依赖于另一个组件的较新版本，但依赖组件固定到较旧版本时，就会发生依赖关系不匹配。可以通过使所有组件依赖相同的版本来修复该错误，以便版本在它们之间同步。除非你确定如何处理这些版本，否则最好将所有蓝图提供的软件包保持在相同的最新版本 (`0.0.x`) 下。以下示例显示如何配置`package.json`文件，使所有依赖关系都依赖于同一个版本：

```
...
"@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，使用以下命令：

```
yarn install
```

使用 yarn 和 npm

蓝图使用纱线作为工具。使用 npm 和 Yarn 会导致工具问题，因为每种依赖树的解析方式不同。为避免此类问题，最好只使用 Yarn。

如果您不小心使用 npm 安装了依赖项，则可以删除生成的`package-lock.json`文件，并确保使用所需的依赖项更新您的`.projenrc.ts`文件。您可以使用 Projen 重新生成项目的配置。

使用以下内容从模型中再生：

```
yarn projen
```

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以引入可用的新功能。除非你确定如何处理版本，否则建议你将所有组件保持在同一个版本上。版本在组件之间是同步的，因此所有组件的相同版本可确保它们之间存在适当的依赖关系。

使用 Yarn 工作空间 monorepo

使用以下命令从自定义蓝图存储库的根目录升级实用程序和组件：

```
yarn upgrade @amazon-codecatalyst/*
```

如果你没有使用 monorepo，请使用以下命令：

```
yarn upgrade –pattern @amazon-codecatalyst/*
```

可用于升级刀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手动增加到最新版本：。`yarn`所有组件和实用程序都应具有相同的版本。

投稿

蓝图软件开发套件 (SDK) 是一个开源库，你可以为之做出贡献。作为贡献者，请考虑贡献指南、反馈和缺陷。有关更多信息，请参阅蓝图 GitHub 存储库。

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，请参阅的配额 CodeCatalyst。

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。她邀请朋友李娟、萨恩维·萨卡尔和豪尔赫·索萨和她一起参与这个项目。Li Juan 查看了源代码库中的示例代码，并决定进行一些快速更改，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，然后指定一个新的分支，即####。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。然后 Li 创建了一个拉取请求。作为创建拉取请求的一部分，李加入 Jorge Souza 和 Saanvi Sarkar 作为审阅者，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。从 Git 客户端或集成开发环境 (IDE) 克隆和使用源存储库时，您可以创建和使用个人访问令牌 (PAT) 作为应用程序特定的密码。这些 PAT 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存储库访问权限。

CodeCatalyst 源存储库支持拉取请求。这是您和其他项目成员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的简单方法。您可以在 CodeCatalyst 控制台中查看更改并对代码行进行评论。

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项目的一部分使用项目模板创建的，则会将一个或多个工作流程配置为项目的一部分。您可以随时为存储库添加其他工作流程。项目中工作流程的 YAML 配置文件存储在源代码操作中为这些工作流配置的源存储库中。有关更多信息，请参阅 工作流程入门。

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

项目

项目代表一种为开发团队和任务提供支持的协作努力。 CodeCatalyst 创建项目后，您可以添加、更新或删除用户和资源，自定义项目仪表板，并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，也不能将存储库链接到空间中的多个项目。在项目中具有参与者或项目管理员角色的用户可以根据向这些角色授予的权限与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

源存储库是您安全地存储项目代码和文件的地方。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源存储库中的开发文件定义。如果您的源存储库中没有开发文件，则会自动应用默认的开发文件。默认的 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，开发环境配置为具有 2 核处理器、4 GB RAM 和 16 GiB 永久存储空间。

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的过程中创建新分支。

个人访问令牌 (PAT)

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 PAT 访问 CodeCatalyst 资源，包括集成开发环境 (IDE) 和基于 Git 的源存储库。PAT 代表你 CodeCatalyst ，你可以在用户设置中对其进行管理。一个用户可以拥有多个 PAT。个人访问令牌仅显示一次。作为最佳实践，请务必将其安全地存储在本地计算机上。默认情况下，PAT 将在一年后过期。

在使用集成开发环境 (IDE) 时，PAT 等同于 Git 密码。在设置 IDE 以使用 Git 存储库时，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。你可以使用分支来组织你的工作。例如，您可以使用分支来处理新版本或不同版本的文件，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，为该项目创建的源存储库包含名为 main 的分支中的示例文件。主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，则为该项目创建的源存储库除了示例代码、工作流程定义和其他资源外，还包括一个 README.md 文件。如果您在不使用模板的情况下创建源存储库，则会在首次提交时为您添加一个 README.md 文件，并在创建存储库的过程中为您创建一个默认分支。这个默认分支名为 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。您可以更改哪个分支用作默认分支。有关更多信息，请参阅 管理仓库的默认分支。

您无法删除源存储库的默认分支。搜索结果仅包括来自默认分支的结果。

提交

提交是对一个或一组文件的更改。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。提交包含有关变更的信息，包括进行更改的用户的身份、更改的时间和日期、提交标题以及包含的有关变更的任何消息。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。您还可以在提交中添加 Git 标签，以识别特定的提交。

拉取请求

拉取请求是您和其他用户在源存储库中查看、评论和将代码更改从一个分支合并到另一个分支的主要方式。您可以使用拉取请求以协作方式查看代码更改，以了解已发布软件的细微更改或修复、主要功能添加或新版本。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的修订版之间的差异。您可以为各行代码更改添加注释，也可以对整个拉取请求添加评论。

> **Tip**
>
> 在创建拉取请求时，显示的区别是源分支的尖端和目标分支的尖端之间的区别。创建拉取请求后，显示的区别将是您选择的拉取请求的修订版和创建拉取请求时作为目标分支提示的提交。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。每次推送到拉取请求的源分支都会创建一个修订版，其中包含在该推送中包含的提交中所做的更改。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

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

源存储库还可以存储项目的工作流程、通知、问题和其他配置信息的配置文件和其他信息。当您创建需要配置文件的资源时，或者将存储库指定为工作流程的源操作时，配置文件将创建并存储在源存储库中。如果您根据蓝图创建项目，则配置文件将存储在作为项目一部分为您创建的源存储库中。此配置信息存储在存储库默认分支`.codecatalyst`中名为的文件夹中。每当您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 的情况下使用 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 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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而找到您的用户个人资料。

2. 在 PAT 名称中，输入 PAT 的描述性名称。
3. 在到期日期中，保留默认日期或选择日历图标以选择自定义日期。到期日期默认为自当前日期起一年。
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 密钥保存在安全的位置。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后，您将无法检索它。

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，会为您创建资源，包括包含示例代码的源存储库。您可以使用此存储库和代码示例来学习如何：

- 查看项目的源存储库并浏览其内容
- 使用新分支创建开发环境，您可以在其中处理代码
- 更改文件，提交并推送您的更改
- 创建拉取请求并与其他项目成员一起查看您的代码更改
- 查看项目的工作流程：在拉取请求的源分支中自动生成和测试更改
- 将源分支中的更改合并到目标分支并关闭拉取请求
- 查看自动生成和部署的合并更改

要充分利用本教程，请邀请其他人加入您的项目，这样您就可以共同处理拉取请求。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

创建项目是能够协同工作的第一步。您可以使用蓝图来创建项目，该项目还将创建一个包含示例代码的源存储库和工作流程，该工作流程将在您更改代码时自动生成和部署代码。在本教程中，我们将引导您完成使用单页应用程序蓝图创建的项目，但是对于任何具有源存储库的项目，您都可以按照步骤进行操作。如果您在创建项目时没有 IAM 角色，请务必选择一个 IAM 角色或添加一个 IAM 角色。我们建议您为此项目使用CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色。

如果你已经有一个项目，你可以直接跳到查看项目的存储库。

> **Note**
>
> 只有拥有 Space 管理员或超级用户角色的用户才能在中创建项目 CodeCatalyst。如果您没有此角色，并且需要为本教程开发一个项目，请请具有其中一个角色的人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”。
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 “为项目命名” 下，输入要分配给项目的名称及其关联的资源名称。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。或者，您可以选择将蓝图的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。

   根据您要使用的第三方存储库提供商，执行以下任一操作：

   - 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 人际关系 和 通过人际关系访问 GitHub 资源。

   - Bitbucket 存储库：连接 Bitbucket 工作空间。

     选择 “高级” 下拉菜单，选择 Bitbucket 作为存储库提供程序，然后选择要存储蓝图创建的源代码的 Bitbucket 工作空间。

7. 在项目资源下，配置蓝图参数。根据蓝图，您可以选择命名源存储库的名称。

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“查看工作流程”。
9. （可选）从蓝图的卡片中选择查看详细信息以查看有关蓝图的特定详细信息，例如蓝图架构概述、所需的连接和权限以及蓝图创建的资源类型。
10. 选择创建项目。

一旦您创建项目或接受项目邀请并完成登录流程，项目概述页面就会打开。新项目的项目概述页面不包含未解决的问题或拉取请求。您可以选择创建议题并将其分配给自己。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

作为项目的成员，您可以查看该项目的源存储库。您也可以选择创建其他存储库。如果拥有 Space 管理员角色的人安装并配置了GitHub 存储库或 Bitbucket 扩展，您还可以在为扩展程序配置的 GitHub 账户或 Bitbucket 工作空间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 ***sp*** a-app。

导航到项目的源存储库

1. 导航到您的项目，然后执行以下任一操作：
   - 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来筛选存储库列表。
2. 在存储库的主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求的数量和工作流程。默认情况下，会显示默认分支的内容。您可以通过从下拉列表中选择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及其文件分支配置的工作流程和拉取请求的信息。如果您刚刚创建了项目，则构建、测试和部署代码的初始工作流程仍将运行，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，但这会在 CI/C D 中打开“工作流程”页面。在本教程中，请留在概述页面上浏览存储库中的代码。该`README.md`文件的内容显示在此页面的存储库文件下方。在“文件”中，将显示默认分支的内容。如果您有其他分支，则可以更改文件

视图以显示另一个分支的内容。该.codecatalyst文件夹包含用于项目其他部分的代码，例如工作流程 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将其展开。例如，选择旁边的箭头src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。你也可以在控制台中编辑单个文件，但要编辑多个文件，你需要创建一个开发环境。

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。但是，为了有效地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**。你可以选择不同的分支名称，但是通过命名分支**develop**，当你在本教程稍后创建拉取请求时，工作流程将自动运行以生成和测试你的代码。

> **Tip**
>
> 如果您决定在本地克隆存储库，而不是使用开发环境，或者除了使用开发环境外，请确保本地计算机上有 Git，或者您的 IDE 中包含 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。或者，在导航窗格中，选择代码，选择源存储库，然后选择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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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，编辑src文件夹中App.tsx文件中<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并打开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，使用和git push命令提交git commit并推送您的更改。

```
git commit -am "Making an example change"
git push
```

> **Tip**
>
> 在成功运行 Git 命令之前，您可能需要将终端中的目录更改为 Git 存储库目录。

创建拉取请求

您可以使用拉取请求以协作方式查看代码更改，以了解已发布软件的细微更改或修复、主要功能添加或新版本。在本教程中，您将创建一个拉取请求，以查看与主分支相比，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将启动其关联的工作流程（如果有）。

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，选择代码，选择拉取请求，然后选择创建拉取请求。
   - 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已提交代码的存储库。只有在您没有从存储库的主页创建拉取请求时，才会显示此选项。
4. 在 D est ination 分支中，在查看代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含已提交代码的分支。

6. 在 Pull req uest 标题中，输入一个标题，以帮助其他用户了解需要审阅的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如问题链接或更改描述之类的信息。

> **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。将自动生成的描述添加到拉取请求后，您可以对其进行更改。
> 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

8. （可选）在 “问题” 中，选择 “关联问题”，然后从列表中选择议题或输入其 ID。要取消议题的链接，请选择取消链接图标。
9. （可选）在必填审稿人中，选择添加所需的审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，必需的审阅者必须批准更改。

> **Note**
>
> 您不能将审阅者同时添加为必填审阅者和可选审阅者。您不能将自己添加为审阅者。

10. （可选）在可选审阅者中，选择添加可选审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，可选的审阅者不必将更改作为一项要求进行批准。
11. 查看分支之间的差异。拉取请求中显示的区别在于源分支中的修订版本和合并基础之间的变化，合并基础是创建拉取请求时目标分支的头部提交。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都选择了相同的分支。
12. 如果您对拉取请求中包含要查看的代码和更改感到满意，请选择 “创建”。

> **Note**
>
> 创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用 @ 符号和文件名来添加指向资源（例如文件）的链接。

通过选择 “概览”，然后在 “工作流程运行” 下的 “拉取请求详细信息” 区域中查看信息，可以查看有关创建此拉取请求时启动的关联工作流的信息。要查看工作流程运行情况，请选择运行。

> **Tip**
>
> 如果您将分支命名为以外的其他名称**develop**，则工作流程将不会自动运行来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，并请其他项目成员对此发表评论。您也可以选择添加或更改可选或必填的审阅者。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为拉取请求创建修订的。有关更多信息，请参阅查看拉取请求更新拉取请求、在 Amazon 中使用拉取请求查看代码 CodeCatalyst、和查看工作流程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。合并拉取请求还将通过与目标分支关联的所有工作流程启动更改。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进行选择。（可选）选择或取消选择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，拉取请求的状态将更改为“已合并”，并且不再出现在拉取请求的默认视图中。默认视图显示状态为“打开”的拉取请求。您仍然可以查看合并的拉取请求，但不能批准它或更改其状态。

   > **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 拉取请求详情区域概述中的时钟图标会显示尚未批准拉取请求的审阅者。如果所有必需的审阅者都批准了拉取请求，但合并按钮仍未激活，则可能存在合并冲突，或者已超过空间的存储配额。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

现在是时候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了。为此，您可以返回存储库的概述页面并选择相关工作流程图标旁边的数字，或者在导航窗格中选择 C I/CD，然后选择 Workflows。

查看已部署的代码

1. 在 “工作流程” 中onPushToMainDeployPipeline，展开 “最近运行”。

   > **Note**
   >
   > 这是使用单页应用程序蓝图创建的项目的工作流程的默认名称。

2. 最近的运行是由你合并的拉取请求提交到main分支开始的，其状态可能会显示为 “进行中”。从列表中选择成功完成的运行以打开该运行的详细信息。
3. 选择 “变量”。复制 A ppUrl 的值。这是已部署的单页 Web 应用程序的 URL。打开新的浏览器选项卡并粘贴该值以查看已生成和部署的代码。让选项卡保持打开状态。
4. 返回工作流程运行列表，等待最近一次运行完成。出现后，返回您打开的选项卡以查看 Web 应用程序并刷新浏览器。您应该会在合并的拉取请求中看到所做的更改。

清理资源

探索使用源存储库和拉取请求后，您可能需要移除任何不需要的资源。您无法删除拉取请求，但可以将其关闭。您可以删除您创建的任何分支。

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

源存储库是您安全地存储项目代码和文件的地方。它还存储您的源代码历史记录，从第一次提交到最新更改。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程和通知的配置文件和其他信息。此配置信息存储在名为 .codecat alyst 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 存储库链接到您的项目。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程序。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该项目。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存储库并分支到该存储库中，以便在为开发环境选择的集成开发环境 (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 支持凭据管理。

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。该源存储库包含示例代码以及为你创建的工作流程和其他资源的配置信息。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文件，即一个 README.md 文件，您可以随时对其进行编辑或删除。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。

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，添加存储库的描述，以帮助项目中的其他用户了解存储库的用途。
7. （可选）为您计划推送的代码类型添加.gitignore文件。
8. 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，但无法更改或删除默认分支。

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。只有拥有 Space 管理员角色的用户才能安装扩展。安装扩展程序后，您可以链接到配置为由该扩展程序访问的存储库。有关更多信息，请参阅在空间中安装扩展或关注在中关联 GitHub 存储库、Bitbucket 存储库和 Jira 项目 CodeCatalyst。

> **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。

> **Note**
>
> - GitHub 或 Bitbucket 存储库只能链接到空间中的一个 CodeCatalyst 项目。
> - 您不能在 CodeCatalyst 项目中使用空存储库、已存档存储库 GitHub 或 Bitbucket 存储库。
> - 您不能链接与 CodeCatalyst 项目中仓库同名的 GitHub 或 Bitbucket 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 数据中心存储库不兼容。

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 Space 管理员角色的用户必须先为托管存储库的提供商安装扩展程序。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. 根据您选择链接的第三方存储库提供商，执行以下任一操作：

   - 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，为 Amazon CodeCatalyst 应用程序选择 “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。

- 比特存储库：链接 Bitbucket 存储库。

  1. 从 Bitbucket 工作空间下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作空间。
  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. 选择关联。

如果您不想再在中使用 GitHub 或 Bitbucket 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
- 仓库的未处理拉取请求数量

- 存储库的相关工作流程数量
- 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
- 在 Markdown 中呈现的 README.md 文件的内容（如果包含任何 README.md 文件）

该页面还提供指向仓库提交、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。要查看有关链接存储库的信息，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，然后执行以下任一操作：
   - 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存储库名称的一部分来筛选存储库列表。
2. 在存储库的主页上，查看存储库的内容以及有关关联资源的信息，例如拉取请求的数量和工作流程。默认情况下，会显示默认分支的内容。您可以通过从下拉列表中选择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。这可以帮助项目成员了解存储库的用途，并帮助您强制执行团队使用的最佳实践和流程。

> **Note**
>
> 您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。要修改有关链接存储库的信息，必须在托管链接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：
   - 编辑存储库的描述，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 管理仓库的默认分支。
   - 要添加、移除或更改关于哪些项目角色有权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、删除或更改用于将拉取请求合并到分支的批准规则，请在批准规则中，选择编辑。有关更多信息，请参阅 管理合并拉取请求与批准规则的要求。

克隆源存储库

要有效地处理源存储库中的多个文件、分支和提交，请将源存储库克隆到本地计算机，然后使用 Git 客户端或集成开发环境 (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。您也可以选择创建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，然后对其进行克隆。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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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” 下拉菜单中选择要从中创建新分支的分支。
7. （可选）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存储库名称的一部分来筛选存储库列表。
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 客户端或集成开发环境 (IDE) 提示您输入密码时，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，然后使用git clone命令后面的克隆 URL 运行该命令。例如：

```
git clone https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

当系统提示输入密码时，请粘贴您之前保存的 PAT。

> **Note**
>
> 如果您的操作系统提供凭据管理，或者您已经安装了凭据管理系统，则只需提供 PAT 一次。否则，您可能需要为每个 Git 操作提供 PAT。作为最佳实践，请确保您的凭证管理系统安全地存储您的 PAT。请勿将 PAT 作为克隆 URL 字符串的一部分。

- 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 存储库的选项并提供 URL。当提示输入密码时，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。如果有任何工作流程依赖于源存储库，则在删除存储库后，这些工作流程将从项目工作流程列表中删除。引用源存储库的议题不会被删除或更改，但是删除存储库后，任何添加到议题中的源存储库的链接都将失败。

> **Important**
>
> 删除源存储库的操作无法撤消。删除源存储库后，您将无法再对其进行克隆、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。

> **Note**
>
> 您无法在 CodeCatalyst 控制台中删除链接的存储库。要删除链接存储库，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，然后将其删除。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. 在存储库的主页上，选择 “更多”，然后选择 “删除存储库”。
4. 查看分支、拉取请求和相关工作流程信息，以帮助确保您不会删除仍在使用或未完成工作的存储库。如果要继续，请键入 “删除”，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为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。当你在不使用蓝图的情况下创建源存储库时，系统会为你添加一个 README.md 文件作为第一次提交，并会为你创建一个默认分支。这个默认分支名为 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**
>
> 您无法删除默认分支。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，搜索仅显示来自默认分支的结果。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
- 管理仓库的默认分支
- 使用分支规则管理分支允许的操作
- 分支的 Git 命令
- 查看分支和详细信息

- 删除分支

创建分支

您可以使用 CodeCatalyst 控制台在 CodeCatalyst 存储库中创建分支。您创建的分支将在其他用户下次从存储库中提取更改时对他们可见。

> **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。

你也可以使用 Git 来创建分支。有关更多信息，请参阅 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”。

管理仓库的默认分支

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，则为该项目创建的源存储库中的默认分支名为 main。默认分支的内容会自动显示在该存储库的概述页面上。

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。它的名字旁边有一个特殊的标签 “默认”。默认分支是用户使用 Git 客户端将仓库克隆到本地计算机时，在本地存储库（存储库）中用作基础分支或默认分支的分支。它也是创建用于存储工作流 YAML 文件和存储问题信息的工作流程时使用的默认设置。使用 search in 时 CodeCatalyst，仅搜索存储库的默认分支。由于默认分支是项目许多方面的基础，因此如果将分支指定为默认分支，则无法将其删除。但是，您可以选择使用其他分支作为默认分支。如果这样做，则应用于以前的默认分支的任何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。这不适用于链接的仓库。

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”。

   选择要在其中查看设置的存储库，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，将显示指定为默认分支的分支的名称，并在名称旁边显示一个名为 De f ault 的标签。在 Branches 的分支列表中，分支名称旁边会显示相同的标签。
5. 要更改默认分支，请选择编辑。

   > **Note**
   >
   > 要更改默认分支，您必须在项目中具有项目管理员角色。
6. 从下拉列表中选择要设为默认分支的分支的名称，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允许对特定分支执行的操作。分支规则基于用户在项目中的角色。您可以选择将某些预定义的操作（例如将提交推送到分支）限制给在项目中具有特定角色的用户。这可以限制允许哪些角色执行某些操作，从而帮助您保护项目中的特定分支。例如，如果您将分支规则配置为仅允许具有项目管理员角色的用户合并或推送到该分支，则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，如果您选择仅向具有项目管理员角色的用户推送到某个分支，则具有参与者角色的用户将无法在该分支中创建或编辑工作流程，因为工作流程 YAML 存储在该分支中，并且这些用户无法提交和推送对 YAML 的更改。作为最佳实践，请在创建分支规则之后对其进行测试，以确保它们不会产生任何您意想不到的影响。您也可以将分支规则与拉取请求的批准规则结合使用。有关更多信息，请参阅 管理合并拉取请求与批准规则的要求。

> **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。您无法为链接仓库创建分支规则。
> 您只能创建比角色的默认权限更严格的分支规则。您不能创建比用户在项目中的角色所允许的更宽松的分支规则。例如，您无法创建允许具有审阅者角色的用户推送到分支的分支规则。

应用于源存储库默认分支的分支规则的行为方式与应用于其他分支的分支规则略有不同。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适用于它的规则，只是它不再具有防止删除的保护。该保护仅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。要详细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操作的操作具有允许执行该操作的特定角色列表，这些角色与用户在项目中的角色授予的默认权限不同。

> **Note**
>
> 如果您创建分支规则来限制分支的一项或多项操作，则 “删除分支” 操作会自动设置为仅允许具有项目管理员角色的用户删除该分支。

下表列出了允许在分支上执行这些操作的角色的操作和默认设置。

分支操作和角色

分支操作	当未应用分支规则时，允许角色执行此操作
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

您无法删除分支规则，但可以更新分支规则，以允许所有角色的操作在分支上执行此操作，从而有效地删除该规则。

> **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。这不适用于链接存储库。链接存储库不支持中的分支规则 CodeCatalyst。

查看和编辑仓库的分支规则

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，并且与项目中授予这些角色的权限相匹配。“自定义” 表示一个或多个分支操作的规则将允许该分支执行的一个或多个操作限制为不同的角色集。

   要查看分支规则的细节，请选择要查看的分支旁边的 “标准” 或 “自定义” 一词。
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的分支规则中，选择编辑。
6. 在 Branch 中，从下拉列表中选择要为其配置规则的分支的名称。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：

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`
将有关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，将分支从本地存储库推送到远程存储库。还可以在本地仓库中为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设置的指定昵称和指定的分支名称删除远程存储库中的分支。(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 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。您还可以使用 Git 命令和本地操作系统来查看远程和本地分支的此信息。

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”。

   选择要在其中查看分支的存储库。
3. 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、有关最新提交的信息，以及 README.md 文件（如果分支中存在）的内容。要查看其他分支的信息，请从存储库的分支下拉列表中选择该分支。
4. 要查看存储库的所有分支，请选择全部查看。分支页面显示有关每个分支的名称、最近提交和规则的信息。

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，如果您已将包含功能更改的分支合并到默认分支中，并且该功能已发布，则可能需要删除原始功能分支，因为这些更改已经是默认分支的一部分。保持较低的分支数量可以帮助用户找到包含他们想要处理的更改的分支。删除分支时，该分支的副本将保留在本地计算机上存储库的克隆中，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**
>
> 您无法删除存储库的默认分支。

5. 您将看到确认对话框。它显示存储库、未处理的拉取请求数量以及与分支关联的工作流程数量。
6. 要确认删除该分支，请在文本框中键入 del ete，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 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。您可以将文件的不同版本存储在不同的存储库分支中。

要在源存储库中添加或编辑多个文件，可以使用 Git 客户端、开发环境或集成开发环境 (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
- 查看文件
- 查看文件更改的历史记录
- 编辑文件
- 重命名或删除文件

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。此编辑器是在存储库分支中创建或编辑简单文件（例如 README.md 文件）的便捷方法。处理多个文件时，可以考虑创建一个开发环境。

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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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” 下拉菜单中选择要从中创建新分支的分支。

7. （可选）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）如果要在与默认分支不同的分支中创建文件，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名。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。

   如果您对更改感到满意，请选择 “提交”。
6. 在 “文件名” 中，查看文件名并对其进行任何可能的更改。（可选）从 Branch 的可用分支列表中选择要在其中创建文件的分支。在提交消息中，可以选择输入简短但内容丰富的描述，说明您进行此更改的原因。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” 以提交文件并将其推送到源存储库。

您还可以将文件添加到源代码库，方法是将文件克隆到本地计算机，然后使用 Git 客户端或连接的集成开发环境 (IDE) 推送文件和更改。

> **Note**
>
> 如果要添加 Git 子模块，则必须使用 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 客户端或连接的集成开发环境 (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。
3. 通过执行以下任一操作来创建并推送提交：

   - 如果您使用的是 Git 客户端，请在终端或git add命令行运行该命令，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，然后加上单点或双句点，以指示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和所有子目录中的所有更改（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 中创建文件并添加文件，然后提交并推送更改。有关更多信息，请参阅使用开发环境编写和修改代码 CodeCatalyst或查阅您的 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”。

   选择要在其中查看文件的存储库。
3. 将显示默认分支的文件和文件夹列表。文件由 paper 图标表示，而文件夹由文件夹图标表示。
4. 执行以下任一操作：

- 要查看其他分支中的文件和文件夹，请从分支列表中进行选择。
- 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。要查看其他分支中的文件内容，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” 中选择要查看的其他文件。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。

查看文件更改的历史记录

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。这可以帮助您了解通过向您选择查看文件历史记录的分支进行各种提交而对文件所做的更改。例如，如果您在源存储库的**`main`**分支中查看**`readme.md`**文件更改的历史记录，则会看到包含该分支中该文件更改的提交列表。

> **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. 导航到要在其中查看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. 查看包含指定分支中此文件更改的提交列表。要查看特定提交中包含的更改的详细信息，请在列表中选择该提交的提交消息。将显示该提交与其父提交之间的区别。
5. 要查看其他分支中文件更改的历史记录，请使用分支选择器更改该分支的视图，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，请创建开发环境或克隆存储库，然后使用 Git 客户端或集成开发环境 (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. 选择要在其中编辑文件的存储库。选择 “查看分支”，然后选择要在其中工作的分支。从该分支中的文件和文件夹列表中选择文件。

   将显示该文件的内容。
3. 选择编辑。
4. 在编辑器中，编辑文件内容，然后选择 “提交”。或者，在提交更改中，在提交消息中添加有关更改的更多信息。如果您对所做的更改感到满意，请选择 “提交”。

重命名或删除文件

您可以在开发环境、计算机本地或集成开发环境 (IDE) 中重命名或删除文件。重命名或删除文件后，提交这些更改并将其推送到源存储库。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审阅、评论代码更改以及将代码更改从一个分支合并到另一个分支的主要方式。您可以使用拉取请求以协作方式查看代码更改，以了解已发布软件的细微更改或修复、主要功能添加或新版本。如果您使用议题来跟踪项目中的工作，则可以将特定问题链接到拉取请求，以帮助您跟踪拉取请求中的代码更改正在解决哪些问题。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拉取请求的任何必填或可选审阅者发送一封电子邮件。

> **Tip**
>
> 您可以配置哪些拉取请求事件，这些事件将在您的个人资料中收到有关电子邮件的电子邮件。有关更多信息，请参阅 在 Amazon 中管理通知 CodeCatalyst。

拉取请求需要在源存储库中有两个分支：一个包含您要查看的代码的源分支，以及一个目标分支，用于合并已审核的代码。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，显示的区别是源分支的尖端和目标分支的尖端之间的区别。创建拉取请求后，显示的区别将是您选择的拉取请求的修订版和创建拉取请求时作为目标分支提示的提交。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。

在为特定的源存储库和分支创建拉取请求时，您可以在处理项目时创建、查看、查看和关闭它们。您不必查看源存储库即可查看和处理拉取请求。创建拉取请求时，拉取请求状态设置为 “打开”。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”）。

审查完您的代码后，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。拉取请求的源分支中的代码将合并到目标分支中。拉取请求状态将更改为 “已合并”。无法将其更改回 “打开”。
- 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”，并且不会将源分支中的代码合并到目标分支中。

在创建拉取请求之前，请：

- 提交要查看的代码更改并将其推送到分支（源分支）。
- 为您的项目设置通知，以便其他用户可以收到有关您创建拉取请求时运行的任何工作流程的通知。（此步骤是可选的，但建议这样做。）

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，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。创建拉取请求时，您将自动收到有关拉取请求的任何更新以及拉取请求何时合并或关闭的电子邮件。

> **Note**
>
> 在创建拉取请求时，显示的区别是源分支的尖端和目标分支的尖端之间的区别。创建拉取请求后，显示的区别将是您选择的拉取请求的修订版和创建拉取请求时作为目标分支提示的提交。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 “为我写描述” 功能，让 Amazon Q 自动创建拉取请求中包含的更改的描述。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，它总结了这些变化的内容，以及对这些变化的意图和效果的最佳解释。此功能仅在美国西部（俄勒冈）区域可用。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于 “为我写描述” 和 “创建内容摘要” 功能是基于 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执行安全、安保和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，选择代码，选择拉取请求，然后选择创建拉取请求。
   - 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已提交代码的存储库。只有在您没有从存储库的主页创建拉取请求时，才会显示此选项。
4. 在 D est ination 分支中，在查看代码后，选择要将代码合并到的分支。
5. 在源分支中，选择包含已提交代码的分支。
6. 在 Pull req uest 标题中，输入一个标题，以帮助其他用户了解需要审阅的内容及其原因。
7. （可选）在拉取请求描述中，提供诸如问题链接或更改描述之类的信息。

   > **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。将自动生成的描述添加到拉取请求后，您可以对其进行更改。
   > 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

8. （可选）在 “问题” 中，选择 “关联问题”，然后从列表中选择议题或输入其 ID。要取消议题的链接，请选择取消链接图标。
9. （可选）在必填审稿人中，选择添加所需的审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，必需的审阅者必须批准更改。

   > **Note**
   >
   > 您不能将审阅者同时添加为必填审阅者和可选审阅者。您不能将自己添加为审阅者。

10. （可选）在可选审阅者中，选择添加可选审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，可选的审阅者不必将更改作为一项要求进行批准。
11. 查看分支之间的差异。拉取请求中显示的区别在于源分支中的修订版本和合并基础之间的变化，合并基础是创建拉取请求时目标分支的头部提交。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都选择了相同的分支。
12. 如果您对拉取请求中包含要查看的代码和更改感到满意，请选择 “创建”。

> **Note**
>
> 创建拉取请求后，您可以添加评论。可以将评论添加到拉取请求或文件中的各个行中，也可以添加到整个拉取请求中。您可以使用 @ 符号和文件名来添加指向资源（例如文件）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，然后选择包含要查看的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的分支。要查看存储库的所有分支，请选择全部查看。
4. 选择 “更多”，然后选择 “创建拉取请求”。
5. 已为您预先选择存储库和源分支。在 D est ination 分支中，选择审核完代码后要合并的分支。在 P ull requ est 标题中，输入一个标题，该标题将帮助其他项目用户了解必须审阅的内容及其原因。（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程中指定的分支之一匹配，则配置为为拉取请求创建事件运行的工作流程将在拉取请求创建事件后运行。

6. 查看分支之间的差异。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都选择了相同的分支。
7. （可选）在 “问题” 中，选择 “关联问题”，然后从列表中选择议题或输入其 ID。要取消议题的链接，请选择取消链接图标。

8. （可选）在必填审稿人中，选择添加所需的审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，必需的审阅者必须批准更改。

> **Note**
>
> 您不能将审阅者添加为必填和可选。您无法将自己添加为审阅者。

9. （可选）在可选审阅者中，选择添加可选审阅者。从项目成员列表中进行选择以添加他们。在将拉取请求合并到目标分支之前，可选的审阅者不必批准更改。
10. 如果您对拉取请求包含您要审核的更改并包括所需的审阅者感到满意，请选择创建。

如果您将任何工作流程配置为在拉取请求中的分支与目标分支匹配的地方运行，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程运行的信息。有关更多信息，请参阅 添加推送、拉动或调度触发器。

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。项目摘要页面显示项目的所有未完成的拉取请求。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。查看拉取请求时，您可以选择汇总为你创建的拉取请求的变更留下的所有评论。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于 “为我写描述” 和 “创建内容摘要” 功能是基于 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执行安全、安保和负责任地使用人工智能 (AI)。

查看未处理的拉取请求

1. 导航到要查看拉取请求的项目。
2. 在项目页面上，将显示已打开的拉取请求，包括有关谁创建了拉取请求、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已打开的拉取请求视图。
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选项之间进行选择。例如，要查看所有拉取请求，请选择任意状态和任意作者。

   或者，在导航窗格中选择 “代码”，然后选择 “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和信息量，请选择齿轮图标。
5. 要查看特定的拉取请求，请从列表中进行选择。
6. 要查看与此拉取请求关联的工作流程运行的状态（如果有），请选择 Overvie w，然后在 “工作流程运行” 下查看拉取请求详细信息区域中的信息。

   如果工作流程配置了拉取请求创建或修订事件，并且工作流程中的目标分支要求与拉取请求中指定的目标分支相匹配，则会运行工作流程。有关更多信息，请参阅 添加推送、拉动或调度触发器。
7. 要查看关联的问题（如果有），请选择 “概述”，然后在 “问题” 下查看 “拉取请求详情” 中的信息。如果您想查看关联的问题，请从列表中选择其 ID。
8. （可选）要创建拉取请求修订版中对代码更改留下的评论摘要，请选择创建内容摘要。摘要将不包括整个拉取请求中留下的任何评论。

   > **Note**
   >
   > 此功能要求为该空间启用生成式 AI 功能，并且仅在美国西部（俄勒冈）地区可用。有关更多信息，请参阅管理生成式 AI 功能。

9. 要查看拉取请求中的代码更改，请选择更改。您可以在 Files changed 中快速查看拉取请求中有多少文件发生了更改，以及拉取请求中哪些文件有评论。文件夹旁边显示的评论数量表示该文件夹中文件的评论数量。展开文件夹，查看该文件夹中每个文件的评论数量。您还可以查看特定代码行上留下的任何注释。

   > **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**
    >
    > 必须将工作流程配置为生成报告，这些报告才能显示在您的拉取请求中。有关更多信息，请参阅 使用工作流程进行测试。

管理合并拉取请求与批准规则的要求

创建拉取请求时，您可以选择为该个人拉取请求添加必填或可选审阅者。但是，您也可以创建所有拉取请求在合并到特定目标分支时必须满足的要求。这些要求称为批准规则。批准规则是为仓库中的分支配置的。当您创建的拉取请求的目标分支已为其配置了批准规则时，除了获得任何所需审阅者的批准外，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。创建批准规则可以帮助您保持合并到分支（例如默认分支）的质量标准。

应用于源存储库默认分支的批准规则的行为方式与应用于其他分支的批准规则略有不同。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。

在创建批准规则时，应考虑项目用户在现在和将来将如何遵守该规则。例如，如果您的项目中有六个用户，并且您创建的批准规则要求五次批准才能将其合并到目标分支，那么您实际上已经创建了一条规则，要求除创建拉取请求的人员以外的所有人批准该拉取请求，然后才能将其合并。

> **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。您无法为链接仓库创建批准规则。

您无法删除批准规则，但可以将其更新为要求零批准，这实际上会删除该规则。

查看和编辑拉取请求的目标分支的批准规则

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”。

   选择要在其中查看批准规则的存储库。
3. 在存储库的概述页面上，选择分支。
4. 在 “批准规则” 列中，选择 “查看” 以查看存储库每个分支的所有规则的状态。

   在 “最小批准数量” 中，该数字对应于拉取请求合并到该分支之前所需的批准数量。
5. 要创建或更改批准规则，请选择管理设置。在源存储库的设置页面的批准规则中，选择编辑。

> **Note**
>
> 您必须具有项目管理员角色才能编辑批准规则。

6. 在 Branch 中，从下拉列表中选择要为其配置批准规则的分支的名称。在最小批准数量中，输入一个数字，然后选择保存。

查看拉取请求

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。您可以在源分支和目标分支之间的区别或拉取请求的修订版之间的差异中为各行代码添加注释。您可以选择创建拉取请求中代码更改时留下的评论摘要，以帮助您快速了解其他用户留下的反馈。您也可以选择创建开发环境来处理代码。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于 “为我写描述” 和 “创建内容摘要” 功能是基于 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执行安全、安保和负责任地使用人工智能 (AI)。

> **Tip**
>
> 您可以配置哪些拉取请求事件，这些事件将在您的个人资料中收到有关电子邮件的电子邮件。有关更多信息，请参阅 在 Amazon 中管理通知 CodeCatalyst。

拉取请求显示拉取请求的修订版与创建拉取请求时目标分支提示的提交之间的区别。这就是所谓的合并基础。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，特别是如果您已打开拉取请求一段时间，请考虑刷新浏览器，确保在开始查看拉取请求之前，有最新的版本可供拉取请求使用。

在 CodeCatalyst 控制台中查看拉取请求

1. 导航到您的项目。
2. 通过执行以下任一操作导航到拉取请求：
   - 如果项目页面上列出了拉取请求，请从列表中进行选择。
   - 如果项目页面上未列出拉取请求，请选择查看全部。使用筛选器和排序来查找拉取请求，然后从列表中进行选择。
   - 在导航窗格中，选择代码，然后选择拉取请求。
3. 从列表中选择要查看的拉取请求。您可以通过在筛选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，您可以查看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，包括有关工作流程运行、关联问题、审阅者、拉取请求作者和可行的合并策略的信息。

   > **Note**
   >
   > 在特定代码行上留下的注释显示在 “更改” 中。

5. （可选）要添加适用于整个拉取请求的评论，请展开 “拉取请求评论”，然后选择 “创建评论”。
6. （可选）要查看此拉取请求修订版本中留下的所有评论的摘要，请选择创建评论摘要。

   > **Note**
   >
   > 此功能要求为该空间启用生成式 AI 功能，并且仅在美国西部（俄勒冈）地区可用。有关更多信息，请参阅管理生成式 AI 功能。

7. 在 Ch anges 中，您可以看到目标分支与拉取请求的最新版本之间的区别。如果有多个修订版，则可以更改在它们之间的差异中比较哪些修订版本。有关修订的更多信息，请参阅修订。

   > **Tip**
   >
   > 您可以在 Files changed 中快速查看拉取请求中有多少文件发生了更改，以及拉取请求中哪些文件有评论。文件夹旁边显示的评论数量表示该文件夹中文件的评论数量。展开文件夹，查看该文件夹中每个文件的评论数量。

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. 要向拉取请求中的某一行添加评论，请转到您要评论的行。选择该行显示的评论图标，输入评论，然后选择 “保存”。
10. 要查看拉取请求中修订版本之间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。修订版中各行的注释保留在这些修订版中。
11. 如果您已将工作流程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，则可以根据拉取请求创建开发环境。选择创建开发环境。（可选）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。如果有多个修订版，则可以更改在它们之间的差异中比较哪些修订版本。您可以按名称、状态、工作流程、操作和类型筛选报告。

> **Note**
>
> 必须将工作流程配置为生成报告，这些报告才能显示在您的拉取请求中。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 使用工作流程进行测试。
15. 如果您被列为该拉取请求的审阅者并想要批准更改，请确保您正在查看最新的修订版，然后选择 Approve（批准）。

> **Note**
>
> 所有必需的审阅者都必须批准拉取请求，然后才能将其合并。

更新拉取请求

通过更新拉取请求，您可以让其他项目成员更轻松地查看代码。您可以更新拉取请求以更改其审阅者、议题链接、拉取请求的标题或描述。例如，您可能需要更改拉取请求所需的审阅者，以删除正在度假的人，然后添加其他人。您还可以通过将提交推送到已打开的拉取请求的源分支来更新拉取请求，并进行进一步的代码更改。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。项目成员可以在拉取请求中查看修订版之间的差异。

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. 在项目页面的 “打开拉取请求” 下，选择要向审阅者通报最新情况的拉取请求。或者，在导航窗格中，选择代码，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，选择加号以添加必填或可选审阅者。选择审阅者旁边的 X 可将其作为可选或必填审阅者删除。
4. （可选）在概述中的拉取请求详细信息区域中，选择关联议题以将议题链接到拉取请求，然后从列表中选择议题或输入其 ID。要取消关联某个议题，请选择要取消关联的议题旁边的取消链接图标。

更新拉取请求源分支中的文件和代码

1. 要更新多个文件，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. 要更新源分支中的单个文件，可以像处理多个文件一样使用 Git 客户端或 IDE。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标题或描述的项目。
2. 项目页面显示未完成的拉取请求，包括拉取请求的创建者、包含拉取请求分支的存储库以及拉取请求的创建时间等信息。您可以按源存储库筛选已打开的拉取请求视图。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，然后将其选中。
4. 在概述中，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。您还可以将拉取请求与您选择的 Git 合并策略合并，方法是在本地计算机

或开发环境中运行git merge命令，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批准和批准规则要求的拉取请求。

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。如果存在冲突，或者无法完成合并，则合并按钮处于非活动状态，并显示 “不可合并” 标签。在这种情况下，您必须获得所有所需批准者的批准，必要时在本地解决冲突，并在合并之前推送这些更改。合并拉取请求将自动向拉取请求的创建者以及任何必填或可选的审阅者发送一封电子邮件。它不会自动关闭或更改与拉取请求相关的任何议题的状态。

> **Tip**
>
> 您可以配置哪些拉取请求事件，这些事件将在您的个人资料中收到有关电子邮件的电子邮件。有关更多信息，请参阅 在 Amazon 中管理通知 CodeCatalyst。

合并拉取请求

1. 导航到要合并拉取请求的项目。
2. 在项目页面的 “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 “查看所有拉取请求”，然后从列表中进行选择。或者，在导航窗格中，选择代码，选择拉取请求，然后选择要合并的拉取请求。选择 Merge (合并)。
3. 从拉取请求的可用合并策略中进行选择。（可选）选择或取消选择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 “拉取请求详情” 区域中的时钟图标会显示尚未批准拉取请求的审阅者。如果所有必需的审阅者都批准了拉

取请求，但 “合并” 按钮仍处于非活动状态，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。

忽略合并要求

如果您具有项目管理员角色，则可以选择合并尚未满足所需批准和批准规则所有要求的拉取请求。这被称为覆盖拉取请求的要求。如果所需的审阅者不可用，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批准规则的分支中，则可以选择这样做。

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 “改写批准要求”。
2. 在 Override reason 中，详细说明为什么要合并此拉取请求而不符合批准规则和所需的审阅者要求。虽然这是可选的，但强烈建议这样做。
3. （可选）选择合并策略，或接受默认策略。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. 选择或取消选择合并时删除源分支的选项。我们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查该决定。
5. 选择 Merge (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。以下命令是您可以使用的部分选项。有关更多信息，请参阅 Git 网站上的可用文档。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中，指定快进合并。这将合并分支并将目标分支指针移动到源分支的尖端。	`git checkout destination-branch-name`
	
	`git merge --ff-only source-branch-name`

将源分支合并到目标分支中，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中。	`git checkout destination-branch-name`
	
	`git merge --squash source-branch-name`

将源分支合并到目标分支中，指定三向合并。这将创建合并提交，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`

删除本地仓库中的源分支。在合并到目标分支并将更改推送到源存储库之后，这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，但可以帮助您确定哪些拉取请求需要操作，哪些拉取请求不再相关。如果您不再计划合并这些更改，或者更改已被另一个拉取请求合并，我们建议您关闭拉取请求。

关闭拉取请求将自动向拉取请求的创建者以及任何必填或可选的审阅者发送一封电子邮件。它不会自动更改与拉取请求相关的任何问题的状态。

> **Note**
>
> 拉取请求关闭后，您无法重新打开它。

关闭拉取请求

1. 导航到要关闭拉取请求的项目。
2. 在项目页面上，将显示未完成的拉取请求。选择要关闭的拉取请求。
3. 选择关闭。
4. 查看信息，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。每当用户提交更改并将其推送到分支时，该信息都会被保存。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。您还可以在提交中添加 Git 标签，以帮助您识别特定的提交。

在亚马逊 CodeCatalyst，您可以：

- 查看分支的提交列表。
- 查看单个提交，包括在提交中与其父提交相比所做的更改。

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

查看对分支的提交

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。这可以帮助您了解谁对分支进行了更改以及何时进行了更改。您还可以查看在特定提交中所做的更改。

> **Tip**
>
> 您还可以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。

您也可以使用 Git 客户端查看提交。有关更多信息，请参阅 Git 文档。

查看提交（控制台）

1. 导航到包含要在其中查看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. 将显示存储库的默认分支，包括有关该分支的最新提交的信息。选择 “提交”。或者，选择 “更多”，然后选择 “查看提交”。
4. 要查看其他分支的提交，请选择分支选择器，然后选择该分支的名称。
5. 要查看有关特定提交的详细信息，请从 “提交标题” 中选择其标题。将显示提交的详细信息，包括有关其父提交的信息，以及通过将父提交与指定提交进行比较而对代码所做的更改。

   > **Tip**
   >
   > 如果一个提交有多个父提交，则可以通过选择父提交 ID 旁边的下拉图标来选择要查看哪个父提交的信息，并显示其更改。

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诸如作者和提交 ID 之类的列。

更改提交的显示方式（控制台）

1. 导航到包含要在其中查看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”。

   选择要在其中更改查看提交方式的存储库。

3. 将显示存储库的默认分支，包括有关该分支的最新提交的信息。选择 “提交”。
4. 选择齿轮图标。
5. 在 “首选项” 中，选择要显示的提交数量，然后选择是否显示有关提交作者、提交日期和提交 ID 的信息。

> **Note**
>
> 您不能在信息显示中隐藏提交标题。

6. 完成更改后，选择 “保存” 以保存更改，或选择 “取消” 将其丢弃。

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，请参阅的配额 CodeCatalyst。

资源	信息
分支名称	允许的字符的任意组合，长度介于 1 到 256 个字符之间，并且在存储库中必须是唯一的。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。
对拉取请求的评论	拉取请求上限为 1,000。
提交消息	最多 1024 个字符。

资源	信息

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
	
	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，请使用适用于 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。

资源	信息
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。但是，作为最佳实践，可以考虑进行多个较小的提交，而不是一次大型提交。
提交的元数据	一次提交的合并元数据（例如，作者信息、日期、父提交列表和提交消息的组合）的最大值为 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
可以关联到拉取请求的 Jira 事务数量	50
空间中已打开的拉取请求数量	亚马逊 CodeCatalyst 空间的最大值为 1,000。
空间中的拉取请求总数	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。

资源	信息
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，长度介于 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 定价 和 源存储库问题疑难解答。
拉取请求的审阅者	一个拉取请求总共最多有 100 位审阅者（可选或必填）。
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的空间的计费等级。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) 处理代码。
- 创建一个空的开发环境，将源存储库中的代码克隆到其中，然后使用支持的 IDE 处理该代码。
- 在 IDE 中创建开发环境并将源存储库克隆到开发环境中

开发文件是一个开放的标准 YAML 文件，用于标准化您的开发环境。换句话说，此文件编纂了您的开发环境所需的开发工具。因此，您可以快速设置开发环境，在项目之间切换，并在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，因为它们使用开发文件来配置给定项目编码、测试和调试所需的所有工具。

开发环境中包含的项目工具和应用程序库由项目源存储库中的 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。这个默认的 devfile 包括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。有关开发文件的更多信息，请参阅 https://devfile.io。

创建开发环境后，只有您可以访问它。在开发环境中，您可以在支持的 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 2 核处理器、4 GB RAM 和 16 GB 永久存储空间。如果您拥有 Space 管理员权限，则可以更改空间的计费等级，以使用不同的开发环境配置选项并管理计算和存储限制。

主题

- 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 中创建开发环境，并将任何源存储库克隆到开发环境中

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 IDE 在其中工作。必须 AWS Toolkit 安装才能在 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” 列将始终显示您在创建此开发环境时指定的分支。即使您创建了另一个分支、切换到开发环境中的另一个分支或克隆了其他存储库，这也适用。如果您创建了一个空的开发环境，则 “资源” 列将为空。

开发环境支持的集成开发环境

您可以将开发环境与以下支持的集成开发环境 (IDE) 配合使用：

- AWS Cloud9
- JetBrains IDE
  - IntelliJ IDEA 旗舰版
  - GoLand
  - PyCharm专业人士
- Visual Studio 代码

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，执行以下任一操作：
   - 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”，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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成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。开发环境将从您从源存储库页面中选择的源存储库创建。

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以编辑开发环境的计算、存储或超时配置。
10. （可选）在 Amazon Virtual Private Cloud（亚马逊 VPC）（可选）中，从下拉菜单中选择要与开发环境关联的 VPC 连接。

如果您的空间设置了默认 VPC，则您的开发环境将连接到该 VPC 运行。您可以通过关联不同的 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

> **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，执行以下任一操作：
   - 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. 从下拉菜单中选择支持的 IDE。请参阅开发环境支持的集成开发环境了解更多信息。
6. 选择 “创建空开发环境”。
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以编辑开发环境的计算、存储或超时配置。
9. （可选）在 Amazon Virtual Private Cloud（亚马逊 VPC）（可选）中，从下拉菜单中选择要与开发环境关联的 VPC 连接。

   如果您的空间设置了默认 VPC，则您的开发环境将连接到该 VPC 运行。您可以通过关联不同的 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

> **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一到两分钟。

> **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。开发环境可以提高开发速度，因为您可以在具有支持的集成开发环境 (IDE) 的特定项目且功能齐全的云开发环境中立即开始编码。

有关在 IDE CodeCatalyst 中使用的信息，请参阅以下文档。

- CodeCatalyst 适用于 JetBrains IDE 的 Amazon
- 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`目录存储了从源存储库中提取的文件和用于配置开发环境的开发文件。该`/home`目录在创建开发环境时为空，用于存储您在使用开发环境时创建的文件。开发环境/

projects和/home目录中的所有内容都是永久存储的，因此，如果您需要切换到其他开发环境、存储库或项目，则可以停止在开发环境中工作。

> **Warning**
>
> 如果任何实例（包括 Web 浏览器、远程 shell 和 IDE）保持连接状态，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。

如果开发环境在创建开发环境期间在 “超时” 字段中选择的时间段内处于空闲状态，则该环境将自动停止。你可以在开发环境闲置之前将其停止。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，则该连接将无法恢复。

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**
>
> 计算使用量仅在开发环境运行时计费，但存储使用量按开发环境存在的整个时间计费。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目录存储了从源存储库中提取的文件和用于配置开发环境的开发文件。该/home目录在创建开发环境时为空，用于存储您在使用开发环境时创建的文件。开发环境/

projects和/home目录中的所有内容都是永久存储的，因此，如果您需要切换到其他开发环境、存储库或项目，稍后再继续在开发环境中工作，则可以停止在开发环境中工作。

如果开发环境在创建开发环境期间在 “超时” 字段中选择的时间段内处于空闲状态，则该环境将自动停止。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**
>
> 即使您删除了创建开发环境的分支，开发环境仍可用且正在运行。如果您想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. 在 (IDE) 中选择 “恢复”。
   - 对于 JetBrains IDE，当提示选择要打开 JetBrains 网关链接的应用程序时，请选择 Gateway-eap。 JetBrains出现提示时，选择 “打开链接” 进行确认。
   - 对于 VS Code IDE，当提示选择要打开 VS Code 链接的应用程序时，选择 VS Code。选择 “打开链接” 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. 导航到要在其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 IDE，当系统提示允许此站点使用 Gateway 打开 JetBrains-gateway 链接时，选择 Open Link 进行确认 JetBrains？ 。
   - 对于 VS Code IDE，当系统提示允许此网站使用 Visual Studio Code 打开 VS Code 链接时，选择 “打开链接” 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. 导航到要在其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 IDE，当系统提示允许此站点使用 Gateway 打开 JetBrains-gateway 链接时，选择 Open Link 进行确认 JetBrains？ 。
   - 对于 VS Code IDE，当系统提示允许此网站使用 Visual Studio Code 打开 VS Code 链接时，选择 “打开链接” 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

在 IDE 运行期间，您可以编辑开发环境。如果您编辑计算或非活动状态超时，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状态超时进行更改。
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，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，则保留的内容将被永久删除。在删除开发环境之前，请务必提交代码更改并将其推送到开发环境的原始源存储库。删除开发环境后，开发环境的计算和存储计费将停止。

删除开发环境后，可能需要几分钟才能更新存储配额。如果您已达到存储配额，则在此期间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，您将无法再将其恢复。

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`**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。
> 即使您删除了开发环境，也可能无法删除 VPC 中的网络接口。确保根据需要清理资源。如果删除已连接 VPC 的开发环境时出现错误，则必须断开陈旧的连接，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 IDE。

> **Note**
>
> 如果要在关闭 IDE 选项卡或窗口后长时间继续使用 SSH，请务必为开发环境设置较高的超时时间，这样它就不会因在 IDE 中处于非活动状态而停止。

先决条件

- 您需要以下操作系统之一：
  - Windows 10 或更高版本且已启用 OpenSSH
  - macOS 和 Bash 版本 3 或更高版本
  - 带有`yum`、`dpkg`或`rpm`包管理器的 Linux 以及 Bash 版本 3 或更高版本
- 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. 选择要使用 SSH 连接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：
   - 如果您尚未执行此操作，请在指定的终端中粘贴并执行第一个命令。该命令下载脚本并在本地环境中执行以下修改，以便您可以使用 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 SSH 客户端用于连接到您的开发环境的~/.aws/codecatalyst-dev-env目录中添加脚本。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 客户端使用该会话来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。每次您的凭证到期时，都应执行此命令。当您在模态 (ssh<destination>) 中执行最后一个命令时，如果您的凭据已过期或者您尚未按照此步骤中的说明执行 SSO 登录，则会收到错误消息。

- 使用第三个命令使用 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>
```

为开发环境配置开发文件

devfile 是一种开放标准，可帮助您在整个团队中自定义开发开发环境。开发文件是一个 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。devfile 特定于为其创建的存储库，您可以为每个存储库创建一个单独的开发文件。您的开发环境支持命令和事件，并提供默认的通用开发文件镜像。

如果您使用空蓝图创建项目，则可以手动创建开发文件。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和开发文件中提取的文件。该/home目录在您首次创建开发环境时为空，用于存储您在使用开发环境时创建的文件。开发环境/projects和/home目录中的所有内容都永久存储。

> **Note**
>
> 只有在更改 devfile 或 devfile 组件名称时，该/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的根目录中不包含开发文件，或者您创建的开发环境没有源存储库，则默认的通用开发文件会自动应用于源存储库。所有 IDE 都使用相同的默认通用开发文件镜像。CodeCatalyst 目前支持开发文件版本 2.0.0。有关开发文件的更多信息，请参阅 Devfile 架构-版本 2.0.0。

> **Note**
>
> 您只能在 devfile 中包含公共容器镜像。

请注意，与 VPC 连接的开发环境仅支持以下开发文件镜像：

- 通用图像
- 私有 Amazon ECR 镜像（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

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开发文件的源存储库的项目。
3. 在导航窗格中，选择 “代码”。
4. 选择 “源存储库”。
5. 选择包含要编辑的开发文件的源存储库。
6. 从文件列表中选择`devfile.yaml`文件。
7. 选择编辑。
8. 编辑开发文件。
9. 选择 “提交”，或者创建拉取请求，以便团队成员可以查看和批准更改。

> **Note**
>
> 如果您编辑开发文件，则必须重新启动开发文件才能使更改生效。这可以通过运行来完成`/aws/mde/mde start --location devfile.yaml`。如果启动开发文件时出现问题，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开发文件，则必须重新启动开发环境才能使更改生效。

你可以通过运行`/aws/mde/mde status`来查看正在使用哪个 devfile。位置字段包含开发文件相对于环境`/projects`文件夹的路径。

```
{
            "status": "STABLE",
            "location": "devfile.yaml"
        }
```

您也可以将默认开发文件`/projects/devfile.yaml`移入您的源代码存储库。要更新开发文件的位置，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开发文件

要更改开发环境的配置，必须编辑开发文件。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如果您在支持的 IDE 中编辑开发文件，则必须提交更改并将其推送到源存储库或创建拉取请求，以便团队成员可以查看和批准对开发文件的编辑。

- 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开发文件
- 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。有关开发文件的更多信息，请参阅 Devfile 架构-版本 2.0.0。

功能	类型
exec	命令
postStart	事件
container	组件
args	组件属性
env	组件属性
mountSources	组件属性
volumeMounts	组件属性

开发环境的开发文件示例

以下是一个简单的开发文件示例。

```
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
```

Devfile 的启动、命令和事件日志会被捕获并存储在中。/aws/mde/logs要调试开发文件行为，请使用有效的开发文件启动开发环境并访问日志。

使用恢复模式对存储库开发文件进行故障排除

如果启动你的开发文件时出现问题，它将进入恢复模式，这样你仍然可以连接到你的环境并修复你的开发文件。在恢复模式下，运行时/aws/mde/mde status不会包含开发文件的位置。

```
{
            "status": "STABLE"
        }
```

你可以检查下方日志中的错误/aws/mde/logs，修复开发文件，然后重试运行/aws/mde/mde start。

为开发环境指定通用开发文件镜像

默认通用映像包括可用于您的 IDE 的最常用的编程语言和相关工具。如果未指定图像，则 CodeCatalyst 提供此图像并包含由维护的工具 CodeCatalyst。要随时获得有关新图片发布的通知，请参阅通过 SNS 订阅通用图像通知。

> **Note**
>
> `public.ecr.aws/aws-mde/universal-image:latest`图像获取`public.ecr.aws/aws-mde/universal-image:3.0`图像。

Amazon CodeCatalyst 支持以下开发者文件镜像。

图像版本	映像标识符
`Universal image 1.0`	`public.ecr.aws/aws-mde/universal-image:1.0`
`Universal image 2.0`	`public.ecr.aws/aws-mde/universal-image:2.0`
`Universal image 3.0`	`public.ecr.aws/aws-mde/universal-image:3.0`

> **Note**
>
> 如果你使用的是 AWS Cloud9，升级到后自动完成功能将不适用于 PHP、Ruby 和 CSS。`universal-image:3.0`

主题

通过 SNS 订阅通用图像通知

CodeCatalyst 提供通用的图像通知服务。您可以使用它来订阅亚马逊简单通知服务 (SNS) Simple Notification Service 主题，该主题将在 CodeCatalyst 通用图像更新发布时通知您。有关 SNS 主题的更多信息，请参阅什么是 Amazon 简单通知服务？ 。

每当发布新的通用图像时，我们都会向订阅者发送通知；本节介绍如何订阅 CodeCatalyst 通用图像更新。

消息示例

```
{
    "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 通用图像更新

1. 在控制面板中打开 Amazon SNS 控制台。
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 Endpoint 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. 您将收到一封主题为 “AWS 通知-订阅确认” 的确认电子邮件。打开电子邮件并选择确认订阅。

使用 Amazon SNS 控制台取消订阅 CodeCatalyst 通用图像更新

1. 在控制面板中打开 Amazon SNS 控制台。
2. 在导航栏中，选择您的 AWS 区域。
3. 在导航窗格中，选择 “订阅”，然后选择要取消订阅的订阅。
4. 选择 “操作”，然后选择 “删除订阅”。
5. 选择删除。

通用镜像 1.0 运行时版本

下表列出了可用的运行时。`universal-image:1.0`

`universal-image:1.0`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 撰写	2.16	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.19	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	14.20	`nodejs: 14.x`
	16.19	`nodejs: 16.x`
openssl	1.0	`openssl: 1.x`

运行时名称	版本	特定主要和最新次要版本
	1.1	
php	7.2	`php: 7.x`
python	3.9	`python: 3.x`
	3.10	
ruby	3.1	`ruby: 3.x`
terraform	1.4	`terraform: 1.x`

通用镜像 2.0 运行时版本

下表列出了可用的运行时。`universal-image:2.0`

`universal-image:2.0`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 撰写	2.17	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.20	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	16.19	`nodejs: 16.x`
openssl	1.0	`openssl: 1.x`
	1.1	

运行时名称	版本	特定主要和最新次要版本
php	7.2	`php: 7.x`
python	3.9	`python: 3.x`
	3.10	
ruby	3.2	`ruby: 3.x`
terraform	1.4	`terraform: 1.x`

通用镜像 3.0 运行时版本

下表列出了可用的运行时。`universal-image:3.0`

`universal-image:3.0`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 撰写	2.17	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.21	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	18.17	`nodejs: 18.x`
	20.6	`nodejs: 20.x`
openssl	3.0	`openssl: 3.x`
php	8.2	`php: 8.x`

运行时名称	版本	特定主要和最新次要版本
python	3.9	`python: 3.x`
	3.11	
ruby	3.2	`ruby: 3.x`
terraform	1.5	`terraform: 1.x`

开发文件命令

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 文档中的添加命令。

以下示例向您展示了如何在开发文件中指定`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，你可以使用 streaming 标志`-s`来实时输出命令的执行情况。

```
/aws/mde/mde -s command <command-id>
```

> **Note**
>
> command-id必须是小写字母。

支持的执行参数 CodeCatalyst

CodeCatalyst 在 devfile 版本 2.0.0 上支持以下exec参数。

- commandLine
- component
- id
- workingDir

开发文件事件

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。

以下示例向您展示了如何在开发文件中添加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定义的顺序执行指定的postStart命令。如果命令失败，开发环境将继续运行，执行输出存储在下的日志中/aws/mde/logs。

开发文件组件

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 文档中的添加组件。

以下示例向您展示了如何在 devfile 中向容器中添加启动命令。

```
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      command: ['sleep', 'infinity']
```

> **Note**
>
> 当容器有短暂的入口命令时，必须包含command: ['sleep', 'infinity']以保持容器运行。

CodeCatalyst 还支持容器组件中的以下属性：argsenv、mountSources、和volumeMounts。

将 VPC 连接与开发环境关联

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程操作并创建符合网络规则并可以访问关联 VPC 中资源的开发环境。

只有在创建开发环境后，您才能将开发环境与 VPC 连接相关联。创建开发环境后，您无法更改与其关联的 VPC 连接。如果您想使用不同的 VPC 连接，则必须删除当前的开发环境并创建新的开发环境。

> **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。

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 VPC 的更多信息，请参阅《CodeCatalyst 管理员指南》 CodeCatalyst中的管理 Amazon VPC。

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，请参阅的配额 CodeCatalyst。

每月开发环境小时数	开发环境的时间受空间总体存储限制的影响。有关更多信息，请参阅 定价 和 开发环境问题疑难解答。
每个空间的开发环境存储量	开发环境存储受到空间总体存储限制的影响。有关更多信息，请参阅 定价 和 开发环境问题疑难解答。
开发环境计算量	开发环境的计算受空间总体存储限制的影响。有关更多信息，请参阅 定价 和 开发环境问题疑难解答。

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。

CodeCatalyst 支持以下软件包格式：

- npm

可以发现包存储库中的包，并在包含该存储库的项目成员之间共享。

要将软件包添加到存储库，请将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。

您可以配置 CodeCatalyst 工作流程，将包发布到软件包存储库并使用来自软件包存储库的 CodeCatalyst 包。有关在工作流程中使用包的更多信息，请参阅使用工作流程发布和导入包。

您可以通过将一个包存储库中的包添加为上游存储库，使其可供同一个项目中的另一个存储库使用。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。

您可以通过将公共的外部 CodeCatalyst 存储库连接到存储库来使开源软件包可供其使用。有关上游存储库和连接到外部存储库的更多信息（包括支持的存储库列表），请参阅配置和使用上游存储库。

主题

套餐概念

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。

一揽子包含：

- 一个名字（例如，webpack是一个流行的 npm 包的名称）
- 一个可选的命名空间（例如，@types在@types/node）
- 一组版本（例如、1.0.01.0.1、1.0.2）
- 包级元数据（例如 npm dist 标签）

Package 命名空间

某些软件包格式支持分层包名，以将软件包组织成逻辑组并帮助避免名称冲突。具有相同名称的包可以存储在不同的命名空间中。例如，npm 支持作用域，而 npm 包的作用域@types/node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。如果您无法对软件包名称进行分组，则可能更难避免名称冲突。

软件包版本

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

资产是存储在中的单个文件 CodeCatalyst ，它与软件包版本相关联，例如 npm .tgz 文件。

Package 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。每个包存储库都提供端点，用于使用 Node.js CLI (npm) 之类的工具获取和发布包。每个空间中最多可以创建 1,000 个软件包存储库。

您可以使用上游存储库将软件包存储库链接到另一个存储库。当您将包存储库链接为上游存储库时，可以通过配置的存储库使用链接存储库中的包。有关更多信息，请参阅 上游存储库。

Gateway 存储库是一种特殊类型的软件包存储库，用于从官方外部软件包授权机构提取和存储软件包。有关更多信息，请参阅 网关存储库。

网关存储库

网关存储库是一种特殊类型的软件包存储库，它与支持的外部官方软件包授权机构相连。当您将网关存储库添加为上游存储库时，您可以从相应的官方软件包授权机构使用软件包。您的下游存储库不与公共存储库通信，相反，所有内容都由网关存储库进行中介。以这种方式使用的软件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，但必须在每个项目中创建它们才能使用。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。

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包存储库端点访问软件包存储库中包含的软件包版本时，软件包存储库就是另一个软件包存储库的上游。通过上游关系，从客户端的角度来看，两个软件包存储库的内容可以有效地合并。

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。

配置和使用软件包存储库

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

- 创建软件包存储库
- 连接到软件包存储库
- 编辑软件包存储库
- 删除软件包存储库

创建软件包存储库

要在中创建软件包存储库，请执行以下步骤 CodeCatalyst。

创建软件包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建包存储库的项目。
3. 从导航窗格中选择 “软件包”。
4. 在 Pac kage 存储库页面上，选择创建包存储库。
5. 在 Pac kage 存储库详情部分，添加以下内容：
   a. 存储库名称。考虑使用描述性名称，其中包含诸如您的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）存储库描述。当一个项目中有多个跨多个团队的多个仓库时，仓库描述特别有用。
6. 在 “编辑上游存储库” 部分，添加要通过软件包存储库访问的所有 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 软件包存储库。
   - 当从软件包存储库请求软件包时，将按照上游存储库在此列表中显示的顺序对其进行搜索。找到包裹后， CodeCatalyst 将停止搜索。要更改上游存储库的顺序，可以将存储库拖放到列表中，也可以使用 “重新排序” 按钮。
7. 选择 “创建” 以创建您的软件包存储库。

连接到软件包存储库

要将包发布到 CodeCatalyst 或从中使用 CodeCatalyst，您必须使用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，则可以按照中的说明进行创建创建软件包存储库。

有关如何将 npm 包管理器连接到软件 CodeCatalyst 包存储库的说明，请参阅配置和使用 npm。

编辑软件包存储库

执行以下步骤来编辑软件包存储库的描述和上游存储库。

编辑软件包存储库

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

2. 导航到包含要编辑的包存储库的项目。
3. 从导航窗格中选择 “软件包”。
4. 在 Pac kage 存储库页面上，选择要删除的仓库。
5. 选择 “操作” 下拉列表并选择 “编辑”。
6. 编辑存储库描述和上游存储库。有关上游存储库的更多信息，请参阅配置和使用上游存储库。
7. 选择保存。

删除软件包存储库

要在中删除软件包存储库，请执行以下步骤 CodeCatalyst。

删除软件包存储库

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到包含要删除的包存储库的项目。
3. 从导航窗格中选择 “软件包”。
4. 在 Pac kage 存储库页面上，选择要删除的仓库。
5. 选择 “操作” 下拉列表并选择 “删除”。
6. 查看所提供的有关删除软件包存储库的效果的信息。
7. 在`delete`输入字段中输入并选择 “删除”。

配置和使用上游存储库

您可以将网关存储库和其他软件包存储库作为上游连接到您的软件包存储库。这使包管理器客户端能够使用单个软件包存储库端点访问包含在多个软件包存储库中的软件包。以下是使用上游存储库的主要好处：

- 您只需要为包管理器配置一个存储库端点，即可从多个来源提取数据。
- 从上游存储库中消耗的包存储在下游存储库中，这样可以确保即使上游存储库出现意外中断，您的软件包也可用。

您可以在创建软件包存储库时添加上游存储库。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，该包存储库将连接到网关存储库的相应公共包存储库。有关支持的公共软件包存储库的列表，请参阅支持的外部软件包存储库及其网关存储库。

您可以将多个存储库链接到一起作为上游存储库。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team-repo作为上游存储库添加到project-repo。在这种情况下，您只需将包管理器配置project-repo为用于从project-repo、team-reponpm-public-registry-gateway、和中提取包即可npmjs.com。

主题

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 Packages (程序包)。
2. 在 Pac kage 存储库页面上，选择要向其添加上游存储库的软件包存储库。
3. 选择 “操作” 下拉菜单并选择 “编辑”。
4. 在上游存储库部分，选择添加上游存储库。
5. 选择搜索栏以显示和搜索可用存储库列表。您可以将支持的公共包存储库或其他 CodeCatalyst 存储库添加为上游存储库。找到要添加的存储库后，请从列表中进行选择。

> **Note**
>
> 在 Gateway 存储库npm-public-registry-gateway中，有一个存储库。要连接到公共外部包颁发机构（例如 npmjs.com），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。这样可以节省时间和数据传输，因为项目中的所有软件包存储库都将使用网关存储库中的包。

6. 选择所有要添加为上游存储库的存储库后，选择添加。
7. 有关更改上游存储库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包管理器从上游存储库获取软件包。您无需更新软件包管理器配置。有关从上游存储库请求软件包版本的更多信息，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软件包的顺序。

编辑上游存储库的搜索顺序

1. 在导航窗格中，选择 Packages (程序包)。
2. 在 “存储库” 页面上，选择要编辑其上游存储库搜索顺序的包存储库。
3. 选择 “操作” 下拉列表并选择 “编辑”。
4. 在上游存储库部分，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，或者使用重新排序按钮。
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，软件包管理器（例如）从名downstream为npm且具有多个上游存储库的软件包存储库中请求软件包版本。当请求包裹时，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。
- 如果上游存储库downstream或其上游存储库都不包含软件包版本，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。

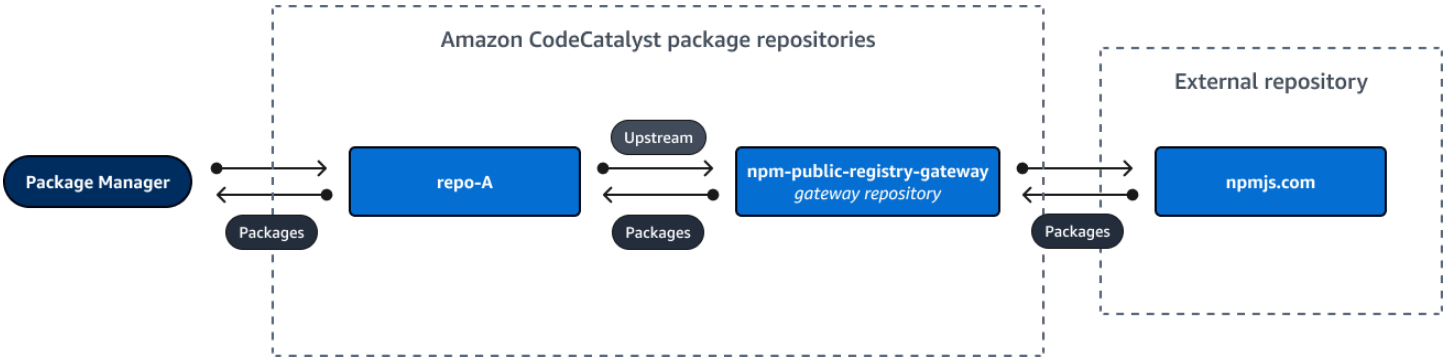
上游存储库的程序包保留

如果在上游存储库中找到了请求的软件包版本，则会保留对该版本的引用，并且在请求该版本的存储库中始终可用。这样可以确保在上游存储库意外中断时您可以访问您的软件包。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）。

