用户指南

AWS CodeCommit

API 版本 2015-04-13

AWS CodeCommit: 用户指南

Table of Contents

什么是 AWS CodeCommit？

AWS CodeCommit 是一项由 Amazon Web Services 托管的版本控制服务，可让您在云中通过私有方式存储和管理资产（例如文档、源代码和二进制文件）。有关 CodeCommit 的定价信息，请参阅定价。

> **Note**
>
> CodeCommit 适用于许多合规性计划。有关 AWS 和合规性工作的详细信息，请参阅 AWS 按合规性计划提供的范围内服务。
> 这是一项符合 HIPAA 要求的服务。有关AWS、《1996 年健康保险可携性与责任法》(HIPAA) 以及使用AWS服务处理、存储和传输受保护的医疗信息 (PHI) 的更多信息，请参阅 HIPAA 概述。
> 有关该服务以及规定安全管理最佳实践的安全管理标准 ISO 27001 的信息，请参阅 ISO 27001 概述。
> 有关此服务和支付卡行业数据安全标准 (PCI DSS) 的信息，请参阅 PCI DSS 概述。
> 有关此服务和美国联邦信息处理标准 (FIPS) 第 140-2 版美国政府标准（其中规定了对保护敏感信息的加密模块的安全要求）的信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版概览 和 Git 连接端点。

主题

CodeCommit 简介

CodeCommit 是一项安全、高度可扩展的托管式源代码控制服务，用于托管私有 Git 存储库。借助 CodeCommit，您无需管理自己的源代码控制系统，也无需担心其基础设施的扩展。您可以使用 CodeCommit 存储从代码到二进制文件的一切内容。它支持 Git 的标准功能，可与您现有的基于 Git 的工具无缝协作。

使用 CodeCommit，您可以：

- 受益于 AWS 托管的完全托管服务。CodeCommit 可提供高服务可用性和高持久性，并能消除管理您自己的硬件和软件的管理开销。没有需要预置和扩展的硬件，也没有需要安装、配置和更新的服务器软件。
- 安全地存储您的代码：CodeCommit 存储库在静止和传输时处于加密状态。
- 协作处理代码：CodeCommit 存储库支持拉取请求，使用户可以相互查看和评论各自的代码更改，然后再将其合并到分支，还支持通过通知功能自动向用户发送有关拉取请求和评论的电子邮件等。
- 轻松扩展您的版本控制项目：CodeCommit 存储库可以扩展以满足您的开发需求。该服务能够处理具有大量文件或分支、大尺寸文件及冗长版本历史记录的存储库。
- 随时随地存储：CodeCommit 不限制您的存储库大小，也不限制可存储的文件类型。
- 与其他 AWS 及第三方服务集成。CodeCommit 将您的存储库放置在靠近您托管在 AWS 云中的其他生产资源的位置，这有助于提高您的开发周期的速度和频率。它与 IAM 集成，并且可与其他 AWS 服务结合使用以及与其他存储库同时使用。有关更多信息，请参阅产品和服务与 AWS CodeCommit。
- 轻松地从远程存储库迁移文件。您可以从任意 Git 存储库迁移到 CodeCommit。
- 使用自己熟悉的 Git 工具：CodeCommit 支持 Git 命令及其自己的 AWS CLI 命令和 API。

CodeCommit、Git 以及选择合适的 AWS 服务来满足您的需求

作为一项基于 Git 的服务，CodeCommit 非常适合用于满足大多数版本控制需求。未对文件大小、文件类型和存储库大小施加任何限制。但是，Git 有一些固有的限制，这些限制会对某些类型的操作的性能产生负面影响，特别是随着时间的推移。通过避免在其他 AWS 服务更适合于任务的使用案例中使用 CodeCommit 存储库，可以避免 CodeCommit 存储库性能的潜在下降。您还可以为复杂的存储库优化 Git 性能。以下是一些使用案例，其中 Git（也就是 CodeCommit）可能不是最适合您的解决方案，或者您可能需要采取额外步骤来优化 Git。

使用案例	描述	要考虑的其他服务
经常更改的大文件	Git 使用增量编码来存储各个文件版本之间的差异。例如，如果您更改文档中的几个单词，Git 将只存储这些已更改的单词。如果您的文件或对象的大小超过 5 MB，并且进行了许	要对大型文件进行版本控制，请考虑使用 Amazon Simple Storage Service (Amazon S3)。有关更多信息，请参阅《Amazon Simple Storage

使用案例	描述	要考虑的其他服务
	多更改，则 Git 可能需要重新构造一个很大的增量差异链。当这些文件随着时间的推移而增长时，这会在本地计算机和 CodeCommit 中使用越来越多的计算资源。	Service 用户指南》中的使用版本控制。
数据库	随着时间的推移，Git 存储库会变得越来越大。由于版本控制将跟踪所有更改，因此，任何更改都将增大您的存储库。换句话说，在提交数据时，即使删除了提交中的数据，也会将数据添加到存储库中。随着时间的推移，需要处理和传输的数据会越来越多，Git 的速度将变慢。这对数据库使用案例尤其不利。Git 不是作为数据库设计的。	要创建和使用具有一致性能的数据库（而不管大小如何），请考虑使用 Amazon DynamoDB。有关更多信息，请参阅 Amazon DynamoDB 入门指南。
审核跟踪	通常，审核跟踪会保持很长一段时间，并且由系统进程以非常频繁的节奏连续生成。Git 旨在安全地存储由开发人员组在开发周期中生成的源代码。快速更改的存储库会不断存储以编程方式生成的系统更改，但性能会随着时间的推移而降低。	要存储审计跟踪记录，请考虑使用 Amazon Simple Storage Service (Amazon S3)。 要审计 AWS 活动，请考虑使用 AWS CloudTrail、AWS Config 或 Amazon CloudWatch，具体取决于您的使用案例。

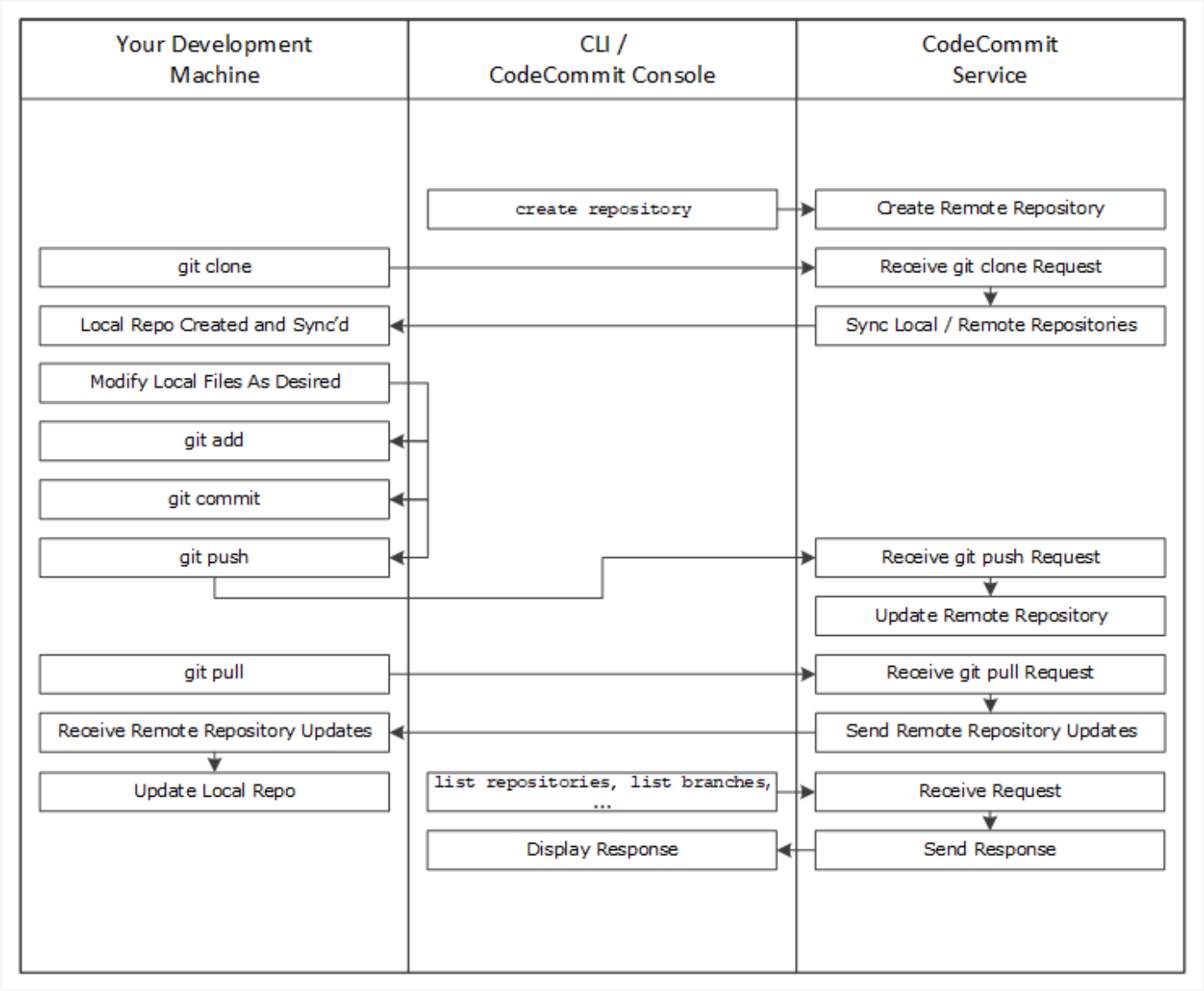
使用案例	描述	要考虑的其他服务
备份	Git 是为开发人员编写的版本源代码而设计的。您可以将提交推送到两个远程存储库（包括一个 CodeCommit 存储库）作为备份策略。但是，Git 并不是设计用来处理计算机文件系统、数据库转储或类似备份内容的备份。这样做可能会降低系统速度，增加克隆和推送存储库所需的时间。	有关如何备份到 AWS 云的信息，请参阅备份和还原。

大量分支或引用	当 Git 客户端推送或提取存储库数据时，即使您只对单个分支感兴趣，远程服务器也必须发送所有分支和引用（如标签）。如果您有数千个分支和引用，这可能需要一些时间来处理和发送（打包协商）并导致明显缓慢的存储库响应。您拥有的分支和标签越多，此过程所需的时间就越长。我们建议使用 CodeCommit，但需删除不再需要的分支和标签。	要分析 CodeCommit 存储库中的引用数以确定可能不需要的引用，您可使用以下命令之一：
		• Linux、macOS、Unix 或 Windows 上的 Bash 模拟器：
		`git ls-remote | wc -l`
		• Powershell：
		`git ls-remote | Measure-Object -line`

CodeCommit 如何运作？

CodeCommit 会给基于 Git 的存储库的用户带来熟悉的使用感受，即使是不熟悉此类存储库的用户，也会发现转换到 CodeCommit 的过程十分简单。CodeCommit 提供一个控制台，您可以使用它轻松创建存储库并列出现有的存储库和分支。用户只需执行几个简单的步骤，就能找到存储库的相关信息、将其克隆到自己的计算机、创建可进行更改的本地存储库，然后将更改推送到 CodeCommit 存储库。用户可以在本地计算机上使用命令行或使用基于 GUI 的编辑器执行作业。

下图显示了使用开发计算机、AWS CLI 或 CodeCommit 控制台和 CodeCommit 服务创建和管理存储库的过程：



1. 使用 AWS CLI 或 CodeCommit 控制台创建 CodeCommit 存储库。
2. 在开发计算机上，使用 Git 运行 git clone 并指定 CodeCommit 存储库的名称。这将创建一个连接到 CodeCommit 存储库的本地存储库。
3. 在您的开发计算机上使用本地存储库修改（添加、编辑和删除）文件，然后运行 git add 将修改后的文件暂存在本地。运行 git commit 以在本地提交文件，然后运行 git push 以将文件发送到 CodeCommit 存储库。

4. 下载其他用户的更改。运行 git pull 以将 CodeCommit 存储库中的文件与您的本地存储库同步。这可确保您使用最新版本的文件。

您可以使用 AWS CLI 或 CodeCommit 控制台跟踪和管理自己的存储库。

CodeCommit 与 Amazon S3 中的文件版本控制有何区别？

CodeCommit 针对团队软件开发进行了优化。它能够管理多个文件的更改批次（当其他开发人员也在进行更改时，可能会发生这种情况）。Amazon S3 版本控制支持恢复以往版本的文件，但它不侧重于软件开发团队需要的协作文件跟踪功能。

如何开始使用 CodeCommit？

要开始使用 CodeCommit，请执行以下操作：

1. 按照设置 中的步骤准备您的开发计算机。
2. 按照开始使用中的一个或多个教程中的步骤操作。
3. 在 CodeCommit 中创建版本控制项目，或将版本控制项目迁移到 CodeCommit。

哪里可以找到有关 Git 的更多信息？

如果您还不知道它，应该先了解如何使用 Git。下面是一些有用的资源：

- Pro Git - Pro Git 图书的在线版本。作者是 Scott Chacon。由 Apress 出版。
- Git Immersion - 一个以实验为本的探索之旅，旨在指导您掌握使用 Git 的基础知识。由 Neo Innovation, Inc. 发布。
- Git 参考 - 一个在线快速参考，也可用作更深入的 Git 教程。由 GitHub 团队发布。
- Git 备忘单 - 包含基本 Git 命令的语法。由 GitHub 团队发布。
- Git 袖珍指南。作者是 Richard E. Silverman。由 O'Reilly Media, Inc. 发布。

对 AWS CodeCommit 进行设置

您可以登录到 AWS Management Console并直接从 控制台在存储库中上传、添加或编辑文件AWS CodeCommit。这是一种执行更改的快速方法。但是，如果您希望处理多个文件、处理跨分支文件等，请考虑设置您的本地计算机来使用存储库。设置 CodeCommit 的最简单方法是为 AWS CodeCommit 配置 HTTPS Git 凭证。该 HTTPS 身份验证方法：

- 使用静态用户名和密码。
- 适用于 CodeCommit 支持的所有操作系统。
- 兼容支持 Git 凭证的集成开发环境 (IDE) 及其他开发工具。

如果出于操作原因您不想或不能使用 Git 凭证，也可以使用其他方法。例如，如果您使用联合访问、临时凭证或 Web 身份提供程序访问 CodeCommit 存储库，则无法使用 Git 凭证。我们建议您使用 `git-remote-codecommit` 命令设置本地计算机。请仔细查看这些选项，以确定最适合您的替代方法。

- 使用 Git 凭证进行设置
- 使用其他方法进行设置
- CodeCommit、Git 及其他组件的兼容性

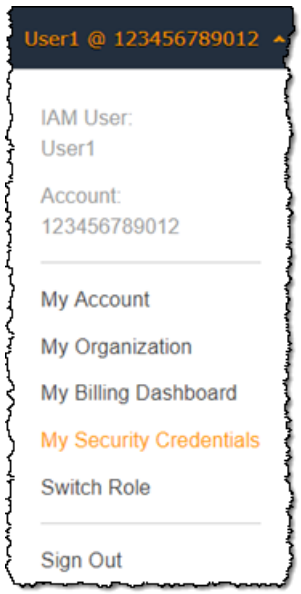
有关将 CodeCommit 与 Amazon Virtual Private Cloud 一起使用的信息，请参阅AWS CodeCommit 与接口 VPC 终端节点一起使用。

查看和管理您的凭证

您可以在 AWS 控制台中通过我的安全凭证查看和管理您的 CodeCommit 凭证。

> **Note**
>
> 此选项不适用于使用联合访问权限、临时证书或 Web 身份提供商的用户。

1. 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/。
2. 在右上角的导航栏中，选择您的用户名，然后选择我的安全凭证。