通过上游关系获取软件包

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 https://npmjs.com 有上游连接。



如果配置npm为使用`repo-A`存储库，则运行`npm install`会启动将包从 https://npmjs.com 复制到`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
```

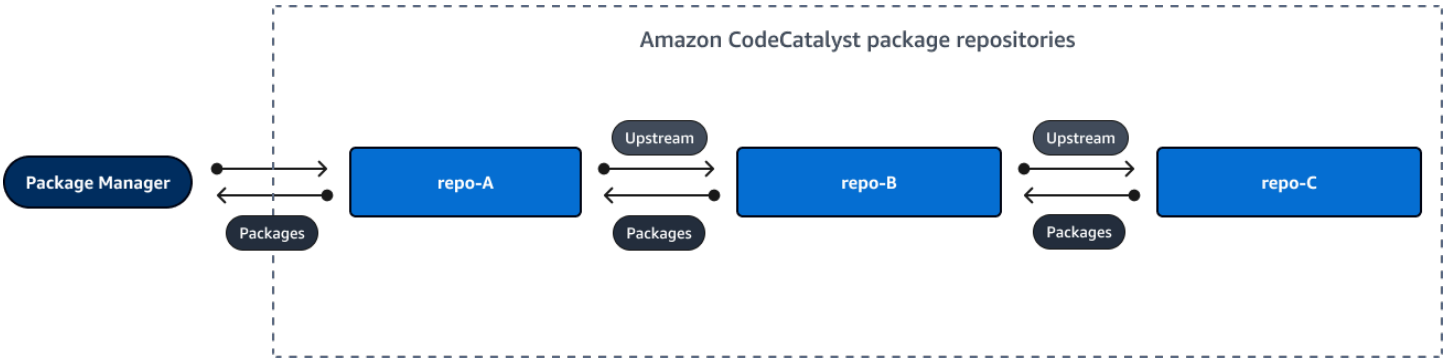
运行后npm install，仅repo-A包含最新版本 (lodash 4.17.20)，因为这是npm从中repo-A获取的版本。

由于与 https://npmjs.com npm-public-registry-gateway 有外部上游连接，所以从 https://npmjs.com 导入的所有软件包版本都存储在中npm-public-registry-gateway。这些软件包版本可以由任何与之有上游连接的下游存储库获取。npm-public-registry-gateway

的内容为您npm-public-registry-gateway提供了一种查看随着时间的推移从 https://npmjs.com 导入的所有软件包和软件包版本的方法。

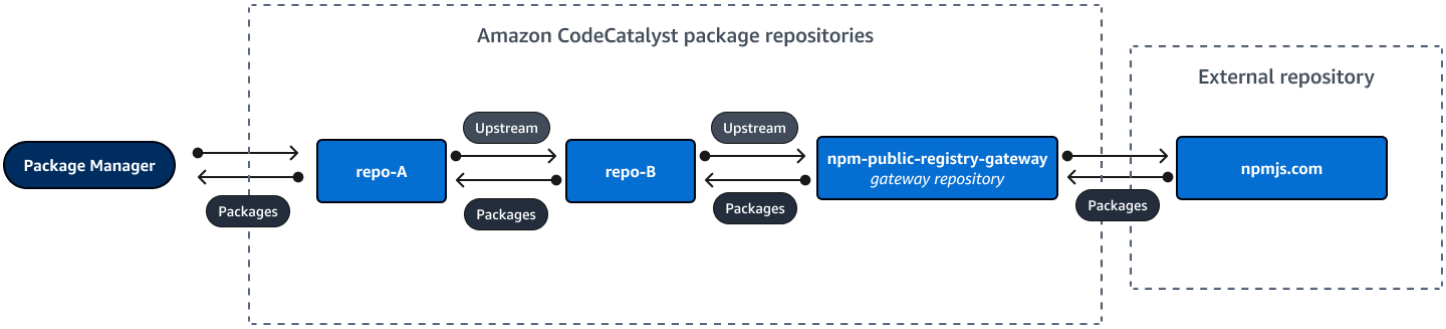
中间存储库中的程序包保留

CodeCatalyst 允许您链接上游存储库。例如，repo-A可以repo-B作为上游存储库，repo-B也可以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。



当包管理器连接到存储库repo-A并从存储库中获取软件包版本时repo-C，软件包版本不会保留在存储库repo-B中。软件包版本仅保留在最远的下游存储库中，在本示例中为。repo-A它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-Brepo-Crepo-D、和，并且连接的包管理器从中repo-A获取软件包版本，则包版本将保留在或中repo-D，repo-A但不保留在repo-B或repo-C中。

从公共包存储库中提取包版本时，软件包保留行为类似，唯一的不同是包版本始终保留在与公共仓库有直接上游连接的网关存储库中。例如，repo-A具有repo-B作为上游存储库。 repo-Bnpm-public-registry-gateway作为上游存储库，它与公共存储库 npmjs.com 有上游连接；参见下图。



如果连接到的软件包管理器**repo-A**请求特定的软件包版本，例如 l odash 4.17.20，而该软件包版本不存在于三个存储库中的任何一个存储库中，则将从 npmjs.com 获取该版本。当获取 l odash 4.17.20 时，它会保留在中，**repo-A**因为这是最远的下游存储库，并且**npm-public-registry-gateway**它与公共外部存储库 npmjs.com 有上游连接。 l@@ odash 4.17.20 没有保留在中，repo-B因为那是一个中间存储库。

移除上游存储库

如果您不想再访问上游存储库中的软件包，可以将上游存储库从软件包存储库中移除。

> **Warning**
>
> 移除上游存储库时，可能会中断上游关系链，这可能会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 Packages (程序包)。
2. 在 Pac kage 存储库页面上，选择要从中移除上游存储库的包存储库。
3. 选择 “操作” 下拉列表并选择 “编辑”。
4. 在上游存储库部分，找到要删除的上游存储库，然后选择移除。
5. 删除完上游存储库后，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软件包。这样可以节省时间和数据传输，因为项目中的所有软件包存储库都使用网关存储库中的包。

使用网关存储库连接到公共存储库

1. 在导航窗格中，选择 Packages (程序包)。
2. 在 Pac kage 存储库页面的 Gateway 存储库中，您可以查看支持的网关存储库列表及其描述。要使用网关存储库，必须先创建它。如果网关存储库已创建，则会显示其创建日期和时间。如果没有，请选择 “创建” 进行创建。
3. 选择要连接到公共存储库的软件包存储库。
4. 选择 “操作” 下拉菜单并选择 “编辑”。
5. 要连接到公共存储库，请添加与您要作为上游存储库连接的公共存储库对应的网关存储库。

   在 “编辑上游存储库” 部分中，选择 “添加 CodeCatalyst 存储库”。
6. 网关存储库部分列出了所有可用的网关存储库。当您找到与要连接的公共外部存储库相对应的网关存储库时，请从列表中选择该存储库并选择添加。
7. 向存储库请求包时，按上游存储库在 “编辑上游存储库” 列表中显示的顺序 CodeCatalyst 搜索上游存储库。找到包裹后， CodeCatalyst 停止搜索。要更改上游存储库的顺序，请将存储库拖放到列表中，或者使用重新排序箭头。
8. 完成添加和订购上游存储库后，选择 “保存”。

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包管理器从与其对应的公共外部包存储库中获取软件包。您无需更新软件包管理器配置。以这种方式使用的软件包同时存储在网关存储库和本地软件包存储库中。有关从上游存储库请求软件包版本的更多信息，请参阅请求包含上游存储库的程序包版本。

支持的外部软件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。

存储库包类型	描述	网关存储库名称
npm	npm 公有注册表	`npm-public-registry-gateway`

发布和修改软件包

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。 CodeCatalyst 支持 npm 包格式。本节提供有关发布、查看和删除包以及更新包版本状态的信息。

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。发布软件包版本的步骤如下：

将包版本发布到 CodeCatalyst 包存储库

1. 如果还没有，请创建一个软件包存储库。
2. 将您的包管理器连接到您的包存储库。有关如何将 npm 包管理器连接到软件 CodeCatalyst 包存储库的说明，请参阅配置和使用 npm。
3. 使用连接的软件包管理器发布您的软件包版本。

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 包发布到包存储库 `lodash@1.0`myrepo，并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软件包名称和版本相同的包，这可能会导致意外行为。

您仍然可以发布上游存储库中存在的软件包名称的不同版本。例如，如果存在`lodash@1.0`于上游存储库中，但`lodash@1.1`不存在，则可以发布`lodash@1.1`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。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其中包含不同内容的软件包资产。由于 npm 仅支持每个包版本的单个资源，因此要修改已发布的软件包版本，必须先将其删除。

查看软件包版本详情

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。

查看软件包版本详情

1. 在导航窗格中，选择 Packages (程序包)。
2. 在 P ackage 存储库页面上，选择包含要查看其详细信息的软件包版本的存储库。
3. 在 “包” 表中搜索软件包版本。您可以使用搜索栏按软件包名称筛选软件包。从列表中选择软件包。
4. 在 Pac kage 详情页面中，选择 Versions，然后选择要查看的版本。

删除软件包版本

您可以从 CodeCatalyst 控制台的 Package 版本详情页面删除软件包版本。

删除软件包版本

1. 在导航窗格中，选择 Packages (程序包)。
2. 在 P ackage 存储库页面上，选择包含要删除的软件包版本的存储库。
3. 搜索并从表格中选择软件包。

4. 在 Pac kage 详情页面上，选择 Versions，然后选择要删除的版本。
5. 在 Pac kage 版本详情页面上，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”。

更新软件包版本的状态

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。有关软件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。

更新软件包版本的状态

1. 在导航窗格中，选择 Packages (程序包)。
2. 在 P ackage 存储库页面上，选择包含要更新其状态的软件包版本的存储库。
3. 搜索并从表格中选择软件包。
4. 在 P ackage 详情页面上，选择版本，然后选择要查看的版本。
5. 在 Pack age 版本详情页面上，选择 “操作”，然后选择 “取消列出”、“存档” 或 “处置”。有关每个软件包版本状态的信息，请参见程序包版本状态。
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 “取消列出”、“存档” 或 “处置”。

程序包版本状态

以下是软件包版本状态的可能值。您可以在控制台中更改软件包版本状态。有关更多信息，请参阅 更新软件包版本的状态。

- 已发布：软件包版本已成功发布，可以由软件包管理器申请。软件包版本将包含在返回给软件包管理器的软件包版本列表中；例如，在的输出中`npm view <package-name> versions`。程序包版本的所有资产均可从存储库中获得。
- 未列出：软件包版本资产可供从存储库下载，但软件包版本未包含在返回给包管理器的版本列表中。例如，对于 npm 软件包，的输出`npm view <package-name> versions`不包括软件包版本。这意味着 npm 依赖关系解析逻辑不会选择软件包版本，因为该版本未出现在可用版本列表中。但是，如果`npm package-lock.json`文件中已经引用了未列出的软件包版本，则仍然可以下载和安装该版本；例如，在运行`npm ci`时。
- 已存档：无法下载包版本资产。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。如果您的应用程序版本依赖于更新为已存档的版本，

则除非软件包版本已在本地缓存，否则构建将失败。您不能使用包管理器或构建工具重新发布存档包版本，因为它仍然存在于存储库中。但是，您可以在控制台中将软件包版本状态更改回 “未上市” 或 “已发布”。

- 已处置：软件包版本未出现在列表中，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。无法使用软件包版本，因为资源已被删除。当软件包版本被标记为 “已处置” 时，您无需支付软件包资产的存储费用。

除了上述列表中的状态外，还可以删除软件包版本。删除包版本后，存储库中将不包含该软件包版本，您可以使用包管理器或构建工具自由地重新发布该软件包版本。

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从外部公共存储库中提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共存储库提取来添加软件包的版本，则您很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。要保护自己免受依赖关系替换攻击，请在存储库中的软件包上配置软件包来源控件，以限制将该软件包的版本添加到存储库的方式。

您应该考虑配置软件包来源控件，使不同软件包的新版本既来自内部来源（例如直接发布），也来自外部来源（例如公共存储库）。默认情况下，包源控制是根据软件包的第一个版本添加到存储库的方式来配置的。

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。

Publish

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

软件包的默认软件包来源控件将基于该软件包的第一个版本添加到软件包存储库中的方式。

- 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

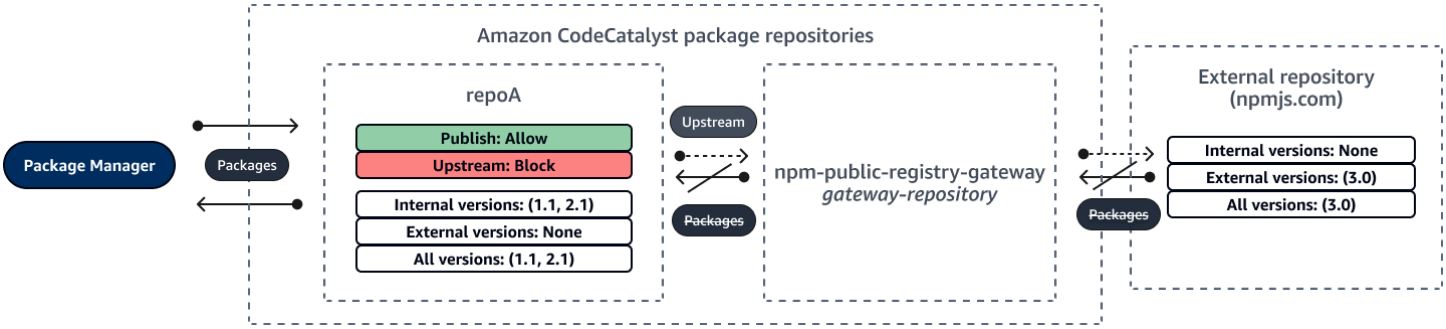
本节介绍将软件包版本添加到软件 CodeCatalyst 包存储库的一些常见场景。Package Origin 控制设置是根据第一个包版本的添加方式为新包设置的。

在以下情况下，内部软件包将直接从包管理器发布到您的存储库，例如您维护的软件包。外部程序包是存在于公有存储库中的程序包，可以通过外部连接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。这可能是针对内部软件包的企图依赖替换攻击，也可能是巧合。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。

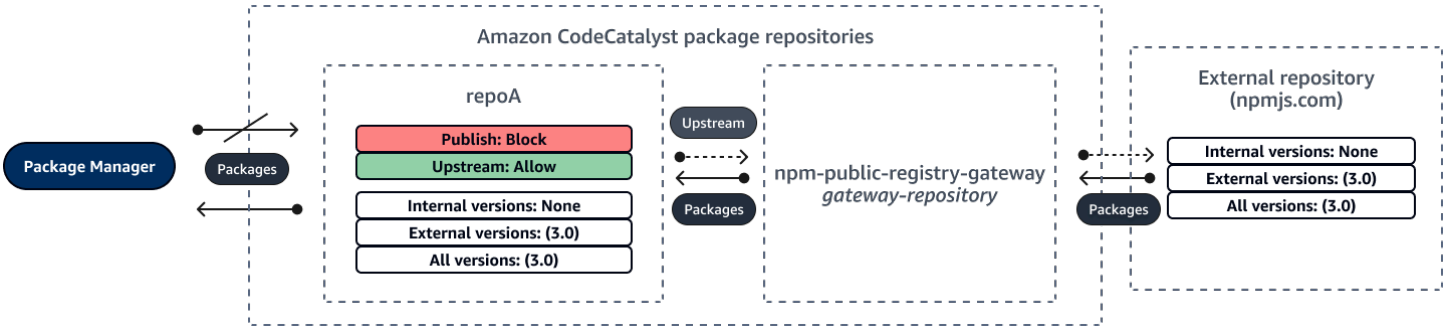
在下图中，RepoA 是您的 CodeCatalyst 软件包存储库，它与公共存储库有外部连接。您的存储库包含 packageA 的版本 1.1 和 2.1，但版本 3.0 已发布到公有存储库。通常，RepoA 会在软件包管理器请求软件包后提取版本 3.0。由于软件包提取设置为 “阻止”，因此版本 3.0 不会提取到您的 CodeCatalyst 软件包存储库中，也无法供与其连接的包管理员使用。



已为现有外部程序包发布内部程序包版本

在这种情况下，包名为 packageB 的外部存在于您已连接到存储库的公共存储库中。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。你可能不知道公共包并试图发布一个名字不相关的包，或者你可能正在尝试发布一个经过补丁的版本，或者你可能正在尝试直接发布外部已经存在的确切包版本。 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，它与公共存储库有外部连接。您的软件包存储库包含从公共存储库中提取的 3.0 版本。您想将 1.2 版本发布到您的软件包存储库。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 1.2 版。



发布现有外部程序包的已修补程序包版本

在这种情况下，包名为 packageB 的外部存在于您已连接到软件包存储库的公共存储库中。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。您的团队决定将此软件包的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

Package Origin 控制是根据将软件包的第一个软件包版本添加到包存储库中的方式自动配置的。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。

添加或编辑包裹来源控制

1. 在导航窗格中，选择 Packages (程序包)。
2. 选择包含要编辑的软件包的软件包存储库。
3. 在 “包” 表格中，搜索并选择要编辑的包。
4. 在包裹摘要页面上，选择编辑起源控件。
5. 在 Origin 控件中，选择要为此包设置的包裹来源控制。必须同时设置两个包源控制设置，即 “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，假设你想将 npm 包`lodash@1.0`发布到存储库`myrepo`，并且`myrepo`有一个与 npmjs.com 有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。由于无法访问包版本，您可以`lodash`将任何尚不存在的版本发布到存储库中。
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。

依赖替换攻击属于被称为软件供应链攻击的攻击子集。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。这会导致所需的软件包和恶意软件包之间 “混淆” 或 “替换”，从而导致代码被泄露。

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

- 配置和使用 npm
- npm 标签处理

配置和使用 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

使用 npm 配置 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 npm 的更多信息，请参阅 np m 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 Packages (程序包)。
4. 从列表中选择您的软件包存储库。
5. 选择 “连接到存储库”。
6. 在配置详细信息中，在 Pack age Manager 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。如果您已经有代币，则可以使用它。如果不是，则可以使用以下步骤创建一个。

   a. （可选）：更新 PAT 名称和到期日期。

   b. 选择创建令牌。

c. 将您的 PAT 复制并存储在安全的地方。

> **Warning**
>
> 关闭对话框后，您将无法再次查看或复制您的 PAT。凭证的有效期应该是短的，以最大限度地减少攻击者在盗用凭据后可以使用凭证的时间长度。

9. 从项目的根目录运行以下命令，使用软件包存储库配置 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，请创建该文件。
- 将软件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**
>
> 如果您要从控制台说明中复制，则以下命令中的值会为您更新，无需更改。

- 用您的*###*替换 CodeCatalyst 用户名。
- 将 *PAT* 替换为您的 CodeCatalyst PAT。
- 将 *space_name ##### CodeCatalyst ####*。
- 将 *proj_name ##### CodeCatalyst ####*。
- 将 re *po_name ##### CodeCatalyst ########*。

```
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

按照中的步骤将 npm 连接到存储库后使用 npm 配置 CodeCatalyst，即可在存储库上运行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，方法是为存储库配置与连接到 npmjs.com 的网关存储库的上游连接。npm-public-registry-gateway从 npmjs 安装的软件包会被提取并存储在网关存储库和最远的下游软件包存储库中。

从 npmjs 安装软件包

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 npm 配置 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命令安装软件包。

```
npm install package_name
```

有关从上游存储库请求软件包的更多信息，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 CodeCatalyst 包存储库

完成后使用 npm 配置 CodeCatalyst，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 包的信息，请参阅在 npm Docs 上创建 Node.js 模块。

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

- 支持的与软件包存储库交互的命令
- 支持的客户端命令
- 不受支持的命令

支持的与软件包存储库交互的命令

本节列出了npm客户端向其配置的注册表发出一个或多个请求的npm命令（例如，`npm config set registry`）。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	描述
bugs	猜测软件包的错误跟踪器网址的位置，然后它会尝试将其打开。
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	猜测软件包的文档 URL 的位置，然后它会尝试使用 `--browser` config 参数将其打开。
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。

命令	描述
install-ci-test	从零开始安装一个项目并运行测试。别名：npm cit。此命令运行一个npm ci，然后紧接着运行npm test。
install-test	安装程序包并运行测试。运行一个npm install，然后立即运行npm test。
outdated	检查已配置的注册表以确定是否有任何已安装的软件包已过期。
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。
update	猜测软件包的存储库 URL 的位置，然后它尝试使用 --browser config 参数将其打开。
view	显示程序包元数据。也可以用来打印元数据属性。

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	描述
垃圾桶（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。

命令	描述
dedupe	搜索本地包树，并尝试通过将依赖关系进一步向上移动来简化结构，使多个依赖包可以更有效地共享依赖关系。
edit	编辑已安装的程序包。在当前工作目录中选择一个依赖关系，然后在默认编辑器中打开该包目录。
explore	浏览已安装的程序包。在指定已安装软件包的目录中生成一个子外壳。如果指定了命令，则该命令将在子外壳中运行，然后子外壳会立即关闭。
help	获取有关 npm 的帮助。
help-search	搜索 npm 帮助文档。
init	创建 `package.json` 文件。
link	符号链接软件包目录。
ls	列出已安装的程序包。
pack	将程序包打包成 tarball。
prefix	显示前缀。除非同时指定，否则`-g`这是最接近包含`package.json` 文件的父目录。
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	运行软件包的停止、重启和启动脚本以及相关的预脚本和后脚本。
根	将有效`node_modules` 目录打印为标准输出。
run-script	运行任意程序包脚本。
shrinkwrap	锁定依赖项版本以供发布。

命令	描述
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	描述	注意
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。
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、删除、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 npm 配置 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。

命令	描述	注意
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。
搜索	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。
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 npm 配置 CodeCatalyst。
unpublish	从注册表中删除程序包。	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软件包。
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，并且您对每个流使用不同的标签（例如、`stablebeta`、`dev`、`canary`）。有关更多信息，请参阅 np m 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，项目仅对稳定版本使用最新标签。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，名为的存储库`R`有一个名为的上游存储库`U`。下表显示了两个存储库中都`web-helper`存在的名为的软件包的标签。

存储库	软件包名称	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 客户端从存储库中获取`web-helper`软件包的标签时`R`，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。

存储库	软件包名称	程序包标签	上次更新
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 客户端从存储库中获取包 web-helper 的标签时`R`，最新的标签将别名为 1.0.1 版本，因为它是在上次更新的。这使得通过运行可以轻松使用上游存储库中尚未存在于本地存储库中的新软件包版本`npm update`。

套餐配额

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，请参阅的配额 CodeCatalyst。

资源	默认限额
Package 存储库	每个空间最多 1000 个。
直接上游存储库	每个软件包存储库最多 10 个。
已搜索上游软件包存储库	每个请求的软件包版本最多可以搜索 25 个上游存储库。
Package 资源文件大小	每个软件包资产的最大容量为 5GB。
Package 资产	每个软件包版本最多 150 个。

使用中的工作流程构建、测试和部署 CodeCatalyst

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。自动执行此操作的方法是通过工作流程。

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

关于工作流程定义文件

工作流程定义文件是描述您的工作流程的 YAML 文件。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是.yml 或.yaml。

以下是一个简单的工作流程定义文件示例。我们将在下表中解释此示例的每一行。

```
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	描述
`Name: MyWorkflow`	指定工作流程的名称。有关该Name属性的更多信息，请参阅顶级属性。
`SchemaVersion: 1.0`	指定工作流程架构版本。有关该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:`	指定将导致工作流程运行启动的逻辑。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程。

`- Type: PUSH`
`  Branches:`
`  - main`
	表示每当您将代码推送到默认源存储库的main分支时，工作流程都必须启动。有关工作流程来源的更多信息，请参阅将工作流程连接到源存储库。

`Actions:`	定义工作流程运行期间要执行的任务。在此示例中，该Actions部分定义了一个名为的操作Build。有关操作的更多信息，请参阅配置工作流程执行的操作。
`Build:`	定义动Build作的属性。有关生成操作的更多信息，请参阅使用工作流程进行构建。
`Identifier: aws/build@v1`	为生成操作指定唯一的硬编码标识符。

`Inputs:`
`  Sources:`
`  - WorkflowSource`
	表示生成操作应在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 将工作流程连接到源存储库。

`Configuration:`	包含特定于生成操作的配置属性。

行	描述

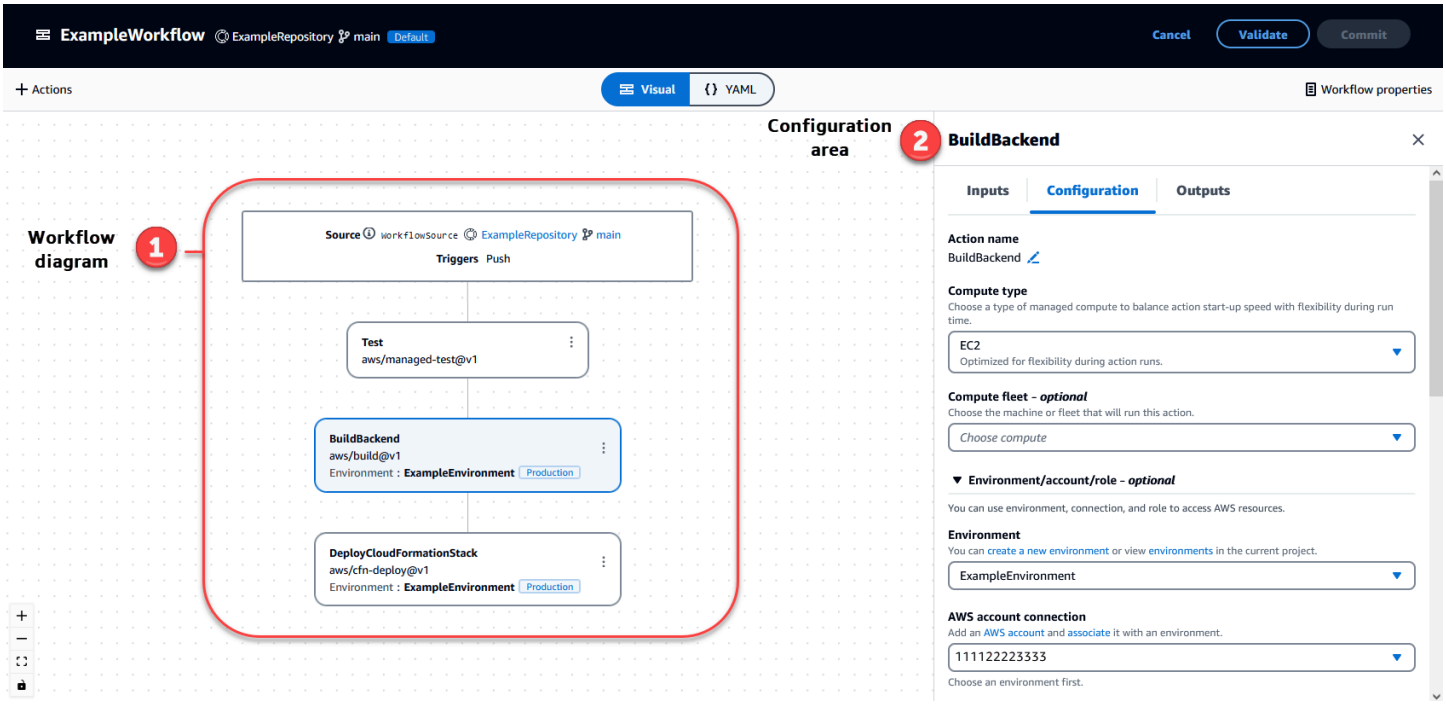
`Steps:`
`  - Run: docker build -t MyApp:lat`
`est .`
	告诉编译操作构建名为的 Docker 镜像MyApp并使用latest标记。

有关工作流定义文件中所有可用属性的完整列表，请参阅工作流程 YAML 定义。

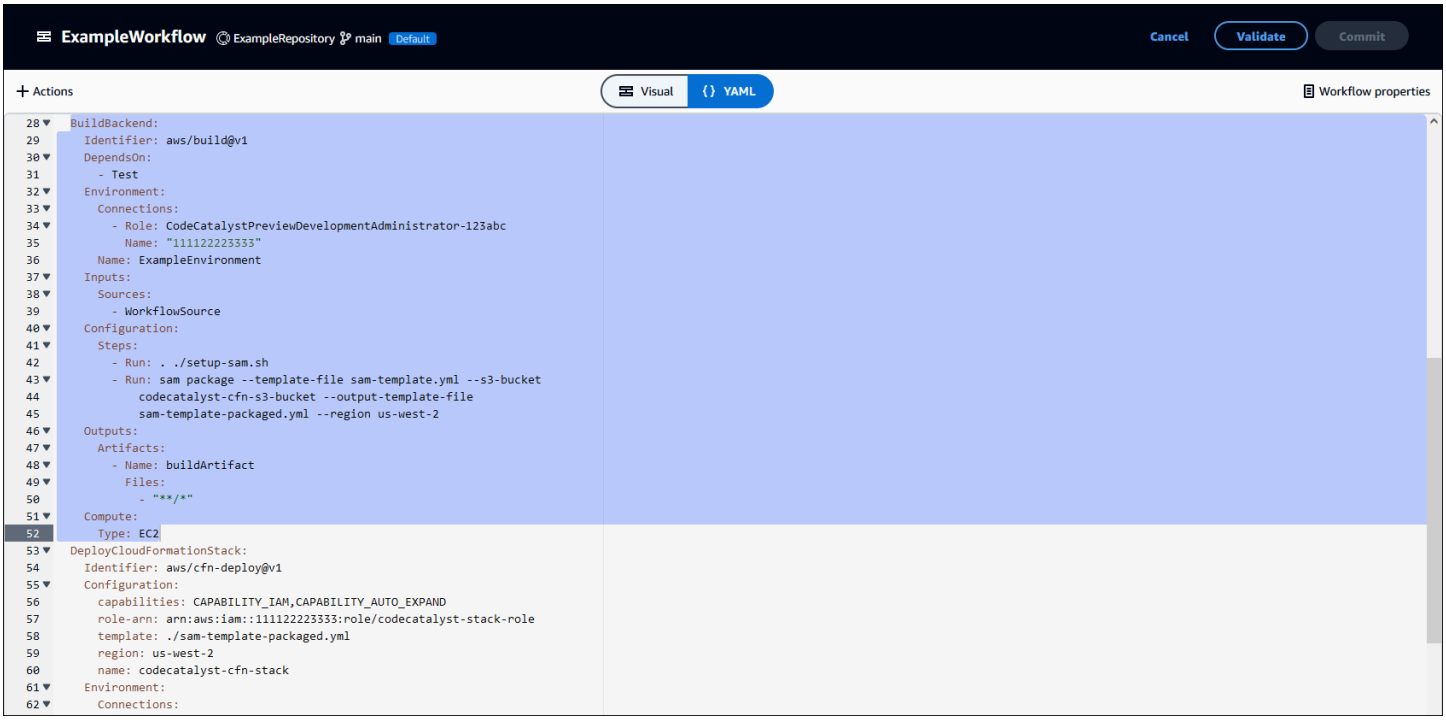
使用 CodeCatalyst 控制台的视觉编辑器和 YAML 编辑器

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。这些编辑器提供了有用的文件验证，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程。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程定义文件。可视化编辑器包括显示工作流程主要组件的工作流程图 (1) 和配置区域 (2)。



或者，您可以使用 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），或者添加可视化编辑器未提供的高级属性。

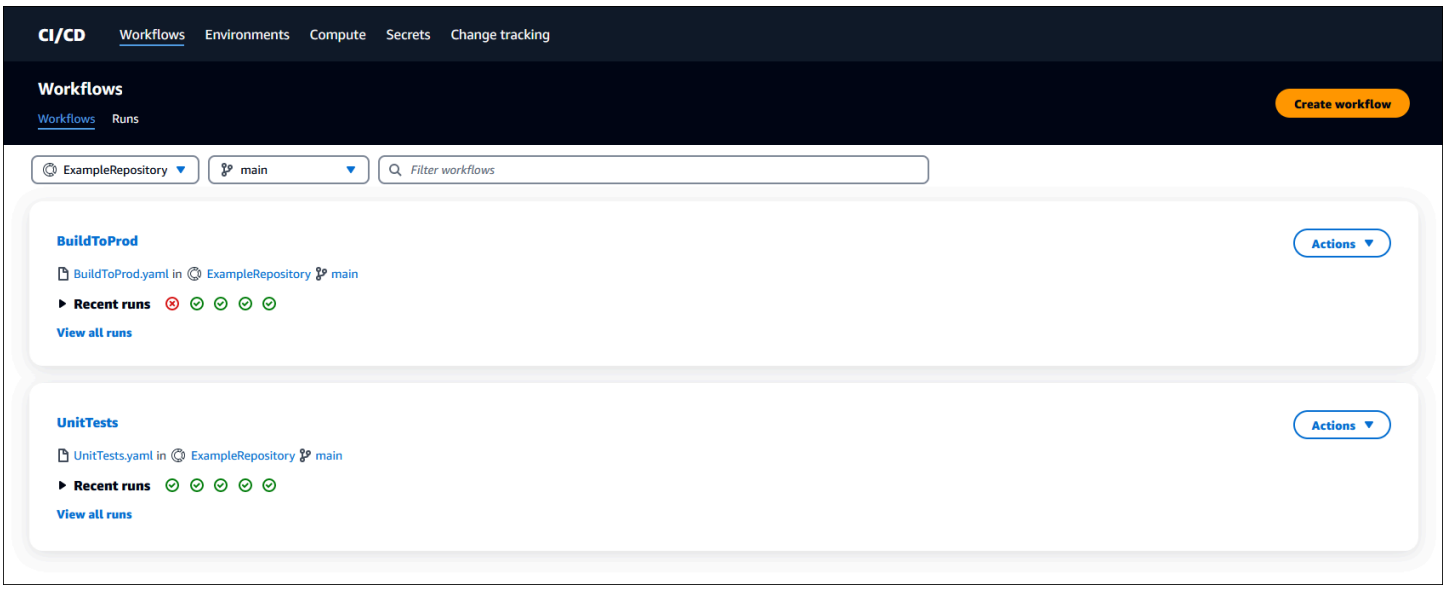


您可以从可视化编辑器切换到 YAML 编辑器，以查看您的配置对底层 YAML 代码的影响。

发现工作流程

您可以在工作流程摘要页面上查看您的工作流程，以及您在同一项目中设置的其他工作流程。

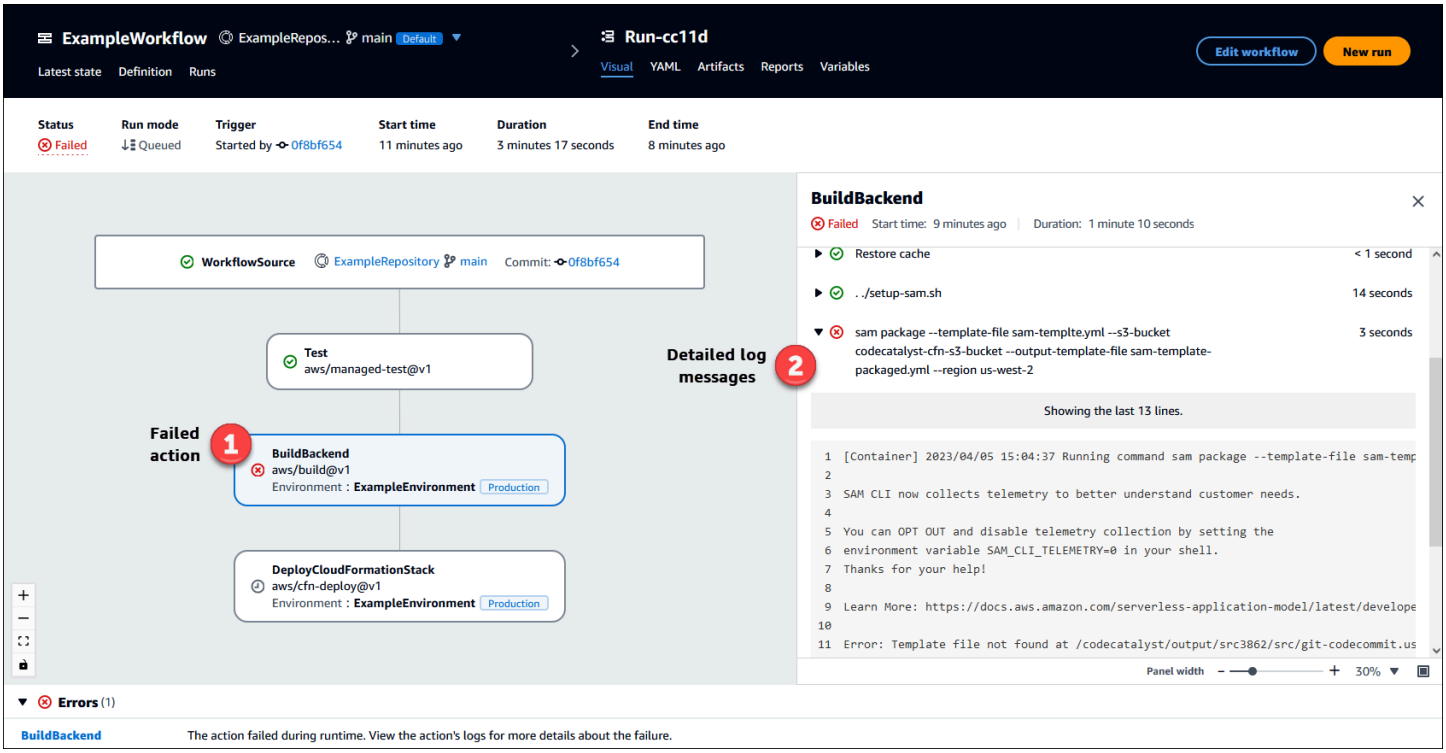
下图显示了工作流程摘要页面。它包含两个工作流程：BuildToProd和UnitTests。你可以看到两者都运行过几次。您可以选择 “最近运行” 以快速查看运行历史记录，也可以选择工作流程名称以查看工作流程的 YAML 代码和其他详细信息。



查看工作流程运行详情

您可以通过在工作流程摘要页面中选择运行来查看工作流程运行的详细信息。

下图显示了名为 run-cc11d 的工作流程运行的详细信息，该工作流程是在提交源代码时自动启动的。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对问题进行故障排除。有关工作流程运行的更多信息，请参阅运行工作流。



后续步骤

要了解有关工作流程概念的更多信息，请参阅工作流程概念。

要创建您的第一个工作流程，请参阅工作流程入门。

工作流程概念

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

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定

义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

工作流程定义文件

工作流程定义文件是描述您的工作流程的 YAML 文件。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是.yml 或.yaml。

有关工作流程定义文件的更多信息，请参阅工作流程 YAML 定义。

操作

操作是工作流程的主要组成部分，它定义了工作流程运行期间要执行的逻辑工作单元或任务。通常，一个工作流程包括多个按顺序运行或并行运行的操作，具体取决于您的配置方式。

有关操作的更多信息，请参阅配置工作流程执行的操作。

行动小组

一个操作组包含一个或多个操作。将操作分组到操作组可以帮助您保持工作流程井井有条，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行动组。

构件

构件是工作流程操作的输出，通常由文件夹或文件存档组成。构件之所以重要，是因为它们允许您在操作之间共享文件和信息。

有关构件的更多信息，请参阅 使用构件在工作流程中的操作之间共享数据。

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，请参阅为工作流程配置计算和运行时环境 Docker 镜像。

环境

不要将环境与开发环境混淆，它是代码部署到的地方。它通常包含正在运行的应用程序的实例及其关联的基础架构。您可以为环境命名，例如开发、测试、暂存或生产。 CodeCatalyst 对环境生成的任何部署都将显示在 “环境” 页面上。要设置环境，请为其命名（例如）`my-production-environment`，然后将其与您的关联 AWS 账户。

除了显示部署信息外，环境还可以作为向工作流程操作分配 AWS IAM 角色的机制。

有关环境的更多信息，请参阅部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

盖茨

门禁是一个工作流组件，除非满足某些条件，否则您可以使用它来阻止工作流程继续运行。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。

您可以在工作流程中的操作序列之间或在第一个操作（源代码下载后立即运行）之前添加门禁。如果你需要的话，你也可以在最后一个动作之后添加大门。

有关门的更多信息，请参阅对工作流程运行进行门控。

盖茨

门禁是一个工作流组件，除非满足某些条件，否则您可以使用它来阻止工作流程继续运行。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。

您可以在工作流程中的操作序列之间或在第一个操作（源代码下载后立即运行）之前添加门禁。如果你需要的话，你也可以在最后一个动作之后添加大门。

有关门的更多信息，请参阅对工作流程运行进行门控。

报告

报告包含有关工作流程运行期间发生的测试的详细信息。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。您可以使用报告来帮助解决工作流程中的问题。如果您有来自多个工作流程的许多报告，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

运行是工作流程的单次迭代。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。

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其操作所需的文件。例如，工作流程操作可能连接到源存储库以获取应用程序源文件以生成应用程序。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

Variables

变量是一个键值对，其中包含可在工作流程中引用的信息。 CodeCatalyst

有关变量的更多信息，请参阅在工作流程中配置和使用变量。

工作流程触发器

工作流程触发器，或者简称触发器，允许您在发生某些事件（例如代码推送）时自动启动工作流程运行。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导致工作流程运行启动。
- 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导致工作流程运行启动。
- 计划-计划触发器使工作流程按您定义的计划启动。可以考虑使用计划触发器来运行软件的夜间版本，以便软件开发人员在第二天早上准备好最新版本。

您可以单独使用推送、拉取请求和调度触发器，也可以在同一个工作流程中组合使用。

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程。

工作流程入门

在本教程中，您将学习如何创建和配置您的第一个工作流程。

> **Tip**
>
> 更喜欢从预配置的工作流程开始？ 请参阅使用蓝图创建项目，其中包含使用正常工作流程设置项目的说明、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 空间中，你需要一个空的、从头开始的 CodeCatalyst 项目，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，在本教程结束时，创建新的项目和源存储库可以更轻松地进行清理。

步骤 1：创建和配置您的工作流程

在此步骤中，您将创建和配置一个工作流程，该工作流程会在进行更改时自动生成和测试源代码。

创建您的工作流程

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择 “创建工作流程”。

   工作流程定义文件显示在 CodeCatalyst 控制台的 YAML 编辑器中。

配置您的工作流程

您可以在可视化编辑器或 YAML 编辑器中配置工作流程。让我们从 YAML 编辑器开始，然后切换到可视化编辑器。

1. 选择 + Actions 可查看可添加到工作流程中的工作流程操作列表。
2. 在 “构建” 操作中，选择 + 将操作的 YAML 添加到您的工作流程定义文件中。现在，您的工作流程如下所示。

```
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 Visual 可在可视化编辑器中查看工作流定义文件。可视化编辑器中的字段允许您配置 YAML 编辑器中显示的 YAML 属性。

第 2 步：通过提交保存工作流程

在此步骤中，您将保存所做的更改。由于工作流程以.yaml文件形式存储在存储库中，因此您可以通过提交来保存更改。

提交您的工作流程更改

1. （可选）选择 “验证” 以确保工作流程的 YAML 代码有效。
2. 选择 Commit (提交)。
3. 在工作流文件名中，输入工作流程配置文件的名称，例如**my-first-workflow**。
4. 在提交消息中，输入一条消息来标识你的提交，比如**create my-first-workflow.yaml**。
5. 在存储库中，选择要在其中保存工作流程的存储库 (codecatalyst-repository)。
6. 在分支名称中，选择要在其中保存工作流程的分支 (main)。
7. 选择 Commit (提交)。

您的新工作流程将显示在工作流程列表中。可能需要几分钟才能出现。

由于工作流程是与提交一起保存的，并且由于工作流程配置了代码推送触发器，因此保存工作流程会自动启动工作流程运行。

步骤 3：查看运行结果

在此步骤中，您将导航到从提交开始运行的运行并查看结果。

查看运行结果

1. 选择工作流程的名称，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. 在工作流程运行图中，选择生成操作（例如，build_F 0）。
3. 查看日志、报告、配置和变量选项卡的内容。这些选项卡显示生成操作的结果。

   有关更多信息，请参阅 查看生成操作的结果。

（可选）步骤 4：清理

在此步骤中，您将清理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。删除项目也会删除源存储库和工作流程。

使用工作流程进行构建

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？
- 生成操作的好处
- 构建操作的替代方案
- 添加生成操作
- 查看生成操作的结果

- 教程：将构件上传到 Amazon S3
- 生成和测试操作 YAML 定义

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。

构建操作是一个工作流程构建块，用于编译源代码、运行单元测试并生成准备部署的工件。

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高级步骤如下。

生成应用程序（高级任务）

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。在工作流程中，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 工作流程入门。
3. （可选）在工作流程中，您可以添加一个触发器，该触发器指示将导致工作流程自动启动的事件。有关更多信息，请参阅使用触发器自动启动工作流程
4. 在工作流程中，您可以添加编译和打包应用程序或资源源代码的生成操作。或者，如果您不想将测试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。有关测试和部署操作的更多信息，请参阅添加生成操作。
5. （可选）在工作流程中，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 使用工作流程进行测试 和 使用工作流程进行部署。
6. 您可以手动启动工作流程，也可以通过触发器自动启动工作流程。该工作流按顺序运行构建、测试和部署操作，以构建、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 启动工作流程手动运行。

生成操作的好处

在工作流程中使用生成操作具有以下好处：

- 完全托管 — 构建操作无需设置、修补、更新和管理自己的生成服务器。

- 按需 — 构建操作可按需扩展，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 为工作流程配置计算和运行时环境 Docker 镜像。
- 开箱即用 — CodeCatalyst 包括预打包的运行时环境 Docker 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 Docker 镜像。

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？ 。

添加生成操作

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生成操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。
5. 选择 “视觉”。
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，请参阅生成和测试操作 YAML 定义。本参考提供了在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
9. （可选）选择 “验证” 以在提交之前验证工作流程的 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 编辑器添加生成操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。
5. 选择 YAML。
6. 选择操作。
7. 在操作中，选择构建。
8. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明生成和测试操作 YAML 定义。
9. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
10. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

生成操作定义

生成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，请参见生成和测试操作 YAML 定义中的工作流程 YAML 定义。

查看生成操作的结果

按照以下说明查看生成操作的结果，包括生成的日志、报告和变量。

查看生成操作的结果

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
3. 在工作流程图中，选择生成操作的名称，例如 “构建”。
4. 要查看生成运行的日志，请选择日志。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. 要查看生成操作生成的测试报告，请选择 Reports，或者在导航窗格中选择 Report s。有关更多信息，请参阅 质量报告类型。

6. 要查看用于生成操作的配置，请选择配置。有关更多信息，请参阅 添加生成操作。
7. 要查看生成操作使用的变量，请选择变量。有关更多信息，请参阅 在工作流程中配置和使用变量。

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 CodeCatalyst 的工作流程将项目上传到 Amazon S3 存储桶。当工作流程启动时，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生成构件中。第二个构建操作会将构件上传到 Amazon S3。您需要将工作流程配置为在每次将提交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。
- 在你的空间中，你需要一个空的、从头开始的 CodeCatalyst项目，名为：

  ```
  codecatalyst-artifact-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-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。

有关更多信息，请参阅 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

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**
>
> 如果您已经有为其他教程创建的构建角色，则也可以将其用于本教程。只要确保它具有以下步骤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。

创建生成角色

1. 为该角色创建策略，如下所示：
    a. 登录到 AWS。
    b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。
    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：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

```
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

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. 在 “角色名称” 中，输入：

```
codecatalyst-s3-build-role
```

i. 在角色描述中，输入：

```
CodeCatalyst build role
```

j. 选择 创建角色。

现在，您已经创建了一个包含信任策略和权限策略的构建角色。

第 2 步：创建 Amazon S3 存储桶

在此步骤中，您将创建一个 Amazon S3 存储桶，用于上传Hello.txt和Goodbye.txt项目。

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. 在存储桶名称中，输入：

```
codecatalyst-artifact-bucket
```

4. 对于 AWS 区域（亚马逊云科技区域），选择一个区域。本教程假设您选择了美国西部（俄勒冈）us-west-2。有关 Amazon S3 支持的区域的信息，请参阅中的亚马逊简单存储服务终端节点和配额AWS 一般参考。
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 区域创建了一个名**codecatalyst-artifact-bucket**为的存储桶。

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。此存储库用于存储教程的工作流程定义文件。

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的存储库codecatalyst-artifact-source-repository。

步骤 4：创建工作流程

在此步骤中，您将创建一个由以下按顺序运行的构建块组成的工作流程：

- 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程运行。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程。

- 名为 “GenerateFiles-触发时GenerateFiles” 的生成操作会创建两个文件Hello.txt和Goodbye.txt，并将它们打包到名为的输出构件中codecatalystArtifact。
- 另一个名为的构建操作 Upload — GenerateFiles 操作完成后，该Upload操作将运行 AWS CLI 命令，将源存储库codecatalystArtifact和中的文件上传aws s3 sync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，因此您无需对其进行安装或配置。

  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 Docker 镜像。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。

有关生成操作的更多信息，请参阅使用工作流程进行构建。

创建工作流

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择 “创建工作流程”。
3. 删除 YAML 示例代码。
4. 添加以下 YAML 代码：

```
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
```

在上面的代码中，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。
- *codetalyst-s3-build-role ############*的名称。步骤 1：创建 AWS 角色
- *codecatalyst-artifact-bucket*使用您在中创建的 Amazon S3 的名称第 2 步：创建 Amazon S3 存储桶。

有关此文件中属性的信息，请参阅生成和测试操作 YAML 定义。

5. （可选）选择验证以确保 YAML 代码在提交之前有效。
6. 选择 Commit (提交)。
7. 在“提交工作流程”对话框中，输入以下内容：

   a. 对于工作流程文件名，保留默认值codecatalyst-artifact-workflow。

   b. 在“提交消息”中，输入：

   ```
   add initial workflow file
   ```

   c. 对于“存储库”，选择codecatalyst-artifact-source-repository。

   d. 在“分支名称”中，选择“主分支”。

   e. 选择 Commit (提交)。

现在，您已经创建了一个工作流程。由于在工作流程顶部定义了触发器，因此工作流程运行会自动启动。具体而言，当您将codecatalyst-artifact-workflow.yaml文件提交（并推送）到源存储库时，触发器会启动工作流程运行。

查看正在运行的工作流程

1. 在导航窗格中，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程: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. “上载” 操作完成后，请执行以下操作：
   - 如果工作流程运行成功，请转到下一个过程。
   - 如果工作流程运行失败，请选择 Logs 来解决问题。

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，它应该包含以下文件和文件夹：

```
.
|— .aws/
|— .git/
|Goodbye.txt
|Hello.txt
|REAME.md
```

Goodbye.txt和Hello.txt文件之所以被上传，是因为它们是codecatalystArtifact构件的一部分。.aws/.git/、和README.md文件之所以上传，是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. 删除工作codecatalyst-artifact-workflow流程。

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存储桶中的文件。
   c. 删除codecatalyst-artifact-bucket存储桶。
2. 在 IAM 中进行清理，如下所示：
   a. 通过 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。

生成和测试操作 YAML 定义

以下是生成和测试操作的 YAML 定义。两个操作只有一个参考文献，因为它们的 YAML 属性非常相似。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

在以下代码中选择 YAML 属性以查看其描述。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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"
```

操作名称

（必需）

指定操作的名称。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

对应的 UI：“配置” 选项卡/ 操作名称

Identifier

（####/Identifier）

标识操作。除非要更改版本，否则不要更改此属性。有关更多信息，请参阅 指定操作的主版本、次要版本或补丁版本。

aws/build@v1用于生成操作。

aws/managed-test@v1用于测试操作。

############/####/aws/build @v1 |aws/managed-test @v1 ##

DependsOn

（*####*/DependsOn）

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关“依赖”功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入”选项卡/ 依赖- 可选

Compute

（*####*/Compute）

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

（*####/Compute/*类型）

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

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

（ *#### /Compute/* ***Fleet*** ）

（ 可选 ）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，`Fleet`则默认为`Linux.x86-64.Large`。

对应的 UI：“配置” 选项卡/ “计算舰队”

Timeout

（ *####*/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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Environment

（ *####*/Environment ）

（ 可选 ）

指定要用于操作的 CodeCatalyst 环境。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的 UI：“配置” 选项卡/ 环境

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/ 环境名称

Connections

（*#### /Environment/* ***Connections*** ）

（ 可选 ）

指定要与操作关联的账户连接。您最多可以在下指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的 UI：“配置” 选项卡/

Name

（*#### /Environment/Connections/* ***Name*** ）

（ 可选 ）

指定账户连接的名称。

对应的 UI：“配置” 选项卡/

Role

（*#### /Environment/Connections/* ***Role*** ）

（可选）

指定此操作用于访问和操作 Amazon S3 和 Amazon ECR 等 AWS 服务的 IAM 角色的名称。确保将此角色添加到您的账户关联中。要向账户连接添加 IAM 角色，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 只要角色具有足够的权限，您就可以在此处指定该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

> **Warning**
>
> 将权限限制为生成和测试操作所需的权限。使用具有更广泛权限的角色可能会带来安全风险。

对应的 UI：“配置” 选项卡/

Caching

（*####*/Caching）

（可选）

在此部分中，您可以指定缓存来保存磁盘上的文件，并在后续的工作流程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，请参阅在工作流程运行之间缓存文件。

对应的用户界面：配置选项卡/ 文件缓存- 可选

FileCaching

（*#### /Caching/* ***FileCaching***）

（可选）

指定缓存序列配置的部分。

对应的用户界面：配置选项卡/文件缓存-可选/添加缓存

密钥名称-1

（*####/Caching/FileCaching/####-* 1）

（可选）

指定主缓存属性名称的名称。缓存属性名称在您的工作流程中必须是唯一的。每个操作中最多可以有五个条目FileCaching。

相应的用户界面：配置选项卡/文件缓存-可选/添加缓存/密钥

Path

（*####/Caching/FileCaching/##-## -1/#**Path***

（可选）

为您的缓存指定关联路径。

相应的 UI：配置选项卡/文件缓存-可选/添加缓存/ 路径

RestoreKeys

（*####/Caching/FileCaching/##-## -1/#**RestoreKeys***

（可选）

指定在找不到主缓存属性时用作备用还原密钥。恢复密钥名称在您的工作流程中必须是唯一的。每个缓存中最多可以有五个条目RestoreKeys。

相应的用户界面：配置选项卡/文件缓存-可选/添加缓存/还原密钥-可选

Inputs

（*####*/Inputs）

（可选）

该Inputs部分定义了操作在工作流程运行期间所需的数据。

> **Note**
>
> 每个构建操作或测试操作最多允许四个输入（一个源和三个工件）。变量不计入此总数。

如果您需要引用驻留在不同输入（比如源和构件）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，以区分主输入中的文件。有关更多信息，请参阅 示例：引用多个构件中的文件。

相应的 UI：“输入” 选项卡

Sources

（*#### /Inputs/* ***Sources***）

（可选）

指定代表操作所需的源存储库的标签。当前，唯一支持的标签是WorkflowSource，它表示存储工作流程定义文件的源存储库。

如果省略源，则必须在下*action-name*/Inputs/Artifacts方指定至少一个输入对象。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

对应的用户界面：无

Artifacts - input

（*#### /Inputs/* ***Artifacts***）

（可选）

指定要作为此操作输入的先前操作中的对象。在之前的操作中，这些构件必须已定义为输出对象。

如果您未指定任何输入构件，则必须至少在下方指定一个源存储库*action-name*/Inputs/Sources。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 如果 Artifacts -可选下拉列表不可用（可视化编辑器），或者在验证 YAML（YAML 编辑器）时出现错误，则可能是因为该操作仅支持一个输入。在这种情况下，请尝试删除源输入。

相应的 UI：“输入” 选项卡/ 工件- 可选

Variables - input

（*#### /Inputs/* ***Variables***）

（可选）

指定一个名称/值对序列，这些对定义了要提供给操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的 UI：“输入” 选项卡/ “变量”- 可选

Outputs

（*####*/Outputs）

（可选）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts - output

（*#### /Outputs/* ***Artifacts***）

（可选）

指定操作生成的对象的名称。Artifact 名称在工作流程中必须是唯一的，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 (-) 和其他特殊字符。不能使用引号在输出对象名称中启用空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/ 工件

Name

（*#### /Outputs/Artifacts/* ***Name*** ）

（如果包含Artifacts - output，则为必填项）

指定操作生成的对象的名称。Artifact 名称在工作流程中必须是唯一的，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 (-) 和其他特殊字符。不能使用引号在输出对象名称中启用空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/构件/新建输出/构建构件名称

Files

（*#### /Outputs/Artifacts/* ***Files*** ）

（如果包含Artifacts - output，则为必填项）

指定操作输出的对象中 CodeCatalyst 包含的文件。这些文件由工作流程操作在运行时生成，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中的对象中，并且是相对于源存储库或项目根目录的。你可以使用 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，请参见语法指南和惯例。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个工件或来源中找到它。有关更多信息，请参阅 引用源存储库中的文件 和 在构件中引用文件。

相应的用户界面：输出选项卡/构件/新输出/构建生成的文件

Variables - output

（*#### /Outputs/* ***Variables*** ）

（可选）

指定要导出操作的变量，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），请参阅在工作流程中配置和使用变量。

相应的用户界面：“输出” 选项卡/变量/ 添加变量

变量名 1

（*####/Outputs/Variables/####-* 1 ）

（可选）

指定要导出操作的变量的名称。此变量必须已经在同一操作的`Inputs`或`Steps`部分中定义。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的用户界面：“输出” 选项卡/变量/添加变量/名称

AutoDiscoverReports

（*#### /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。

相应的 UI：“输出” 选项卡/报告/自动发现报告

Enabled

（ *#### /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`则默认为`true`。

相应的 UI：“输出” 选项卡/报告/自动发现报告

ReportNamePrefix

（ *#### /Outputs/AutoDiscoverReports/* ***ReportNamePrefix*** ）

（如果包含并启用AutoDiscoverReports，则为必填项）

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” 选项卡/报告/前缀名称

IncludePaths

（ *#### /Outputs/AutoDiscoverReports/* ***IncludePaths*** ）

Or

（*####/Outputs/Reports/#### -1/#**IncludePaths***

（如果AutoDiscoverReports包含并启用，或者包含在内，Reports则为必填项）

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，请参见语法指南和惯例。

如果省略此属性，则默认值为"**/*"，这意味着搜索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。

相应的用户界面：

- 输出选项卡/报告/自动发现报告/包含/排除路径/包含路径
- ***#####/##/######/####-1 /##/####/####***

ExcludePaths

（*#### /Outputs/AutoDiscoverReports/ **ExcludePaths***）

Or

（*####/Outputs/Reports/#### -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，请参见语法指南和惯例。

相应的用户界面：

- 输出选项卡/报告/自动发现报告/包含/排除路径/排除路径
- ***#####/##/######/####-1 /##/####/####***

SuccessCriteria

(*#### /Outputs/AutoDiscoverReports/* ***SuccessCriteria***)

Or

(*####/Outputs/Reports/#### -1/#****SuccessCriteria***

(可选)

为测试、代码覆盖率、软件组合分析 (SCA) 和静态分析 (SA) 报告指定成功标准。

有关更多信息，请参阅 为报告配置成功标准。

相应的 UI：“输出” 选项卡/报告/成功标准

PassRate

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***PassRate***)

Or

(*####/Outputs/Reports/##/SuccessCriteria/****PassRate****##-1*)

(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。

相应的 UI：“输出” 选项卡/报告/成功标准/通过率

LineCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***LineCoverage*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****LineCoverage****##-1* ）

（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。线路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。

相应的用户界面：输出选项卡/报告/成功标准/行覆盖率

BranchCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***BranchCoverage*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****BranchCoverage****##-1* ）

（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。

相应的用户界面：输出选项卡/报告/成功标准/分支覆盖率

Vulnerabilities

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***Vulnerabilities*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****Vulnerabilities****##-1* ）

（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，必须指定：

- 要计入的漏洞的最低严重性。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则将对CRITICAL漏洞HIGH进行统计。
- 您希望允许的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，请参阅软件成分分析报告。

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” 选项卡/报告/成功标准/漏洞

StaticAnalysisBug

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***StaticAnalysisBug*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****StaticAnalysisBug****##-1* ）

（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，必须指定：

- 您要计入的错误的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则会HIGH对CRITICAL错误进行统计。
- 您希望允许的指定严重程度的错误的最大数量。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。

相应的 UI：“输出” 选项卡/报告/成功标准/错误

StaticAnalysisSecurity

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisSecurity***）

Or

（*####/Outputs/Reports/##/SuccessCriteria/**StaticAnalysisSecurity**##-1*）

（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，必须指定：

- 您要计入的安全漏洞的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则将对HIGHCRITICAL安全漏洞进行统计。
- 您希望允许的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” 选项卡/报告/成功标准/安全漏洞

StaticAnalysisQuality

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisQuality***）

Or

（*####/Outputs/Reports/##/SuccessCriteria/**StaticAnalysisQuality**##-1*）

（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，必须指定：

- 您要计入的质量问题的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择 HIGHHIGH，则将统计CRITICAL质量问题。
- 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 UI：“输出” 选项卡/报告/成功标准/质量问题

StaticAnalysisFinding

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisFinding***）

Or

（*####/Outputs/Reports/##/SuccessCriteria/**StaticAnalysisFinding**##-1*）

（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。要指定调查结果，必须指定：

- 要计入计数的发现的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则将HIGH对CRITICAL结果进行统计。
- 您希望允许的指定严重性的最大发现次数。超过此数字会导致 CodeCatalyst 报告被标记为失败。有效值为整数。

调查结果仅适用于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” 选项卡/报告/成功标准/调查结果

Reports

（*#### /Outputs/* ***Reports*** ）

（可选）

指定测试报告配置的部分。

相应的 UI：“输出” 选项卡/ “报告”

报告名称 1

（*####/Outputs/Reports/####-1* ）

（如果包含Reports，则为必填项）

您要为将从原始 CodeCatalyst 报告生成的报告命名。

相应的 UI：“输出” 选项卡/报告/手动配置报告/报告名称

Format

（*####/Outputs/Reports/#### -1/#****Format***

（如果包含Reports，则为必填项）

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，请指定黄瓜（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 nUni t（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，指定 Visual Studio TRX（可视化编辑器）或VISUALSTUDIOTRX（YAML 编辑器）。
  - 对于 TestNG XML，请指定 te stNG（可视化编辑器）或TESTNGXML（YAML 编辑器）。
- 有关代码覆盖率报告：

- 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）。

UI#“##” ###/##/######/##/####/####-1 /#########

Configuration

（*####*/Configuration）

（必填）可以在其中定义操作配置属性的部分。

对应的 UI：“配置” 选项卡

Container

（*#### /Configuration/* ***Container***）

（可选）

指定操作用于完成其处理的 Docker 镜像或容器。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，则该映像可以存储在 Amazon ECR、Docker Hub 或其他注册表中。如果您未指定 Docker 镜像，则该操作将使用其中一个活动镜像进行处理。有关默认使用哪张活动图像的信息，请参见活跃的图像。

有关指定自己的 Docker 镜像的更多信息，请参阅为操作分配自定义运行时环境 Docker 镜像。

相应的 UI：运行时环境 Docker 镜像- 可选

Registry

（*#### /Configuration/Container/* ***Registry***）

（如果包含Container，则为必填项）

指定存储图像的注册表。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）

  镜像存储在 Docker Hub 镜像注册表中。

- 其他注册表（可视化编辑器）或Other（YAML 编辑器）

  图像存储在自定义图像注册表中。可以使用任何公开可用的注册表。

- Amazon 弹性容器注册表（可视化编辑器）或ECR（YAML 编辑器）

  该图像存储在 Amazon 弹性容器注册表镜像存储库中。要使用 Amazon ECR 存储库中的图像，此操作需要访问亚马逊 ECR。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：

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

```
}
```

有关创建 IAM 角色的更多信息，请参阅 I A M 用户指南中的使用自定义信任策略（控制台）创建角色。

创建角色后，必须通过环境将其分配给操作。有关更多信息，请参阅 将环境、账户连接和 IAM 角色与工作流程操作关联。

相应的用户界面：亚马逊弹性容器注册表、Docker Hub 和其他注册表选项

Image

（*#### /Configuration/Container/* ***Image***）

（如果包含Container，则为必填项）

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，请将镜像设置为 Docker Hub 镜像名称和可选标签。

  例如：`postgres:latest`
- 如果您使用的是亚马逊 ECR 注册表，请将映像设置为亚马逊 ECR 注册表 URI。

  例如：`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。

相应的用户界面：运行时环境 docker 镜像（如果注册表是**CODECATALYST**）、Docker Hub 镜像（如果注册表是 Docker Hub）、ECR 镜像 URL（如果注册表是亚马逊弹性容器注册表）和镜像 URL（如果注册表是其他注册表）。

Steps

（*#### /Configuration/* ***Steps***）

（必需）

指定要在安装、配置和运行生成工具的操作期间运行的 shell 命令。

以下是如何构建 npm 项目的示例：

```
Steps:
  - Run: npm install
  - Run: npm run build
```

以下是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，请参见引用源存储库中的文件和在构件中引用文件。

相应的 UI：“配置” 选项卡/ Shell 命令

Packages

（*#### /Configuration/* ***Packages***）

（可选）

在此部分中，您可以指定操作用于解析依赖关系的包存储库。软件包允许您安全地存储和共享用于应用程序开发的软件包。

有关软件包的更多信息，请参阅在中发布和共享软件包 CodeCatalyst。

相应的 UI：“配置” 选项卡/ “包”

NpmConfiguration

（*#### /Configuration/Packages/* ***NpmConfiguration***）

（如果包含Packages，则为必填项）

定义 npm 包格式配置的部分。此配置由工作流程运行期间的操作使用。

有关 npm 软件包配置的更多信息，请参阅使用 npm。

对应的用户界面：配置选项卡/包/添加配置/npm

PackageRegistries

（*#### /Configuration/Packages/NpmConfiguration/* ***PackageRegistries***）

（如果包含Packages，则为必填项）

您可以在其中定义一系列软件包存储库的配置属性的部分。

相应的 UI：“配置” 选项卡/包/添加配置/npm/ 添加软件包存储库

PackagesRepository

（*#### /Configuration/Packages/NpmConfiguration/PackageRegistries/* ***PackagesRepository***）

（如果包含Packages，则为必填项）

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，则最后一个存储库将优先。

有关软件包存储库的更多信息，请参阅Package 存储库。

对应的 UI：“配置” 选项卡/Packages/Add Configuration/NPM/Add 软件包存储库/Package

Scopes

（*#### /Configuration/Packages/NpmConfiguration/PackageRegistries/* ***Scopes***）

（可选）

指定要在软件包注册表中定义的范围序列。定义作用域时，将指定的包存储库配置为所有列出的作用域的注册表。如果通过 npm 客户端请求具有作用域的软件包，它将使用该存储库而不是默认存储库。每个作用域名称必须以 “@” 为前缀。

如果您包含覆盖作用域，则最后一个存储库将优先。

如果Scopes省略，则将指定的包存储库配置为该操作使用的所有包的默认注册表。

有关作用域的更多信息，请参阅Package 命名空间和作用域包。

相应的用户界面：配置选项卡/包/添加配置/NPM/添加软件包存储库/作用域-可选

使用工作流程进行测试

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。这些工作流程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程操作。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
- 在操作中跳过失败的测试
- 集成 universal-test-runner 到测试操作中
- 在操作中配置质量报告
- 重试报告的测试用例
- 在中进行测试的最佳实践 CodeCatalyst
- 软件组成分析和静态分析报告中支持 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
- 软件成分分析报告
- 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。如果您有来自多个版本的许多测试报告，则可以使用测试报告来查看失败率，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

衡量您的测试涵盖了多少陈述。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

衡量您的测试涵盖了控制结构（例如`if`或`case`语句）中每个可能的分支中有多少个分支。

```
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）
- 三叶草 XML（版本 3，.xml）
- XML 覆盖范围 (.xml)
- LCOV (.info)

软件成分分析报告

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。您可以发现和解析 SARIF 报告，这些报告详细说明了不同严重程度的漏洞及其修复方法。从最严重到最不严重的有效严重性值为：CRITICAL、HIGH、MEDIUM、LOW、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、质量问题和其他漏洞。从最严重到最不严重的有效严重性值为：CRITICALHIGH、MEDIUM、LOW、和INFORMATIONAL。

CodeCatalyst 提供了以下 SA 指标：

错误

识别源代码中发现的许多可能的错误。这些错误可能包括与内存安全有关的问题。以下是错误的示例。

```
// 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

识别源代码中发现的许多可能的安全漏洞。这些安全漏洞可能包括以明文形式存储您的秘密令牌之类的问题。

质量问题

识别源代码中发现的许多可能的质量问题。这些质量问题可能包括与风格惯例有关的问题。以下是质量问题的示例。

```
// The function name doesn't adhere to the style convention of camelCase
int SUBTRACT(int x, int y) {
  return x-y
}
```

其他漏洞

识别源代码中可能存在的许多其他漏洞。

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 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 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。

5. 选择 “视觉”。
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，请参阅生成和测试操作 YAML 定义。本参考提供了在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
9. （可选）选择 “验证” 以在提交之前验证工作流程的 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 编辑器添加生成操作

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。
5. 选择 YAML。
6. 选择操作。
7. 在操作中，选择测试。
8. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明生成和测试操作 YAML 定义。
9. （可选）选择 “验证” 以在提交之前验证工作流程的 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，请参见生成和测试操作 YAML 定义中的工作流程 YAML 定义。

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 C I/CD，然后选择工作流程。
2. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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 Reports，或者在导航窗格中选择 Report s。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 在工作流程中配置和使用变量。

在操作中跳过失败的测试

如果您的操作有多个测试命令，则即使之前的命令失败，您也可能希望允许操作中的后续测试命令运行。例如，在以下命令中，即使`test1`失败，您也可能希望`test2`始终运行。

```
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。您可以通过将错误输出重定向到，允许操作步骤继续运行。`null`你可以通过在命令中添加`2>/dev/null`来做到这一点。经过此修改，前面的示例将如下所示。

```
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`命令的状态将得到尊重，但`npm run test1`命令返回的任何错误都将被忽略。因此，该`npm run test2`命令已运行。通过这样做，无论是否出现错误，您都可以同时查看两个报告。

集成 universal-test-runner 到测试操作中

测试操作与开源命令行工具集成universal-test-runner。此工具提供高级测试功能，例如重试测试报告中的一个或多个测试用例。 universal-test-runner使用测试执行协议对给定框架中的任何语言运行测试。 universal-test-runner支持以下框架：

- Gradle
- 开玩笑
- Maven
- pytest
- .NET

universal-test-runner仅安装在用于测试操作的精选图像上。如果您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npm使用 shell 命令universal-test-runner进行安装- Run: npm install -g @aws/universal-test-runner。有关通过 shell 命令在容器中安装 Node.js 的更多信息，请参阅安装和更新节点版本管理器。

有关的更多信息universal-test-runner，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 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。
4. 选择编辑。
5. 选择 “视觉”。
6. 选择操作。
7. 在操作中，选择测试。
8. 在 “配置” 选项卡上，使用您选择的支持的框架更新示例代码，从而完成 “命令行管理程序命令” 字段。例如，要使用支持的框架，您可以使用类似于以下内容的Run命令。

```
- Run: run-tests <framework>
```

如果不支持你想要的框架，可以考虑贡献一个自定义的适配器或运行器。有关命令行管理程序命令字段的说明，请参见Steps。

9. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。
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，您可以使用类似于以下内容的Run命令。

```
Configuration:
  Steps:
    - Run: run-tests <framework>
```

如果不支持你想要的框架，可以考虑贡献一个自定义的适配器或运行器。有关 Steps 属性的描述，请参阅Steps。

9. （可选）选择 “验证” 以在提交之前验证工作流程的 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

- 为报告配置成功标准
- 质量报告 YAML 示例

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。

您也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以及文件格式。有关更多信息，请参阅质量报告类型。

为报告配置成功标准

您可以设置用于确定测试、代码覆盖率、软件组合分析 (SCA) 或静态分析 (SA) 报告的成功标准的值。

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，它会显示成功标准是否得到满足，以及在多大程度上得到满足。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为：CRITICAL、、HIGH、MEDIUMLOW、INFORMATIONAL。如果您将标准设置为扫描一个HIGH严重程度的漏洞，则如果至少存在一个严重漏洞或没有HIGH严重性级别的漏洞，但至少存在一个HIGH严重级别较高的漏洞（例如一个严重漏洞），则报告将失败。CRITICAL

如果您未指定成功标准，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关联的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 C I/CD，然后选择工作流程。
2. 选择包含生成报告的操作的工作流程。这是您要为其应用成功标准的报告。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。

3. 选择编辑。
4. 选择 “视觉”。
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：

   及格率

   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。

   线路覆盖范围

   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。线路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。

   分支机构覆盖范围

   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，必须指定：

   - 要计入的漏洞的最低严重性。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

     例如，如果您选择HIGH，则将对CRITICAL漏洞HIGH进行统计。
   - 您希望允许的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，请参阅软件成分分析报告。

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，必须指定：

- 要计入的错误的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则会HIGH对CRITICAL错误进行统计。
- 您希望允许的指定严重程度的错误的最大数量。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，必须指定：

- 您要计入的安全漏洞的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则将对HIGHCRITICAL安全漏洞进行统计。
- 您希望允许的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，必须指定：

- 要计入的质量问题的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

例如，如果您选择 HIGHHIGH，则将统计CRITICAL质量问题。

- 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 Commit (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 启动工作流程手动运行。

YAML

配置成功标准

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择包含生成报告的操作的工作流程。这是您要为其应用成功标准的报告。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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，请参阅生成和测试操作 YAML 定义。
8. 选择 Commit (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。有关启动工作流程的更多信息，请参阅启动工作流程手动运行。

质量报告 YAML 示例

以下示例说明如何手动配置四个报告：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
```

重试报告的测试用例

如果您的报告因多个测试用例而失败，则只能重试这些单独的测试。这使您可以快速检查测试用例的质量，并确定解决问题的后续步骤，例如使用中断的依赖关系或启动工作流程重新运行。您的测试操作包含仅`universal-test-runner`重试选定的测试用例，而不是整个操作。每次只能在每个操作中重试一组选定的测试用例，并且每份测试报告只能重试五次。有关更多信息，请参阅 集成 universal-test-runner 到测试操作中。

> **Note**
>
> 如果您在报告中重试测试用例，则不会影响生成原始报告的工作流程的状态。

按照以下说明重试报告中的测试用例。

重试报告的测试用例

1. 在导航窗格中，选择 Reports。
2. 选择您的报告的名称。您可以按名称、状态、存储库、分支或报告类型进行筛选。
3. 在报告名称下方，选择结果。
4. 选择要重试的测试用例，选择 “重新运行”，然后选择 “选定的测试用例”。
5. 重试完成后，选择横幅上的 “刷新” 并查看更新的结果。

在中进行测试的最佳实践 CodeCatalyst

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
- 成功标准
- 包含/排除路径

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。自动发现有助于确保即使发现的输出的名称或路径发生变化，仍

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。但是，如果您使用自动发现，则必须考虑此功能的以下某些方面：

- 当您在操作中激活自动发现时，所有自动发现的相同类型的报告都将共享相同的成功标准。例如，诸如最低通过率之类的共享标准将适用于所有自动发现的测试报告。如果您需要为相同类型的报告设置不同的标准，则必须明确配置这些报告中的每一个报告。
- 自动发现还可以查找由您的依赖项生成的报告，如果配置了成功标准，则可能无法对这些报告执行操作。此问题可以通过更新排除路径配置来解决。
- 不能保证自动发现每次都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的报告，则应明确配置报告。例如，如果测试作为构建的一部分停止运行，则测试框架将不会生成任何输出，因此不会生成任何测试报告，操作可能会成功。如果您希望操作的成功取决于该特定测试，则必须明确配置该报告。

> **Tip**
>
> 在开始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。这会调用项目中所有文件（包括子目录中的文件）生成报告。

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强制执行质量阈值。例如，如果自动发现了两个代码覆盖率报告，一个的行覆盖率为 80%，另一个的行覆盖率为 60%，则有以下选项：

- 将线路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。要解除对工作流程的封锁，请在项目中添加新的测试，直到第二份报告的行覆盖率超过 80%。
- 将线路覆盖率的自动发现成功标准设置为 60%。这将导致两个报告都通过，从而导致操作成功。然后，你可以在第二份报告中着手增加代码覆盖率。但是，使用这种方法，您不能保证第一份报告中的覆盖率不会降至80%以下。
- 使用可视化编辑器或为每个报告添加明确的 YAML 部分和路径，显式配置其中一个或两个报告。这将允许您为每个报告配置单独的成功标准和自定义名称。但是，使用这种方法，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 为报告配置成功标准。

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。要忽略目录中的所有文件，请使用 `**/*` glob 模式。

以下是可能的 glob 模式的示例。

模式	描述
`*.*`	匹配当前目录中所有包含点的对象名称
`*.xml`	匹配当前目录中以结尾的所有对象名称 `.xml`
`*.{xml,txt}`	匹配当前目录中以`.xml`或结尾的所有对象名称 `.txt`
`**/*.xml`	匹配所有以结尾的目录中的对象名称 `.xml`
`testFolder`	匹配名为的对象`testFolder` ，将其视为文件
`testFolder/*`	匹配子文件夹中一个级别中的对象`testFolder` ，例如 `testFolder/file.xml`
`testFolder/*/*`	匹配子文件夹中两个级别的对象`testFolder` ，例如 `testFolder/reportsFolder/file.xml`

模式	描述
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。如果两者都IncludePathsExcludePaths包含同一个文件夹，则不会扫描该文件夹以获取报告。

软件组成分析和静态分析报告中支持 SARIF 属性

SARIF（静态分析结果交换格式）是一种输出文件格式，可在中的软件组成分析和静态分析报告中CodeCatalyst使用。以下示例说明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
- run 对象
- toolComponent 对象
- reportingDescriptor 对象
- result 对象
- location 对象
- physicalLocation 对象
- logicalLocation 对象
- fix 对象

`sarifLog` 对象

名称	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的数组，每个运行表示分析工具的单次运行。

`run` 对象

名称	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。

名称	必需	描述
results[]	是	上显示的分析工具的结果 CodeCatalyst。

toolComponent 对象

名称	必需	描述
name	是	分析工具的名称。
properties.artifactScanned	否	该工具分析的工件总数。
rules[]	是	代表规则的reporting Descriptor 对象数组。根据这些规则，分析工具会在所分析的代码中发现问题。

reportingDescriptor 对象

名称	必需	描述
id	是	用于引用调查结果的规则的唯一标识符。 最大长度：1,024 个字符
name	否	规则的显示名称。 最大长度：1,024 个字符
shortDescription.text	否	对规则的简短描述。 最大长度：3,000 个字符
fullDescription.text	否	规则的完整描述。

名称	必需	描述
		最大长度：3,000 个字符
helpUri	否	可以本地化为包含规则主要文档的绝对 URI 的字符串。 最大长度：3,000 个字符
properties.unscore	否	一个标志，用于指示扫描结果是否已被评分。
properties.score.severity	否	一组固定的字符串，用于指定调查结果的严重性级别。 最大长度：1,024 个字符
properties.cvssv3_baseSeverity	否	通用漏洞评分系统 v3.1 的定性严重性评级。
properties.cvssv3_baseScore	否	CVSS v3 基础分数介于 0.0-10.0 之间。
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 基础分数介于 0.0-10.0 之间。
properties.severity	否	一组固定的字符串，用于指定调查结果的严重性级别。 最大长度：1,024 个字符
defaultConfiguration.level	否	规则的默认严重性。

`result` 对象

名称	必需	描述
`ruleId`	是	用于引用调查结果的规则的唯一标识符。 最大长度：1,024 个字符
`ruleIndex`	是	工具组件中关联规则的索引`rules[]`。
`message.text`	是	一条消息，用于描述结果并显示每个发现的消息。 最大长度：3,000 个字符
`rank`	否	介于 0.0 到 100.0（含）之间的值，表示结果的优先级或重要性。此比例值为 0.0 表示最低优先级，100.0 表示最高优先级。
`level`	否	结果的严重程度。 最大长度：1,024 个字符
`properties.unscore`	否	一个标志，用于指示扫描结果是否已被评分。
`properties.score.severity`	否	一组固定的字符串，用于指定调查结果的严重性级别。 最大长度：1,024 个字符
`properties.cvssv3_baseSeverity`	否	通用漏洞评分系统 v3.1 的定性严重性评级。
`properties.cvssv3_baseScore`	否	CVSS v3 基础分数介于 0.0-10.0 之间。

名称	必需	描述
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 基础分数介于 0.0-10.0 之间。
properties.severity	否	一组固定的字符串，用于指定调查结果的严重性级别。 最大长度：1,024 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。 最大location对象数：10
relatedLocations[]	否	调查结果中参考的其他地点列表。 最大location对象数：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

名称	必需	描述
physicalLocation	是	标识构件和区域。

名称	必需	描述
logicalLocations[]	否	按名称描述的一组位置，不提及工件。

physicalLocation 对象

名称	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	指示文件位置的后备 URI。如果artifactLocation.uri 返回为空，则使用此选项。
region.startLine	是	该区域中第一个字符的行号。
region.startColumn	是	该区域中第一个字符的列号。
region.endLine	是	该区域最后一个字符的行号。
region.endColumn	是	该区域最后一个字符的列号。

logicalLocation 对象

名称	必需	描述
fullyQualifiedName	否	描述结果位置的其他信息。 最大长度：1,024 个字符

`fix` 对象

名称	必需	描述
`description.text`	否	显示每项发现的建议的消息。 最大长度：3,000 个字符
`artifactChanges.[0].artifactLocation.uri`	否	表示需要更新的构件位置的 URI。

使用工作流程进行部署

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，使用蓝/绿方案）。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高级步骤如下。

部署应用程序（高级任务）

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，您可以创建工作流程。在工作流程中，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 工作流程入门。
3. 在工作流程中，您可以添加触发器、生成操作以及测试操作（可选）。有关更多信息，请参阅 使用触发器自动启动工作流程、添加生成操作 和 添加测试操作。
4. 在工作流程中，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。）

5. 您可以手动启动工作流程，也可以通过触发器自动启动工作流程。该工作流按顺序运行构建、测试和部署操作，将您的应用程序和资源部署到目标。有关更多信息，请参阅 启动工作流程手动运行。

部署操作列表

以下部署操作可用：

- 部署 AWS CloudFormation 堆栈-此操作 AWS 基于您提供的AWS 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 使用工作流程部署 AWS CloudFormation 堆栈。
- 部署到 Amazon ECS — 此操作会注册您提供的任务定义文件。有关更多信息，请参阅 使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)。
- 部署到 Kubernetes 集群 — 此操作将应用程序部署到亚马逊 Elastic Kubernetes Service 集群。有关更多信息，请参阅 使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service。
- 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 Cloud Development Kit (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。要了解有关环境页面和查看部署的更多信息，请参阅部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和查看部署状态、提交和拉取请求。

部署操作的好处

在工作流程中使用部署操作具有以下好处：

- 部署历史记录-查看部署历史记录，以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-观察部署在工作流程的各个阶段的进展。
- 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

您不必使用部署操作，但建议使用部署操作，因为它们具有上一节中概述的优点。相反，您可以使用以下CodeCatalyst 操作：

- 生成操作。

  通常，如果要部署到不存在相应部署操作的目标，或者想要对部署过程进行更多控制，则可以使用生成操作。有关使用生成操作部署资源的更多信息，请参阅使用工作流程进行构建。
- 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 将 GitHub 操作集成到工作流程中

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？
- AWS CloudFormation — 看看什么是 AWS CloudFormation？

使用 CodeDeploy、 CodeBuild CodePipeline、和 CloudFormation 服务进行复杂的企业部署。

主题

- 使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)
- 使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service
- 使用工作流程部署 AWS CloudFormation 堆栈
- 使用工作流程部署 AWS Cloud Development Kit (AWS CDK) 应用程序
- 使用工作流程引导 AWS CDK 应用程序
- 使用工作流程将文件发布到 Amazon S3
- 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC
- 在工作流程图中显示已部署应用程序的 URL
- 移除部署目标

- 通过提交跟踪部署状态
- 查看部署日志
- 查看部署状态、提交和拉取请求

使用工作流程将应用程序部署到 Amazon 弹性容器服务 (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 ECS 集群中。 CodeCatalyst 为此，您必须将 “部署到 Amazon ECS” 操作添加到您的工作流程中。此操作会注册您提供的任务定义文件。注册后，任务定义将由在您的 Amazon ECS 集群中运行的 Amazon EC S 服务实例化。“实例化任务定义” 等同于将应用程序部署到 Amazon ECS 中。

要使用此操作，您必须准备好 Amazon ECS 集群、服务和任务定义文件。

有关 Amazon ECS 的更多信息，请参阅亚马逊弹性容器服务开发人员指南。

> **Tip**
>
> 有关向您展示如何使用 “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**
>
> 要查看部署到 Amazon ECS 操作的有效示例，请使用带蓝图的 Node.js API AWS Fargate或带 AWS Fargate蓝图的 Java API 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

- 教程：将应用程序部署到 Amazon ECS
- 添加 “部署到 Amazon ECS” 操作
- “部署到 Amazon ECS” 操作生成的变量
- “部署到 Amazon ECS” 操作 YAML 定义

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 镜像构建的简单的 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程。

> **Tip**
>
> 您可以使用蓝图为您完成完整的 Amazon ECS 设置，而不必仔细阅读本教程。你需要将 Node.js API 与蓝图一起使用， AWS Fargate或者在 AWS Fargate蓝图中使用 Java API。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。
- 在你的空间中，你需要一个空的、从头开始的 CodeCatalyst项目，名为：

  ```
  codecatalyst-ecs-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-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。

有关更多信息，请参阅 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

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。完成教程后，请删除此用户。

> **Note**
>
> 在本教程中，请勿使用 root 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 身份中心用户

1. 登录 AWS Management Console 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 账户选项卡来验证连接了哪个账户。有关更多信息，请参阅 创建空间。

2. 在导航窗格中，选择 Users，然后选择 Add user。
3. 在用户名中，输入：

   ```
   CodeCatalystECSUser
   ```

4. 在 “密码” 下，选择 “生成可与该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 中尚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 “将用户添加到群组” 页面上，选择 “下一步”。
10. 在查看并添加用户页面上，查看信息并选择添加用户。

    将出现 “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CodeCatalystECSUser稍后会将此权限集分配给。

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。此策略向所有人提供完全权限 AWS 服务。
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。

要以 ecsuSer 的身份 CodeCatalyst注销并重新登录

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。您应该早点将此信息复制到文本编辑器中。

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

2. 退出 AWS。
3. 将 AWS 访问门户 URL 粘贴到浏览器的地址栏中。
4. 使用的用户名和一次性密码登录CodeCatalystECSUser。
5. 在新密码中，输入密码，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 Management Console 出现了。现在，您已CodeCatalystECSUser使用相应的权限登录。

启动实 AWS CloudShell 例

1. 在顶部导航栏中，选择 AWS 图标
   ()。Co
   的主页 AWS Management Console 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受支持 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 CloudShell 终端中，输入：

```
aws --version
```

2. 检查是否出现了版本。

   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

在本节中，您将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被您的工作流程部署的 Hello World 应用程序所取代。占位符应用程序是 Apache Web 服务器。

有关 Amazon ECS 的更多信息，请参阅亚马逊弹性容器服务开发人员指南。

完成以下一系列步骤来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 (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. 在提示符下输入以下代码：

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

   c. 将光标放在最后一个大括号 (}) 之后。

   d. 按下**Enter**然后**Ctrl+d**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. 请注意角色的"Arn":值，例如arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role。稍后您将需要此亚马逊资源名称 (ARN)。

要创建 Amazon ECS 集群

该集群将包含 Apache 占位符应用程序，稍后还包含 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）验证集群是否已成功创建：

```
aws ecs list-clusters
```

codecatalyst-ecs-cluster集群的 ARN 应出现在列表中，表示已成功创建。

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/-* role codecatalyst-ecs-task-execution

使用您在中记下的任务执行角色的 ARN。创建任务执行角色

3. 将光标放在最后一个大括号 (}) 之后。
4. 按下**Enter**然后**Ctrl+d**保存文件并退出 cat。

向 Amazon ECS 注册任务定义文件

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）验证任务定义是否已注册：

```
aws ecs list-task-definitions
```

codecatalyst-ecs-task-def任务定义应出现在列表中。

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. 同样CodeCatalystECSUser，如果您尚未切换到 Amazon 弹性容器服务控制台，请切换到该控制台。
2. 选择您之前创建的集群codecatalyst-ecs-cluster。
3. 在 “服务” 选项卡中，选择 “创建”。
4. 在 “创建” 页面中，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 “家庭” 下拉列表中，选择：

      codecatalyst-ecs-task-def

   d. 在服务名称中，输入：

      ```
      codecatalyst-ecs-service
      ```

   e. 对于 “所需任务”，请输入：

      ```
      3
      ```

      在本教程中，每个任务都会启动一个 Docker 容器。

   f. 展开 “网络” 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。这就是本教程所需要的全部内容。

      > **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 和在您的 VPC 中创建子网。

   i. 对于 “安全组”，选择 “创建新的安全组”，然后执行以下操作：

      i. 在安全组名称中，输入：

```
codecatalyst-ecs-security-group
```

ii. 在安全组描述中，输入：

```
CodeCatalyst ECS security group
```

iii. 选择 添加规则。对于 “类型”，选择 “HTTP”，对于 “源”，选择 “任何地方”。

j. 在底部，选择创建。

k. 等待服务创建完成。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 “上次状态” 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，选择三个任务中的任意一个。
2. 在 “公有 IP” 字段中，选择开放地址。

   此时会出现一个`It Works!`页面。这表明 Amazon ECS 服务成功启动了一项任务，该任务启动了带有 Apache 映像的 Docker 容器。

   在本教程中，您已经手动部署了 Amazon ECS 集群、服务和任务定义以及 Apache 占位符应用程序。所有这些项目都准备就绪后，你就可以创建工作流程了，用教程的 Hello World 应用程序替换 Apache 占位符应用程序。

步骤 3：创建 Amazon ECR 镜像存储库

在本节中，您将在 Amazon Elastic Container Registry (Amazon ECR) 中创建私有镜像存储库。此存储库存储本教程的 Docker 镜像，该镜像将替换您之前部署的 Apache 占位符镜像。

有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

在 Amazon ECR 中创建镜像存储库

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. 请记下该“repositoryUri”:值，例如111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo。

   稍后在向工作流程中添加存储库时需要它。

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 EC2 的权限。
- 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**
>
> 为了节省时间，您可以创建一个名为角色的CodeCatalystWorkflowDevelopmentRole-*spaceName*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。了解该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性较低的场景中使用此角色。本教程假设您正在创建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 Management Console 或 AWS CLI。

AWS Management Console

要创建生成和部署角色，请完成以下一系列步骤。

创建生成角色

1. 为该角色创建策略，如下所示：

   a. 登录到 AWS。

   b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

   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：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

e. 选择下一步。

f. 在权限策略中codecatalyst-ecs-build-policy，搜索并选中其复选框。

g. 选择下一步。

h. 在 “角色名称” 中，输入：

```
codecatalyst-ecs-build-role
```

i. 在角色描述中，输入：

```
CodeCatalyst ECS build role
```

j. 选择 创建角色。

现在，您已经创建了一个包含权限策略和信任策略的构建角色。

3. 按如下方式获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（codecatalyst-ecs-build-role）。

c. 从列表中选择角色。

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. 为该角色创建策略，如下所示：

a. 登录到 AWS。

b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

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：

```
{
    "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**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符。然后，您可以使用资源名称缩小策略的范围。
>
> ```
> "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. 创建部署角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 选择下一步。

f. 在权限策略中codecatalyst-ecs-deploy-policy，搜索并选中其复选框。

g. 选择下一步。

h. 在 “角色名称” 中，输入：

```
codecatalyst-ecs-deploy-role
```

i. 在角色描述中，输入：

```
CodeCatalyst ECS deploy role
```

j. 选择 创建角色。

现在，您已经使用信任策略创建了一个部署角色。

3. 按如下方式获取部署角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（codecatalyst-ecs-deploy-role）。

c. 从列表中选择角色。

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

要创建生成和部署角色，请完成以下一系列步骤。

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：

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

3. 将光标放在最后一个大括号 (}) 之后。
4. 按下**Enter**然后**Ctrl+d**保存文件并退出 cat。

创建生成策略和生成角色

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. 出现提示时，输入以下代码：

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
   ```

```
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

c. 将光标放在最后一个大括号 (}) 之后。

d. 按下**Enter**然后**Ctrl+d**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，记下该"arn":值，例如arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy。稍后你需要这个 ARN。

4. 创建构建角色并将信任策略附加到该角色上：

```
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 ##########*的 ARN 所取代。

6. 显示构建角色的详细信息：

```
aws iam get-role \
      --role-name codecatalyst-ecs-build-role
```

7. 请注意角色的"Arn":值，例如arn:aws:iam::111122223333:role/codecatalyst-ecs-build-role。稍后你需要这个 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. 出现提示时，输入以下代码：

   ```
   {
       "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**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

c. 将光标放在最后一个大括号 (}) 之后。

d. 按下**Enter**然后**Ctrl+d**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。稍后你需要这个 ARN。
4. 创建 deploy 角色并将信任策略附加到该角色上：

```
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*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. 请注意角色的"Arn":值，例如arn:aws:iam::111122223333:role/codecatalyst-ecs-deploy-role。稍后你需要这个 ARN。

第 5 步：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。

向账户连接添加构建和部署角色

1. 在中 CodeCatalyst，导航到您的空间。
2. 选择 AWS accounts (账户)。此时将显示账户关联列表。
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。

   将出现一个下拉列表。该列表显示了所有具有信任策略的 IAM 角色，其中包括`codecatalyst-runner.amazonaws.com`和`codecatalyst.amazonaws.com`服务委托人。

6. 在下拉列表中，选择`codecatalyst-ecs-build-role`，然后选择添加角色。

   > **Note**
   >
   > 如果你看见`The security token included in the request is invalid`，可能是因为你没有正确的权限。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，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 Add role (添加角色)。

   现在，您已将构建和部署角色添加到您的空间。

8. 复制 A mazon CodeCatalyst 显示名称的值。稍后，在创建工作流程时，您将需要此值。

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。

第 7 步：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 — 描述用于你的 Docker 镜像的基础镜像以及应用于它的 Docker 命令。
- `taskdef.json`文件-定义在集群中启动任务时要使用的 Docker 镜像。

文件夹结构如下：

```
.
|— 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.js

index.html

该`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. 在 “文件名” 栏中输入：

```
public-html/index.html
```

> **Important**
>
> 要创建同名文件夹，请务必public-html/添加前缀。index.html预计将在此文件夹中。

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”。

   将index.html以public-html文件夹的形式添加到您的存储库中。

Dockerfile

Dockerfile 描述了要使用的基本 Docker 镜像以及要应用于它的 Docker 命令。有关 Dockerfile 的更多信息，请参阅 Dockerfile 参考。

此处指定的 Dockerfile 表示要使用 Apache 2.4 基础镜像 ()。`httpd`它还包括将名为的源文件复制`index.html`到提供网页的 Apache 服务器上的文件夹的说明。Dockerfile 中的`EXPOSE`指令告诉 Docker 容器正在端口 80 上监听。

添加 Dockerfile

1. 在源存储库中，选择 “创建文件”。
2. 在 “文件名” 栏中输入：

   ```
   Dockerfile
   ```

   请勿包含文件扩展名。

   > **Important**
   >
   > Dockerfile 必须位于存储库的根文件夹中。工作流程的`Docker build`命令希望它在那里。

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”。

   Dockerfile 已添加到您的存储库中。

taskdef.js

您在此步骤中添加`taskdef.json`的文件与您已经在中指定的文件相同，但步骤 2：将占位符应用程序部署到 Amazon ECS有以下区别：

此处的任务定义没有在 (`httpd:2.4`) `image:` 字段中指定硬编码的 Docker 映像名称，而是使用几个变量来表示图像：和。`$REPOSITORY_URI` `$IMAGE_TAG`当您在后续步骤中运行工作流程时，这些变量将被工作流程的生成操作生成的实际值所取代。

有关任务定义参数的详细信息，请参阅 Amazon 弹性容器服务开发者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. 在 “文件名” 栏中输入：

   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"
}
```

   在前面的代码中，替换

   arn: aws: iam:: account_id: role/-role codecatalyst-ecs-task-execution

使用您在中记下的任务执行角色的 ARN。创建任务执行角色

4. 选择 “提交”，然后再次选择 “提交”。

   该taskdef.json文件已添加到您的存储库中。

步骤 8：创建并运行工作流程

在此步骤中，您将创建一个工作流程，该工作流程将源文件生成到 Docker 映像中，然后将该映像部署到您的 Amazon ECS 集群。此部署取代了现有的 Apache 占位符应用程序。

该工作流由以下按顺序运行的构建块组成：

- 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程运行。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程。
- 构建操作 (BuildBackend) — 触发后，该操作使用 Dockerfile 构建 Docker 映像并将映像推送到 Amazon ECR。构建操作还会taskdef.json使用正确的image字段值更新，然后创建此文件的输出构件。此构件用作部署操作的输入，接下来是部署操作。

  有关生成操作的更多信息，请参阅使用工作流程进行构建。
- 部署操作 (DeployToECS)-构建操作完成后，部署操作将查找生成操作 (TaskDefArtifact) 生成的输出项目，找到其taskdef.json内部并将其注册到您的 Amazon ECS 服务。然后，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 Workflows。
2. 选择 “创建工作流程”。
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：

```
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
```

在前面的代码中，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。显示名称可能是一个数字。有关更多信息，请参阅 第 5 步：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*（在属性中Value:），其中包含你在中创建的亚马逊 ECR 存储库的 URI。步骤 3：创建 Amazon ECR 镜像存储库
- *111122223333.dkr#ecr.us-west-2.amazonaws.com#####Run: aws ecr##### ECR* 存储库的 URI 没有图片后缀 ()。/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。
- 两个带有您的地区代码的 *us-west-2* 实例。 AWS 有关区域代码的列表，请参阅中的区域终端节点AWS 一般参考。

> **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 Rend er Amazon ECS 任务定义操作来更新存储库和映像名称，而不是使用前面工作流程代码中显示的和sed命令来更新存储库和映像名称。find有关更多信息，请参阅 使用工作流程修改 Amazon ECS 任务定义文件。

8. （可选）选择验证以确保 YAML 代码在提交之前有效。
9. 选择 Commit (提交)。
10. 在 “提交工作流程” 对话框中，输入以下内容：

    a. 对于 “提交消息”，删除文本并输入：

    ```
    Add first workflow
    ```

    b. 对于 “存储库”，选择codecatalyst-ecs-source-repository。

    c. 在 “分支名称” 中，选择 “主分支”。

    d. 选择 Commit (提交)。

现在，您已经创建了一个工作流程。由于在工作流程顶部定义了触发器，因此工作流程运行会自动启动。具体而言，当您将workflow.yaml文件提交（并推送）到源存储库时，触发器会启动工作流程运行。

查看工作流程运行进度

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程`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，请参阅查看工作流程运行状态和详细信息。

验证部署

1. 打开 Amazon ECS 经典控制台：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意一项。
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，表示 Amazon ECS 服务成功部署了您的应用程序。

第 9 步：对源文件进行更改

在本节中，您将对源存储库中的`index.html`文件进行更改。此更改会导致工作流程生成新的 Docker 映像，使用提交 ID 对其进行标记，将其推送到 Amazon ECR，然后将其部署到 Amazon ECS。

要更改 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`。

   的内容出`index.html`现。
3. 选择编辑。
4. 在第 14 行，将`Hello World`文本更改为`Tutorial complete!`。
5. 选择 “提交”，然后再次选择 “提交”。

   提交会导致启动新的工作流程。

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. 观看部署进度：
   a. 在导航窗格中，选择 C I/CD，然后选择工作流程。
   b. 选择codecatalyst-ecs-workflow查看最新运行情况。
   c. 选择BuildBackend、和 DeployToECS 以查看工作流程的运行进度。
8. 验证您的应用程序是否已更新，如下所示：
   a. 打开 Amazon ECS 经典控制台：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意一项。
   e. 在 “公有 IP” 字段中，选择开放地址。

      此时会出现一个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，以免被收费。

在中 AWS Management Console，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 Amazon EC2 中，删除codecatalyst-ecs-security-group。
4. 在 IAM 身份中心中，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” 操作添加到您的工作流程中。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署到 Amazon ECS” 操作，然后执行以下任一操作：
   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or
   - 选择部署到 Amazon ECS。将出现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程图中，然后打开其配置窗格。
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 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。

11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署到 Amazon ECS” 操作，然后执行以下任一操作：
   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or
   - 选择部署到 Amazon ECS。将出现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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“部署到 Amazon ECS” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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” 操作生成的变量

部署到 Amazon ECS 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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
cluster	在工作流程运行期间部署到的 Amazon ECS 集群的名称。 例如：`codecatalyst-ecs-cluster`
部署平台	部署平台的名称。 硬编码为。`AWS:ECS`
服务	在工作流程运行期间部署到的 Amazon ECS 服务的名称。 例如：`codecatalyst-ecs-service`
task-definition-arn	在工作流程运行期间注册的任务定义的 Amazon 资源名称 (ARN)。 例如：`arn:aws:ecs:us-west-2:111122223333:task-definition/codecatalyst-task-def:8` 前面示例`:8`中的表示已注册的修订版。
部署网址	指向 Amazon ECS 控制台的 “事件” 选项卡的链接，您可以在其中查看与工作流程运行关联的 Amazon ECS 部署的详细信息。 例如：`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 例如：`us-west-2`

“部署到 Amazon ECS” 操作 YAML 定义

以下是 “部署到 Amazon ECS” 操作的 YAM L 定义。要了解如何使用此操作，请参阅使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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: DeployToECS
    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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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 指定操作的主版本、次要版本或补丁版本。

默认值：aws/ecs-deploy@v1。

对应的用户界面：工作流程图/ecs DeployAction _nn/ aws/ecs-deploy @v1 标签

DependsOn

(*ECSDeployAction*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入” 选项卡/ 依赖- 可选

Compute

(*ECSDeployAction*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*ECSDeployAction*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：“配置” 选项卡/高级-可选/计算类型

Fleet

(*ECSDeployAction*/Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队示例：Linux.x86-64.Large，Linux.x86-64.XLarge。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

相应的 UI：“配置” 选项卡/高级-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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 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。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*ECSDeployAction*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

(*ECSDeployAction*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Name

(*ECSDeployAction*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

(*ECSDeployAction*/Environment/Connections/Role)

（必需）

指定 “部署到 Amazon ECS” 操作用于访问的 IAM 角色的名称 AWS。请确保此角色包含以下策略：

- 以下权限策略：

> **Warning**
>
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "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**
>
> 首次使用该角色时，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：

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

确保已将此角色添加到您的账户关联中。要了解有关向账户连接添加 IAM 角色的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角

色的名称。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。了解该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性较低的场景中使用此角色。

对应的用户界面：配置选项卡/'环境/账户/角色'/角色

Inputs

(*ECSDeployAction*/Inputs)

（可选）

本Inputs节定义了工作流程运行期间ECSDeployAction所需的数据。

> **Note**
>
> 每个 “部署到 Amazon ECS” 操作只允许输入一个输入（源或项目）。

相应的 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，则为必填项）

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，则它必须位于另一个操作生成的对象中。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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先前操作的输出对象中，则为必填项）

如果要部署的任务定义文件包含在先前操作生成的对象中，请在此处指定该对象。如果任务定义文件不包含在构件中，则该文件必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Configuration

(*ECSDeployAction*/Configuration)

（必需）

您可以在其中定义操作的配置属性的部分。

相应的 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，请参阅中的区域终端节点AWS 一般参考。

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” 操作会将您的容器化应用程序作为一项任务部署到该集群中。有关 Amazon ECS 集群的更多信息，请参阅《亚马逊弹性容器服务开发人员指南》中的集群。

对应的 UI：“配置” 选项卡/ 集群

service

(*ECSDeployAction*/Configuration/service)

（必需）

指定用于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《亚马逊弹性容器服务开发人员指南》中的 Amazon EC S 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。如果文件位于您的源存储库中，则该路径是相对于源存储库根文件夹的路径。如果您的文件位于先前工作流程操作的对象中，则该路径是相对于对象根文件夹的路径。有关任务定义文件的更多信息，请参阅《亚马逊弹性容器服务开发者指南》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，Amazon ECS 服务可以在不更改服务定义的情况下启动新的部署。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。有关强制进行新部署的更多信息，请参阅《Amazon 弹性容器服务开发人员指南》中的更新服务。

默认：`false`

相应的 UI：“配置” 选项卡/ 强制部署新的服务

codedeploy-appspec

(*ECSDeployAction*/Configuration/codedeploy-appspec)

（如果您已将 Amazon ECS 服务配置为使用蓝/绿部署，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 服务配置为执行蓝/绿部署时，才提供 CodeDeploy 信息。对于滚动更新部署（默认），请省略 CodeDeploy 信息。有关 Amazon ECS 部署的更多信息，请参阅《亚马逊弹性容器服务开发人员指南》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。要让它们出现，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`，则为必填项）

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`，则为必填项）

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

使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service

> **Tip**
>
> 有关向您展示如何使用 “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” 操作添加到您的工作流程中。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亚马逊 Elastic Kubernetes Service (EKS) 中设置的 Kubernetes 集群。有关示例清单，请参阅部署 .yaml中的教程：将应用程序部署到 Amazon EKS。

有关 Kubernetes 的更多信息，请参阅 Kubernetes 文档。

有关 Amazon EKS 的更多信息，请参阅什么是亚马逊 EKS？ 在 Amazon EKS 用户指南中。

“部署到 Kubernetes 集群” 操作的工作原理

部署到 Kubernetes 集群的工作原理如下：

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。该操作配置为指`kubectl`向您在配置操作时提供的 Amazon EKS 集群。接下来，该`kubectl`实用程序是运行该`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 *`my-manifest.yaml`* 中的指令，将您的应用程序作为一组容器和容器部署到已配置的集群中。有关此命令的更多信息，请参阅 Kubernetes 参考文档中的 k ub ectl 应用主题。

主题

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 部署的应用程序是一个简单的 “Hello，World！” 网站基于 Apache 网络服务器 Docker 镜像构建。本教程将引导您完成所需的准备工作，例如设置开发计算机和 Amazon EKS 集群，然后介绍如何创建工作流程来构建应用程序并将其部署到集群中。

初始部署完成后，本教程将指导您对应用程序源进行更改。此更改会生成一个新的 Docker 镜像，并将其推送到包含新修订信息的 Docker 镜像存储库。然后，Docker 镜像的新修订版部署到亚马逊 EKS 中。

> **Tip**
>
> 与其完成本教程的学习，不如使用蓝图为您完成完整的 Amazon EKS 设置。您需要使用 E KS 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。
- 在你的空间中，你需要一个空的、从头开始的 CodeCatalyst项目，名为：

  ```
  codecatalyst-eks-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，你需要一个名为：的空 CodeCatalyst 源代码库

  ```
  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 C CodeCatalyst I/CD 环境（不是开发环境），名为：

  ```
  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。

  有关更多信息，请参阅 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

第 1 步：设置开发机器

本教程的第一步是使用本教程中要使用的几个工具来配置开发机器。这些工具是：

- `eksctl`实用程序 — 用于创建集群
- `kubectl`实用程序 — 的先决条件 `eksctl`
- 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。
- 在亚马逊 EKS 用户指南中安装 kubectl。
- 在《亚马逊 EKS 用户指南》中安装或升级 eksctl。
- 在《AWS Command Line Interface 用户指南》 AWS CLI中安装或更新最新版本的。

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 “主分支”，然后选择 “创建”。

   您的开发环境将在新的浏览器选项卡中启动，您的存储库 (codecatalyst-eks-source-repository) 已克隆到其中。

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. 检查是否出现了版本。

   你现在已经安装好了kubectl。

安装和配置 eksctl

> **Note**
>
> eksctl不是严格必需的，因为你可以kubectl改用。但是，eksctl它具有自动执行大部分集群配置的好处，因此是本教程推荐的工具。

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. 检查是否出现了版本。

   你现在已经安装好了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 用户指南 AWS CLI中的配置。

1. 按如下方式创建 IAM 身份中心用户：

   a. 登录 AWS Management Console 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 身份中心，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 账户选项卡来验证连接了哪个账户。有关更多信息，请参阅 创建空间。

   b. 在导航窗格中，选择 Users，然后选择 Add user。

   c. 在用户名中，输入：

      ```
      codecatalyst-eks-user
      ```

   d. 在“密码”下，选择“生成可与该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 中尚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 “将用户添加到群组” 页面上，选择 “下一步”。

k. 在查看并添加用户页面上，查看信息并选择添加用户。

将出现 “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. 按如下方式创建权限集：

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。此策略向所有人提供完全权限 AWS 服务。

c. 选择下一步。

d. 在权限集名称中，删除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. 获取codecatalyst-eks-user的访问密钥和会话令牌，如下所示：

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。您应该早点将此信息复制到文本编辑器中。

> **Note**
>
> 如果您没有这些信息，请前往 IAM Identity Center 的codecatalyst-eks-user详细信息页面，选择重置密码，生成一次性密码 [...] ，然后再次重置密码以在屏幕上显示信息。

b. 退出 AWS。

c. 将 AWS 访问门户 URL 粘贴到浏览器的地址栏中。

d. 使用以下方式登录：

- 用户名:

```
codecatalyst-eks-user
```

- 密码:

one-time-password

e. 在设置新密码中，输入新密码并选择设置新密码。

屏幕上会出现一个 AWS 账户框。

f. 选择 AWS 账户，然后选择 AWS 账户 向其分配codecatalyst-eks-user用户和权限集的名称。

g. 在旁边codecatalyst-eks-permission-set，选择命令行或编程访问。

h. 复制页面中间的命令。它们看起来类似于以下内容：

```
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类似以下内容的消息：
> `Unable to locate credentials. You can configure credentials by running "aws configure".`
> 或者：
> `ExpiredToken: The security token included in the request is expired`
> ... 这是因为您的 AWS CLI 会话已过期。在这种情况下，请不要运行该`aws configure`命令。相反，请使用本过程的第 4 步（`Obtain codecatalyst-eks-user's access key and session token`以开头）中的说明刷新会话。

步骤 2：创建 Amazon EKS 集群

在本节中，您将在 Amazon EKS 中创建一个集群。以下说明描述了使用创建集群的快速方法`eksctl`，但如果您需要详细说明，请参阅：

- 亚马逊 EKS 用户指南中的 eksctl 入门

  或者

- 控制台入门和 AWS CLI Amazon EKS 用户指南（本主题提供创建集群的`kubectl`说明）

> **Note**
>
> 与 Amazon EKS 的 CodeCatalyst 集成不支持@@ 私有集群。

开始之前

确保您已在开发计算机上完成以下任务：

- 已安装该eksctl实用程序。
- 已安装该kubectl实用程序。
- 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，请参阅第 1 步：设置开发机器。

创建集群

> **Important**
>
> 请勿使用 Amazon EKS 服务的用户界面创建集群，因为集群配置不正确。使用该eksctl实用程序，如以下步骤所述。

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，将显示一条类似于以下内容的消息：

   ```
   EKS cluster "codecatalyst-eks-cluster" in "us-west-2" region is ready
   ```

   > **Note**
   >
   > AWS 创建集群时，您将看到多waiting for CloudFormation stack条消息。这是预期行为。

3. 验证您的集群是否已成功创建：

```
kubectl cluster-info
```

您将看到一条类似于以下内容的消息，表示集群创建成功：

```
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
```

第 3 步：创建 Amazon ECR 镜像存储库

在本节中，您将在 Amazon Elastic Container Registry (Amazon ECR) 中创建私有镜像存储库。此存储库存储了本教程的 Docker 镜像。

有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

在 Amazon ECR 中创建镜像存储库

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。

   本教程假设您命名了存储库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. 请记下该“repositoryUri”:值，例如111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo。

   稍后在向工作流程中添加存储库时需要它。

步骤 4：添加源文件

在本节中，您将向源存储库中添加应用程序源文件 (codecatalyst-eks-source-repository)。它们包括：

- index.html文件 — 显示 “Hello，World！” 浏览器中的消息。
- Dockerfile — 描述用于你的 Docker 镜像的基础镜像以及应用于它的 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文件显示 “Hello，World！” 浏览器中的消息。

添加 index.html 文件

1. 转到您的开发环境。
2. 在中codecatalyst-eks-source-repository，创建一个名为的文件夹public-html。
3. 在中/public-html，创建一个名为的文件index.html，其中包含以下内容：

```
<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
```

将index.html以public-html文件夹的形式添加到您的存储库中。

Dockerfile

Dockerfile 描述了要使用的基本 Docker 镜像以及要应用于它的 Docker 命令。有关 Dockerfile 的更多信息，请参阅 Dockerfile 参考。

此处指定的 Dockerfile 表示要使用 Apache 2.4 基础镜像 ()。httpd它还包括将名为的源文件复制index.html到提供网页的 Apache 服务器上的文件夹的说明。Dockerfile 中的EXPOSE指令告诉 Docker 容器正在端口 80 上监听。

添加 Dockerfile

1. 在中codecatalyst-eks-source-repository，创建一个名为的文件Dockerfile，其中包含以下内容：

```
FROM httpd:2.4
COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
EXPOSE 80
```

请勿包含文件扩展名。

> **Important**
>
> Dockerfile 必须位于存储库的根文件夹中。工作流程的`Docker build`命令希望它在那里。

2. 添加、提交和推送：

```
git add .
git commit -m "add Dockerfile"
git push
```

Dockerfile 已添加到您的存储库中。

部署 .yaml

在本节中，您将向存储库中添加`deployment.yaml`文件。该`deployment.yaml`文件是一个 Kubernetes 清单，它定义了两种要运行的 Kubernetes 资源类型或类型：“服务” 和 “部署”。

- “服务” 将负载均衡器部署到 Amazon EC2 中。负载均衡器为您提供面向互联网的公共 URL 和标准端口（端口 80），您可以使用它们浏览到 “Hello，World！” 应用程序的修订。
- “部署” 部署了三个 pod，每个 pod 将包含一个带有 “Hello，World！” 的 Docker 容器 应用程序的修订。这三个 Pod 将部署到您创建集群时创建的节点上。

本教程中的清单很短；但是，清单可以包含任意数量的 Kubernetes 资源类型，例如 pod、作业、入口和网络策略。此外，如果您的部署很复杂，则可以使用多个清单文件。

添加部署.yaml 文件

1. 在中`codecatalyst-eks-source-repository`，创建一个名为的文件夹`Kubernetes`。
2. 在中`/Kubernetes`，创建一个名为的文件`deployment.yaml`，其中包含以下内容：

```
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
```

该deployment.yaml文件将添加到存储库中名为的文件夹中Kubernetes。

现在，您已经添加了所有源文件。

花点时间仔细检查您的工作，并确保将所有文件放在正确的文件夹中。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 EC2 的权限。
- 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**
>
> 为了节省时间，您可以创建一个名为角色的CodeCatalystWorkflowDevelopmentRole-*spaceName*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。了解该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性较低的场景中使用此角色。本教程假设您正在创建前面列出的两个角色。

要创建生成和部署角色，请完成以下一系列步骤。

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，创建一个名为的文件codecatalyst-eks-trust-policy.json，其中包含以下内容：

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

2. 为生成角色创建生成策略

- 在Cloud9-*long-string*目录中，创建一个名为的文件codecatalyst-eks-build-policy.json，其中包含以下内容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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，创建一个名为的文件codecatalyst-eks-deploy-policy.json，其中包含以下内容：

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

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

现在，您已经在开发环境中添加了三个策略文档。你的目录结构现在如下所示：

```
|— 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，记下该"arn":值，例如arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy。稍后你需要这个 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。稍后你需要这个 ARN。

6. 创建生成角色

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 ###########*的 ARN 所取代。

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. 请注意角色的"Arn":值，例如arn:aws:iam::111122223333:role/codecatalyst-eks-build-role。稍后你需要这个 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 ##########*的 ARN 所取代。

4. 按 Enter 键。
5. 输入：

```
aws iam get-role \
      --role-name codecatalyst-eks-deploy-role
```

6. 按 Enter 键。
7. 请注意角色的"Arn":值，例如arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role。稍后你需要这个 ARN。

现在，您已经创建了构建和部署角色并记下了他们的 ARN。

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。这样，这些角色就可以在您的工作流程中使用。

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。此时会出现账户列表。
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（可能是一个数字。）稍后，在创建工作流程时，您将需要此值。
5. 选择显示名称。
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。

   将出现一个下拉列表。该列表显示构建和部署角色，以及具有包括codecatalyst-runner.amazonaws.com和codecatalyst.amazonaws.com服务委托人的信任策略的任何其他 IAM 角色。

8. 从下拉列表中添加：

   - `codecatalyst-eks-build-role`
   - `codecatalyst-eks-deploy-role`

   > **Note**
   >
   > 如果你看见`The security token included in the request is invalid`，可能是因为你没有正确的权限。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

您必须将您在中创建的部署角色添加到 Kubernetes `ConfigMap` 文件中步骤 5：创建 AWS 角色，才能让 “部署到 Kubernetes 集群” 操作（在您的工作流程中）能够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### ######### AR* N。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，请参阅管理 IAM 用户和角色。

将显示一条类似于以下内容的消息：

```
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. 用红色斜体添加文本：

```
# 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### ######### AR* N。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 A mazon EKS 用户指南中的启用 IAM 委托人访问您的集群。

现在，您已经授予了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：创建并运行工作流程

在此步骤中，您将创建一个工作流程，该工作流程将源文件生成为 Docker 映像，然后将该映像部署到 Amazon EKS 集群中的树舱中。

该工作流由以下按顺序运行的构建块组成：

- 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程运行。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程。
- 构建操作 (BuildBackend) — 触发后，该操作使用 Dockerfile 构建 Docker 映像并将映像推送到 Amazon ECR。生成操作还会使用正确的值更新deployment.yaml文件中的$REPOSITORY_URI和$IMAGE_TAG变量，然后创建该文件和该Kubernetes文件夹中任何其他文件的输出对象。在本教程中，Kubernetes文件夹中唯一的文件是，deployment.yaml但您可以包含更多文件。该构件用作部署操作的输入，接下来是部署操作。

  有关生成操作的更多信息，请参阅使用工作流程进行构建。
- 部署操作 (DeployToEKS)-生成操作完成后，部署操作将查找生成操作 (Manifests) 生成的输出对象，并在其中找到deployment.yaml文件。然后，该操作按照deployment.yaml文件中的说明运行三个 pod，每个吊舱都包含一个 “Hello，World！” Docker 容器 — 在你的 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 C I/CD，然后选择工作流程。
4. 选择 “创建工作流程”。
5. 对于源存储库，选择codecatalyst-eks-source-repository。
6. 对于 Branch，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程定义文件：

> **Note**
>
> 有关工作流程定义文件的更多信息，请参阅工作流程 YAML 定义。

```
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/
```

在前面的代码中，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。显示名称可能是一个数字。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*（在属性中Value:），其中包含你在中创建的亚马逊 ECR 存储库的 URI。第 3 步：创建 Amazon ECR 镜像存储库
- *111122223333.dkr#ecr.us-west-2.amazonaws.com#####Run: aws ecr##### ECR* 存储库的 URI 没有图片后缀 ()。/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。
- 两个带有您的地区代码的 *us-west-2* 实例。 AWS 有关区域代码的列表，请参阅中的区域终端节点AWS 一般参考。

> **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）选择验证以确保 YAML 代码在提交之前有效。
11. 选择 Commit (提交)。
12. 在“提交工作流程”对话框中，输入以下内容：

    a. 对于“提交消息”，删除文本并输入：

    ```
    Add first workflow
    ```

    b. 对于“存储库”，选择codecatalyst-eks-source-repository。

    c. 在“分支名称”中，选择“主分支”。

    d. 选择 Commit (提交)。

现在，您已经创建了一个工作流程。由于在工作流程顶部定义了触发器，因此工作流程运行会自动启动。具体而言，当您将workflow.yaml文件提交（并推送）到源存储库时，触发器会启动工作流程运行。

查看工作流程运行进度

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程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，请参阅查看工作流程运行状态和详细信息。

验证部署

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

2. 在左侧的底部附近，选择负载均衡器。
3. 选择在 Kubernetes 部署过程中创建的负载均衡器。如果您不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. 选择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。

   '你好，世界！' 网页出现在您的浏览器中，表示您已成功部署应用程序。

第 9 步：对源文件进行更改

在本节中，您将对源存储库中的index.html文件进行更改。此更改会导致工作流程生成新的 Docker 映像，使用提交 ID 对其进行标记，将其推送到 Amazon ECR，然后将其部署到 Amazon ECS。

要更改 index.html

1. 转到您的开发环境。
2. 在终端提示符下，切换到您的源存储库：

```
cd /projects/codecatalyst-eks-source-repository
```

3. 获取最新的工作流程更改：

```
git pull
```

4. 打开 codecatalyst-eks-source-repository/public-html/index.html。
5. 在第 14 行，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
```

   工作流程运行会自动启动。

7. （可选）输入：

```
git show HEAD
```

记下index.html变更的提交 ID。此提交 ID 将被标记为 Docker 镜像，该镜像将由您刚开始的工作流程运行部署。

8. 观看部署进度：

   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查看最新运行情况。

   c. 选择 BuildBackend，然后选择 DeployToEKS 以查看工作流程的运行进度。

9. 验证您的应用程序是否已更新，如下所示：

   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

   b. 在左侧的底部附近，选择负载均衡器。

   c. 选择在 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。

   '教程完成了！' 网页出现在您的浏览器中，表示您成功部署了应用程序的新修订版。

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 进行标记。

清理

您应该清理环境，这样您就不会因为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. 删除您的集群：

   - 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 AWS CloudFormation 堆栈、节点组（在 Amazon EC2 中）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 身份中心中，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

按照以下说明将 “部署到 Kubernetes 集群” 操作添加到您的工作流程中。

开始之前

在将 “部署到 Kubernetes 集群” 操作添加到工作流程之前，必须准备好以下内容：

> **Tip**
>
> 要快速设置这些先决条件，请按照中的说明进行操作教程：将应用程序部署到 Amazon EKS。

- 亚马逊 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。
- 至少有一个 Dockerfile 描述了如何将你的应用程序组装成 Docker 镜像。有关 Dockerfiles 的更多信息，请参阅 Dockerfile 参考文档。
- 至少有一个 Kubernetes 清单文件，该文件在 Kubernetes 文档中被称为配置文件或配置。有关更多信息，请参阅 Kubernetes 文档中的管理资源。
- 一个 IAM 角色，它使部署到 Kubernetes 集群操作能够访问您的 Amazon EKS 集群并与之交互。有关更多信息，请参阅中的Role主题“部署到 Kubernetes 集群” 操作 YAML 定义。

  创建此角色后，必须将其添加到：

  - 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，请参阅向账户连接添加 IAM 角色。
- 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 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。
- 支持的源存储库 CodeCatalyst。存储库存储您的应用程序源文件、Dockerfiles 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署到 Kubernetes 集群” 操作，然后执行以下操作之一：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择部署到 Kubernetes 集群。将出现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程图中，然后打开其配置窗格。
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 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。

7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署到 Kubernetes 集群” 操作，然后执行以下操作之一：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择部署到 Kubernetes 集群。将出现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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“部署到 Kubernetes 集群” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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” 操作生成的变量

部署到 Kubernetes 集群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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
cluster	在工作流程运行期间部署到的 Amazon EKS 集群的 Amazon.com 资源名称 (ARN)。 例如：`arn:aws:eks:us-west-2:111122223333:cluster/codecatalyst-eks-cluster`
部署平台	部署平台的名称。 硬编码为。`AWS:EKS`
metadata	预留。与工作流程运行期间部署的集群相关的 JSON 格式的元数据。

键	值
命名空间	部署集群的 Kubernetes 命名空间。 例如：`default`
资源	预留。与工作流程运行期间部署的资源相关的 JSON 格式的元数据。
服务器	您可以使用管理工具（例如）与集群通信的 API 服务器终端节点的名称`kubectl`。 有关 API 服务终端节点的更多信息，请参阅 Amazon EKS 用户指南中的 Amazon E KS 集群终端节点访问控制。 例如：`https://random-string .gr7.us-west-2.eks.amazonaws.com`

“部署到 Kubernetes 集群” 操作 YAML 定义

以下是 “部署到 Kubernetes 集群” 操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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。除非要更改版本，否则请勿更改此属性。有关更多信息，请参阅 指定操作的主版本、次要版本或补丁版本。

默认值：aws/kubernetes-deploy@v1。

对应的用户界面：工作流程图/ DeployToKubernetesCluster _nn/ aws/kubernetes-deploy @v1 标签

DependsOn

(*DeployToKubernetesCluster*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关“依赖”功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入”选项卡/ 依赖- 可选

Compute

(*DeployToKubernetesCluster*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*DeployToKubernetesCluster*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。

- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：“配置” 选项卡/高级-可选/计算类型

Fleet

(*DeployToKubernetesCluster*/Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：Linux.x86-64.Large，Linux.x86-64.XLarge。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

相应的 UI：“配置” 选项卡/高级-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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 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。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*DeployToKubernetesCluster*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

(*DeployToKubernetesCluster*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Name

(*DeployToKubernetesCluster*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

(*DeployToKubernetesCluster*/Environment/Connections/Role)

（必需）

指定 “部署到 Kubernetes” 集群操作用于访问的 IAM 角色的名称。 AWS请确保此角色包含以下策略：

- 以下权限策略：

> **Warning**
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

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

> **Note**
> 首次使用该角色时，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：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

确保将此角色添加到：

- 您的账户关联。要了解有关向账户连接添加 IAM 角色的更多信息，请参阅向账户连接添加 IAM 角色。
- 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**
>
> 如果你愿意，你可以在这里指定`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色的名称。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。了解该`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性较低的场景中使用此角色。如果您决定使

用该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色，请务必按照文档中管理 IAM 用户和角色中的说明将其添加到您的 ConfigMap eksctl文件中。

对应的用户界面：配置选项卡/'环境/账户/角色'/角色

Inputs

(*DeployToKubernetesCluster*/Inputs)

（可选）

本Inputs节定义了工作流程运行期间DeployToKubernetesCluster所需的数据。

> **Note**
>
> 每个 “部署到 Amazon EK S” 操作只允许输入一个输入（源或项目）。

相应的 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，则为必填项）

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，则它们必须位于其他操作生成的项目中。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 UI：“输入” 选项卡/ “来源”- 可选

Artifacts - input

(*DeployToKubernetesCluster*/Inputs/Artifacts)

（如果清单文件存储在先前操作的输出项目中，则为必填项）

如果 Kubernetes 清单文件包含在先前操作生成的构件中，请在此处指定该构件。如果您的清单文件不包含在构件中，则它们必须位于您的源存储库中。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Configuration

(*DeployToKubernetesCluster*/Configuration)

（必需）

您可以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）

指定将 Kubernetes 应用程序部署到的 Kubernetes 命名空间。`default`如果您没有在集群中使用命名空间，请使用。有关命名空间的更多信息，请参阅 Kubernetes 文档中的使用 Kubernetes 命名空间细分集群。

如果省略命名空间，`default`则使用值。

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，请参阅中的区域终端节点AWS 一般参考。

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 集群操作会将您的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 文档中被称为配置文件、配置文件或简称为配置。

如果您使用多个清单文件，请将它们放在一个文件夹中并引用该文件夹。Manifest 文件由 Kubernetes 按字母数字进行处理，因此请务必在文件名前加上递增的数字或字母，以控制处理顺序。例如：

```
00-namespace.yaml

01-deployment.yaml
```

如果清单文件位于源存储库中，则该路径是相对于源存储库根文件夹的路径。如果文件位于先前工作流程操作的对象中，则该路径相对于对象根文件夹。

示例：

```
Manifests/

deployment.yaml

my-deployment.yml
```

不要使用通配符 (*)。

> **Note**
>
> 不@@ 支持 Helm 图表和自定义文件。

有关清单文件的更多信息，请参阅 Kubernetes 文档中的组织资源配置。

相应的 UI：“配置” 选项卡/ “清单”

使用工作流程部署 AWS 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 AWS CloudFormation 堆栈” 操作添加到工作流程中。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：

- AWS CloudFormation 模板-有关更多信息，请参阅使用 AWS 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：使用部署无服务器应用程序 AWS CloudFormation。

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。然后，该操作会等待更改部署完毕，并根据结果将自己标记为成功或失败。

如果您已经有包含要部署的资源的 AWS 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 AWS 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 AWS CloudFormation stack 操作中使用什么。

> **Tip**
>
> 有关向您展示如何使用 “部署 AWS CloudFormation 堆栈” 操作部署无服务器应用程序的教程，请参阅教程：使用部署无服务器应用程序 AWS CloudFormation。

主题

- 教程：使用部署无服务器应用程序 AWS CloudFormation
- 添加 “部署 AWS CloudFormation 堆栈” 操作

- 配置回滚
- “部署 AWS CloudFormation 堆栈” 操作生成的变量
- “部署 AWS CloudFormation 堆栈” 操作 YAML 定义

教程：使用部署无服务器应用程序 AWS CloudFormation

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简单的 Web 应用程序，它输出 “Hello World” 消息。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AWS CloudFormation。

主题

- 先决条件
- 步骤 1：创建源存储库
- 步骤 2：创建 AWS 角色
- 步骤 3：将 AWS 角色添加到 CodeCatalyst
- 步骤 4：创建 Amazon S3 存储桶
- 第 5 步：添加源文件
- 步骤 6：创建并运行工作流程
- 第 7 步：做出改变
- 清理

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。
- 在你的空间中，你需要一个空的、从头开始的 CodeCatalyst项目，名为：

```
codecatalyst-cfn-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-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。

  有关更多信息，请参阅 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。

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的存储库`codecatalyst-cfn-source-repository`。

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 AWS 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 AWS CloudFormation 堆栈操作是您的工作流程的一部分。
- 构建角色-授予 CodeCatalyst 构建操作权限以访问您的 AWS 账户并写入存储您的无服务器应用程序包的 Amazon S3。生成操作是您的工作流程的一部分。

- 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**
>
> 为了节省时间，您可以创建一个名为角色的`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*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。了解该`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色具有非常广泛的权限，可能会带来安全风险。我们建议您仅在教程和安全性较低的场景中使用此角色。本教程假设您正在创建前面列出的三个角色。

> **Note**
>
> 还需要一个 Lambda 执行角色，但您无需立即创建它，因为当您在步骤 5 中运行工作流程时，`sam-template.yml`文件会为您创建它。

创建部署角色

1. 为该角色创建策略，如下所示：
   a. 登录到 AWS。
   b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   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：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建部署角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

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. 在 “角色名称” 中，输入：

```
codecatalyst-deploy-role
```

i. 在角色描述中，输入：

```
CodeCatalyst deploy role
```

j. 选择 创建角色。

现在，您已经创建了一个带有信任策略和权限策略的部署角色。

3. 按如下方式获取部署角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（codecatalyst-deploy-role）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的部署角色并获得了其 ARN。

创建生成角色

1. 为该角色创建策略，如下所示：

   a. 登录到 AWS。

   b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

   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：

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

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

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. 在 “角色名称” 中，输入：

```
codecatalyst-build-role
```

i. 在角色描述中，输入：

```
CodeCatalyst build role
```

j. 选择 创建角色。

现在，您已经创建了一个包含信任策略和权限策略的构建角色。

3. 按如下方式获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（`codecatalyst-build-role`）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的构建角色并获得了其 ARN。

创建堆栈角色

1. AWS 使用要部署堆栈的账户登录。
2. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 按如下方式创建堆栈角色：

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. 在底部，选择下一步。

g. 使用搜索框找到以下权限策略，然后选中相应的复选框。

> **Note**
>
> 如果您搜索的政策未显示，请务必选择 “清除筛选条件”，然后重试。

- CloudWatchFullAccess
- AWS CloudFormationFullAccess
- IAM FullAccess
- AWS Lambda_ FullAccess
- 亚马逊 API GatewayAdministrator
- 亚马逊 3 FullAccess
- 亚马逊 EC2 ContainerRegistryFullAccess

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。有关更多信息，请参阅《AWS Serverless Application Model 开发人员指南》中的权限。

h. 选择下一步。

i. 在 “角色名称” 中，输入：

```
codecatalyst-stack-role
```

j. 选择 创建角色。

4. 按如下方式获取堆栈角色的 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（`codecatalyst-stack-role`）。

c. 从列表中选择角色。

d. 在摘要部分，复制 ARN 值。稍后您将需要用到它。

现在，您已经创建了具有相应权限的堆栈角色，并且已获得其 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) 添加到连接中。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。

向账户连接中添加构建和部署角色

1. 在中 CodeCatalyst，导航到您的空间。
2. 选择 AWS accounts (账户)。此时将显示账户关联列表。
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。

   将出现一个下拉列表。该列表显示了所有具有信任策略的 IAM 角色，其中包括codecatalyst-runner.amazonaws.com和codecatalyst.amazonaws.com服务委托人。
6. 在下拉列表中，选择codecatalyst-build-role，然后选择添加角色。
7. 选择添加 IAM 角色，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 Add role (添加角色)。

   现在，您已将构建和部署角色添加到您的空间。
8. 复制 A mazon CodeCatalyst 显示名称的值。稍后，在创建工作流程时，您将需要此值。

步骤 4：创建 Amazon S3 存储桶

在此步骤中，您将创建一个 Amazon S3 存储桶，用于存储无服务器应用程序的部署包.zip 文件。

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. 在存储桶名称中，输入：

```
codecatalyst-cfn-s3-bucket
```

4. 对于 AWS 区域（亚马逊云科技区域），选择一个区域。本教程假设您选择了美国西部（俄勒冈）us-west-2。有关 Amazon S3 支持的区域的信息，请参阅中的亚马逊简单存储服务终端节点和配额AWS 一般参考。
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 区域创建了一个名**codecatalyst-cfn-s3-bucket**为的存储桶。

第 5 步：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。该`hello-world`文件夹包含您将要部署的应用程序文件。该`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

该`.npmignore`文件指示 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 不包括该`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. 选择你的项目，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. 在文件名中输入：

```
.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，您已经在存储库的根目录.npmignore中创建了一个名为的文件。

打包.json 文件

该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、依赖关系以及描述如何与应用程序交互和运行应用程序的其他详细信息。

本教程package.json中的包括依赖项列表和test脚本。测试脚本执行以下操作：

- 使用 mocha，测试脚本运行中指定的单元测试，hello-world/tests/unit/并使用 x unit 报告器将结果写入junit.xml文件。
- 使用伊斯坦布尔 (nyc)，测试脚本使用三叶草报告器生成代码覆盖率报告 (clover.xml)。有关更多信息，请参阅伊斯坦布尔文档中的使用备用报告器。

添加 package.json 文件

1. 在存储库中的 “文件” 中，选择 “创建文件”。
2. 在文件名中输入：

```
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，您已在存储库的根目录中`package.json`添加了一个名为的文件。

sam-template.yml 文件

该`sam-template.yml`文件包含部署 Lambda 函数和 API Gateway 以及一起配置它们的说明。它遵循AWS Serverless Application Model 模板规范，该规范扩展了 AWS CloudFormation 模板规范。

在本教程中，您将使用 AWS SAM 模板而不是常规 AWS 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`创建了一个 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。

尽管本教程使用预先编写的模板，但您可以使用构建操作生成一个模板作为工作流程的一部分。有关更多信息，请参阅 使用工作流程部署 AWS 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. 在文件名中输入：

```
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，您已在存储库的根文件夹sam-template.yml下添加了一个名为的文件。

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。工作流使用此实用程序打包hello-world源代码。

添加 setup-sam.sh 文件

1. 在存储库中的 “文件” 中，选择 “创建文件”。
2. 在文件名中输入：

   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，将 *us-west-2* 替换为您所在的地区。 AWS

4. 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setup-sam.sh添加了一个名为的文件。

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. 在文件名中输入：

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，您已经创建了一个名为的文件夹hello-world和一个名为的文件app.js。

test-handler.js 文件

该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. 在文件名中输入：

   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，您已在该hello-world/tests/unit文件夹test-handler.js下添加了一个名为的文件。

现在，您已经添加了所有源文件。

花点时间仔细检查您的工作，并确保将所有文件放在正确的文件夹中。文件夹结构如下：

```
.
|— hello-world
|   |— tests
|      |— unit
|         |— test-handler.js
|   |— app.js
|— .npmignore
|— README.md
|— package.json
|— sam-template.yml
|— setup-sam.sh
```

步骤 6：创建并运行工作流程

在此步骤中，您将创建一个工作流程，用于打包您的 Lambda 源代码并进行部署。该工作流由以下按顺序运行的构建块组成：

- 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程运行。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程。

- 测试操作 (Test)-触发时，此操作将安装 N ode 包管理器 (npm)，然后运行该npm run test命令。此命令告诉 npm 运行package.json文件中定义的test脚本。该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，请参阅打包.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，请参阅使用工作流程进行测试。

- 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，这是 Lambda 服务所期望的。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。

  有关生成操作的更多信息，请参阅使用工作流程进行构建。

- 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 C I/CD，然后选择工作流程。
2. 选择 “创建工作流程”。
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：

```
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
```

在前面的代码中，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。显示名称可能是一个数字。有关更多信息，请参阅 步骤 3：将 AWS 角色添加到 CodeCatalyst。
- *codecatalyst-build-role*使用您在中创建的构建角色的名称步骤 2：创建 AWS 角色。
- *codecatalyst-cfn-s3 ###*，名称为您在中创建的 Amazon S3 存储桶。步骤 4：创建 Amazon S3 存储桶
- 两个 *us-west-2* 实例，分别是您的 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）。这些区域可能有所不同。本教程假设两个区域都设置为us-west-2。有关 Amazon S3 和支持的区域的详细信息 AWS CloudFormation，请参阅中的服务终端节点和配额AWS 一般参考。
- *codecatalyst-deploy-role*使用您在中创建的部署角色的名称步骤 2：创建 AWS 角色。
- *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################### (AR*N)。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，请将、和 arn*: aws: iam:: 111122223333: StackRole role/ ##codecatalyst-build-role####### AR* N。*codecatalyst-deploy-role*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属性的信息，请参阅“部署 AWS CloudFormation 堆栈” 操作 YAML 定义。

8. （可选）选择验证以确保 YAML 代码在提交之前有效。
9. 选择 Commit (提交)。
10. 在 “提交工作流程” 对话框中，输入以下内容：

    a. 对于工作流程文件名，请保留默认值codecatalyst-cfn-workflow。

    b. 在 “提交消息” 中，输入：

    ```
    add initial workflow file
    ```

    c. 对于 “存储库”，选择codecatalyst-cfn-source-repository。

    d. 在 “分支名称” 中，选择 “主分支”。

    e. 选择 Commit (提交)。

现在，您已经创建了一个工作流程。由于在工作流程顶部定义了触发器，因此工作流程运行会自动启动。具体而言，当您将codecatalyst-cfn-workflow.yaml文件提交（并推送）到源存储库时，触发器会启动工作流程运行。

查看正在运行的工作流程

1. 在导航窗格中，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程: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，请参阅查看工作流程运行状态和详细信息。
8. DeployCloudFormationStack操作完成后，请执行以下操作：

   - 如果工作流程运行成功，请转到下一个过程。
   - 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 AWS CloudFormation 控制台删除`codecatalyst-cfn-stack`堆栈。

验证部署

1. 成功部署后，从靠近顶部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。

   将显示来自 Lambda 函数的 hello wor ld JSON 消息，表示工作流程已成功部署和配置 Lambda 函数和 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 在工作流程图中显示已部署应用程序的 URL。

验证单元测试结果和代码覆盖率

1. 在工作流程图中，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 Management Console 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 名称来自工作流程定义文件 (`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. 你可以看到 API Gateway 是如何触发该函数的。此集成是根据 AWS SAM `AWS::Serverless::Function`资源类型自动配置的。

第 7 步：做出改变

在此步骤中，您将对 Lambda 源代码进行更改并提交。此提交将启动新的工作流程运行。此次运行以蓝绿色方案部署新的 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 来源

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，改`hello world`为**`Tutorial complete!`**。
   e. 选择 “提交”，然后再次选择 “提交”。

      提交会导致工作流程运行启动。此次运行将失败，因为您尚未更新单元测试以反映名称的更改。
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，更改`hello world`为**`Tutorial complete!`**。
   d. 选择 “提交”，然后再次选择 “提交”。

      提交会导致另一个工作流程开始运行。这次运行将成功。

6. 在导航窗格中，选择 C I/CD，然后选择工作流程。
7. 选择codecatalyst-cfn-workflow，然后选择 “运行”。
8. 选择最近一次运行的运行 ID。它应该仍在进行中。
9. 选择 “测试” BuildBackend、“和” DeployCloudFormationStack以查看工作流程的运行进度。
10. 工作流程完成后，选择顶部附近的变量 (7)。
11. 旁边 HelloWorldApi，将 https:// URL 粘贴到浏览器中。

    浏览器中会显示一条Tutorial complete!消息，表明您的新应用程序已成功部署。

清理

清理本教程中使用的文件和服务，以免被收费。

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 Management Console

1. 清理一下 CloudFormation，如下所示：
   a. 打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
   b. 删除codecatalyst-cfn-stack。

      删除堆栈会从 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. 删除存储桶中的内容。

d. 删除存储桶。

3. 在 IAM 中进行清理，如下所示：

   a. 通过 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 AWS CloudFormation 堆栈。

添加 “部署 AWS CloudFormation 堆栈” 操作

按照以下说明将 “部署 AWS 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 AWS 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署 AWS 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程图中，然后打开其配置窗格。

     Or

- 选择 “部署 AWS CloudFormation 堆栈”。将出现 “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程图中，然后打开其配置窗格。

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 AWS CloudFormation 堆栈” 操作 YAML 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 AWS 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 “部署 AWS 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程图中，然后打开其配置窗格。

     Or

   - 选择 “部署 AWS CloudFormation 堆栈”。将出现 “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程图中，然后打开其配置窗格。

10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“部署 AWS CloudFormation 堆栈” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 AWS CloudFormation 堆栈操作失败，它将导致堆栈回滚 AWS 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部署 AWS CloudFormation 堆栈操作。有关更多信息，请参阅 使用工作流程部署 AWS CloudFormation 堆栈。
2. 在 “部署 AWS 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。有关使用相应权限创建此角色的信息，请参阅步骤 2：创建 AWS 角色。

为 “部署 AWS 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 C I/CD，然后选择工作流程。
4. 选择包含 “部署 AWS CloudFormation 堆栈” 操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。

7. 选择您的部署 AWS 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 ARN 下，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`。您最多可以有五个回滚触发器。

      > **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。在所有堆栈资源部署完毕后开始监控。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 AWS 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 C I/CD，然后选择工作流程。
4. 选择包含 “部署 AWS CloudFormation 堆栈” 操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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，请参见“部署 AWS CloudFormation 堆栈” 操作 YAML 定义。
8. 在 De ploy AWS CloudFormation 堆栈操作的`role-arn`属性中指定的角色中，确保包含CloudWatchFullAccess权限。有关使用相应权限创建此角色的信息，请参阅步骤 2：创建 AWS 角色。

Visual

关闭 “部署 AWS 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 C I/CD，然后选择工作流程。
4. 选择包含 “部署 AWS CloudFormation 堆栈” 操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 选择您的部署 AWS 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. 打开 “禁用回滚”。

YAML

关闭 “部署 AWS 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 C I/CD，然后选择工作流程。
4. 选择包含 “部署 AWS CloudFormation 堆栈” 操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。

7. 在 YAML 代码中添加该`disable-rollback: 1`属性以停止回滚。有关此属性的解释，请参见“部署 AWS CloudFormation 堆栈” 操作 YAML 定义。

“部署 AWS CloudFormation 堆栈” 操作生成的变量

部署 AWS 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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
部署平台	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 例如：`us-west-2`
堆栈 ID	已部署堆栈的亚马逊资源名称 (ARN)。 例如：`arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cfn-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`

“部署 AWS CloudFormation 堆栈” 操作 YAML 定义

以下是 “部署 AWS CloudFormation 堆栈” 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 CloudFormation 堆栈。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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。除非要更改版本，否则请勿更改此属性。有关更多信息，请参阅 指定操作的主版本、次要版本或补丁版本。

默认值：aws/cfn-deploy@v1。

对应的用户界面：工作流程图/ DeployCloudFormationStack _nn/ aws/cfn-deploy @v1 标签

DependsOn

(*DeployCloudFormationStack*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入” 选项卡/ 依赖- 可选

Compute

(*DeployCloudFormationStack*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*DeployCloudFormationStack*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：“配置” 选项卡/高级-可选/计算类型

Fleet

(*DeployCloudFormationStack*/Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：Linux.x86-64.Large，Linux.x86-64.XLarge。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

相应的 UI：“配置” 选项卡/高级-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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ 超时（以分钟为单位）- 可选

Environment

(*DeployCloudFormationStack*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*DeployCloudFormationStack*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

(*DeployCloudFormationStack*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Name

(*DeployCloudFormationStack*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

(*DeployCloudFormationStack*/Environment/Connections/Role)

（必需）

指定 De ploy AWS CloudFormation 堆栈操作用于访问的 IAM 角色的名称 AWS 和 AWS CloudFormation 服务。

确保此角色包含以下策略：

- 以下权限策略：

> **Warning**
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
```

```
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 首次使用该角色时，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：

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

确保已将此角色添加到您空间中的账户连接中。要了解有关向账户连接添加 IAM 角色的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

对应的用户界面：配置选项卡/'环境/账户/角色'/角色

Inputs

(*DeployCloudFormationStack*/Inputs)

（可选）

本Inputs节定义了工作流程运行期间DeployCloudFormationStack所需的数据。

> **Note**
>
> 每个 Deplo y AWS 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（比如源和构件）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，以区分主输入中的文件。有关更多信息，请参阅 示例：引用多个构件中的文件。

相应的 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。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Configuration

(*DeployCloudFormationStack*/Configuration)

（必需）

您可以在其中定义操作的配置属性的部分。

相应的 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 AWS CloudFormation 堆栈指定名称。

对应的用户界面：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。该模板可以采用 JSON 或 YAML 格式，可以位于源存储库、先前操作中的项目或 Amazon S3 存储桶中。如果模板文件位于源存储库或项目中，则该路径是相对于源存储库或项目根目录的路径。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，以指明要在哪个工件或来源中找到它。有关更多信息，请参阅 引用源存储库中的文件 和 在构件中引用文件。

对应的用户界面：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。这些策略取决于您的堆栈中的资源。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。如果您按照中描述的教程进行操作教

程：使用部署无服务器应用程序 AWS CloudFormation，则其中包含一个名为的过程，创建堆栈角色该过程列出了部署典型的无服务器应用程序堆栈时堆栈角色所需的权限。

> **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。

- 以下信任政策：

```
{
    "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关联关联。要了解有关将 IAM 角色与账户关联的更多信息，请参阅向账户连接添加 IAM 角色。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的堆栈角色下拉列表中；但是，仍然可以使用 YAML 编辑器在`role-arn`字段中指定角色 ARN。

> **Note**
>
> 如果你愿意，你可以在这里指定`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色的名称。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。了解该`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性较低的场景中使用此角色。

对应的用户界面：配置选项卡/ 堆栈角色—— 可选

capabilities

(*DeployCloudFormationStack*/Configuration/capabilities)

（必需）

指定允许 AWS CloudFormation 创建特定堆栈所需的 IAM 功能列表。在大多数情况下，您可以保留`capabilities`默认值`CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND`。

如果您`##[error] requires capabilities: [`*capability-name*`]`在 Deploy AWS CloudFormation 堆栈操作的日志中看到，请参阅以获取如何修复 IAM 功能错误？有关如何修复问题的信息。

有关 IAM 功能的更多信息，请参阅 IAM 用户指南中的在 AWS CloudFormation 模板中确认 IAM 资源。

对应的用户界面：“配置” 选项卡/高级/ 功能

parameter-overrides

(*DeployCloudFormationStack*/Configuration/parameter-overrides)

（可选）

在您的 AWS 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。

该`parameter-overrides`物业接受：

- 包含参数和值的 JSON 文件。
- 以逗号分隔的参数和值列表。

指定 JSON 文件

1. 确保 JSON 文件使用以下语法之一：

```
{
  "Parameters": {
    "Param1": "Value1",
    "Param2": "Value2",
```

```
    ...
  }
}
```

或者...

```
[
  {
     "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. 使用以下格式之一指定 JSON 文件的路径：

   - 如果您的 JSON 文件位于先前操作的输出项目中，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。

   - 如果您的 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 文件位于您的源存储库中，位于名为的文件夹中 `my/folder`

```
##My workflow YAML
...
Actions:
  MyDeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      parameter-overrides: file:///sources/WorkflowSource/my/folder/params.json
```

使用以逗号分隔的参数列表

- 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 AWS CloudFormation 模板：

```
##My CloudFormation template

Description: My AWS 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:
...
```

... 你可以按如下方式设置该parameter-overrides属性：

```
##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。

相应的用户界面：

- “配置” 选项卡/高级/ 参数覆盖
- “配置” 选项卡/高级/参数覆盖/ 使用文件指定覆盖
- “配置” 选项卡/高级/参数覆盖/ 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。默认设置是在不停止的情况下创建和运行变更集。有关更多信息，请参阅《AWS CLI 命令参考》中的 deplo AWS 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。

相应的 UI：“配置” 选项卡/高级/未执行更改集

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 AWS 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。）默认情况下，如果更改集为空，则允许操作继续执行，即使堆栈未更新，也会返回一条UPDATE_COMPLETE消息。

有关此设置的更多信息，请参阅《AWS CLI 命令参考》中的 dep AWS CloudFormation loy 参数。有关变更集的更多信息，请参阅《AWS CloudFormation 用户指南》中的使用变更集更新堆栈。

对应的用户界面：配置选项卡/高级/变更集为空时失败

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。回滚会将堆栈返回到上次已知的稳定状态。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 AWS CloudFormation loy 参数。

有关 Depl oy AWS 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。

相应的 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 AWS 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。

对应的用户界面：“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，则由于超时而无法创建堆栈并回滚堆栈。

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 AWS 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 AWS CloudFormation 事件发送一条通知。（事件显示在 AWS CloudFormation 控制台堆栈的 “事件” 选项卡中。）您最多可以指定五个主题。有关更多信息，请参阅 Amazon SNS 是什么？。

相应的 UI：“配置” 选项卡/高级/通知 ARN

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`。您最多可以有五个回滚触发器。

> **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。

相应的 UI：“配置” 选项卡/高级/监控警报 ARN

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。在所有堆栈资源部署完毕后开始监控。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。

对应的用户界面：“配置” 选项卡/高级/ “监视时间”

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，您可以使用它们来标识堆栈，用于成本分配等目的。有关什么是标签以及如何使用标签的更多信息，请参阅《Amazon EC2 用户指南》中的标记资源。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 AWS CloudFormation 堆栈选项。

密钥可以包含字母数字字符或空格，最多可包含 127 个字符。值可以包含字母数字字符或空格，最多可包含 255 个字符。

您可以为每个堆栈添加最多 50 个唯一标签。

对应的用户界面：配置选项卡/高级/标签

使用工作流程部署 AWS Cloud Development Kit (AWS CDK) 应用程序

本节介绍如何使用工作流程将 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程中。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中。 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。

何时使用 “部AWS CDK 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 部署的工作原理如下：

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）下载到构建映像。 CodeCatalyst

   如果您指定了 1.0.13 或更高版本，则该操作与特定版本的 CDK CLI 捆绑在一起，因此不会进行下载。
2. 该操作使用 CDK CLI 来运行该`cdk deploy`命令。此命令将您的 AWS CDK 应用程序合成并部署到中。 AWS有关此命令的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) 开发人员指南》中的 AWS CDK Toolkit（cdk 命令）主题。

“部AWS CDK 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**
>
> 您也许可以覆盖默认值。有关更多信息，请参阅 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

该动作可以部署多少堆栈？

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。

主题

- 部署应用程序的工作流程示例 AWS CDK
- 添加 “部AWS CDK 署” 操作
- “AWS CDK 部署” 操作生成的变量
- “AWS CDK 部署” 动作 YAML 定义

部署应用程序的工作流程示例 AWS CDK

以下示例工作流程包括AWS CDK 部署操作和AWS CDK 引导操作。该工作流由以下按顺序运行的构建块组成：

> **Note**
>
> 以下工作流程示例仅用于说明目的，如果不进行其他配置，则无法运行。它旨在举一个示例，说明使用AWS CDK 部署操作配置工作流程时会是什么样子。

- 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程运行。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程。
- AWS CDK 引导操作 (CDKBootstrap)-触发后，该操作会将CDKToolkit引导堆栈部署到。 AWS如果环境中已经存在CDKToolkit堆栈，则将在必要时对其进行升级；否则，不会发生任何事情，操作将被标记为成功。
- 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 AWS CloudFormation 模板中定义的堆栈部署到中。 AWS

```
Name: codecatalyst-cdk-deploy-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
```

```
      - main
Actions:
  CDKBootstrap:
    Identifier: aws/cdk-bootstrap@v1
    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@v1
    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 “部AWS CDK 署” 操作

按照以下说明将AWS CDK 部署操作添加到您的工作流程中。

开始之前

在将AWS CDK 部署操作添加到工作流程之前，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) 开发人员指南中的如何引导中描述的方法之一。
   - 使用引AWS CDK 导操作。您可以在与AWS CDK 部署相同的工作流程中添加此操作，也可以在不同的工作流程中添加此操作。只需确保引导操作在运行AWS CDK 部署操作之前至少运行一次，以便必要的资源到位即可。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《开发人员指南》中的 Bootstrapping。AWS Cloud Development Kit (AWS CDK)

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索AWS CDK 部署操作，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择AWS CDK 部署。将出现 “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程图中，然后打开其配置窗格。
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” 动作 YAML 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 部署操作因`npm install`错误而失败，如何修复 “npm 安装” 错误？请参阅，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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索AWS CDK 部署操作，然后执行以下任一操作：
   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or
   - 选择AWS CDK 部署。将出现 “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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“AWS CDK 部署” 动作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 部署操作因`npm install`错误而失败，如何修复 “npm 安装” 错误？请参阅，了解有关如何修复错误的信息。

“AWS CDK 部署” 操作生成的变量

AWS CDK 部署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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
堆栈 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)。 例如：`arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-app-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
部署平台	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 例如：`us-west-2`
跳过部署	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变化，则将跳过堆栈部署。 只有当该变量的值为时，才会生成该变量`true`。

键	值
	硬编码为。`true`
AWS 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想要的，则可以使用 `CfnOutputVariables` YAML 属性配置不同的变量。有关更多信息，请参阅中的CfnOutputVariables属性描述“AWS CDK 部署” 动作 YAML 定义。

“AWS CDK 部署” 动作 YAML 定义

以下是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 Cloud Development Kit (AWS CDK) 应用程序。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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@v1
    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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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 指定操作的主版本、次要版本或补丁版本。

默认值：`aws/cdk-deploy@v1`。

对应的用户界面：工作流程图/ CDKDeploy _nn/ aws/cdk-deploy @v1 标签

DependsOn

(*CDKDeploy*/DependsOn)

（可选）

指定必须成功运行才能运行AWS CDK 部署操作的操作或操作组。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@v1
  DependsOn:
    - CDKBootstrap
```

> **Note**
>
> 引导是部署应用程序的必备先决条件。 AWS CDK 如果您未在工作流程中包含 AWS CDK Bootstrap 操作，则在运行部署操作之前，必须找到另一种部署 AWS CDK 引导堆栈的方法 AWS CDK 。有关更多信息，请参阅使用工作流程部署 AWS Cloud Development Kit (AWS CDK) 应用程序中的添加 “部AWS CDK 署” 操作。

有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入” 选项卡/ 依赖- 可选

Compute

(*CDKDeploy*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*CDKDeploy*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：“配置” 选项卡/高级-可选/计算类型

Fleet

(*CDKDeploy*/Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

相应的 UI：“配置” 选项卡/高级-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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Inputs

(*CDKDeploy*/Inputs)

（可选）

本Inputs节定义了工作流程运行期间CDKDeploy所需的数据。

> **Note**
>
> 每个AWS CDK 部署操作只允许输入一个输入（源或构件）。

相应的 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。在开始AWS CDK 部署过程之前，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。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输入” 选项卡/ 工件- 可选

Outputs

(*CDKDeploy*/Outputs)

（可选）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts - output

(*CDKDeploy*/Outputs/Artifacts

（可选）

指定操作生成的对象。您可以在其他操作中引用这些构件作为输入。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/ 工件

Name

(*CDKDeploy*/Outputs/Artifacts/Name)

（如果包含Artifacts - output，则为必填项）

指定将包含在运行时由AWS CDK 部署操作合成的 AWS 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，但不会将其保存在构件中。考虑将合成后的模板保存在工件中，以保留其记录，用于测试或故障排除。

对应的用户界面：输出选项卡/构件/添加构件/构建构件名称

Files

(*CDKDeploy*/Outputs/Artifacts/Files)

（如果包含Artifacts - output，则为必填项）

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 AWS CloudFormation 模板。

> **Note**
>
> `cdk.out`是保存合成文件的默认目录。如果您指定的输出目录不是在`cdk.json`文件`cdk.out`中，请在此处指定该目录，而不是`cdk.out`。

相应的用户界面：输出选项卡/构件/添加工件/构建生成的文件

Environment

(*CDKDeploy*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*CDKDeploy*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

(*CDKDeploy*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Name

(*CDKDeploy*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

(*CDKDeploy*/Environment/Connections/Role)

（必需）

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。请确保此角色包括：

- 以下权限策略：

> **Warning**
>
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "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::aws-account:role/cdk-*"
        }
    ]
}
```

- 以下自定义信任策略：

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

确保已将此角色添加到您的账户关联中。要了解有关向账户连接添加 IAM 角色的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

对应的用户界面：配置选项卡/'环境/账户/角色'/角色

Configuration

(*CDKDeploy*/Configuration)

（必需）

您可以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，堆栈名称以*##*斜体突出显示。如果你的入口点文件使用的是不同的语言，它看起来会很相似。