3. 选择 AWS CodeCommit 凭证选项卡。

使用 Git 凭证进行设置

利用 HTTPS 连接和 Git 凭证，您可以在 IAM 中生成静态用户名和密码。然后，您可以在 Git 和支持 Git 用户名及密码身份验证的任何第三方工具中使用这些凭证。大多数 IDE 和开发工具都支持这种方法。它是配合 CodeCommit 使用时最简单、最轻松的连接方法。

- 适用于使用 Git 凭证的 HTTPS 用户：按照以下说明使用 Git 凭证设置本地计算机与 CodeCommit 存储库之间的连接。
- 从开发工具进行的连接：按照以下准则使用 Git 凭证设置 IDE 或其他开发工具与 CodeCommit 存储库之间的连接。支持 Git 凭证的 IDE 包括 (但不限于) Visual Studio、Eclipse、Xcode 和 IntelliJ。

使用其他方法进行设置

您可以使用 SSH 协议而不是 HTTPS 来连接 CodeCommit 存储库。借助 SSH 连接，您可以在本地计算机上创建公有和私有密钥文件，以供 Git 和 CodeCommit 进行 SSH 身份验证。您将公有密钥关联到 IAM 用户，并将私有密钥存储在本地计算机上。SSH 需要手动创建和管理公有密钥和私有密钥文件，因此，您可能会发现配合 Git 凭证来使用 CodeCommit 会更简单、更容易。

与 Git 凭证不同，SSH 连接设置因本地计算机上的操作系统而异。

- 适用于不使用 AWS CLI 的 SSH 用户：如果您已有公有-私有密钥对并且很熟悉您本地计算机上的 SSH 连接，请按照这些简要说明操作。

- 适用于 Linux、macOS 或 Unix 上的 SSH 连接：请按照这些分步演练说明在 Linux、macOS 或 Unix 操作系统上创建公有密钥/私有密钥对并设置连接。
- 适用于 Windows 上的 SSH 连接：请按照这些分步演练说明在 Windows 操作系统上创建公有密钥/私有密钥对并设置连接。

如果要使用联合访问、身份提供程序或临时凭证连接到 CodeCommit 和 AWS，或者您不想为用户配置 IAM 用户或为 IAM 用户配置 Git 凭证，则可以通过以下两种方式之一建立与 CodeCommit 存储库的连接：

- 安装和使用 git-remote-codecommit（推荐）。
- 安装并使用 AWS CLI 中包含的凭证辅助程序。

这两种方法都支持无需 IAM 用户即可访问 CodeCommit 存储库，这意味着您可以使用联合访问和临时凭证连接到存储库。git-remote-codecommit 实用程序是推荐采用的方法。它扩展 Git，并与各种 Git 版本和凭证辅助程序兼容。但是，并非所有 IDE 都支持 `git-remote-codecommit` 使用的克隆 URL 格式。您可能需要手动将存储库克隆到本地计算机，然后才能在 IDE 中使用它们。

- 请按照使用 git-remote-codecommit 设置到 AWS CodeCommit 存储库的 HTTPS 连接的步骤中的说明在 Windows、Linux、macOS 或 Unix 上安装和设置 git-remote-codecommit。

每当 Git 需要使用 AWS 进行身份验证来与 CodeCommit 存储库交互时，AWS CLI 中包含的凭证助手允许 Git 使用 HTTPS 和 IAM 用户凭证或 Amazon EC2 实例角色的加密签名版本。某些操作系统和 Git 版本有自己的凭证辅助程序，它们会与 AWS CLI 中包含的凭证辅助程序发生冲突。它们可能导致 CodeCommit 连接问题。

- 适用于在 Linux、macOS 或 Unix 上使用 AWS CLI 凭证助手进行 HTTPS 连接：按照这些分步演练说明在 Linux、macOS 或 Unix 系统上安装和设置凭证助手。
- 适用于在 Windows 上使用 AWS CLI 凭证助手进行 HTTPS 连接：按照这些分步演练说明在 Windows 系统上安装和设置凭证辅助程序。

如果您正在连接到在其他 Amazon Web Services 账户中托管的 CodeCommit 存储库，可以使用 AWS CLI 附带的角色、策略和凭证助手来配置访问权限和设置连接。

- 使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限：按照以下说明完成分步演练，来配置在一个 Amazon Web Services 账户中对另一个 Amazon Web Services 账户的 IAM 组中的用户执行跨账户访问。

CodeCommit、Git 及其他组件的兼容性

在使用 CodeCommit, 时，可使用 Git。您也可以使用其他程序。下表提供了有关版本兼容性的最新指南。作为最佳做法，我们建议您使用最新版本的 Git 和其他软件。

AWS CodeCommit 的版本兼容性信息

组件	版本
Git	CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。
Curl	CodeCommit 需要 curl 7.33 及更高版本。但 HTTPS 和 curl 更新 7.41.0 存在一个已知问题。有关更多信息，请参阅故障排除。
Python（仅限 git-remote-codecommit）	git-remote-codecommit 需要版本 3 及更高版本。
Pip（仅限 git-remote-codecommit）	git-remote-codecommit 需要 9.0.3 及更高版本。
AWS CLI（仅 git-remote-codecommit）	我们建议所有 CodeCommit 用户使用最新版本的 AWS CLI 的版本 2。git-remote-codecommit 需要 AWS CLI 版本 2 才能支持 AWS SSO 和需要临时凭证的连接，例如联合用户。

适用于使用 Git 凭证的 HTTPS 用户的设置

设置 AWS CodeCommit 存储库连接的最简单方法是在 IAM 控制台 CodeCommit 中配置 Git 证书，然后使用这些证书进行 HTTPS 连接。您还可以将这些凭证用于支持使用静态用户名和密码进行 HTTPS 身份验证的任何第三方工具或集成式开发环境 (IDE)。有关示例，请参阅从开发工具进行的连接。

> **Note**
>
> 如果您之前已将本地计算机配置为使用凭据助手 CodeCommit，则必须先编辑您的.gitconfig 文件以从文件中删除凭证帮助程序信息，然后才能使用 Git 凭据。如果您的本地计算机运行的是 macOS，则您可能需要通过 Keychain Access 清除缓存的凭证。

步骤 1：的初始配置 CodeCommit

按照以下步骤设置 Amazon Web Services 账户、创建 IAM 用户并配置访问权限 CodeCommit。

创建和配置用于访问的 IAM 用户 CodeCommit

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

   > **Note**
   >
   > CodeCommit 要求 AWS Key Management Service。如果您使用的是现有的 IAM 用户，请确保该用户没有明确拒绝所要求的 AWS KMS 操作的策略 CodeCommit。有关更多信息，请参阅 AWS KMS 和加密。

3. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置 CodeCommit 访问权限的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中，选择AWSCodeCommitPowerUser或其他托管策略进行 CodeCommit 访问。有关更多信息，请参阅 适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他群组和用户共享仓库访问权限的更多信息，请参阅共享存储库和AWS CodeCommit 的身份验证和访问控制。

如果要在中使用 AWS CLI 命令 CodeCommit，请安装 AWS CLI。我们建议您创建配置文件以便 AWS CLI 与一起使用 CodeCommit。有关更多信息，请参阅命令行参考和使用命名配置文件。

步骤 2：安装 Git

要处理 CodeCommit 存储库中的文件、提交和其他信息，必须在本地计算机上安装 Git。CodeCommit 支持 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议您访问 Git 下载等网站。

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能更改可能会影响其工作方式 CodeCommit。如果您在使用特定版本的 Git 时遇到问题 CodeCommit，请查看中的信息故障排除。

步骤 3：创建 Git 凭据，以便通过 HTTPS 连接到 CodeCommit

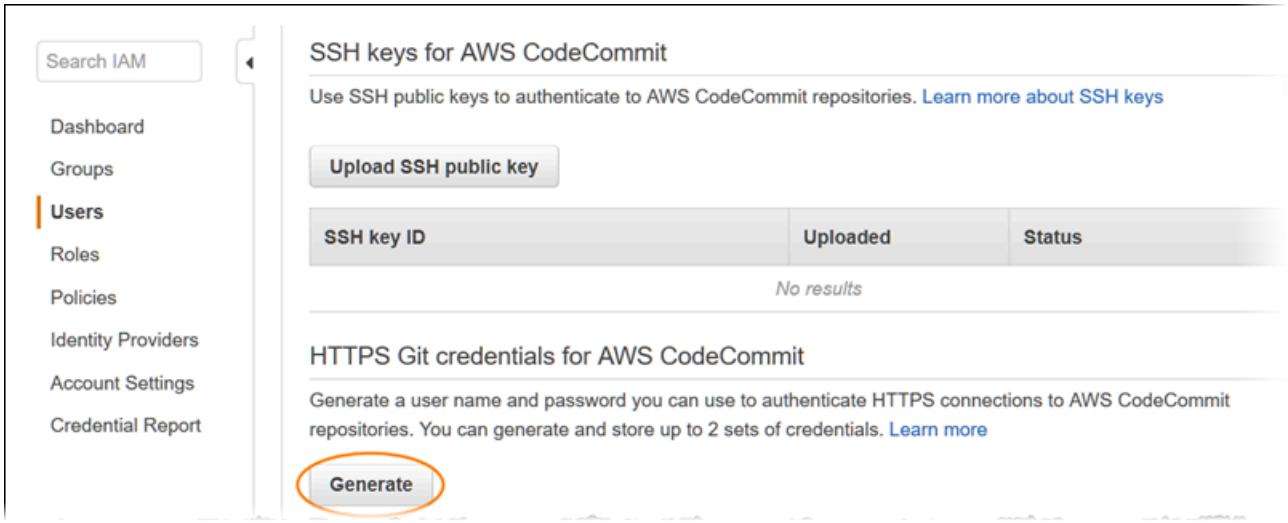
安装 Git 后，在 IAM 中为您的 IAM 用户创建 Git 凭证。

要为设置 HTTPS Git 凭证 CodeCommit

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。

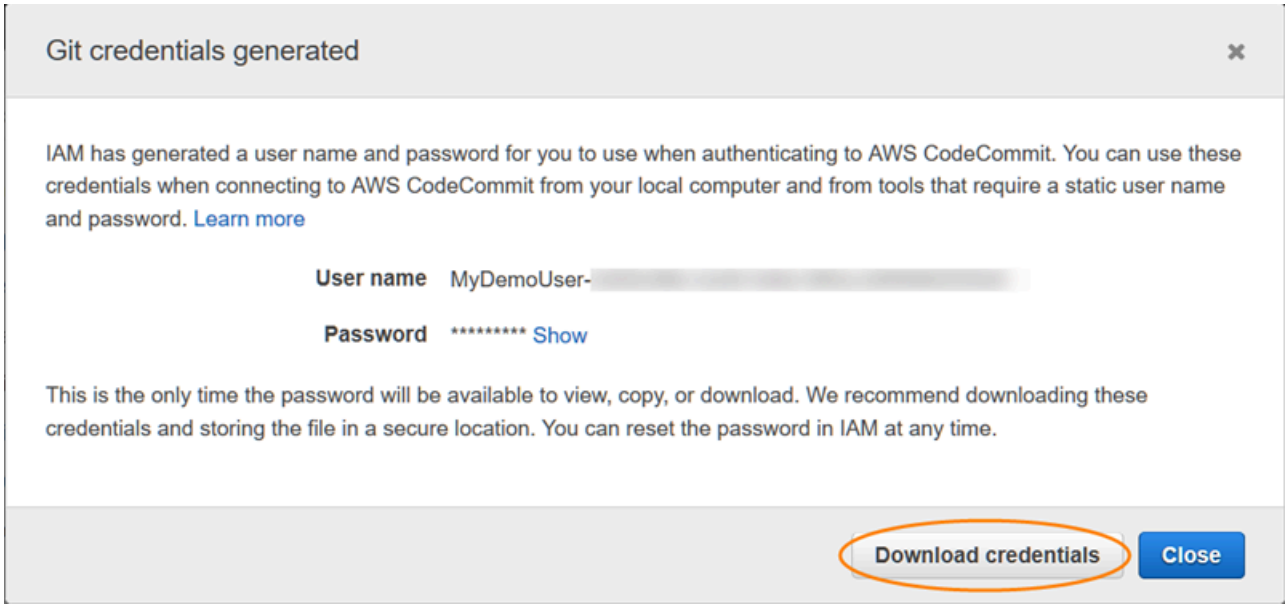
   请务必以 IAM 用户身份登录，该用户将创建并使用 Git 证书进行连接 CodeCommit。
2. 在 IAM 控制台的导航窗格中，选择用户，然后从用户列表中选择您的 IAM 用户。

   > **Note**
   >
   > 您可以在 “我的安全 CodeCommit 证书” 中直接查看和管理您的证书。有关更多信息，请参阅 查看和管理您的凭证。
3. 在用户详细信息页面上，选择安全证书选项卡，然后在的 HTTPS Git 凭据中 AWS CodeCommit，选择生成。



> **Note**
>
> 您无法为 Git 凭证选择自己的用户名或密码。有关更多信息，请参阅将 Git 凭据和 HTTPS 与一起使用 CodeCommit。

4. 通过以下两种方式之一复制 IAM 为您生成的用户名和密码：显示这些信息，然后将其复制并粘贴到本地计算机上安全的文件中；或选择下载凭证)，将这些信息下载为 .CSV 文件。您需要此信息才能连接 CodeCommit。



保存您的凭证后，选择 Close。

> **Important**
>
> 这是您保存该用户名和密码的唯一机会。如果您未保存它们，可以从 IAM 控制台复制用户名，但无法查找密码。此时，您必须重置密码，然后保存它。

步骤 4：Connect 连接到 CodeCommit 控制台并克隆存储库

如果管理员已经向您发送了 CodeCommit存储库的名称和连接详细信息，则可以跳过此步骤直接克隆存储库。

连接到存储 CodeCommit 库

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择存储库的创建 AWS 区域 位置。存储库特定于 AWS 区域。有关更多信息，请参阅 区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。
   - 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

   > **Note**
   >
   > 如果您看到的是欢迎页面而不是存储库列表，则说明您登录的 AWS 区域 位置中没有与您的 AWS 账户关联的存储库。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 打开终端、命令行或 Git shell。使用复制的 HTTPS 克隆 URL 运行 git clone 命令以克隆存储库。例如，要将名为的存储库克隆*MyDemoRepo*到*my-demo-repo*位于美国东部（俄亥俄州）地区的本地存储库，请执行以下操作：

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

在第一次连接时，系统会提示您提供该存储库的用户名和密码。根据本地计算机的配置，操作系统的凭证管理系统、您的 Git 版本的凭证管理器实用程序（例如，Windows 版 Git 中包含的 Git Credential Manager）、您的 IDE 或 Git 本身都可能会产生该提示。输入在 IAM 中为 Git 凭证（您在步骤 3：创建 Git 凭据，以便通过 HTTPS 连接到 CodeCommit中创建的凭证）生成的用户名和密码。根据您的操作系统及其他软件，该信息可能保存在凭证存储或凭证管理实用程序中。如果是这样，除非您在 IAM 中更改密码、停用 Git 凭证或删除 Git 凭证，否则应该不会再出现提示。