```
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来部署所有堆栈。

对应的用户界面：配置选项卡/ 堆栈名称

Region

(*CDKDeploy*/Configuration/Region)

（必需）

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。

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。标签适用于堆栈本身以及堆栈中的单个资源。有关标记的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) 开发人员指南》中的添加标签。

对应的用户界面：配置选项卡/高级-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) 开发人员指南》中的运行时上下文。

对应的用户界面：配置选项卡/高级-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，以及引导操作的 1.0.8 或更高版本。AWS CDK

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`。在构建和部署应用程序时，请考虑指定完整版本以确保一致性和稳定性。

  Or

- `latest`。 考虑指定`latest`以利用 CDK CLI 的最新功能和修复程序。

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

有关您可以使用的支持 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：

- “部AWS CDK 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，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。

对应的用户界面：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`结构作为工作流输出变量公开。然后，您可以在工作流程的后续操作中引用工作流程输出变量。有关中变量的更多信息 CodeCatalyst，请参阅在工作流程中配置和使用变量。

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, 'my-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'bucketName', {
      value: bucket.bucketName,
      description: 'The name of the s3 bucket',
      exportName: 'myBucket',
    });
    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',
    });
    ...
  }
}
```

... 你的CfnOutputVariables房产看起来像这样：

```
Configuration:
  ...
  CfnOutputVariables: '["bucketName","tableName"]'
```

... 然后该操作会生成以下工作流程输出变量：

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” 操作生成的变量 。

> **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 AWS 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/ AWS 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 AWS 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。当该`--app`选项存在时，`cdk synth`操作不会发生。

如果您未指定云程序集目录，则AWS CDK 部署操作将运行不带该`--app`选项的`cdk deploy`命令。如果没有该`--app`选项，该`cdk deploy`操作将同时合成 (`cdk synth`) 并将您的 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组件指定为：

- 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其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关系的云程序集，在生产阶段使用另一组依赖关系（如果这些依赖关系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的精确对等，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）来指定已经合成的云程序集。
- 在 AWS CDK 应用程序中使用非标准包管理器和工具

  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。如果操作无法使用这些工具成功运行您的应用程序，则合成将不会发生，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）中指定它。

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《开发人员指南》中的部署堆栈、合成堆栈和跳过合成。AWS Cloud Development Kit (AWS CDK)

有关云程序集的信息，请参阅《AWS Cloud Development Kit (AWS CDK) API 参考》中的 Cloud Assembly。

对应的 UI：“配置” 选项卡/ 云端装配目录路径

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程中。AWS CDK bootstrap 操作使用现代模板在您的 AWS 环境中配置引导堆栈。如果引导堆栈已经存在，则该操作将在必要时对其进行更新。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《开发人员指南》中的引导。AWS Cloud Development Kit (AWS CDK)

何时使用 “AWS CDK 引导” 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@@ 勿使用此操作：

- 您已经使用另一种机制部署了引导堆栈，并且希望保持其完好无损（无更新）。
- 您想使用自定义引导模板，但引AWS CDK 导操作不支持该模板。

“AWS CDK bootstrap” 操作的工作原理

引AWS CDK 导程序的工作原理如下：

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，则该操作与特定版本的 CDK CLI 捆绑在一起，因此不会进行下载。
2. 该操作使用 CDK CLI 来运行该`cdk bootstrap`命令。此命令执行《开发人员指南》的 Bootstrappin g 主题中描述的引导任务。AWS Cloud Development Kit (AWS CDK)

“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**
>
> 您也许可以覆盖默认值。有关更多信息，请参阅 中的 CdkCliVersion。

“AWS CDK bootstrap” 操作版本	AWS CDK CLI 版本
1.0.0 — 1.0.7	最新
1.0.8 或更高版本	2.99.1

主题

引导应用程序的工作流程示例 AWS CDK

有关包含AWS CDK 引导操作的工作流程，请参阅部署应用程序的工作流程示例 AWS CDK中的。使用工作流程部署 AWS Cloud Development Kit (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程中。

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。您可以使用支持的任何编程语言编写应用程序 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

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索AWS CDK 引导操作，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择AWS CDK 引导。出现操作详细信息对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
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 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 引导操作因`npm install`错误而失败，如何修复 “npm 安装” 错误？请参阅，了解有关如何修复错误的信息。

YAML

使用 YAML 编辑器添加 “AWS CDK 引导” 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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
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 + 将其添加到工作流程图并打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“AWS CDK 引导” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 引导操作因`npm install`错误而失败，如何修复 “npm 安装” 错误？请参阅，了解有关如何修复错误的信息。

“AWS CDK bootstrap” 操作生成的变量

AWS CDK bootstrap 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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
部署平台	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 AWS CDK 引导堆栈的区域代码。 AWS 区域 例如：`us-west-2`
堆栈 ID	已部署的 AWS CDK 引导堆栈的 Amazon 资源名称 (ARN)。

键	值
	例如：arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-bootstrap-stack/6aad4380-100a-11ec-a10a-03b8a84d40df
跳过部署	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变化，则将跳过堆栈部署。 只有当该变量的值为时，才会生成该变量true。 硬编码为。true

“AWS CDK 引导” 操作 YAML 定义

以下是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。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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@v1
    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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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。除非要更改版本，否则请勿更改此属性。有关更多信息，请参阅 指定操作的主版本、次要版本或补丁版本。

默认值：aws/cdk-bootstrap@v1。

对应的用户界面：工作流程图/ CDKBootstrapAction _nn/ aws/cdk-bootstrap @v1 标签

DependsOn

(*CDKBootstrapAction*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入” 选项卡/ 依赖- 可选

Compute

(*CDKBootstrapAction*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*CDKBootstrapAction*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

优化了动作启动速度。

有关计算类型的更多信息，请参阅计算类型。

相应的 UI：“配置” 选项卡/高级-可选/计算类型

Fleet

(*CDKBootstrapAction*/Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，`Fleet`则默认为`Linux.x86-64.Large`。

相应的 UI：“配置” 选项卡/高级-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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Inputs

(*CDKBootstrapAction*/Inputs)

（可选）

本`Inputs`节定义了工作流程运行期间AWS CDK 引导操作所需的数据。

相应的 UI：“输入” 选项卡

> **Note**
>
> 每个AWS CDK 引导操作只允许输入一个输入（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AWS CDK 启动引导过程之前，bootstrap 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。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输入” 选项卡/ 工件- 可选

Outputs

(*CDKBootstrapAction*/Outputs)

（可选）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts - output

(*CDKBootstrapAction*/Outputs/Artifacts)

（可选）

指定操作生成的对象。您可以在其他操作中引用这些构件作为输入。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/ 工件

Name

(*CDKBootstrapAction*/Outputs/Artifacts/Name)

（如果包含Artifacts - output，则为必填项）

指定将包含在运行时由AWS CDK 引导操作合成的 AWS 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，但不会将其保存在构件中。考虑将合成后的模板保存在工件中，以保留其记录，用于测试或故障排除。

对应的用户界面：输出选项卡/构件/添加构件/构建构件名称

Files

(*CDKBootstrapAction*/Outputs/Artifacts/Files)

（如果包含Artifacts - output，则为必填项）

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 AWS CloudFormation 模板。

> **Note**
>
> `cdk.out`是保存合成文件的默认目录。如果您指定的输出目录不是在`cdk.json`文件`cdk.out`中，请在此处指定该目录，而不是`cdk.out`。

相应的用户界面：输出选项卡/构件/添加工件/构建生成的文件

Environment

(*CDKBootstrapAction*/Environment)

（必需）

指定要用于操作的 CodeCatalyst 环境。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*CDKBootstrapAction*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

(*CDKBootstrapAction*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Name

(*CDKBootstrapAction*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

(*CDKBootstrapAction*/Environment/Connections/Role)

（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。请确保此角色包含以下策略：

> **Note**
>
> 以下权限策略中显示的权限是`cdk bootstrap`命令执行引导所需的权限。如果更改其 bootstrap 命令，这些权限可能会 AWS CDK 发生变化。

> **Warning**
>
> 仅在AWS CDK 引导操作中使用此角色。它非常宽松，将其与其他操作一起使用可能会带来安全风险。

- 以下权限策略：

  > **Warning**
  >
  > 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "ssm:GetParameterHistory",
                "ecr:PutImageScanningConfiguration",
                "cloudformation:*",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "ssm:GetParameters",
                "iam:PutRolePolicy",
                "ssm:GetParameter",
                "ssm:DeleteParameters",
                "ecr:DeleteRepository",
```

```
                "ssm:PutParameter",
                "ssm:DeleteParameter",
                "iam:PassRole",
                "ecr:SetRepositoryPolicy",
                "ssm:GetParametersByPath",
                "ecr:DescribeRepositories",
                "ecr:GetLifecyclePolicy"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:aws-account:parameter/cdk-bootstrap/*",
                "arn:aws:cloudformation:aws-region:aws-account:stack/CDKToolkit/*",
                "arn:aws:ecr:aws-region:aws-account:repository/cdk-*",
                "arn:aws:iam::aws-account:role/cdk-*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:CreateUploadBucket",
                "cloudformation:ListExports",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:SetTypeDefaultVersion",
                "cloudformation:RegisterPublisher",
                "cloudformation:ActivateType",
                "cloudformation:ListTypes",
                "cloudformation:DeactivateType",
                "cloudformation:SetTypeConfiguration",
                "cloudformation:DeregisterType",
                "cloudformation:ListTypeRegistrations",
                "cloudformation:EstimateTemplateCost",
                "cloudformation:DescribeAccountLimits",
                "cloudformation:BatchDescribeTypeConfigurations",
                "cloudformation:CreateStackSet",
                "cloudformation:ListStacks",
                "cloudformation:DescribeType",
                "cloudformation:ListImports",
                "s3:*",
                "cloudformation:PublishType",
                "ecr:CreateRepository",
                "cloudformation:DescribePublisher",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:TestType",
```

```
                "cloudformation:ValidateTemplate",
                "cloudformation:ListTypeVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色时，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：

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

确保已将此角色添加到您的账户关联中。要了解有关向账户连接添加 IAM 角色的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

对应的用户界面：配置选项卡/'环境/账户/角色'/角色

Configuration

(*CDKBootstrapAction*/Configuration)

（必需）

您可以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 AWS 区域 将引导堆栈部署到哪里。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，以及引导操作的 1.0.8 或更高版本。AWS CDK

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`。在构建和部署应用程序时，请考虑指定完整版本以确保一致性和稳定性。

  Or

- `latest`。 考虑指定`latest`以利用 CDK CLI 的最新功能和修复程序。

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

有关您可以使用的支持 CDK CLI 版本的列表，请参阅AWS CDK 版本。

如果省略此属性，则操作将使用以下主题之一中描述的默认 AWS CDK CLI 版本：

- “部AWS CDK 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

使用工作流程将文件发布到 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 发布操作添加到您的工作流程中。Amazon S3 发布操作将文件从源目录复制到 Amazon S3 存储桶。源目录可以位于：

- 源存储库，或
- 由另一个工作流程操作生成的输出对象

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 中的文件的工作流程。

  例如，您可能有一个工作流程来构建要托管在 Amazon S3 中的静态网站。在这种情况下，您的工作流程将包括用于构建网站的 HTML 和支持文件的构建操作，以及用于将文件复制到 A mazon S3 的 Amazon S3 发布操作。

- 您有一个源存储库，其中包含要存储在 Amazon S3 中的文件。

  例如，您可能有一个包含应用程序源文件的源存储库，您想每晚将其存档到 Amazon S3。

主题

向 Amazon S3 发布文件的工作流程示例

以下示例工作流程包括 Amazon S3 发布操作和构建操作。该工作流程会构建一个静态文档网站，然后将其发布到托管该网站的 Amazon S3 上。该工作流由以下按顺序运行的构建块组成：

> **Note**
>
> 以下工作流程示例仅用于说明目的，仅适用于其他配置。

- 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程运行。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程。
- 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的构件中MyDocsSite。有关生成操作的更多信息，请参阅使用工作流程进行构建。
- A mazon S3 发布操作 (PublishToS3) — 构建操作完成后，此操作会将MyDocsSite项目中的网站复制到 Amazon S3 进行托管。

```
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: my-bucket
      SourcePath: /artifacts/PublishToS3/MyDocSite/site
      TargetPath: my/docs/site
```

添加 “Amazon S3 发布” 操作

按照以下说明将 Amazon S3 发布操作添加到您的工作流程中。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 A mazon S3 发布操作，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择 Amazon S3 发布。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
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“亚马逊 S3 发布” 操作 YAML 定义。本参考提供了在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。

8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索 A mazon S3 发布操作，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择 Amazon S3 发布。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“亚马逊 S3 发布” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
12. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

“亚马逊 S3 发布” 操作 YAML 定义

以下是 Amazon S3 发布操作的 YAM L 定义。要了解如何使用此操作，请参阅使用工作流程将文件发布到 Amazon S3。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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: s3-bucket-name
      TargetPath: my/target
```

S3Publish

（必需）

指定操作的名称。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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 指定操作的主版本、次要版本或补丁版本。

默认值：aws/s3-publish@v1。

对应的用户界面：工作流程图/ S3Publish _nn/ aws/s3-publish @v1 标签

DependsOn

(*S3Publish*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关“依赖”功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入”选项卡/ 依赖- 可选

Compute

(*S3Publish*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*S3Publish*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

优化了动作启动速度。

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程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，`Fleet`则默认为`Linux.x86-64.Large`。

对应的 UI：“配置” 选项卡/ “计算舰队”

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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Inputs

(*S3Publish*/Inputs)

（可选）

本`Inputs`节定义了工作流程运行期间`S3Publish`所需的数据。

> **Note**
>
> 每个AWS CDK 部署操作最多允许四个输入（一个源和三个工件）。变量不计入此总数。

如果您需要引用驻留在不同输入（比如源和构件）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，以区分主输入中的文件。有关更多信息，请参阅 示例：引用多个构件中的文件。

相应的 UI：“输入” 选项卡

Sources

(*S3Publish*/Inputs/Sources)

（如果您要发布到 Amazon S3 的文件存储在源存储库中，则为必填项）

如果您要发布到 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`。

如果您要发布到 Amazon S3 的文件不包含在源存储库中，则它们必须位于由其他操作生成的项目中。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 UI：“输入” 选项卡/ “来源”- 可选

Artifacts - input

(*S3Publish*/Inputs/Artifacts)

（如果您要发布到 Amazon S3 的文件存储在先前操作的输出项目中，则为必填项）

如果您要发布到 Amazon S3 的文件包含在先前操作生成的项目中，请在此处指定该项目。如果您的文件不包含在构件中，则它们必须位于您的源存储库中。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Variables - input

(*S3Publish*/Inputs/Variables)

（可选）

指定一系列名称/值对，这些对定义要用于操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的 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。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/环境

Name

(*S3Publish*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/环境

Connections

(*S3Publish*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接`Environment`。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/连接

Name

(*S3Publish*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/连接

Role

(*S3Publish*/Environment/Connections/Role)

（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。请确保此角色包括：

- 以下权限策略：

> **Warning**
>
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "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：

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

请确保此角色与您的账户关联关联。要了解有关将 IAM 角色与账户关联的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/角色

Configuration

(*S3Publish*/Configuration)

（必需）

您可以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。该目录或文件可以位于源存储库或先前操作中的对象中，并且是相对于源存储库或项目根目录的。

示例：

指定会`./myFolder/`将的内容复制`/myFolder`到 Amazon S3，并保留底层目录结构。

仅指定`myfile.txt`到 Amazon S3 的`./myFolder/myfile.txt`副本。（目录结构已删除。）

不能使用通配符。

> **Note**
>
> 您可能需要在目录或文件路径中添加前缀，以指明要在哪个工件或来源中找到它。有关更多信息，请参阅 引用源存储库中的文件 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）

指定您要在其中发布文件的 Amazon S3 存储桶的名称。

对应的用户界面：配置选项卡/ 目标存储桶- 可选

TargetPath

(*S3Publish*/Configuration/TargetPath)

（可选）

在 Amazon S3 中指定要在其中发布文件的目录的名称和路径。如果该目录不存在，则会创建该目录。目录路径不得包含存储桶名称。

示例：

```
myS3Folder
```

```
./myS3Folder/myS3Subfolder
```

相应的 UI：“配置” 选项卡/ 目标目录- 可选

部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC

您可以通过 CodeCatalyst 工作流程操作将您的应用程序和其他资源部署到您的 AWS 账户 或 Amazon VPC 中。为此，必须设置 CodeCatalyst 环境。

不要将环境与开发环境混淆，它是代码部署到的地方。它通常包含正在运行的应用程序的实例及其关联的基础架构。您可以为环境命名，例如开发、测试、暂存或生产。 CodeCatalyst 对环境生成的任何部署都将显示在 “环境” 页面上。要设置环境，请为其命名（例如）`my-production-environment`，然后将其与您的关联 AWS 账户。

除了显示部署信息外，环境还可以作为向工作流程操作分配 AWS IAM 角色的机制。

单个工作流程中能否存在多个环境？

是。如果工作流程包含多个操作，则可以为每个操作分配一个环境。例如，您的工作流程可能包含两个部署操作，其中一个被分配一个`my-staging-enviroment`环境，另一个分配一个`my-production-environment`环境。

哪些操作支持环境？

以下操作支持在 “环境” 页面上显示其部署信息：

- 部署 AWS CloudFormation 堆栈-有关更多信息，请参阅 使用工作流程部署 AWS CloudFormation 堆栈
- 部署到 Amazon ECS — 有关更多信息，请参阅 使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)

- 部署到 Kubernetes 集群 — 有关更多信息，请参阅 使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service
- AWS CDK 部署-有关更多信息，请参阅 使用工作流程部署 AWS Cloud Development Kit (AWS CDK) 应用程序

> **Note**
>
> 如果您想允许某项操作在您的 AWS 账户中访问和执行操作，则需要将其与环境相关联。许多操作都支持环境关联，包括但不限于前面列出的操作。您可以分辨出哪些操作支持环境关联，因为它们将在可视化编辑器的 “配置” 选项卡上包含一个 “环境” 下拉列表。

支持的区域

环境页面可以显示任何 AWS 区域的资源。

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。

主题

- 创建环境
- 将环境、账户连接和 IAM 角色与工作流程操作关联
- 将 VPC 连接与环境关联
- 将 AWS 账户 与环境关联

创建环境

按照以下说明创建一个空环境，您可以稍后将其与工作流程操作相关联。

开始前的准备工作

你需要以下内容：

- 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。每个环境最多只能使用一个账户连接。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。

> **Note**
>
> 您可以在没有账户连接的情况下创建环境；但是，您需要稍后再回来添加连接。

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，选择以下选项之一：

   - 非生产 — 在将应用程序投入生产之前，您可以测试应用程序以确保其按预期运行的环境。
   - P@@ ro duction — 一个 “实时” 环境，可公开使用，用于托管您最终确定的应用程序。

   如果您选择 Production，则用户界面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 VPC 连接中，选择要与此环境关联的 VPC 连接。有关创建此 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。
7. （可选）在描述中，输入描述，例如**`Production environment for the hello-world app`**。
8. 在 “连接-可选” 中，选择要与此环境关联的 AWS 账户连接。确保账户连接包含您要与环境关联的 IAM 角色。有关创建此连接的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。
9. 选择 “创建环境”。 CodeCatalyst 创建一个空环境。

后续步骤

- 现在，您已经创建了一个环境，可以将其与工作流程操作相关联了。有关更多信息，请参阅 将环境、账户连接和 IAM 角色与工作流程操作关联。

将环境、账户连接和 IAM 角色与工作流程操作关联

当您将环境、账户连接和 IAM 角色与支持的工作流程操作关联时，该操作即可使用该 IAM 角色。除了获得 IAM 角色的访问权限外，该操作还可能将其部署信息导入到环境页面中。有关更多信息，请参阅哪些操作支持环境？。

按照以下说明将环境、账户连接和 IAM 角色与操作相关联。

步骤 1：将环境、账户连接和角色关联到工作流程操作

使用以下步骤将环境、账户连接和角色与工作流程操作相关联。

Visual

使用可视化编辑器将环境、账户连接和角色与工作流程操作相关联

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择环境支持的操作。有关更多信息，请参阅 哪些操作支持环境？。
8. 选择 “配置” 选项卡，并在字段中指定信息，如下所示。

   环境

   指定要用于操作的 CodeCatalyst 环境。

   有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

   账户连接或连接-可选（以可用者为准）

   指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接`Environment`。

   有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

角色

指定此操作用于访问和操作 Amazon S3 和 Amazon ECR 等 AWS 服务的 IAM 角色的名称。确保将此角色添加到您的账户关联中。要向账户连接添加 IAM 角色，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 只要角色具有足够的权限，您就可以在此处指定该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

如果您在列表中看不到该角色，那是因为您尚未将其与账户关联关联。有关更多信息，请参阅向账户连接添加 IAM 角色。

9. （可选）选择 “验证” 以在提交之前验证工作流程的 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 编辑器将环境、账户连接和角色与工作流程操作相关联

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要与环境关联的工作流程操作中，添加类似于以下内容的代码：

```
action-name
```

```
  Environment:
    Name: environment-name
    Connections:
      - Name: account-connection-name
        Role: iam-role-name
```

有关更多信息，工作流程 YAML 定义请参阅您的操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

步骤 2：填充环境

将环境、账户连接和角色与工作流程操作关联后，可以在环境页面中填充部署信息。按照以下说明填充 “环境” 页面。

> **Note**
>
> 只有一部分工作流操作支持 “环境” 页面。有关更多信息，请参阅 哪些操作支持环境？。

填充环境

1. 如果您在中提交更改时工作流程运行未自动启动步骤 1：将环境、账户连接和角色关联到工作流程操作，请按如下方式手动启动运行：
   a. 在导航窗格中，选择 C I/CD，然后选择工作流程。
   b. 选择要开始运行的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
   c. 选择运行。

   工作流程运行会启动新的部署，这会 CodeCatalyst 导致在 “环境” 下添加您的应用程序资源信息。
2. 验证您的应用程序资源是否出现在您的环境下：
   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，并验证部署的状态是否为 “成功”。这表示工作流程运行成功部署了您的应用程序资源。

d. 选择 “部署目标” 选项卡，并确认您的应用程序资源已显示。

将 VPC 连接与环境关联

在具有 VPC 连接的环境中配置操作时，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 VPC 指定的资源。一个或多个环境可以使用同一 VPC 连接。

按照以下说明将 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，选择所需的 VPC 连接，然后选择确认。这会将您选择的 VPC 连接与此环境相关联。

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。

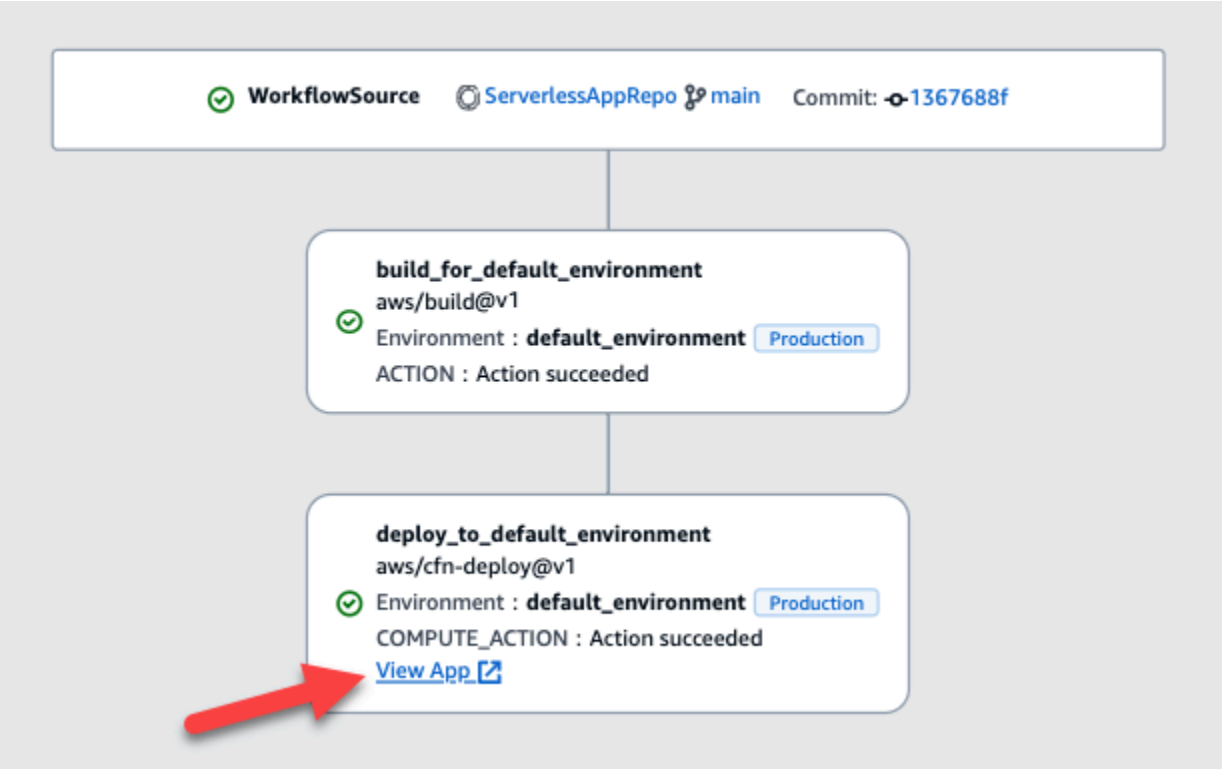
将 AWS 账户 与环境关联

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 “环境属性” 选项卡。
6. 选择 “关联” AWS 账户，选择您想要的 AWS 账户，然后选择 “助理”。这会将您选择 AWS 账户 的环境与该环境相关联。

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。

在工作流程图中显示已部署应用程序的 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。以下工作流程图显示了出现在操作底部的 View App 网址。



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网址。

要启用此功能，请在操作中添加一个输出变量，其名称应包含`appurl`、或`endpointurl`。您可以使用带或不带连接破折号 (-)、下划线 (_) 或空格 ( ) 的名称。该字符串不区分大小写。将变量的值设置为已部署应用程序的`http`或 `https` URL。

> **Note**
>
> 如果您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，请参阅以下过程之一：

- 在 “AWS CDK 部署” 操作中显示应用程序网址
- 在 “部署 AWS 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网址

配置完网址后，请按照以下说明验证它是否按预期显示：

- 验证应用程序 URL 是否已添加

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）：

   - 密钥名称必须包含`appurl`、或`endpointurl`、带或不带连接短划线 (-)、下划线 (_) 或空格 ( )。该字符串不区分大小写。
   - 该值必须是已部署应用程序的`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, 'my-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. 继续执行验证应用程序 URL 是否已添加。

在 “部署 AWS CloudFormation 堆栈” 操作中显示应用程序 URL

1. 如果您使用的是 De ploy AWS CloudFormation stack 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：

   - 密钥（也称为逻辑 ID）必须包含appurl、或endpointurl、带或不带连接破折号 (-)、下划线 (_) 或空格 ( )。该字符串不区分大小写。
   - 该值必须是已部署应用程序的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. 继续执行验证应用程序 URL 是否已添加。

在所有其他操作中显示应用程序网址

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部分定义环境变量。变量必须具有以下特征：

   - name必须包含appurl、或endpointurl、带或不带连接破折号 (-)、下划线 (_) 或空格 ( )。该字符串不区分大小写。
   - 该值必须是已部署应用程序的http或 https URL。

   例如，生成操作可能如下所示：

   ```
   Build-action:
     Identifier: aws/build@v1
     Inputs:
       Variables:
         - Name: APP-URL
           Value: https://mycompany.myapp.com
   ```

   ... 或者这个：

   ```
   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，请参见定义变量。

2. 导出变量。

   例如，您的构建操作可能如下所示：

   ```
   Build-action:
     ...
     Outputs:
       Variables:
         - APP-URL
   ```

   有关导出变量的信息，请参见导出变量以便其他操作可以使用它。

3. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。

4. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

5. 继续执行验证应用程序 URL 是否已添加。

验证应用程序 URL 是否已添加

- 如果工作流程尚未自动启动，则启动该运行。新运行的应用程序网址应在其工作流程图中显示为可点击的链接。有关开始运行的更多信息，请参阅启动工作流程手动运行。

移除部署目标

您可以从 CodeCatalyst 控制台的 “环境” 页面中移除部署目标，例如 Amazon ECS 集群或 AWS 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：

- 您删除了部署到目标的工作流程。
- 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，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。
5. 选择 “部署目标” 选项卡。
6. 选择要移除的部署目标旁边的单选按钮。
7. 选择移除。

   目标已从页面中移除。

通过提交跟踪部署状态

在开发生命周期的任何时候，都必须了解特定提交的部署状态，例如错误修复、新功能或其他有影响力的更改。考虑以下场景，在这些场景中，部署状态跟踪功能对开发团队很有帮助：

- 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。
- 作为发布经理，你需要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：

- 提交清单。
- 包含提交的部署的状态。
- 成功部署提交的环境。
- 针对您的 CI/CD 工作流程中的提交运行的任何测试的状态。

以下过程详细介绍了如何导航到该视图并使用该视图来跟踪项目中的更改。

> **Note**
>
> 只有CodeCatalyst 存储库才支持通过提交跟踪部署状态。您不能在GitHub 或 Bitbucket 存储库中使用此功能。

通过提交来跟踪部署状态

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”。

   此时将显示提交列表。

   对于每次提交，您都可以查看以下内容：

- 提交信息，例如 ID、作者、消息以及提交时间。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。
- 每个环境的部署状态。有关更多信息，请参阅 部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC。
- 测试和代码覆盖率结果。有关更多信息，请参阅 使用工作流程进行测试。

> **Note**
>
> 不显示软件组成分析 (SCA) 结果。

6. （可选）要查看有关与特定提交相关的更改的更多信息，包括最新部署、详细的代码覆盖率和单元测试信息，请选择查看该提交的详细信息。

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。

您可以从工作流程或环境开始查看日志。

从工作流程开始查看部署操作的日志

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择 “运行”。
6. 选择部署应用程序的工作流程运行。
7. 在工作流程图中，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。

从环境开始查看部署操作的日志

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. 选择您的应用程序的部署环境。
5. 在 “部署” 活动中，找到 “工作流程运行 ID” 列，然后选择部署堆栈的工作流程运行。
6. 在工作流程图中，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。

查看部署状态、提交和拉取请求

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。

从工作流程开始查看部署信息

- 转到部署应用程序的工作流程运行。有关说明，请参阅查看工作流程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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 在工作流程图中，选择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，了解特定于操作的部署信息。

创建工作流

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

使用以下过程在中创建工作流 CodeCatalyst。

有关工作流的更多信息，请参阅使用中的工作流程构建、测试和部署 CodeCatalyst。

Visual

使用可视化编辑器创建工作流程

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 C I/CD，然后选择工作流程。
4. 选择 “创建工作流程”。

   将出现 “创建工作流程” 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。该文件将存储在所选存储库的`~/.codecatalyst/workflows/`文件夹中。如果不存在源存储库，请创建一个。
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 “视觉”。
9. 构建工作流程：

   a. （可选）在工作流程图中，选择 “源和触发器” 框。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 添加推送、拉动或调度触发器。

   b. 选择 + 操作（左上角）。将出现 “操作” 目录。

   c. 选择动作内的加号 (+)，将其添加到工作流程中。使用右侧的窗格配置操作。有关更多信息，请参阅 向 CodeCatalyst 工作流程添加操作。

   d. （可选）选择 “工作流属性”（右上角）。将出现 “工作流程” 属性窗格。配置工作流程名称、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 为工作流程配置计算和运行时环境 Docker 镜像。
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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. 对于工作流程文件名，请保留默认名称或输入您自己的名称。

    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 “创建工作流程” 对话框中指定的存储库和分支。如果你愿意，你可以立即更改存储库和分支。

> **Note**
>
> 提交工作流程定义文件后，该文件无法与其他存储库或分支关联，因此请务必谨慎选择它们。

d. 选择 “提交” 以提交工作流定义文件。

YAML

使用 YAML 编辑器创建工作流程

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 C I/CD，然后选择工作流程。
4. 选择 “创建工作流程”。

   将出现 “创建工作流程” 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。该文件将存储在所选存储库的`~/.codecatalyst/workflows/`文件夹中。如果不存在源存储库，请创建一个。
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. 构建工作流程：

   a. （可选）在 YAML 代码中添加触发器。有关更多信息，请参阅 添加推送、拉动或调度触发器。

   b. 选择 + 操作（左上角）。将出现 “操作” 目录。

   c. 选择动作内的加号 (+)，将其添加到工作流程中。使用右侧的窗格配置操作。有关更多信息，请参阅 向 CodeCatalyst 工作流程添加操作。

   d. （可选）选择 “工作流属性”（右上角）。将出现 “工作流程” 属性窗格。配置工作流程名称、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 为工作流程配置计算和运行时环境 Docker 镜像。
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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. 对于工作流程文件名，请保留默认名称或输入您自己的名称。

    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 “创建工作流程” 对话框中指定的存储库和分支。如果你愿意，你可以立即更改存储库和分支。

    > **Note**
    >
    > 提交工作流程定义文件后，该文件无法与其他存储库或分支关联，因此请务必谨慎选择它们。

    d. 选择 “提交” 以提交工作流定义文件。

运行工作流

运行是工作流程的单次迭代。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，也可以通过工作流程触发器自动启动运行。工作流程触发器的一个例子可能是软件开发人员将提交推送到你的主分支。

如果您错误地启动了工作流程，也可以在工作流程的中途手动停止该运行。

如果大约在同一时间启动多个工作流程运行，则可以配置这些运行的排队方式。你可以使用默认的排队行为，即跑步按开始顺序一个接一个地排队，也可以让较晚的运行取代（或 “接管”），以加快整个跑步的速度。也可以将您的工作流程设置为并行运行，这样就不会等待任何其他运行了。

手动或自动启动工作流程运行后，您可以查看运行状态和其他详细信息。例如，你可以看到它是何时启动的、由谁启动的，以及它是否还在运行。

主题

- 配置运行的排队行为
- 在工作流程运行之间缓存文件
- 查看工作流程运行状态和详细信息

启动工作流程手动运行

使用以下步骤手动启动工作流程运行。

> **Note**
>
> 您也可以通过配置触发器自动启动工作流程运行。

要启动工作流程，请手动运行

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 C I/CD，然后选择工作流程。
4. 选择要运行的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择 Run（运行）。

使用触发器自动启动工作流程

工作流程触发器，或者简称触发器，允许您在发生某些事件（例如代码推送）时自动启动工作流程运行。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导致工作流程运行启动。
- 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导致工作流程运行启动。
- 计划-计划触发器使工作流程按您定义的计划启动。可以考虑使用计划触发器来运行软件的夜间版本，以便软件开发人员在第二天早上准备好最新版本。

您可以单独使用推送、拉取请求和调度触发器，也可以在同一个工作流程中组合使用。

> **Tip**
>
> 要查看触发器的运行情况，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程。在蓝图的工作流程定义文件中查找该`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

常见的触发器配置

本节介绍如何为常见的软件发布和分支策略设置触发器。

软件发布和分支策略：

- 源存储库中有应用程序代码。
- 您的`main`分支包含已完成的代码，这些代码始终可以发布。
- 您的软件开发人员在分支之外的功能分支中进行`main`更改。
- 您的软件开发人员会创建一个拉取请求，要求在功能准备就绪`main`时将其功能分支合并到该分支中。

  您希望此拉取请求自动启动工作流程，该工作流程使用软件开发人员功能分支上的文件构建和测试应用程序，但不部署应用程序。
- 你们的软件开发人员会检查版本和测试，以确保一切看起来都很好。然后，他们将拉取请求合并到`main`分支中。

  您希望合并自动启动构建和部署应用程序代码的工作流程。

建议的工作流程/触发器配置：

鉴于前面概述的软件分支策略，您可能需要使用两个工作流程：

- 当创建或修改拉取请求时，Workflo@@ w 1 会生成和测试您的应用程序。
- 合并拉取请求后，Workflo@@ w 2 会生成和部署您的应用程序。

工作流程 1 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
```

每当软件开发人员创建拉取请求（或修改拉取请求），要求将其功能分支合并到分支时，先前的触发代码就会自动启动工作流程运行。main CodeCatalyst 使用源分支（即开发者的功能分支）中的代码启动工作流程运行。该工作流程构建和部署应用程序。

工作流程 2 如下所示：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildAction:
    instructions-for-building-the-app
  DeployAction:
    instructions-for-deploying-the-app
```

在前面的触发器代码中，当main发生合并时，PUSH触发器被激活。 CodeCatalyst 使用main分支中的代码（现在包含拉取请求中的代码）启动工作流程运行。该工作流程构建和部署应用程序。

有关向工作流程定义文件添加触发器的说明，请参阅添加推送、拉动或调度触发器。

有关触发器的更多示例和其他说明，请参阅触发器示例。

分支时的触发注意事项

本节介绍在设置包含分支的触发器时需要考虑的一些主要注意事项。

- 注意事项 1：对于推送和拉取请求触发器，如果要指定分支，则必须在触发器配置中指定目标（或“收件人”）分支。切勿指定源（或“来自”）分支。

  在以下示例中，从任何分支推送到都会main激活工作流程。

  ```
  Triggers:
    - Type: PUSH
      Branches:
        - main
  ```

  在以下示例中，来自任何分支的拉取请求都会main激活工作流程。

  ```
  Triggers:
    - Type: PULLREQUEST
      Branches:
        - main
      Events:
        - OPEN
        - REVISION
  ```

- 注意事项 2：对于推送触发器，激活工作流程后，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。
- 注意事项 3：对于拉取请求触发器，激活工作流程后，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）。
- 注意事项 4：一个分支中完全相同的触发器可能无法在另一个分支中运行。

  考虑以下按下触发器：

  ```
  Triggers:
    - Type: PUSH
      Branches:
        - main
  ```

  如果包含此触发器的工作流程定义文件存在于中main并被克隆到中test，则工作流程将永远不会使用中的文件自动启动test（尽管如果使用中的文件，则可以手动启动工作流程test）。查看注意事项 1 和 2，了解为什么工作流程永远不会使用中的文件自动运行test。

还要考虑以下拉取请求触发器：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

如果中存在包含此触发器的工作流程定义文件`main`，则工作流程将永远不会使用中的文件运行`main`。（但是，如果您在中创建`test`分支`main`，则工作流程将使用中的文件运行`test`。）查看注意事项 1 和 3 以了解原因。

添加推送、拉动或调度触发器

按照以下说明向工作流程添加推送、拉取或调度触发器。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，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。您可以使用以下值之一：

- 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程运行。工作流程运行将使用您要推送到的分支（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程运行。工作流程运行将使用您要从中提取的分支（即源分支）中的文件。

- 日程安排（可视化编辑器）或SCHEDULE（YAML 编辑器）

  计划触发器按您指定的 cron 表达式定义的计划启动工作流运行。将使用分支的文件为源存储库中的每个分支启动单独的工作流程运行。（要限制触发器激活的分支，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，请遵循以下准则：

  - 每个工作流程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，请参阅触发器示例。

拉取请求的事件

仅当您选择了拉取请求触发器类型时，才会显示此字段。

指定将启动工作流程运行的拉取请求事件的类型。以下是有效值：

- 拉取请求已创建（可视化编辑器）或OPEN（YAML 编辑器）

  创建拉取请求后，工作流程就会启动。

- 拉取请求已关闭（可视化编辑器）或CLOSED（YAML 编辑器）

  当拉取请求关闭时，工作流程就会开始运行。CLOSED事件的行为很棘手，最好通过一个例子来理解。请参阅示例：带有拉动、分支和 “CLOSED” 事件的触发器了解更多信息。

- 对拉取请求（可视化编辑器）或**REVISION**（YAML 编辑器）进行了新的修订

创建拉取请求的修订版后，工作流程运行即开始。第一个修订版是在创建拉取请求时创建的。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版。如果您在拉取请求触发器中包含该REVISION事件，则可以省略该OPEN事件，因为REVISION是的超集。OPEN

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，请参阅触发器示例。

计划

仅当您选择了 “计划” 触发器类型时，才会显示此字段。

指定 cron 表达式，用于描述您希望计划的工作流程何时运行。

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：

####days-of-month##days-of-week##

cron 表达式的示例

分钟	小时	一个月中的天数	月	一周中的几天	年	含义
0	0	?	*	MON-FRI	*	每周一至周五的午夜 (UTC +0) 运行工作流程。
0	2	*	*	?	*	每天凌晨 2:00（UTC +0）运行工作流程。

分钟	小时	一个月中的天数	月	一周中的几天	年	含义
15	22	*	*	?	*	每天晚上 10:15（UTC +0）运行工作流程。
0/30	22-2	?	*	SAT-SUN	*	周六至周日每隔 30 分钟运行一次工作流程，时间为起始日晚上 10:00 至次日凌晨 2:00（UTC +0）。
45	13	L	*	?	2023-2027	在 2023 年至 2027 年（含）之间的当月最后一天下午 1:45（UTC +0）运行工作流程。

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) 指定时间。不支持其他时区。
- 配置两次运行之间的间隔至少为 30 分钟。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，请参阅触发器示例。

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，以便知道何时开始工作流程运行。你可以使用正则表达式模式来定义你的分支名称。例如，用于匹配所有`main.*`以开头的分支`main`。

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，请指定要推送到的分支，即目标分支。将使用匹配分支中的文件，为每个匹配的分支启动一个工作流程。

  示例：`main.*`、`mainline`
- 对于拉取请求触发器，请指定要推送到的分支，即目标分支。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）中的源文件启动一个工作流程运行。

  示例：`main.*`、`mainline`、`v1\-.*`（匹配以开头的分支`v1-`）
- 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。将使用匹配分支中的工作流程定义文件和源文件，为每个匹配的分支启动一个工作流程。

  示例：`main.*`、`version\-1\.0`

> **Note**
>
> 如果您未指定分支，则触发器会监视源存储库中的所有分支，并将使用以下中的工作流程定义文件和源文件启动工作流程运行：

- 你要推送到的分支（用于推送触发器）。有关更多信息，请参阅 示例：一个简单的代码推送触发器。
- 您要从中提取的分支（用于拉取请求触发器）。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（用于计划触发器）。源存储库中的每个分支将启动一个工作流程运行。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，请参阅分支时的触发注意事项。

有关更多示例，请参阅触发器示例。

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，以便知道何时开始工作流程运行。您可以使用正则表达式来匹配文件名或路径。

有关示例，请参阅触发器示例。

10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 以以下示例为指导，添加`Triggers`章节和基础属性。有关更多信息，请参阅 工作流程 YAML 定义中的 Triggers。

代码推送触发器可能如下所示：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

拉取请求触发器可能如下所示：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - OPEN
      - REVISION
      - CLOSED
```

调度触发器可能如下所示：

```
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、拉取请求和调度触发器的更多示例，请参阅触发器示例。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

触发器示例

以下示例说明如何在工作流程定义文件中添加不同类型的触发器。

主题

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，每当将代码推送到源存储库中的任何分支时，该触发器就会启动工作流程运行。

激活此触发器后，使用您要推送到的分支（即目标分支）中的文件 CodeCatalyst 启动工作流程运行。

例如，如果您将提交推送到`main`，则使用工作流定义文件和其他源文件 CodeCatalyst 启动工作流程运行。`main`

再举一个例子，如果您将提交推送到`feature-branch-123`，则使用工作流定义文件和其他源文件 CodeCatalyst 启动工作流程运行。`feature-branch-123`

```
Triggers:
  - Type: PUSH
```

> **Note**
>
> 如果您希望只有在推送到时才启动工作流程`main`，请参阅示例：一个简单的 “推到主线” 触发器。

示例：一个简单的 “推到主线” 触发器

以下示例显示了一个触发器，每当将代码推送到源存储`main`库中的分支（仅推送到分支）时，该`main`触发器就会启动工作流程运行。

```
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，每当在源存储库中创建或修改拉取请求时，该触发器都会启动工作流程运行。

激活此触发器后，使用工作流程定义文件和您要提取的分支（即源分支）中的其他源文件 CodeCatalyst 启动工作流程运行。

例如，如果您使用名为的源分支`feature-123`和名为的目标分支创建拉取请求`main`，则使用工作流定义文件和其他源文件 CodeCatalyst 启动工作流程运行。`feature-123`

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
      - REVISION
```

示例：一个简单的计划触发器

以下示例显示了一个触发器，该触发器在每周一至周五的午夜 (UTC+0) 开始工作流程运行。

激活此触发器后，将为源存储库中包含带有此触发器的工作流程定义文件的每个分支 CodeCatalyst 启动一个工作流程运行。

例如，如果您的源存储库中有三个分支、`main`、`release-v1`、`feature-123`，并且每个分支都包含一个带有触发器的工作流程定义文件，则会 CodeCatalyst 启动三个工作流程运行：一个使用中的文件，另一个使用中的文件`main`，另一个使用中的文件`release-v1`，另一个使用中的文件，另一个使用中的文件`feature-123`。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 ? * MON-FRI *"
```

有关可以在该`Expression`属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有调度和分支的触发器

以下示例显示了每天下午 6:15（UTC+0）启动工作流程运行的触发器。

激活此触发器后，使用main分支中的文件 CodeCatalyst 启动工作流程运行，然后为以开头的每个分支开始额外运行release-。

例如，如果您的源存储库中有名为mainrelease-v1bugfix-1、、和bugfix-2的分支，则会 CodeCatalyst 启动两个工作流程运行：一次使用中的文件main，另一次使用中的文件release-v1。它不会启动bugfix-1和bugfix-1分支的工作流程运行。

```
Triggers:
  - Type: SCHEDULE
    Expression: "15 18 * * ? *"
    Branches:
      - main
      - release\-.*
```

有关可以在该Expression属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有调度、推送和分支的触发器

以下示例显示了一个触发器，该触发器在每天午夜 (UTC+0) 以及每当将代码推送到分支时启动工作流程。main

在本示例中：

- 工作流程运行从每天午夜开始。工作流程运行使用main分支中的工作流定义文件和其他源文件。
- 每当你向main分支推送提交时，工作流程也会开始运行。工作流程运行使用工作流定义文件和目标分支中的其他源文件 (main)。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 * * ? *"
    Branches:
      - main
  - Type: PUSH
    Branches:
      - main
```

有关可以在该Expression属性中使用的 cron 表达式的更多示例，请参阅Expression。

示例：带有拉动和分支的扳机

以下示例显示了一个触发器，每当有人打开或修改目标分支名main为的拉取请求时，该触发器就会启动工作流程运行。尽管Triggers配置中指定的分支是main，但工作流程运行将使用工作流定义文件和源分支（即您要从中提取的分支）中的其他源文件。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

示例：带有拉动、分支和 “CLOSED” 事件的触发器

以下示例显示了一个触发器，该触发器会在以开头的分支上关闭拉取请求时启动工作流程运行main。

在本示例中：

- 当您关闭以开头的目标分支的拉取请求时main，工作流程会自动开始运行，该工作流程将使用（现已关闭）源分支中的工作流程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在不合并拉取请求的情况下关闭拉取请求。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - CLOSED
```

示例：带有推送、分支和文件的触发器

以下示例显示了一个触发器，每当对main分支上的文件或src目录中的任何filename.txt文件进行更改时，该触发器就会启动工作流程运行。

激活此触发器后，使用main分支中的工作流定义文件和其他源文件 CodeCatalyst 启动工作流程运行。

```
Triggers:
  - Type: PUSH
    Branches:
      - main
    FilesChanged:
      - filename.txt
      - src\/.*
```

停止工作流运行

使用以下步骤停止正在运行的工作流程。如果跑步是意外启动的，你可能需要停止跑步。

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 。任何没有机会启动的操作都不会启动，而是会被标记为 “已放弃”。

> **Note**
>
> 如果运行已排队（也就是说，它没有正在进行的操作），则运行将立即停止。

停止工作流程运行

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 C I/CD，然后选择工作流程。
4. 在 “工作流程” 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。

对工作流程运行进行门控

门禁是一个工作流组件，除非满足某些条件，否则您可以使用它来阻止工作流程继续运行。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。

您可以在工作流程中的操作序列之间或在第一个操作（源代码下载后立即运行）之前添加门禁。如果你需要的话，你也可以在最后一个动作之后添加大门。

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 要求工作流程运行获得批准。

我可以设置一个门来与另一个动作并行运行吗？

不是。 盖茨只能在动作之前或之后逃跑。有关更多信息，请参阅 在门和操作之间设置依赖关系。

我能否使用门禁来阻止工作流程运行？

是的，有资格。

您可以阻止工作流程运行执行任务，这与阻止其启动略有不同。

要防止工作流执行任务，请在工作流程中的第一个操作之前添加一个门。在这种情况下，工作流程将开始运行，这意味着它将下载您的源存储库文件，但在门解锁之前，它将无法执行任务。

> **Note**
>
> 启动后被门屏蔽的工作流程仍计入每个空间配额和其他配额的最大并发工作流程运行次数。为确保不超过工作流程配额，请考虑使用工作流程触发器有条件地启动工作流程，而不是使用门禁。还可以考虑使用拉取请求批准规则而不是门禁。有关配额、触发器和拉取请求批准规则的更多信息，请参阅工作流程配额使用触发器自动启动工作流程、和管理合并拉取请求与批准规则的要求。

大门的局限性

盖茨有以下限制：

- Gates 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。
- 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行动组。

主题

- 为工作流程添加门禁
- 在门和操作之间设置依赖关系
- 指定门的主版本、次要版本或补丁版本

为工作流程添加门禁

按照以下说明向工作流程添加门禁，然后对其进行配置。

添加和配置门

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左边，选择盖茨。
8. 在大门目录中，搜索大门，然后选择加号 (+)，将该门添加到您的工作流程中。
9. 配置大门。选择 Visual 使用可视化编辑器，或选择 YAML 使用 YAML 编辑器。有关详细说明，请参阅：
   - 将 “批准” 门禁添加到工作流程
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。

在门和操作之间设置依赖关系

您可以将大门设置为在动作、操作组或大门之前或之后运行。例如，您可以设置一个`Approval`门以在`Deploy`操作之前运行。在这种情况下，据说`Deploy`动作取决于大`Approval`门。

要在门和操作之间设置依赖关系，请配置门或操作的 “依赖于” 属性。有关说明，请参阅设置操作之间的依赖关系。所引用的说明涉及工作流程操作，但同样适用于大门。

有关如何使用门设置 Depend s on 属性的示例，请参阅示例：配置 “批准” 门。

指定门的主版本、次要版本或补丁版本

默认情况下，向工作流程添加门禁时， 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-Gate:
  Identifier: aws/approval@v1
```

您可以延长版本，以便工作流程使用门的特定主要或次要版本。有关说明，请参阅指定操作的主版本、次要版本或补丁版本。所引用的主题涉及工作流程操作，但同样适用于大门。

要求工作流程运行获得批准

您可以将工作流程运行配置为需要批准才能继续。为此，您必须在工作流程中添加批准门。批准门禁可阻止工作流程继续进行，直到一组用户或一组用户在 CodeCatalyst 控制台中提交一项或多项批准。获得所有批准后，门将被“解锁”，并允许恢复工作流程运行。

在工作流程中使用批准门禁，让您的开发、运营和领导团队有机会在将更改部署给更广泛的受众之前对其进行审查。

如何解锁批准门？

要解锁批准门，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，并且允许每个用户提交一份批准。
- 条件 2：所有批准必须在登机口超时之前提交。门将在激活 14 天后超时。此时间段不可配置。
- 条件 3：任何人都不得拒绝工作流程运行。一次拒绝将导致工作流程运行失败。
- 条件 4：（仅在使用被取代的运行模式时适用。）该运行不得被以后的运行所取代。有关更多信息，请参阅 工作流程批准如何适用于排队、取代和并行运行模式？。

如果不满足任何条件，则 CodeCatalyst 停止工作流程并将运行状态设置为“失败”（在条件 1 到 3 的情况下）或“已取代”（在条件 4 的情况下）。

何时使用“批准”门

通常，您会在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的工作流程中使用批准门。通过在部署到生产环境之前设置门槛，您可以让审阅者有机会在新的软件修订版向公众发布之前对其进行验证。

谁可以提供批准？

任何身为项目成员且具有“参与者”或“项目管理员”角色的用户都可以提供批准。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。

如何通知用户需要批准？

要通知用户需要批准，您必须：

- CodeCatalyst 给他们发一条 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 “批准” 和 “拒绝” 按钮所在的页面，然后将该页面的 URL 粘贴到发给批准者的电子邮件或消息应用程序中。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程运行。

我能否使用 “批准” 门禁来阻止工作流程的启动？

是的，有资格。有关更多信息，请参阅 我能否使用门禁来阻止工作流程运行？。

工作流程批准如何适用于排队、取代和并行运行模式？

使用排队、取代或并行运行模式时，批准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、关于被取代的运行模式、关于并行运行模式部分以熟悉这些运行模式。对它们有了基本的了解后，请返回本节，了解当存在批准门时，这些运行模式是如何工作的。

当存在批准门禁时，将按以下方式处理运行：

- 如果您使用的是排队运行模式，则跑步将在当前在门口等待批准的跑步后面排队。当该门被解锁（即所有批准都已给出）时，队列中的下一次运行将进入大门，等待批准。此过程继续，排队的运行将通过门 one-by-one进行处理。 Figure 1说明了这个过程。
- 如果您使用的是被取代的运行模式，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门口的队列中，而是取代（接管）较早的运行。没有队列，任何目前在门口等待批准的跑步都将被取消并被更新的跑步所取代。 Figure 2说明了这个过程。
- 如果您使用的是并行运行模式，则以并行方式开始运行，并且不会形成队列。由于门前没有跑步，因此每次运行都会立即由门处理。 Figure 3说明了这个过程。

图 1：“排队运行模式” 和批准门

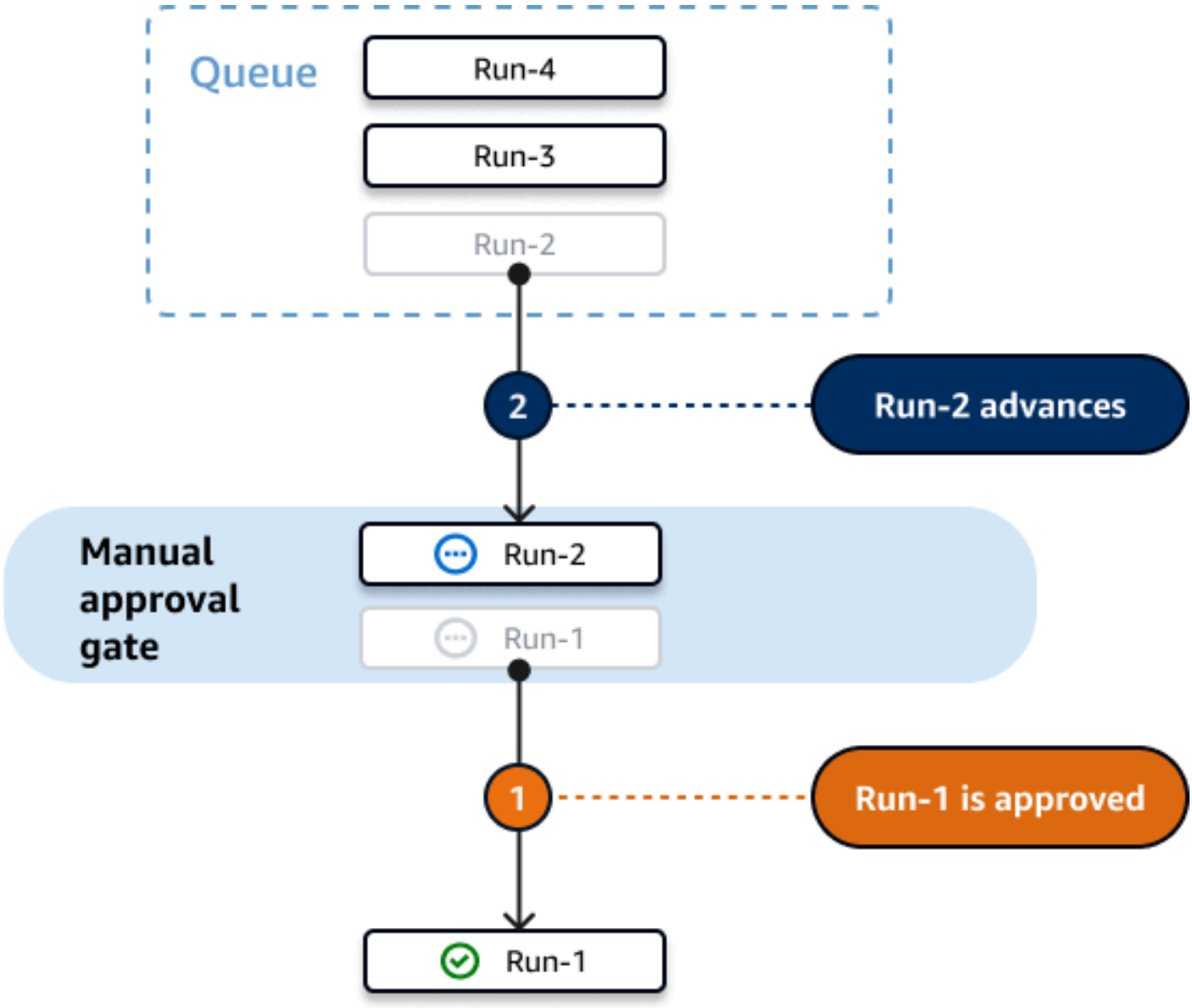


图 2：“被取代的运行模式” 和批准门

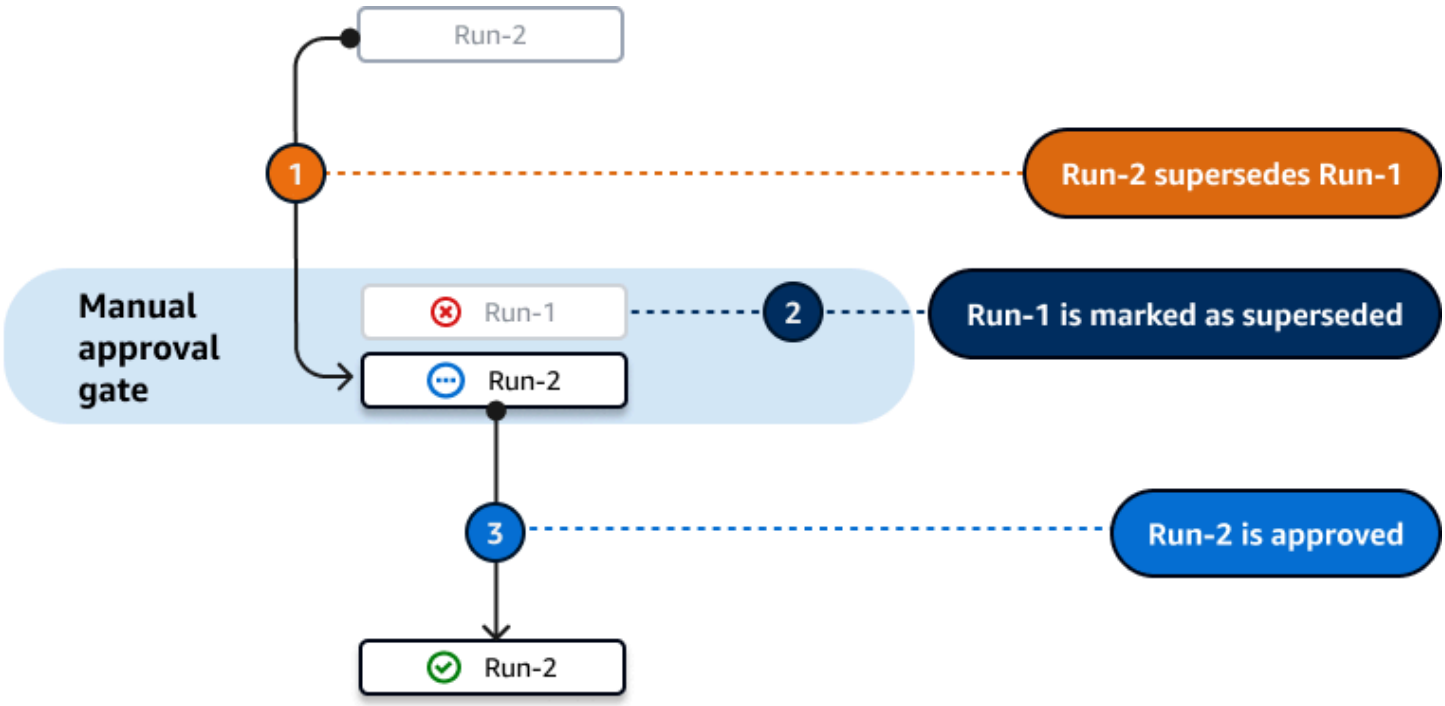
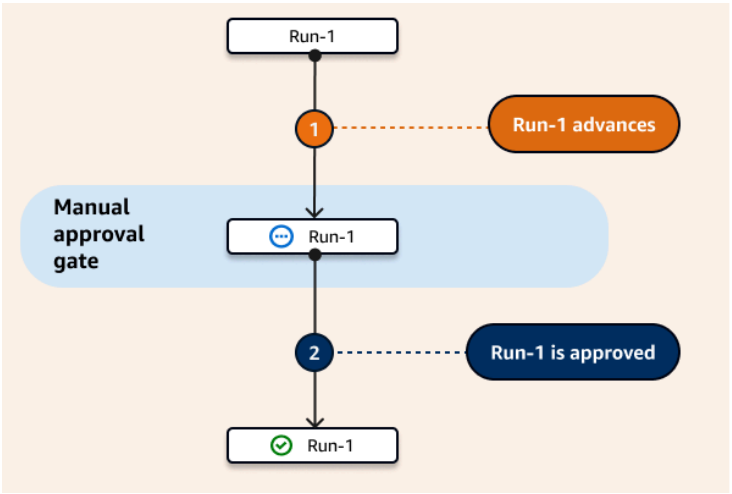
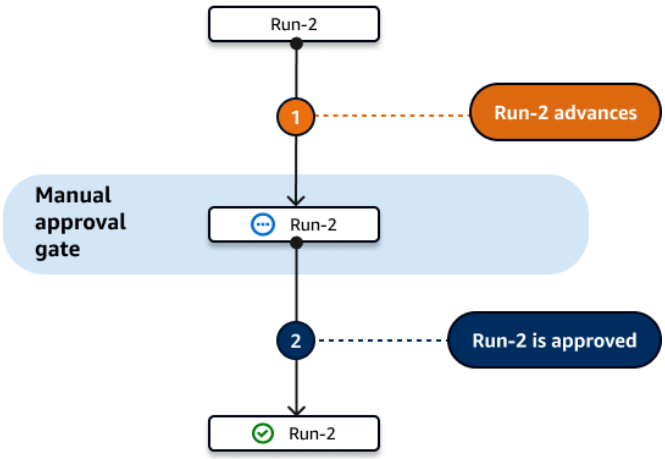


图 3：“并行运行模式” 和批准门





主题

- 示例：配置 “批准” 门
- 将 “批准” 门禁添加到工作流程
- 配置批准通知
- 批准或拒绝工作流程运行
- “批准” 门 YAML 定义

示例：配置 “批准” 门

以下示例说明如何在名为和的两个操作Approval_01之间添加一个名Staging为的批准门Production。Staging动作首先运行，Approval_01大门在第二位，Production动作最后运行。只有在Approval_01大门解锁时才会执行该Production操作。该DependsOn属性可确保StagingApproval_01、和Production阶段按顺序运行。

有关批准门禁的更多信息，请参阅要求工作流程运行获得批准。

```
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:
    ...
```

将 “批准” 门禁添加到工作流程

要将您的工作流程配置为需要批准，您必须在工作流程中添加批准门。按照以下说明向您的工作流程添加批准门禁。

有关此门的更多信息，请参阅要求工作流程运行获得批准。

Visual

向工作流程添加 “批准” 门槛（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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 在左上角，选择盖茨。
7. 在盖茨目录的批准中，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门运行的操作、操作组或门。默认情况下，向工作流程添加门禁时，门禁设置为取决于工作流程中的最后一个操作。如果删除此属性，则门将不依赖于任何东西，并且将首先运行，然后再执行其他操作。

   > **Note**
   >
   > 必须将门配置为在操作、操作组或门控之前或之后运行。无法将其设置为与其他操作、操作组和门并行运行。

   有关 “依赖于” 功能的更多信息，请参阅在门和操作之间设置依赖关系。
9. 选择配置选项卡。
10. 在门名称字段中，执行以下操作。

    指定您要给门起的名称。在工作流程中，所有门名都必须是唯一的。门名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。您不能使用引号在门名中启用特殊字符和空格。
11. （可选）在 “批准次数” 字段中，执行以下操作。

    指定解锁批准门所需的最小批准数量。最少为 1。最大值为2。如果省略，则默认为1。

    > **Note**
    >
    > 如果要省略该`ApprovalsRequired`属性，请从工作流程定义文件中移除该门的`Configuration`部分。
12. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
13. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

YAML

向工作流程添加 “批准” 门槛（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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 以以下示例为指导，添加Approval章节和基础属性。有关更多信息，请参阅 工作流程 YAML 定义中的 “批准” 门 YAML 定义。

```
Actions:
  MyApproval_01:
    Identifier: aws/approval@v1
    DependsOn:
      - PreviousAction
    Configuration:
      ApprovalsRequired: 2
```

   有关另一个示例，请参阅示例：配置 “批准” 门。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 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

您可以向 Slack 频道 CodeCatalyst 发送通知，告知用户工作流程运行需要获得批准。用户会看到通知并单击其中的链接。该链接会将他们带到 CodeCatalyst 批准页面，在那里他们可以批准或拒绝工作流程。

您还可以配置通知以通知用户工作流程已获得批准、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

确保已在工作流程中添加了批准门禁。有关更多信息，请参阅 将 “批准” 门禁添加到工作流程。

向 Slack 频道发送工作流程批准通知

1. 使用 Slack CodeCatalyst 进行配置。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程的 CodeCatalyst 项目中，启用通知（如果尚未启用）。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. 开启待审批工作流程，然后选择用于发送通知的 CodeCatalyst Slack 频道。
   e. （可选）开启其他通知，提醒人们注意已批准、已拒绝和已过期的批准。您可以启用 “工作流程运行已批准”、“工作流程运行已拒绝”、“取代工作流程审批” 和 “工作流程审批超时”。在每条通知旁边，选择发送通知的 CodeCatalyst Slack 频道。
   f. 选择保存。

批准或拒绝工作流程运行

需要批准或拒绝包含批准门禁的工作流程运行。用户可以从以下开始提供批准或拒绝：

- 控制 CodeCatalyst 台
- 团队成员提供的链接
- 自动发送的 Slack 通知

在用户表示批准或拒绝后，此决定无法撤消。

> **Note**
>
> 只有某些用户可以批准或拒绝工作流程运行。有关更多信息，请参阅 谁可以提供批准？。

开始前的准备工作

确保已在工作流程中添加了批准门禁。有关更多信息，请参阅 将 “批准” 门禁添加到工作流程。

要批准或拒绝工作流程，请从 CodeCatalyst 控制台开始运行

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 在工作流程图中，选择代表批准门的方框。

   出现侧面板。

   > **Note**
   >
   > 此时，如果您愿意，可以将此页面的 URL 发送给其他批准者。

6. 在“审核决定”下，选择“批准”或“拒绝”。
7. （可选）在“评论-可选”中，输入注释，说明您批准或拒绝工作流程运行的原因。
8. 选择提交。

要批准或拒绝工作流程，请从团队成员提供的链接开始运行

1. 选择您的团队成员发送给您的链接。（您可以让您的团队成员阅读前面的步骤以获取链接。）
2. 如果询问 CodeCatalyst，请登录。

   您将被重定向到工作流程运行批准页面。
3. 在“审核决定”下，选择“批准”或“拒绝”。
4. （可选）在“评论-可选”中，输入注释，说明您批准或拒绝工作流程运行的原因。
5. 选择提交。

要批准或拒绝工作流程，请从自动的 Slack 通知开始运行

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. 如果询问 CodeCatalyst，请登录。

   您将被重定向到工作流程运行页面。

4. 在工作流程图中，选择批准门槛。
5. 在 “审核决定” 下，选择 “批准” 或 “拒绝”。
6. （可选）在 “评论-可选” 中，输入注释，说明您批准或拒绝工作流程运行的原因。
7. 选择提交。

“批准” 门 YAML 定义

以下是 YAML 对批准门的定义。要了解如何使用此门，请参阅要求工作流程运行获得批准。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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）

指定您要给门起的名称。在工作流程中，所有门名必须是唯一的。门名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。您不能使用引号在门名中启用特殊字符和空格。

默认值：Approval_nn。

对应的用户界面：配置选项卡/ 门名称

Identifier

(*Approval*/Identifier)

（必需）

标识大门。批准门支持版本1.0.0。除非要缩短版本，否则不要更改此属性。有关更多信息，请参阅指定操作的主版本、次要版本或补丁版本。

默认值：aws/approval@v1。

对应的用户界面：工作流程图/ Approval _nn/ aws/approver @v1 标签

DependsOn

(*Approval*/DependsOn)

（可选）

指定必须成功运行才能使该门运行的操作、操作组或门。默认情况下，向工作流程添加门禁时，门禁设置为取决于工作流程中的最后一个操作。如果删除此属性，则门将不依赖于任何东西，并且将首先运行，然后再执行其他操作。

> **Note**
>
> 必须将门配置为在操作、操作组或门控之前或之后运行。无法将其设置为与其他操作、操作组和门并行运行。

有关“依赖于”功能的更多信息，请参阅在门和操作之间设置依赖关系。

相应的 UI：“输入”选项卡/ 取决于

Configuration

(*Approval*/Configuration)

（可选）

可以在其中定义门的配置属性的部分。

相应的 UI：“配置” 选项卡

ApprovalsRequired

(*Approval*/Configuration/ApprovalsRequired)

（可选）

指定解锁批准门所需的最小批准数量。最少为 1。最大值为2。如果省略，则默认为1。

> **Note**
>
> 如果要省略该ApprovalsRequired属性，请从工作流程定义文件中移除该门的Configuration部分。

相应的 UI：“配置” 选项卡/ 批准数量

配置运行的排队行为

默认情况下，当多个工作流程同时运行时，会将它们 CodeCatalyst 排成队列，并按启动顺序逐一处理。您可以通过指定运行模式来更改此默认行为。有几种运行模式：

- （默认）排队运行模式- CodeCatalyst 进程一个接一个地运行
- 被取代的运行模式 — CodeCatalyst 进程一个接一个地运行，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

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，等待的运行形成队列。

队列位于操作和操作组的入口处，因此您可以在同一个工作流程中拥有多个队列（请参阅Figure 1）。当排队的跑步进入操作时，该操作将被锁定，其他跑步都无法进入。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。

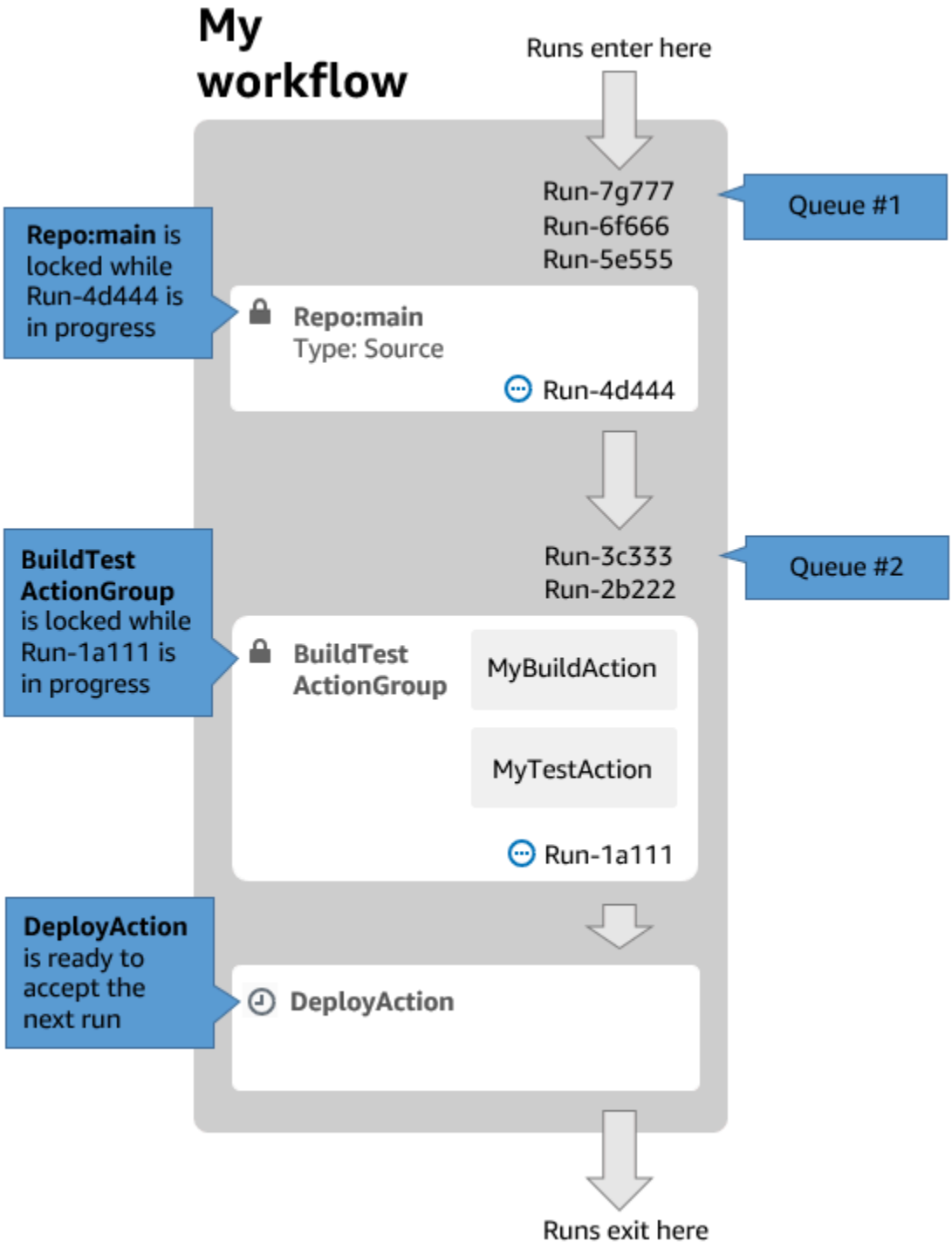
Figure 1说明了在排队运行模式下配置的工作流程。它显示了：

- Seven 在工作流程中顺其自然。
- 两个队列：一个在输入源 (repo: Main) 的条目之外，另一个在操作的条目之外。BuildTestActionGroup
- 两个锁定的方块：输入源 (repo: Main) 和。BuildTestActionGroup

以下是工作流程运行完成处理后会发生的情况：

- 当 run-4d444 完成源存储库的克隆后，它将退出输入源并加入 run-3c333 之后的队列。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操作并进入动作。DeployAction然后，run-2b222 将进入操作。BuildTestActionGroup

图 1：在 “排队运行模式” 下配置的工作流程



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 one-to-one 关系——使用取代模式时，可能会对这些功能进行分组。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 将启动，依此类推。如果您要使用取代模式而不是队列模式，则您的功能（和提交）将在运行中组合在一起，取代其他功能（和提交）。

- 您希望避免使用并行模式时可能出现的竞争条件和意外问题。例如，如果两位软件开发人员 Wang 和 Saanvi 在大致相同的时间启动工作流程以部署到 Amazon ECS 集群，那么 Wang 的运行可能会开始在集群上进行集成测试，而 Saanvi 的运行会向集群部署新的应用程序代码，从而导致 Wang 的测试失败或测试错误的代码。再举一个例子，你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的负载。例如，如果您的工作流程中有三个操作，则最多可以同时运行三个操作。对一次可以运行的次数施加限制可以使运行吞吐量更具可预测性。
- 您想限制工作流程向第三方服务发出的请求数量。例如，您的工作流程可能有一个生成操作，其中包括从 Docker Hub 提取映像的说明。Docker Hub 将每个账户每小时可以发出的拉取请求数量限制为一定数量，如果你超过限制，你将被封锁。使用排队运行模式降低运行吞吐量将减少每小时向 Docker Hub 发出的请求，从而限制锁定以及由此导致的构建和运行失败的可能性。

最大队列大小：50

有关最大队列大小的注意事项：

- 最大队列大小是指工作流中允许在所有队列中运行的最大次数。
- 如果队列的运行长度超过 50 次，则 CodeCatalyst 丢弃第 51 次及随后的运行。

失败行为：

如果某项运行在操作处理时变得无响应，则其后面的运行将暂停在队列中，直到操作超时。一小时后操作超时。

如果操作内部的运行失败，则允许其后面的第一个排队运行继续进行。

关于被取代的运行模式

被取代的运行模式与排队运行模式相同，不同之处在于：

- 如果排队的跑步赶上队列中的另一场运行，则后一次运行将取代（接管）先前的运行，而较早的运行将被取消并标记为 “已取代”。
- 作为第一个 bullet 中描述的行为的结果，当使用取代的运行模式时，队列只能包含一次运行。

以中的工作流程Figure 1为指导，将取代的运行模式应用于此工作流程将导致以下结果：

- run-7g777 将取代队列中的其他两次运行，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，成为队列 #2 中唯一剩下的跑步。run-2b222 将被取消。

如果需要，请使用已取代的运行模式：

- 吞吐量比排队模式更好
- 与排队模式相比，向第三方服务发出的请求更少；如果第三方服务有速率限制（例如 Docker Hub），则这是有利的

关于并行运行模式

在并行运行模式下，运行相互独立，无需等待其他运行完成后再开始。没有队列，运行吞吐量仅受工作流程内部操作完成速度的限制。

在开发环境中使用并行运行模式，在这种环境中，每个用户都有自己的功能分支，并部署到其他用户未共享的目标。

> **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，因为可能会导致竞争条件。当并行工作流运行尝试同时更改共享资源时，就会出现争用条件，从而导致不可预测的结果。

最大并行运行次数：每个 CodeCatalyst空间 1000

配置运行模式

您可以将运行模式设置为排队、取代或并行。默认为已排队。

当您将运行模式从排队或取代更改为 parallel 时，会 CodeCatalyst 取消排队的运行，并允许当前由操作处理的运行在取消之前完成。

当您将运行模式从 parallel 更改为排队或被取代时， CodeCatalyst 让所有当前正在运行的并行运行完成。将运行模式更改为排队或已取代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. 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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. 已排队 — 请参阅 关于排队运行模式
   b. 已@@ 取代 — 请参阅 关于被取代的运行模式
   c. 平行 — 参见 关于并行运行模式
8. （可选）选择 “验证” 以在提交之前验证工作流程的 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. 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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 “” 一顶级属性节中对RunMode属性的描述工作流程 YAML 定义。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。

9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

在工作流程运行之间缓存文件

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，并在后续的工作流程运行中从该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

文件缓存允许您将数据组织成多个缓存，每个缓存都在`FileCaching`属性下引用。每个缓存都会保存由给定路径指定的目录。将在 future 工作流程运行中恢复指定的目录。以下是使用名为和的多个缓存进行缓存的 YAML 代码片段示例。`cacheKey1` `cacheKey2`