如果您没有在本地计算机上配置凭证存储或凭证管理实用程序，则可以安装一个。有关 Git 及其管理凭证的更多信息，请参阅 Git 文档中的凭证存储。

有关更多信息，请参阅 通过克隆 CodeCommit 存储库来连接存储库 和 创建提交。

后续步骤

您已满足先决条件。按照中的步骤入门 CodeCommit 开始使用 CodeCommit。

要了解如何创建和推送您的第一个提交，请参阅在中创建提交 AWS CodeCommit。如果您刚刚接触 Git，您可能还需要查看哪里可以找到有关 Git 的更多信息？和Git 和 AWS CodeCommit 入门中的信息。

使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤

如果要使用根账户、联合访问或临时凭证连接到 CodeCommit，则应使用 git-remote-codecommit 设置访问权限。此实用程序提供一种通过扩展 Git 从 CodeCommit 存储库中推送和拉取代码的简单方法。这是支持使用联合访问、身份提供程序和临时凭证建立连接的推荐方法。要向联合身份分配权限，您可以创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。

您还可以将 git-remote-codecommit 与 IAM 用户结合使用。与其他 HTTPS 连接方法不同，git-remote-codecommit 不要求为用户设置 Git 凭证。

> **Note**
>
> 某些 IDE 不支持 `git-remote-codecommit` 使用的克隆 URL 格式。您可能需要手动将存储库克隆到本地计算机，然后才能在您首选的 IDE 中使用它们。有关更多信息，请参阅 git-remote-codecommit 和 AWS CodeCommit 疑难解答。

编写这些过程时，假设您有一个 Amazon Web Services 账户，在 CodeCommit 中创建了至少一个存储库，并在连接到 CodeCommit 存储库时使用具有托管策略的 IAM 用户。有关如何为联合身份用户和其他轮换凭证类型配置访问权限的信息，请参阅使用轮换凭证连接到 AWS CodeCommit 存储库。

主题

步骤 0：安装 git-remote-codecommit 的先决条件

您必须在本地计算机上安装一些先决条件，然后才能使用 git-remote-codecommit。其中包括：

- Python（版本 3 或更高版本）及其程序包管理器 pip（如果尚未安装）。要下载并安装 Python 的最新版本，请访问 Python 网站。
- Git

> **Note**
>
> 在 Windows 上安装 Python 时，请确保选择了将 Python 添加到路径的选项。

git-remote-codecommit 需要 pip 版本 9.0.3 或更高版本。要检查您的 pip 版本，请打开终端或命令行并运行以下命令：

```
pip --version
```

您可以运行以下两个命令将您的 pip 版本更新到最新版本：

```
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
```

要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议您访问 Git 下载等网站。

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

步骤 1：CodeCommit 的初始配置

请按照以下步骤创建 IAM 用户、使用适当的策略对其进行配置、获取访问密钥和秘密密钥，以及安装和配置 AWS CLI。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅 AWS KMS 和加密。

3. 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/。

4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅 适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

安装和配置 AWS CLI

1. 在本地计算机上，下载并安装 AWS CLI。这是从命令行与 CodeCommit 进行交互的先决条件。我们建议您安装 AWS CLI 版本 2。它是 AWS CLI 的最新主版本，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书与 git-remote-codecommit 的 AWS CLI 版本。

   有关更多信息，请参阅使用 AWS 命令行界面进行设置。

   > **Note**
   >
   > CodeCommit 仅适用于 AWS CLI 1.7.38 及更高版本。作为最佳做法，请安装 AWS CLI 或将其升级到可用的最新版本。要确定您安装的 AWS CLI 的版本，请运行 aws --version 命令。
   > 要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证 AWS CLI 中的 CodeCommit 命令是否已安装：

   ```
   aws codecommit help
   ```

   该命令返回 CodeCommit 命令的列表。

3. 使用 configure 命令通过配置文件来配置 AWS CLI，如下所示：

   ```
   aws configure
   ```

出现提示时，指定要用于 CodeCommit 的 IAM 用户的 AWS 访问密钥和 AWS 秘密访问密钥。此外，请务必指定存储库所在的 AWS 区域，例如 us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

有关创建和配置相关配置文件以与 AWS CLI 配合使用的详细信息，请参阅以下内容：

- 命名配置文件
- 在 AWS CLI 中使用 IAM 角色
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到另一个 AWS 区域中的存储库或资源，您必须使用默认区域名称重新配置 AWS CLI。CodeCommit 支持的默认区域名称包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1

- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

有关 CodeCommit 和 AWS 区域的更多信息，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

步骤 2：安装 git-remote-codecommit

请按照以下步骤安装 git-remote-codecommit。

安装 git-remote-codecommit

1. 在终端或命令行中，运行以下命令：

```
pip install git-remote-codecommit
```

> **Note**
>
> 根据操作系统和配置的不同，您可能需要以提升权限（例如 sudo）运行此命令，或使用 --user 参数安装到不需要特殊权限的目录，例如您的当前用户账户。例如，在运行 Linux、macOS 或 Unix 的计算机上：
>
> ```
> sudo pip install git-remote-codecommit
> ```
>
> 在运行 Windows 的计算机上：
>
> ```
> pip install --user git-remote-codecommit
> ```

2. 监控安装过程，直到您看到类似于以下内容的成功消息。

步骤 3：连接 CodeCommit 控制台并克隆存储库

如果管理员已经向您发送了要与 git-remote-codecommit 结合使用的 CodeCommit 存储库克隆 URL，则可以跳过连接到控制台的操作，直接克隆存储库。

连接 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅 区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。
   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。
   - 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

> **Note**
>
> 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 在终端或命令提示符处，使用 git clone 命令克隆存储库。请使用您复制的 HTTPS git-remote-codecommit URL 以及 AWS CLI 配置文件的名称（如果您创建了命名配置文件）。如果未指定配置文件，则该命令将采用默认配置文件。这将在运行命令的目录的子目录中创建本地存储库。例如，要将名为 *MyDemoRepo* 的存储库克隆成名为 *my-demo-repo* 的本地存储库，请运行以下命令：

```
git clone codecommit://MyDemoRepo my-demo-repo
```

要使用名为 *CodeCommitProfile* 的配置文件来克隆相同的存储库：

```
git clone codecommit://CodeCommitProfile@MyDemoRepo my-demo-repo
```

要在与配置文件中配置的 AWS 区域不同的区域中克隆存储库，请包含 AWS 区域名称。例如：

```
git clone codecommit::ap-northeast-1://MyDemoRepo my-demo-repo
```

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

要了解如何创建和推送您的第一个提交，请参阅在中创建提交 AWS CodeCommit。如果您刚刚接触 Git，您可能还需要查看哪里可以找到有关 Git 的更多信息？和Git 和 AWS CodeCommit 入门中的信息。

使用 Git 凭证从开发工具设置连接

在 IAM 控制台 AWS CodeCommit 中为配置了 Git 凭证后，您可以将这些证书与任何支持 Git 证书的开发工具一起使用。例如，您可以在 Visual Studio、Eclipse AWS Cloud9、Xcode、IntelliJ 或任何集成

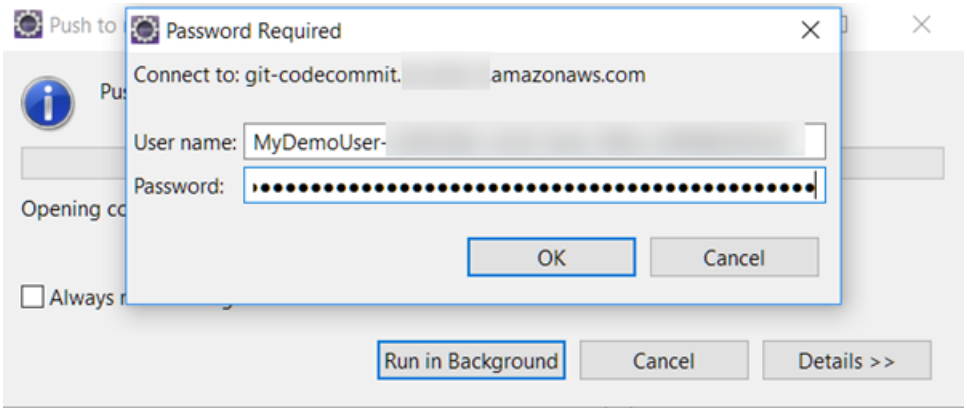
Git 凭据的集成开发环境 (IDE) 中配置对 CodeCommit存储库的访问权限。配置访问后，您可以编辑代码、提交更改并从 IDE 或其他开发工具直接推送。

> **Note**
>
> 如果您使用联合访问、临时证书或 Web 身份提供程序访问 CodeCommit 存储库，则无法使用 Git 凭证。我们建议您使用 `git-remote-codecommit` 命令设置本地计算机。但是，并非所有 IDE 都与 Git 远程辅助程序（如 git-remote-codecommit）完全兼容。如果遇到问题，请参阅git-remote-codecommit 和 AWS CodeCommit 疑难解答。

主题

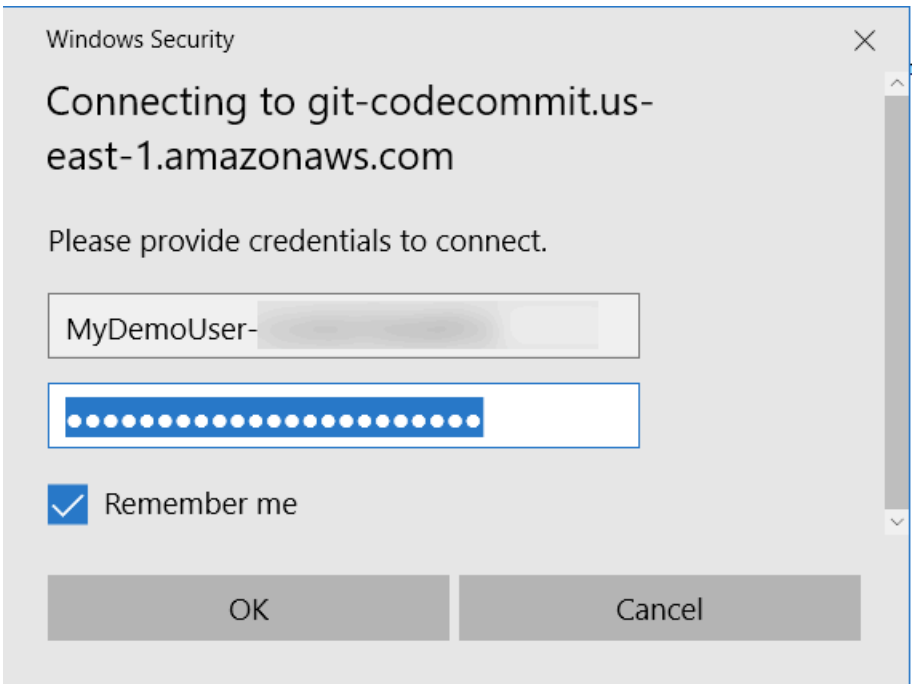
- 将 AWS Cloud9 与 AWS CodeCommit 集成
- 将 Visual Studio 与 AWS CodeCommit 集成
- 将 Eclipse 与 AWS CodeCommit 集成

当 IDE 或开发工具提示您输入用于连接 CodeCommit存储库的用户名和密码时，请提供您在 IAM 中创建的用户名和密码的 Git 凭证。例如，如果系统提示您在 Eclipse 中输入用户名和密码，您将提供自己的 Git 凭证，如下所示：



有关 AWS 区域 和终端节点的更多信息 CodeCommit，请参阅区域和 Git 连接端点。

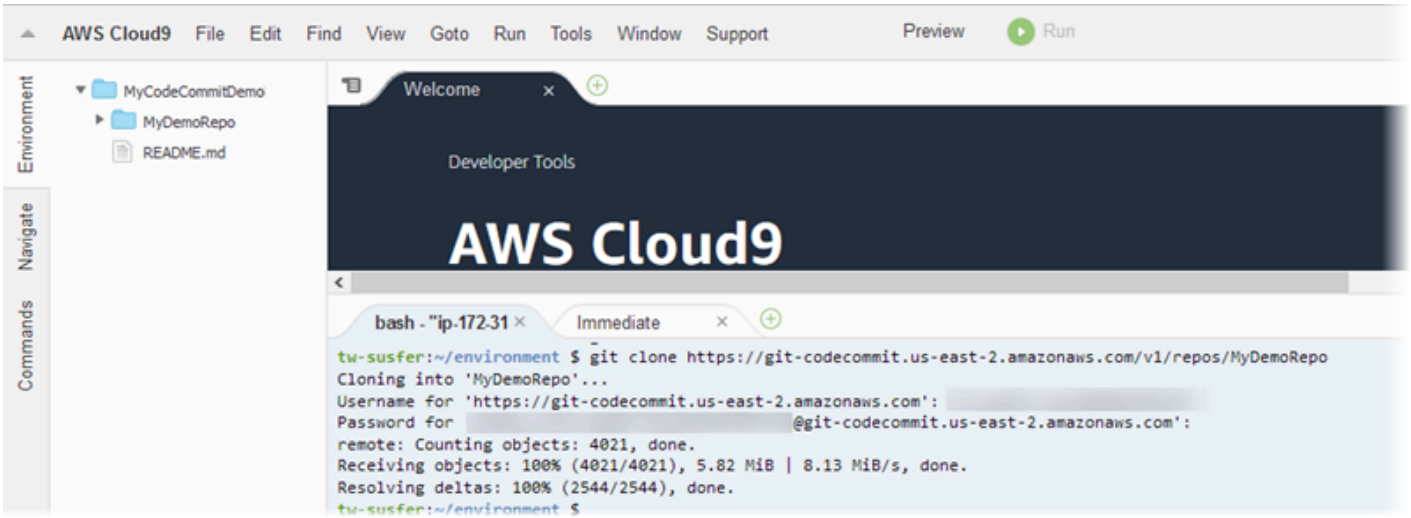
您可能还会看到操作系统提示存储用户名和密码。例如，在 Windows 中，应如下所示提供 Git 凭证：



有关为特定软件程序或开发工具配置 Git 凭证的信息，请参阅产品文档。

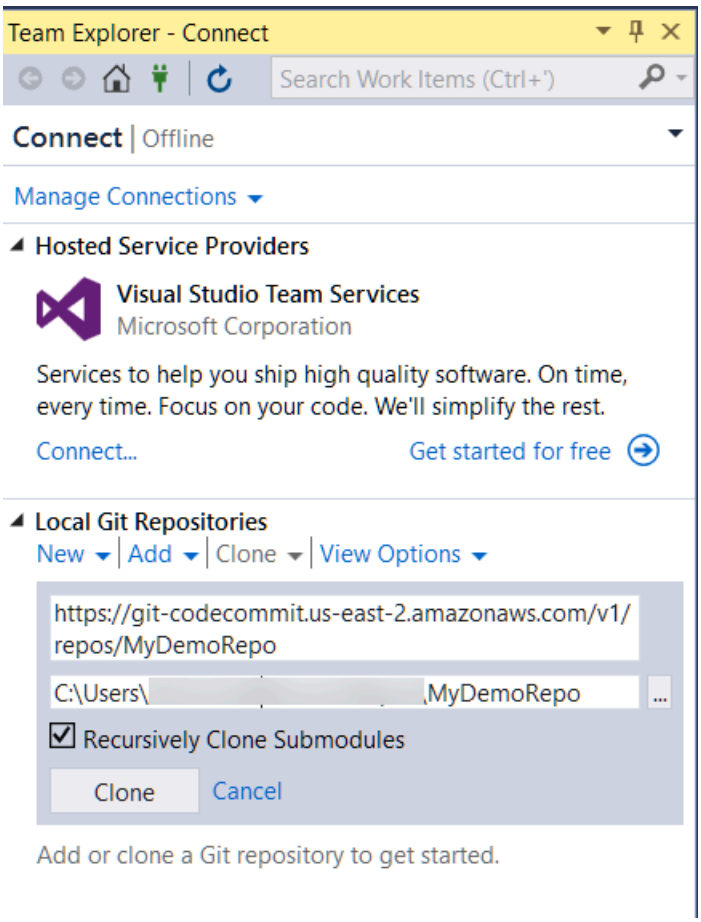
下面是不全面的 IDE 列表。提供这些链接只是为了帮助您了解有关这些工具的更多信息。 AWS 对这些主题中的任何内容概不负责。

- AWS Cloud9



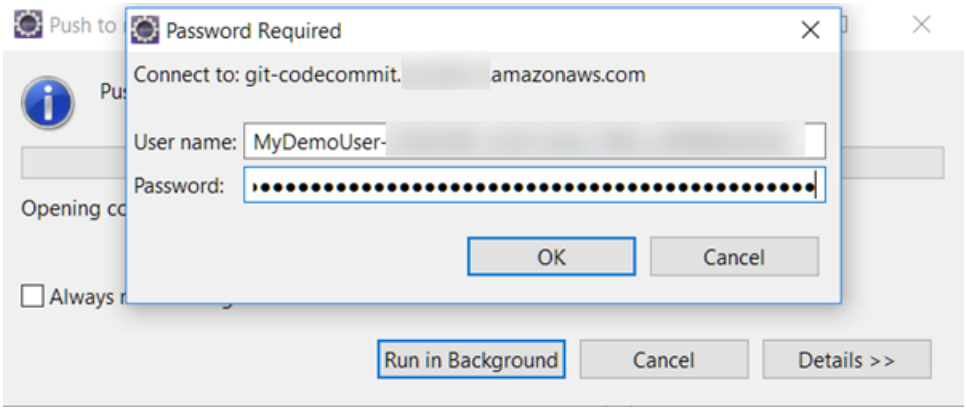
- Visual Studio

  或者，也可以安装 AWS Toolkit for Visual Studio。有关更多信息，请参阅 将 Visual Studio 与 AWS CodeCommit 集成。



- EGit with Eclipse

  或者，也可以安装 AWS Toolkit for Eclipse。有关更多信息，请参阅 将 Eclipse 与 AWS CodeCommit 集成。



- XCode

将 AWS Cloud9 与 AWS CodeCommit 集成

您可以使用 AWS Cloud9 在 CodeCommit 存储库中进行代码更改。AWS Cloud9 包含一套工具，可用于编写代码以及构建、运行、测试、调试和发布软件。您可以克隆现有存储库、创建存储库、向存储库提交和推送代码更改等，所有这些操作都在 AWS Cloud9 EC2 开发环境中完成。AWS Cloud9 EC2 开发环境通常已使用 AWS CLI、Amazon EC2 角色和 Git 进行预配置，因此，在大多数情况下，您可以运行几条简单命令并开始与存储库进行交互。

要将 AWS Cloud9 与 CodeCommit 结合使用，您需要：

- 一个在 Amazon Linux 上运行的 AWS Cloud9 EC2 开发环境。
- 在 Web 浏览器中打开的 AWS Cloud9 IDE。
- 一个 IAM 用户，该用户具有一个 CodeCommit 托管策略并且已应用一个 AWS Cloud9 托管策略。

  有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略和了解并获取您的安全凭证。

> **Note**
>
> 本主题介绍如何设置 CodeCommit 和 AWS Cloud9 的集成，以便从互联网进行一般访问。您可以在隔离的环境中设置对 CodeCommit 和 AWS Cloud9 的访问，但这需要额外的步骤。有关更多信息，请参阅：
>
> - AWS CodeCommit 与接口 VPC 终端节点一起使用
> - 使用 AWS Systems Manager 访问非摄取 Amazon EC2 实例
> - 使用共享环境
> - 与其他账户共享 VPC
> - 博客文章：隔离对 AWS Cloud9 环境的网络访问

主题

步骤 1：创建 AWS Cloud9 开发环境

AWS Cloud9 将在 Amazon EC2 实例上托管开发环境。这是最简单的集成方法，因为您可以使用实例的 AWS 托管临时凭证来连接到您的 CodeCommit 存储库。如果您想改用自己的服务器，请参阅 AWS Cloud9 用户指南。

创建 AWS Cloud9 环境

1. 以您配置的 IAM 用户身份登录 AWS 并打开 AWS Cloud9 控制台。
2. 在 AWS Cloud9 控制台中，选择创建环境。
3. 在步骤 1: 命名环境中，输入环境的名称和可选描述，然后选择下一步。
4. 在步骤 2: 配置设置中，配置您的环境，如下所示：
   - 在 Environment type 中，选择 Create a new instance for environment (EC2)。
   - 在 Instance type 中，为开发环境选择适当的实例类型。例如，如果您只探索该服务，则可以选择默认值 t2.micro。如果您打算将此环境用于开发工作，请选择更大的实例类型。
   - 接受其他默认设置；除非您出于特定原因不这样做（例如，您的组织使用特定 VPC，或者您的 Amazon Web Services 账户未配置任何 VPC），然后选择下一步。
5. 在步骤 3: 审核 中，审查您的设置。如果您需要进行任何更改，请选择 Previous step。否则，请选择 Create environment。

   创建一个环境，首次连接到此环境需要几分钟的时间。如果连接所花时间似乎太长，请参阅《AWS Cloud9 用户指南》中的故障排除。
6. 在连接到您的环境后，请检查是否已安装 Git，并通过在终端窗口中运行 git --version 命令查看它是否为受支持的版本。

   如果未安装 Git，或者 Git 不是受支持的版本，请安装受支持的版本。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。要安装 Git，建议您访问 Git 下载等网站。

> **Tip**
>
> 根据环境的操作系统，您也许能够使用带 sudo 选项的 yum 命令安装更新，包括 Git。例如，管理命令序列可能与以下三条命令类似：
>
> ```
> sudo yum -y update
> sudo yum -y install git
> ```

```
git --version
```

7. 通过运行 git config 命令配置要与 Git 提交关联的用户名和电子邮件。例如：

```
git config --global user.name "Mary Major"
git config --global user.email mary.major@example.com
```

步骤 2：在 AWS Cloud9 EC2 开发环境中配置 AWS CLI 凭证助手

在创建 AWS Cloud9 环境后，您可以配置 AWS CLI 凭证助手来管理与 CodeCommit 存储库的连接凭证。AWS Cloud9 开发环境附带了与 IAM 用户关联的 AWS 托管临时凭证。您可将这些凭证与 AWS CLI 凭证辅助程序一起使用。

1. 打开终端窗口并运行以下命令来验证是否安装了 AWS CLI：

```
aws --version
```

如果成功，此命令将返回当前安装的版本的 AWS CLI。要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 在终端上，运行以下命令来配置 AWS CLI 凭证辅助程序以进行 HTTPS 连接：

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 该凭证助手将为开发环境使用默认的 Amazon EC2 实例角色。如果您打算使用开发环境连接到未在 CodeCommit 中托管的存储库，请配置与这些存储库的 SSH 连接，或配置本地 `.gitconfig` 文件以在连接到其他存储库时使用备用凭证管理系统。有关更多信息，请参阅 Git 网站上的 Git 工具 - 凭证存储。

步骤 3：将 CodeCommit 存储库克隆到 AWS Cloud9 EC2 开发环境中

在配置 AWS CLI 凭证助手后，您可以在其上克隆您的 CodeCommit 存储库。然后，您便可以开始使用代码。

1. 在终端，运行 git clone 命令，并指定要克隆的存储库的 HTTPS 克隆 URL。例如，如果您需要在美国东部（俄亥俄州）区域中克隆名为 MyDemoRepo 的存储库，可输入：