```
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。当配置的队列或按需计算机在本地缓存中遇到缓存丢失的情况时，将从远程缓存中恢复依赖关系。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。这样可以保护每个工作流程免受其他工作流程的侵害，这些数据可能会推送影响构建或部署的错误数据。限制是通过缓存作用域来强制执行的，缓存范围将缓存隔离到每个工作流程和分支配对。例如，分支`workflow-Afeature-A`中的文件缓存与兄弟分支`workflow-Afeature-B`中的文件缓存不同。

当工作流程查找指定的文件缓存但无法找到时，就会发生缓存失误。发生这种情况可能有多种原因，例如创建了新分支或引用了新的缓存但尚未创建时。它也可能发生在缓存过期时，默认情况下，缓存过期发生在上次使用后 14 天。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。备用缓存是备用缓存，它提供了恢复部分缓存的机会，部分缓存可能是较旧版本的缓存。首先通过在属性名称下`FileCaching`搜索匹配项来恢复缓存，如果找不到，则进行评估`RestoreKeys`。如果属性名称和所有属性都出现缓存失误`RestoreKeys`，则工作流程将继续运行，因为缓存是尽力而为，不能保证。

创建缓存

您可以按照以下说明向工作流程添加缓存。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要在其中添加缓存的操作。
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：

   密钥

   指定主缓存属性名称的名称。缓存属性名称在您的工作流程中必须是唯一的。每个操作中最多可以有五个条目`FileCaching`。

   路径

   为您的缓存指定关联路径。

   恢复密钥-可选

   指定在找不到主缓存属性时用作备用还原密钥。恢复密钥名称在您的工作流程中必须是唯一的。每个缓存中最多可以有五个条目`RestoreKeys`。
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。

6. 选择 YAML。
7. 在工作流程操作中，添加类似于以下内容的代码：

```
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” 以在提交之前验证工作流程的 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

以下是属性名称和的限制`RestoreKeys`：

- 在工作流程中，名称必须是唯一的。
- 名称仅限于字母数字字符（A-Z、a-z、0-9）、连字符 (-) 和下划线 (_)。
- 名称最多可包含 180 个字符。
- 每个动作中最多可以有五个缓存。`FileCaching`
- 每个缓存中最多可以有五个条目`RestoreKeys`。

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。

查看工作流程运行状态和详细信息

您可以查看单个工作流程运行的状态和详细信息，也可以同时查看多个运行的状态和详细信息。

有关可能的运行状态的列表，请参阅工作流程运行状态。

> **Note**
>
> 您还可以查看工作流程状态，该状态不同于工作流程运行状态。有关更多信息，请参阅 查看工作流程状态。

主题

查看单次运行的状态和详细信息

您可能需要查看单个工作流程运行的状态和详细信息，以检查它是否成功，查看它在什么时候完成，或者查看谁或什么启动了它。

查看单次运行的状态和详细信息

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 在工作流程的名称下，选择 “运行”。
6. 在 “运行历史记录” 的 “运行 ID” 列中，选择一次运行。例如，Run-95a4d。
7. 在跑步名称下，执行以下操作之一：

   - 可视化以查看显示工作流程运行操作及其状态的工作流程图（请参阅工作流程运行状态）。此视图还显示运行期间使用的源存储库和分支。

     在工作流程图中，选择一个操作以查看运行期间该操作生成的日志、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。有关查看生成或部署日志的更多信息，请参阅查看生成操作的结果或查看部署日志。
   - YAML 查看运行时使用的工作流程定义文件。

- 用于查看工作流程运行生成的构件。有关构件的更多信息，请参阅 使用构件在工作流程中的操作之间共享数据。
- 报告，用于查看工作流程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。
- 用于查看工作流程运行产生的输出变量的变量。有关变量的更多信息，请参阅在工作流程中配置和使用变量。

> **Note**
>
> 如果删除了运行的父工作流程，则在运行详细信息页面的顶部会显示一条表明这一事实的消息。

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程运行的状态和详细信息，了解项目中正在进行的工作流程活动，并了解工作流程的整体运行状况。

查看项目中所有运行的状态和详细信息

1. 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 C I/CD，然后选择工作流程。
4. 在 “工作流程” 下，选择 “运行”。

   将显示项目中所有存储库中所有工作流程、所有分支中的所有运行情况。

   该页面包括以下各列：

   - 运行 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-工作流程运行的处理状态。有关运行状态的更多信息，请参阅工作流程运行状态。
   - 触发器-启动工作流程运行的人、提交、拉取请求 (PR) 或计划。有关更多信息，请参阅 使用触发器自动启动工作流程。
   - 工作流程-启动运行的工作流程的名称，以及工作流定义文件所在的源存储库和分支。您可能需要扩大列宽才能看到此信息。

> **Note**
>
> 如果将此列设置为 “不可用”，则通常是因为关联的工作流程已被删除或移动。

- 开始时间-工作流程运行的开始时间。
- 持续时间-处理工作流程运行所花费的时间。持续时间过长或很短可能表示存在问题。
- 结束时间-工作流程运行结束的时间。

查看特定工作流程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程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程中造成瓶颈，或者查看哪些运行当前正在进行或已完成。

查看特定工作流程所有运行的状态和详细信息

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 在工作流程的名称下，选择 “运行”。

   将显示与所选工作流程关联的运行。

   该页面分为两个部分：

   - 活动运行-显示正在进行的运行。这些运行将处于以下状态之一：进行中。
   - 运行历史记录-显示已完成（即未进行中）的运行。

     有关运行状态的更多信息，请参阅工作流程运行状态。

在工作流程图中查看工作流的运行情况

您可以查看工作流程中所有运行的状态，因为它们一起完成工作流程。运行结果显示在工作流程图中（而不是列表视图）。这可以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程中一起进行的状态

> **Note**
>
> 仅当您的工作流程使用排队或被取代的运行模式时，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. 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 C I/CD，然后选择工作流程。
4. 选择包含要查看的运行的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。

   > **Note**
   >
   > 确保你查看的是工作流程页面，而不是运行页面。

5. 选择左上角的 “最新状态” 选项卡。

   将出现工作流程图。

6. 查看工作流程图。该图显示了工作流程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：

   - 出现在顶部、源代码之前的跑步将排队等候开始。
   - 在操作之间出现的运行将排队等候下一个操作的处理。
   - 出现在动作中的运行是 1. 当前正在由该操作处理，2. 已完成该操作的处理，或者 3. 未被该操作处理（通常是因为之前的操作失败）。

配置工作流程执行的操作

操作是工作流程的主要组成部分，它定义了工作流程运行期间要执行的逻辑工作单元或任务。通常，一个工作流程包括多个按顺序运行或并行运行的操作，具体取决于您的配置方式。

主题

- 向 CodeCatalyst 工作流程添加操作
- 从工作流程中移除操作
- 开发自定义操作
- 将操作分组为行动组
- 将操作配置为依赖于其他操作
- 使用构件在工作流程中的操作之间共享数据
- 指定操作的主版本、次要版本或补丁版本
- 确定操作的可用版本
- 查看动作的源代码
- 将 GitHub 操作集成到工作流程中

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 实验室行动
- GitHub 行动
- 第三方操作

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

  此操作会生成您的工件并在 Docker 容器中运行单元测试。有关更多信息，请参阅 添加生成操作。
- 测试

  此操作会针对您的应用程序或工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。

- 亚马逊 S3 发布

  此操作会将您的应用程序项目复制到 Amazon S3 存储桶。有关更多信息，请参阅 使用工作流程将文件发布到 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 Cloud Development Kit (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 使用工作流程调用 AWS Lambda 函数。
- GitHub 行动

  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 使用工作流程调用 AWS Lambda 函数。
- 部署 AWS CloudFormation 堆栈

  此操作会部署 AWS CloudFormation 堆栈。有关更多信息，请参阅 使用工作流程部署 AWS 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 使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)。
- 部署到 Kubernetes 集群

  此操作将应用程序部署到 Kubernetes 集群。有关更多信息，请参阅 使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service。
- 渲染 Amazon ECS 任务定义

  此操作将容器映像 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 使用工作流程修改 Amazon ECS 任务定义文件。

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，请参阅向 CodeCatalyst 工作流程添加操作。

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

  此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部署您的无服务器应用程序。
- 使亚马逊 CloudFront 缓存失效

  此操作会使给定路径集的 CloudFront 缓存失效。
- 传出 Webhook

  此操作允许用户使用 HTTPS 请求将工作流程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。
- 发布到亚马逊 SNS

  此操作允许用户通过创建主题、发布主题或订阅主题来与 Amazon SNS 集成。
- 推送到亚马逊 ECR

  此操作构建 Docker 映像并将其发布到亚马逊弹性容器注册表 (Amazon ECR) Registry 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 社区版

  此操作运行 Terraform 社区版`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，请参阅向 CodeCatalyst 工作流程添加操作。

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 将 GitHub 操作集成到工作流程中。

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他文档。

有关向工作流程添加第三方操作和查看其自述文件的信息，请参阅向 CodeCatalyst 工作流程添加操作。

向 CodeCatalyst 工作流程添加操作

按照以下说明向工作流程添加操作，然后对其进行配置。

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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 在左上角，选择 + 操作，将出现操作目录。
7. 在下拉列表中，执行以下任一操作：

   - 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。
     - 第三方操作带有按***###***分类的标签，其中 v *endor* 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 (+) 将操作添加到您的工作流程中。
   - 选择操作名称以查看其自述文件。
9. 配置操作。选择 Visual 使用可视化编辑器，或选择 YAML 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：

   - 添加生成操作
   - 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 AWS CloudFormation 堆栈” 操作
   - 添加 “部AWS CDK 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：

   - 该操作的自述文件。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：

- 将 GitHub 操作集成到工作流程中

有关添加第三方操作的说明，请参阅：

- 该操作的自述文件。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。

从工作流程中移除操作

按照以下说明从工作流程中移除操作。

Visual

使用可视化编辑器移除动作

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，在要删除的操作中，选择垂直省略号图标，然后选择移除。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 找到 YAML 中包含您要删除的操作的部分。

   选择该部分，然后按键盘上的删除键。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 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软件包。
2. 创建 CodeCatalyst 存储库来存储您的操作代码。
3. 初始化动作。这列出了操作所需的源文件，包括可以用自己的代码更新的动作定义文件 (`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。

将操作分组为行动组

一个操作组包含一个或多个操作。将操作分组到操作组可以帮助您保持工作流程井井有条，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在其他操作组或操作中。

主题

-
- 示例：定义两个操作组

按照以下说明定义操作组。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在中Actions，添加类似于以下内容的代码：

```
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，请参阅中对action-group-name属性的描述工作流程 YAML 定义。操作

8. （可选）选择 “验证” 以在提交之前验证工作流程的 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操作组：BuildAndTest和Deploy。
该BuildAndTest组包括两个操作（Build和Test），该Deploy组还包括两个操作（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:
          ...
```

将操作配置为依赖于其他操作

默认情况下，当您将操作添加到工作流程时，它们将在可视化编辑器中并排添加。这意味着当您启动工作流程运行时，这些操作将并行运行。如果要让操作按顺序运行（并在可视化编辑器中垂直显示），则

必须设置它们之间的依赖关系。例如，您可以将Test操作设置为依赖于该Build操作，以便测试操作在生成操作之后运行。

可以在动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。请查阅设置依赖关系的指南以确保您的依赖项设置符合工作流程的 YAML 语法。

主题

- 设置操作之间的依赖关系
- 设置依赖关系的指南
- 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

按照以下说明设置工作流程中操作之间的依赖关系。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，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运行此操作的操作、操作组或门。

   有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在依赖于另一个操作的操作中，添加类似于以下内容的代码：

```
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。有关一般指南，请参阅设置依赖关系的指南。有关更多信息，请参阅中工作流程 YAML 定义针对您的操作的DependsOn属性的描述。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

设置依赖关系的指南

配置依赖关系时，请遵循以下准则：

- 如果某项操作位于组内，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，但不能依赖于不同级别的其他操作和操作组。

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程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

以下示例说明如何使用DependsOn属性将Test操作配置为依赖于该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该FirstAction操作。请注意，操作组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该BuildAndTestGroup操作组。请注意，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FirstAction依赖于SecondAction操作、操作和BuildAndTestGroup操作组。请注意，DeployGroup这与FirstActionSecondAction、和处于同一级别BuildAndTestGroup。

```
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:
      ...
```

使用构件在工作流程中的操作之间共享数据

构件是工作流程操作的输出，通常由文件夹或文件存档组成。构件之所以重要，是因为它们允许您在操作之间共享文件和信息。

例如，您可能有一个生成sam-template.yml文件的生成操作，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生成操作与部署操作共享sam-template.yml文件。代码可能看起来像这样：

```
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，生成操作 (BuildAction) 生成一个sam-template.yml文件，然后将其添加到名为的输出构件中MYARTIFACT。随后的部署操作 (DeployAction) 指定MYARTIFACT为输入，允许其访问该sam-template.yml文件。

我能否在不将工件指定为输出和输入的情况下共享它们？

是的，您可以在操作之间共享工件，而无需在操作的 YAML Inputs 代码的Outputs和部分中指定它们。为此，必须开启计算共享。有关计算共享以及如何在开启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**
>
> 尽管计算共享功能允许您通过消除对Outputs和Inputs部分的需求来简化工作流程的 YAML 代码，但该功能存在一些局限性，在开启之前您应该注意这些局限性。有关这些限制的信息，请参阅计算共享的注意事项。

我能否在工作流程之间共享工件？

不可以，您不能在不同的工作流程之间共享构件；但是，您可以在同一工作流程中的操作之间共享构件。

主题

- 定义输出对象
- 定义输入工件
- 在构件中引用文件
- 正在下载工件
- 文物示例

定义输出对象

按照以下说明定义要输出操作的对象。然后，该构件可供其他操作使用。

> **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它们，请仔细阅读随后的可视化编辑器说明，并查看该操作是否包含 “输出” 选项卡上的 “输出构件” 按钮。如果是，则支持输出工件。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，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 “添加对象”，然后在字段中输入信息，如下所示。

    构建构件名称

    指定操作生成的对象的名称。Artifact 名称在工作流程中必须是唯一的，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 (-) 和其他特殊字符。不能使用引号在输出对象名称中启用空格、连字符和其他特殊字符。

    有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

    由 build 生成的文件

    指定操作输出的对象中 CodeCatalyst 包含的文件。这些文件由工作流程操作在运行时生成，也可在源存储库中找到。文件路径可以位于源存储库或先前操作中的对象中，并且是相对于源存储库或项目根目录的。你可以使用 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，请参见语法指南和惯例。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个工件或来源中找到它。有关更多信息，请参阅 引用源存储库中的文件 和 在构件中引用文件。

11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流程操作中，添加类似于以下内容的代码：

```
action-name:
  Outputs:
    Artifacts:
      - Name: artifact-name
        Files:
          - file-path-1
```

```
            - file-path-2
```

有关更多示例，请参阅文物示例。有关更多信息，工作流程 YAML 定义请参阅您的操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

定义输入工件

如果要使用其他操作生成的对象，则必须将其指定为当前操作的输入。您可以将多个构件指定为输入，具体取决于操作。有关更多信息，工作流程 YAML 定义请参阅您的操作。

> **Note**
>
> 您不能引用其他工作流程中的构件。

按照以下说明将来自另一个操作的对象指定为当前操作的输入。

先决条件

在开始之前，请确保已输出其他操作的构件。有关更多信息，请参阅 定义输出对象。输出构件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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要将对象指定为输入的操作。
8. 选择输入。
9. 在 “构件-可选” 中，执行以下操作：

指定要作为此操作输入的先前操作中的对象。在之前的操作中，这些构件必须已定义为输出对象。

如果您未指定任何输入对象，则必须至少在下方指定一个源存储库*action-name*/Inputs/Sources。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 如果 Artifacts -可选下拉列表不可用（可视化编辑器），或者在验证 YAML（YAML 编辑器）时出现错误，则可能是因为该操作仅支持一个输入。在这种情况下，请尝试移除源输入。

10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要将构件指定为输入的操作中，添加类似于以下内容的代码：

```
action-name:
  Inputs:
    Artifacts:
      - artifact-name
```

有关更多示例，请参阅文物示例。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 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

如果您的文件位于某个对象中，并且需要在其中一个工作流程操作中引用该文件，请完成以下步骤。

> **Note**
>
> 另请参阅 引用源存储库中的文件。

引用构件中的文件

- 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
      Artifacts:
        - artifact-name
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_artifact-name/build-output && cat file.txt
```

在前面的代码中，该操作在####构件的build-output目录中查找并显示该file.txt文件。

有关更多示例，请参阅文物示例。

> **Note**
>
> 你可以省略$CATALYST_SOURCE_DIR_*artifact-name*/前缀，具体取决于你配置操作的方式。有关更多信息，请参阅以下指南。

有关如何引用变量的指导：

- 如果您的操作仅包含下方的一个项目Inputs（例如，它包含一个输入构件而不包含源），则可以省略前缀，只指定相对于构件根目录的文件路径。
- 如果文件位于主输入中，也可以省略前缀。主输入要么是列出的第一个输入工件（如果没有）WorkflowSource。WorkflowSource
- 根据您使用的操作，前缀可能会有所不同。有关更多信息，请参阅下表。

操作类型	要使用的文件路径前缀	示例
生成操作，测试操作	`$CATALYST_SOURCE_DIR_` *artifact-name* `/`	`$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt`

所有其他操作	`$CATALYST_SOURCE_DIR_` *artifact-name* `/`
	
	或者
	
	`/artifacts/` *current-action-name* */artifact-name* */##### $CATALYST_SOURCE_DIR_* *artifact-name* *#####/#*
		`$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt`
		
		或者
		
		`/artifacts/MyCurrentAction/MyArtifact/folder1/file.txt`

正在下载工件

您可以下载并检查工作流程操作生成的构件以进行故障排除。您可以下载两种类型的神器：

- 源构件-包含运行开始时存在的源存储库内容快照的对象。
- 工作流程构件-在工作流程配置文件的Outputs属性中定义的工件。

下载工作流程输出的构件

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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. a在工作流程的名称下，选择运行。
6. 在 “运行历史记录” 的 “运行 ID” 列中，选择一次运行。例如，Run-95a4d。
7. 在跑步名称下，选择工件。
8. 在构件旁边，选择下载。下载存档文件。它的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提取存档。

文物示例

以下示例说明如何在工作流程定义文件中输出、输入和引用工件。

主题

- 示例：输出构件
- 示例：输入由其他操作生成的构件
- 示例：引用多个构件中的文件
- 示例：在单个构件中引用文件
- 示例：存在对象时引用对象中的文件 WorkflowSource

示例：输出构件

以下示例说明如何输出包含两个.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输出调用的ARTIFACT4构件并将其输入到BuildActionB。BuildActionA

```
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输出两个名为 and in 的工件BuildActionC，ART5然后ART6在（下方）中引用两个名为file5.txt（在 artifact 中 ART5ART6）和 file6.txtBuildActionD（在构件中Steps）的文件。

> **Note**
>
> 有关引用文件的更多信息，请参阅在构件中引用文件。

> **Note**
>
> 尽管该示例显示了正在使用的$CATALYST_SOURCE_DIR_ART5前缀，但您可以省略它。这是因为ART5这是主要输入。要了解有关主要输入的更多信息，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

以下示例向您展示如何输出一个名为的工件BuildActionE，然后ART7在（下方ART7）中引用file7.txtBuildActionF（在构件中Steps）。

请注意，该引用并不像在目录前面那样要求在build-output目录前面加上$CATALYST_SOURCE_DIR_####前缀。示例：引用多个构件中的文件这是因为下面只指定了一个项目Inputs。

> **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：存在对象时引用对象中的文件 WorkflowSource

以下示例向您展示如何输出一个名为的工件BuildActionG，然后ART8在（下方ART8）中引用file8.txtBuildActionH（在构件中Steps）。

请注意，参考文献需要$CATALYST_SOURCE_DIR_#####缀，就像在中一样。示例：引用多个构件中的文件这是因为下方指定了多个项目Inputs（一个源和一个构件），所以你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
```

指定操作的主版本、次要版本或补丁版本

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
```

您可以缩短Identifier属性中的完整版本，以便工作流程始终使用操作的最新次要版本或补丁版本。

例如，如果您指定：

```
My-CloudFormation-Action:
  Identifier: aws/cfn-deploy@v1.0
```

... 最新的补丁版本是1.0.4，那么操作将使用1.0.4。比如说，如果发布了更高版本1.0.5，则该操作将使用1.0.5。比如说，如果发布了次要版本1.1.0，那么该操作将继续使用1.0.5。

有关指定版本的详细说明，请参阅以下主题之一。

使用以下说明来指明您希望您的工作流程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，也可以指定特定的补丁版本。

我们建议使用操作的最新次要版本或补丁版本。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

将工作流程配置为使用最新版本的操作或特定的补丁版本

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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，然后将版本设置为以下任一值：

   - actionid *en* tifier @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。
   - 动作标识符 @v *##*。 mino@@ *r* — 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。
   - 动作标识符 @v *##*。 *####*。 p@@ *atch*-使用此语法让工作流程使用特定的补丁版本。

   > **Note**
   >
   > 如果您不确定有哪些版本可用，请参阅确定操作的可用版本。

   > **Note**
   >
   > 您不能省略主要版本。

9. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
10. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

确定操作的可用版本

按照以下说明来确定哪些版本的操作可供您在工作流程中使用。

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 C I/CD，然后选择工作流程。

   b. 选择任何工作流程的名称，或创建一个。有关创建工作流程的信息，请参阅创建工作流。

   c. 选择编辑。

   d. 在左上角，选择 + 操作以打开操作目录。

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. 搜索动作，然后选择其名称。不要选择加号 (+)。

此时将显示有关该操作的详细信息。

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。

查看动作的源代码

您可以查看操作的源代码，以确保它不包含风险代码、安全漏洞或其他缺陷。

按照以下说明查看CodeCatalyst、CodeCatalyst 实验室或第三方操作的源代码。

> **Note**
>
> 要查看某项GitHub操作的源代码，请前往 GitHub Marketplace 中该操作的页面。该页面包含指向操作存储库的链接，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

查看动作的源代码

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 C I/CD，然后选择工作流程。

   b. 选择任何工作流程的名称或创建一个。有关创建工作流程的信息，请参阅创建工作流。

   c. 选择编辑。

   d. 在左上角，选择 + 操作以打开操作目录。

   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. 搜索动作，然后选择其名称。不要选择加号 (+)。

      将显示有关该操作的详细信息。

4. 在底部附近的操作详细信息对话框中，选择下载。

   将出现一个页面，显示操作源代码所在的 Amazon S3 存储桶。有关亚马逊 S3 的信息，请参阅什么是亚马逊 S3？ 在 Amazon 简单存储服务用户指南中。

5. 检查代码，确保其符合您对质量和安全性的期望。

将 GitHub 操作集成到工作流程中

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 操作。

  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。要使操作正常运行，必须使用包装器。

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 操作交互？

是。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 输出参数以便其他操作可以使用它 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。有关图像和工具的更多信息，请参阅指定运行时环境 Docker 镜像。

- 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。
  - https://github.com/actions 中列出了几乎所有的操作。
- 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。请勿以用户 1001 的身份运行该操作。（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。在工作流程中，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 工作流程入门。
2. 在工作流程中，您可以添加精选 GitHub 操作或添加 GitHub 操作操作。
3. 您可以执行以下任一操作：
   - 如果您选择添加精选动作，请对其进行配置。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）在工作流程中，您可以添加其他操作，例如生成和测试操作。有关更多信息，请参阅 使用中的工作流程构建、测试和部署 CodeCatalyst。
5. 您可以手动启动工作流程，也可以通过触发器自动启动工作流程。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 启动工作流程手动运行。

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的 build 机器。

我也可以使用 GitHub 工作流程吗？

不是。

主题

教程：在工作流程中使用 GitHub操作的 Lint 代码

在本教程中，您将将 Super-Linter GitHub 操作添加到亚马逊 CodeCatalyst 工作流程中。Super-Linter 操作检查代码，查找代码存在错误、格式问题和可疑结构的区域，然后将结果输出到控制台）。 CodeCatalyst将 linter 添加到工作流程后，您可以运行工作流程来整理示例 Node.js 应用程序 () `app.js`。然后，您可以修复报告的问题，并再次运行工作流程以查看修复是否奏效。

> **Tip**
>
> 可以考虑使用 Super-Linter 来清理 YAML 文件，例如模板。AWS CloudFormation

主题

先决条件

在开始之前，你需要：

- 一个连接的 CodeCatalyst 空间 AWS 账户。有关更多信息，请参阅 创建空间。
- 您的 CodeCatalyst 空间中有一个名为的空项目`codecatalyst-linter-project`。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，您将在中创建源存储库 CodeCatalyst。您将使用此存储库来存储本教程的示例应用程序源文件。`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. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
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。

第 2 步：添加 app.js 文件

在此步骤中，您将向源存储库中添加一个`app.js`文件。`app.js`包含的函数代码有一些错误，linter 会发现这些错误。

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. 在 “文件名” 中，输入app.js。保留其他默认选项。

7. 选择 Commit (提交)。

   现在，您已经创建了一个名为的文件`app.js`。

步骤 3：创建运行 Super-Linter 操作的工作流程

在此步骤中，您将创建一个在将代码推送到源存储库时运行 Super-Linter 操作的工作流程。该工作流程由以下构建块组成，您可以在 YAML 文件中定义这些构件：

- 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程运行。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程。
- “GitHub 操作” 操作 — 触发后，“GitHub 操作” 操作会运行 Super-Linter 操作，该操作反过来会检查源存储库中的所有文件。如果 linter 发现问题，则工作流程操作将失败。

创建运行 Super-Linter 操作的工作流程

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 C I/CD，然后选择工作流程。
3. 选择 “创建工作流程”。
4. 对于源存储库，选择`codecatalyst-linter-source-repository`。
5. 对于 Branch，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
```

在前面的代码中，按照本*github-action-code*过程的以下步骤中的说明替换为 Super-Linter 操作代码。

9. 前往 Market [place GitHub 中的 Super-Linter 页面](#)。
10. 在`steps:`（小写）下，找到代码并将其粘贴到`Steps:`（大写）下 CodeCatalyst的工作流程中。

    调整 GitHub 操作代码以符合 CodeCatalyst 标准，如以下代码所示。

    现在，您的 CodeCatalyst 工作流程如下所示：

```
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）选择验证以确保 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，您已经创建了一个工作流程。由于在工作流程顶部定义了触发器，因此工作流程运行会自动启动。

查看正在运行的工作流程

1. 在导航窗格中，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程:`codecatalyst-linter-workflow`.
3. 在工作流程图中，选择SuperLinterAction。
4. 等待操作失败。之所以会出现这种故障，是因为 linter 在代码中发现了问题。

5. 让 CodeCatalyst 控制台保持打开状态并转至第 4 步：修复 Super-Linter 发现的问题。

第 4 步：修复 Super-Linter 发现的问题

Super-Linter 应该在`app.js`代码以及源代码库中包含`README.md`的文件中发现了问题。

为了解决 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 Code Base。

   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，在那里你可以找到问题的起点。它们看起来类似于以下内容：

```
/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. 在源存储库`README.md`中修复`app.js`和并提交您的更改。

> **Tip**
>
> 要修复此问题`README.md`，请在代码块中添加`markdown`，如下所示：
>
> ```
> ```markdown
> Setup examples:
> ...
> ```
> ```

   您的更改会自动启动另一个工作流程。等待工作流程完成。如果您修复了所有问题，则工作流程应该会成功。

清理

进行清理 CodeCatalyst 以从您的环境中删除本教程的痕迹。

要清理干净 CodeCatalyst

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 删除`codecatalyst-linter-source-repository`。
3. 删除`codecatalyst-linter-workflow`。

在本教程中，你学习了如何将 Super-Linter GitHub 操作添加到 CodeCatalyst 工作流程中以整理一些代码。

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，请参阅将 GitHub 操作集成到工作流程中：

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：在工作流程中使用 GitHub操作的 Lint 代码。

Visual

使用可视化编辑器添加 “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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。

7. 在左上角，选择 + 操作以打开操作目录。
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程图中，然后打开其配置窗格。

     或者

   - 选择 “GitHub 操作”。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。

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 动作” 动作 YAML 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
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程图中，然后打开其配置窗格。

     或者

- 选择 “GitHub 操作”。将出现 “操作详细信息” 对话框。在此对话框中：
  - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。

10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的解释“GitHub 动作” 动作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
12. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

“GitHub 动作” 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，请参见“GitHub 动作” 动作 YAML 定义中的工作流程 YAML 定义。

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，您需要用自己的值替换`with:`下面的空字符串。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，请参阅将 GitHub 操作集成到工作流程中。

> **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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
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程图中，然后打开其配置窗格。

     或者

   - 选择 GitHub 操作的名称。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
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 动作” 动作 YAML 定义。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” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。

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程图中，然后打开其配置窗格。

     或者

   - 选择 GitHub 操作的名称。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
10. 根据需要修改 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 动作” 动作 YAML 定义。

    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” 以在提交之前验证工作流程的 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 输出参数以便其他操作可以使用它

您可以在 CodeCatalyst 工作流程中使用GitHub 输出参数。

> **Note**
>
> 输出参数的另一个词是变量。由于在其文档中 GitHub 使用了 “输出参数” 一词，所以我们也将使用这个术语。

按照以下说明从 GitHub 操作中导出 GitHub 输出参数，以便其他 CodeCatalyst 工作流程操作可以使用该参数。

导出 GitHub 输出参数

1. 打开工作流程并选择 “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 操作” 操作中，添加一个包含如下基础`Variables`属性的`Outputs`部分：

```
Actions:
  MyGitHubAction:
```

```
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'step-id_output-name'
```

替换：

- *step-id*，其中包含 GitHub 操作steps部分中id:属性的值。
- *#######*参数的 GitHub 名称。

示例

以下示例说明如何导出名为的 GitHub 输出参数SELECTEDCOLOR。

```
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

使用以下说明来引用 GitHub 输出参数。

引用 GitHub 输出参数

1. 完成导出 GitHub 输出参数以便其他操作可以使用它中的步骤。

   GitHub 输出参数现在可用于其他操作。
2. 注意输出参数的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}
```

替换：

- *ac* tion-name，其名称为生成输出参数的动 CodeCatalystGitHub 作（请勿使用 GitHub 操作的name或id）。
- ou@@ *tput-name*，其中包含你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中设置变量GitHubActionA、输出SELECTEDCOLOR变量，然后在中引用该变量BuildActionB。

```
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 动作” 动作 YAML 定义

以下是 “GitHub操作” 操作的 YAML 定义。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

在以下代码中选择 YAML 属性以查看其描述。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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
```

动作名称

（必需）

指定操作的名称。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

UI#“##” ###/

Identifier

（*####*/Identifier）

标识操作。除非要更改版本，否则请勿更改此属性。有关更多信息，请参阅 指定操作的主版本、次要版本或补丁版本。

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/ @v1 标签 github-actions-runner

DependsOn

（*####*/DependsOn）

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关“依赖”功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入”选项卡/ 依赖- 可选

Compute

（*####*/Compute）

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Fleet

（*#### /Compute/* ***Fleet*** ）

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，`Fleet`则默认为`Linux.x86-64.Large`。

相应的 UI：“配置” 选项卡/ “计算舰队”- 可选

Timeout

（*####*/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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Environment

（*####*/Environment ）

（可选）

指定要用于操作的 CodeCatalyst 环境。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/环境/账户/角色

Name

（*#### /Environment/* ***Name*** ）

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Connections

（*#### /Environment/* ***Connections*** ）

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

对应的用户界面：无

Name

（*#### /Environment/Connections/* ***Name*** ）

（可选）

指定账户连接的名称。

对应的用户界面：配置选项卡/'环境/账户/角色'/账户连接AWS

Role

（*#### /Environment/Connections/* ***Role*** ）

（可选）

指定此操作用于访问和操作 Amazon S3 和 Amazon ECR 等 AWS 服务的 IAM 角色的名称。确保将此角色添加到您的账户关联中。要向账户连接添加 IAM 角色，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 只要角色具有足够的权限，您就可以在此处指定该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性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：配置选项卡/'环境/账户/角色'/角色

Inputs

（####/Inputs）

（可选）

该Inputs部分定义了操作在工作流程运行期间所需的数据。

> **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（比如源和构件）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，以区分主输入中的文件。有关更多信息，请参阅 示例：引用多个构件中的文件。

相应的 UI：“输入” 选项卡

Sources

（*#### /Inputs/* **Sources**）

（可选）

指定代表操作所需的源存储库的标签。当前，唯一支持的标签是WorkflowSource，它表示存储工作流程定义文件的源存储库。

如果省略了源，则必须在下*action-name*/Inputs/Artifacts方指定至少一个输入对象。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 UI：“输入” 选项卡/ “来源”- 可选

Artifacts - input

（*#### /Inputs/* **Artifacts**）

（可选）

指定要作为此操作输入的先前操作中的对象。在之前的操作中，这些构件必须已定义为输出对象。

如果您未指定任何输入构件，则必须至少在下方指定一个源存储库*action-name*/Inputs/Sources。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 如果 Artifacts -可选下拉列表不可用（可视化编辑器），或者在验证 YAML（YAML 编辑器）时出现错误，则可能是因为该操作仅支持一个输入。在这种情况下，请尝试移除源输入。

相应的 UI：“输入” 选项卡/ 工件- 可选

Variables - input

（*#### /Inputs/* **Variables**）

（可选）

指定一个名称/值对序列，这些对定义了要提供给操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的 UI：“输入” 选项卡/ “变量”- 可选

Outputs

（*####*/Outputs）

（可选）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts - output

（*#### /Outputs/* ***Artifacts***）

（可选）

指定操作生成的对象的名称。Artifact 名称在工作流程中必须是唯一的，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 (-) 和其他特殊字符。不能使用引号在输出对象名称中启用空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/ 工件

Name

（*#### /Outputs/Artifacts/* ***Name***）

（如果包含Artifacts - output，则为必填项）

指定操作生成的对象的名称。Artifact 名称在工作流程中必须是唯一的，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 (-) 和其他特殊字符。不能使用引号在输出对象名称中启用空格、连字符和其他特殊字符。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

对应的用户界面：输出选项卡/构件/添加构件/构建构件名称

Files

（*#### /Outputs/Artifacts/* ***Files*** ）

（如果包含Artifacts - output，则为必填项）

指定由操作输出的构件中 CodeCatalyst 包含的文件。这些文件由工作流程操作在运行时生成，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中的对象中，并且是相对于源存储库或项目根目录的。你可以使用 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，请参见语法指南和惯例。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

> **Note**
>
> 您可能需要在文件路径中添加前缀，以指明要在哪个工件或来源中找到它。有关更多信息，请参阅 引用源存储库中的文件 和 在构件中引用文件。

相应的用户界面：输出选项卡/构件/添加工件/构建生成的文件

Variables - output

（*#### /Outputs/* ***Variables*** ）

（可选）

指定要导出操作的变量，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），请参阅在工作流程中配置和使用变量。

相应的用户界面：“输出” 选项卡/变量/ 添加变量

变量名-1

（*####/Outputs/Variables****####-*** 1）

（可选）

指定要导出操作的变量的名称。此变量必须已经在同一操作的`Inputs`或`Steps`部分中定义。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的用户界面：“输出” 选项卡/变量/添加变量/名称

AutoDiscoverReports

（*#### /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属性限制要检查哪些文件。

对应的用户界面：无

Enabled

（*#### /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`则默认为`true`。

相应的 UI：“输出” 选项卡/报告/自动发现报告

ReportNamePrefix

（*#### /Outputs/AutoDiscoverReports/* ***ReportNamePrefix*** ）

（如果包含并启用AutoDiscoverReports，则为必填项）

为其找到的所有报告指定前缀，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” 选项卡/报告/自动发现报告/报告前缀

IncludePaths

（*#### /Outputs/AutoDiscoverReports/* ***IncludePaths*** ）

Or

（*####/Outputs/Reports/#### -1/#****IncludePaths***

（如果AutoDiscoverReports包含并启用，或者包含在内，Reports则为必填项）

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，请参见语法指南和惯例。

如果省略此属性，则默认值为"**/*"，这意味着搜索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。

对应的用户界面：

- 输出选项卡/报告/自动发现报告/'包含/排除路径'/包含路径
- ***#####/##/######/####-1 /'##/####'/####***

ExcludePaths

（ *#### /Outputs/AutoDiscoverReports/* ***ExcludePaths*** ）

Or

（ *####/Outputs/Reports/#### -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，请参见语法指南和惯例。

对应的用户界面：

- 输出选项卡/报告/自动发现报告/'包含/排除路径'/排除路径
- ***#####/##/######/####-1 /'##/####'/####***

SuccessCriteria

（ *#### /Outputs/AutoDiscoverReports/* ***SuccessCriteria*** ）

Or

(*####/Outputs/Reports/#### -1/#**SuccessCriteria*

（可选）

为测试、代码覆盖率、软件组合分析 (SCA) 和静态分析 (SA) 报告指定成功标准。

有关更多信息，请参阅 为报告配置成功标准。

对应的用户界面：

- “输出” 选项卡/报告/自动发现报告/成功标准
- ***#####/##/######/####-1 /####***

PassRate

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/ **PassRate***)

Or

(*####/Outputs/Reports/##/SuccessCriteria/**PassRate**##-1*)

（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” 选项卡/报告/自动发现报告/成功标准/通过率
- ***#####/##/######/####-1 /####/###***

LineCoverage

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/ **LineCoverage***)

Or

(*####/Outputs/Reports/##/SuccessCriteria/**LineCoverage**##-1*)

（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。线路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” 选项卡/报告/自动发现报告/成功标准/行覆盖率
- ***#####/##/######/####-1 /####/####***

BranchCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***BranchCoverage*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****BranchCoverage****##-1* ）

（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。

对应的用户界面：

- 输出选项卡/报告/自动发现报告/成功标准/分支覆盖率
- ***#####/##/######/####-1 /####/######***

Vulnerabilities

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***Vulnerabilities*** ）

Or

（*####/Outputs/Reports/##/SuccessCriteria/****Vulnerabilities****##-1* ）

（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，必须指定：

- 要计入的漏洞的最低严重程度。有效值（从最严重到最不严重）为：CRITICAL、HIGH、MEDIUM、LOW、INFORMATIONAL。

  例如，如果您选择HIGH，则将对CRITICAL漏洞HIGH进行统计。
- 您希望允许的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，请参阅软件成分分析报告。

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。

对应的用户界面：

- “输出” 选项卡/报告/自动发现报告/成功标准/漏洞
- *#####/##/######/####-1 /####/##*

Reports

（*#### /Outputs/* ***Reports*** ）

（可选）

指定测试报告配置的部分。

相应的 UI：“输出” 选项卡/ “报告”

报告名称 1

（*####/Outputs/Reports/####-1* ）

（如果包含Reports，则为必填项）

您要为将从原始 CodeCatalyst 报告生成的报告命名。

相应的 UI：“输出” 选项卡/报告/手动配置报告/报告名称

Format

（*####/Outputs/Reports/#### -1/#****Format***

（如果包含Reports，则为必填项）

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，请指定黄瓜（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 nUni t（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，指定 Visual Studio TRX（可视化编辑器）或VISUALSTUDIOTRX（YAML 编辑器）。
  - 对于 TestNG XML，请指定 te stNG（可视化编辑器）或TESTNGXML（YAML 编辑器）。
- 有关代码覆盖率报告：
  - 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）。

UI#“##” ###/##/######/####/####-1 /#########

Configuration

（*####*/Configuration）

（必填）一个部分，您可以在其中定义操作的配置属性。

对应的 UI：“配置” 选项卡

Steps

（*#### /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。按照以下准则添加代码：

1. 将 GitHub 操作`steps:`部分中的代码粘贴到 CodeCatalyst 工作流程的`Steps:`部分。该代码以破折号 (-) 开头，看起来与以下内容类似。

   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`#####`代码，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，工作流程 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范围之内。

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，请参见引用源存储库中的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

为工作流程配置计算和运行时环境 Docker 镜像

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程中任何操作的属性。同样，如果计算被定义为任何操作的属性，则无法在工作流程中对其进行定义。

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 EC2 在操作运行期间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 支持更快的工作流程操作运行。Lambda 允许您运行基本工作流程，这些工作流程可以构建、测试和部署具有常见运行时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 EC2 计算类型：

- Lambda 不支持来自指定注册表的运行时环境镜像。
- Lambda 不支持需要根权限的工具。对于`yum`或等工具`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 的构建或运行。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 不支持写入外部文件。`/tmp`配置工作流程操作时，您可以重新配置要安装或写入的工具。`/tmp`如果要安装构建操作`npm`，请务必将其配置为安装到`/tmp`。
- Lambda 不支持长度超过 15 分钟的运行时间。

计算实例集

CodeCatalyst 提供以下计算队列：

- 按需实例集
- 已配置的舰队

对于按需队列，当工作流操作开始时，工作流会配置所需的资源。操作结束后，机器将被销毁。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 EC2 提供支持并由维护的机器。CodeCatalyst使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。使用已配置的队列，您的计算机将始终处于运行状态，并且只要配置完毕，就会产生成本。

要创建、更新或删除舰队，您必须拥有 Space 管理员角色或项目管理员角色。

按需车队房产

CodeCatalyst 提供以下按需队列：

名称	操作系统	架构	vCPU	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
					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**
>
> 按需车队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`。

已配置的舰队属性

已配置的队列包含以下属性：

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 队列。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

机器类型

每个实例的计算机类型。以下计算机类型可用：

vCPU	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

容量

分配给队列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

定义操作数量超过队列容量时的行为。

按需预置额外容量

其他计算机是按需设置的，这些计算机会根据正在运行的新操作自动向上扩展，然后在操作完成时缩小到基本容量。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。

创建已配置的舰队

按照以下说明创建已配置的队列。

> **Note**
>
> 已配置的舰队将在闲置 2 周后停用。如果再次使用，它们将自动重新激活，但是这种重新激活可能会导致延迟。

创建已配置的舰队

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 选择创建已配置舰队。
3. 在已配置的舰队名称文本字段中，输入您的舰队的名称。
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，为您的计算机选择计算机类型。
6. 在容量文本字段中，输入队列中计算机的最大数量。
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，请参阅已配置的舰队属性。
8. 选择创建。

创建已配置的队列后，您就可以将其分配给操作了。有关更多信息，请参阅 为操作分配预配置的队列或按需计算。

编辑已配置的队列

按照以下说明编辑已配置的队列。

> **Note**
>
> 已配置的舰队将在闲置 2 周后停用。如果再次使用，它们将自动重新激活，但是这种重新激活可能会导致延迟。

编辑已配置的舰队

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 在已配置舰队列表中，选择要编辑的舰队。
3. 选择编辑。
4. 在容量文本字段中，输入队列中计算机的最大数量。
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，请参阅已配置的舰队属性。
6. 选择保存。

删除已配置的舰队

按照以下说明删除已配置的队列。

删除已配置的舰队

> **Warning**
>
> 在删除已配置的队列之前，请从操作的 YAML 代码中删除该`Fleet`属性，将其从所有操作中删除。任何在删除已配置队列后继续引用该队列的操作都将在下次运行该操作时失败。

1. 在导航窗格中，选择 CI/CD，然后选择计算。
2. 在已配置舰队列表中，选择要删除的舰队。

3. 选择删除。
4. 输入**delete**确认删除。
5. 选择 删除。

为操作分配预配置的队列或按需计算

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。要改用预配置的队列，或者使用其他按需队列（例如）`Linux.x86-64.2XLarge`，请使用以下说明。

Visual

开始前的准备工作

- 如果要分配已配置的队列，则必须先创建已配置的队列。有关更多信息，请参阅 创建已配置的舰队。

为操作分配已配置的队列或其他舰队类型

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要为其分配已配置队列或新队列类型的操作。
8. 选择配置选项卡。
9. 在计算队列中，执行以下操作：

   指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

   使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

10. （可选）选择 “验证” 以在提交之前验证工作流程的 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

- 如果要分配已配置的队列，则必须先创建已配置的队列。有关更多信息，请参阅 创建已配置的舰队。

为操作分配已配置的队列或其他舰队类型

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。 /
5. 选择编辑。
6. 选择 YAML。
7. 找到您要为其分配已配置队列或新舰队类型的操作。
8. 在操作中，添加一个Compute属性并设置Fleet为您的舰队名称或按需队列类型。有关更多信息，请参阅中生成和测试操作 YAML 定义针对您的操作的Fleet属性的描述。
9. （可选）选择 “验证” 以在提交之前验证工作流程的 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，工作流程中的操作在队列中的不同实例上运行。这种行为为操作提供了对输入状态的隔离性和可预测性。默认行为需要进行显式配置，以便在操作之间共享上下文，例如文件和变量。

计算共享功能允许您在同一个实例上运行工作流程中的所有操作。使用计算共享可以缩短工作流程运行时间，因为配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），而无需进行额外的工作流程配置。

当使用计算共享运行工作流程时，默认队列或指定队列中的实例将在该工作流中的所有操作持续时间内保留。工作流程运行完成后，实例预留即被释放。

主题

在共享计算上运行多个操作

您可以在工作流程级别使用定义 YAML 中的`Compute`属性来指定操作的队列和计算共享属性。您也可以使用中的可视化编辑器配置计算属性 CodeCatalyst。要指定队列，请设置现有队列的名称，将计算类型设置为 EC2，然后开启计算共享。

> **Note**
>
> 只有当计算类型设置为 EC2 时，才支持计算共享，而且 Windows Server 2022 操作系统不支持计算共享。有关计算队列、计算类型和属性的更多信息，请参阅为工作流程配置计算和运行时环境 Docker 镜像。

> **Note**
>
> 如果您使用的是免费套餐，并且在工作流程定义 YAML 中手动指定`Linux.x86-64.XLarge`或`Linux.x86-64.2XLarge`队列，则该操作仍将在默认队列 (`Linux.x86-64.Large`) 上运行。有关计算可用性和定价的更多信息，请参阅分层选项表。

开启计算共享后，将自动跨操作复制包含工作流程源的文件夹。您无需配置输出工件，也无需在整个工作流程定义（YAML 文件）中将其作为输入工件引用。作为工作流程作者，您需要使用输入和输出连接环境变量，就像不使用计算共享一样。如果要在工作流程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 使用构件在工作流程中的操作之间共享数据 和 在工作流程运行之间缓存文件。

您的工作流程定义文件所在的源存储库由标签标识`WorkflowSource`。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程运行中的后续操作使用。通过操作（例

如添加、修改或删除文件）对包含工作流源的文件夹所做的任何更改也将在工作流程的后续操作中显示。您可以在任何工作流程操作中引用位于工作流源文件夹中的文件，就像在不使用计算共享的情况下一样。有关更多信息，请参阅 引用源存储库中的文件。

> **Note**
>
> 计算共享工作流程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工件，但不支持输入工件。

计算共享的注意事项

您可以使用计算共享功能运行工作流程，以加快工作流程运行并在使用相同实例的工作流程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	没有计算共享
计算类型	Amazon EC2	亚马逊 EC2、AWS Lambda
实例配置	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	亚马逊 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/`
Workflow 结构	操作只能按顺序运行	操作可以并行运行
跨工作流程操作访问数据	访问缓存的工作流程源 (`WorkflowSource`)	访问共享工件的输出（需要额外配置）

开启计算共享

按照以下说明为工作流程开启计算共享。

Visual

使用可视化编辑器开启计算共享

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。
5. 选择编辑。
6. 选择 “视觉”。
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 “计算队列-可选” 下拉菜单中，选择要用于运行工作流程操作的队列。您可以选择按需队列，也可以创建并选择预配置的队列。有关更多信息，请参阅创建已配置的舰队和为操作分配预配置的队列或按需计算
10. 切换开关以开启计算共享，让工作流程中的操作在同一个队列上运行。
11. （可选）选择工作流程的运行模式。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 编辑器开启计算共享

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。
5. 选择编辑。
6. 选择 YAML。
7. 开启计算共享，将`SharedInstance`字段设置为`TRUE`和设置`Type`为`EC2`。设置`Fleet`为要用于运行工作流程操作的计算队列。您可以选择按需队列，也可以创建并选择预配置的队列。有关更多信息，请参阅创建已配置的舰队和为操作分配预配置的队列或按需计算

   在工作流程 YAML 中，添加类似于以下内容的代码：

```
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” 以在提交之前验证工作流程的 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：亚马逊 S3 发布

示例：亚马逊 S3 发布

以下工作流程示例展示了如何通过两种方式执行 Amazon S3 发布操作：首先使用输入项目，然后使用计算共享。使用计算共享，不需要输入工件，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`属性来维护连续操作；构建操作必须成功运行才能运行 S3 发布。

- 如果没有计算共享，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: dev-bucket
```

- 通过设置SharedInstance为使用计算共享时TRUE，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程源来共享构件。输入工件不是必需的，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: dev-bucket
```

指定运行时环境 Docker 镜像

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。只有生成和测试操作支持自定义镜像。有关更多信息，请参阅 为操作分配自定义运行时环境 Docker 镜像。

主题

活跃的图像

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。目前有两组活跃图片：一组于2024年3月发布，另一组于2022年11月发布。

一个动作是使用 2024 年 3 月还是 2022 年 11 月的图片取决于动作：

- 在 2024 年 3 月 26 日当天或之后添加到工作流程中的构建和测试操作将在其 YAML 定义中包含一个明确指定 2024 年 3 月图像的Container部分。您可以选择删除该Container部分以恢复到 2022 年 11 月的图片。
- 在 2024 年 3 月 26 日之前添加到工作流程中的生成和测试操作将不会在其 YAML 定义中包含任何Container部分，因此将使用 2022 年 11 月的图像。您可以保留 2022 年 11 月的映像，也可以对其进行升级。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 镜像” 下拉列表中选择 2024 年 3 月的镜像。此选择将在动作的 YAML 定义中添加一个Container部分，该部分将填充相应的 2024 年 3 月图片。
- 所有其他操作都将使用 2022 年 11 月的图片，无论它们何时被添加到工作流程中。目前无法将这些操作升级为使用 2024 年 3 月的图像。

主题

2024 年 3 月的图片

2024 年 3 月的图片是提供的最新图片。 CodeCatalyst每种计算类型/舰队组合都有一个 2024 年 3 月的映像。

下表显示了 2024 年 3 月的每张映像上安装的工具。

2024 年 3 月的图片工具

工具	CodeCatalyst 适用于 Linux 的 Amazon EC2 x86_64-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的亚马逊 EC2 Arm64-**CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst 适用于 Linux 的 Arm64-**CodeCatalystLinuxLambda_Arm64:2024_**
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

工具	CodeCatalyst 适用于 Linux 的 Amazon EC2 x86_64-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst 适用于 Linux 的亚马逊 EC2 Arm64-**CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst 适用于 Linux 的 Arm64-**CodeCatalystLinuxLambda_Arm64:2024_**
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
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 月的图片

每种计算类型/舰队组合都有一个 2022 年 11 月的映像。如果您配置了预配置的队列，则还有 2022 年 11 月的 Windows 映像可用于构建操作。

下表显示了 2022 年 11 月的每张映像上安装的工具。

2022 年 11 月图片工具

工具	CodeCatalyst 适用于 Linux 的 Amazon EC2 x86_64-**CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst 适用于 Linux 的 Lambda x86_64-**CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst 适用于 Linux 的亚马逊 EC2 Arm64-**CodeCatalystLinux_Arm64:2022_11**	CodeCatalyst 适用于 Linux 的 Arm64-**CodeCatalystLinuxLambda_Arm64:2022_**
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

如果现用图像不包含我需要的工具怎么办？

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 镜像。有关更多信息，请参阅 为操作分配自定义运行时环境 Docker 镜像。

  > **Note**
  >
  > 如果要提供自定义运行时环境 Docker 镜像，请确保您的自定义镜像中安装了 Git。

- 您可以让工作流程的生成或测试操作安装所需的工具。

  例如，您可以在生成或测试操作的 YAML 代码`Steps`部分中包含以下说明：

  ```
  Configuration:
    Steps:
      - Run: ./setup-script
  ```

  然后，*`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
  ```

  有关生成操作 YAML 的更多信息，请参阅生成和测试操作 YAML 定义。

为操作分配自定义运行时环境 Docker 镜像

如果您不想使用提供的 A ctive 镜像 CodeCatalyst，则可以提供自定义运行时环境 Docker 镜像。如果要提供自定义镜像，请确保其中安装了 Git。镜像可以存放在 Docker Hub、Amazon 弹性容器注册表或任何公共存储库中。

要了解如何创建自定义 Docker 镜像，请参阅 Docker 文档中的容器化应用程序。

按照以下说明将您的自定义运行时环境 Docker 镜像分配给操作。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 AWS 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 镜像

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。
5. 选择 “视觉”。
6. 在工作流程图中，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，填写以下字段。

   运行时环境 Docker 镜像-可选

   指定存储图像的注册表。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）

     镜像存储在 Docker Hub 镜像注册表中。
   - 其他注册表（可视化编辑器）或`Other`（YAML 编辑器）

     镜像存储在自定义镜像注册表中。可以使用任何公开可用的注册表。
   - Amazon 弹性容器注册表（可视化编辑器）或`ECR`（YAML 编辑器）

该图像存储在 Amazon 弹性容器注册表镜像存储库中。要使用 Amazon ECR 存储库中的图像，此操作需要访问 Amazon ECR。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：

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

有关创建 IAM 角色的更多信息，请参阅 I A M 用户指南中的使用自定义信任策略（控制台）创建角色。

创建角色后，必须通过环境将其分配给操作。有关更多信息，请参阅 将环境、账户连接和 IAM 角色与工作流程操作关联。

ECR 图片网址、Docker Hub 图片或图片网址

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，请将镜像设置为 Docker Hub 镜像名称和可选标签。

  例如：postgres:latest
- 如果您使用的是亚马逊 ECR 注册表，请将映像设置为 Amazon ECR 注册表 URI。

  例如：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” 以在提交之前验证工作流程的 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 镜像

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
3. 选择编辑。
4. 选择 YAML。
5. 找到要为其分配运行时环境 Docker 镜像的操作。
6. 在操作中，添加一个Container章节以及底层Registry和Image属性。有关更多信息，请参阅中针对您的操作的Container、Registry和Image属性的描述。操作

7. （可选）选择“验证”以在提交之前验证工作流程的 YAML 代码。
8. 选择“提交”，输入提交消息，然后再次选择“提交”。

示例

以下示例说明如何将自定义运行时环境 Docker 镜像分配给工作流程定义文件中的操作。

主题

- 示例：使用自定义运行时环境 Docker 镜像通过 Amazon ECR 添加对 Node.js 18 的支持
- 示例：使用自定义运行时环境 Docker 镜像通过 Docker Hub 添加对 Node.js 18 的支持

示例：使用自定义运行时环境 Docker 镜像通过 Amazon ECR 添加对 Node.js 18 的支持

以下示例展示了如何使用自定义运行时环境 Docker 镜像通过 Amazon EC R 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 镜像通过 Docker Hub 添加对 Node.js 18 的支持

以下示例展示了如何使用自定义运行时环境 Docker 镜像通过 Docker Hub 添加对 Node.js 18 的支持。

```
Configuration:
  Container:
    Registry: DockerHub
    Image: node:18.18.2
```

将工作流程连接到源存储库

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其操作所需的文件。例如，工作流程操作可能连接到源存储库以获取应用程序源文件以生成应用程序。

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 和 Bitbucket 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

指定用于存储工作流程定义文件的源

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 或 Bitbucket 存储库，请参阅。为带有扩展程序的项目添加功能 CodeCatalyst

您的工作流程定义文件所在的源存储库由标签标识`WorkflowSource`。

> **Note**
>
> 首次提交工作流程定义文件时，您可以指定工作流程定义文件所在的源存储库。提交后，存储库和工作流程定义文件将永久链接在一起。在初始提交后更改存储库的唯一方法是在不同的存储库中重新创建工作流程。

指定用于存储工作流程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 C I/CD，然后选择工作流程。
4. 选择创建工作流程并创建工作流程。有关更多信息，请参阅 使用可视化编辑器创建工作流程。

   在工作流程创建过程中，系统会要求您指定要 CodeCatalyst存储工作流程定义文件的存储库。

指定工作流程操作将使用的来源

按照以下说明指定用于工作流程操作的源存储库。启动时，该操作将配置的源存储库中的文件捆绑到一个构件中，将该构件下载到运行该操作的运行时环境 Docker 镜像，然后使用下载的文件完成其处理。

> **Note**
>
> 目前，在一个工作流程操作中，您只能指定一个源存储库，即工作流程定义文件所在的源存储库（在.codecatalyst/workflows/目录中）。此源存储库由标签表示WorkflowSource。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要在其中指定来源的操作。
8. 选择输入。
9. 在 S ources 中-可选，执行以下操作：

   指定代表操作所需的源存储库的标签。当前，唯一支持的标签是WorkflowSource，它表示存储工作流程定义文件的源存储库。

   如果省略了源，则必须在下*action-name*/Inputs/Artifacts方指定至少一个输入对象。

   更多有关来源的信息，请参阅 将工作流程连接到源存储库。
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在操作中，添加类似于以下内容的代码：

```
action-name:
 Inputs:
   Sources:
     - WorkflowSource
```

有关更多信息，请参阅中对Sources属性的描述以工作流程 YAML 定义供您操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

引用源存储库中的文件

如果您的文件位于源存储库中，并且需要在其中一个工作流程操作中引用这些文件，请完成以下步骤。

> **Note**
>
> 另请参阅 在构件中引用文件。

引用源存储库中的文件

- 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
```

在前面的代码中，该操作在WorkflowSource源存储库根my-app目录的目录中查找并显示该file1.jar文件。

源生成的变量（“BranchName” 和 CommidId “”）

当您的工作流程运行时， CodeCatalyst 源代码会生成并设置 CommitId “BranchName” 和 “” 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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
CommitId	提交 ID 代表工作流程运行开始时仓库的状态。 例如：example3819261db00a3ab59468c8b 另请参阅：示例：引用 “CommitId” 预定义变量
BranchName	启动工作流程的分支的名称。 示例：main、feature/branch 、test-LiJuan 。 另请参阅：示例：引用 “BranchName” 预定义变量

使用工作流程发布和导入包

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。

一揽子包含：

- 一个名字（例如，webpack是一个流行的 npm 包的名称）
- 一个可选的命名空间（例如，@types在@types/node）

- 一组版本（例如、1.0.01.0.1、1.0.2）
- 包级元数据（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包。

有关软件包的更多信息，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

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。您的软件包存储库必须配置包格式，例如 npm。您也可以选择为所选软件包存储库添加一系列范围。

按照以下说明指定要用于工作流程操作的包配置。

Visual

指定操作将使用的包配置（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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要使用包存储库配置的操作。

8. 选择 “套餐”。
9. 从 “添加配置” 下拉菜单中，选择要用于工作流程操作的包配置。
10. 选择 “添加软件包存储库”。
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。

    有关软件包存储库的更多信息，请参阅Package 存储库。
12. （可选）在 S copes-可选中，指定要在软件包注册表中定义的范围序列。

    定义作用域时，将指定的包存储库配置为所有列出的作用域的注册表。如果通过 npm 客户端请求具有作用域的软件包，它将使用该存储库而不是默认存储库。每个作用域名称必须以 “@” 为前缀。

    如果Scopes省略，则将指定的包存储库配置为该操作使用的所有包的默认注册表。

    有关作用域的更多信息，请参阅Package 命名空间和作用域包。
13. 选择添加。
14. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在操作中，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，请参阅中对Packages属性的描述以生成和测试操作 YAML 定义供您操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

在工作流程中指定软件包存储库的示例

以下示例说明如何在工作流定义文件中引用包。

主题

- 示例：使用定义软件包 NpmConfiguration
- 示例：覆盖默认注册表
- 示例：覆盖软件包注册表中的作用域

示例：使用定义软件包 **NpmConfiguration**

以下示例说明如何在工作流程定义文件NpmConfiguration中使用定义包。

```
Actions:
  Build:
  Identifier: aws/build-beta@v1
  Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: main-repo
            - PackagesRepository: scoped-repo
              Scopes:
                - "@scope1"
```

这个例子这样配置 npm 客户端：

```
default: main-repo
@scope1: scoped-repo
```

在此示例中，定义了两个存储库。默认注册表按其main-repo定义进行设置，没有作用域。范围@scope1是在中PackageRegistries为配置的scoped-repo。

示例：覆盖默认注册表

以下示例向您展示如何覆盖默认注册表。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-repo-1
    - PackagesRepository: my-repo-2
    - PackagesRepository: my-repo-3
```

这个例子这样配置 npm 客户端：

```
default: my-repo-3
```

如果您指定多个默认存储库，则最后一个存储库将优先。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该存储库。这会覆盖存储库my-repo-1和。my-repo-2

示例：覆盖软件包注册表中的作用域

以下示例向您展示了如何覆盖软件包注册表中的作用域。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-default-repo
    - PackagesRepository: my-repo-1
      Scopes:
        - "@scope1"
        - "@scope2"
    - PackagesRepository: my-repo-2
      Scopes:
        - "@scope2"
```

这个例子这样配置 npm 客户端：

```
default: my-default-repo
@scope1: my-repo-1
@scope2: my-repo-2
```

如果您包含覆盖作用域，则最后一个存储库将优先。在本示例中，上次在中配置该范围`@scope2`的时间`PackageRegistries`是`my-repo-2`。这会覆盖为`@scope2`配置`my-repo-1`的范围。

使用工作流程调用 AWS 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程中。AWS Lambda 调用操作会调用您指定的 Lambda 函数。

除了AWS Lambda 调用您的函数外，invoke 操作还将从 Lambda 函数收到的响应负载中的每个顶级密钥转换为工作流程输出变量。然后，可以在后续的工作流程操作中引用这些变量。如果您不希望所有顶级键都转换为变量，则可以使用过滤器来指定确切的变量。有关更多信息，请参阅中的ResponseFilters属性描述“AWS Lambda 调用” 操作 YAML 定义。

何时使用此操作

如果您想向工作流程中添加封装在 Lambda 函数中并由 Lambda 函数执行的功能，请使用此操作。

例如，您可能希望您的工作流程在开始构建应用程序之前向 Slack 频道发送`Build started`通知。在这种情况下，您的工作流程将包括AWS Lambda 调用 Lambda 以发送 Slack 通知的调用操作，以及用于构建应用程序的构建操作。

再举一个例子，你可能希望你的工作流程在部署应用程序之前对其进行漏洞扫描。在这种情况下，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 来扫描漏洞，使用部署操作来部署扫描的应用程序。

主题

- 调用 Lambda 函数的工作流程示例
- 添加 “AWS Lambda 调用” 操作
- “AWS Lambda 调用” 操作生成的变量
- “AWS Lambda 调用” 操作 YAML 定义

调用 Lambda 函数的工作流程示例

> **Note**
>
> 以下示例工作流程仅用于说明目的，需要额外的设置工作才能正常运行。它旨在举一个示例，说明使用AWS Lambda 调用操作配置工作流程时的样子。

以下工作流程包括AWS Lambda 调用操作和部署操作。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。 AWS CloudFormation 该工作流由以下按顺序运行的构建块组成：

- 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程运行。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程。
- AWS Lambda 调用操作 (LambdaNotify)-触发后，此操作将调用指定 AWS 账户和区域（my-aws-account、和）中的 Notify-Start Lambda 函数。us-west-2调用时，Lambda 函数会发送一条 Slack 通知，表示部署已开始。
- 部署 AWS CloudFormation 堆栈操作 (Deploy)-AWS Lambda 调用操作完成后，部署 AWS 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 AWS CloudFormation 堆栈” 操作的更多信息，请参阅使用工作流程部署 AWS CloudFormation 堆栈。

```
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程中。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索AWS Lambda 调用操作，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

- 选择AWS Lambda 调用。将出现 “操作详细信息” 对话框。在此对话框中：
  - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。

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 定义。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 “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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
8. 从下拉列表中选择 A mazon CodeCatalyst。
9. 搜索AWS Lambda 调用操作，然后执行以下任一操作：
   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择AWS Lambda 调用。将出现 “操作详细信息” 对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“AWS Lambda 调用” 操作 YAML 定义。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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” 操作生成的变量

默认情况下，AWS Lambda 调用操作会在 Lambda 响应负载中为每个顶级密钥生成一个变量。

例如，如果响应负载如下所示：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 那么该操作将生成以下变量。

键	值
name	Saanvi
location	Seattle
department	{“公司”：“亚马逊”，“团队”：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 定义中的 ResponseFilters。

运行时由 “AWS Lambda invoke” 操作生成和设置的变量称为预定义变量。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

“AWS Lambda 调用” 操作 YAML 定义

以下是AWS Lambda 调用操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流程调用 AWS Lambda 函数。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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.
  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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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 指定操作的主版本、次要版本或补丁版本。

默认值：aws/lambda-invoke@v1。

对应的用户界面：工作流程图/ LambdaInvoke _nn/ aws/lambda-invoke @v1 标签

DependsOn

(*LambdaInvoke*/DependsOn)

（可选）

指定必须成功运行才能运行此操作的操作、操作组或门。

有关“依赖”功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入”选项卡/ 依赖- 可选

Compute

(*LambdaInvoke*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*LambdaInvoke*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

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程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队示例：`Linux.x86-64.Large`，`Linux.x86-64.XLarge`。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，`Fleet`则默认为`Linux.x86-64.Large`。

对应的 UI：“配置” 选项卡/ “计算舰队”

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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Inputs

(*LambdaInvoke*/Inputs)

（必需）

本`Inputs`节定义了AWS Lambda 调用操作在工作流程运行期间所需的数据。

> **Note**
>
> 每个AWS Lambda 调用操作只允许一个输入（源或构件）。变量不计入此总数。

相应的 UI：“输入” 选项卡

Sources

(*LambdaInvoke*/Inputs/Sources)

（如果RequestPayloadFile已提供，则为必填项）

如果要将请求负载 JSON 文件传递给AWS Lambda 调用操作，并且此有效负载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`。

如果您的请求负载文件不包含在源存储库中，则该文件必须位于另一个操作生成的项目中。

有关负载文件的更多信息，请参阅RequestPayloadFile。

> **Note**
>
> 您可以使用`RequestPayload`属性将有效负载的 JSON 代码直接添加到操作中，而不必指定负载文件。有关更多信息，请参阅 RequestPayload。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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）

如果要将请求负载 JSON 文件传递给AWS Lambda 调用操作，并且此有效负载文件包含在先前操作的输出项目中，请在此处指定该构件。

有关负载文件的更多信息，请参阅RequestPayloadFile。

> **Note**
>
> 您可以使用`RequestPayload`属性将有效负载的 JSON 代码直接添加到操作中，而不必指定负载文件。有关更多信息，请参阅 RequestPayload。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Variables - input

(*LambdaInvoke*/Inputs/Variables)

（可选）

指定一个名称/值对序列，这些对定义了要提供给操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的 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。

有关环境的更多信息，请参见部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC和创建环境。

对应的用户界面：配置选项卡/'环境/账户/角色'/环境

Name

(*LambdaInvoke*/Environment/Name)

（如果包含Environment，则为必填项）

指定要与操作关联的现有环境的名称。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/环境

Connections

(*LambdaInvoke*/Environment/Connections)

（如果包含Environment，则为必填项）

指定要与操作关联的账户连接。您最多可以在下方指定一个账户连接Environment。

有关账户关联的更多信息，请参阅允许在已连接的情况下访问 AWS 资源 AWS 账户。有关如何将账户关联与您的环境关联的信息，请参阅创建环境。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/连接

Name

(*LambdaInvoke*/Environment/Connections/Name)

（必需）

指定账户连接的名称。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/连接

Role

(*LambdaInvoke*/Environment/Connections/Role)

（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。请确保此角色包括：

- 以下权限策略：

> **Warning**
>
> 将权限限制为以下策略中显示的权限。使用具有更广泛权限的角色可能会带来安全风险。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:aws-region:aws-account:function:function-
name"
        }
```

```
    ]
}
```

- 以下自定义信任策略：

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

请确保此角色与您的账户关联关联。要了解有关将 IAM 角色与账户关联的更多信息，请参阅向账户连接添加 IAM 角色。

> **Note**
>
> 如果你愿意，你可以在这里指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名称。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。了解该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性较低的场景中使用此角色。

相应的 UI：“配置” 选项卡/'环境/连接/角色'/角色

Configuration

(*LambdaInvoke*/Configuration)

（必需）

您可以在其中定义操作的配置属性的部分。

相应的 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亚马逊资源名称 (ARN)。您可以在 Lambda 控制台中找到名称或 ARN。

> **Note**
>
> Lambda 函数所在的 AWS 账户可以与下指定的账户不同。`Connections:`

对应的用户界面：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，请参阅中的区域终端节点AWS一般参考。

相应的 UI：配置选项卡/ 目标存储桶- 可选

RequestPayload

(*LambdaInvoke*/Configuration/RequestPayload)

（可选）

如果要将请求负载传递给AWS Lambda 调用操作，请在此处以 JSON 格式指定请求负载。

请求负载示例：

```
'{ "key": "value" }'
```

如果您不想将请求负载传递给 Lambda 函数，请省略此属性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同时指定两者。

有关请求负载的更多信息，请参阅 AWS Lambda API 参考中的调用主题。

相应的 UI：“配置” 选项卡/ “请求有效负载”- 可选

RequestPayloadFile

(*LambdaInvoke*/Configuration/RequestPayloadFile)

（可选）

如果要将请求负载传递给 AWS Lambda invoke 操作，请在此处指定此请求负载文件的路径。该文件必须采用 JSON 格式。

请求负载文件可以驻留在源存储库中，也可以位于先前操作的对象中。文件路径是相对于源存储库或工件根目录的。

如果您不想将请求负载传递给 Lambda 函数，请省略此属性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同时指定两者。

有关请求负载文件的更多信息，请参阅 AWS Lambda API 参考中的调用主题。

对应的用户界面：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AWS Lambda 调用的 AWS Lambda 函数失败，是否也要将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程中的后续操作。

默认情况下，如果 Lambda 函数失败（在可视化编辑器或 `false` YAML 编辑器中为“关闭”），则操作失败。

相应的 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

默认为 Tai l。

有关日志类型的更多信息，请参阅 AWS Lambda API 参考中的调用主题。

有关查看日志的详细信息，请参阅 查看工作流程运行状态和详细信息。

对应的用户界面：配置选项卡/ 日志类型

ResponseFilters

(*LambdaInvoke*/Configuration/ResponseFilters)

（可选）

在 Lambda 响应负载中指定要转换为输出变量的密钥。然后，您可以在工作流程的后续操作中引用输出变量。有关中变量的更多信息 CodeCatalyst，请参阅在工作流程中配置和使用变量。

例如，如果您的响应负载如下所示：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
```

```
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 你的响应过滤器如下所示：

```
Configuration:
  ...
  ResponseFilters: '{"name": ".name", "company": ".department.company"}'
```

... 然后该操作生成以下输出变量：

键	值
name	Saanvi
company	Amazon

然后，您可以在后续操作中引用name和company变量。

如果您未在中指定任何密钥ResponseFilters，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” 操作生成的变量 。

考虑使用响应过滤器将生成的输出变量限制为仅包含您实际要使用的变量。

相应的 UI：“配置” 选项卡/ “响应过滤器”- 可选

Outputs

(*LambdaInvoke*/Outputs)

（可选）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts

(*LambdaInvoke*/Outputs/Artifacts)

（可选）

指定操作生成的对象。您可以在其他操作中引用这些构件作为输入。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

对应的用户界面：“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 Lambda 函数返回的 Lambda 响应负载的工件的名称。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 有关查看日志的详细信息，请参阅 查看工作流程运行状态和详细信息。

对应的用户界面：“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 响应负载文件。

相应的用户界面：输出选项卡/构件/构建生成的文件

使用工作流程修改 Amazon ECS 任务定义文件

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” 操作添加到您的工作流程中。此操作使用工作流程在运行时提供的 Docker 映像名称更新任务定义文件中的图像字段。

> **Note**
>
> 您也可以使用此操作使用环境变量更新任务定义的`environment`字段。

何时使用此操作

如果您的工作流程使用动态内容（例如提交 ID 或时间戳）来构建 Docker 镜像并对其进行标记，请使用此方法。

如果您的任务定义文件包含始终保持不变的图像值，请不要使用此操作。在这种情况下，您可以手动将图像的名称输入到任务定义文件中。

“渲染 Amazon ECS 任务定义” 操作的工作原理

您必须使用渲染 Amazon ECS 任务定义操作以及工作流程中的构建和部署到 Amazon ECS 操作。这些操作共同起作用如下：

1. 构建操作会构建 Docker 镜像，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，您的构建操作可能如下所示：

```
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，该`docker build -t`指令指示构建 Docker 镜像，并在操作运行时使用提交 ID 对其进行标记。生成的图像名称可能如下所示：

   `MyDockerImage:a37bd7e`

2. “渲染 Amazon ECS 任务定义” 操作会将动态生成的图像名称添加到您的任务定义文件中，如下所示：`MyDockerImage:a37bd7e`

```
{
    "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
                }
            ]
        }
    ],
...
}
```

或者，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 弹性容器服务开发人员指南中的指定环境变量。

3. “部署到 Amazon ECS” 操作将更新的任务定义文件注册到 Amazon ECS。注册更新的任务定义文件会将新映像部署`MyDockerImage:a37bd7e`到 Amazon ECS 中。

主题

- 修改 Amazon ECS 任务定义文件的工作流程示例

- 添加 “渲染 Amazon ECS 任务定义” 操作
- 查看更新的任务定义文件
- “渲染 Amazon ECS 任务定义” 操作生成的变量
- “渲染 Amazon ECS 任务定义” 操作 YAML 定义参考

修改 Amazon ECS 任务定义文件的工作流程示例

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程示例。该工作流程的目的是构建 Docker 映像并将其部署到您的 Amazon ECS 集群中。该工作流由以下按顺序运行的构建块组成：