   ```
   git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   ```

   > **Tip**
   >
   > 您可以通过在 CodeCommit 控制台中选择克隆 URL找到仓库的克隆 URL。

2. 在克隆完成后，在侧面导航中展开存储库的文件夹，然后选择要打开的文件以进行编辑。或者，选择文件，然后选择新文件以创建文件。
3. 在完成文件的编辑和创建后，请在终端窗口中，将目录更改为您的克隆存储库，然后提交和推送更改。例如，如果您添加了一个名为 *MyFile.py* 的新文件，则：

   ```
   cd MyDemoRepo
   git commit -a MyFile.py
   git commit -m "Added a new file with some code improvements"
   git push
   ```

后续步骤

有关更多信息，请参阅 AWS Cloud9 用户指南和用于 AWS Cloud9 的 CodeCommit 示例。有关将 Git 与 CodeCommit 结合使用的更多信息，请参阅 Git 和 AWS CodeCommit 入门。

将 Visual Studio 与 AWS CodeCommit 集成

您可以使用 Visual Studio 在 CodeCommit 存储库中进行代码更改。AWS Toolkit for Visual Studio 现在包含的功能可让您在 Visual Studio 中工作时更轻松、更方便地使用 CodeCommit。Toolkit for Visual Studio 集成可以与 Git 凭证和 IAM 用户结合使用。您可以克隆现有存储库、创建存储库、向存储库提交和推送代码更改等。

> **Important**
>
> Toolkit for Visual Studio 只能在 Windows 操作系统上安装。如果您要寻找有关使用 Visual Studio Code 的信息，请参阅 AWS Toolkit for Visual Studio Code。

如果您之前使用过 Toolkit for Visual Studio，可能已经熟悉如何设置包含访问密钥和秘密密钥的 AWS 凭证配置文件。凭证配置文件在 Toolkit for Visual Studio 中用于启用对 AWS 服务 API 的调用（例如，供 Amazon S3 列出桶或供 CodeCommit 列出存储库）。要向 CodeCommit 存储库中拉取和推送代码，您还需要 Git 凭证。如果您没有 Git 凭证，Toolkit for Visual Studio 可以为您生成并应用这些凭证。这可为您节省大量时间。

要使用 Visual Studio 和 CodeCommit，您需要：

- 已配置一组有效凭证（访问密钥和秘密密钥）的 IAM 用户。此 IAM 用户还应该满足以下条件：

  已对其应用某个 CodeCommit 托管策略以及 IAMSelfManageServiceSpecificCredentials 托管策略。

  或

  如果 IAM 用户已配置 Git 凭证，则需要一个 CodeCommit 托管策略或等效权限。

  有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略和了解并获取您的安全凭证。
- AWS Toolkit for Visual Studio 安装在您已安装了 Visual Studio 的计算机上。有关更多信息，请参阅设置 AWS Toolkit for Visual Studio。

有关使用 AWS Toolkit for Visual Studio 和 CodeCommit 的更多信息，请参阅 《Toolkit for Visual Studio 用户指南》中的使用 AWS CodeCommit 和 Visual Studio Team Explorer。

将 Eclipse 与 AWS CodeCommit 集成

您可以使用 Eclipse 在 CodeCommit 存储库中进行代码更改。Toolkit for Eclipse 集成可以与 Git 凭证和 IAM 用户结合使用。您可以克隆现有存储库、创建存储库、向存储库提交和推送代码更改等。

要将 Toolkit for Eclipse 与 CodeCommit 结合使用，您需要：

- 在本地计算机上安装 Eclipse。
- 已配置一组有效凭证（访问密钥和秘密密钥）的 IAM 用户。此 IAM 用户还应该满足以下条件：

  已对其应用某个 CodeCommit 托管策略以及 IAMSelfManageServiceSpecificCredentials 托管策略。

或

如果 IAM 用户已配置 Git 凭证，则需要一个 CodeCommit 托管策略或等效权限。

有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略和了解并获取您的安全凭证。

- 在 IAM 中为用户配置一套有效的 Git 凭证。有关更多信息，请参阅步骤 3：创建 Git 凭据，以便通过 HTTPS 连接到 CodeCommit。

主题

步骤 1：为您的 IAM 用户获取访问密钥和秘密密钥

如果已安装 Eclipse 的计算机上尚未设置凭证配置文件，您可以使用 AWS CLI 和 aws configure 命令配置一个。或者，您可以执行本过程中的步骤来创建和下载您的凭证。出现提示时，将其提供给 Toolkit for Eclipse。

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
员工身份 （在 IAM Identity Center 中管理的用户）	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中

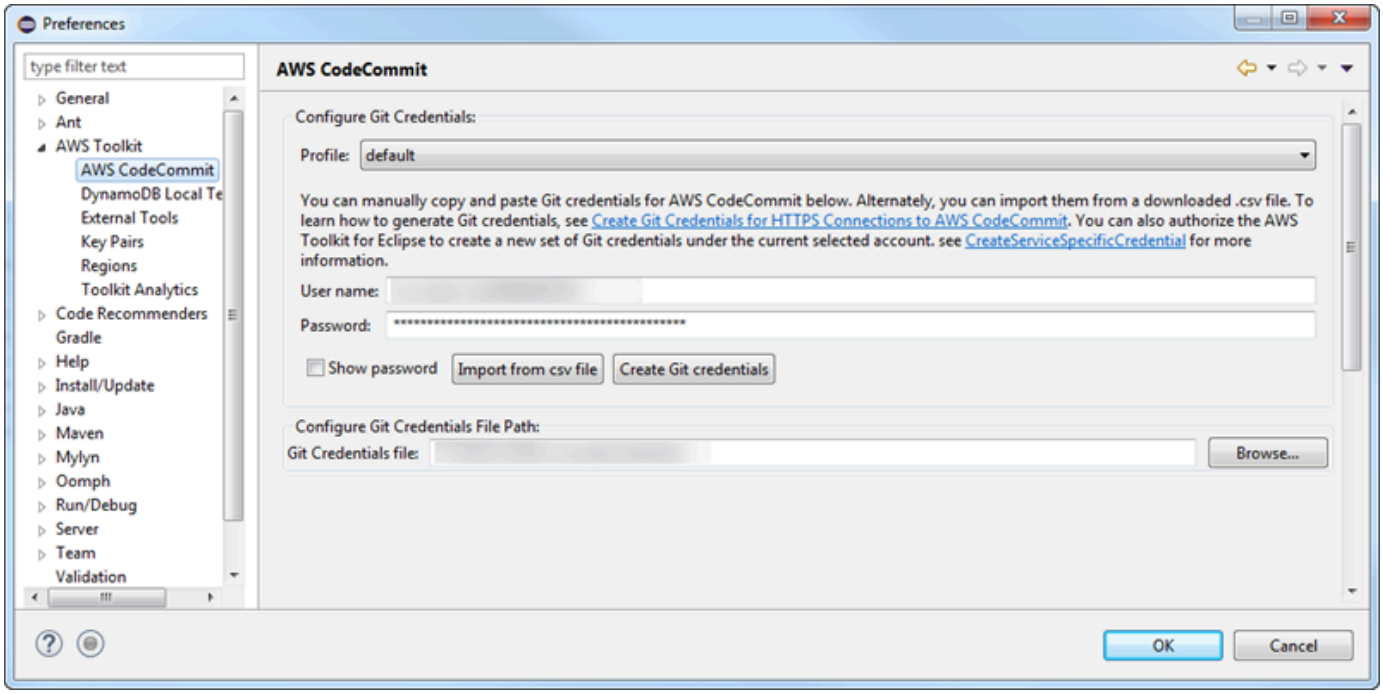
哪个用户需要编程式访问权限？	目的	方式
		的配置 AWS CLI 以使用 AWS IAM Identity Center。 • 有关 AWS SDK、工具和 AWS API 的更多信息，请参阅《AWS SDK 和工具参考指南》中的 IAM Identity Center 身份验证。
IAM	使用临时凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、AWS SDK 或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。 • 有关 AWS SDK 和工具的更多信息，请参阅《AWS SDK 和工具参考指南》中的使用长期凭证进行身份验证。 • 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。

步骤 2：安装 AWS Toolkit for Eclipse 并连接到 CodeCommit

Toolkit for Eclipse 是一个可添加到 Eclipse 的软件包。用 AWS 凭证配置文件安装和配置后，您可以在 Eclipse 中从 AWS Explorer 连接到 CodeCommit。

安装包含 AWS CodeCommit 模块的 Toolkit for Eclipse 并配置对项目存储库的访问权限

1. 如果您还没有安装受支持的版本，请将 Toolkit for Eclipse 安装在您的本地计算机上。如果需要更新 Toolkit for Eclipse 的版本，请按照设置 Toolkit 中的说明操作。
2. 在 Eclipse 中，遵循首次运行体验，或从 Eclipse 菜单系统（位置取决于版本和操作系统）打开 Preferences (首选项) 并选择 AWS Toolkit。
3. 请执行下列操作之一：
   - 如果您遵循首次运行经验，请在提示您设置凭证配置文件时提供 AWS 安全凭证。
   - 如果在 Preferences (首选项) 中进行配置，并且已在计算机上配置凭证配置文件，请从 Default Profile (默认配置文件) 中选择它。
   - 如果在 Preferences (首选项) 中进行配置，但看不到要使用的配置文件或列表为空，请选择 Add profile (添加配置文件)。在配置文件详细信息中，输入 IAM 用户的配置文件名称和凭证（访问密钥和秘密密钥），或输入凭证文件的位置。
   - 如果在 Preferences (首选项) 中进行配置并且未设置配置文件，请使用相关链接来注册账户或管理现有 AWS 安全凭证。
4. 在 Eclipse 中，展开AWS工具包菜单并选择 AWS CodeCommit。选择凭证配置文件，然后输入或从 .csv 文件中导入 Git 凭证的用户名和密码。选择 Apply，然后选择 OK。



使用配置文件登录以后，AWS CodeCommit 连接面板会出现在 Team Explorer 中，其中包含用于克隆、创建或注销的选项。选择克隆可将现有 CodeCommit 存储库克隆到本地计算机，这样您就可以开始处理代码。这是最常用的选项。

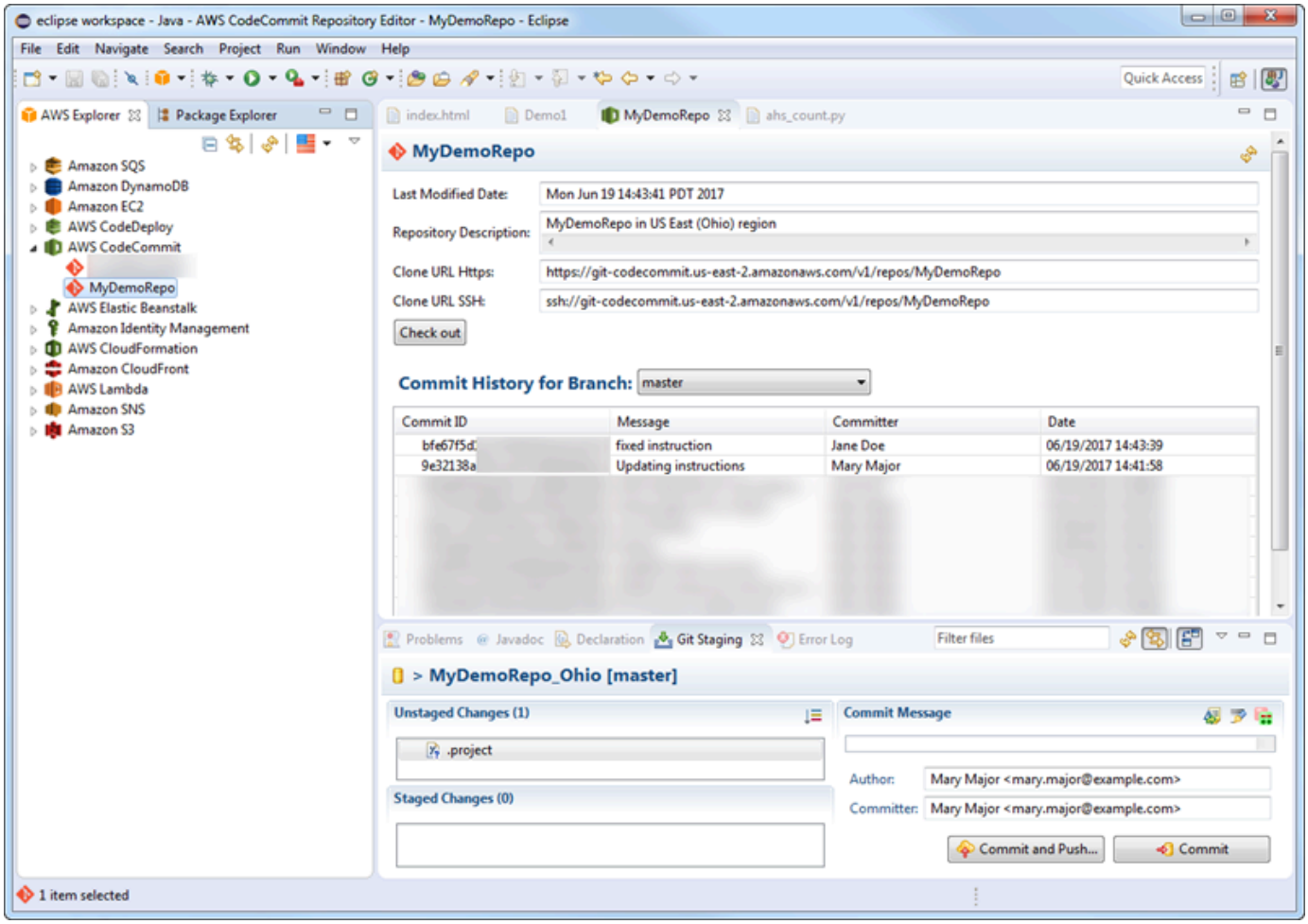
如果没有任何存储库或需要创建存储库，请选择 Create (创建)。

从 Eclipse 中克隆 CodeCommit 存储库

在配置凭证后，通过在 Eclipse 中签出存储库可来将其克隆为计算机本地存储库。然后，您便可以开始使用代码。

1. 在 Eclipse 中，打开 AWS Explorer。有关在何处找到它的更多信息，请参阅如何访问 AWS Explorer。展开 AWS CodeCommit，然后选择要使用的 CodeCommit 存储库。您可以查看存储库的提交历史记录和其他详细信息，这些信息可以帮助您确定这是不是要克隆的存储库和分支。

   > **Note**
   >
   > 如果看不到您的存储库，请选择标记图标以打开 AWS 区域菜单，然后选择创建存储库时所在的 AWS 区域。



2. 选择 Check out (签出)，按照说明进行操作以将存储库克隆到本地计算机。
3. 完成克隆项目后，您就可以开始在 Eclipse 中编辑代码并将更改暂存、提交和推送到 CodeCommit 中的项目存储库。

从 Eclipse 中创建 CodeCommit 存储库

您可以使用 Toolkit for Eclipse 从 Eclipse 中创建 CodeCommit 存储库。在创建存储库的过程中，您还会将其克隆到您计算机上的本地存储库，从而可以立即使用该存储库。

1. 在 AWS Explorer 中，右键单击 AWS CodeCommit，然后选择创建存储库。

> **Note**
>
> 存储库是特定于区域的。在创建存储库之前，请务必选择正确的 AWS 区域。开始存储库创建过程后，将无法选择 AWS 区域。

2. 在 Repository Name (存储库名称) 中，为此存储库输入名称。一个 Amazon Web Services 账户中的存储库名称必须是唯一的。这些名称有字符和长度限制。有关更多信息，请参阅配额。在 Repository Description (存储库描述) 中，输入此存储库的可选描述。这有助于他人了解此存储库的用途，还有助于将其与区域中的其他存储库进行区分。选择 OK（确定）。
3. 在 AWS Explorer 中，展开 AWS CodeCommit，然后选择刚创建的 CodeCommit 存储库。您将看到此存储库没有提交历史记录。选择 Check out (签出)，按照说明进行操作以将存储库克隆到本地计算机。

使用 CodeCommit 存储库

在连接到 CodeCommit 之后，您可以在 AWS Explorer 中看到按 AWS 区域列出与您的账户关联的存储库的列表。选择标志菜单以更改区域。

> **Note**
>
> CodeCommit 可能无法在 Toolkit for Eclipse 支持的所有 AWS 区域中都可用。

在 Toolkit for Eclipse 中，可以从导航和包资源管理器视图浏览这些存储库的内容。要打开文件，请从列表中选择该文件。

Toolkit for Eclipse 中用于 CodeCommit 存储库的 Git 操作的工作方式与用于其他基于 Git 的任何存储库时完全相同。您可以更改代码、添加文件并创建本地提交。如果准备共享，您可以使用 Git 暂存选项将您的提交推送到 CodeCommit 存储库。如果尚未在 Git 配置文件中配置作者和提交者信息，可在提交和推送之前执行此操作。用于 IAM 用户的 Git 凭证已在本地存储且与所连接的 AWS 凭证配置文件相关联，因此，当您推送到 CodeCommit 时，系统将不再提示您提供这些凭证。

有关使用 Toolkit for Eclipse 的详细信息，请参阅 AWS Toolkit for Eclipse 入门指南。

适用于不使用 AWS CLI 的 SSH 用户的设置

如果您打算为存储库使用 SSH 连接，则可在不安装 AWS CLI 的情况下连接 AWS CodeCommit。AWS CLI 包含在您使用和管理 CodeCommit 存储库时非常有用的命令，但对于初始设置来说，它不是必需的。

本主题假定：

- 您已设置具有 CodeCommit 所需策略或权限的 IAM 用户，以及上传密钥所需的 IAMUserSSHKeys 托管策略或等效权限。有关更多信息，请参阅将基于身份的策略 (IAM policy) 用于 CodeCommit。
- 您已有或知道如何创建公有-私有密钥对。我们强烈建议您为 SSH 密钥使用安全密码。
- 您熟悉 SSH、Git 客户端及其配置文件。
- （如果您使用的是 Windows）您已安装命令行实用程序，例如模拟 bash shell 的 Git Bash。

如果您需要更多指导，请按照适用于 Linux、macOS 或 Unix 上的 SSH 连接或适用于 Windows 上的 SSH 连接中的说明操作。

主题

步骤 1：将公有密钥关联到 IAM 用户

1. 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/。
2. 在 IAM 控制台的导航窗格中，选择用户，然后从用户列表中选择您的 IAM 用户。
3. 在 Security Credentials 选项卡上选择 Upload SSH public key。
4. 将 SSH 公有密钥的内容粘贴到该字段中，然后选择 Upload SSH Key (上传 SSH 密钥)。

> **Tip**
>
> 公有-私有密钥对必须为 SSH-2 RSA、采用 OpenSSH 格式且包含 2048 位。该密钥看上去类似于以下内容：

```
ssh-rsa EXAMPLE-
AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-
name@ip-192-0-2-137
```

IAM 只接受 OpenSSH 格式的公有密钥。如果您以其他格式提供自己的公有密钥，系统会显示一条错误消息，指出该密钥格式无效。

5. 复制 SSH 密钥 ID（例如，*APKAEIBAERJR2EXAMPLE*），然后关闭控制台。

步骤 2：将 CodeCommit 添加到 SSH 配置中

1. 在终端（Linux、macOS 或 Unix）或 bash 仿真器 (Windows) 中，通过键入 cat>> ~/.ssh/config 来编辑 SSH 配置文件：

```
Host git-codecommit.*.amazonaws.com
User Your-SSH-Key-ID, such as APKAEIBAERJR2EXAMPLE
IdentityFile Your-Private-Key-File, such as ~/.ssh/codecommit_rsa or ~/.ssh/id_rsa
```

> **Tip**
>
> 如果您有多个 SSH 配置，请确保在内容前后包含空白行。保存文件 (同时按 Ctrl 和 d 键)。

2. 运行下面的命令测试您的 SSH 配置：

```
ssh git-codecommit.us-east-2.amazonaws.com
```

出现提示时，输入 SSH 密钥文件的密码。如果一切配置正确，您应会看到下面的成功消息：

```
You have successfully authenticated over SSH. You can use Git to interact with
 CodeCommit.
```

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

要连接到存储库，请按照连接存储库中的步骤操作。要创建存储库，请按照创建存储库中的步骤操作。

在 Linux, macOS 或 Unix 上设置到 AWS CodeCommit 存储库的 SSH 连接的步骤

在首次连接到 CodeCommit 之前，您必须先完成初始配置步骤。设置好计算机和 AWS 配置文件后，您可以连接到 CodeCommit 存储库并将该存储库克隆到您的计算机（也称为创建本地存储库）。如果您刚刚接触 Git，我们建议您查看哪里可以找到有关 Git 的更多信息？中的信息。

主题

步骤 1：CodeCommit 的初始配置

按照以下步骤设置 Amazon Web Services 账户、创建 IAM 用户并配置对 CodeCommit 的访问。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

> **Note**
>
> CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

> **Note**
>
> 如果您想使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

步骤 2：安装 Git

要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议您访问 Git 下载等网站。

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

第 3 步：在 Linux、macOS 或 Unix 上配置凭证

SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥

为 Git 和 CodeCommit 设置公有密钥和私有密钥

1. 在本地计算机上的终端中，运行 ssh-keygen 命令，并按照说明将文件保存到您的配置文件的 .ssh 目录中。

   > **Note**
   >
   > 请务必与系统管理员确认应将密钥文件存储在何处以及应使用何种文件命名模式。

   例如：

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/user-name/.ssh/id_rsa): Type /home/
your-user-name/.ssh/ and a file name here, for example /home/your-user-name/.ssh/
codecommit_rsa

Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in /home/user-name/.ssh/codecommit_rsa.
Your public key has been saved in /home/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
The key's randomart image is:
+--[ RSA 2048]----+
|        E.+.o*.++|
```

```
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

这会生成：

- *codecommit_rsa* 文件，该文件为私有密钥文件。
- *codecommit_rsa*.pub 文件，该文件为公有密钥文件。

> **Tip**
>
> 默认情况下，ssh-keygen 生成 2048 位密钥。您可以使用 -t 和 -b 参数来指定密钥的类型和长度。如果您想要 rsa 格式的 4096 位密钥，可以通过使用以下参数来运行命令：
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> 有关 SSH 密钥所需的格式和长度的更多信息，请参阅将 IAM 与 CodeCommit 配合使用。

2. 运行以下命令显示公有密钥文件 (*codecommit_rsa*.pub) 的值：

```
cat ~/.ssh/codecommit_rsa.pub
```

复制该值。其内容类似于以下所示：

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@ip-192-0-2-137
```

3. 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/。

> **Note**
>
> 您可以在我的安全凭证中直接查看和管理您的 CodeCommit 凭证。有关更多信息，请参阅查看和管理您的凭证。

4. 在 IAM 控制台的导航窗格中，选择用户，然后从用户列表中选择您的 IAM 用户。
5. 在用户详细信息页面上，选择 Security Credentials 选项卡，然后选择 Upload SSH public key。
6. 将 SSH 公有密钥的内容粘贴到该字段中，然后选择 Upload SSH public key。
7. 复制或保存 SSH Key ID 中的信息 (例如，*APKAEIBAERJR2EXAMPLE*)。

> **Note**
>
> 如果您上传了多个 SSH 密钥 ID，则按密钥 ID 的字母顺序（而不是按上传日期）列出密钥。请确保已复制与正确上传日期关联的密钥 ID。

8. 在本地计算机上，使用文本编辑器在 ~/.ssh 目录中创建一个配置文件，然后向该文件中添加以下行，其中 *User* 的值是您之前复制的 SSH 密钥 ID：

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
>
> 如果您给私有密钥文件起了一个 *codecommit_rsa* 以外的名称，请务必在此处使用自己指定的名称。
>
> 您可以设置对多个 Amazon Web Services 账户中存储库的 SSH 访问权限。有关更多信息，请参阅SSH 与 AWS CodeCommit 的连接问题排查。

将该文件命名为 `config` 并保存。

9. 在终端中，运行下面的命令更改配置文件的权限：

```
chmod 600 config
```

10. 运行下面的命令测试您的 SSH 配置：

```
ssh git-codecommit.us-east-2.amazonaws.com
```

系统会要求您确认连接，因为 `git-codecommit.us-east-2.amazonaws.com` 尚未包含在您的已知主机文件中。CodeCommit 服务器指纹作为验证的一部分显示（适用于 MD5 的 `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e` 或适用于 SHA256 的 `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`）。

> **Note**
>
> CodeCommit 服务器指纹对每个 AWS 区域都是唯一的。要查看某个 AWS 区域的服务器指纹，请参阅的服务器指纹 CodeCommit。

确认连接后，您应会看到已将服务器添加到已知主机文件的确认消息和成功连接消息。如果未看到成功消息，请仔细检查您是否已将 `config` 文件保存到为访问 CodeCommit 而配置的 IAM 用户的 ~/.ssh 目录中，以及您是否指定了正确的私有密钥文件。

如需获取帮助您排查问题的信息，请运行 `ssh` 命令并指定 `-v` 参数：例如：

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

有关可帮助您解决连接问题的信息，请参阅 SSH 与 AWS CodeCommit 的连接问题排查。

步骤 4：连接 CodeCommit 控制台并克隆存储库

如果管理员已将 CodeCommit 存储库的名称和连接详细信息发送给您，您可以跳过该步骤并直接克隆存储库。

连接 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。

2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。
   - 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

   > **Note**
   >
   > 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 打开 终端。在 /tmp 目录中，使用您复制的 SSH URL，运行 git clone 命令以克隆存储库。例如，要将名为 *MyDemoRepo* 的存储库克隆成美国东部（俄亥俄州）区域中名为 *my-demo-repo* 的存储库，请运行以下命令：

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

   > **Note**
   >
   > 如果连接测试成功，但克隆命令失败，则可能是您没有访问配置文件所需的权限，或有其他设置与您的配置文件冲突。尝试再次连接，这次在命令中包含 SSH 密钥 ID。例如：
   >
   > ```
   > git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   > ```
   >
   > 有关更多信息，请参阅访问错误：已将公有密钥成功上传到 IAM，但在 Linux、macOS 或 Unix 系统上进行连接时失败。

   有关如何连接存储库的更多信息，请参阅通过克隆 CodeCommit 存储库来连接存储库。

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

在 Windows 上设置到 AWS CodeCommit 存储库的 SSH 连接的步骤

在首次连接到 AWS CodeCommit 之前，您必须先完成初始配置步骤。设置好计算机和 AWS 配置文件后，您可以连接到 CodeCommit 存储库并将该存储库克隆到您的计算机（也称为创建本地存储库）。如果您刚刚接触 Git，我们建议您查看哪里可以找到有关 Git 的更多信息？中的信息。

主题

步骤 1：CodeCommit 的初始配置

按照以下步骤设置 Amazon Web Services 账户、创建 IAM 用户并配置对 CodeCommit 的访问。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

> **Note**
>
> CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

> **Note**
>
> 如果您想使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

步骤 2：安装 Git

要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议您访问 Git 下载等网站。

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

如果您安装的 Git 版本不包含 Bash 仿真器（例如 Git Bash），请安装一个。配置 SSH 连接时，您将使用该仿真器而不是 Windows 命令行。

步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥

在 Windows 上为 Git 和 CodeCommit 设置公有密钥和私有密钥

1. 打开 Bash 仿真器。

> **Note**
>
> 您可能需要使用管理权限运行仿真器。

在仿真器中运行 ssh-keygen 命令，按照说明将文件保存到您的配置文件的 .ssh 目录。

例如：

```
$ ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/drive/Users/user-name/.ssh/id_rsa): Type a
 file name here, for example /c/Users/user-name/.ssh/codecommit_rsa

Enter passphrase (empty for no passphrase): <Type a passphrase, and then press
 Enter>
Enter same passphrase again: <Type the passphrase again, and then press Enter>

Your identification has been saved in drive/Users/user-name/.ssh/codecommit_rsa.
Your public key has been saved in drive/Users/user-name/.ssh/codecommit_rsa.pub.
The key fingerprint is:
45:63:d5:99:0e:99:73:50:5e:d4:b3:2d:86:4a:2c:14 user-name@client-name
The key's randomart image is:
+--[ RSA 2048]----+
|        E.+.o*.++|
|        .o .=.=o.|
|       . ..  *. +|
|        ..o . +..|
|        So . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

这会生成：

- *codecommit_rsa* 文件，该文件为私有密钥文件。
- *codecommit_rsa*.pub 文件，该文件为公有密钥文件。

> **Tip**
>
> 默认情况下，ssh-keygen 生成 2048 位密钥。您可以使用 -t 和 -b 参数来指定密钥的类型和长度。如果您想要 rsa 格式的 4096 位密钥，可以通过使用以下参数来运行命令：
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> 有关 SSH 密钥所需的格式和长度的更多信息，请参阅将 IAM 与 CodeCommit 配合使用。

2. 运行以下命令显示公有密钥文件 (*codecommit_rsa*.pub) 的值：

```
cd .ssh
notepad codecommit_rsa.pub
```

复制文件内容，然后在不保存的情况下关闭记事本。该文件的内容类似于以下内容：

```
ssh-rsa EXAMPLE-AfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJB
gNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb2
5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhc
NMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAw
DgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAS=EXAMPLE user-name@computer-name
```

3. 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/。

> **Note**
>
> 您可以在我的安全凭证中直接查看和管理您的 CodeCommit 凭证。有关更多信息，请参阅查看和管理您的凭证。

4. 在 IAM 控制台的导航窗格中，选择用户，然后从用户列表中选择您的 IAM 用户。
5. 在用户详细信息页面上，选择 Security Credentials 选项卡，然后选择 Upload SSH public key。

6. 将 SSH 公有密钥的内容粘贴到该字段中，然后选择 Upload SSH public key。
7. 复制或保存 SSH Key ID 中的信息 (例如，*APKAEIBAERJR2EXAMPLE*)。

> **Note**
>
> 如果您上传了多个 SSH 密钥 ID，则按密钥 ID 的字母顺序（而不是按上传日期）列出密钥。请确保已复制与正确上传日期关联的密钥 ID。

8. 在 Bash 仿真器中，运行以下命令在 ~/.ssh 目录中创建一个配置文件，如果文件已存在，则编辑该文件：

```
notepad ~/.ssh/config
```

9. 将以下行添加到该文件中，其中 *User* 的值是您之前复制的 SSH 密钥 ID，*IdentityFile* 的值是私有密钥文件的路径和名称：

```
Host git-codecommit.*.amazonaws.com
  User APKAEIBAERJR2EXAMPLE
  IdentityFile ~/.ssh/codecommit_rsa
```

> **Note**
>
> 如果您给私有密钥文件起了一个 *codecommit_rsa* 以外的名称，请务必在此处使用自己指定的名称。
>
> 您可以设置对多个 Amazon Web Services 账户中存储库的 SSH 访问权限。有关更多信息，请参阅SSH 与 AWS CodeCommit 的连接问题排查。

将文件另存为 config（不是 config.txt），然后关闭记事本。

> **Important**
>
> 文件名必须为 `config`，不带文件扩展名。否则，SSH 连接将失败。

10. 运行下面的命令测试您的 SSH 配置：

```
ssh git-codecommit.us-east-2.amazonaws.com
```

系统会要求您确认连接，因为 `git-codecommit.us-east-2.amazonaws.com` 尚未包含在您的已知主机文件中。CodeCommit 服务器指纹作为验证的一部分显示（适用于 MD5 的 `a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e` 或适用于 SHA256 的 `3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`）。

> **Note**
>
> CodeCommit 服务器指纹对每个 AWS 区域都是唯一的。要查看某个 AWS 区域的服务器指纹，请参阅的服务器指纹 CodeCommit。

确认连接后，您应会看到已将服务器添加到已知主机文件的确认消息和成功连接消息。如果未看到成功消息，请仔细检查以下事项：已将 `config` 文件保存在您配置的用于访问 CodeCommit 的 IAM 用户的 ~/.ssh 目录中；`config` 文件没有文件扩展名（例如，不能将其命名为 config.txt）；已指定正确的私有密钥文件（*`codecommit_rsa`*，不是 *`codecommit_rsa`*.pub）。

要对问题进行排查，请运行 `ssh` 命令并指定 `-v` 参数。例如：

```
ssh -v git-codecommit.us-east-2.amazonaws.com
```

有关可帮助您解决连接问题的信息，请参阅 SSH 与 AWS CodeCommit 的连接问题排查。

步骤 4：连接 CodeCommit 控制台并克隆存储库

如果管理员已将 CodeCommit 存储库的名称和连接详细信息发送给您，您可以跳过该步骤并直接克隆存储库。

连接 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。
   - 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

   > **Note**
   >
   > 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 在 Bash 模拟器中，使用您复制的 SSH URL 运行 git clone 命令以克隆存储库。该命令会在运行命令的目录的子目录中创建本地存储库。例如，要将名为 *MyDemoRepo* 的存储库克隆成美国东部（俄亥俄州）区域中名为 *my-demo-repo* 的存储库，请运行以下命令：

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

   或者，打开命令提示符，使用您上传到 IAM 的公有密钥的 URL 和 SSH 密钥 ID 运行 git clone 命令。这将在运行命令的目录的子目录中创建本地存储库。例如，要将名为 *MyDemoRepo* 的存储库克隆成名为 *my-demo-repo* 的本地存储库，请运行以下命令：

   ```
   git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
   ```

   有关更多信息，请参阅 通过克隆 CodeCommit 存储库来连接存储库 和 创建提交：

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

使用 AWS CLI 凭证助手在 Linux、macOS 或 Unix 上对 AWS CodeCommit 存储库进行 HTTPS 连接的设置步骤

在首次连接到 AWS CodeCommit 之前，您必须先完成初始配置步骤。对于大多数用户来说，可通过执行适用于使用 Git 凭证的 HTTPS 用户中的步骤来轻松完成这一过程。但是，如果要使用根账户、联合访问或临时凭证连接 CodeCommit，可以使用 AWS CLI 中包含的凭证助手。

> **Note**
>
> 虽然使用联合访问、身份提供者或临时凭证连接 CodeCommit 时，凭证助手是一种受支持的方法，但推荐的方法是安装并使用 git-remote-codecommit 实用程序。有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

主题

步骤 1：CodeCommit 的初始配置

按照以下步骤设置 Amazon Web Services 账户、创建和配置 IAM 用户并安装 AWS CLI。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

> **Note**
>
> CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

安装和配置 AWS CLI

1. 在本地计算机上，下载并安装 AWS CLI。这是从命令行与 CodeCommit 进行交互的先决条件。我们建议您安装 AWS CLI 版本 2。它是 AWS CLI 的最新主版本，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书与 git-remote-codecommit 的 AWS CLI 版本。

   有关更多信息，请参阅使用 AWS 命令行界面进行设置。

   > **Note**
   >
   > CodeCommit 仅适用于 AWS CLI 1.7.38 及更高版本。作为最佳做法，请安装 AWS CLI 或将其升级到可用的最新版本。要确定您安装的 AWS CLI 的版本，请运行 aws --version 命令。
   > 要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证 AWS CLI 中的 CodeCommit 命令是否已安装：

```
aws codecommit help
```

该命令返回 CodeCommit 命令的列表。

3. 使用 configure 命令通过配置文件来配置 AWS CLI，如下所示：

```
aws configure
```

出现提示时，指定要用于 CodeCommit 的 IAM 用户的 AWS 访问密钥和 AWS 秘密访问密钥。此外，请务必指定存储库所在的 AWS 区域，例如 us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

有关创建和配置相关配置文件以与 AWS CLI 配合使用的详细信息，请参阅以下内容：

- 命名配置文件
- 在 AWS CLI 中使用 IAM 角色
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到另一个 AWS 区域中的存储库或资源，您必须使用默认区域名称重新配置 AWS CLI。CodeCommit 支持的默认区域名称包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1

- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

有关 CodeCommit 和 AWS 区域的更多信息，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

步骤 2：安装 Git

要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议您访问 Git 下载等网站。

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

步骤 3：设置凭证助手

1. 从终端使用 Git 运行 git config，在其中指定为 AWS 凭证配置文件使用 Git 凭证辅助程序，并使 Git 凭证辅助程序发送存储库的路径：

```
git config --global credential.helper '!aws codecommit credential-helper $@'
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 凭证助手使用默认的 AWS 凭证配置文件或 Amazon EC2 实例角色。如果您创建了用于 CodeCommit 的 AWS 凭证配置文件，则可以指定要使用的配置文件，如 `CodeCommitProfile`：
>
> ```
> git config --global credential.helper '!aws --profile CodeCommitProfile
>  codecommit credential-helper $@'
> ```
>
> 如果您的配置文件名称包含空格，请确保用引号 (") 将名称引起来。
> 您可以使用 `--local` 而不是 `--global` 配置每个存储库的配置文件，而不是进行全局配置。

Git 凭证辅助程序将以下值写入 `~/.gitconfig`：

```
[credential]
    helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
    UseHttpPath = true
```

> **Important**
>
> 如果要对 CodeCommit 使用同一本地计算机上的不同 IAM 用户，您必须再次运行 git config 命令并指定不同的 AWS 凭证配置文件。

2. 运行 git config --global --edit 验证前面的值是否已写入 `~/.gitconfig`。如果成功，您应能看到前面的值（除了 Git 全局配置文件中可能已有的值）。要退出，通常需要键入 **`:q`**，然后按 Enter。

   如果在配置凭证辅助程序后遇到问题，请参阅故障排除。

   > **Important**
   >
   > 如果您使用的是 macOS，请通过以下步骤确保凭证助手配置正确。

3. 如果您使用的是 macOS，请使用 HTTPS 连接到 CodeCommit 存储库。首次使用 HTTPS 连接到 CodeCommit 存储库后，后续访问会在约 15 分钟后失败。macOS 上的默认 Git 版本使用 Keychain Access 实用程序存储凭证。为安全起见，为访问您的 CodeCommit 存储库而生成的密码是临时的，因此密钥链中存储的凭证将在约 15 分钟后失效。为防止使用这些过期凭证，您必须执行以下操作之一：

   - 安装默认不使用密钥链的 Git 版本。
   - 将 Keychain Access 实用程序配置为不为 CodeCommit 存储库提供凭证。

   1. 打开 Keychain Access 实用程序。(您可以使用 Finder 查找它。)
   2. 搜索 `git-codecommit.us-east-2.amazonaws.com`。突出显示该行，打开上下文菜单或右键单击它，然后选择 Get Info。
   3. 选择 Access Control 选项卡。
   4. 在 Confirm before allowing access (运行访问前进行确认) 中，选择 `git-credential-osxkeychain`，然后选择减号将它从列表中删除。

      > **Note**
      >
      > 在从列表中删除 `git-credential-osxkeychain` 后，当您运行 Git 命令时，会看到一条弹出消息。选择拒绝以继续。如果您不希望显示弹出窗口，可考虑以下其他选项：
      >
      > - 使用 SSH 而不是 HTTPS 连接 CodeCommit。有关更多信息，请参阅适用于 Linux、macOS 或 Unix 上的 SSH 连接。