- 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程运行。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程。
- 构建操作 (BuildDocker)-触发后，该操作使用 Dockerfile 构建 Docker 镜像，使用提交 ID 对其进行标记，然后将映像推送到 Amazon ECR。有关生成操作的更多信息，请参阅使用工作流程进行构建。
- Render 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taskdef.json位于源存储库根目录中的现有存储库。它使用新的文件名 (task-definition-*random-string*.json) 保存更新的文件，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的变量，task-definition并将其设置为新任务定义文件的名称。对象和变量将用于部署操作，接下来是部署操作。
- 部署到 Amazon ECS 操作 (DeployToECS) — 完成 “渲染 Amazon ECS” 任务定义操作后，“部署到 Amazon ECS” 操作将查找渲染操作生成的输出项目 (TaskDefArtifact)，在其中找到task-definition-*random-string*.json文件，然后将其注册到您的 Amazon ECS 服务。然后，Amazon ECS 服务按照task-definition-*random-string*.json文件中的说明在您的 Amazon ECS 集群中运行 Amazon ECS 任务和关联的 Docker 镜像容器。

```
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

按照以下说明将 “渲染 Amazon ECS” 任务定义操作添加到您的工作流程中。

先决条件

在开始之前，请确保您的工作流程包含动态生成 Docker 映像的生成操作。有关详细信息，请参阅前面的示例工作流程。

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在左上角，选择 + 操作以打开操作目录。
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，然后执行以下任一操作：
   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or
   - 选择渲染 Amazon ECS 任务定义。出现操作详细信息对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
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 定义参考。本参考提供了有关在 YAML 和可视编辑器中显示的每个字段（以及相应的 YAML 属性值）的详细信息。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。

4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在左上角，选择 + 操作以打开操作目录。
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，然后执行以下任一操作：

   - 选择加号 (+) 将操作添加到工作流程图中，然后打开其配置窗格。

     Or

   - 选择渲染 Amazon ECS 任务定义。出现操作详细信息对话框。在此对话框中：
     - （可选）选择 “查看源代码” 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程图中，然后打开其配置窗格。
10. 根据需要修改 YAML 代码中的属性。中提供了每个可用属性的说明“渲染 Amazon ECS 任务定义” 操作 YAML 定义参考。
11. （可选）选择 “验证” 以在提交之前验证工作流程的 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

添加渲染操作后，按照中的说明将 “部署到 Amazon ECS” 操作添加到您的工作流程中使用工作流程将应用程序部署到 Amazon 弹性容器服务 (ECS)。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对象。它包含更新的任务定义文件。

   有关构件的更多信息，请参阅 使用构件在工作流程中的操作之间共享数据。
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 *-n* ame 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，请参阅在工作流程中配置和使用变量。

   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程。

查看更新的任务定义文件

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” 操作处理完毕后。

1. 查找包含已完成渲染操作的运行：
   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 C I/CD，然后选择工作流程。
   d. 选择包含渲染操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
   e. 选择包含已完成渲染操作的运行。
2. 在工作流程图中，选择渲染操作。
3. 选择 “输出”。
4. 选择变量。
5. 显示任务定义文件名。它看起来类似于`task-definition--259-0a2r7gxlTF5X-.json`。

查看更新的任务定义文件的内容

1. 查找包含已完成渲染操作的运行：
   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 C I/CD，然后选择工作流程。
   d. 选择包含渲染操作的工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
   e. 选择包含已完成渲染操作的运行。
2. 在工作流程运行中，在顶部，在 Visual 和 YAML 旁边，选择工作流程输出。
3. 在 “构件” 部分，选择包含已更新的任务定义文件的对象旁边的 “下载”。此构件的 “制作者” 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” 操作生成的变量

“渲染 Amazon ECS 任务定义” 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。

有关在工作流程中引用这些变量的信息，请参阅使用预定义的变量。

键	值
任务定义	为通过 “渲染 Amazon ECS 任务定义” 操作更新的任务定义文件指定的名称。名称遵循以下格式：`task-definition-` *`random-string`* `.json`。 例如：`task-definition--259-0a2r7gxlTF5Xr.json`

“渲染 Amazon ECS 任务定义” 操作 YAML 定义参考

以下是 “渲染 Amazon ECS” 任务定义操作的 YAML 定义。要了解如何使用此操作，请参阅使用工作流程修改 Amazon ECS 任务定义文件。

此操作定义作为一个部分存在于更广泛的工作流程定义文件中。有关此文件的更多信息，请参阅 工作流程 YAML 定义。

> **Note**
>
> 接下来的大多数 YAML 属性在可视化编辑器中都有相应的 UI 元素。要查找用户界面元素，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.
```

```yaml
  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。所有操作名称在工作流程中必须是唯一的。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启用特殊字符和空格。

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 指定操作的主版本、次要版本或补丁版本。

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运行此操作的操作、操作组或门。

有关 “依赖” 功能的更多信息，请参阅。将操作配置为依赖于其他操作

对应的用户界面：“输入” 选项卡/ 依赖- 可选

Compute

(*ECSRenderTaskDefinition*/Compute)

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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。

对应的用户界面：无

Type

(*ECSRenderTaskDefinition*/Compute/Type)

（如果包含Compute，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

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程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队示例：Linux.x86-64.Large，Linux.x86-64.XLarge。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

相应的 UI：“配置” 选项卡/ “计算舰队”

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，最大值如中所述工作流程配额。默认超时与最大超时相同。

相应的 UI：“配置” 选项卡/ “超时”- 可选

Inputs

(*ECSRenderTaskDefinition*/Inputs)

（可选）

本Inputs节定义了工作流程运行期间ECSRenderTaskDefinition所需的数据。

> **Note**
>
> 每个 “渲染 Amazon ECS” 任务定义操作只允许输入一个输入（源或构件）。变量不计入此总数。

相应的 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，则为必填项）

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，则该文件必须位于另一个操作生成的对象中。

更多有关来源的信息，请参阅 将工作流程连接到源存储库。

相应的 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先前操作的输出对象中，则为必填项）

如果要部署的任务定义文件包含在先前操作生成的对象中，请在此处指定该对象。如果任务定义文件不包含在构件中，则该文件必须位于源存储库中。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“配置” 选项卡/ 工件- 可选

Variables - input

(*ECSRenderTaskDefinition*/Inputs/Variables)

（必需）

指定一系列名称/值对，这些对定义要用于操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。

相应的 UI：“输入” 选项卡/ “变量”- 可选

Configuration

(*ECSRenderTaskDefinition*/Configuration)

（必需）

您可以在其中定义操作的配置属性的部分。

相应的 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。如果文件位于您的源存储库中，则该路径是相对于源存储库根文件夹的路径。如果您的文件位于先前工作流程操作的对象中，则该路径是相对于对象根文件夹的路径。有关任务定义文件的更多信息，请参阅《Amazon 弹性容器服务开发者指南》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）

指定要在其中运行 Docker 镜像的容器的名称。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 弹性容器服务开发者指南》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。该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字段中已经存在一个值，则该操作会将其覆盖。可以在图像名称中包含变量。

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 comm *it-id* 是工作流程在运行时生成的提交 ID。

#####my-ecr-repo/image-repo:$(date +%m-%d-%y-%H-%m-%s)####### my-ecr-repo/image-repo: ## +%m-%d-%y-%h-%m-%s #############my-ecr-repo########### (ECR) # URI### +% m-%d-%y-%h-%s ################### month-day-year-hour-minute-second

有关该image字段的更多信息，请参阅 Amazon 弹性容器服务开发者指南中的图片。有关变量的更多信息，请参阅在工作流程中配置和使用变量。

对应的 UI：“配置” 选项卡/ 镜像名称

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。该操作会将变量添加到任务定义文件中的 “containerDefinitions,environment” 字段。如果文件中已存在变量，

则该操作将覆盖现有变量的值并添加任何新变量。有关 Amazon ECS 环境变量的更多信息，请参阅《亚马逊弹性容器服务开发人员指南》中的指定环境变量。

相应的 UI：配置选项卡/ 环境变量- 可选

Outputs

(*ECSRenderTaskDefinition*/Outputs)

（必需）

定义操作在工作流程运行期间输出的数据。

相应的 UI：“输出” 选项卡

Artifacts

(*ECSRenderTaskDefinition*/Outputs/Artifacts)

（必需）

指定操作生成的对象。您可以在其他操作中引用这些构件作为输入。

有关构件的更多信息（包括示例），请参阅使用构件在工作流程中的操作之间共享数据。

相应的 UI：“输出” 选项卡/ 工件

Name

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Name)

（必需）

指定将包含更新的任务定义文件的对象的名称。默认值为 `MyTaskDefinitionArtifact`。然后，您必须将此项目指定为 “部署到 Amazon ECS” 操作的输入。要了解如何将此项目添加为 “部署到 Amazon ECS” 操作的输入，请参阅修改 Amazon ECS 任务定义文件的工作流程示例。

对应的用户界面：“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-*以便将更新的任务定义文件（以task-definition-开头）包括在内。

对应的用户界面：“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的任务定义文件的名称（例如task-definition-*random-string*.json）。然后，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” 操作中，请参阅修改 Amazon ECS 任务定义文件的工作流程示例。

默认：task-definition

相应的 UI：“输出” 选项卡/变量/ “名称” 字段

在工作流程中配置和使用变量

变量是一个键值对，其中包含可在工作流程中引用的信息。 CodeCatalyst

您可以在工作流程中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-这些是工作流程自动发出的键值对。您无需对其进行定义。

> **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 输出参数以便其他操作可以使用它和引用 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。有两种类型：

- 纯文本变量，或者简称变量 — 这些是您在工作流程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。密钥（名称）是一个公共标签，其值包含您要保密的信息。您只能在工作流程定义文件中指定密钥。在工作流程定义文件中，使用密码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。

主题

定义变量

您可以通过两种方式定义变量：

- 在工作流程操作`Inputs`部分 — 请参阅 “输入” 部分中的定义变量
- 在工作流程操作`Steps`部分 — 请参阅 “步骤” 部分中的 “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，请参阅用户定义变量的示例。

Visual

在 “输入” 部分定义变量（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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要在其中设置变量的操作。
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：

   指定一系列名称/值对，这些对定义要用于操作的输入变量。变量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。不能使用引号在变量名中启用特殊字符和空格。

   有关变量的更多信息（包括示例），请参阅在工作流程中配置和使用变量。
10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流程操作中，添加类似于以下内容的代码：

```
action-name:
  Inputs:
    Variables:
      - Name: variable-name
        Value: variable-value
```

有关更多示例，请参阅用户定义变量的示例。有关更多信息，工作流程 YAML 定义请参阅您的操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择要在其中设置变量的操作。
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 bash 命令Steps中。
- 要隐式定义变量，请在操作Steps部分引用的文件中指定该变量。

  有关示例，请参阅用户定义变量的示例。有关更多信息，工作流程 YAML 定义请参阅操作的。

10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在工作流程操作中，在操作Steps部分中以显式或隐式方式定义变量。

   - 要显式定义变量，请将其直接包含在 bash 命令Steps中。
   - 要隐式定义变量，请在操作Steps部分引用的文件中指定该变量。

   有关示例，请参阅用户定义变量的示例。有关更多信息，工作流程 YAML 定义请参阅操作的。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

定义一个秘密

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 在工作流程中配置和使用密钥。

例如，你可以定义一个如下所示的密钥：

- 名称（密钥）：**my-password**
- 值：**^*H3#!b9**

定义密钥后，您可以在工作流程定义文件中指定密钥的密钥 (**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可以使用它

按照以下说明从动作中导出变量，以便可以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，请仔细阅读随后的可视化编辑器说明，并查看该操作是否包含 “输出” 选项卡上的 “变量” 按钮。如果是，则支持导出变量。
- 要从 GitHub 动作中导出变量，请参阅导出 GitHub 输出参数以便其他操作可以使用它。

先决条件

确保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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，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：

   指定要导出操作的变量的名称。此变量必须已经在同一操作的Inputs或Steps部分中定义。

10. （可选）选择 “验证” 以在提交之前验证工作流程的 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在要从中导出变量的操作中，添加类似于以下内容的代码：

```
action-name:
  Outputs:
    Variables:
      - Name: variable-name
```

有关更多示例，请参阅用户定义变量的示例。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

在定义变量的动作中引用变量

使用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。

先决条件

确保您已经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

在定义变量的动作中引用该变量

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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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，请参阅用户定义变量的示例。有关更多信息，请参阅中有关您的操作的参考信息工作流程 YAML 定义。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

通过另一个操作引用变量输出

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。

先决条件

确保已导出要引用的变量。有关更多信息，请参阅 导出变量以便其他操作可以使用它。

Visual

不可用。选择 YAML 以查看 YAML 说明。

YAML

通过其他操作引用变量输出（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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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}
   ```

   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。

   - *actionnam* e，其中包含输出变量的操作的名称。

- 带有@@ ####的变量名。

例如：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: ${MyFirstAction.TIMESTAMP}
```

有关更多示例，请参阅用户定义变量的示例。有关更多信息，工作流程 YAML 定义请参阅您的操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

引用机密

有关在工作流程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。

用户定义变量的示例

以下示例说明如何在工作流定义文件中定义和引用变量。

示例

- 示例：使用 Inputs 属性定义变量
- 示例：使用 Steps 属性定义变量
- 示例：使用 Outputs 属性导出变量
- 示例：引用在同一操作中定义的变量
- 示例：引用在另一个操作中定义的变量
- 示例：引用密钥

示例：使用 Inputs 属性定义变量

以下示例向您展示了如何在一Inputs节中定义两个变量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

以下示例向您展示了如何在该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

以下示例向您展示如何定义两个变量TIMESTAMP、REPOSITORY-URI和，以及如何使用Outputs部分将其导出。

```
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

以下示例说明如何在中指定VAR1变量MyBuildAction，然后使用在同一个操作中引用该变量$VAR1。

```
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: VAR1
          Value: my-value
    Configuration:
      Steps:
        - Run: $VAR1
```

示例：引用在另一个操作中定义的变量

以下示例向您展示了如何在中指定TIMESTAMP变量BuildActionA，使用该Outputs属性将其导出，然后在 using 中BuildActionB引用该变量${BuildActionA.TIMESTAMP}。

```
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

以下示例向您展示如何引用my-password密钥。my-password这是秘密的钥匙。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 在工作流程中配置和使用密钥。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: curl -u LiJuan:${Secrets.my-password} https://example.com
```

使用预定义的变量

预定义变量是键值对，由工作流程自动发出，可供您在工作流程操作中使用。

您可以在任何工作流程操作中使用预定义的变量。

主题

引用预定义变量

按照以下说明引用预定义变量。

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 确定您的工作流程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. 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 CodeCatalyst 操作中，使用以下语法添加预定义的变量引用：

```
${action-group-name.action-name-or-WorkflowSource.variable-name}
```

替换：

- *action-group-name*用行动组的名字。

> **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-or-WorkflowSource* 搭配：

  输出变量的操作的名称。

  或者

  WorkflowSource，如果变量是BranchName或CommitId变量。

- 带有@@ *####*的变量名。

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，请参阅预定义变量的示例。有关更多信息，工作流程 YAML 定义请参阅您的操作。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

确定您的工作流程会发出哪些预定义变量

您可以通过两种方式确定您的工作流程会发出哪些预定义变量：

- 运行一次工作流程。运行完成后，工作流程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 查看工作流程运行状态和详细信息。
- 请咨询预定义变量列表。此参考文献列出了每个预定义变量的变量名（键）和值。

> **Note**
>
> 中列出了工作流程变量的最大总大小工作流程配额。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。

预定义变量的示例

以下示例说明如何在工作流定义文件中引用预定义变量。

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

以下示例向您展示了如何在`MyBuildAction`操作中引用`CommitId`预定义变量。`CommitId`变量由自动输出 CodeCatalyst。

尽管该示例显示了生成操作中使用的变量，但您可以在任何操作CommitId中使用。

```
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

以下示例向您展示了如何在CDKDeploy操作中引用BranchName预定义变量。BranchName变量由自动输出 CodeCatalyst。

尽管该示例显示了AWS CDK 部署操作中使用的变量，但您可以在任何操作BranchName中使用。

```
CDKDeploy:
    Identifier: aws/cdk-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: app-stack-${WorkflowSource.BranchName}
```

预定义变量列表

请参阅以下章节，查看 CodeCatalyst 操作自动生成的预定义变量。

> **Note**
>
> 此列表仅包括 CodeCatalyst 源发出的预定义变量和CodeCatalyst 操作。如果您使用的是其他类型的操作，例如 GitHub 操作或 CodeCatalyst 实验室操作，请参阅确定您的工作流程会发出哪些预定义变量。

列表

> **Note**
>
> 并非所有 CodeCatalyst 操作都会生成预定义的变量。如果该操作不在列表中，则它不会生成变量。

- 源生成的变量（“BranchName” 和 CommidId “”）
- “部署 AWS CloudFormation 堆栈” 操作生成的变量
- “部署到 Amazon ECS” 操作生成的变量
- “部署到 Kubernetes 集群” 操作生成的变量
- “AWS CDK 部署” 操作生成的变量
- “AWS CDK bootstrap” 操作生成的变量
- “AWS Lambda 调用” 操作生成的变量
- “渲染 Amazon ECS 任务定义” 操作生成的变量

在工作流程中配置和使用密钥

有时您可能需要在工作流程中使用敏感数据，例如身份验证凭据。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它们。同样，不应在任何工作流程定义中直接使用这些值，因为它们将作为文件显示在存储库中。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，每个操作最多可以有五个密钥。

> **Note**
>
> 密钥只能用于替换工作流程定义文件中的密码和敏感信息。

主题

- 创建密钥
- 编辑密钥
- 使用密钥
- 删除密钥

创建密钥

使用以下步骤创建密钥。该密钥包含您想要隐藏的敏感信息。

> **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 C I/CD，然后选择密钥。
3. 选择创建密钥。
4. 输入以下信息：

   名称

   输入您的密钥的名称。

   值

   输入密钥的值。这是您想要隐藏的敏感信息。默认情况下，不显示该值。要显示该值，请选择“显示值”。

   描述

   （可选）输入您的密钥的描述。
5. 选择创建。

编辑密钥

使用以下步骤编辑密钥。

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 C I/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。这是您要隐藏的值。默认情况下，不显示该值。

   描述

   （可选）输入您的密钥的描述。
6. 选择保存。

使用密钥

要在工作流程操作中使用密钥，必须获取密钥的参考标识符并在工作流程操作中使用该标识符。

主题

获取密钥的标识符

使用以下步骤获取密钥的引用标识符。您需要将此标识符添加到您的工作流程中。

获取密钥的参考标识符

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择密钥。
3. 在密钥列表中，找到您要使用的密钥。
4. 在参考编号列中，复制密钥的标识符。以下是参考编号的语法：

```
${Secrets.<name>}
```

在工作流程中引用密钥

使用以下步骤在工作流程中引用密钥。

引用机密

1. 在导航窗格中，选择 C I/CD，然后选择工作流程。
2. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
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命令中使用存储为机密的用户名和密码，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” 以在提交之前验证工作流程的 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

使用以下步骤删除密钥和机密引用标识符。

> **Note**
>
> 在删除密钥之前，我们建议您从所有工作流程操作中移除该密钥的参考标识符。如果您删除密钥而不删除引用标识符，则该操作将在下次运行时失败。

从工作流程中删除密钥的参考标识符

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/CD，然后选择工作流程。
3. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
4. 选择编辑。
5. 选择 YAML。
6. 在工作流程中搜索以下字符串：

```
${Secrets.
```

这将找到所有机密的所有引用标识符。

7. 删除所选密钥的引用标识符，或将其替换为纯文本值。
8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。

删除秘密

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择 C I/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。

查看工作流程状态

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程的运行状态，该状态与工作流程状态不同。有关更多信息，请参阅 查看工作流程运行状态和详细信息。

有关可能的工作流程状态列表，请参阅工作流程状态。

查看工作流程的状态

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 C I/CD，然后选择工作流程。
4. 找到要查看其状态的工作流程。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。

   状态与工作流程一起显示在列表中。
5. （可选）选择工作流的名称，然后找到 “工作流定义” 字段。它显示工作流程状态。

工作流程配额

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，请参阅的配额 CodeCatalyst。

每个空间的最大工作流程数	800
工作流程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
单源事件中处理的最大文件数	4,000
每个空间的最大活跃舰队数量	10
每个队列的最大活跃计算实例数	20
每个操作的最大输入构件数	10
每个操作的最大输出对象数	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。

在工作流程运行期间保留生成工件的最大天数	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
每个代码覆盖率报告的最大文件数	20000
每份报告的最大软件组成分析结果数	20000
每个静态分析报告的最大文件数	20000
每个空间的最大并发工作流程运行次数	100
每个工作流程的最大操作数	50
每个工作流程同时运行的最大操作数	50
每个空间同时运行的最大操作数	200
操作可以运行的最大时间	对于生成和测试操作，超时时间为 8 小时。 对于所有其他操作，超时时间为 1 小时。
每个空间关联的最大环境 AWS 账户 数	5000
每次操作的最大密钥数	5
每个空间的最大密钥数	500,000

工作流程运行状态

工作流程运行可能处于以下状态之一：

- 成功-工作流程运行已成功处理。
- 失败-工作流程运行中的一个或多个操作失败。
- 进行中-工作流程运行当前正在处理中。
- 已停止 — 有人在工作流程运行期间停止了工作流程的运行。
- 正在@@ 停止-工作流程运行当前正在停止。

- 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-仅在配置了被取代的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它。

工作流程状态

工作流程可以具有以下状态之一：

- 有效 -工作流程可运行，可通过触发器激活。

  要将工作流程标记为有效，必须满足以下两个条件：

  - 工作流程定义文件必须有效。
  - 工作流程必须没有触发器、没有推送触发器或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 分支时的触发注意事项。
- 无效-工作流程的定义文件无效。工作流程不能手动运行，也不能通过触发器自动运行。无效的工作流程与工作流定义一起在 CodeCatalyst 控制台中显示 ***n*** 条错误消息（或类似消息）。

  要将工作流程标记为无效，必须满足以下条件：

  - 工作流程定义文件一定配置错误。

    要修复配置错误的工作流程定义文件，请参阅如何修复 “工作流定义有 *n* #错误” 错误？。
- 非活动-工作流定义有效，但不能手动运行，也不能通过触发器自动运行。

  要将工作流程标记为非活动状态，必须满足以下两个条件：

  - 工作流程定义文件必须有效。
  - 工作流定义文件必须包含一个推送触发器，该触发器指定的分支与工作流定义文件当前所在的分支不同。有关更多信息，请参阅 分支时的触发注意事项。

    要将工作流程从 “非活动” 切换为 “活动”，请参阅如何修复 “工作流程处于非活动状态” 消息？。

> **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程运行时被发现。

工作流程 YAML 定义

以下是工作流程定义文件的参考文档。

工作流程定义文件是描述您的工作流程的 YAML 文件。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是.yml 或.yaml。

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相应的 UI 元素。要查找用户界面元素，请使用 Ctrl+F。 该元素将与其关联的 YAML 属性一起列出。

主题

工作流程定义文件示例

以下是一个简单的工作流程定义文件示例。它包括一些顶级属性、一个Triggers部分和一个包含两个操作的Actions部分：Build和Test。有关更多信息，请参阅 关于工作流程定义文件。

```
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
```

语法指南和惯例

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 语法指南

工作流程定义文件是用 YAML 编写的，并遵循 YAML 1.1 规范，因此该规范中允许的任何内容也允许在工作流程 YAML 中使用。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：工作流程定义文件区分大小写，因此请务必使用本文档中显示的大小写。
- 特殊字符：我们建议在包含以下任意特殊字符的属性值周围使用引号或双引号：、、、、&、、、、、、、、<、>
  {、、、、、、、、}、、[、、]、、、、、、*、、#、、?、、|、、-、、、、、、、、、、、、=、、!、、%
  ，

  如果不包括引号，则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（而不是属性值）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来启用属性名称中的特殊字符和空格。

  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（例如\n或\t），请遵循以下准则：
  - 使用单引号将转义码作为字符串返回。例如，`'my string \n my string'`，返回字符串`my string \n my string`。
  - 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
```

- 评论:前面注释为. #

  例如：

```
Name: MyWorkflow
# This is a comment.
SchemaVersion: 1.0
```

- Triple dash (**`---`**)：不要`---`在你的 YAML 代码中使用。 CodeCatalyst 忽略之后的所有内容。`---`

命名约定

在本指南中，我们使用术语属性和章节来指代工作流程定义文件中的主要项目。

- 属性是任何包含冒号 (**`:`**) 的项目。例如，在以下代码片段中，以下所有属性均为属性：`Name`、`SchemaVersion`、`RunMode`、`TriggersType`、和`Branches`。
- 分区是任何具有子属性的属性。在以下代码片段中，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。工作流程名称显示在工作流程列表中，并在通知和日志中提及。工作流程名称和工作流定义文件名可以匹配，也可以用不同的方式命名。工作流程名称不必是唯一的。工作流程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程名称中启用特殊字符和空格。

对应的用户界面：可视化编辑器/工作流程属性/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`。

对应的用户界面：无

RunMode

（可选）

如何 CodeCatalyst 处理多次运行。您可以使用以下值之一：

- `QUEUED`— 多个运行排队并一个接一个地运行。一个队列中最多可以运行 50 次。
- `SUPERSEDED`— 多个运行排队并一个接一个地运行。一个队列只能运行一次，因此，如果两次运行最终在同一个队列中，则较晚的运行将取代（接管）先前的运行，而较早的运行将被取消。
- `PARALLEL`— 同时进行多次运行。

如果省略此属性，则默认为`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：可视化编辑器/工作流属性/高级/运行模式

Compute

（可选）

用于运行工作流程操作的计算引擎。您可以在工作流程级别或操作级别指定计算，但不能同时指定两者。在工作流级别指定时，计算配置将应用于工作流中定义的所有操作。在工作流程级别，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，请参阅为工作流程配置计算和运行时环境 Docker 镜像。

对应的用户界面：无

```
Name: MyWorkflow
```

```
SchemaVersion: 1.0
...
Compute:
  Type: EC2 | Lambda
  Fleet: fleet-name
  SharedInstance: true | false
```

Type

(Compute/Type)

（如果Compute已设置，则为必填项）

计算引擎的类型。您可以使用以下值之一：

- EC2（可视化编辑器）或EC2（YAML 编辑器）

  针对动作运行期间的灵活性进行了优化。
- Lambda（可视化编辑器）或Lambda（YAML 编辑器）

  优化了动作启动速度。

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：可视化编辑器/工作流属性/高级/计算类型

Fleet

(Compute/Fleet)

（可选）

指定将运行您的工作流程或工作流程操作的计算机或机群。对于按需队列，当操作开始时，工作流程会配置所需的资源，操作完成后计算机就会被销毁。按需车队的示例：Linux.x86-64.Large，Linux.x86-64.XLarge。有关按需队列的更多信息，请参阅按需车队房产。

使用已配置的队列，您可以配置一组专用计算机来运行您的工作流程操作。这些计算机处于闲置状态，可以立即处理操作。有关已配置队列的更多信息，请参阅。已配置的舰队属性

如果省略，Fleet则默认为Linux.x86-64.Large。

有关计算队列的更多信息，请参阅计算实例集。

相应的用户界面：可视化编辑器/工作流属性/高级/计算队列

SharedInstance

(Compute/SharedInstance)

（可选）

为您的操作指定计算共享功能。使用计算共享，工作流程中的操作在同一个实例上运行（运行时环境映像）。您可以使用以下值之一：

- TRUE表示运行时环境图像在工作流程操作之间共享。
- FALSE意味着启动一个单独的运行时环境映像并将其用于工作流程中的每个操作，因此，如果没有额外的配置，您就无法共享工件和变量等资源。

有关计算共享的更多信息，请参阅跨操作共享计算。

对应的用户界面：无

Triggers

（可选）

此工作流程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程。

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程。

相应的用户界面：可视化编辑器/工作流程图/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)

（如果Triggers已设置，则为必填项）

指定触发器的类型。您可以使用以下值之一：

- 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程运行。工作流程运行将使用您要推送到的分支（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程运行。工作流程运行将使用您要从中提取的分支（即源分支）中的文件。
- 日程安排（可视化编辑器）或SCHEDULE（YAML 编辑器）

  计划触发器按您指定的 cron 表达式定义的计划启动工作流运行。将使用分支的文件为源存储库中的每个分支启动单独的工作流程运行。（要限制触发器激活的分支，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，请遵循以下准则：

  - 每个工作流程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，请参阅触发器示例。

对应的用户界面：可视化编辑器/工作流程图/触发器/触发器类型

Events

(Triggers/Events)

（如果触发器设置为，Type则为必填项PULLREQUEST）

指定将启动工作流程运行的拉取请求事件的类型。以下是有效值：

- 拉取请求已创建（可视化编辑器）或OPEN（YAML 编辑器）

  创建拉取请求后，工作流程就会启动。
- 拉取请求已关闭（可视化编辑器）或CLOSED（YAML 编辑器）

  当拉取请求关闭时，工作流程就会开始运行。CLOSED事件的行为很棘手，最好通过一个例子来理解。请参阅示例：带有拉动、分支和 “CLOSED” 事件的触发器了解更多信息。
- 对拉取请求（可视化编辑器）或**REVISION**（YAML 编辑器）进行了新的修订

  创建拉取请求的修订版后，工作流程运行即开始。第一个修订版是在创建拉取请求时创建的。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版。如果您在拉取请求触发器中包含该REVISION事件，则可以省略该OPEN事件，因为REVISION是的超集。OPEN

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，请参阅触发器示例。

相应的用户界面：可视化编辑器/工作流程图/触发器/拉取请求事件

Branches

(Triggers/Branches)

（可选）

指定触发器监控的源存储库中的分支，以便知道何时开始工作流程运行。你可以使用正则表达式模式来定义你的分支名称。例如，用于匹配所有main.*以开头的分支main。

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，请指定要推送到的分支，即目标分支。将使用匹配分支中的文件，为每个匹配的分支启动一个工作流程。

  示例：main.*、mainline
- 对于拉取请求触发器，请指定要推送到的分支，即目标分支。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）中的源文件启动一个工作流程运行。

  示例：main.*、mainline、v1\-.*（匹配以开头的分支v1-）
- 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。将使用匹配分支中的工作流程定义文件和源文件，为每个匹配的分支启动一个工作流程。

  示例：main.*、version\-1\.0

> **Note**
>
> 如果您未指定分支，则触发器会监视源存储库中的所有分支，并将使用以下中的工作流程定义文件和源文件启动工作流程运行：
>
> - 你要推送到的分支（用于推送触发器）。有关更多信息，请参阅 示例：一个简单的代码推送触发器。
> - 您要从中提取的分支（用于拉取请求触发器）。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（用于计划触发器）。源存储库中的每个分支将启动一个工作流程运行。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，请参阅分支时的触发注意事项。

有关更多示例，请参阅触发器示例。

相应的用户界面：可视化编辑器/工作流程图/触发器/分支

FilesChanged

(Triggers/FilesChanged)

（如果触发器设置Type为PUSH、或，则为可选PULLREQUEST。 如果将触发器设置Type为，则不支持SCHEDULE。 ）

指定触发器监控的源存储库中的文件或文件夹，以便知道何时开始工作流程运行。您可以使用正则表达式来匹配文件名或路径。

有关示例，请参阅触发器示例。

相应的用户界面：可视化编辑器/工作流程图/触发器/文件已更改

Expression

(Triggers/Expression)

（如果触发器设置为，Type则为必填项SCHEDULE）

指定 cron 表达式，用于描述您希望计划的工作流程何时运行。

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：

####days-of-month##days-of-week##

cron 表达式的示例

分钟	小时	一个月中的天数	月	一周中的几天	年	含义
0	0	?	*	MON-FRI	*	每周一至周五的午夜 (UTC+0) 运行工作流程。
0	2	*	*	?	*	每天凌晨 2:00（UTC +0）运行工作流程。

分钟	小时	一个月中的天数	月	一周中的几天	年	含义
15	22	*	*	?	*	每天晚上 10:15（UTC +0）运行工作流程。
0/30	22-2	?	*	SAT-SUN	*	周六至周日每隔 30 分钟运行一次工作流程，时间为起始日晚上 10:00 至次日凌晨 2:00（UTC +0）。
45	13	L	*	?	2023-2027	在 2023 年至 2027 年（含）之间的当月最后一天下午 1:45（UTC +0）运行工作流程。

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) 指定时间。不支持其他时区。
- 配置两次运行之间的间隔至少为 30 分钟。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，请参阅触发器示例。

相应的用户界面：可视化编辑器/工作流程图/触发器/计划

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`标识生成操作。
- 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。该操作类型主题包含指向每个操作的文档的链接。

以下是工作流程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）

将####替换为要赋予操作的名称。操作名称在工作流程中必须是唯一的，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 语法指南。

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/ **####** /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。将操作分组到操作组可以帮助您保持工作流程井井有条，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程中必须是唯一的，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 语法指南。

有关操作组的更多信息，请参阅将操作分组为行动组。

对应的用户界面：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。每件作品都保存在名为 “问题” 的不同记录中。您可以通过向问题添加任务清单将问题分解为较小的目标。每个问题都可以有描述、受托人、状态和其他属性，您可以对其进行搜索、分组和筛选。您可以使用默认视图查看问题，也可以使用自定义筛选、排序或分组创建自己的视图。有关与议题相关的概念的更多信息，请参阅问题概念。要了解如何创建您的第一期问题，请参阅在中创建问题 CodeCatalyst。

以下是使用议题的团队可能采用的工作流程：

豪尔赫·索萨是一名在项目中工作的开发人员。他和他的项目成员李娟、马特奥·杰克逊和王秀兰合作确定需要做哪些工作。每天，他和他的开发者们都会在王秀兰的带领下举行同步会议。他们通过导航到团队对棋盘的看法来拉动棋盘。通过创建视图，用户和团队可以保存问题的筛选器、分组和排序，以便轻松查看符合其指定标准的问题。他们的视图包含按受托人分组并按优先级排序的问题，以显示每个开发者最重要的问题和问题状态。当 Jorge 被分配要完成的任务时，他通过为每项任务创建一个问题来计划自己的工作。在创建问题时，Jorge 可以选择相应的状态、优先级和工作估算工作。对于较大的问题，Jorge 会在问题中添加任务，将工作分解为较小的目标。豪尔赫用草稿状态创建自己的问题，例如待办事项，因为他不打算立即开始处理这些问题。处于草稿状态的问题将显示在 “草稿” 视图中，在那里需要对它们进行计划和确定优先顺序。Jorge 准备好开始工作后，他会将相应议题的状态更新为另一个类别（“未开始”、“已开始” 或 “已完成”），从而将其移至董事会。在处理每项任务时，团队可以按标题、状态、工作负责人、标签、优先级和估计值进行筛选，以找到与指定参数匹配的特定问题或类似问题。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。随着项目的进展，已完成的问题会累积为 “已完成” 状态。王秀兰决定使用快速存档按钮将其归档，从而将其从视图中删除，这样开发人员就可以专注于与当前和即将开展的工作有关的问题。

在计划工作时，参与项目的开发人员会选择 “排序依据” 和 “分组依据”，以查找他们想要从待办事项列表移至看板的问题。他们可能会选择根据最高优先级的客户请求向董事会添加问题，因此他们按客户请求标签对看板进行分组，并按优先级排序。他们还可以按估计值进行排序，以确保他们承担了可以完成的工作量。项目经理 Saanvi Sarkar 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问题对项目成功的重要性。

主题

- 问题概念
- 跟踪有问题的工作
- 使用待办事项、标签和留言板整理工作

- 中的问题配额 CodeCatalyst

问题概念

创建议题是一种快速有效的方法，可以跟踪项目中正在完成的工作。您可以使用议题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。

活跃的问题

活跃议题是指未处于 “草稿” 状态或未存档的任何议题。换句话说，活跃问题是指处于以下任何状态类别的问题：“未开始”、“已启动” 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 “活动议题” 视图中查看项目中的所有活跃议题。

已存档的问题

已存档的问题是指与您的项目不再相关的问题。例如，如果议题已完成且不再需要在 “完成” 列中查看，或者该议题是在错误的情况下创建的，则可以将其存档。如果需要，可以取消存档已存档的问题。

受让人

受让人是指问题被分配给的人。如果您搜索该人时未出现在列表中，则表示他们尚未添加到您的项目中。要添加它们，请参阅邀请用户加入项目。要允许多个受托人处理一个问题，请参阅。启用或禁用多个受托人多个受托人的问题将出现在你的看板上，头像颜色各不相同，每个头像代表一个受托人。

自定义字段

自定义字段允许您根据跟踪和维护项目内问题的需要自定义问题的不同属性。例如，您可以添加用于路线图的字段、特定的截止日期或请求者字段。

估计

在敏捷开发中，估计值被称为故事点。除了问题的模糊性和复杂性之外，您还可以使用问题的估计值来表示所需的工作量。对于风险较大、难度更大、未知数较大的问题，可以考虑使用更高的估计值。

有关估计类型及其配置方法的更多信息，请参阅配置问题工作量估算。

问题

问题是跟踪与您的项目相关工作的记录。您可以为功能、任务、错误或与项目相关的任何其他工作创建议题。如果你使用的是敏捷开发，问题也可以描述一个长篇故事或用户故事。

标签

该标签用于对问题进行分组、排序和筛选。您可以输入新的标签名称或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。一个议题可以有多个标签，一个标签可以从议题中删除。要自定义标签，请参阅使用标签对作品进行分类。

优先级

优先级是指问题的重要程度。有四个选项：低、中、高和无优先级。

状态和状态类别

状态是问题的当前状态，用于快速检查问题从开始到完成的整个生命周期的进度。所有问题都必须有一个状态，并且每个状态都属于一个状态类别。状态类别用于帮助您整理状态和填充默认议题视图。

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，但不能创建其他状态类别。以下列表在括号中包含默认状态及其状态类别：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）。

有关使用状态的更多信息，请参阅使用自定义状态跟踪工作。

任务

可以向议题中添加@@ 任务，以进一步分解和组织该问题的工作。您可以在创建议题时向议题添加任务，也可以在现有议题中添加任务。查看议题时，您可以对其任务进行重新排序、移除或将其标记为已完成。

视图

CodeCatalyst 项目中的问题显示在视图中。视图可以是以列表格式显示议题的网格视图，也可以是将议题显示为按议题状态整理的列中的图块的看板视图。有四个默认视图，您可以通过自定义分组、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，显示当前未处理的问题。任何以 “草稿” 状态类别创建的议题都会显示在此视图中。团队可以使用此视图来查看哪些问题仍在定义中，或者哪些问题正在等待分配和处理。

- 活动问题视图是当前正在处理的所有问题的看板视图。任何处于 “未开始”、“已开始” 或 “已完成” 状态类别的问题都将显示在此视图中。
- “所有问题” 视图是一个网格视图，显示项目中的所有问题，包括草稿和活动议题。
- 已存档视图显示所有已存档的问题。

跟踪有问题的工作

您可以使用议题来计划和跟踪您在项目上的工作。每期都是一部作品，保存在不同的记录中。可以将问题分解为任务，以进一步组织和跟踪该问题的工作。您还可以在议题之间创建链接以帮助您跟踪相关工作，添加标签以帮助您对工作进行组织和分类，对问题进行分组，为工作分配优先级，并指明工作是否被阻止。

当你准备好处理一个或一组问题时，你可以估算工作量，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。您还可以导出议题，以帮助将其包含的信息转换为其他格式。

在中创建问题 CodeCatalyst

开发团队创建问题以帮助跟踪和管理他们的工作。您可以根据需要在项目中创建问题。例如，您可以创建一个议题来跟踪代码中变量的更新。您可以将议题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。

创建议题

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到要在其中创建议题的项目。
3. 在项目主页上，选择创建问题。或者，在导航窗格中选择 “问题”。
4. 选择 “创建问题”。

> **Note**
>
> 使用网格视图时，您也可以内联添加问题。

5. 输入问题的标题。
6. （可选）输入描述。你可以使用 Markdown 来添加格式。
7. （可选）选择问题的状态、优先级和估计。

> **Note**
>
> 如果项目的估算设置设置为 “隐藏估计”，则不会有 “估计” 字段。

8. （可选）向议题添加任务。任务可用于将议题的工作分解为较小的目标。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其标记为已完成，也可以通过选择并从复选框左侧拖动任务来重新排序任务。
9. （可选）添加现有标签或创建新标签并通过选择 + 添加标签进行添加。
   a. 要添加现有标签，请从列表中选择标签。您可以在字段中输入搜索词，以搜索项目中包含该术语的所有标签。
   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 + 添加工作负责人来添加工作负责人。您可以通过选择 + 添加我来快速将自己添加为受托人。

> **Tip**
>
> 您可以选择将问题分配给 Amazon Q，让 Amazon Q 尝试解决问题。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）区域可用。
> 此功能要求为该空间启用生成式 AI 功能。有关更多信息，请参阅管理生成式 AI 功能。

11. （可选）添加现有的自定义字段或创建新的自定义字段。议题可能有多个自定义字段。
    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搜索词，以搜索项目中包含该术语的所有自定义字段。
    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 “创建问题”。右下角会显示一条通知：如果问题已成功创建，则会显示一条确认消息，说明问题已成功创建。如果问题未成功创建，则会显示一条包含失败原因的错误消息。然后，您可以选择 “重试” 进行编辑并重试创建问题，或者选择 “丢弃” 放弃问题。这两个选项都将关闭通知。

> **Note**
>
> 在创建议题时，您无法将拉取请求与议题相关联。但是，您可以在创建拉取请求后对其进行编辑，以添加拉取请求的链接。

创建和处理分配给 Amazon Q 的问题时的最佳实践

当你创建问题时，有时候其中一些问题会持续下去。造成这种情况的原因可能既复杂又多变。有时候是因为目前尚不清楚应该由谁来处理这个问题。其他时候，问题需要对代码库的特定部分进行研究或拥有专业知识，而该工作的最佳候选人则忙于处理其他问题。通常还有其他紧急工作必须先处理。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。如果您将问题分配给 Amazon Q，它将尝试创建解决方案草案供您评估。这可以帮助您和您的团队将工作重点放在需要您注意的问题上，并对其进行优化，而 Amazon Q 则为您没有资源可以立即解决的问题提供解决方案。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于用于软件开发的 Amazon Q 开发者代理的 “将问题分配给 Amazon Q” 功能是在 Amazon Bedrock 上构建的，因此用户可以充分利用 Amazon Bedrock 中实施的控制措施来加强安全、保障和负责任地使用人工智能 (AI)。

Amazon Q 在简单问题和直截了当的问题上表现最好。为了获得最佳结果，请使用通俗易懂的语言清楚地解释你想做什么。以下是一些最佳实践，可帮助您创建针对 Amazon Q 进行优化的问题以供处理。

> **Important**
>
> 生成式 AI 功能仅在美国西部（俄勒冈）地区可用。

- 简单一点。Amazon Q 最擅长简单的代码更改和修复，可以在问题的标题和描述中进行解释。不要用模糊的标题或过于华丽或矛盾的描述来分配问题。
- 具体一点。您提供的有关解决问题所需的确切更改的信息越多，Amazon Q 就越有可能创建解决该问题的解决方案。如果可能，请包括具体细节，例如要更改的 API 名称、要更新的方法、需要更改的测试以及您可以想到的任何其他细节。
- 在将问题分配给 Amazon Q 之前，请确保问题标题和描述中包含所有细节。 在将问题分配给 Amazon Q 之后，您就无法更改其标题或描述，因此，在将问题分配给 Amazon Q 之前，请确保您拥有问题中所需的所有信息。
- 仅在单一来源存储库中分配需要更改代码的议题。Amazon Q 只能在中处理单源存储库中的代码 CodeCatalyst。不支持链接存储库。在将问题分配给 Amazon Q 之前，请确保该问题只需要在单一来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，Amazon Q 的每一个步骤都需要您的批准。这使您不仅可以在问题评论中与 Amazon Q 互动，还可以就其创建的任何拉取请求进行互动。这为Amazon Q提供了更具互动性的体验，可帮助您调整其方法并完善其为解决问题而创建的代码。

  > **Note**
  >
  > Amazon Q 不会回复问题或拉取请求中的个人评论，但当被要求重新考虑其方法或创建修订版时，它会对其进行审查。

- 请务必仔细查看 Amazon Q 建议的方法。 一旦您批准其方法，Amazon Q 将开始基于该方法生成代码。在告诉 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细节。
- 如果您没有可在审核之前部署工作流程的现有工作流程，请确保仅允许 Amazon Q 处理工作流程。您的项目可能已将工作流程配置为在拉取请求事件上开始运行。如果是这样，Amazon Q 创建的任何拉取请求（包括创建或更新工作流程 YAML）都可能启动拉取请求中包含的那些工作流程。作为最佳实践，不要选择允许 Amazon Q 处理工作流程文件，除非您确定您的项目中没有可以自动运行这些工作流程的工作流程，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管理生成式 AI 功能。

估算问题

在敏捷开发中，估计值被称为故事点。除了问题的模糊性和复杂性之外，您还可以使用问题的估计值来表示所需的工作量。对于风险较大、难度更大、未知数较大的问题，可以考虑使用更高的估计值。

在开始估算问题之前，必须先选择要用于项目的估算类型。默认情况下，有两种类型可供选择。要有效地使用 T 恤尺码或斐波那契排序，您的团队必须根据每种尺寸所代表的内容进行调整。共同决定每个估算值对你意味着什么，然后开始将这些估算值应用于每个问题。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

为问题配置工作量估算

1. 在导航窗格中，选择 “问题”。
2. 选择活动议题以打开问题视图切换器下拉菜单，然后选择设置。
3. 在 “基本设置” 部分的 “估计” 中，选择估计值的显示方式。可用的估算类型有 T 恤尺码、斐波那契测序或隐藏估计值。如果项目的估算设置为 “隐藏估计”，则项目的问题中将没有估算字段。

   更新估算类型后，不会丢失任何数据，并且所有问题的估计值将自动转换。转换映射如下表所示。

T 恤尺码	斐波那契数列
XS	1
XS	2
S	3
M	5
L	8
XL	13

要添加或更改议题的估算值，您可以编辑该问题。

编辑和协作处理中的问题 CodeCatalyst

目录

编辑议题

按照以下步骤编辑问题的标题、描述、状态、受让人、优先级、估算值或标签。

编辑议题

1. 选择要编辑的问题以查看问题详情。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 要编辑问题标题，请选择标题，输入新标题，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。你可以使用 Markdown 来添加格式。
4. 在 “任务” 中，您可以查看和管理议题的任务。有关更多信息，请参阅 管理议题任务。
5. 要编辑状态、估计值或优先级，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，然后从列表中选择标签。您可以在字段中输入搜索词，以搜索项目中包含该术语的所有标签。

   b. 要创建并添加新标签，请选择 + 添加标签，在搜索栏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。如果您从所有问题中移除标签，则该标签将显示在问题设置标签部分的未使用标签部分中。使用筛选条件或向议题添加标签时，未使用的标签会出现在标签列表的末尾。您可以在问题设置中找到所有标签（已使用和未使用）的概述以及包含这些标签的问题。

7. 要分配问题，请在 “工作负责人” 部分中选择 “+ 添加受让人”，然后从列表中搜索并选择受让人。您可以选择 + 添加我，快速将自己添加为受托人。
8. 在 “附件” 中，您可以添加、下载或删除附件。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其 URL 或 ID。要取消拉取请求的链接，请选择取消链接图标。

   > **Tip**
   >
   > 在向议题添加拉取请求的链接后，您可以通过在链接的拉取请求列表中选择其 ID 来快速导航到该议题。您可以使用拉取请求的 URL 来链接与议题板不同的项目中的拉取请求，但只有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。议题可能有多个自定义字段。

    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搜索词，以搜索项目中包含该术语的所有自定义字段。

    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。如果您从所有问题中删除自定义字段，则该自定义字段将被删除，并且在筛选时您将无法再看到该字段。

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。使用以下步骤管理议题的附件。

添加到议题中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。任何用户都可以在可能包含恶意代码或内容的问题中添加附件。确保用户了解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. 选择要为其管理附件的问题。有关查找问题的帮助，请参阅查找和查看问题。
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导航到该文件并将其选中。选择 “打开” 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。

   请注意对附件文件名和内容类型的以下限制：

   - 文件名中不允许使用以下字符：
     - 控制字符：`0x00-0x1f`和 `0x80-0x9f`
     - 保留字符：`/?`、`<`、`>`、`\`、`:`、`*`、`|`、和 `"`
     - Unix 保留的文件名：`.`和 `..`
     - 尾随句点和空格
     - Windows 保留的文件名：`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，请选择要复制其网址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

您可以在议题设置中查看一个表格，其中包含项目中议题中添加的每个附件。此表包含每个附件的详细信息，包括内容类型、添加时间、添加到的问题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存档问题的大型附件，将其删除以腾出存储空间。

> **Important**
>
> Amazon 不会扫描或分析问题的附件 CodeCatalyst。任何用户都可以在可能包含恶意代码或内容的问题中添加附件。确保用户了解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议题附件

1. 在导航窗格中，选择 “问题”。
2. 选择省略号图标并选择设置。
3. 选择 “附件” 选项卡。

管理议题任务

可以向议题中添加@@ 任务，以进一步分解、组织和跟踪该议题的工作情况。

管理议题任务

1. 选择要管理任务的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 在 “任务” 中，您可以查看和管理议题的任务。
    1. 要添加任务，请在文本字段中输入任务名称，然后按 Enter。
    2. 要将任务标记为已完成，请选中该任务的复选框。
    3. 要查看或更新任务的详细信息，请从列表中选择该任务。
    4. 要对任务重新排序，请选择该任务并将其从复选框的左侧拖动。
    5. 要移除任务，请选择该任务的省略号菜单，然后选择移除。

将问题标记为已屏蔽或已解除屏蔽

如果某件事使您无法处理某个问题，则可能需要将其标记为已屏蔽。例如，如果您的问题依赖于对代码库中尚未合并的另一部分的更改，则该问题可能会被阻止。

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。

外部情况得到解决后，您可以解除对问题的封锁。

将问题标记为已屏蔽

1. 打开您要标记为已屏蔽的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 选择 “操作”，然后选择 “标记为已屏蔽”。

解除对问题的封锁

1. 打开您要解除封锁的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 选择 “操作”，然后选择 “标记为已解除封锁”。

添加、编辑或删除评论

你可以对问题发表评论。在评论中，您可以标记其他空间成员、空间中的其他项目、相关问题和代码。

为议题添加评论

1. 导航到您的项目。
2. 在导航栏中选择 “问题”。
3. 选择要在其中添加评论的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
4. 在 “评论” 字段中输入评论。你可以使用 Markdown 来添加格式。
5. 选择发送。

编辑注释

您可以编辑对问题发表的评论。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 “问题”。
3. 选择要在其中编辑评论的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
4. 要编辑评论，请找到您要编辑的评论。

> **Tip**
>
> 您可以先按最旧或最新的评论进行排序。评论一次加载 10 个。

5. 选择省略号图标，然后选择编辑。
6. 编辑评论。你可以使用 Markdown 来添加格式。
7. 选择保存。评论现已更新。

删除注释

您可以删除您对问题发表的评论。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，但带有字样 “此评论已被删除”，取代了原始评论文本。

1. 导航到您的项目。
2. 在导航栏中选择 “问题”。
3. 选择要删除评论的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
4. 选择省略号图标，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、相关问题和代码。这样做会创建一个指向你提及的用户或资源的快速链接。

在评论中给 @mention

1. 导航到您的项目。
2. 在导航栏中选择 “问题”。
3. 选择要编辑的问题以查看问题详情。要获取有关查找问题的帮助，请参阅查找和查看问题。
4. 选择 “添加评论” 文本框。
5. 键入@*user_name*以提及其他用户。
6. 键@*project_name*入提及项目。

7. 键入@*issue_name*或@*issue_number*提及其他问题。
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@mention将填充一份包含与您匹配的术语的前 5 个项目（用户、源存储库、项目等）的列表。

9. 选择您想提及的所需项目。显示项目所在位置的路径将填充到评论文本框中。
10. 完成评论并选择 “发送”。

查找和查看问题

以下各节介绍如何有效地搜索和查看 CodeCatalyst 项目中的问题。

正在搜索问题

您可以通过搜索特定参数来发现问题。有关优化搜索的更多信息，请参阅在中搜索代码、问题、项目和用户 CodeCatalyst。

搜索问题

1. 导航到您的项目。
2. 使用搜索栏搜索问题或与问题相关的信息。您可以使用查询参数来缩小搜索范围。有关更多信息，请参阅 在中搜索代码、问题、项目和用户 CodeCatalyst。

排序问题

默认情况下，中的问题 CodeCatalyst 按手动顺序排序。手动下单按用户移至的顺序显示问题。按手动顺序排序时，您可以拖放议题以更改其顺序。此排序选项在整理待办事项和确定问题的优先顺序时非常有用。

下表显示了如何在网格视图和看板视图中对问题进行排序。

网格视图排序选项	看板视图排序选项
手动订购	手动订购

网格视图排序选项	看板视图排序选项
上次更新	上次更新
优先级	优先级
估计	估计
Title	Title
ID	
Status	
阻止	
自定义字段	

使用以下步骤更改问题的排序方式。

对问题进行排序

1. 导航到您的项目。
2. 在导航窗格中，选择 “问题”。默认视图为看板。
3. （可选）选择 “活动问题” 以打开问题视图切换器下拉菜单以导航到不同的问题视图。
4. 要对网格视图进行排序，有两个选项：

    a. 选择要作为排序依据的字段的标题。选择标题将在升序和降序之间循环。

    b. 选择 “排序依据” 下拉菜单并选择要作为排序依据的参数。问题将按升序排序。
5. 要对看板视图进行排序，请选择 “排序依据” 下拉菜单并选择要作为排序依据的参数。问题将按升序排序。

分组问题

分组用于按多个参数（例如工作负责人、标签和优先级）来组织看板上的问题。

对问题进行分组

1. 导航到您的项目。

2. 在导航窗格中，选择 “问题”。默认视图为看板。
3. （可选）选择 “活动问题” 以打开问题视图切换器下拉菜单以导航到不同的问题视图。
4. 选择 “群组”。
5. 在分组依据中，选择分组依据的参数：
   - 如果您选择 “受让人” 或 “优先级”，请选择 “群组” 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 “显示空群组” 切换开关以显示或隐藏当前未分配任何议题的群组。
7. 视图会在您做出选择时更新。只有与配置的参数匹配的组中才会出现问题。

筛选问题

使用筛选功能查找包含指定名称、优先级、标签、自定义字段或受托人的议题。

筛选问题

1. 导航到您的项目。
2. 在导航窗格中，选择 “问题”。
3. （可选）选择 “活动问题” 以打开问题视图切换器下拉菜单以导航到不同的问题视图。

> **Note**
>
> 要根据问题名称或描述中的字符串进行筛选，请在问题搜索栏中输入该字符串。

4. 选择 “筛选”，然后选择 “+ 添加过滤器”。
5. 选择要筛选的参数。您可以选择多个过滤器和参数。您可以通过选择和或或来配置筛选器以显示与每个筛选器或任何单个筛选器匹配的问题。视图将更新以显示与筛选条件匹配的问题。

正在处理一个问题

每个问题都有生命周期。在中 CodeCatalyst，问题通常以待办事项列表中的草稿形式开始。当要开始处理该问题时，它会被移到另一个状态类别中，并在各种状态之间移动，直到完成，然后将其存档。您可以通过以下方式在议题的整个生命周期中移动或取得进展：

- 你可以在待办事项列表和看板之间移动问题。
- 您可以将进行中的问题移至不同的完成阶段。

- 您可以将已完成的议题存档。

在待办事项列表和看板之间移动问题

开始处理问题后，您可以将问题从待办事项列表移至看板。如果工作被推迟，你也可以将问题移回待办事项中。

在待办事项列表和看板之间移动问题

1. 导航到您的项目。
2. 在导航窗格中，选择 “问题”。默认视图为看板。
3. 要将问题从看板移至待办事项列表，请执行以下操作：
   a. 选择要移动的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
   b. 从 “状态” 下拉菜单中选择 “待办事项”。
4. 要将问题从待办事项列表移至看板，请执行以下操作：
   a. 要导航到待办事项列表，请选择 Board，然后选择待办事项列表。
   b. 选择要移动的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
   c. 选择 “添加到图板”，或选择 “待办事项列表” 以外的状态。

在棋盘上通过生命周期阶段推进问题

您可以将看板内的议题移至不同的状态，直到完成。

在董事会内移动议题

1. 在导航窗格中，选择 “问题”。默认视图为看板。
2. 请执行以下操作之一：
   - 将议题拖放到其他状态。
   - 选择一个问题，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：*####*”。

有关存档议题的信息，请参阅存档问题。

在群组之间移动议题

您可以按各种参数对 “所有问题” 和 “看板” 视图中的议题进行分组。如果问题已分组，则可以将议题从一个组移到另一个组。将议题从一个群组移到另一个群组将自动编辑议题分组的字段，使其与目标群组相匹配。

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。董事会按分组`Assignee`，分为两组，每个受托人各占一组。将问题从王秀兰小组转移到Saanvi Sarkar小组将使该问题的最新受托人分配给Saanvi Sarkar。

存档问题

> **Note**
>
> 议题不是在项目中删除，而是存档。要删除议题，必须删除该项目。

当项目中不再需要议题时，您可以将其存档。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。可以在已存档问题默认视图中查看已存档的问题，如果需要，可以在该视图中取消存档。

如果出现以下情况，则可以存档问题：

- 您已在 “完成” 列中完成问题，不再需要该问题。
- 你没有计划去做这件事。
- 你错误地创建了它。
- 您已达到最大活跃问题数量。

存档议题

1. 打开您要存档的问题。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 选择 “操作”，然后选择 “移至存档”。
3. （可选）要快速存档多个处于 “已完成” 状态的问题，请选择看板上任何 “已完成” 状态顶部的垂直省略号，然后选择 “存档问题”。

取消存档议题

1. 打开您要取消存档的问题。您可以通过从问题视图切换器下拉菜单中打开已存档问题视图来查看已存档问题列表。要获取有关查找问题的帮助，请参阅查找和查看问题。
2. 选择 “取消存档”。

导出问题

您可以将当前视图中的议题导出到.xlsx 文件中。要导出问题，请执行以下步骤。

导出问题

1. 导航到您的项目。
2. 在导航栏中选择 “问题”。
3. 选择活动议题以打开议题视图切换器下拉菜单，然后导航到包含要导出的议题的视图。只有视图中显示的问题才会被导出。
4. 选择省略号菜单，然后选择导出到 Excel。
5. 下载.xlsx 文件。默认情况下，它的标题是项目的名称和导出完成的日期。

使用待办事项、标签和留言板整理工作

并非所有团队的工作方式都是一样的。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。您可以选择哪种估算方法来处理问题，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作品视图。根据团队的工作方式，您可以配置是允许多个受托人处理一个问题，还是只允许将一个问题分配给一个用户。您还可以创建问题的自定义视图，以便以向您或您的团队显示最相关的信息的方式显示工作。

使用标签对作品进行分类

您可以为问题自定义标签。这包括编辑标签和更改颜色。标签可以帮助您对作品进行分类和整理。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
- 编辑标签

- 删除标签

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

使用以下步骤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 “问题”。
2. 选择活动议题以打开问题视图切换器下拉菜单，然后选择设置。
3. 标签图块上有项目中使用的标签列表。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，请选择复选标记图标。
5. 更改后的标签现在显示在您的可用标签列表中。您还可以查看有多少问题正在使用该标签。

> **Note**
>
> 您可以选择每个标签旁边显示的数字，以导航到 “所有问题” 页面并查看包含该标签的所有问题。

删除标签

您目前无法在中删除问题标签 CodeCatalyst。如果您从所有问题中移除标签，则该标签将显示在问题设置标签部分的未使用标签部分中。使用筛选条件或向议题添加标签时，未使用的标签会出现在标签列表的末尾。您可以在问题设置中找到所有标签（已使用和未使用）的概述以及包含这些标签的问题。

使用自定义字段组织工作

您可以创建自定义字段来帮助组织和查看项目的作品。自定义字段已添加到 Filt er 中的可用筛选器列表中，因此您可以按自定义字段筛选问题。自定义字段是名称和值对。您可以按自定义字段的名称进行筛选，然后按该自定义字段的值进行筛选。

一个议题可能有多个自定义字段。

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

要删除自定义字段，您必须将其从添加到的每个问题中移除该自定义字段。删除自定义字段后，您将不再在 “筛选器” 中看到该自定义字段。您可以使用筛选器查看自定义字段的所有问题，并通过编辑议题将其删除。有关更多信息，请参阅查找和查看问题和编辑议题

使用自定义状态跟踪工作

你可以在你的看板上添加自定义状态。每个自定义状态都必须属于以下类别之一：草稿、未开始、已开始或已完成。状态类别用于帮助组织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，请参阅查找和查看问题。

创建状态

1. 在导航窗格中，选择 “问题”。
2. 选择活动议题以打开问题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，自定义状态在您的看板上可见，并在创建问题时显示为一个选项。

编辑状态

1. 在导航窗格中，选择 “问题”。
2. 选择活动议题以打开问题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，然后选择复选标记图标。

   编辑后的状态现在可以在您的图板上看到。

移动状态

1. 在导航窗格中，选择 “问题”。
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. 将状态拖放到你想要的位置。

   > **Note**
   >
   > 您只能在其指定类别内移动状态。

   现在，您的看板上的状态已重新排序。

停用状态

1. 在导航窗格中，选择 “问题”。
2. 选择活动议题以打开问题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，选择状态的开关。状态现已显示为灰色。

   > **Note**
   >
   > 停用状态会显示在看板上，直到所有问题都移出去。不能将问题添加到已停用状态。

5. 要重新激活已停用的状态，请选择该状态的开关。状态不再显示为灰色。

> **Note**
>
> 每个类别中必须至少有一个活跃状态。如果该类别中只有一种状态，则无法将其停用。

配置问题工作量估算

按照以下步骤配置中问题的工作量估算设置。 CodeCatalyst

为问题配置工作量估算

1. 在导航窗格中，选择 “问题”。
2. 选择活动议题以打开问题视图切换器下拉菜单，然后选择设置。
3. 在 “基本设置” 部分的 “估计” 中，选择估计值的显示方式。可用的估算类型有 T 恤尺码、斐波那契测序或隐藏估计值。更新估算类型后，不会丢失任何数据，并且所有问题的估计值将自动转换。转换映射如下表所示。

T 恤尺码	斐波那契数列
XS	1
XS	2
S	3
M	5
L	8
XL	13

启用或禁用多个受托人

按照以下步骤为多个受托人配置问题设置。 CodeCatalyst

启用或禁用多个受托人

1. 在导航窗格中，选择 “问题”。

2. 选择活动议题以打开问题视图切换器下拉菜单，然后选择设置。
3. 在 “基本设置” 部分的 A ssignee 中，切换指示器以允许将多个受托人分配给同一个问题。一个议题最多可以有 10 名受托人。如果您不启用此选项，则只能为一个问题分配一个受让人。

创建问题视图

您可以创建视图来快速查看与一组特定筛选条件匹配的问题。这可以帮助您节省时间并快速查看之前筛选、分组或排序依据的问题。

创建问题视图

1. 在导航窗格中，选择 “问题”。
2. （可选）根据您的用例，您可能需要从现有视图创建视图。要导航到其他视图，请选择活动议题以打开问题视图切换器下拉菜单并选择视图。
3. （可选）在创建视图之前，请配置筛选器、分组和排序。您可以在创建视图时添加这些内容，但如果您之前这样做，则可以在创建视图之前预览视图中显示的内容。
4. 从标题栏中打开问题视图切换器下拉菜单。要创建看板视图，其中根据状态分列查看问题，请在 “看板” 列中选择 +。要创建在列表中查看问题的网格视图，请在 “网格” 列中选择 +。如果您改变主意，可以在创建视图之前更改其类型。
5. 在 “创建视图” 对话框中，输入视图的名称。
6. “筛选器”、“问题分组依据” 和 “问题排序依据” 字段将根据当前视图的设置填充。如有必要，请对其进行更新。
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，请参阅的配额 CodeCatalyst...

资源	默认限额
活跃的问题	每个项目最多 1,000 个。

资源	默认限额
附件大小	每个附件的最大容量为 500MB。 附件的最大总存储空间受空间总体存储限制的影响。有关更多信息，请参阅定价。
问题总数（活跃和已存档）	每个项目最多 10 万个。
已保存的视图	每个项目最多保存 50 个议题视图。
您可以关联到议题的拉取请求数量	每个问题最多 50 个拉取请求。
状态（每个项目）	每个项目最多 50 个。
状态（每期）	每期最多 50 个。
标签（每个项目）	每个项目最多 200 个。
标签（每期）	每期最多 50 个。
自定义字段（每期）	每期最多 50 个。
受托人	每期最多 10 个。
注释	每期最多 1,000 个。
任务	每期最多 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 中不存在生成器 ID 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：

- 作为创建空间的一部分。
- 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（例如项目管理员和参与者）是特定于项目的，因此您可以在一个项目中扮演一个角色，在另一个项目中扮演不同的角色。如果您创建空间，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。当你邀请用户加入项目时，你可以选择你希望他们在项目中扮演的角色，这决定了他们在项目中可以采取和不能采取的行动。大多数在项目上工作的用户只需要参与者角色即可执行任务。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 (IDE) 时还需要个人访问令牌 (PAT) 才能访问项目的源存储库。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，角色指定允许用户使用该项目的资源在项目中执行的操作。用户加入项目后即可获得空间的成员资格。您可以添加或删除空间管理员用户。空间管理员角色的权限是中所有角色中最广泛的。 CodeCatalyst最佳做法是，为用户分配执行工作所需的最小权限。

您可以为空间中的用户分配角色。您也可以为其所属项目中的用户分配角色。每个用户在一个项目或空间中只能有一个角色，但是用户在每个项目和空间中可以拥有不同的角色。例如，用户可能在一个项目中具有项目管理员角色，在另一个项目中具有参与者角色。

主题

了解空间和项目的用户角色

空间有三个角色可用：

- 空间管理员
- 高级用户
- 访问受限

接受项目邀请的用户将在包含该项目的空间中自动分配给他们 “受限” 访问角色。

项目中有四个角色可供成员使用：

- 项目管理员
- 贡献者
- 审稿人
- 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从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至至少一个其他用户，该用户可以在原始空间创建者不可用时扮演此角色。

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。它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 Power user 角色分配给作为团队负责人或经理的用户，他们需要能够在工作中创建项目和管理空间中的用户。

有限访问角色

“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。当用户接受空间中项目的邀请时，该角色会自动分配给他们。它提供了他们在包含该项目的空间内工作所需的有限权限。将受限访问权限角色分配给您直接邀请进入空间的用户，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。

项目角色在空间级别没有任何权限。因此，具有项目管理员角色的用户无法创建其他项目。只有拥有 Space 管理员或超级用户角色的用户才能创建项目。

> **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、工作流程、问题和操作的用户。

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 Reviewer 角色分配给需要能够批准和评论拉取请求、创建、更新、解决和评论问题，以及查看项目中的代码和工作流程的用户。

只读角色

“只读” 角色适用于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能直接与之交互的用户。

查看每个角色的可用权限

下表显示了每个 CodeCatalyst 角色的可用权限。使用链接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
- Custom blueprint permissions
- Notifications permissions
- Search permissions

权限	空间管理员角色	高级用户角色	受限访问角色	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
编辑空间账单详情							
设置并启用单点登录							
移除单点登录							
为空间启用生成式 AI 功能							
为空间禁用生成式 AI 功能							
删除空间							
将其他用户添加到 Space 管理员角色							
将其他用户从空间管理员角色中移除							
创建团队							
删除球队							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
更新小组							
禁用该空间的计算机资源							
为空间启用计算机资源							
创建项目							
将 AWS 账户关联与空间关联							
更新 AWS 账户连接							
解除 AWS 账户与空间的关联							
删除 AWS 账户连接并将其从空间中移除							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
邀请其他人进入空间							
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							
查看 AWS 账户连接							
查看以下各项的事件 CodeCatalyst							
查看空间							
查看球队							
查看 VPC 连接							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
扩展程序权限	空间管理员角色	高级用户角色	受限访问角色	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							
断开 GitHub 账号连接							
连接 Jira 网站							
断开 Jira 网站的连接							
查看已安装扩展的配置详细信息							
查看扩展							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
项目权限	空间管理员角色	高级用户角色	受限访问角色	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							
禁用项目的计算机资源							
为项目启用计算机资源							
删除项目							
邀请用户加入项目							
更改项目中用户的角色							
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							

权限	空间管理员角色	高级用户角色	受限访问角色	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							
链接存储库							
取消存储库的链接							
删除仓库							
编辑存储库设置							
查看存储库							
查看存储库设置							
克隆存储库							

权限	空间管理员角色	高级用户角色	受限访问角色	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							
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							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
删除 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							
为拉取请求创建批准规则							
覆盖拉取请求的合并要求							
更新拉取请求							
更新拉取请求的批准规则							
查看拉取请求							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看拉取请求的批准规则							
关闭拉取请求							
批准拉取请求							
评论拉取请求							
在对拉取请求的评论中与 Amazon Q 互动							
为 Amazon Q 创建的拉取请求创建修订版							
将议题链接到拉取请求							
取消议题与拉取请求的关联							
开发环境权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建自己的开发环境							
停止你自己的开发环境							
停止其他用户创建的开发环境							
恢复你自己的开发环境							
查看您自己的开发环境							
查看其他用户创建的开发环境							
编辑你自己的开发环境							
编辑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
删除你自己的开发环境							
删除其他用户创建的开发环境							
为开发环境创建开发文件							
编辑开发环境的开发文件							
删除开发环境的开发文件							
查看开发环境的开发文件							
Package 存储库和包权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建软件包存储库							
查看软件包存储库							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
编辑软件包存储库							
删除软件包存储库							
创建网关包存储库							
查看网关软件包存储库							
删除网关包存储库							
添加上游软件包存储库							
编辑上游存储库的搜索顺序							
移除上游软件包存储库							
Connect 连接到软件包存储库							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
从软件包存储库中读取软件包							
将软件包发布到软件包存储库							
读取和保留上游存储库中的软件包							
查看软件包							
查看软件包版本							
查看软件包版本资产							
列出软件包版本依赖关系							
更新程序包版本状态							
更新包源配置							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
删除软件包版本							

工作流程权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建工作流程							
更新工作流程							
删除工作流程							
启动工作流程							
停止工作流程							
创建工作流程密钥							
更新工作流程密钥							
删除工作流程密钥							
创建环境							
删除环境							
创建舰队							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
更新舰队							
删除舰队							
管理其他账户中的工作流程资源							
将 VPC 连接与环境关联							
解除 VPC 连接与环境的关联							
将 VPC 连接的环境与工作流程关联							
取消与 VPC 连接的环境与工作流的关联							
批准工作流程运行							
在工作流程中跟踪提交							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看环境							
查看构建操作日志							
查看舰队							
查看测试操作日志							
查看工作流程							
查看工作流程运行情况							
查看工作流程运行结果							
查看工作流程秘密							

发放权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
创建问题							
更新问题							
查看问题							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
存档问题							
将问题分配给 Amazon Q							
在问题评论中与 Amazon Q 互动							
从议题中取消分配 Amazon Q							
更新其他用户创建的问题							
查看对某个问题的评论							
对议题发表评论							
更新对议题的评论							
创建标签							
更新标签							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
查看标签							
为议题添加标签							
从议题中移除标签							
为问题创建自定义状态							
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							
为议题添加附件							
查看问题附件							
从议题中移除附件							
将拉取请求链接到议题							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
取消拉取请求与议题的关联							
链接 Jira 项目							
取消关联 Jira 项目							
自定义蓝图权限	空间管理员角色	高级用户角色	受限访问角色	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							
取消发布预览自定义蓝图							
发布自定义蓝图							
取消发布自定义蓝图							
将自定义蓝图添加到太空蓝图目录							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
从太空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							
将自定义蓝图应用于项目							
取消自定义蓝图与项目的关联							
更新已应用的自定义蓝图的版本							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
编辑自定义蓝图的别名							
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							
为关联的电子邮件账户配置电子邮件通知							

权限	空间管理员角色	高级用户角色	受限访问角色	项目管理员角色	贡献者角色	审阅者角色	只读角色
搜索权限	空间管理员角色	高级用户角色	受限访问角色	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							
在空间中搜寻							

查看和更改用户角色

您可以查看分配给用户的角色。这可以帮助你了解他们在项目中可以采取哪些行动。如果他们需要其他权限，您也可以更改他们的角色。

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权限更改为空间管理员，以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访问空间的用户。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存储库访问权限

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)。您创建的 PAT 与您在所有空间和项目中的 CodeCatalyst用户身份相关联。您可以为自己的 CodeCatalyst身份创建多个 PAT。

您可以查看已创建的 PAT 的名称和到期日期，也可以删除不再需要的 PAT。您只能在创建 PAT 密钥时对其进行复制。

> **Note**
>
> 默认情况下，PAT 在 1 年后过期。

创建 PAT

PAT 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 PAT（控制台）

您可以使用控制台在中创建 PAT。 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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而找到您的用户个人资料。

3. 在个人访问令牌下，选择创建。

   将显示 “创建 PAT” 页面。
4. 在 PAT 名称中，输入 PAT 的描述性名称。
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 密钥存储在可以检索的地方。

> **Important**
>
> PAT 密钥仅显示一次。关闭窗口后无法检索它。如果您未将 PAT 密钥保存在安全位置，则可以创建另一个密钥。

.