- 在 Keychain Access 实用程序中，在针对 `git-codecommit.us-east-2.amazonaws.com` 的访问控制选项卡上，选择允许所有应用程序访问此项目（访问此项目不受限制）选项。这将阻止弹出窗口，但凭证最终会到期（平均而言，大约需要 15 分钟），然后会显示 403 错误消息。如果发生这种情况，您必须删除密钥链项才能恢复功能。
- 有关更多信息，请参阅macOS 版 Git：我成功配置了凭证助手，但在访问我的存储库时被系统拒绝 (403)。

步骤 4：连接 CodeCommit 控制台并克隆存储库

如果管理员已将 CodeCommit 存储库的名称和连接详细信息发送给您，您可以跳过该步骤并直接克隆存储库。

连接 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。
   - 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

   > **Note**
   >
   > 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 打开终端并使用复制的 HTTPS URL 运行 git clone 命令。例如，要将名为 *MyDemoRepo* 的存储库克隆成美国东部（俄亥俄州）区域中名为 *my-demo-repo* 的存储库，请运行以下命令：

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

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

使用 AWS CLI 凭证助手在 Windows 上设置到 AWS CodeCommit 存储库的 HTTPS 连接的步骤

在首次连接到 AWS CodeCommit 之前，您必须先完成初始配置步骤。对于大多数用户来说，可通过执行适用于使用 Git 凭证的 HTTPS 用户中的步骤来轻松完成这一过程。但是，如果要使用根账户、联合访问或临时凭证连接 CodeCommit，可以使用 AWS CLI 中包含的凭证助手。

> **Note**
>
> 虽然使用联合访问、身份提供者或临时凭证连接 CodeCommit 时，凭证助手是一种受支持的方法，但推荐的方法是安装并使用 git-remote-codecommit 实用程序。有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

本主题将指导您完成所需步骤以安装 AWS CLI、设置您的计算机和 AWS 配置文件、连接到 CodeCommit 存储库并将存储库克隆到您的计算机（这也称为创建本地存储库）。如果您刚刚接触 Git，我们建议您查看哪里可以找到有关 Git 的更多信息？中的信息。

主题

步骤 1：CodeCommit 的初始配置

按照以下步骤设置 Amazon Web Services 账户、创建和配置 IAM 用户并安装 AWS CLI。AWS CLI 包含一个凭证助手，您需要对其进行配置，以便通过 HTTPS 方式连接 CodeCommit 存储库。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

安装和配置 AWS CLI

1. 在本地计算机上，下载并安装 AWS CLI。这是从命令行与 CodeCommit 进行交互的先决条件。我们建议您安装 AWS CLI 版本 2。它是 AWS CLI 的最新主版本，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书与 git-remote-codecommit 的 AWS CLI 版本。

有关更多信息，请参阅使用 AWS 命令行界面进行设置。

> **Note**
>
> CodeCommit 仅适用于 AWS CLI 1.7.38 及更高版本。作为最佳做法，请安装 AWS CLI 或将其升级到可用的最新版本。要确定您安装的 AWS CLI 的版本，请运行 aws --version 命令。
>
> 要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证 AWS CLI 中的 CodeCommit 命令是否已安装：

```
aws codecommit help
```

该命令返回 CodeCommit 命令的列表。

3. 使用 configure 命令通过配置文件来配置 AWS CLI，如下所示：

```
aws configure
```

出现提示时，指定要用于 CodeCommit 的 IAM 用户的 AWS 访问密钥和 AWS 秘密访问密钥。此外，请务必指定存储库所在的 AWS 区域，例如 us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

有关创建和配置相关配置文件以与 AWS CLI 配合使用的详细信息，请参阅以下内容：

- 命名配置文件
- 在 AWS CLI 中使用 IAM 角色
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到另一个 AWS 区域中的存储库或资源，您必须使用默认区域名称重新配置 AWS CLI。CodeCommit 支持的默认区域名称包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1

- il-central-1

有关 CodeCommit 和 AWS 区域的更多信息，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

步骤 2：安装 Git

要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

要安装 Git，建议访问 Git for Windows 等网站。如果您使用此链接安装 Git，则可以接受除以下设置之外的所有安装默认设置：

- 在调整 PATH 环境步骤中出现提示时，从命令行中选择使用 Git 的选项。
- （可选）如果您打算将 HTTPS 与 AWS CLI 中包含的凭证助手一起使用，而不是为 CodeCommit 配置 Git 凭证，请在配置额外选项页面上，确保清除启用 Git Credential Manager 选项。仅当 IAM 用户配置 Git 凭证时，Git Credential Manager 才与 CodeCommit 兼容。有关更多信息，请参阅 适用于使用 Git 凭证的 HTTPS 用户 和 Windows 版 Git：我安装了 Windows 版 Git，但在访问我的存储库时被系统拒绝 (403)：

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

步骤 3：设置凭证助手

AWS CLI 包含一个可供您与 CodeCommit 配合使用的 Git 凭证助手。该 Git 凭证助手需要 AWS 凭证配置文件，后者存储着 IAM 用户的 AWS 访问密钥 ID 和 AWS 秘密访问密钥（以及默认的 AWS 区域名称和默认输出格式）的副本。Git 凭证助手使用该信息自动对 CodeCommit 进行身份验证，因此，当您使用 Git 与 CodeCommit 交互时，不必每次都输入该信息。

1. 打开命令提示符，使用 Git 运行 git config，在其中指定为 AWS 凭证配置文件使用 Git 凭证助手，这可以使 Git 凭证助手发送存储库的路径。

```
git config --global credential.helper "!aws codecommit credential-helper $@"
git config --global credential.UseHttpPath true
```

Git 凭证辅助程序向 .gitconfig 文件写入以下内容：

```
[credential]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

> **Important**
>
> - 如果您使用的是 Bash 仿真器而不是 Windows 命令行，则必须使用单引号而不是双引号。
> - 凭证助手将使用默认的 AWS 配置文件或 Amazon EC2 实例角色。如果您已创建要使用的 AWS 凭证配置文件，例如 *CodeCommitProfile*，可以按照如下方式修改命令来使用它：
>
> ```
> git config --global credential.helper "!aws codecommit credential-helper
>  --profile CodeCommitProfile $@"
> ```
>
> 这将向 .gitconfig 文件中写入以下内容：
>
> ```
> [credential]
>     helper = !aws codecommit credential-helper --profile=CodeCommitProfile
>  $@
>     UseHttpPath = true
> ```
>
> - 如果您的配置文件名称包含空格，则在运行此命令后，必须编辑 .gitconfig 文件以使用单引号 (') 将此命令引起来。否则，凭证辅助程序将不起作用。
> - 如果您的 Windows 版 Git 安装包含 Git Credential Manager 实用程序，则在前几次连接尝试后，您会看到 403 错误或向 Credential Manager 实用程序提供凭证的提示。解决该问题的最可靠的方法是卸载并重新安装 Windows 版 Git，安装时不要选中 Git Credential Manager 实用程序选项，因为它与 CodeCommit 不兼容。若要保留 Git

Credential Manager 实用程序，您必须执行额外的配置步骤来使用 CodeCommit，包括手动修改 .gitconfig 文件，以指定在连接 CodeCommit 时使用 AWS CodeCommit 的凭证助手。从 Credential Manager 实用程序（可在“控制面板”中找到该实用程序）中删除任何已存储的凭证。删除所有已存储的凭证后，向 .gitconfig 文件中添加以下内容并将其保存，然后在新的命令提示符窗口中重试连接：

```
[credential "https://git-codecommit.us-east-2.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true

[credential "https://git-codecommit.us-east-1.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

您可能还必须先通过指定 --system 而非 --global 或 --local 来重新配置 git config 设置，之后所有连接才能正常工作。

- 如果您想要在同一台本地计算机上为 CodeCommit 使用不同的 IAM 用户，您应指定 git config --local 而不是 git config --global，并为每个 AWS 凭证配置文件运行配置。

2. 运行 git config --global --edit 验证上面的值是否已写入用户配置文件的 .gitconfig 文件（默认为 `%HOME%\.gitconfig` 或 *`drive`*`:\Users\`*`UserName`*`\.gitconfig`）。如果成功，您应该能够看到前面的值（以及 Git 全局配置文件中已有的值）。要退出，通常需要键入 **`:q`**，然后按 Enter。

步骤 4：连接 CodeCommit 控制台并克隆存储库

如果管理员已将 CodeCommit 存储库的名称和连接详细信息发送给您，您可以跳过该步骤并直接克隆存储库。

连接 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。

- 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

> **Note**
>
> 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. 打开命令提示符，使用您复制的 HTTPS URL 运行 git clone 命令。这将在运行命令的目录的子目录中创建本地存储库。例如，要将名为 *MyDemoRepo* 的存储库克隆成美国东部（俄亥俄州）区域中名为 *my-demo-repo* 的存储库，请运行以下命令：

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

在某些版本的 Windows 上，您可能会看到一条弹出消息，要求您输入用户名和密码。这是 Windows 的内置凭证管理系统，但它与 AWS CodeCommit 的凭证辅助程序不兼容。选择 Cancel (取消)。

后续步骤

您已满足先决条件。按照 入门 CodeCommit 中的步骤开始使用 CodeCommit。

开始使用

开始使用 CodeCommit 的最简单的方法是按照 入门 CodeCommit 中的步骤操作。如果您刚刚开始使用 Git 和 CodeCommit，那么还应当考虑按照 Git 和 CodeCommit 入门中的步骤操作。这可帮助您熟悉 CodeCommit，并掌握有关使用 Git 与 CodeCommit 存储库交互的基础知识。

您还可以按照 CodePipeline 和 CodeCommit 简单管道演练中的教程操作，了解如何在持续交付管道中使用 CodeCommit 存储库。

本部分中的教程假定您已完成先决条件和设置，包括：

- 向 IAM 用户分配权限。
- 在您用于本教程的本地计算机上为 HTTPS 或 SSH 连接设置凭证管理。
- 如果您想要使用命令行或终端完成所有操作，包括创建存储库，则还需要配置 AWS CLI。

主题

入门 AWS CodeCommit

本教程向您展示如何使用一些关键 CodeCommit 功能。首先，请创建一个存储库并提交一些对它的更改。然后，浏览文件并查看更改。您还可以创建拉取请求，以便其他用户可以审核并注释代码的更改。

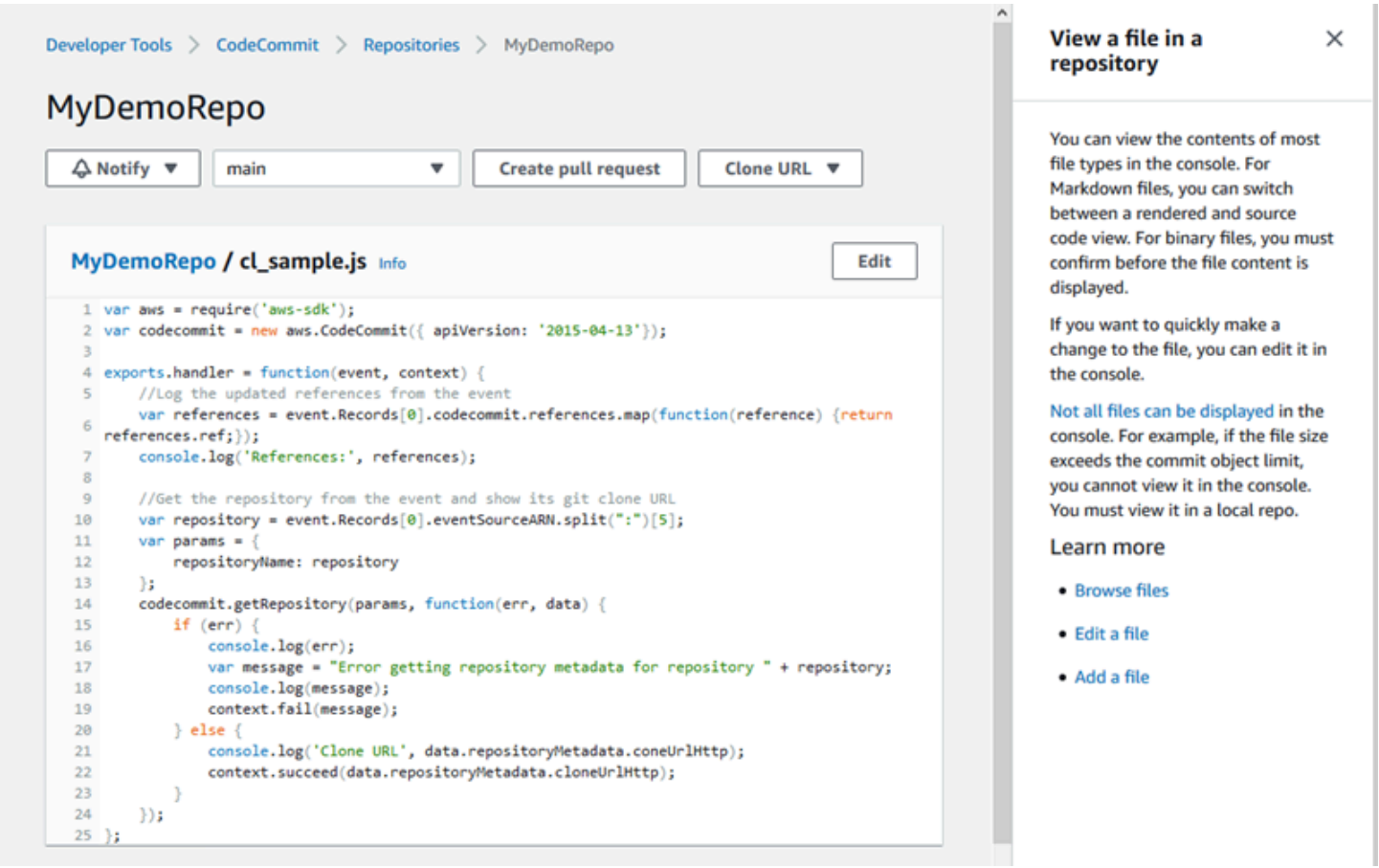
如果要将现有代码迁移到 CodeCommit，请参阅迁移到 AWS CodeCommit。

如果您不熟悉 Git，也可以考虑完成 Git 和 CodeCommit 入门。完成这些教程后，你应该有足够的练习来开始在自己的项目和团队环境中使用 CodeCommit 。

CodeCommit 控制台在可折叠面板中包含有用的信息，您可以通过页面上的信息图标

()

或任何 “信息” 链接打开该面板。您可以随时关闭此面板。



CodeCommit 控制台还提供了一种快速搜索资源的方法，例如存储库、生成项目、部署应用程序和管道。选择转到资源或按下 / 键，然后键入资源的名称。任何匹配结果都会显示在列表中。搜索不区分大小写。您只能看到您有权查看的资源。有关更多信息，请参阅 在控制台中查看资源。

先决条件

在开始之前，必须完成先决条件和设置过程，包括：

- 向 IAM 用户分配权限。
- 在您用来完成本教程的本地计算机上为 HTTPS 或 SSH 连接设置凭证管理。
- AWS CLI 如果要使用命令行或终端执行所有操作（包括创建存储库），请配置。