创建 PAT (CLI)

您可以使用 CLI 在中创建 PAT。 CodeCatalyst

创建个人访问令牌 (AWS CLI)

1. 在终端或命令行中，按如下方式运行create-access-token命令。

```
aws codecatalyst create-access-token
```

如果成功，该命令将返回有关已创建的 PAT 的信息，如下例所示。

```
{
    "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。如果您放错了 PAT 密钥或担心其存储不安全，则可以创建另一个密钥。

您可以使用查看与您的用户帐户关联的 PAT。 AWS CLI您只能查看有关 PAT 的信息，而不能查看 PAT 密钥本身的价值。

> **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。

查看 PAT

您可以在中查看 PAT。 CodeCatalyst该列表显示了您与用户身份关联的所有 PAT。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。过期的 PAT 不会显示，因为它们会在过期后被删除。

查看 PAT（控制台）

您可以使用控制台在中查看与您的用户身份关联的 PAT。 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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而找到您的用户个人资料。

3. 在“个人访问令牌”下，查看当前 PAT 的名称和到期日期。

查看 PAT (CLI)

您可以使用 CLI 在中查看与您的用户身份关联的 PAT。 CodeCatalyst

查看您的个人访问令牌 (AWS CLI)

- 在终端或命令行中，按如下方式运行list-access-tokens命令。

```
aws codecatalyst list-access-tokens
```

如果成功，该命令将返回与您的用户帐户关联的 PAT 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"
        }
    ]
}
```

删除 PAT

您可以在中删除与您的用户身份关联的 PAT。 CodeCatalyst

删除 PAT（控制台）

您可以使用控制台删除中的 PAT。 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**
>
> 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而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 Delete PAT 上：？<name> 页面，要确认删除，请在文本字段中键入 delete。选择删除。

删除 PAT (CLI)

您可以使用删除与您的用户身份关联的 PAT AWS CLI。为此，您必须提供 PAT 的 ID，您可以使用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 AWS CLI 搭配使用的更多信息 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)

- 在终端或命令行运行delete-access-token命令，提供要删除的 PAT 的 ID。例如，运行以下命令删除 ID 为 *123* EXAMPLE 的 PAT。

  ```
  aws codecatalyst delete-access-token --id a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb
  ```

  如果成功，此命令将不返回任何响应。

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**
>
> 您可以在有权访问的 GitHub 组织中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。

建立人际关系

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。

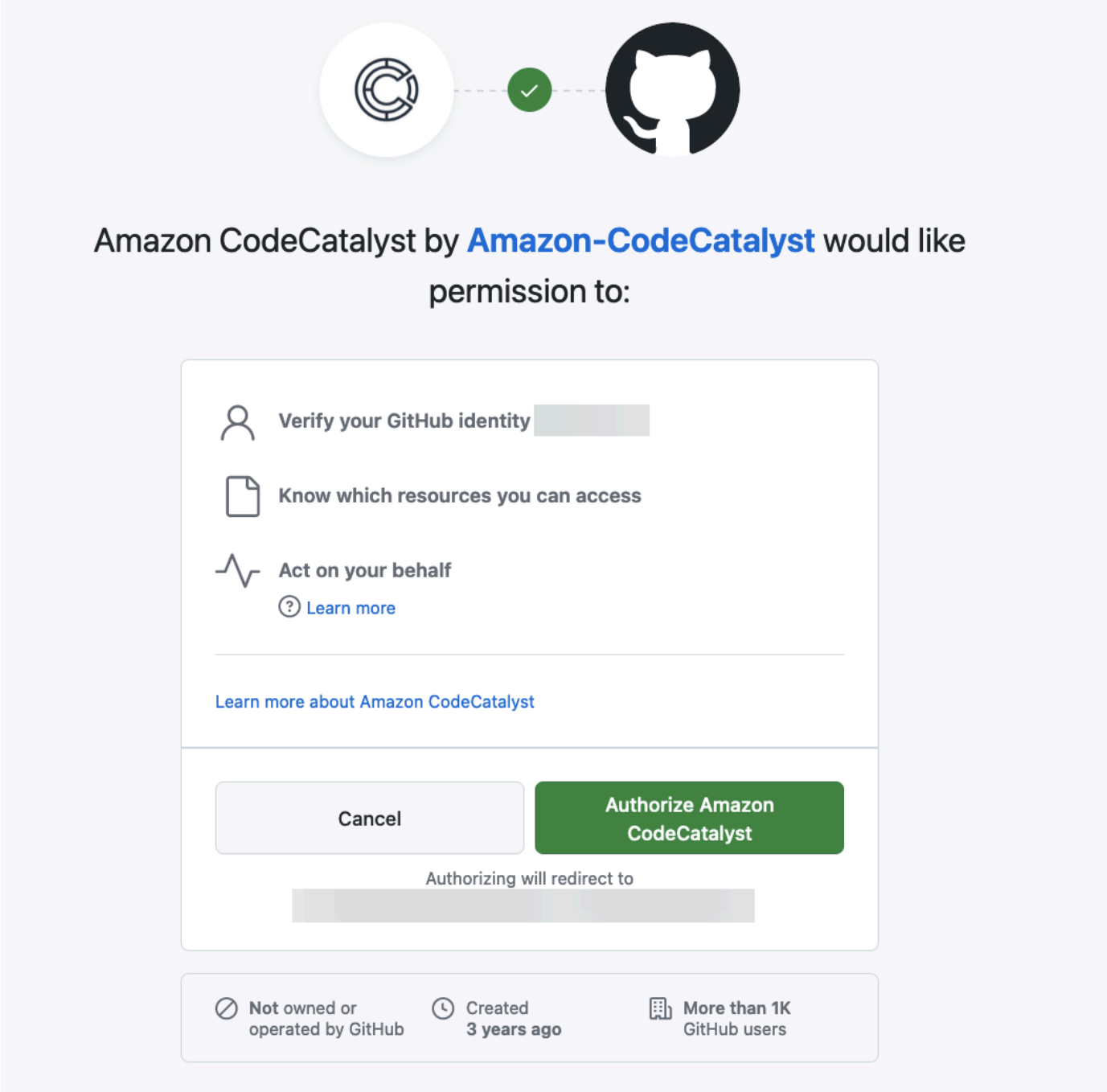
创建个人关系

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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而找到您的用户个人资料。

3. 在 “个人关系” 下，选择 “创建”。

   将显示 “创建连接” 页面。
4. 选择创建。将显示 “创建连接” 页面。
5. 在创建连接页面的提供者中，选择GitHub。在连接名称中，键入连接的名称。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您要安装连接器应用程序。



删除个人联系

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。如果您创建了新的个人连接，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**
   >
   > 您还可以前往项目或空间的成员页面，然后从成员列表中选择您的姓名，从而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” 连接上：？<name> 页面，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，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。要撤消安装，请选择 “撤消”。

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。如果您是空间成员并与其他人合作开展项目，我们特别建议您配置 MFA。由于不止一个人可以访问一个项目，因此存在更多安全漏洞的机会。

启用 MFA 后，您必须 CodeCatalyst 使用电子邮件和密码登录亚马逊。登录的这一部分是第一个因素，即你使用自己知道的东西。然后，您使用密码或安全密钥登录。这是第二个因素，也就是你所拥有的。第二个因素可能是由您的移动设备生成的身份验证码，或者通过点击或按下连接到计算机的安全密钥生成的身份验证码。总而言之，这多种因素通过防止未经授权的访问来提高安全性。

如何注册设备以使用多因素身份验证

使用我的个人资料 > 多重身份验证中的以下步骤注册您的新设备以进行多因素身份验证 (MFA)。

> **Note**
>
> 我们建议您先将相应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，选择带有第一个首字母缩写的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在个人资料页面上，选择管理个人资料和安全。将打开 AWS Builder ID 个人资料页面。
4. 在页面的左侧，选择 “安全”。
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**
        >
        > 这种体验因您的操作系统和浏览器而异，因此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，您可以选择将友好的显示名称与新注册的设备关联起来。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 应用程序支持多台设备，请选择创建新的 MFA 设备的选项。

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 Show QR code（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**
>
> 在为 AWS Builder ID 配置 MFA 设备时，请将 QR 码或密钥的副本保存在安全的地方。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，您可以重新同步装置。

4. 选择 Assign MFA (分配 MFA)。MFA 设备现在可以开始生成一次性密码了，现在可以开始使用了。

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。用户可以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，能够生成六位数的身份验证码。

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

尽管任何符合 TOTP 的应用程序都可使用 IAM Identity Center MFA，但下表列出了可供选择的知名第三方身份验证器应用程序。

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

更换您的 MFA 设备

注册 MFA 设备后，您可以更改其名称或将其删除。我们建议始终至少启用一台 MFA 设备，以获得额外的安全保护。您最多可以注册五台设备。要了解如何添加更多内容，请参阅如何注册设备以使用多因素身份验证。

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，选择带有第一个首字母缩写的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在个人资料页面上，选择管理个人资料和安全。将打开 AWS Builder ID 个人资料页面。
4. 选择页面左侧的多重身份验证。当你到达页面时，你会看到 “重命名” 显示为灰色。
5. 选择要更改的 MFA 设备。选择 Rename (重命名)。然后弹出一个模态。
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。重命名的设备显示在多重身份验证设备 (MFA) 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，选择带有第一个首字母缩写的图标旁边的箭头，然后选择用户个人资料。CodeCatalyst个人资料页面打开。
3. 在个人资料页面上，选择管理个人资料和安全。将打开 AWS Builder ID 个人资料页面。
4. 选择页面左侧的多重身份验证。当你到达该页面时，你会看到 “删除” 显示为灰色。
5. 选择要更改的 MFA 设备。选择删除。出现一个模态，上面写着 “删除 MFA 设备？” 。按照说明删除您的设备。
6. 选择删除。已删除的设备不再出现在多重身份验证设备 (MFA) 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 的数据保护 CodeCatalyst
- 身份与访问管理与亚马逊 CodeCatalyst
- Amazon 合规性验证 CodeCatalyst

- Amazon 的弹性 CodeCatalyst
- Amazon 的基础设施安全 CodeCatalyst
- Amazon 中的配置和漏洞分析 CodeCatalyst
- 您在亚马逊上的数据和隐私 CodeCatalyst
- Amazon 工作流程操作的最佳实践 CodeCatalyst
- 了解 CodeCatalyst 信任模型

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设施上的内容的控制。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，我们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“姓名”字段）中输入机密或敏感信息，例如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、项目或部署舰队名称中输入机密或敏感信息。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账单或诊断日志，也可能包含在 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 AWS SDK。

如果您提供外部服务器的 URL，我们强烈建议您不要在 URL 中包含任何安全凭据信息，以验证您对该服务器的请求。

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 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。所有数据在传输过程中和静态时都经过加密。服务创建或存储的任何数据，包括服务的任何元数据，均以本机方式存储在服务中并经过加密。

> **Note**
>
> 虽然有关问题的信息安全地存储在服务中，但有关未解决问题的信息也存储在浏览器的本地缓存中，您可以在其中查看议题板、待办事项和个别问题。为了获得最佳安全性，请务必清除浏览器缓存以删除此信息。

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 工作流程操作的最佳实践 CodeCatalyst。

密钥管理

CodeCatalyst 不支持密钥管理。

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。项目数据和元数据永远不会离开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AWS 区域时选择美国西部（俄勒冈），则您的数据将不会复制到中国区域的区域或AWS GovCloud (US)。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，用于表示空间的名称，以确保空间名称的唯一性。这个值不是人类可读的，也不会暴露空格的实际名称
- 空间的唯一 ID
- 空间的元数据，可帮助跨空间导航
- 空间AWS 区域所在的位置
- 空间中所有项目的唯一 ID
- 表示用户在空间或项目中的角色的角色 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独一无二的 CodeCatalyst。请务必不要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。

身份与访问管理与亚马逊 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。这称为服务角色。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 中基于身份的策略
- IAM 中的策略操作

- IAM 中的策略资源
- IAM 中的策略条件密钥
- 基于身份的连接策略示例 CodeCatalyst
- 使用标签控制对账户连接资源的访问权限
- CodeCatalyst 权限参考
- 将服务相关角色用于 CodeCatalyst
- AWSAmazon 的托管政策 CodeCatalyst
- 使用 IAM 角色授予对项目 AWS 资源的访问权限

IAM 中基于身份的策略

基于身份的策略是您可以附加到身份的 JSON 权限策略文档。该身份可以是一个用户、一组用户或一个角色。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。

适用于 CodeCatalyst 的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，哪位委托人可以在什么资源上执行哪些操作，在什么条件下可以执行哪些操作。

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，哪位委托人可以在什么资源上执行哪些操作，在什么条件下可以执行哪些操作。

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 中的策略条件密钥

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，哪位委托人可以在什么资源上执行哪些操作，在什么条件下可以执行哪些操作。

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。账户关联用于授权向空间添加 AWS 账户 内容。在互联中使用基于身份的策略。 AWS 账户

默认情况下，用户和角色无权创建或修改 CodeCatalyst资源。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。IAM 管理员必须创建 IAM policy，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。

以下示例 IAM 策略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 Management Console 与请求中指定的帐户相同的帐户登录。

```
{
  "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，您可以通过选择中的选项来创建CodeCatalystWorkflowDevelopmentRole-***spaceName***角色 AWS Management Console。在示例策略中，`iam:PassRole`操作的条件包括的服务委托人。CodeCatalyst只有具有该访问权限的角色才会在中创建 AWS Management Console。

```
{
    "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：拒绝管理连接

以下权限策略不允许用户管理 CodeCatalyst 和之间的连接 AWS 账户。

```
{
    "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权限

标签可以附加到资源上，也可以在请求中传递给支持标记的服务。策略中的资源可以有标签，策略中的某些操作可以包含标签。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以下：

- 根据连接资源已有的标签，哪些用户可以对连接资源执行操作。
- 哪些标签可以在操作的请求中传递。
- 是否特定标签键可在请求中使用。

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 中的策略条件密钥。

示例 1：允许根据请求中的标签执行操作

以下策略向用户授予批准账户连接的权限。

为此，如果请求指定一个名为 Project 的带有值为 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。

```
{
  "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：允许基于资源标签的操作

以下策略授予用户对账户连接资源执行操作和获取相关信息的权限。

为此，如果连接的标签名Project为该值，则允许执行特定操作ProjectA。（aws:ResourceTag 条件键用于控制可以通过 IAM 请求传递哪些标签。）

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "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。以下部分介绍与连接账户相关的仅限权限的操作。

账户连接所需的权限

使用账户连接需要以下权限。

CodeCatalyst 账户连接权限	所需的权限	资源
AcceptConnection	需要接受将此账户连接到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 (*)。
AssociateIamRoleToConnection	需要将 IAM 角色关联到账户连接。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DeleteConnection	删除账户连接所必需的。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DisassociateIamRoleFromConnection	需要解除 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	用于描述账户连接的账单授权。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	需要建立账户连接。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecatalyst:region: *account_I*

CodeCatalyst 账户连接权限	所需的权限	资源
		D :/connect ions/ *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 (*) 。
ListConnections	需要列出非待处理的账户连接。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	列出与账户连接关联的 IAM 角色所必需的。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*
ListTagsForResource	列出与账户连接关联的标签所必需的。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*
PutBillingAuthorization	需要为账户关联创建或更新账单授权。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *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	创建或编辑与账户连接关联的标签所必需的。这只是一种 IAM 策略权限，不是 API 操作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*

CodeCatalyst 账户连接权限	所需的权限	资源
UntagResource	需要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*

IAM 身份中心应用程序所需的权限

使用 IAM 身份中心应用程序需要以下权限。

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	arn:aws:codecatalyst:region: *account_I*

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
	程序的关联。这只是一种 IAM 策略权限，不是 API 操作。	*D* :/identity-center-applicati ons/ *identity-center-application_ID*
CreateIdentityCenterApplication	创建 IAM 身份中心应用程序所必需的。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*
CreateSpaceAdminRoleAssignment	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*
DeleteIdentityCenterApplication	需要删除 IAM 身份中心应用程序。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*
DisassociateIdentityCenterApplicationFromSpace	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*

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
DisassociateIdentityFromIdentityCenterApplication	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:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
GetIdentityCenterApplication	获取有关 IAM 身份中心应用程序的信息所必需的。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*
ListIdentityCenterApplications	需要查看账户中所有 IAM 身份中心应用程序的列表。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*

CodeCatalyst IAM 身份中心应用程序的权限	所需的权限	资源
SynchronizeIdentityCenterApplication	需要将 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`
UpdateIdentityCenterApplication	更新 IAM 身份中心应用程序所必需的。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`

将服务相关角色用于 CodeCatalyst

亚马逊 CodeCatalyst 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与之直接关联的 IAM 角色的独特类型。 CodeCatalyst服务相关角色由服务预定义 CodeCatalyst ，包括该服务代表您调用其他AWS服务所需的所有权限。

服务相关角色使设置变得 CodeCatalyst 更加容易，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，除非另有定义，否则 CodeCatalyst 只能担任其角色。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。这样可以保护您的 CodeCatalyst 资源，因为您不会无意中删除访问资源的权限。

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。选择是，可转到查看该服务的服务相关角色文档的链接。

的服务相关角色权限 CodeCatalyst

CodeCatalyst 使用名为的服务相关角色 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization— 允许 Amazon 代表您对应用程序实例配置文件以及关联的目录用户和群组进行 CodeCatalyst 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

名为的角色权限策略 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy CodeCatalyst 允许对指定资源完成以下操作：

- 操作：View application instance profiles and associated directory users and groups对于 CodeCatalyst spaces that support identity federation and SSO users and groups

您必须配置允许用户、组或角色创建、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。

为创建服务相关角色 CodeCatalyst

您无需手动创建服务相关角色。当您在AWS Management Console、或 AWS API 中创建空间时，CodeCatalyst 会为您创建服务相关角色。AWS CLI

> **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。另外，如果您在 2023 年 11 月 17 日该 CodeCatalyst 服务开始支持服务相关角色之前使用该服务，则在您的账户中 CodeCatalyst 创建了该 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。创建空间时， CodeCatalyst 会再次为您创建服务相关角色。

您还可以使用 IAM 控制台通过 View 应用程序实例配置文件和关联的目录用户和群组用例创建服务相关角色。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。

编辑的服务相关角色 CodeCatalyst

CodeCatalyst 不允许您编辑 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。

删除的服务相关角色 CodeCatalyst

无需手动删除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。当您删除AWS Management Console、或 AWS API 中的空格时，会 CodeCatalyst 清理资源并为您删除服务相关角色。AWS CLI

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**
>
> 如果您尝试删除资源时 CodeCatalyst 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

要删除使用的 CodeCatalyst 资源 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization

- 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 服务相关角色支持的区域

CodeCatalyst 支持在提供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	Support in CodeCatalyst
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
美国西部（北加利福尼亚）	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

区域名称	区域标识	Support in CodeCatalyst
中东（巴林）	me-south-1	否
中东（阿联酋）	me-central-1	否
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

AWSAmazon 的托管政策 CodeCatalyst

AWS 托管式策略是由 AWS 创建和管理的独立策略。AWS 托管式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 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

无法更改AWS托管式策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管式策略：AmazonCodeCatalystSupportAccess

该政策向所有空间管理员和空间成员授予使用与空间账单账户关联的商业版或企业版高级支持计划的权限。这些权限允许空间管理员和成员使用高级支持计划，为他们在权限策略中拥有权限的资源提供 CodeCatalyst 权限。

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。

```
{
  "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 托管式策略：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 Management Console。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 Management Console。

```
{
    "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 托管式策略：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 Management Console。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 Management Console。

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

AWS 托管式策略：
AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy

您不能将AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy; 附加到您的 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，请参阅将服务相关角色用于 CodeCatalyst。

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以及 SSO 用户和群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。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid":
 "AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy",
   "Effect": "Allow",
   "Action": [
    "sso:ListInstances",
    "sso:ListApplications",
    "sso:ListApplicationAssignments",
    "sso:DescribeInstance",
    "sso:DescribeApplication"
   ],
   "Resource": "*"
  }
 ]
}
```

对 AWS 托管式策略的 CodeCatalyst 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。要获得有关此页面变更的自动提醒，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。	2023 年 11 月 17 日

更改	描述	日期
	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。	
AmazonCodeCatalyst SupportAccess：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
AmazonCodeCatalyst FullAccess：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
AmazonCodeCatalyst ReadOnlyAccess：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 已开始跟踪更改	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。然后，您可以创建以下服务角色并在连接账户时将其关联。

有关您在 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，也可以创建自定义服务角色并手动配置这些 IAM 策略和角色。最佳做法是，为这些角色分配所需的最少权限。

> **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AWS Support适用于 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。这称为服务角色。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。服务角色的范围仅限于空间，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。每个账户中只能为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。

该CodeCatalystWorkflowDevelopmentRole-*spaceName*角色所附的政策旨在与使用该空间中的蓝图创建的项目合作。它允许这些项目中的用户使用连接中的资源开发、构建、测试和部署代码 AWS 账户。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

按照以下步骤创建将用于您空间中的工作流程的CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。对于您想要在项目中使用的 IAM 角色的每个账户，您都必须向自己的空间添加一个角色，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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。
5. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建了一个服务角色，其中包含开发角色的权限策略和信任策略。该角色将有一个名字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，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。

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。这称为支持服务角色。创建支持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，然后为该角色选择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。服务角色的范围仅限于空间，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。

附加到该AWSRoleForCodeCatalystSupport角色的策略是提供对支持权限的访问权限的托管策略。有关更多信息，请参阅 AWS 托管式策略：AmazonCodeCatalystSupportAccess。

策略的信任角色 CodeCatalyst 允许代入该角色。

```
{
    "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

按照以下步骤创建将在您所在空间中用于支持案例的AWSRoleForCodeCatalystSupport角色。必须将该角色添加到该空间的指定账单账户中。

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 Management Console，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 Settings (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。将显示AWS 账户 详细信息页面。
4. 从中选择 “管理角色” AWS Management Console。

   “将 IAM 角色添加到 Amazon CodeCatalyst 空间” 页面将在中打开 AWS Management Console。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。此选项创建的服务角色包含预览版开发角色的权限策略和信任策略。该角色的AWSRoleForCodeCatalystSupport名称将附加唯一标识符。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程操作创建角色。创建 IAM 角色后，复制角色 ARN，将 IAM 角色添加到您的账户连接中，并将其与您的项目环境关联。要了解更多信息，请参阅向账户连接添加 IAM 角色。

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 AWS CloudFormation 资源上运行任务。 AWS 账户

此角色授予执行以下操作的权限：

- 写入亚马逊 S3 存储桶。
- 使用 Support 来支持资源构建 AWS CloudFormation。这需要亚马逊 S3 访问权限。

此角色使用以下策略：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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 AWS 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 AWS CloudFormation 资源上运行任务。 AWS 账户

此角色授予执行以下操作的权限：

- 使用 Support 来支持资源构建 AWS 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 AWS CloudFormation。

应将以下 AWS 托管策略附加到此角色：

- AWSCloudFormationFullAccess
- IAM FullAccess
- 亚马逊 3 FullAccess
- 亚马逊 API GatewayAdministrator
- 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 AWS 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的权限：

- 写入亚马逊 S3 存储桶。
- Support 构建 CDK 构造和 AWS CloudFormation 资源堆栈。这需要访问 Amazon S3 以进行项目存储，访问 Amazon ECR 以获得图像存储库支持，并访问 SSM 以进行系统管理和监控虚拟实例。

此角色使用以下策略：

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

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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 CloudFormation

对于使用 CodeCatalyst 的工作流程部署操作 AWS 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 AWS CloudFormation AWS 账户资源运行任务。

此角色授予执行以下操作的权限：

- CodeCatalyst 允许调用 λ 函数以通过执行蓝/绿部署。 AWS CloudFormation
- CodeCatalyst 允许在中创建和更新堆栈和变更集。 AWS CloudFormation

此角色使用以下策略：

```
{"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**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 为 Amazon EC2 部署角色

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 EC2 Auto Scaling 的权限。

此角色授予执行以下操作的权限：

- 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 Amazon EC2 实例。
- 读取、创建、更新和删除 Amazon EC2 Auto Scaling 组、生命周期挂钩和扩展策略。
- 将信息发布到 Amazon SNS 主题。
- 检索有关 CloudWatch 警报的信息。
- 阅读并更新 Elastic Load Balancing。

此角色使用以下策略：

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
 "autoscaling:CompleteLifecycleAction",
"autoscaling:DeleteLifecycleHook",
"autoscaling:DescribeAutoScalingGroups",
"autoscaling:DescribeLifecycleHooks",
"autoscaling:PutLifecycleHook",
"autoscaling:RecordLifecycleActionHeartbeat",
"autoscaling:CreateAutoScalingGroup",
"autoscaling:UpdateAutoScalingGroup",
"autoscaling:EnableMetricsCollection",
"autoscaling:DescribePolicies",
"autoscaling:DescribeScheduledActions",
"autoscaling:DescribeNotificationConfigurations",
"autoscaling:SuspendProcesses",
"autoscaling:ResumeProcesses",
"autoscaling:AttachLoadBalancers",
"autoscaling:AttachLoadBalancerTargetGroups",
"autoscaling:PutScalingPolicy",
"autoscaling:PutScheduledUpdateGroupAction",
"autoscaling:PutNotificationConfiguration",
"autoscaling:PutWarmPool",
"autoscaling:DescribeScalingActivities",
"autoscaling:DeleteAutoScalingGroup",
"ec2:DescribeInstances",
"ec2:DescribeInstanceStatus",
"ec2:TerminateInstances",
"tag:GetResources",
"sns:Publish",
"cloudwatch:DescribeAlarms",
"cloudwatch:PutMetricAlarm",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
"elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
"elasticloadbalancing:DescribeTargetGroups",
"elasticloadbalancing:DescribeTargetHealth",
"elasticloadbalancing:RegisterTargets",
```

```
"elasticloadbalancing:DeregisterTargets"
],
"Resource": "resource_ARN"
      }
   ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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。

此角色使用以下策略：

```
{
    "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**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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。

此角色使用以下策略：

```
*{*
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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。

此角色使用以下策略：

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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 AWS CloudFormation 资源。

此角色授予执行以下操作的权限：

- 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。 AWS CloudFormation

此角色使用以下策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Note

首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，然后在策略可用后使用资源名称缩小策略范围。

```
"Resource": "*"
```

CodeCatalyst Amazon EC2 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 EC2 资源上运行任务。该CodeCatalystWorkflowDevelopmentRole-*spaceName*服务角色不包括对 Amazon EC2 的权限或所描述的针对亚马逊的操作 CloudWatch。

此角色授予执行以下操作的权限：

- 获取 Amazon EC2 实例的状态。
- 获取 Amazon EC2 实例的 CloudWatch 指标。

此角色使用以下策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe",
            "Resource": "resource_ARN"
        }
    ]
```

```
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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的权限：

- 阅读 Amazon ECS 任务集。
- 检索有关 CloudWatch 警报的信息。

此角色使用以下策略：

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
       {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeRules"
            ],
            "Resource": "resource_ARN",
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam:::role/ecsTaskExecutionRole",
                "arn:aws:iam:::role/ECSTaskExecution"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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

此角色授予以下权限：

- 阅读 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
```

```
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，然后在策略可用后使用资源名称缩小策略范围。
>
> ```
> "Resource": "*"
> ```

手动为工作流程操作创建角色

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。

按照以下步骤在 IAM 中创建这些角色。

创建部署角色

1. 为该角色创建策略，如下所示：
    a. 登录到 AWS。
    b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。
    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：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
```

```
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 按如下方式创建部署角色：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

```
{
```

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

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. 在 “角色名称” 中，输入：

```
codecatalyst-deploy-role
```

i. 在角色描述中，输入：

```
CodeCatalyst deploy role
```

j. 选择 创建角色。

现在，您已经创建了一个带有信任策略和权限策略的部署角色。

3. 按如下方式获取部署角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（codecatalyst-deploy-role）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的部署角色并获得了其 ARN。

创建生成角色

1. 为该角色创建策略，如下所示：

   a. 登录到 AWS。

   b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

   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：

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

   > **Note**
   >
   > 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

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. 在 “角色名称” 中，输入：

```
codecatalyst-build-role
```

i. 在角色描述中，输入：

```
CodeCatalyst build role
```

j. 选择 创建角色。

现在，您已经创建了一个包含信任策略和权限策略的构建角色。

3. 按如下方式获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（`codecatalyst-build-role`）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的构建角色并获得了其 ARN。

创建堆栈角色

> **Note**
>
> 尽管出于安全考虑，建议您创建堆栈角色，但您不必创建堆栈角色。如果您未创建堆栈角色，则需要将本过程中进一步描述的权限策略添加到部署角色中。

1. AWS 使用要部署堆栈的账户登录。
2. 通过 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. 在顶部，选择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，可以暂时将其省略。在测试操作时，如果您没有正确的权限，则 AWS 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. 在 “角色名称” 中，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，输入您刚刚创建的角色的名称（`codecatalyst-stack-role`）。
    c. 从列表中选择角色。
    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 AWS 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。

按照以下步骤在 IAM 中创建角色。

创建生成角色

1. 为该角色创建策略，如下所示：

a. 登录到 AWS。

b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

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：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

e. 选择下一步。

f. 将权限策略附加到构建角色。在添加权限页面的权限策略部分，搜索codecatalyst-webapp-build-policy并选中其复选框。

g. 选择下一步。

h. 在 “角色名称” 中，输入：

```
codecatalyst-webapp-build-role
```

i. 在角色描述中，输入：

```
CodeCatalyst Web app build role
```

j. 选择 创建角色。

现在，您已经创建了一个包含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. codecatalyst-webapp-build-role选择显示其详细信息。

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。

现在，您已将权限策略附加到构建角色。构建角色现在有两个策略：权限策略和信任策略。

4. 按如下方式获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（ codecatalyst-webapp-build-role ）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的构建角色并获得了其 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。

按照以下步骤在 IAM 中创建角色。

为创建生成角色 CloudFormation

1. 为该角色创建策略，如下所示：

a. 登录到 AWS。

b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

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：

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

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

   a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

   e. 选择下一步。

   f. 将权限策略附加到构建角色。在添加权限页面的权限策略部分，搜索codecatalyst-SAM-build-policy并选中其复选框。

   g. 选择下一步。

   h. 在 “角色名称” 中，输入：

```
codecatalyst-SAM-build-role
```

   i. 在角色描述中，输入：

```
CodeCatalyst SAM build role
```

   j. 选择 创建角色。

现在，您已经创建了一个包含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. codecatalyst-SAM-build-role选择显示其详细信息。

   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。

   现在，您已将权限策略附加到构建角色。构建角色现在有两个策略：权限策略和信任策略。

4. 按如下方式获取构建角色 ARN：

   a. 在导航窗格中，选择角色。

   b. 在搜索框中，输入您刚刚创建的角色的名称（ codecatalyst-SAM-build-role ）。

   c. 从列表中选择角色。

   此时将显示该角色的 “摘要” 页面。

   d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的构建角色并获得了其 ARN。

为 SAM 创建部署角色

1. 为该角色创建策略，如下所示：

   a. 登录到 AWS。

   b. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。

   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：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 首次使用该角色运行工作流程操作时，请在资源策略语句中使用通配符，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. 创建生成角色，如下所示：

a. 在导航窗格中，选择 Roles（角色），然后选择 Create role（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：

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

e. 选择下一步。

f. 将权限策略附加到构建角色。在添加权限页面的权限策略部分，搜索`codecatalyst-SAM-deploy-policy`并选中其复选框。

g. 选择下一步。

h. 在 “角色名称” 中，输入：

```
codecatalyst-SAM-deploy-role
```

i. 在角色描述中，输入：

```
CodeCatalyst SAM deploy role
```

j. 选择 创建角色。

现在，您已经创建了一个包含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. codecatalyst-SAM-deploy-role选择显示其详细信息。

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。

现在，您已将权限策略附加到构建角色。构建角色现在有两个策略：权限策略和信任策略。

4. 按如下方式获取构建角色 ARN：

a. 在导航窗格中，选择角色。

b. 在搜索框中，输入您刚刚创建的角色的名称（ codecatalyst-SAM-deploy-role ）。

c. 从列表中选择角色。

此时将显示该角色的 “摘要” 页面。

d. 在顶部，复制 ARN 值。

现在，您已经创建了具有相应权限的构建角色并获得了其 ARN。

Amazon 合规性验证 CodeCatalyst

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规性 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合AWS 规资源 — 此工作簿和指南集可能适用于您的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

Amazon 的弹性 CodeCatalyst

AWS 全球基础设施围绕 AWS 区域和可用区而构建。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅 AWS 全球基础设施。要详细了解跨哪些 CodeCatalyst 数据进行复制AWS 区域，请参阅Amazon 的数据保护 CodeCatalyst。

Amazon 的基础设施安全 CodeCatalyst

作为一项托管服务，Amazon CodeCatalyst 受到AWS全球网络安全的保护。有关 AWS 安全服务以及 AWS 如何保护基础架构的信息，请参阅 AWS 云安全。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已发布的 API 调用 CodeCatalyst 通过网络进行访问。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（Ephemeral Diffie-Hellman）或 ECDHE（Elliptic Curve Ephemeral Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

Amazon 中的配置和漏洞分析 CodeCatalyst

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。

您在亚马逊上的数据和隐私 CodeCatalyst

Amazon 非常 CodeCatalyst 重视您的隐私，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，请参阅中的请求您的数据AWS 一般参考。

正在删除您的AWS建筑商 ID 个人资料

删除您的个人资料是一项永久性操作，无法撤销。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。此过程最多可能需要 90 天才能完成。

删除您的个人资料后，您将无法在 Amazon 上访问或恢复您的数据 CodeCatalyst。这包括个人访问令牌、角色、用户会员资格以及您作为唯一成员的任何 Amazon CodeCatalyst 空间。您无法再登录亚马逊 CodeCatalyst。

有关如何删除您的AWS建筑商 ID 个人资料的信息，请参阅中的删除您的AWS建筑商 ID AWS 一般参考。

Amazon 工作流程操作的最佳实践 CodeCatalyst

在开发工作流程时，有许多安全最佳实践需要考虑 CodeCatalyst。以下是一般指导方针，并不代表完整的安全解决方案。这些最佳实践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

请勿在您的 YAML 中嵌入敏感信息。我们建议您使用密钥，而不是在您的 YAML 中嵌入凭证、CodeCatalyst 密钥或令牌。密钥提供了一种在 YAML 中存储和引用敏感信息的简便方法。

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，可以在项目、空间或更广泛的社区中共享。使用他人的代码可以极大地提高便利性和效率，但也会引入新的威胁载体。请查看以下章节，确保您遵循最佳实践，以确保 CI/CD 工作流程的安全。

GitHub 行动

GitHub Actions 是开源的，由社区构建和维护。我们遵循责任共担模型，并将 A GitHub ctions 源代码视为由您负责的客户数据。 GitHub 可以向操作授予访问密钥、存储库令牌、源代码、账户链接和计算时间的权限。确保您对计划执行的 GitHub 操作的可信度和安全性充满信心。

针对 GitHub 操作的更具体的指导和安全最佳实践：

- 安全加固
- 阻止 pwn 请求
- 不可信的输入
- 如何信任你的积木

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。角色信任策略是必需的基于资源的策略（将附加到 IAM 中的角色）。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。

在以下信任策略中，`Principal`元素中列出的服务主体被授予基于资源的策略的权限，该`Condition`区块用于限制对范围缩小资源的访问权限。

```
"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 政策声明中的空间 ID 用作资源 ARN。

最佳做法是在策略中尽可能缩小权限范围。

- 您可以使用aws:SourceArn条件键中的通配符 (*) 来指定空间中的所有项目。project/*
- 您可以使用在aws:SourceArn条件键中为空间中的特定项目指定资源级权限。project/*projectId*

的服务负责人 CodeCatalyst

您可以使用基于资源的 JSON 策略中的Principal元素来指定允许或拒绝访问资源的委托人。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该Principal元素；同样，您也无法将用户组标识为策略（例如基于资源的策略）中的委托人，因为组与权限而不是身份验证有关，并且委托人是经过身份验证的 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

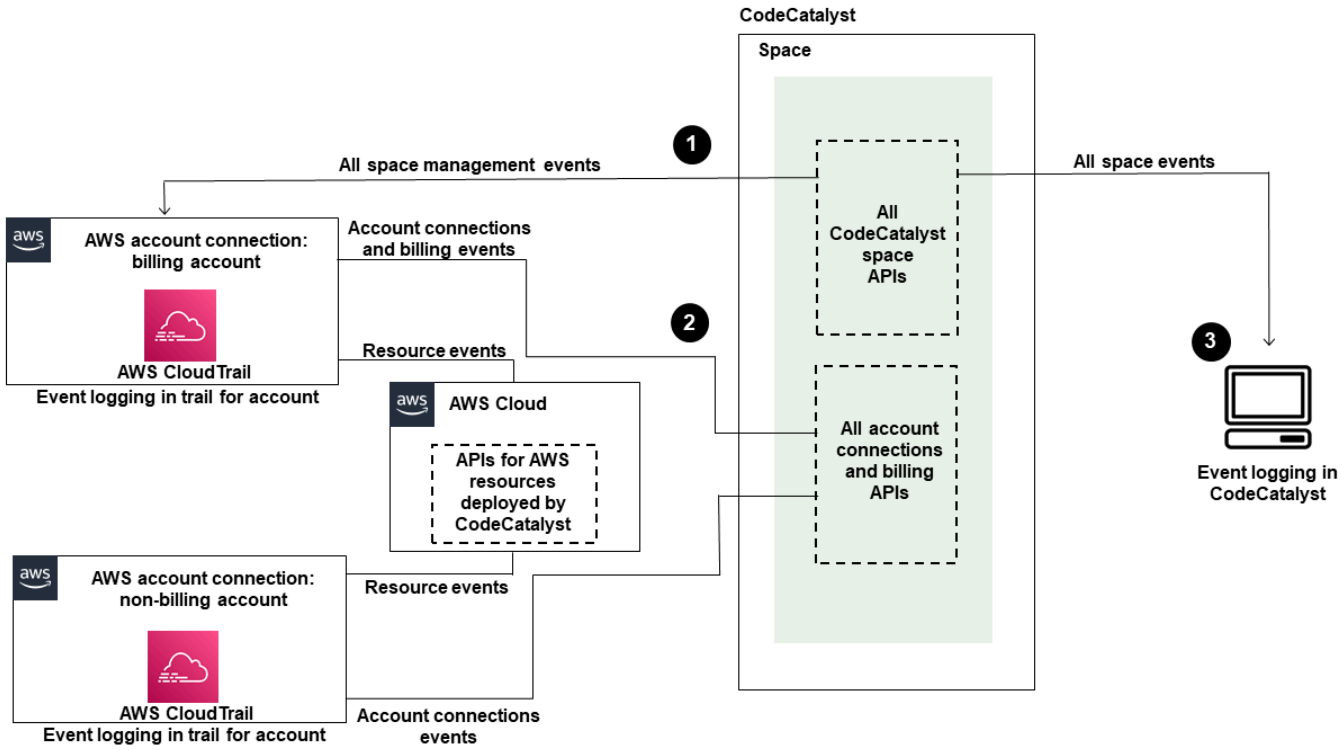
有关更多信息，请参阅《IAM 用户指南》中的 AWS JSON 策略元素：主体。

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设置的跟踪和指定存储桶进行记录 AWS 账户。

下图显示了账单账户如何登录空间的所有管理事件，而相应账户的账户连接/账单事件和 AWS 资源事件是如何登录 CloudTrail 的。 CloudTrail



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。如果账单账户更改为另一个 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CodeCatalyst 信息在 CloudTrail。
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CodeCatalyst 信息在 CloudTrail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 还可以使用中的list-event-logs命令查看空间中所有 CodeCatalyst API 的事件 AWS CLI。

主题

- AWS 账户 使用 AWS CloudTrail 日志记录监控 API 调用
- 使用事件记录访问已记录的事件

AWS 账户 使用 AWS CloudTrail 日志记录监控 API 调用

CodeCatalyst Amazon 与 AWS CloudTrail一项服务集成，该服务可记录用户、角色或用户所采取的操作 AWS 服务。 CloudTrail 将代表连接的 API 调用捕获 AWS 账户 为事件。 CodeCatalyst 如果您创建了跟踪，则可以启用向 S3 存储桶持续传输事件，包括的事件 CodeCatalyst。 CloudTrail 如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。

CodeCatalyst 支持将以下操作作为事件记录在 CloudTrail 日志文件中：

- CodeCatalyst 空间的管理事件将记录在空间 AWS 账户 的指定账单账户中。有关更多信息，请参阅 CodeCatalyst 太空事件。

  > **Note**
  >
  > 可以使用 CLI 访问 CodeCatalyst 空间的数据事件，详情请参见使用事件记录访问已记录的事件。

- 在连接中发生 CodeCatalyst 的工作流操作中使用的资源事件 AWS 账户 将作为事件记录在连接中 AWS 账户。有关更多信息，请参阅 CodeCatalyst 账户连接和账单事件。

> **Important**
>
> 虽然可以将多个账户与一个空间关联，但在 CodeCatalyst 空间和项目中 CloudTrail 登录事件仅适用于该结算帐号。

空间账单账户 AWS 账户 是针对超出 AWS 免费套餐的 CodeCatalyst 资源收取的费用。一个空间可以关联多个账户，而只有一个账户可以作为指定的结算账号。该空间的账单账户或其他关联账户可以具有 IAM 角色，用于通过 CodeCatalyst 工作流程部署 AWS 资源和基础设施，例如 Amazon ECS 集群或 S3 存储桶。您可以使用工作流程 YAML 来识别您部署到 AWS 账户 的。

> **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。 CloudTrail 必须为每个部署资源 AWS 账户 的地方单独配置日志记录。

CodeCatalyst 登录关联账户包括以下注意事项：

- 对 CloudTrail 事件的访问由关联账户中的 IAM 管理，而不是在 CodeCatalyst。
- 第三方连接（例如链接到 GitHub 存储库）将导致第三方资源名称记录在 CloudTrail 日志中。

> **Note**
>
> CloudTrail CodeCatalyst 事件的记录是在空间级别进行的，不会按项目边界隔离事件。

有关的更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> 本节介绍 CloudTrail 记录在已登录 CodeCatalyst空间中的所有事件以及与之连接 AWS 账户 的事件 CodeCatalyst。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 使用事件记录访问已记录的事件。

CodeCatalyst 太空事件

CodeCatalyst 用于管理空间级和项目级资源的操作记录在空间的账单账户中。在 CloudTrail 记录 CodeCatalyst 空间时，记录事件时要考虑以下注意事项。

- CloudTrail 事件适用于整个空间，不限于任何单个项目。
- 当你连接到 CodeCatalyst 空间时 AWS 账户 ，账户连接的可记录事件将在该空间中记录。 AWS 账户启用此连接后，您将无法将其禁用。
- 当您将 AWS 账户 连接到某个 CodeCatalyst 空间并将其指定为该空间的结算账户时，将在该空间中记录事件 AWS 账户。启用此连接后，您将无法将其禁用。

  空间级和项目级资源的事件仅记录在结算账号中。要更改 CloudTrail 目标账户，请在中更新结算账号 CodeCatalyst。在下一个月度账单周期开始时，更改将对中的新账单账户生效 CodeCatalyst。之后， CloudTrail目标账户即会更新。

以下是中 AWS 与管理空间级和项目级资源的操作相关的事件示例。 CodeCatalyst以下 API 是通过 SDK 和 CLI 发布的。活动将记录在 CodeCatalyst 空间的 AWS 账户 指定账单账户中。

- CreateDevEnvironment
- CreateProject
- DeleteDevEnvironment
- GetDevEnvironment
- GetProject
- GetSpace
- GetSubscription
- ListDevEnvironments
- ListDevEnvironmentSessions
- ListEventLogs
- ListProjects
- ListSourceRepositories
- StartDevEnvironment
- StartDevEnvironmentSession
- StopDevEnvironment
- StopDevEnvironmentSession
- UpdateDevEnvironment

CodeCatalyst 账户连接和账单事件

以下是中 AWS 与账户关联操作或账单相关的事件示例： CodeCatalyst

- AcceptConnection
- AssociateIAMRoletoConnection
- DeleteConnection
- DissassociateIAMRolefromConnection
- GetBillingAuthorization
- GetConnection
- GetPendingConnection
- ListConnections
- ListIAMRolesforConnection

- PutBillingAuthorization
- RejectConnection

CodeCatalyst 信息在 CloudTrail

CloudTrail 在您创建该账户 AWS 账户 时已启用。当您将其连接到空间时，该 CodeCatalyst 空间中发生的事件将 CloudTrail 记录 AWS 账户 在该 AWS 账户中。 AWS 账户 中的 CodeCatalyst 可记录事件与该账户中的其他可记录 CloudTrail 事件一起记录在已连接账户的 CloudTrail 日志和 CloudTrail 控制台 AWS 的事件历史记录中。

每个事件或日记账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其 AWS 生成器 ID 发出。
- 请求是使用根证书还是 AWS Identity and Access Management (IAM) 用户凭证发出。
- 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 用户身份元素。

访问 CloudTrail 事件

要持续记录您的事件 AWS 账户，包括中的 CodeCatalyst活动事件 AWS 账户，请创建跟踪。跟踪允许 CloudTrail将日志文件传送到 S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。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
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个地区的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求

参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

中的 CodeCatalyst 账户连接事件示例 AWS

以下示例显示了演示该`ListConnections`操作的 CloudTrail 日志条目。对于 AWS 账户 已连接到空间的，`ListConnections`则用于查看与此相关的所有账户连接 AWS 账户。 CodeCatalyst 事件将记录在中 AWS 账户 指定的中`accountId`，其值`arn`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"
}
```

中的 CodeCatalyst 项目资源事件示例 AWS

以下示例显示了演示该CreateDevEnvironment操作的 CloudTrail 日志条目。与空间关联且是该空间的指定账单账户的，用于空间中的项目级活动，例如创建开发环境。 AWS 账户

在accountId字段下userIdentity，这是托管所有 AWS 生成器 ID 身份的身份池的 IAM 身份中心账户 ID (432677196278)。此账户 ID 包含有关活动 CodeCatalyst 用户的以下信息。

- 该type字段表示请求的 IAM 实体的类型。对于空间和项目资源 CodeCatalyst 的事件，此值为IdentityCenterUser。该accountId字段指定拥有用于获取凭证的实体的账户。
- 该userId字段包含用户的 AWS 生成器 ID 标识符。
- 该identityStoreArn字段包含身份存储账户和用户的角色 ARN。

该recipientAccountId字段包含该空间账单账户的账户 ID，此处的示例值为 111122223333。

有关更多信息，请参阅CloudTrail 用户身份元素。

```
{
 "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**
>
> 在某些情况下，可能不知道用户代理。在这种情况下， CodeCatalyst 将在 CloudTrail 事件的`userAgent`字段`Unknown`中提供一个值。

查询您的 CodeCatalyst 活动轨迹

您可以使用 Amazon Athena 中的查询表创建和管理对 CloudTrail 日志的查询。有关创建查询的更多信息，请参阅 Amazon Athen a 用户指南中的查询 AWS CloudTrail 日志。

使用事件记录访问已记录的事件

当用户在 Amazon 中执行操作时 CodeCatalyst，这些操作会被记录为事件。您可以使用 AWS CLI 来查看空间中指定时间范围内的事件日志。您可以查看这些事件以查看在空间中执行的操作，包括操作的日期和时间、执行操作的用户的姓名以及用户发出请求的 IP 地址。

> **Note**
>
> CodeCatalyst 空间的管理事件是为关联 CloudTrail 的结算账号登录的。有关由记录的 CodeCatalyst 管理事件的更多信息 CloudTrail，请参阅CodeCatalyst 信息在 CloudTrail。

要查看空间的事件日志，您必须已安装并配置了 AWS CLI 空间的配置文件 CodeCatalyst，并且必须具有该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**
>
> 要查看代表连接 CodeCatalyst 中发生的事件的日志记录 AWS 账户，或者查看关联账单账户中空间或项目资源的事件日志记录，可以使用 AWS CloudTrail。有关更多信息，请参阅 AWS 账户 使用 AWS CloudTrail 日志记录监控 API 调用。

1. 打开终端或命令行并运行aws codecatalyst list-event-logs命令，指定：

   - 带有**--space-name**选项的空间的名称。
   - 您要开始查看事件的日期和时间，采用协调世界时 (UTC) 时间戳格式，如 RFC 3339 中指定，带有选项。**--start-time**
   - 您想要停止查看事件的日期和时间，采用协调世界时 (UTC) 时间戳格式，如 RFC 3339 中指定，带有选项。**--end-time**
   - （可选）在单个响应中返回的最大结果数，带**--max-results**选项。如果结果数大于您指定的数字，则响应中将包含一个可用于返回下一个结果的nextToken元素。
   - （可选）使用**--event-name**选项，将结果限制为要返回的特定事件类型。

   此示例返回名为 *2022-11-30 ExampleCorp# 2022- 12-01* 期间的空间中记录的事件，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. 如果事件发生在此时间范围内，则该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选项和返回的令牌的值再次运行该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，请参阅的配额 CodeCatalyst。

资源	信息
中的别名 CodeCatalyst	允许的字符的任意组合，长度介于 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。别名不能： • 包含少于 3 个字符 • 包含空格或以下任意字符：? ^ * [\ ~ :
用户每天发送的最大邀请数量	500

资源	信息
每天向一个电子邮件地址发送的最大邀请数量	25
每位用户的最大个人访问令牌 (PAT) 数量	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	长度介于 8 到 64 个字符之间的允许字符的任意组合。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	1 到 100 个字符之间允许的任意字符组合
距离项目成员邀请到期的时间	24 小时后过期
距离空间成员邀请到期的时间	24 小时后过期
距离电子邮件地址验证到期的时间	发送 10 分钟后过期

故障排除

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。

注册时遇到问题

注册时您可能会遇到一些问题。我们有一些解决方案。

我的电子邮件地址已被使用

如果您输入的电子邮件已在使用中，并且您认出它是您自己的电子邮件，那么您可能已经有了我们的个人资料。使用此现有身份登录。如果您不拥有现有电子邮件，请使用另一封未使用的电子邮件进行注册。

我无法完成电子邮件验证

如果您还没有收到验证邮件

1. 检查您的群发邮件文件夹、垃圾邮件文件夹和已删除电子邮件文件夹。

   > **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，而不将其视为垃圾邮件或群发邮件。

2. 等待 5 分钟，刷新您的收件箱。再次检查您的垃圾邮件、垃圾邮件和已删除邮件文件夹。
3. 如果您仍然看不到验证电子邮件，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。

我的密码不符合最低要求

为了安全起见，您的密码必须包含 8-20 个字符（包括大写和小写字母）以及数字。

登录时出现问题

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，都必须遵守以下要求：

- 密码区分大小写。
- 密码长度必须介于 8 到 64 个字符之间，包括大写和小写字母、数字以及至少一个非字母数字字符。
- 你不能重复使用最后三个密码 。

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 设备添加到您的配置文件中。

我无法添加 MFA 设备

如果您发现无法添加其他 MFA 设备，则该设备可能已达到您可以注册的 MFA 设备上限。在添加新 MFA 设备之前，您可能需要移除现有 MFA 设备。

我无法删除 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。更多有关添加其他 MFA 设备的信息，请参阅 如何注册设备以使用多因素身份验证。

退出时出现问题

我找不到在哪里退出

在页面的右上角，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。

由于工作流程失败，我收到 “角色不存在” 错误

问题：通过 Web 应用程序或无服务器蓝图创建项目后，工作流程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程权限的 IAM 角色并将该 IAM 角色添加到工作流程 YAML 后，工作流程仍会失败，因为可能需要将 IAM 角色添加到您的账户连接中。将 IAM 角色添加到您的空间的账户连接中，详情请参见向账户连接添加 IAM 角色。

我因工作流程失败而收到角色错误

问题：通过 Web 应用程序或无服务器蓝图创建项目后，工作流程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。如果您的空间已有活动 AWS 账户 连接，请创建并添加一个具有运行工作流程操作权限的 IAM 角色。将 IAM 角色添加到您的账户连接中，详情请参见向账户连接添加 IAM 角色。

可能的解决方案：如果项目是在未指定连接的情况下创建的，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 角色添加到您的部署环境中。

我需要在项目工作流程中更新 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程。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 Commit (提交)。

   如果在主线分支上，则工作流程会自动启动。否则，要重新运行工作流程，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，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。如果有审阅权限，则会显示审阅请求链接。
3. 选择链接，然后在出现提示时确认您的凭据。输入您的凭据并选择 “验证”。
4. 接受新权限，指明要应用权限的存储库，然后选择 “保存”。

如何填写支持表格？

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应用蓝图时，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，可帮助开发者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 项目时关联 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 数据中心存储库不兼容。

安装和配置 Bitbucket 存储库扩展程序后，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 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
- 使用蓝图创建项目或应用蓝图时，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 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规划、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

扩展的概念

以下是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扩展程序 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 工作空间，然后关联比特桶存储库。
- 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

本教程提供了以下三个任务的演练：

1. 安装GitHub 存储库、Bitbucket 存储库或 Jira 软件扩展程序。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 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 工作空间或 Jira 网站连接到。 CodeCatalyst

   > **Important**
   >
   > 要将您的 GitHub 账户、Bitbucket 工作空间或 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 人际关系 和 通过人际关系访问 GitHub 资源。

3. 将您的 GitHub 存储库、Bitbucket 存储库或 Jira 项目链接到您的 CodeCatalyst 项目。

> **Important**
>
> - 虽然您可以以贡献者的身份链接 GitHub 或 Bitbucket 存储库，但只能以 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。

> **Note**
>
> - GitHub 或 Bitbucket 存储库只能链接到空间中的一个 CodeCatalyst 项目。
> - 您不能在 CodeCatalyst 项目中使用空存储库、已存档存储库 GitHub 或 Bitbucket 存储库。
> - 您不能链接与 CodeCatalyst 项目中仓库同名的 GitHub 或 Bitbucket 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 数据中心存储库不兼容。
> - 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。

您还可以安装GitHub 存储库或 Bitbucket 存储库扩展，连接到您的 GitHub 账户或 Bitbucket 工作空间，并在创建新 CodeCatalyst 项目时链接第三方存储库。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 存储库和 Bitbucket 存储库允许您在中使用 GitHub 或 Bitbucket 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 存储库或 Jira So ftware。您也可以根据类别筛选附加信息。
4. （可选）要查看有关扩展程序的更多详细信息，例如扩展程序将拥有的权限，请选择扩展名称。
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。

安装扩展程序后，您将进入扩展详细信息页面。根据您安装的扩展程序，您可以查看和管理连接的提供商和关联的资源。

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 工作空间或 Jira 网站连接到您的空间。 CodeCatalyst

将您的 GitHub 账户、Bitbucket 工作空间或 Jira 网站连接到 CodeCatalyst

- 根据您安装的第三方扩展程序，执行以下任一操作：
  - 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 工作空间。
    1. 在 Conn ected Bitbucket 工作空间选项卡中，选择 Conn ect Bitbucket 工作空间以转到 Bitbucket 的外部
    2. 使用您的 Bitbucket 凭据登录到您的 Bitbucket
    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”。
       a. 如果您是首次授予工作区 CodeCatalyst 访问权限，请选择前往设置以启用 Bitbucket 的开发模式。
       b. 在工作区的 “已安装的应用程序” 页面上，选中 “启用开发模式” 复选框以允许安装 CodeCatalyst，然后从中重新连接 CodeCatalyst。

一旦您允许安装 CodeCatalyst，它就可以访问您在 Bitbucket 工作区中的所有存储库。

> **Tip**
>
> 如果您之前已将 Bitbucket 工作空间连接到该空间，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。为要允许的工作空间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。

- Jira Software：连接 Jira 站点。

  1. 在 Conn ected Jira 站点选项卡中，选择 Connect 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 Get started 进入最后一步。

  3. 根据您的角色，执行以下任一操作：

     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，系统会自动将您定向到下一步。

     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，请参阅谁可以安装应用程序？。

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

使用您的存储库 GitHub 或 Bitbucket 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 存储库或 Jira CodeCatalyst 项目链接到项目

- 根据您安装的第三方扩展程序和连接的提供商，执行以下任一操作：
  - 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**
       > 如果存储库的名称显示为灰色，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. 选择关联。

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 存储库和 Jira 项目 CodeCatalyst。

- 比特存储库：链接 Bitbucket 存储库。

1. 在关联的 Bitbucket 存储库选项卡中，选择链接 Bitbucket 存储库
2. 从 Bitbucket 工作空间下拉列表中，选择包含要链接的存储库的 Bitbucket 工作空间。
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**
>
> 如果存储库的名称显示为灰色，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. 选择关联。

如果您不想再在中使用 Bitbucket 存储库 CodeCatalyst，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 存储库和 Jira 项目 CodeCatalyst。

- Jira Software：链接 Jira 项目。

1. 在 “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，选择包含您要关联的项目的 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. 选择关联。

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 存储库和 Jira 项目 CodeCatalyst。

您也可以在 C od e 中将您的 GitHub 或 Bitbucket 存储库链接到源存储库中的项目。有关更多信息，请参阅 链接来自关联第三方提供商的资源。

后续步骤

安装GitHub 存储库或 Bitbucket 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。您还可以使用蓝图生成的代码在关联的 GitHub 账户或 Bitbucket 工作空间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程 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。有关将拉取请求与 Jira 议题关联的更多信息，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**
>
> 要安装扩展程序，您必须使用在空间中具有空间管理员角色的帐户登录。

> **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 您可以搜索附加信息或根据类别筛选附加信息。
4. （可选）选择扩展程序的名称以查看有关扩展程序的更多详细信息，例如扩展程序将拥有的权限。
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。

安装扩展程序后，您将看到已安装扩展程序的详细信息页面。浏览选项卡，了解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对扩展程序进行进一步配置。

在空间中卸载扩展程序

您可以卸载以前安装在您的 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**
>
> 要卸载扩展程序，您必须使用在空间中具有空间管理员角色的帐户登录。

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. 执行以下任一操作，查看您的空间中已安装的扩展列表：

a. 选择 “设置”，然后选择 “已安装的扩展”。

b. 在顶部菜

单 

选择 “目录” 图标。

中

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” 来卸载扩展程序。

关联 GitHub 账户、Bitbucket 工作空间和 Jira 站点 CodeCatalyst

要使用 GitHub 或 Bitbucket 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 人际关系 和 通过人际关系访问 GitHub 资源。

将您的 GitHub 账户、Bitbucket 工作空间或 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. 执行以下任一操作，查看您的空间中已安装的扩展列表：

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 存储库或 Jira So ftware。
5. 根据您选择配置的第三方扩展，执行以下任一操作：

- 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 Save。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 工作空间。

  1. 在 Conn ected Bitbucket 工作空间选项卡中，选择 Conn ect Bitbucket 工作空间以转到 Bitbucket 的外部
  2. 使用您的 Bitbucket 凭据登录到您的 Bitbucket
  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”。

a. 如果您是首次授予工作区 CodeCatalyst 访问权限，请选择前往设置以启用 Bitbucket 的开发模式。

b. 在工作区的 “已安装的应用程序” 页面上，选中 “启用开发模式” 复选框以允许安装 CodeCatalyst，然后从中重新连接 CodeCatalyst。

一旦您允许安装 CodeCatalyst，它就可以访问您在 Bitbucket 工作区中的所有存储库。

> **Tip**
>
> 如果您之前已将 Bitbucket 工作空间连接到该空间，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。为要允许的工作空间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。

- Jira Software：连接 Jira 站点。

1. 在 Conn ected Jira 站点选项卡中，选择 Connect 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 Get started 进入最后一步。

3. 根据您的角色，执行以下任一操作：

   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，系统会自动将您定向到下一步。

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，请参阅谁可以安装应用程序？。

   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。

如果您不想再在中使用 GitHub 存储库、Bitbucket 存储库或 Jira 问题 CodeCatalyst，则可以断开第三方来源的连接。当 GitHub 账户或 Bitbucket 工作空间断开连接时，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 工作空间和 Jira 网站的连接 CodeCatalyst。

断开 GitHub 账户、Bitbucket 工作空间和 Jira 网站的连接 CodeCatalyst

如果您不想再在中使用 GitHub 存储库、Bitbucket 存储库或 Jira 问题 CodeCatalyst，则可以断开第三方源的连接。 GitHub 账户或 Bitbucket 工作空间断开连接后，存储库中的事件将无法启动工作流程运行，并且您将无法在开发环境中使用这些存储库。 CodeCatalyst 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 工作空间的连接，必须先取消所有关联的 Bitbucket 存储库与该工作空间的关联。
- 要断开 Jira 站点的连接，必须先取消所有关联的 Jira 项目与该账户的关联。

有关更多信息，请参阅 取消关联 GitHub 存储库、Bitbucket 存储库和 Jira 项目 CodeCatalyst。

断开 GitHub 项目、Bitbucket 工作空间或 Jira 站点的连接

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下任一操作，查看您的空间中已安装的扩展列表：

   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 存储库或 Jira So ftware。
5. 根据您选择配置的第三方扩展，执行以下任一操作：

   - 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 工作空间的连接。

     在 “已连接的 Bitbucket 工作空间” 选项卡中，选择要断开连接的 Bitbucket 工作空间，然后选择 “断开连接 Bitbucket 工作空间”。
   - Jira Softwar e：断开与 Jira 网站的连接。

     在 “已连接 Jira 站点” 选项卡中，选择要断开连接的 Jira 站点，然后选择 “断开连接 J ira 站点”。
6. 在 “断开连接” 对话框中，查看断开帐户连接的影响。
7. 在文本输入字段中输入 “断开连接”，然后选择 “断开连接”。

在中关联 GitHub 存储库、Bitbucket 存储库和 Jira 项目 CodeCatalyst

在使用 GitHub 或 Bitbucket 存储库或管理 Jira 项目之前，必须将该存储库或项目所属的第三方源与您的 CodeCatalyst 空间相关联。有关更多信息，请参阅 关联 GitHub 账户、Bitbucket 工作空间和 Jira 站点 CodeCatalyst。

您可以在工作流程中使用链接存储库 GitHub 或 Bitbucket 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。链接存储库还可以与开发环境一起使用，以创建、更新和删除链接存储库中的文件。您可以通过存储库 GitHub 或 Bitbucket 存储库扩展的详细信息页面或 CodeCatalyst 项目本身的 “代码” 中的 “源存储库” 视图将或 Bitbuck et 存储库链接到项目。GitHub

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

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 存储库和 Jira 项目 CodeCatalyst。

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。

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

- GitHub 或 Bitbucket 存储库只能链接到空间中的一个 CodeCatalyst 项目。
- 您不能在 CodeCatalyst 项目中使用空存储库、已存档存储库 GitHub 或 Bitbucket 存储库。
- 您不能链接与 CodeCatalyst 项目中仓库同名的 GitHub 或 Bitbucket 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 数据中心存储库不兼容。
- 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

链接来自关联第三方提供商的资源

从扩展详细信息页面将 GitHub 存储库、Bitbucket 存储库或 Jira CodeCatalyst 项目链接到项目

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 导航到您的 CodeCatalyst 空间。
3. 执行以下任一操作，查看空间空间中已安装的扩展列表：

   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 存储库或 Jira So ftware。
5. 根据您选择配置的第三方扩展，执行以下任一操作：

   - 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**
>
> 如果存储库的名称显示为灰色，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。您可以配置关联账户 CodeCatalyst 中可以使用哪些 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. 选择关联。

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 存储库和 Jira 项目 CodeCatalyst。

- 比特存储库：链接 Bitbucket 存储库。

1. 在关联的 Bitbucket 存储库选项卡中，选择链接 Bitbucket 存储库
2. 从 Bitbucket 工作空间下拉列表中，选择包含要链接的存储库的 Bitbucket 工作空间。
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**
>
> 如果存储库的名称显示为灰色，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. 选择关联。

如果您不想再在中使用 Bitbucket 存储库 CodeCatalyst，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 存储库和 Jira 项目 CodeCatalyst。

- Jira Software：链接 Jira 项目。

1. 在 “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，选择包含要关联的项目的 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. 选择关联。

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 取消关联 GitHub 存储库、Bitbucket 存储库和 Jira 项目 CodeCatalyst。

从 CodeCatalyst 项目的源存储库页面将 GitHub 或 Bitbucket 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。
6. 根据您选择链接的第三方存储库提供商，执行以下任一操作：

- 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。您可以配置关联账户 CodeCatalyst 中可以使用哪些 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。

- 比特存储库：链接 Bitbucket 存储库。

1. 从 Bitbucket 工作空间下拉菜单中，选择包含要链接的存储库的 Bitbucket 工作空间。
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。

7. 选择关联。

如果您不想再在中使用 GitHub 或 Bitbucket 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 存储库和 Jira 项目 CodeCatalyst。

将您的 GitHub 或 Bitbucket 存储库关联到您的 CodeCatalyst 项目后，您可以在 CodeCatalyst 工作流程和开发环境中使用它。您还可以将链接存储库与 Amazon Q Developer、蓝图等配合使用。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程 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。有关将拉取请求与 Jira 议题关联的更多信息，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

创建新 CodeCatalyst 项目时，您可以将 GitHub 或 Bitbucket 存储库链接到新 CodeCatalyst 项目。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 存储库和 Jira 项目 CodeCatalyst

如果您不想再使用 GitHub 存储库或 Bitbucket 存储库或在中管理 Jira 项目 CodeCatalyst，则可以取消该存储库或项目与您的项目的链接。 CodeCatalyst

取消关联 GitHub 或 Bitbucket 存储库不会删除存储库或对其进行任何更改。它不会删除存储在该链接存储库中的任何工作流程配置文件。但是，一旦您取消了 GitHub 或 Bitbucket 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。您可以从存储库 GitHub 或 Bitbucket 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” 视图中取消或 Bitbuck et 存储库与项目的链接。GitHub

取消关联 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 GitHub 仓库或 Bitbucket 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. 执行以下任一操作，查看您的空间中已安装的扩展列表：

   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 存储库或 Jira So ftware。
5. 根据您选择配置的第三方扩展，执行以下任一操作：

   - 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，选择要取消关联的 Bitbucket 存储库，然后选择取消关联 Bitbucket 存储库。

   - Jira Software：取消关联 Jira 项目。

     在 Jira 项目选项卡中，选择要取消关联的 Jira 项目，然后选择取消关联 Jira 项目。
6. 在 “取消链接” 对话框中，查看解除存储库链接的影响。
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 CodeCatalyst 项目中 GitHub 或 Bitbucket 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

链接 GitHub 或 Bitbucket 存储库后，您可以查看它们 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

- 在中搜索 Jira 问题 CodeCatalyst

在中查看第三方存储库 CodeCatalyst

您可以在项目的源存储库列表中 GitHub 或从存储库或 Bitbucket 存储库扩展详情页面查看链接GitHub 存储库或 Bitbucket 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链接存储库中的代码。

要查看链接的存储库 GitHub 或 Bitbucket 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。

从扩展详细信息页面查看链接存储库 GitHub 或 Bitbucket 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. 根据您要查看的第三方存储库，请执行以下任一操作：
   - 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

列表中显示了链接到您的 CodeCatalyst 项目的 GitHub 或 Bitbucket 存储库。选择 GitHub 或 Bitbucket 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 或 Bitbucket 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。

在关联的 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，在关联的 Jira 项目中搜索要链接到拉取请求的议题或 Jira 事务。

在第三方存储库事件发生后自动启动工作流程

您可以使用链接存储库 GitHub 或 Bitbucket 存储库作为工作流程的来源，在该工作流程中，对链接存储库 GitHub 或 Bitbucket 存储库中指定分支的更改会自动启动工作流程运行。

工作流程是一个自动化过程，它描述了如何构建、测试和部署您的代码，作为持续集成和持续交付 (CI/CD) 系统的一部分。工作流程定义了在工作流程运行期间要执行的一系列步骤或操作。工作流程还定义了导致工作流程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程，请使用蓝图创建一个项目。每个蓝图都部署了一个可以正常运行的工作流程，您可以对其进行查看、运行和试验。

当您将工作流程配置为使用链接存储库 GitHub 或 Bitbucket 存储库时，工作流程配置文件存储在该存储库 GitHub 或 Bitbucket 存储库中。工作流程配置是一个 YAML 文件，用于定义工作流程名称、触发器、资源、构件和操作。有关工作流程配置文件的更多信息，请参阅工作流程 YAML 定义。

工作流程配置文件必须位于您 GitHub 或 Bitbucket 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程。有关更多信息，请参阅工作流程入门和将工作流程连接到源存储库。

添加触发器以启动工作流程运行

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。要自动启动工作流程运行，请在工作流程配置文件的`Triggers`部分添加触发器。

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，每当将代码推送到源存储库中的任何分支时，该触发器就会启动工作流程运行。

```
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，每当针对源存储库中的任何分支创建拉取请求时，该触发器就会启动工作流程运行。

```
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程。

使用 GitHub 企业云限制 IP 访问

您可以通过设置规则或配置，根据 IP 地址限制对您 GitHub 或 Bitbucket 存储库的访问权限。您可以通过第三方提供商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商，请参阅以下内容之一：

- 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 工作空间以限制对特定 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

第三方扩展程序使用以下 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：

```
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
```

工作流程失败时阻止第三方拉取请求合并

将 GitHub 存储库链接到后 CodeCatalyst，您可以为拉取请求添加 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 CodeCatalyst 每个工作流程的运行状态也反映为 GitHub 或 Bitbucket 中提交状态的一部分。推送新提交时，新提交的新工作流程运行状态将反映到该新提交中 GitHub 。如果您再次运行工作流程进行提交，则新的工作流程运行状态将覆盖该提交和工作流程的先前状态。

您可以在中设置分支保护规则 GitHub ，以便在最新提交的工作流程运行状态为失败时阻止拉取请求合并。使用分支保护规则，最新提交的状态会影响合并拉取请求的能力 GitHub。有关更多信息，请参阅

“关于状态检查 GitHub的文档” 和 “关于受保护的分支”。要了解有关工作流程的更多信息，请参阅运行工作流和使用触发器自动启动工作流程。

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。关联 Jira 事务后，该议题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。拉取请求可以关联到一个或多个 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，选择包含要查看的更改的分支。
7. 从目标分支下拉菜单中，选择要合并已审核更改的分支。
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. 选择 Jira 事务-可选字段的关联事务，选择下拉列表，然后从关联的 Jira 项目中搜索要添加的 Jira 事务。
10. 选择要添加到拉取请求中的 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。摘要包括工作流程运行、关联的问题、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 中查看拉取请求时，关联的拉取请求和任何与拉取请求相关的工作流程事件的状态都将显示在 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 事件会同步，因此成功运行的工作流程也会发送到 Jira 议题。

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 “其他信息” 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）要查看最新版本，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。

在中搜索代码、问题、项目和用户 CodeCatalyst

使用中的搜索栏或专用的搜索结果窗口 CodeCatalyst 来搜索代码、问题、项目和用户 CodeCatalyst。

通过在搜索栏中输入姓名、描述和状态等查询，即可在空间和项目中查找资源。您也可以使用搜索查询语言来优化搜索查询。

主题

要搜寻

1. 在顶部导航栏的搜索栏中，输入搜索查询。
2. （可选）使用 CodeCatalyst搜索查询语言优化您的搜索查询。有关更多信息，请参阅 完善您的搜索查询。
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。
   - 要在您当前所在空间的所有项目中搜索资源，请选择此空间。
4. 通过执行以下任一操作，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，选择在项目名称中查看所有结果 | space-name 以查看所有搜索结果。
   - 按 Enter 键查看所有搜索结果。

> **Tip**
>
> 在拉取请求评论或描述中，或者在问题评论或描述中提及其他项目用户，方法是使用 @ 符号后面加上他们的显示名称或用户名。您还可以使用 @ 符号和议题或代码文件名称来链接到议题或代码文件等资源。

完善您的搜索查询

如果搜索后找不到要查找的内容，则可以使用专门 CodeCatalyst的查询语言来优化搜索。单个字段没有字符限制，但整个查询的字符限制为 1,024 个字符。

主题

按类型细化

要将搜索范围缩小到特定类型的信息，请在搜索*type:result-type*中加入，其中*###*#为code、issueproject、或。user

示例：

- type:code AND java— 在包含 “java” 的代码相关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 “错误” 的问题相关字段中显示问题结果。

  有关更多信息，请参阅 问题字段。
- type:user AND MaryMajor— 在包含 “MaryMajor” 的用户相关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 “数据馈送器” 的项目结果。

  有关更多信息，请参阅 项目字段。

按领域细化

要将搜索范围缩小到特定字段，请在搜索*field-name:query*中加入，其中 *field-name* 为titleusernameproject、description、、等，*query* 是您要搜索的文本。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 “错误” 的结果。
- `username:John`— 显示用户名包含 “John” 的结果。
- `project:DataFeeder`— 在项目 “DataFeeder” 中显示结果。查询不区分大小写。
- `description:overview`— 显示描述包含 “概述” 的结果。

使用布尔运算符进行优化

要指定搜索短语的限制，可以使用布尔运算符AND`OR`、和`NOT`。如果您列出多个短语，则`OR`默认使用这些短语进行 CodeCatalyst连接。您可以使用圆括号对搜索短语进行分组。

- `exception AND type:code`— 仅显示 “异常” 的代码结果。
- `path:README.md AND repo:ServerlessAPI`— 显示带有 “README.md” 的路径的结果，其中存储库名为 “ServerlessAPI”。
- `buildspec.yml AND (repo:ServerlessAPI OR ServerlessWebApp)`— 显示存储库为 “ServerlessAPI” 或 “” 的 “buildspec.yml” 的结果。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

要将搜索范围缩小到特定项目，请在搜索*`project:name AND query`*中加入，其中 *`name`* 是您要搜索的项目，*`##`*的是您要搜索的内容。

- `project:name AND query`— 显示路径中包含查询和项目名称的结果。

使用搜索时的注意事项

内容更新延迟 — 内容更新（例如名称更改或问题重新分配）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出现在搜索结果中。

转义特殊字符 —在搜索查询中需要特别考虑以下特殊字符:`+ - & & || ! ( ) { } [ ] ^ " ~ * ? : \`. 特殊字符不会影响查询，您必须移除它们或对其进行转义。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 [功能] 应为 功能或\ [功能\]。

缩小搜索范围-搜索不区分大小写。全小写搜索可防止您的查询在大小写变化时分开单词。例如，要仅查询MyService和MyService，请考虑进行查询myservice以避免仅包含my或的结果service。

默认情况下，搜索使用OR-wise连词连接单词和单词的一部分。例如，new function可以返回同时包含newfunction和的结果，也可以返回仅包含new或的结果function。为避免后者，请将多个单词与合并AND。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最新提交的代码结果。要在其他分支或提交上查找代码，可以考虑在本地克隆存储库，在开发环境中打开分支，或者在 CodeCatalyst 用户界面中查看分支和详细信息。更改默认分支会导致更新可通过搜索发现的文件。有关更多信息，请参阅 管理仓库的默认分支。

可搜索字段参考

CodeCatalyst 当您输入搜索查询时，会搜索以下字段。别名是可用于在高级查询语言中引用该字段的另一个名称。

代码字段

字段	别名	描述
分支名称	分支	代码文件所在分支的名称。
代码	不适用	有关代码内容的信息，以代码片段的形式表示，源代码中与搜索结果相匹配的部分。
CommitID	不适用	上次更新返回的代码文件的提交 ID。可能是中指定的分支名称末端的提交 ID，也可能不是。branchName
提交消息	不适用	上次更新代码文件的提交消息。可能是中指定的分支名称末端的提交消息，也可能不是。branchName 如果未提供提交消息，则此值将为空字符串。

字段	别名	描述
filePath	path	此代码文件的文件路径。
lastUpdatedBy	不适用	CodeCatalyst 上次更新代码文件的用户。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。
lastUpdatedBy我是	不适用	系统生成的上次更新代码文件的用户的唯一 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（采用协调世界时 (UTC) 时间戳）。
projectId	不适用	系统生成的项目唯一 ID。
projectName	项目名称，项目	显示包含已提交代码文件的源存储库的项目名称。
存储库 ID	RepoID	系统生成的源存储库的唯一 ID。
repositoryName	存储库、存储库	显示提交代码文件的源存储库的名称。

问题字段

字段	别名	描述
AssigneeID	AssigneeID	系统生成的分配给该问题的用户的唯一 ID。
受托人	受托人	分配给议题的用户的用户名。

字段	别名	描述
创建者	不适用	显示创建议题的用户的姓名。
createdById	不适用	系统生成的问题创建用户的唯一 ID。
CreatedTime	不适用	问题出现的时间（以协调世界时 (UTC) 时间戳为单位）。
description	不适用	问题描述。
已存档	archived	表示是否在已存档状态下创建议题的布尔值。
isBlock	blocked	表示问题是否被标记为已阻止的布尔值。
LabeLids	LabeLid	系统生成的问题标签的唯一 ID。
lastUpdatedBy	不适用	显示上次更新问题的用户姓名。
lastUpdatedBy我是	不适用	系统生成的上次更新问题的用户的唯一 ID。
lastUpdatedTime	不适用	问题上次更新的时间（以协调世界时 (UTC) 时间戳为单位）。
priority	不适用	问题的优先级（如果已分配）。
projectId	不适用	系统生成的项目唯一 ID。
projectName	项目名称，项目	可以在其中找到此问题的项目。

字段	别名	描述
ShortID	不适用	问题的缩短、自动递增的标识符。
status	不适用	问题的状态，指示问题是在待办事项中还是在船上专栏中。
StatusID	不适用	状态的系统标识符。
title	不适用	问题的标题。

项目字段

字段	别名	描述
description	不适用	项目描述。
lastUpdatedTime	不适用	项目元数据上次更新的时间（采用协调世界时 (UTC) 时间戳）。
projectName	项目	空间中项目的名称。
项目路径	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。

用户字段

字段	别名	描述
displayName	不适用	中用于用户的名称 CodeCatalyst。显示名称不是唯一的。
email	不适用	用户的电子邮件地址。

字段	别名	描述
lastUpdatedTime	不适用	上次更新用户元数据的时间（采用协调世界时 (UTC) 时间戳）。
userName	username	用户在注册时选择的用户名 CodeCatalyst。与显示名称不同，用户名无法更改。

对亚马逊进行故障排除 CodeCatalyst

以下信息可以帮助您解决中的常见问题 CodeCatalyst。您还可以使用 Amazon CodeCatalyst 健康报告来确定是否存在可能影响您的体验的服务问题。

主题

对一般访问问题进行故障排除

我忘记密码了

问题：我忘记了用于AWS生成器 ID 和 Amazon 的密码 CodeCatalyst。

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 Continue（继续）。
2. 选择“忘记密码？”
3. 我们将向您发送一封包含更改密码链接的电子邮件。如果您在收件箱中没有看到该电子邮件，请检查您的垃圾邮件文件夹。

Amazon 的部分或全部商品 CodeCatalyst 不可用

问题：我导航到控制台或点击了 CodeCatalyst 控制台的链接，但我看到一个错误。

可能的修复方法：出现此问题的最常见原因是，您要么点击了未被邀请访问的项目或空间的链接，要么该服务存在一般可用性问题。查看 He alth 报告，查看该服务是否存在任何已知问题。如果不是，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。如果您尚未被邀请参与任何项目或空间，则可以注册并创建自己的空间和项目。

我无法在中创建项目 CodeCatalyst

问题：我想创建一个项目，但是 “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不具有 Space 管理员角色的 AWS Builder ID 登录控制台。您必须具有此角色才能在空间中创建项目。

如果您确实具有此角色并且该按钮显示为不可用，则该服务可能存在暂时性问题。请刷新浏览器，然后重试。

对支持问题进行故障排除

我在访问 AWS Support Amazon 时出现错误 CodeCatalyst

问题：当我选择 “f AWS Support or Amazon CodeCatalyst ” 选项时，我收到以下错误消息：

`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：将所需的角色添加到空间AWS 账户的结算账号中。指定为空间账单账户的账户使用AWSRoleForCodeCatalystSupport角色和AmazonCodeCatalystSupportAccess托管策略。有关更多信息，请参阅为您的账户和空间创建AWSRoleForCodeCatalystSupport角色：

> **Note**
>
> AWSBuilder ID 只能获得对他们进行身份验证的别名的支持，并且只能获得基于权限的资源的支持 CodeCatalyst。该空间中的所有用户均可获得账户和账单支持。但是，构建者只能获得他们有权访问的资源和信息的支持 CodeCatalyst。

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：需要向空间账单账户中添加商业支持或企业支持计划，以便空间中的用户可以创建技术支持案例。请您的空间管理员向您的空间账单账户添加AWS Support套餐，或访问 https://repost.aws/ 向AWS社区提问。

我的支持案例账户已不再与我的空间关联 CodeCatalyst

问题：我的支持案例账户不再与我的空间相关联 CodeCatalyst。

修复：如果具有空间管理员角色的用户切换空间账单账户，则会断开AWS Support计划和所有相关案例与空间的连接。在 Amazon 中，与旧空间账单账户相关的AWS Support案例将不再可见 CodeCatalyst。AWS Support该账单账户的根用户可以从中查看和解决旧案例，还可以设置 IAM 权限以AWS Support供其他用户查看和解决旧案例。AWS Management Console您将无法继续通过旧空间账单账户获得技术支持AWS Management Console，但是在您的AWS Support计划取消之前，您可以获得其他服务的技术支持。 CodeCatalyst

有关更多信息，请参阅《AWS Support用户指南》中的更新、解决和重新审理您的问题。

我无法在 Amazon AWS 服务 中为另一位客户提交AWS Support支持案例 CodeCatalyst

问题：我无法在 for AWS 服务 中AWS Support为其他人打开支持案例 CodeCatalyst。

可能的修复方法：您只能从中AWS Support打开 CodeCatalyst 支持案例 CodeCatalyst。如果您需要为从 CodeCatalyst 其他服务AWS、Amazon 或其他第三方服务部署的服务或资源提供支持，则需要通过 AWS Management Console或第三方服务支持渠道创建案例。有关更多信息，请参阅《AWS Support用户指南》中的创建支持案例和案例管理。

Amazon 的部分或全部商品 CodeCatalyst 不可用

问题：我导航到控制台或点击了 CodeCatalyst 控制台的链接，但我看到一个错误。

可能的修复方法：出现此问题的最常见原因是，您要么点击了未被邀请访问的项目或空间的链接，要么该服务存在一般可用性问题。查看 He alth 报告，查看该服务是否存在任何已知问题。如果不是，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。如果您尚未被邀请参与任何项目或空间，则可以注册并创建自己的空间和项目。

我无法在中创建项目 CodeCatalyst

问题：我想创建一个项目，但是“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不具有 Space 管理员角色的 AWS Builder ID 登录控制台。您必须具有此角色才能在空间中创建项目。

如果您确实具有此角色并且该按钮显示为不可用，则该服务可能存在暂时性问题。请刷新浏览器，然后重试。

我想通过以下方式提交反馈 CodeCatalyst

问题：我在中发现了一个错误 CodeCatalyst ，我想提交反馈。

可能的修复方法：您可以直接在中提交反馈 CodeCatalyst。

1. 打开 CodeCatalyst 控制台，网址为 https://codecatalyst.aws/。
2. 在导航窗格中，选择“提供反馈”。
3. 从下拉菜单中选择反馈类型，然后输入您的反馈。

源存储库问题疑难解答

以下信息可以帮助您解决中源存储库的常见问题 CodeCatalyst。

主题

我的空间已达到最大存储空间并看到警告或错误

问题：我想向中的一个或多个源代码库提交代码 CodeCatalyst，但我看到了错误。在控制台中，我在源存储库页面上看到一条消息，说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，或者将计费等级更改为具有更多存储空间的计费等级。

- 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空间，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，您可以将账单级别更改为具有更多存储空间的计费等级。有关更多信息，请参阅《Amazon CodeCatalyst 管理员指南》中的更改 CodeCatalyst 账单等级。

我在尝试克隆或推送到 Amazon CodeCatalyst 源存储库时收到错误消息

问题：当我尝试将源存储库克隆到本地计算机或集成开发环境 (IDE) 时，出现权限错误。

可能的修复方法：您的 AWS 建筑商 ID 可能没有个人访问令牌 (PAT)，可能没有使用 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存储库访问权限。
- 请确保您已接受包含源存储库的项目的邀请，并且您仍然是该项目的成员。如果您不是源存储库的活跃成员，则无法克隆该项目。登录控制台并尝试导航到要克隆源存储库的空间和项目。如果您在该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，或者您尚未接受该项目的邀请。有关更多信息，请参阅 接受邀请并创建您的 AWS 建筑商 ID。
- 确保您的克隆命令格式正确，并包含您的 AWS 生成器 ID。例如：

```
https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

- 使用 AWS CLI 来确保您的 PAT 与您的 AWS 建筑商 ID 相关联，并且未过期。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存储库访问权限。
- 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消息

问题：当我尝试推送到源存储库时，我收到权限错误。

可能的修复方法：在项目中，你可能没有一个角色允许你提交代码变更并将其推送到项目中。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。

如果您的角色允许提交和推送更改，则您尝试提交更改的分支可能已为其配置了分支规则，该规则禁止您将代码更改推送到该分支。尝试创建一个分支并将您的代码推送到该分支。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代码库

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代码库。

可能的修复：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，则可以在中为该项目创建源存储库 CodeCatalyst。如果拥有 Space 管理员角色的用户安装了GitHub 存储库并将其关联到 GitHub 帐户，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。

我的源代码库是全新的，但包含一个提交

问题：我刚刚创建了一个源存储库。它应该是空的，但里面有一个提交、一个分支和一个`README.md`文件。

可能的修复：这是预期的行为。中的所有源存储库都 CodeCatalyst 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 客户端中创建其他分支。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我的默认分支

问题：我的源存储库有一个名为的默认分支`main`，但我想要一个不同的分支作为我的默认分支。

可能的修复方法：您无法在中更改或删除源存储库中的默认分支 CodeCatalyst。您可以创建其他分支并在工作流程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。

我收到了关于拉取请求中活动的电子邮件

问题：我没有注册或配置有关拉取请求活动的电子邮件通知，但无论如何我都会收到。

可能的修复方法：自动发送有关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，但我失去了令牌的价值，也无法在 CodeCatalyst 控制台中找到它。

可能的修复方法：解决此问题的最快方法是创建另一个 PAT，并将您的凭据管理器或 IDE 配置为使用此新 PAT。我们仅在您创建 PAT 时显示其值。如果您丢失了此值，则无法对其进行检索。有关更多信息，请参阅 使用个人访问令牌向用户授予存储库访问权限。

拉取请求不会显示我期望的更改

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：

- 您可能正在查看旧版本之间的更改，或者可能没有查看最新的更改。刷新浏览器，确保选择了要查看的版本之间的比较。
- 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础与您选择的任何版本之间的区别。创建拉取请求时，显示的区别是源分支的尖端和目标分支的尖端之间的区别。创建拉取请求后，显示的区别在于修订版与其合并基础之间的区别。合并基础是创建修订版时目标分支尖端的提交。合并基础可以在不同版本之间发生变化。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。

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：

- 您的拉取请求的所有必需审阅者都必须批准拉取请求，然后才能将其合并。查看姓名旁边带有时钟图标的审阅者所需审阅者名单。时钟图标表示审阅者尚未批准拉取请求。

> **Note**
>
> 如果在批准拉取请求之前已将所需的审阅者从您的项目中移除，则您无法合并拉取请求。关闭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，然后使用 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。

对项目和蓝图进行故障排除

本节可以帮助您解决在 Amazon CodeCatalyst 中处理项目和蓝图时可能遇到的一些常见问题。

带有AWS Fargate蓝图的 Java API 缺少 ap ache-maven-3.8.6 的依赖关系

问题：对于通过 Java API 创建的带有AWS Fargate蓝图的项目，工作流程失败并显示缺少apache-maven-3.8.6依赖项的错误。工作流程失败，输出类似于以下示例：

```
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：使用以下步骤更新蓝图 Dockerfile。

1. 在搜索栏中，输入apache-maven-3.8.6以在使用带有蓝图的 Java API 创建的项目中找到 dockerfile。AWS Fargate
2. 更新 Dockerfile (/static-assets/app/Dockerfile) 以maven:3.9.0-amazoncorretto-11用作基础镜像，并移除对软件包的apache-maven-3.8.6依赖。

3. （推荐）我们还建议将 Maven 堆大小更新为 6 GB。

以下是 Dockerfile 的示例。

```
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工作流程时，工作流程无法运行，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，以允许工作流程成功运行。

> **Note**
>
> 这些步骤适用于中的以下工作流程 CodeCatalyst：

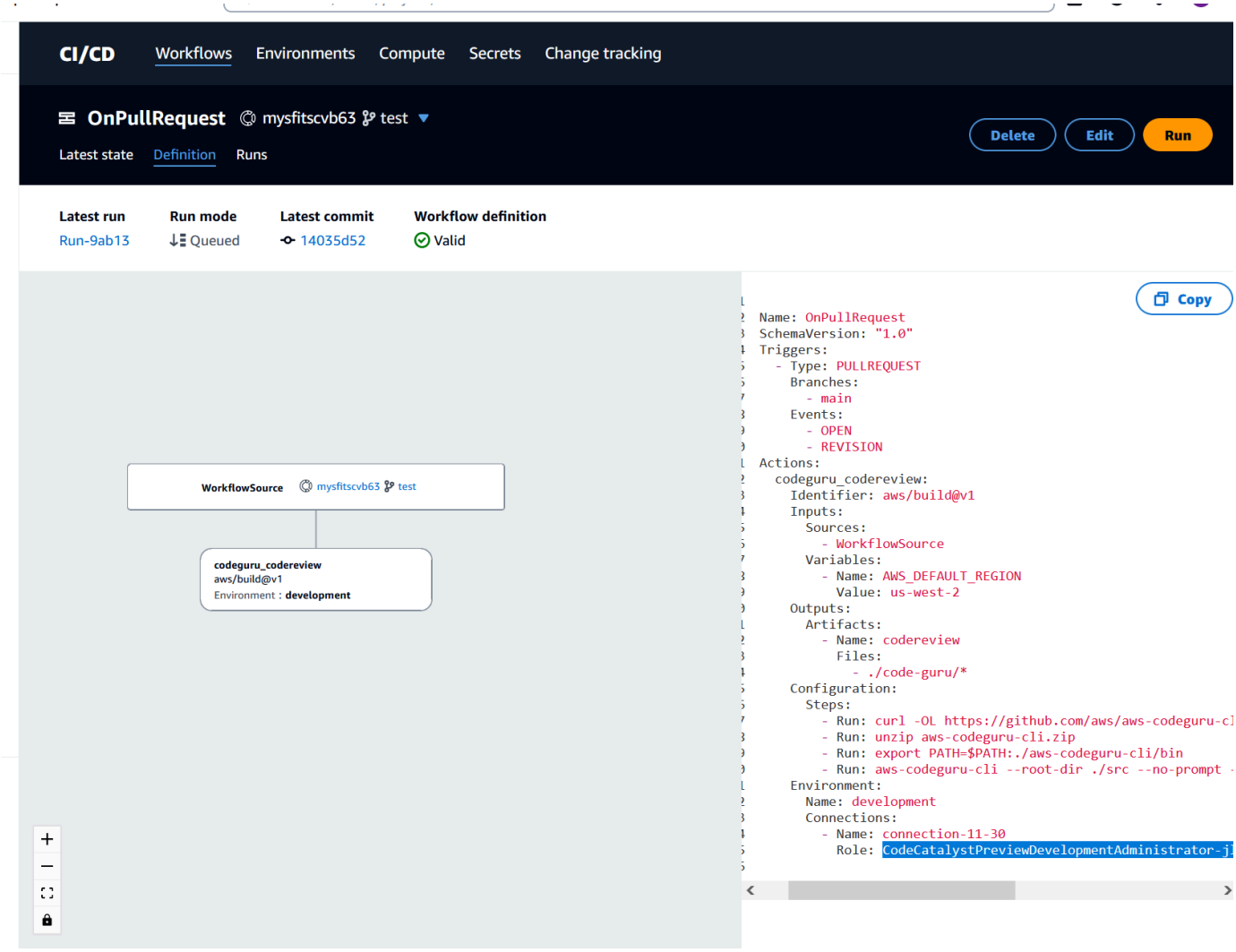
- 为使用现代三层 Web 应用程序蓝图创建的项目提供OnPullRequest的工作流程。CodeCatalyst
- 通过访问 Amazon CodeGuru 或 Amazon CodeGuru Reviewer 的操作将工作流程添加到项目中 CodeCatalyst 。

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程存储在源存储库的 /.codetalyst/workflows 目录中。除非您要向现有工作流程添加新的角色 ID，否则无需修改工作流程 YAML。有关 YAML 模板元素和格式的信息，请参阅工作流程 YAML 定义。

以下是编辑角色策略和验证工作流程 YAML 时要遵循的高级步骤。

在工作流程 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 “工作流程”。
4. 选择标题为的工作流程OnPullRequest。选择 Definition (定义) 选项卡。
5. 在工作流程 YAML 中，在 codeguru_codereview 操作下的`Role:`字段中，记下角色名称。这是您要在 IAM 中修改的策略中的角色。以下示例显示了角色名称。



6. 请执行下列操作之一：

- （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CodeCatalystWorkflowDevelopmentRole-*spaceName*名称将附加唯一标识符。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，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 Management Console，然后通过以下网址打开 IAM 控制台：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。添加这些权限后，权限策略应类似于以下内容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "apigateway:*",
                "ecr:*",
                "ecs:*",
                "ssm:*",
                "codedeploy:*",
                "s3:*",
                "iam:DeleteRole",
                "iam:UpdateRole",
                "iam:Get*",
                "iam:TagRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:CreatePolicyVersion",
                "iam:DeletePolicyVersion",
                "iam:PutRolePermissionsBoundary",
                "iam:DeleteRolePermissionsBoundary",
                "sts:AssumeRole",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule",
                "cloudwatch:DescribeAlarms",
                "sns:Publish",
```

```
                "sns:ListTopics",
                "codeguru-reviewer:*",
                "codeguru:*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

9. 修改政策后，返回 CodeCatalyst 并重新开始运行工作流程。

还在想解决你的问题吗？

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。

工作流程问题疑难解答

要解决与 Amazon 工作流程相关的问题，请参阅以下章节 CodeCatalyst。有关工作流的更多信息，请参阅使用中的工作流程构建、测试和部署 CodeCatalyst。

主题

- 如何修复 “工作流程处于非活动状态” 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 安装” 错误？
- 为什么多个工作流程同名？
- 我能否将我的工作流程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程添加操作？
- 为什么我的工作流程成功验证但在运行时失败？
- 自动发现不会发现任何与我的操作相关的报告

- 配置成功标准后，我对自动发现的报告执行操作失败
- 自动发现会生成我不想要的报告
- 自动发现为单个测试框架生成许多小报告
- CI/CD 下列出的工作流程与源存储库中的工作流程不匹配
- 我无法创建或更新工作流程

如何修复“工作流程处于非活动状态”消息？

问题：在 CodeCatalyst 控制台的 C I/CD “工作流程”下，您的工作流程将显示以下消息：

```
Workflow is inactive.
```

此消息表示工作流程定义文件包含不适用于您当前所在分支的触发器。例如，您的工作流程定义文件可能包含引用您的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也不会启动工作流程在中运行main，因此在该分支 CodeCatalyst 上停用工作流程并将其标记为Inactive。main

可能的修复措施：

如果要在功能分支上启动工作流程，可以执行以下操作：

- 在您的功能分支中，在工作流程定义文件中，从该Triggers部分中移除该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（包括您的功能分支）时激活。如果激活了触发器，则 CodeCatalyst 将使用工作流程定义文件和你要推送到的任何分支中的源文件启动工作流程运行。
- 在您的功能分支中，在工作流程定义文件中，删除该Triggers部分并手动运行工作流程。
- 在您的功能分支中，在工作流程定义文件中，更改该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。

有关编辑工作流程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程。

如何修复 “工作流定义有 *n* #错误” 错误？

问题：您会看到以下任何错误消息：

错误 1：

在 C I/CD 的 “工作流程” 页面中，在您的工作流程名称下，您会看到：

```
Workflow definition has n errors
```

错误 2：

编辑工作流程时，选择 “验证” 按钮， CodeCatalyst 控制台顶部会显示以下消息：

```
The workflow definition has errors. Fix the errors and choose Validate to
verify your changes.
```

错误 3：

导航到工作流程的详细信息页面后，您会在工作流程定义字段中看到以下错误：

```
n errors
```

可能的修复措施：

- 选择 CI/CD，选择 Workflows，然后选择出现错误的工作流程的名称。在靠近顶部的 “工作流程定义” 字段中，选择错误链接。有关错误的详细信息显示在页面底部。按照错误中的疑难解答提示修复问题。
- 确保工作流程定义文件是 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程定义文件中，请参阅或查阅您的操作文档，该文档已链接至 from 向 CodeCatalyst 工作流程添加操作。工作流程 YAML 定义
- 确保正确转义星号 (*) 和其他特殊字符。要避开它们，请添加单引号或双引号。例如：

```
Outputs:
```

```
Artifacts:
  - Name: myartifact
    Files:
      - "**/*"
```

有关工作流程定义文件中特殊字符的更多信息，请参见语法指南和惯例。

- 确保工作流程定义文件中的 YAML 属性使用正确的大小写。有关大小写规则的更多信息，请参阅语法指南和惯例。要确定每个属性的正确大小写，请参阅或查阅您的操作文档，该文档链接至 from 向 CodeCatalyst 工作流程添加操作。工作流程 YAML 定义
- 确保该SchemaVersion属性存在并在工作流程定义文件中设置为正确的版本。有关更多信息，请参阅 SchemaVersion。
- 确保工作流程定义文件中的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，或者参阅触发器参考文档，网址为Triggers。
- 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 将操作配置为依赖于其他操作。
- 确保工作流程定义文件中的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，或者查阅操作的文档，该文档的链接来自于向 CodeCatalyst 工作流程添加操作。
- 确保所有输入工件都有相应的输出伪影。有关更多信息，请参阅 定义输出对象。
- 确保导出在一个操作中定义的变量，以便可以在其他操作中使用它们。有关更多信息，请参阅 导出变量以便其他操作可以使用它。

如何修复 “找不到凭证” 和 “ExpiredToken” 错误？

问题：在完成操作时教程：将应用程序部署到 Amazon EKS，你会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
```

可能的修复措施：

这些错误表明您用于访问 AWS 服务的凭证已过期。在这种情况下，请不要运行该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. 请确保您拥有正在使用的用户的 AWS 访问门户 URL、用户名和密码，完成了 Amazon EKS 教程 (codecatalyst-eks-user)。完成本教程后，您应该已经配置第 1 步：设置开发机器了这些项目。

   > **Note**
   >
   > 如果您没有这些信息，请前往 IAM Identity Center 的codecatalyst-eks-user详细信息页面，选择重置密码，生成一次性密码 [...] ，然后再次重置密码以在屏幕上显示信息。

2. 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。

     Or

   - 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 AWS 账户 向其分配codecatalyst-eks-user用户和权限集的名称。
5. 在权限集名称 (codecatalyst-eks-permission-set) 旁边，选择命令行或编程访问权限。
6. 复制页面中间的命令。它们看起来类似于以下内容：

   ```
   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符中，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 CLIeksctl、和kubectl命令现在应该可以正常工作了。

如何修复 “无法连接到服务器” 错误？

问题：在学习中描述的教程时教程：将应用程序部署到 Amazon EKS，你会在开发计算机的终端窗口中看到一条类似于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`

可能的修复措施：

此错误通常表示该kubectl实用程序用于连接您的 Amazon EKS 集群的证书已过期。要解决此问题，请在终端提示符下输入以下命令来刷新凭据：

```
aws eks update-kubeconfig --name codecatalyst-eks-cluster --region us-west-2
```

其中：

- *codecatalyst-eks-cluster*将替换为您的 Amazon EKS 集群的名称。
- *us-west-2* 被替换为部署集群 AWS 的区域。

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，但在工作流程的可视化编辑器CodeDeploy AppSpec中看不到诸如此类的 CodeDeploy 字段。之所以出现此问题，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。

可能的修复措施：

- 在 “部署到亚马逊 ECS” 操作的 “配置” 选项卡上选择其他 Amazon ECS 服务。有关更多信息，请参阅 使用工作流程将应用程序部署到 Amazon 弹性容器服务 (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 弹性容器服务开发人员指南 CodeDeploy中的蓝/绿部署。

如何修复 IAM 功能错误？

问题：您正在使用部署 AWS CloudFormation 堆栈操作，并且可以在部署 AWS CloudFormation 堆栈操作的日志##[error] `requires capabilities: [capability-name]`中看到。

可能的修复方法：完成以下步骤，将该功能添加到工作流程定义文件中。有关 IAM 功能的更多信息，请参阅 IAM 用户指南中的在 AWS CloudFormation 模板中确认 IAM 资源。

Visual

使用可视化编辑器添加 IAM 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 “视觉”。
7. 在工作流程图中，选择您的部署 AWS 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，选中错误消息中提及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” 以在提交之前验证工作流程的 YAML 代码。
12. 选择 “提交”，输入提交消息，然后再次选择 “提交”。
13. 如果新的工作流程运行未自动启动，请手动运行工作流程以查看更改是否修复了错误。有关手动运行工作流程的更多信息，请参阅启动工作流程手动运行。

YAML

使用 YAML 编辑器添加 IAM 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 C I/CD，然后选择工作流程。
4. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。
5. 选择编辑。
6. 选择 YAML。
7. 在 “部署 AWS 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。使用逗号和空格列出多项功能。有关更多信息，请参阅中对capabilities属性的描述“部署 AWS CloudFormation 堆栈” 操作 YAML 定义。

8. （可选）选择 “验证” 以在提交之前验证工作流程的 YAML 代码。
9. 选择 “提交”，输入提交消息，然后再次选择 “提交”。
10. 如果新的工作流程运行未自动启动，请手动运行工作流程以查看更改是否修复了错误。有关手动运行工作流程的更多信息，请参阅启动工作流程手动运行。

如何修复 “npm 安装” 错误？

问题：您的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他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。你将在cdk.json文件中引用这些秘密，而不是提供等效的明文信息。要创建机密，请执行以下操作：
   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 C I/CD，然后选择密钥。
   d. 使用以下属性创建两个密钥：

第一个秘密	第二个秘密
名称：npmUsername	名称：npmAuthToken

第一个秘密	第二个秘密
值：*npm-usern* ame，其中 *npm-usern am* e 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.	值:*npm-auth-token* ，其中*npm-auth-token* 是用于向您的私有 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 文档中的关于访问令牌。 （可选）描述：The access token used to authenticate to the private npm registry.

有关密钥的更多信息，请参阅在工作流程中配置和使用密钥。

2. 将密钥作为环境变量添加到您的 AWS CDK 操作中。该操作将在运行时用实数值替换变量。要添加秘密，请执行以下操作：

   a. 在导航窗格中，选择 C I/CD，然后选择工作流程。

   b. 选择工作流程的名称。您可以按定义工作流程的源存储库或分支名称进行筛选，也可以按工作流程名称进行筛选。

   c. 选择编辑。

   d. 选择 “视觉”。

   e. 在工作流程图中，选择您的 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 值：${Secrets.npmUsername}	名称：NPMTOKEN 值：${Secrets.npmAuthToken}

现在，你有两个包含对机密的引用的变量。

您的工作流程定义文件 YAML 代码应类似于以下内容：

> **Note**
>
> 以下代码示例来自AWS CDK 引导操作；AWS CDK 部署操作将与之类似。

```
Name: CDK_Bootstrap_Action
SchemaVersion: 1.0
Actions:
  CDKBootstrapAction:
    Identifier: aws/cdk-bootstrap@v1
    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。

更新你的 cdk.json 文件

1. 切换到 AWS CDK 项目的根目录，然后打开该cdk.json文件。
2. 找到该"app":属性，然后将其更改为包含以####显示的代码：

> **Note**
>
> 以下示例代码来自一个 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，替换：

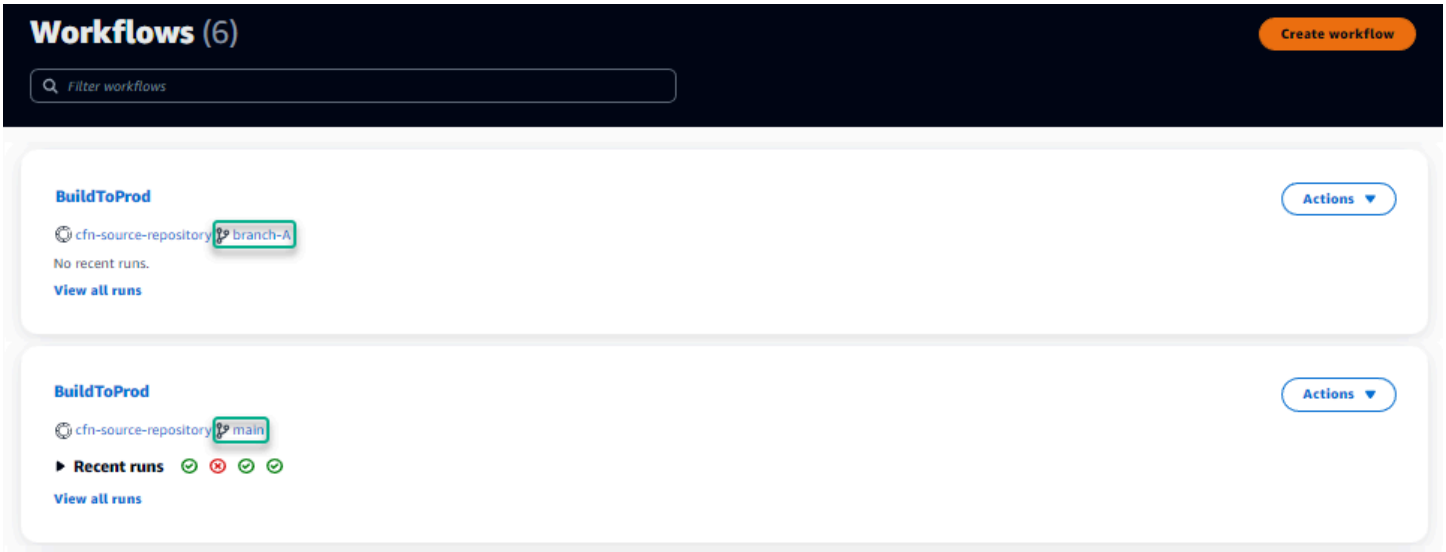
   - y@@ *our-registry/Folder/CDK-package/################*的路径。 AWS CDK
   - *hello-cdk.ts|.js*，上面写着你的入口点文件的名字。这可能是 `.ts` (TypeScript) 或 `.js` (JavaScript) 文件，具体取决于你使用的语言。

   > **Note**
   >
   > 该操作将使用你在 Secrets 中指定的 ***npm ########## NPM USER*** 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，请参阅启动工作流程手动运行。

为什么多个工作流程同名？

工作流程存储在每个存储库的每个分支上。如果两个不同的工作流程存在于不同的分支中，则它们可以具有相同的名称。在 “工作流程” 页面中，您可以通过查看分支名称来区分同名工作流程。有关更多信息，请参阅 使用 Amazon 中的分支来整理源代码 CodeCatalyst。



我能否将我的工作流程定义文件存储在另一个文件夹中？

不，您必须将所有工作流程定义文件存储在.codecatalyst/workflows文件夹中。如果您使用的是包含多个逻辑项目的 mono 存储库，请将所有工作流程定义文件放在该.codecatalyst/workflows文件夹中，然后使用 Trigger 中的FilesChanged属性在指定的项目路径上触发工作流程。有关更多信息，请参阅 使用触发器自动启动工作流程。

如何按顺序向我的工作流程添加操作？

默认情况下，当你向工作流程中添加操作时，它将没有依赖关系，将与其他操作并行运行。

如果要按顺序排列动作，则可以通过设置DependsOn字段来设置对另一个动作的依赖关系。您也可以将操作配置为使用作为其他操作输出的构件或变量。有关更多信息，请参阅 将操作配置为依赖于其他操作。

为什么我的工作流程成功验证但在运行时失败？

如果您使用Validate按钮验证了工作流程，但无论如何您的工作流程都失败了，则可能是因为验证器存在限制。

在提交过程中，在工作流程配置中引用密钥、环境或队列等 CodeCatalyst 资源时出现的任何错误都不会被记录下来。如果使用了任何无效的引用，则只有在运行工作流程时才会发现错误。同样，如果操作

配置中存在任何错误，例如缺少必填字段或操作属性中有错别字，则只有在工作流程运行时才会识别出来。有关更多信息，请参阅 创建工作流。

自动发现不会发现任何与我的操作相关的报告

问题：我为运行测试的操作配置了自动发现，但未发现任何报告。 CodeCatalyst

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一种可以 CodeCatalyst 理解的格式生成输出。例如，如果您想pytest CodeCatalyst 允许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，在配置为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。
- 确保将其配置IncludePaths为包括整个文件系统 (**/*)，除非您故意排除某些文件夹。同样，请确保ExcludePaths不要排除您期望报告所在的目录。
- 如果您手动将报告配置为使用特定的输出文件，则该报告将被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：

- ExcludePaths修改IncludePaths或排除您不感兴趣的报告。
- 更新成功标准以允许所有报告通过。例如，如果发现两个报告，其中一个的线路覆盖率为 50%，另一个报告为 70%，则将最小线覆盖率调整为 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。这允许您为该特定报告配置不同的成功标准。有关更多信息，请参阅 为报告配置成功标准。

自动发现会生成我不想要的报告

问题：当我启用自动发现功能时，它会生成我不想要的报告。例如，为存储在我的应用程序依赖项中的文件 CodeCatalyst 生成代码覆盖率报告node_modules。

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，我注意到自动发现会生成大量报告。例如，在使用Maven Surefire 插件时，自动发现会为每个测试类生成不同的报告。

可能的修复：您的框架可能能够将输出聚合到单个文件中。例如，如果您使用的是 Maven Surefire 插件，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 下列出的工作流程与源存储库中的工作流程不匹配

问题：C I/CD、Workflows 页面上显示的工作流程与源存储库中~/.codecatalyst/workflows/文件夹中的工作流程不匹配。您可能会看到以下不匹配项：

- 工作流程显示在 “工作流程” 页面上，但源存储库中不存在相应的工作流程定义文件。
- 您的源存储库中存在工作流程定义文件，但相应的工作流程未显示在 “工作流程” 页面上。
- 源存储库和工作流页面中都存在工作流程，但两者不同。

如果工作流程页面没有时间刷新，或者超过了工作流程配额，则可能会出现此问题。

可能的修复措施：

- 等待。提交源代码后，通常需要等待两三秒钟才能在 “工作流程” 页面上看到更改。
- 如果您已超过工作流程配额，请执行以下任一操作：

> **Note**
>
> 要确定是否已超过工作流程配额，请根据源存储库或工作流程页面上的工作流程查看工作流程配额并交叉检查记录的配额。没有错误消息表明已超出配额，因此您必须自己进行调查。

- 如果您已超过每个空间的最大工作流程数量配额，请删除一些工作流程，然后对工作流程定义文件执行测试提交。测试提交的一个例子可能是向文件添加一个空格。
- 如果您已超过最大工作流程定义文件大小配额，请更改工作流程定义文件以缩短其长度。
- 如果您已超过在单个源事件中处理的最大工作流程文件数量配额，请执行多次测试提交。修改的工作流程数量少于每次提交的最大数量。
- 通过开启付费等级计费来增加工作流程配额。有关更多信息，请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

我无法创建或更新工作流程

问题：我想创建或更新工作流程，但是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。工作流程的 YAML 文件存储在存储库中。有关更多信息，请参阅 工作流程定义文件。Space 管理员角色、项目管理员角色和参与者角色均有权提交代码并将其推送到项目中的仓库。

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程 YAML 的更改，则可能为该分支配置了分支规则，该规则禁止具有该角色的用户将代码推送到该特定分支。尝试在不同的分支中创建工作流程，或者将您的更改提交到其他分支。有关更多信息，请参阅 使用分支规则管理分支允许的操作。

疑难解答中搜索的问题 CodeCatalyst

要解决与搜索有关的问题，请参阅以下章节 CodeCatalyst。有关工作流的更多信息，请参阅在中搜索代码、问题、项目和用户 CodeCatalyst。

主题

- 我在我的项目中找不到用户
- 我在项目或空间中看不到我要找的东西
- 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

我在我的项目中找不到用户

问题：当我尝试查看用户的详细信息时，我在项目中看不到他们的信息。

可能的修复：搜索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，请切换到 “此空间” QuickSearch，或者移除您可能使用高级查询语言指定的任何项目筛选器。

我在项目或空间中看不到我要找的东西

问题：当我尝试搜索特定信息时，结果不会出现。

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。

对于最近未更新的资源，您可能需要调整搜索范围。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。有关完善查询的更多信息，请参阅完善您的搜索查询。

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会发生变化，因此目前尚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相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。

浏览结果时，您可能会看到以下消息：“测试” 没有结果。如果您无法访问其余结果，则会收到消息。

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，无论是通过编程方式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，请在几秒钟后重试。

解决与您的空间关联的账户相关的问题

在中 CodeCatalyst，您可以AWS 账户向空间添加以授予资源权限和计费。以下信息可以帮助您解决中关联账户的常见问题 CodeCatalyst。

主题

- 我的AWS 账户连接请求收到一个无效的令牌错误
- 我的 Amazon CodeCatalyst 项目工作流程失败，配置的账户、环境或 IAM 角色出现错误
- 我需要关联的账户、角色和环境才能创建项目

- 我无法访问中的 Amazon CodeCatalyst Spaces 页面 AWS Management Console
- 我想要一个不同的账号作为我的结算账号

我的AWS 账户连接请求收到一个无效的令牌错误

问题：使用连接令牌创建连接请求时，该页面不接受该令牌，并显示一条错误消息，指出该令牌无效。

可能的修复方法：请务必提供要添加到空间的账户 ID。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才能添加帐户。

当您选择验证帐户时，将在中打开一个新的浏览器窗口AWS Management Console。需要使用相同的帐户才能在控制台端登录。验证以下内容后再试一次：

- 您已AWS Management Console使用要添加到空间AWS 账户的相同信息登录。
- 您已在选择美国西部（俄勒冈）AWS 区域(`us-west-2`) 的情况下登录。AWS Management Console
- 如果您是从账单页面进入的，并且想要将其添加为空间的指定结算账号，请确保该账户还不是其他空间的结算账号。AWS 账户

我的 Amazon CodeCatalyst 项目工作流程失败，配置的账户、环境或 IAM 角色出现错误

问题：当工作流程运行但未找到与您的空间关联的已配置账户或 IAM 角色时，您必须在工作流程 YAML 中手动填写角色、连接和环境字段。查看失败的工作流程操作，并注意错误消息是否如下所示：

- 该角色不可用于与环境关联的连接。
- 操作未成功。状态：失败；为账户连接或环境提供的值无效。验证连接是否与您的空间相关联以及环境是否与您的项目关联。
- 操作未成功。状态：失败；为 IAM 角色提供的值无效。验证名称是否存在，IAM 角色已添加到您的账户连接中，并且该连接已与您的 Amazon CodeCatalyst 空间关联

可能的修复方法：确保工作流程 YAML 字段的 “环境”、“连接” 和 “角色” 值准确无误。需要环境 CodeCatalyst 的工作流程操作是运行AWS资源或生成AWS资源堆栈的生成或部署操作。

选择失败的工作流程操作块，然后选择 Visual。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程。使用以下步骤编辑您的工作流程 YAML：

- 展开该/.codecatalyst目录，然后展开该/workflows目录。打开工作流程 YAML 文件。确保在您为工作流程配置的 YAML 中指定了 IAM 角色和账户信息。例如：

```
Actions:
  cdk_bootstrap:
    Identifier: action-@v1
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: Staging
      Connections:
        - Name: account-connection
          Role: build-role
```

  要使用AWS资源运行 CodeCatalyst 工作流程生成和部署操作，需要使用 “环境”、“连接” 和 “角色” 属性。有关示例，请参阅环境、连接和角色的 CodeCatalyst 构建操作参考 YAML 参数。
- 确保您的空间中已添加一个账户，并确保该账户已向该账户添加了一个或多个相应的 IAM 角色。如果您拥有 Space 管理员角色，则可以调整或添加帐户。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。

我需要关联的账户、角色和环境才能创建项目

问题：在项目创建选项中，我的项目要么在我的空间中没有可用的已添加帐户，要么我需要在我的空间中添加另一个账户才能使用我的项目。

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AWS 账户将其添加到您的项目中。您还必须拥有一个拥有管理权限或可以与AWS管理员合作AWS 账户的地方。

要确保在项目创建屏幕中显示账户和角色，您必须先添加账户和角色。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角色策略的角色策略创建服务CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。该角色的CodeCatalystWorkflowDevelopmentRole-*spaceName*名称将附加唯一标识符。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。该角色已添加到您的账户，可在中的项目创建页面中找到 CodeCatalyst。

我无法访问中的 Amazon CodeCatalyst Spaces 页面 AWS Management Console

问题：当我尝试访问中的 Amazon CodeCatalyst 页面AWS Management Console以向我的 CodeCatalyst 空间添加账户或向中的账户添加角色时AWS，我收到权限错误。

可能的修复措施：

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AWS 账户将其添加到您的项目中。您还必须拥有一个拥有管理权限或可以与AWS管理员合作AWS 账户的地方。您必须首先确保使用您想要管理AWS Management Console的相同账户登录。登录后AWS Management Console，您可以打开控制台并重试。

在 AWS Management Console https://us-west-2.console.aws.amazon.com/codecatalyst/home?region=us-west-2#/ 中打开亚马逊 CodeCatalyst 页面。

我想要一个不同的账号作为我的结算账号

问题：当我设置 CodeCatalyst 登录名时，我完成了几个步骤来设置我的空间并关联已获得授权的人 AWS 账户。现在，我想授权另一个账号进行计费。

可能的修复方法：如果您拥有 Space 管理员角色，则可以为您的空间授权结算账号。您还必须拥有一个拥有管理权限或可以与AWS管理员合作AWS 账户的地方。

有关更多信息，请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

开发环境问题疑难解答

要解决与开发环境相关的问题，请参阅以下章节。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

- 我无法通过 SSH 连接到我的开发环境
- 我无法通过 SSH 连接到我的开发环境，因为缺少本地 SSH 配置
- 我无法通过 SSH 连接到我的开发环境，因为我的AWS Configcodecatalyst配置文件有问题
- 对 IDE 问题进行故障排除
- 对开发文件问题进行故障排除

由于配额问题，我的开发环境创建失败

问题：我想在中创建一个开发环境 CodeCatalyst，但我看到一个错误。在控制台中，我在 “开发环境” 页面上看到一条消息，提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，或者如果您拥有空间管理员角色，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，查看使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删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，请检查您的 CodeCatalyst 角色是否允许您创建开发环境。空间管理员角色、项目管理员角色和参与者角色都有创建开发环境的权限。有关更多信息，请参阅使用用户角色授予访问权限：

我无法将更改从我的开发环境推送到存储库中的特定分支

问题：我想提交开发环境中的代码更改并将其推送到源存储库中的分支，但我看到了错误。

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、项目管理员角色和参与者角色均有权将代码推送到项目中的仓库。

如果您拥有 “贡献者” 角色但无法将代码推送到特定分支，则可能为特定分支配置了分支规则，该规则禁止具有该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：我的开发环境在我停止后没有恢复。

可能的修复方法：要修复问题，请查看 Amazon CodeCatalyst 空间的 “账单” 选项卡，查看使用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以提高计费等级。

我的开发环境已断开连接

问题：我在使用开发环境时已断开连接。

可能的修复方法：要修复问题，请检查您的互联网连接。如果您未连接到互联网，请在您的开发环境中连接并继续工作。

我与 VPC 连接的开发环境出现故障

问题：我将 VPC 连接关联到我的开发环境，但它遇到了错误。

可能的修复方法：Docker 使用一种称为桥接网络的链路层设备，该设备使连接到同一桥接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Docker使用相同的 IPv4 CIDR 地址块引起的 IP 地址冲突，请配置与不同的 CIDR 块。`172.17.0.0/16`

> **Note**
>
> 您无法更改现有 VPC 或子网的 IP 地址范围。

我找不到我的项目在哪个目录

问题：我找不到我的项目所在的目录。

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

要对通过 SSH 与开发环境的连接进行故障排除，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，因为缺少本地 SSH 配置

如果缺少本地 SSH 配置 (`~/.ssh/config`) 或`Host codecatalyst-dev-env*`部分内容已过期，则您将无法通过 SSH 连接到您的开发环境。要解决此问题，请删除该`Host codecatalyst-dev-`

env*部分并再次执行 SSH Access 模式中的第一个命令。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AWS Config**codecatalyst**配置文件有问题

确保您的codecatalyst个人资料的 AWS Config (~/.aws/config) 与中描述的相匹配设置为AWS CLI与一起使用 CodeCatalyst。如果不是，请删除的配置文件codecatalyst并再次执行 SSH Access 模式中的第一个命令。有关更多信息，请参阅 使用 SSH 连接到开发环境。

对 IDE 问题进行故障排除

要解决中与 IDE 相关的问题，请参阅以下章节 CodeCatalyst。有关 IDE 的更多信息，请参阅在 IDE 中创建开发环境。

主题

- 我的运行时镜像版本不匹配 AWS Cloud9
- 我无法在/projects/projects中访问我的文件 AWS Cloud9
- 我无法AWS Cloud9使用自定义 devfile 启动我的开发环境
- 我在里面遇到了问题 AWS Cloud9
- 在 JetBrains，我无法通过以下方式连接到我的开发环境 CodeCatalyst
- 我无法AWS Toolkit为我的 IDE 安装
- 在我的 IDE 中，我无法启动我的开发环境

我的运行时镜像版本不匹配 AWS Cloud9

AWS Cloud9正在使用不同版本的前端资源和后端运行时镜像。使用不同的版本可能会导致 Git 扩展AWS Toolkit无法正常运行。要修复此问题，请导航到开发环境仪表板，停止开发环境，然后重新启动。要使用 API 修复问题，请使用 UpdateDevEnvironment API 更新运行时。有关更多信息，请参阅 Amazon CodeCatalyst API 参考UpdateDevEnvironment中的。

我无法在**/projects/projects**中访问我的文件 AWS Cloud9

AWS Cloud9编辑器无法访问目录中的文件/projects/projects。要解决此问题，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AWS Cloud9使用自定义 devfile 启动我的开发环境

您的开发文件镜像可能与不兼容。AWS Cloud9要修复此问题，请查看存储库中的开发文件和相应的开发环境，然后创建一个新的开发环境以继续。

我在里面遇到了问题 AWS Cloud9

有关其他问题，请查看《AWS Cloud9用户指南》中的 “疑难解答” 部分。

在 JetBrains，我无法通过以下方式连接到我的开发环境 CodeCatalyst

要修复此问题，请检查您是否只 JetBrains 安装了最新版本的。如果您有多个版本，请卸载旧版本，然后通过关闭 IDE 和浏览器重新注册协议处理程序。然后再次打开 JetBrains 并注册协议处理程序。

我无法AWS Toolkit为我的 IDE 安装

要修复 VS Code 的此问题，请AWS Toolkit for Visual Studio Code从中手动安装GitHub。

要修复此问题 JetBrains，请AWS Toolkit for JetBrains从中手动安装GitHub。

在我的 IDE 中，我无法启动我的开发环境

要修复 VS Code 的此问题，请检查是否已AWS Toolkit for Visual Studio Code安装最新版本的 VS 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 f CodeCatalyst or VS Code。

要修复此问题 JetBrains，请检查您是否AWS Toolkit for JetBrains安装了最新版本的。 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 Amazon 了解 CodeCatalyst 详情 JetBrains。

对开发文件问题进行故障排除

要解决中与 devfiles 相关的问题，请参阅以下章节。 CodeCatalyst有关开发文件的更多信息，请参阅为开发环境配置开发文件。

主题

- 尽管我在自定义开发文件中实现了自定义镜像，但我的开发环境仍在使用默认的通用开发文件
- 我的项目没有使用默认的通用开发文件在我的开发环境中构建
- 我想为开发环境移动存储库 devfile
- 我在启动我的开发文件时遇到了问题
- 我不确定如何查看我的开发文件状态

- 我的开发文件与最新镜像中提供的工具不兼容

尽管我在自定义开发文件中实现了自定义镜像，但我的开发环境仍在使用默认的通用开发文件

如果在启动使用自定义开发文件的开发环境时 CodeCatalyst 遇到错误，则开发环境默认为默认的通用开发文件。要解决这个问题，你可以在下面的日志中查看确切的错误/aws/mde/logs/devfile.log。您还可以在日志中检查postStart执行是否成功:/aws/mde/logs/devfileCommand.log.

我的项目没有使用默认的通用开发文件在我的开发环境中构建

要修复此问题，请检查您是否未使用自定义开发文件。如果您没有使用自定义 devfile，请在项目的源存储库中查看该devfile.yaml文件以查找并修复所有错误。

我想为开发环境移动存储库 devfile

您可以将默认开发文件/projects/devfile.yaml移入您的源代码存储库。要更新开发文件的位置，请使用以下命令：/aws/mde/mde start --location *repository-name*/devfile.yaml。

我在启动我的开发文件时遇到了问题

如果启动你的开发文件时出现问题，它将进入恢复模式，这样你仍然可以连接到你的环境并修复你的开发文件。在恢复模式下，运行时/aws/mde/mde status不会包含开发文件的位置。

```
{
    "status": "STABLE"
}
```

你可以检查下方日志中的错误/aws/mde/logs，修复开发文件，然后重试运行/aws/mde/mde start。

我不确定如何查看我的开发文件状态

你可以通过运行/aws/mde/mde status来检查你的开发文件状态。运行此命令后，您可能会看到以下内容之一：

- {"status": "STABLE", "location": "devfile.yaml" }

  这表明你的开发文件是正确的。

- `{"status": "STABLE" }`

  这表示您的开发文件无法启动并且已进入恢复模式。

您可以在下面的日志中查看确切的错误`/aws/mde/logs/devfile.log`。

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开发文件镜像：

我的开发文件与最新镜像中提供的工具不兼容

在您的开发环境中，`devfile`或者如果`latest`工具没有特定项目所需的工具，则`devfile` `postStart`可能会失败。要修复此问题，请执行以下操作：

1. 导航到您的开发文件。
2. 在你的开发文件中，改为更精细的图像版本。`latest`它可能看起来类似于以下内容：

```
components:
  - container:
      image: public.ecr.aws/amazonlinux/universal-image:1.0
```

3. 使用更新的开发文件创建新的开发环境。

对问题进行故障排除

以下信息可以帮助您解决中的常见问题 CodeCatalyst。

主题

- 我无法为我的问题选择受托人

我无法为我的问题选择受托人

问题：创建议题时，受托人列表为空。

可能的修复方法：受托人列表直接链接到列为项目成员的 CodeCatalyst 用户。要验证用户配置文件访问是否正常运行，请选择配置文件图标，然后选择用户配置文件。如果未填充用户配置文件信息，请检查运行状况报告是否存在任何事件。如果确实填充了，请提交服务单。

对 Amazon CodeCatalyst 和AWS软件开发工具包之间的问题进行故障排除 AWS CLI

以下信息可以帮助您在使用 CodeCatalyst 和AWS CLI或AWS软件开发工具包时解决常见问题。

主题

当我在命令行或终端输入aws codecatalyst时，我收到一条错误消息，说这是一个无效的选择

问题：当我尝试使用wit AWS CLI h时 CodeCatalyst，其中一个或多个aws codecatalyst命令不被识别为有效。

解决方案：此问题的最常见原因是您使用的版本不包含最新服务和命令的最新更新。AWS CLI请更新您的安装，AWS CLI然后重试。有关更多信息，请参阅 设置为AWS CLI与一起使用 CodeCatalyst。

我在运行aws codecatalyst命令时收到凭证错误

问题：当我尝试使用with时 AWS CLI CodeCatalyst，我收到一条消息，上面写着`You can configure credentials by running "aws configure".`或`Unable to locate authorization token`。

解决方案：必须配置配置AWS CLI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。

通过运行 CodeCatalyst 状况报告了解当前的服务状态

Amazon CodeCatalyst 健康报告是一个公共控制面板，可为用户提供有关资源性能和具有广泛影响的服务可用性的 up-to-the-minute 通知的 CodeCatalyst 汇总列表。您可以查看哪些资源存在问题并可能影响中的应用程序 CodeCatalyst。这使您可以跟踪系统范围内的中断和其他资源停机时间。事件发生时，运行状况报告图标上会出现一个蓝色指示器。此外，还 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，近乎实时地提供事件的详细信息和历史记录。

控制面板提供所有活动事件的列表以及过去 30 天内发生的多达 100 起以前事件的记录。您可以根据事件的更新日期整理事件列表。您也可以刷新事件列表以获取最新的更新。

以下是使用 CodeCatalyst 运行状况报告的可能工作流程：

Mateo Jackson 是 Budding Space 的开发者，拥有 Space 管理员权限。在尝试创建拉取请求时，他不断收到错误消息。他查看了自己的电子邮件，发现自己收到了一封自动生成的系统事件电子邮件，该电子邮件 CodeCatalyst 提供了影响他空间的系统问题的详细历史记录。他选择 “查看更新”，然后转到 CodeCatalyst 运行状况报告，在那里他可以查看所有系统报告的事件。他从列表中选择事件以了解更多信息。将打开一个分屏界面，提供事件上次更新的时间戳、历史记录、受影响的功能、开始时间和当前状态。他还可以看到问题仍在继续，但服务团队已开始解决这个问题。每当事件的历史或状态有更新时，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查看 CodeCatalyst 健康报告。

CodeCatalyst 健康报告概念

学习以下概念将帮助您了解 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

该事件是影响其中的应用程序和资源的系统事件 CodeCatalyst。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解决该事件。

Status

状态是事件的实时状态。它将显示为 “正在进行” 或 “已解决”。

受影响的能力

受影响的功能是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区域，包括拉取请求、问题、工作流程、测试、部署和来源。

更新于

Updated on 提供事件上次更新的时间戳。

AWS Support适用于 Amazon CodeCatalyst

创建空间时，必须连接AWS 账户并指定其为空间的结算账号。AWS 账户您指定为账单账户也是您访问亚马逊AWS Support套餐的地方 CodeCatalyst。如果您需要支持，可以从指定的支持案例中创建支持案例AWS 账户。

CodeCatalyst 空间中的用户使用中的 f AWS Support or Amazon CodeCatalyst 页面 CodeCatalyst 来管理支持案例。您可以升级到诸如 B usiness S AWS Support upport 或 Enterprise Support 之类的计划，以便在中创建和管理 CodeCatalyst 技术支持案例 CodeCatalyst。对于支持案例，可通过电话、网络或聊天获得支持。

Amazon 只能支持特定于 CodeCatalyst 服务和资源的案例 CodeCatalyst。AWS Support CodeCatalyst 资源包括在内部部署的资源 CodeCatalyst 以及由用户在中部署的资源 CodeCatalyst，但不包括为其他AWS或第三方服务部署的资源。如果您需要任何其他AWS服务的支持，则必须通过将其打开AWS Management Console。

要更改您的支持计划，请参阅更改支持计划。

> **Note**
>
> Developer Support 计划不是为生产环境设计的。如果空间账单账户有 Developer Support 计划，则该计划不会级联到其AWS Support中的 CodeCatalyst所有空间管理员和空间成员。

Amaz AWS Support on 账单 CodeCatalyst

当您在中创建空间时 CodeCatalyst，空间中的用户可以创建和管理来自 Amazon AWS Support 的支持案例 CodeCatalyst。您可以创建两种类型的客户案例：

- 该空间中的所有 CodeCatalyst 用户均可使用@@ 账户和账单支持案例。您可以根据自己的权限在中获得有关账单和账户问题的帮助 CodeCatalyst。
- 技术支持案例将您与技术支持工程师联系起来，寻求与服务相关的技术问题和第三方应用程序扩展方面的帮助。如果您拥有“基本”支持计划，则无法创建技术支持案例。

  AWS 账户被指定为空间账单账户的用户必须拥有商业支持或企业支持计划，以便该空间AWS Support CodeCatalyst 用于技术案例。

> **Note**
>
> 如果您的空间 CodeCatalyst 来自一个没有商业支持或企业支持计划的账户AWS Support用于亚马逊，那么您仍然可以将亚马逊AWS Support CodeCatalyst 用于账户和账单案例。

要获得技术支持，您必须通过 CodeCatalyst 控制台打开所有案例。您无法 CodeCatalyst 从AWS Support中为创建技术支持案例AWS Management Console。

> **Note**
>
> Amazon 不提供@@ 提高服务限额AWS Support的请求 CodeCatalyst。这些请求只能由空间账单账户的 root 用户提交AWS Support Center Console。

AWS Supportfor Amazon CodeCatalyst 具有与之相同的支持协议AWS Support，但有以下注意事项：

- AWS Support适用于中的支持案例的严重性列表、响应时间和 SL A CodeCatalyst，AWS Support详见选择严重性。
- 空间管理员和空间成员不能使用 Slack 中的 AWS Support API、AWS SDK 或AWS Support应用程序为其创建案例。 CodeCatalyst CodeCatalyst 只能从以下地址提交支持案例 CodeCatalyst。

> **Note**
>
> CodeCatalyst 未与AWS Trusted Advisor我们的AWS事件检测和响应完全集成。验证集成方式 CodeCatalyst ，确保您的业务实践与当前的集成保持一致。

您必须是要请求支持的空间中的用户。

> **Note**
>
> 如果您的办公空间中有多个构建器，我们建议您购买 Business Support 或 Enterprise Support 计划。这些计划为最多可容纳5,000名建筑商的空间提供技术支持。

被AWS 账户指定为空间的账单账户使用AWSRoleForCodeCatalystSupport角色和AmazonCodeCatalystSupportAccess托管策略。这允许空间中的 CodeCatalyst 用户访问 for Amazon CodeCatalyst 页面。AWS Support有关此角色和策略的更多信息，请参阅AmazonCodeCatalystSupportAccess。有关账单的其他注意事项，请参阅《Amazon CodeCatalyst 管理员指南》中的管理账单。

以下是构建者在以下位置创建支持案例的可能流程 CodeCatalyst：

马特奥·杰克逊（Mateo Jackson）是一个项目的开发者. CodeCatalyst 注册管理亚马逊账单的 CodeCatalyst 并升级AWS 账户到 Business Support 计划后，该领域的所有构建者都可以创建技术支持案例。AWS SupportMateo 为其项目中失败的工作流程提交了技术支持案例。Mateo 使用 AWS Support fo CodeCatalyst r Amazon 页面填写表单并创建案例，并在请求中提供工作流程 ID 和其他详细信息。该案例使用案例 ID 创建，包括AWS 账户指定为账单账户并与该空间的支持计划关联的账户 ID。

虽然所有构建者都可以在中AWS Support为创建支持案例 CodeCatalyst，但您无需为每个创建的案例付费。根据您在空间账单账单账户中购买的AWS Support高级套餐，您几乎可以无限量打开案例和联系人。

> **Note**
>
> 空间账单账户是您AWS 账户为 CodeCatalyst 用户和资源收取的费用。如果您已部署到其他服务AWS 账户，请AWS Support通过联系以AWS Management Console获取有关部署到其他服务的资源方面的帮助。
> 您可以从工作流程中识别AWS 账户您部署到的。

为 Amazon 设置您的空间 AWS Support CodeCatalyst

AWS Supportf CodeCatalyst or Amazon 在 AWS Support API 集成过程中管理支持案例 CodeCatalyst。

该AWSRoleForCodeCatalystSupport角色是一个服务角色，用于您所在领域的支持案例。必须将该角色添加到该空间的指定账单账户中。有关更多信息或要创建角色，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，您必须创建角色才能 CodeCatalyst 让支持人员为您的空间工作。如果在 2023 年 4 月 20 日之后创建空间，则可以在空间创建期间、账单详情页面或点击中的 CodeCatalyst支持横幅链接来创建角色 CodeCatalyst。

为您的空间设置支持

1. 创建 CodeCatalyst 空间时，系统会指示您关联结算账号。该空间的指定账单账户将按以下方式计费。AWS有关创建空间的更多信息，请参阅创建您的第一个空间和开发角色（无需邀请即可开始）。
2. 创建 CodeCatalyst 空间时，可以使用该选项创建允许 CodeCatalyst 用户访问支持的`AWSRoleForCodeCatalystSupport`服务角色。该角色使用托管策略`AmazonCodeCatalystSupportAccess`。必须将该角色添加到空间的AWS账户指定账号中。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. 对于空间的指定账单账户，建议空间管理员为该空间购买商业支持或企业支持计划AWS 账户。该领域的所有成员都将能够管理来自Amazon AWS Support 的支持案例 CodeCatalyst，并且支持渠道将与您购买的集成完成的AWS Support计划保持一致。
4. 要在中创建和管理支持案例 CodeCatalyst，请参阅在中创建 CodeCatalyst 支持案例 CodeCatalyst。

访问 CodeCatalyst 中的支持 AWS Management Console

如果某个空间已启用支持的账单账户断开连接，则与之前的空间账单账户和关联的支持计划相关的AWS Support案例将不再显示在 AWS Support Amazon 中 CodeCatalyst。该账单账户的根用户可以从中查看和解决旧案例，还可以设置 IAM 权限AWS Support供其他用户查看和解决旧案例。AWS Management Console您仍然可以从所有其他支持计划中受益，AWS 服务并完成以前未解决的任何 CodeCatalyst 支持案例。AWS Management Console

有关更多信息，请参阅《AWS Support用户指南》中的更新、解决和重新审理您的问题。

也 CodeCatalyst 可以在中打开有关一般操作方法信息的支持案例AWS Management Console，但无法通过此渠道获得技术支持。 CodeCatalyst有关更多信息，请参阅AWS Support用户指南中的创建支持案例和案例管理。

以下是用户解决支持 CodeCatalyst 案例的可能流程AWS Management Console：

虽然所有建筑商都可以通过 Amazon 创建支持案例 CodeCatalyst，但支持请求是从指定为该空间账单账户的账户中计费的。AWS Support马特奥·杰克逊（Mateo Jackson）是一个项目的开发人员，CodeCatalyst 他为项目中失败的工作流程提出了技术支持案例。但是，注册亚马逊 CodeCatalyst 并购买了 Business Su AWS Support pport 计划的空间的账单账户已与该空间断开连接。Mateo 查看最新沟通并解决待处理案例的唯一方法 CodeCatalyst 是在AWS Support中心管理问题编号。AWS Management Console为此，Mateo 从其支持案例中关联的先前空间账单账户的根用户那里获得了 IAM 权限，并通过AWS Support控制台解决问题。

> **Important**
>
> 如果您更改了空间的指定账单账户，则在月底之前仍然可以访问您的AWS Support套餐。AWS Management Console您需要使用更新的账单账户AWS Support进行重新购买，才能继续访问之前在中 CodeCatalyst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账单账户，以免对通过 AWS Support Amazon 访问支持案例产生任何影响 CodeCatalyst。

在中创建 CodeCatalyst 支持案例 CodeCatalyst

您可以在 for Amazon CodeCatalyst 页面上AWS Support创建支持案例。

AWSBuilder ID 只能获得对他们进行身份验证的别名的支持，并且只能获得基于其权限的资源的支持。账户和账单选项适用于所有空间管理员和空间成员。但是，用户只能获得他们有权访问的资源的支持，CodeCatalyst 而不能获得与账户账单管理相关的支持。

您可以使用空间的 for CodeCatalyst 页面为您的 CodeCatalyst 资源创建账户和账单案例或技术支持案例。AWS Support

> **Note**
>
> Amazon 只能支持特定于 CodeCatalyst 服务和资源的案例 CodeCatalyst。AWS Support CodeCatalyst 资源包括在内部部署的资源 CodeCatalyst 以及由用户在中部署的资源 CodeCatalyst，但不包括为其他AWS或第三方服务部署的资源。如果您需要任何其他AWS服务的支持，则必须通过将其打开AWS Management Console。

要在中创建支持案例 CodeCatalyst

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，请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 Supp ort。
4. 选择 Create case（创建案例）。
5. 请选择以下任一选项：

   - Account and billing（账户和账单）
   - Technical（技术）

> **Note**
>
> 在 AWS Support Amazon 中 CodeCatalyst，如果将商业支持或企业支持计划添加到空间账单账户，则所有空间管理员和空间成员都将获得 CodeCatalyst 技术案例支持。有关故障排除信息，请参阅我无法为自己的空间创建技术支持案例。
> AWS Support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AWS Support高级套餐，才能在所有空间中获得技术支持。

6. 选择 Service（服务）、Category（类别）和 Severity（严重性）。有关选择严重性的信息，请参阅选择严重性。

   - 一般指南
   - 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 Next step: Additional information（下一步：其他信息）。
8. 在 Additional information（其他信息）页面上，对于 Subject（主题），请为您的问题输入一个标题。
9. 对于 Description（描述），请按照提示操作以描述您的情况，例如：

- 特定于的故障排除信息 CodeCatalyst，例如工作流程 ID、日志或屏幕截图
- 您收到的错误消息
- 您遵循的故障排除步骤

> **Note**
>
> 不要在信件中共享任何敏感信息，例如凭证、信用卡、签名 URL 或个人身份信息。

10. （可选）选择 Attach files（附加文件）以为您的工单添加任何相关文件，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. 在生成器名称中，将自动填充与您的AWS建造者 ID 关联的全名。
13. （可选）在 “项目名称” 中选择项目（如果适用）。

> **Note**
>
> 您只会看到您有权访问的项目。如果您需要访问其他项目，请在创建支持案例之前要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. 在首选联系人语言中，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. 查看您的案例详情，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，所有有权访问您的支持案例中定义的空间和项目（如果选择）的成员都可以查看。目前无法省略个人用户的支持案例。

在中解决支持案例 CodeCatalyst

您可以从 for Amazon CodeCatalyst 页面解决未解决AWS Support的支持案例。

在要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 Space 管理员角色，或者在创建案例时选择了项目，则还需要拥有该项目的成员资格才能查看和解决案例。

要解决悬而未决的支持案例 CodeCatalyst

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，请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 SupportAmazon CodeCatalyst。
4. 选择您要管理的支持案例的链接。选择 Resolve case（解决案例）。

在中重新打开支持案例 CodeCatalyst

您可以使用 for Amazon CodeCatalyst 页面上的 “重新打开已解决AWS Support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，请打开一个新问题并将之前的问题编号作为参考。
> 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。

要重新打开支持案例 CodeCatalyst

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，请在顶部导航栏中选择一个空间。

3. 在页面顶部，选择？ 图标，然后选择适用于 AWS Support CodeCatalyst。
4. 选择您要管理的支持案例的链接。选择 Reopen（重新打开）。在确认屏幕上选择 “确定”，然后选择 “提交”。

5. 在描述中填写有关同一问题的最新信息。不要在信件中共享任何敏感信息，例如凭证、信用卡、签名 URL 或个人身份信息。

的配额 CodeCatalyst

下表描述了 Amazon 的配额和限制 CodeCatalyst。您可以在以下主题中找到有关特定 CodeCatalyst 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
- 工作流程配额
- 中开发环境的配额 CodeCatalyst
- 项目配额
- 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格数	5
用户在一个日历月内可以创建的最大空间数	5
空间的最小数量 AWS 账户	1
空间的最大账号连接数	5000
AWS 账户 作为空间账单账户的最大数量	1
一个空间的最大 VPC 连接数	100
空间中项目的最大数量	100
用户可以加入的最大项目数	1000
空间描述	空间描述是可选的。如果指定，则其长度必须介于 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、破折号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。您不能重复使用已删除空间的名称。 空间名称的长度必须介于 3 到 63 个字符之间。它们还必须以字母数字字符开头。空间名称可以包含字母、数字、句点、下划线和破折号的任意组合。它们不能包含以下任何字符： `! ? @ # $ % ^ & * ( ) + = { } [ ] | 八 > < ~ ` ' " ; :`

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
新内容：增加了处理人际关系的步骤	添加了创建和删除个人联系的步骤。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
更新内容：使用带有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创建 GitHub 或 Bitbucket 存储库 使用蓝图创建项目在项目中应用蓝图以添加资源，或者创建自定义蓝图。	2024年6月5日
新内容：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” 动作 YAML 定义	修复了这个`CdkRootRootPath` 例子。	2024 年 5 月 28 日
更新的内容	更新了主题标题并重新组织了内容，以提高可读性和发现性。如果您想就这些更改提供反馈，请使用此提供反馈链接。	2024 年 5 月 17 日
新内容：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文件更改历史记录的新功能。	2024 年 5 月 1 日

更新内容：教程：使用生成式 AI 功能	更新了教程，以反映与 Amazon Q 开发者的集成。	2024 年 4 月 29 日
更新内容：教程：使用生成式 AI 功能	更新了教程，以反映允许 Amazon Q 分析问题的复杂性、建议和创建任务以及处理议题中的任务。	2024 年 4 月 22 日
新内容：对工作流程运行进行门控	添加了对工作流程运行进行门控要求工作流程运行获得批准、和其他几个与工作流程批准相关的主题。	2024 年 4 月 22 日
新内容：教程：使用可组合的 PDK 蓝图创建全栈应用程序	添加了有关在亚马逊 CodeCatalyst 项目中使用 AWS 项目开发套件 (AWS PDK) 蓝图的新教程。	2024 年 4 月 9 日
新内容：使用任务将问题分解为较小的目标	添加了支持问题中任务启动的内容。可以向议题中添加任务，以进一步分解、组织和跟踪该议题的工作情况。	2024 年 4 月 4 日
更新的内容：使用构件在工作流程中的操作之间共享数据	更新了使用构件在工作流程中的操作之间共享数据主题，增加了两个新的副主题：我能否在不将工件指定为输出和输入的情况下共享它们？和 我能否在工作流程之间共享工件？	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
新内容：“AWS CDK 部署” 动作 YAML 定义	向中添加了一个新`CloudAssemblyRootPath` 属性“AWS CDK 部署” 动作 YAML 定义。	2024 年 4 月 1 日

更新的内容：指定运行时环境 Docker 镜像	更新了指定运行时环境 Docker 镜像主题，加入了有关 2024 年 3 月新版运行时环境镜像的信息。	2024 年 3 月 26 日
更新内容：使用角色	将角色权限信息整合到单个表中。该表位于新查看每个角色的可用权限主题中。	2024 年 3 月 18 日
新内容：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 18 日
新内容：示例：带有拉动和分支的扳机	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程主题中做了一些小的更正。	2024 年 3 月 11 日
更新内容：使用角色	更新了角色的文档，增加了创建、删除和查看环境的权限。	2024 年 3 月 4 日
更新内容：教程：使用生成式 AI 功能	更新了教程，以反映在创建问题并将其分配给 Amazon Q 时所做的更改。	2024 年 3 月 4 日
新内容：议题组件	添加了有关如何以自定义蓝图开发者的身份使用议题组件的新内容。	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

更新内容：处理拉取请求	更新了文档，以反映新功能，包括批准规则和合并拉取请求的首要要求。	2024 年 2 月 15 日
更新内容：合并拉取请求	为拉取请求添加了文档，以包含有关重写合并要求的信息，以合并尚未获得所需审阅者批准或符合批准规则的拉取请求。	2024 年 2 月 15 日
新内容：管理批准规则	添加了拉取请求文档，以包含有关创建和管理批准规则的信息。	2024 年 2 月 15 日
更新内容：使用角色	更新了角色的文档，增加了使用批准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* #错误” 错误？	更新了本如何修复 “工作流定义有 *n* #错误” 错误？节以包含更多疑难解答提示。	2024 年 2 月 9 日
新内容：查看工作流程状态	添加了描述工作流程状态的部分。	2024 年 2 月 9 日
新内容：查看工作流程状态	添加了描述工作流程状态的部分。	2024 年 2 月 9 日
更新内容：工作流程配额	使用每个工作流程的最大操作数和与 AWS 账户 每个空间配额关联的最大环境数更新了工作流程配额主题。	2024 年 2 月 7 日
更新的内容：创建环境	更新了该创建环境部分，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
更新内容：创建问题	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
新内容：教程：使用生成式 AI 功能	添加了在 Amazon 中使用生成式 AI 功能的教程 CodeCatalyst。	2023 年 11 月 28 日
新内容：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	使用修复和故障排除信息更新了教程。	2023 年 11 月 22 日
更新的内容：使用触发器自动启动工作流程	修复了一些与拉取请求触发器相关的示例和描述。添加了一个分支时的触发注意事项章节。	2023 年 11 月 22 日
新内容：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

更新内容：使用角色	更新了角色的文档，使其包括与团队合作的权限、VPC 连接、单点登录和计算机资源。	2023 年 11 月 16 日
更新内容：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	使用空间配额的最大 VPC 连接数更新了的配额 CodeCatalyst主题。	2023 年 11 月 16 日
新内容：管理空间和 CodeCatalyst 项目的团队	添加了有关使用带空间的团队的信息。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
新内容：管理空间中蓝图和工作流程的计算机资源	添加了有关使用带空格的计算机资源的信息。请参阅 允许计算机资源访问空间 。	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 允许对计算机资源进行项目访问。	2023 年 11 月 16 日
新内容：将 VPC 连接与环境关联	添加了有关将 VPC 连接与环境关联的文档，该文档可在工作流程中使用。	2023 年 11 月 16 日
新内容：将 VPC 连接与开发环境关联	添加了有关使用具有 VPC 连接的开发环境的文档。	2023 年 11 月 16 日
新增内容	《Amazon CodeCatalyst 管理员指南》的首次发布。	2023 年 11 月 16 日

新内容：“AWS CDK 部署” 动作 YAML 定义	向“AWS CDK 部署” 动作 YAML 定义和添加了一个新CdkCliVersion 属性“AWS CDK 引导” 操作 YAML 定义。	2023 年 11 月 14 日
更新内容：使用角色	更新了角色的文档，增加了使用分支规则的权限。	2023 年 11 月 13 日
更新内容：源代码库、工作流程和开发环境问题疑难解答	更新了疑难解答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：生成和测试操作 YAML 定义	更新了该Environment 属性的文档。现在，它是生成和测试操作的可选字段。	2023 年 11 月 13 日
新内容：管理分支规则	为分支添加了文档，其中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
更新内容：处理拉取请求	更新了文档，以反映拉取请求信息显示方式的变化。	2023 年 11 月 10 日
更新的内容：在工作流程运行之间缓存文件	更新了文档，增加了文件缓存限制。	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，提及了 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
新增和更新内容：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 输出参数以便其他操作可以使用它	更新了示例，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
新内容：使用触发器自动启动工作流程	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 定义	在“部署到 Kubernetes 集群” 操作 YAML 定义和教程：将应用程序部署到 Amazon EKS主题中添加了有关使用该CodeCatalystWorkflowDevelopmentRole-*spaceName* 角色的信息。	2023 年 9 月 22 日
更新内容：该角色的新角色名称和CodeCatalystWorkflowDevelopmentRole-***spaceName*** 策略	将开发者角色名称更改的步骤和角色描述更新为CodeCatalystWorkflowDevelopmentRole-***spaceName*** 。开发者角色现在使用AdministratorAccess AWS 托管策略。请参阅了解CodeCatalystWorkflowDevelopmentRole-***spaceName*** 服务角色和创建您的第一个空间和开发角色（无需邀请即可开始）。	2023 年 9 月 20 日
更新的内容：在工作流程中配置和使用变量	引入了两个新概念：用户定义变量和预定义变量。这些概念应使本在工作流程中配置和使用变量节更易于阅读和理解。	2023 年 9 月 19 日
更新内容：处理提交	更新了文档，以反映显示信息的变化，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

新内容：使用触发器自动启动工作流程	在使用触发器自动启动工作流程主题中添加了以下示例：示例：一个简单的 “推到主线” 触发器	2023 年 9 月 6 日
更新内容：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显示顺序的变化。	2023 年 8 月 30 日
新内容：查看和更改默认分支	为分支添加了文档，以包含有关查看和更改源存储库默认分支的信息。	2023 年 8 月 30 日
更新的内容：“部署到 Kubernetes 集群” 操作 YAML 定义	在中的Manifests属性描述中添加了有关 Helm 和 Kustomize 的注释。“部署到 Kubernetes 集群” 操作 YAML 定义	2023 年 8 月 15 日
新内容：管理议题附件	添加了有关处理和管理议题附件的文档。	2023 年 8 月 15 日
更新的内容：使用触发器自动启动工作流程	改进和扩展了与工作流程触发器相关的文档。	2023 年 8 月 11 日
新内容：角色权限疑难解答	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
新内容：如何修复 “工作流程处于非活动状态” 消息？	添加了以下疑难解答主题：如何修复 “工作流程处于非活动状态” 消息？	2023 年 8 月 11 日

新内容：使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service	添加了部署到 Kubernetes 集群操作的文档。有关更多信息，请参阅 使用工作流程将应用程序部署到亚马逊 Elastic Kubernetes Service 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程	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：工作流程配额	使用操作可以运行配额的最大时间更新了工作流程配额主题。	2023 年 6 月 27 日
更新的内容：工作流程 YAML 定义	修复了`Compute`代码块中的格式错误。	2023 年 6 月 27 日
更新内容：数据保护	更新了文档，增加了有关数据复制的更多信息。	2023 年 6 月 26 日
新内容：指定操作的主版本、次要版本或补丁版本	添加了一个指定操作的主版本、次要版本或补丁版本主题。	2023 年 6 月 21 日

更新的内容：部署到环境中的 VPC AWS 账户 和带有 CodeCatalyst环境的 VPC	澄清了该哪些操作支持环境？部分。	2023 年 6 月 14 日
更新内容：重新整理的问题文档	重组了大多数问题文档，以更好地与整体文档集和用户流程保持一致。	2023 年 5 月 31 日
更新内容：问题视图切换器	更新了各种用户流程，使其与更新的议题视图切换器保持一致。	2023 年 5 月 31 日
更新内容：管理通知	更新了通知文档，以包含有关配置个人 Slack 通知的信息。	2023 年 5 月 30 日
更新内容：管理通知	更新了通知文档，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：将工作流程连接到源存储库	简化了中的说明引用源存储库中的文件。	2023 年 5 月 10 日
更新的内容：工作流程配额	使用输出变量值的最大长度配额更新了工作流程配额主题。	2023 年 5 月 10 日
新内容：使用构件在工作流程中的操作之间共享数据	添加了两个示例：示例：在单个构件中引用文件和示例：存在对象时引用对象中的文件 WorkflowSource 。	2023 年 5 月 10 日

更新内容：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
更新内容：管理通知	更新了通知文档，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 托管式策略：AmazonCodeCatalystFullAccessAWS 托管式策略：AmazonCodeCatalystReadOnlyAccess、和AWS 托管式策略：AmazonCodeCatalystSupportAccess托管策略。请参阅 对 AWS 托管式策略的 CodeCatalyst 更新。	2023 年 4 月 20 日
新内容：移除部署目标	添加了一个移除部署目标主题。	2023 年 4 月 20 日
新内容：操作类型	添加了一个CodeCatalyst 行动主题。	2023 年 4 月 20 日
用于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角色的信息。请参阅 移除或更改具有 Space 管理员角色的用户的角色。	2023 年 4 月 19 日
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
更新内容：查找和查看问题	重新组织了查找和查看问题主题和副主题。	2023 年 4 月 19 日

更新的内容：为工作流程配置计算和运行时环境 Docker 镜像	在亚马逊 Linux 上增加了对 Arm64 架构的支持 2.	2023 年 4 月 19 日
新内容：在群组内移动议题	添加了有关在 Bo ard 和 “所有议题” 视图的群组内移动议题的文档。	2023 年 4 月 19 日
更新的内容：工作流程配额	更新了缺少配额工作流程配额的主题，并将单个操作输出变量配额的最大总大小更新为 120 KB（从 2 KB）。	2023 年 4 月 18 日
新内容：查看动作的源代码	添加了一个查看动作的源代码主题。	2023 年 4 月 18 日
新内容：重试报告的测试用例	添加了一个重试报告的测试用例主题。	2023 年 4 月 11 日
新内容：停止工作流运行	添加了一个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 IAM 连接资源策略的信息。请参阅使用标签控制对账户连接资源的访问权限和CodeCatalyst 权限参考。	2023 年 4 月 6 日
新内容：操作类型	添加了一个操作类型主题。	2023 年 4 月 6 日
更新的内容：“部署 AWS CloudFormation 堆栈” 操作 YAML 定义	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：使用链接 GitHub 存储库 CodeCatalyst 在中创建项目	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

更新内容：处理通知	更新了通知文档，以包含有关配置有关项目事件的电子邮件的信息。	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	通过删除登录页面和合并话题更新了 “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
更新内容：使用角色	更新了 Space 管理员、项目管理员和参与者角色的文档，增加了将议题链接到拉取请求的权限。	2023 年 3 月 13 日
更新内容：处理拉取请求	更新了拉取请求的文档，以包含有关将议题与拉取请求关联的信息。	2023 年 3 月 13 日
更新内容：处理问题	更新了议题文档，以包含有关将议题与拉取请求关联的信息。	2023 年 3 月 13 日
新内容：查看项目中所有运行的状态和详细信息	添加了描述新的聚合工作流程运行页面的部分。	2023 年 3 月 8 日
新内容：如何修复 “工作流定义有 *n* #错误” 错误？	添加了有关如何解决 “工作流程定义有错误” 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的用户界面。	2023 年 3 月 3 日

新内容：集成 universal-test-runner 到测试操作中	添加了一个集成 universal-test-runner 到测试操作中主题。	2023 年 3 月 3 日
更新的内容：使用中的工作流程构建、测试和部署 CodeCatalyst	更新了多个部分，以反映工作流程摘要页面上新的源存储库、分支和工作流程名称筛选器。	2023 年 3 月 2 日
新内容：通过提交跟踪部署状态	添加了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
新内容：源生成的变量（“BranchName” 和 CommidId “”）	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 PAT 的步骤。	2023 年 2 月 15 日
更新的内容：指定运行时环境 Docker 镜像	在 “默认图像工具版本” 表中添加了更多工具。	2023 年 1 月 10 日
更新的内容：使用构件在工作流程中的操作之间共享数据	修复了神器路径。	2023 年 1 月 3 日
更新的内容：“GitHub 动作” 动作 YAML 定义	修复了该`Steps`部分中的代码片段。	2023 年 1 月 3 日
更新的内容：将工作流程连接到源存储库	修复了源路径。	2023 年 1 月 3 日
更新内容：更新拉取请求	更新了文档，以包含有关更新拉取请求的必填审稿人或可选审阅者的信息。	2022 年 12 月 23 日

新内容：在工作流程运行之间缓存文件	在工作流程中添加了一个用于文件缓存的页面。	2022 年 12 月 20 日
更新内容：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
新内容：“AWS CDK 部署” 动作 YAML 定义	添加了一个新`CdkRootPath` 属性。	2022 年 12 月 16 日
新内容：跨操作共享计算	添加了一个跨操作共享计算主题。	2022 年 12 月 14 日
更新的内容：使用构件在工作流程中的操作之间共享数据	修复了显示如何指定输入工件的示例。	2022 年 12 月 13 日
新内容：“GitHub 动作” 动作 YAML 定义	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，在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

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。

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。