主题

- 步骤 1：创建 CodeCommit 存储库
- 步骤 2：向您的存储库添加文件
- 步骤 3：浏览存储库的内容

步骤 1：创建 CodeCommit 存储库

您可以使用 CodeCommit 控制台创建 CodeCommit 存储库。如果您已有可用于本教程的存储库，可以跳过该步骤。

> **Note**
>
> 根据您的使用情况，您可能需要为创建或访问存储库付费。有关更多信息，请参阅 CodeCommit 产品信息页面上的定价。

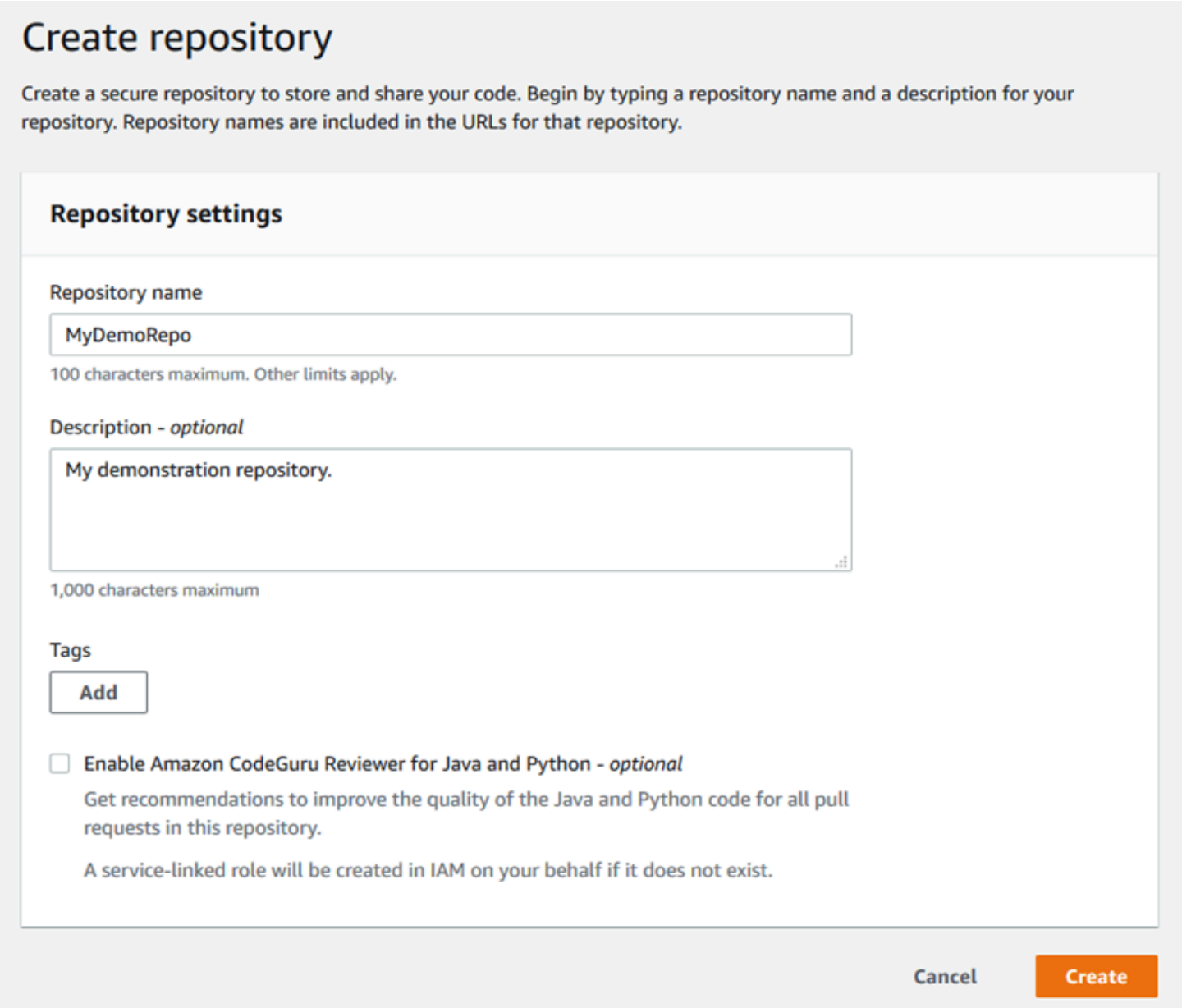
创建 CodeCommit 存储库

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 使用区域选择器选择要在 AWS 区域 哪里创建存储库。有关更多信息，请参阅 区域和 Git 连接端点。
3. 在存储库页面上，选择创建存储库。
4. 在创建存储库页面上的存储库名称中，输入存储库的名称（例如，**MyDemoRepo**）。

   > **Note**
   >
   > 存储库名称区分大小写，且不能超过 100 个字符。有关更多信息，请参阅限制。

5. （可选）在描述中，输入描述（例如，**My demonstration repository**）。这可以帮助您及其他用户了解存储库的用途。
6. （可选）选择 Add t ag，向存储库添加一个或多个存储库标签（可帮助您组织和管理 AWS 资源的自定义属性标签）。有关更多信息，请参阅 在中标记存储库 AWS CodeCommit。
7. （可选）展开其他配置以指定是使用默认密钥 AWS 托管式密钥 还是您自己的客户托管密钥来加密和解密此存储库中的数据。如果您选择使用自己的客户托管密钥，则必须确保该密钥在您创建存储库 AWS 区域 的地方可用，并且该密钥处于活动状态。有关更多信息，请参阅 AWS Key Management Service 和 AWS CodeCommit 存储库加密。

8. （可选）如果此存储库将包含 Java 或 Python 代码，并且您想让 CodeGuru Reviewer 分析该代码，请选择“为 Java 和 Python 启用 Ama zon CodeGuru Reviewer”。 CodeGuru Reviewer 使用多个机器学习模型来查找代码缺陷，并自动在拉取请求中提出改进和修复建议。有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南。
9. 选择创建。



> **Note**
>
> 如果存储库使用了 MyDemoRepo 以外的名称，请务必在剩余步骤中使用该名称。

当存储库打开时，您将看到有关如何直接从 CodeCommit 控制台添加文件的信息。

步骤 2：向您的存储库添加文件

可以通过以下方法向您的存储库添加文件：

- 在 CodeCommit 控制台中创建文件。如果您在控制台中为存储库创建第一个文件，则系统会为您创建一个名为 main 的分支。此分支是存储库的默认分支。
- 使用 CodeCommit 控制台从本地计算机上传文件。如果您从控制台为存储库上传第一个文件，则系统会为您创建一个名为 main 的分支。此分支是存储库的默认分支。
- 使用 Git 客户端将仓库克隆到本地计算机，然后向仓 CodeCommit 库添加、提交和推送文件。在 Git 中执行第一个提交时，系统会为您创建一个分支，并且会将其设置为存储库的默认分支。分支的名称是 Git 客户端的默认选项。请考虑将 Git 客户端配置为使用 main 作为初始分支的名称。

> **Note**
>
> 您可以随时创建分支并更改存储库的默认分支。有关更多信息，请参阅 使用 AWS CodeCommit 存储库中的分支。

最简单的入门方法是打开 CodeCommit 控制台并添加文件。这样，您还可以为存储库创建一个名为 main 的默认分支。有关如何使用向存储库添加文件和创建首次提交的说明 AWS CLI，请参阅使用创建存储库的第一个提交 AWS CLI。

向存储库添加文件

1. 在存储库的导航栏中，选择代码。
2. 选择添加文件，然后选择是创建文件还是从您的计算机上传文件。本教程向您展示了如何执行这两个操作。
3. 要添加文件，请执行以下操作：

   a. 在分支下拉列表中，选择要添加文件的分支。已为您自动选择默认分支。在此处所示的示例中，默认分支名为 *main*。如果要将文件添加到不同的分支，请选择不同的分支。

   b. 在文件名中，为文件输入名称。在代码编辑器中，输入文件的代码。

   c. 在作者姓名中，输入您希望显示给其他存储库用户的姓名。

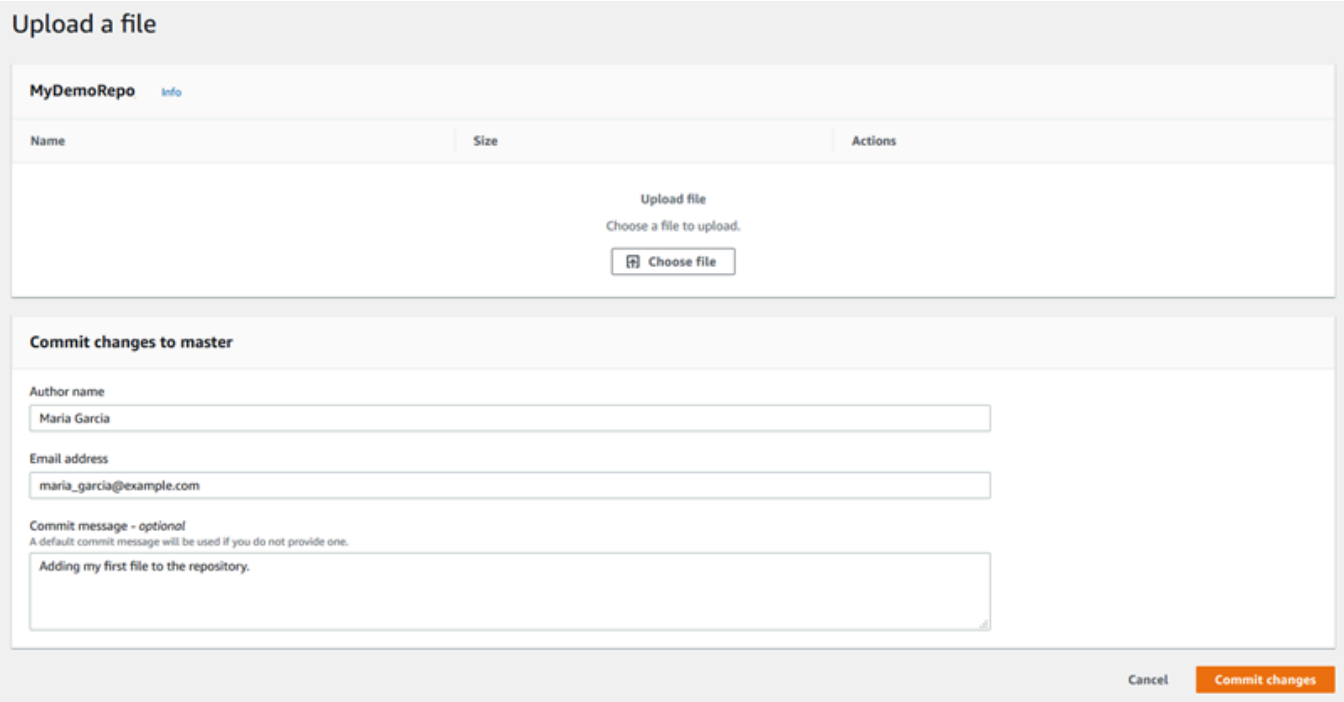
   d. 在电子邮件地址中，输入电子邮件地址。

e. （可选）在提交消息中，输入一条简短的消息。虽然这是可选的，但建议您添加提交消息以帮助您的团队成员了解您添加此文件的原因。如果您没有输入提交消息，将使用默认消息。

f. 选择提交更改。

要上传文件，请执行以下操作：

- 如果正在上传文件，请选择要上传的文件。



- 在作者姓名中，输入您希望显示给其他存储库用户的姓名。
- 在电子邮件地址中，输入电子邮件地址。
- （可选）在提交消息中，输入一条简短的消息。虽然这是可选的，但建议您添加提交消息以帮助您的团队成员了解您添加此文件的原因。如果您没有输入提交消息，将使用默认消息。
- 选择提交更改。

有关更多信息，请参阅 使用 AWS CodeCommit 存储库中的文件。

要使用 Git 客户端克隆仓库，请在本地计算机上安装 Git，然后克隆 CodeCommit 仓库。将一些文件添加到本地存储库并将其推送到 CodeCommit 存储库。有关深入介绍，请尝试 Git 和 CodeCommit 入门。如果您熟悉 Git，但不确定如何使用 CodeCommit 存储库执行此操作，则可以在创建提交步骤 2：创建本地存储库、或中查看示例和说明连接存储库。

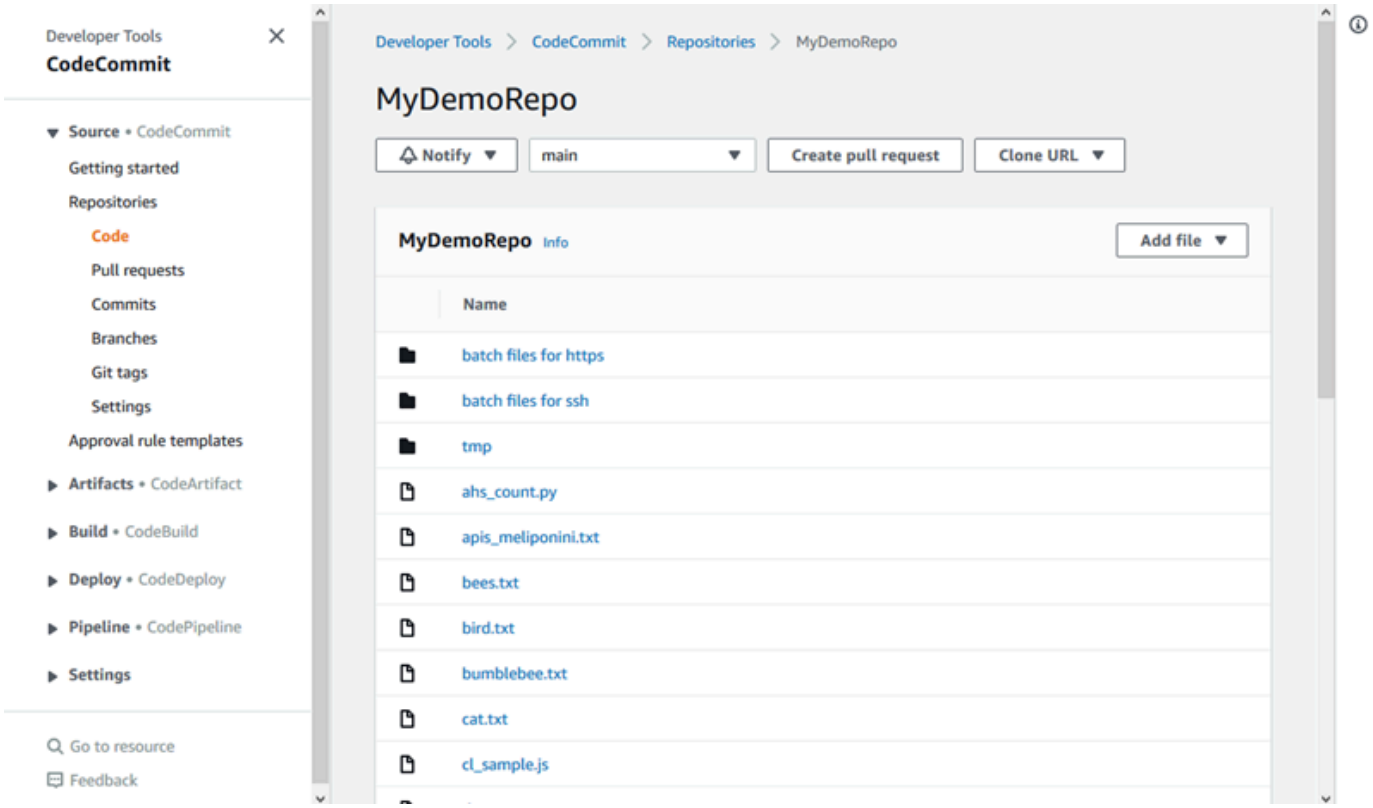
将一些文件添加到 CodeCommit 存储库后，可以在控制台中查看它们。

步骤 3：浏览存储库的内容

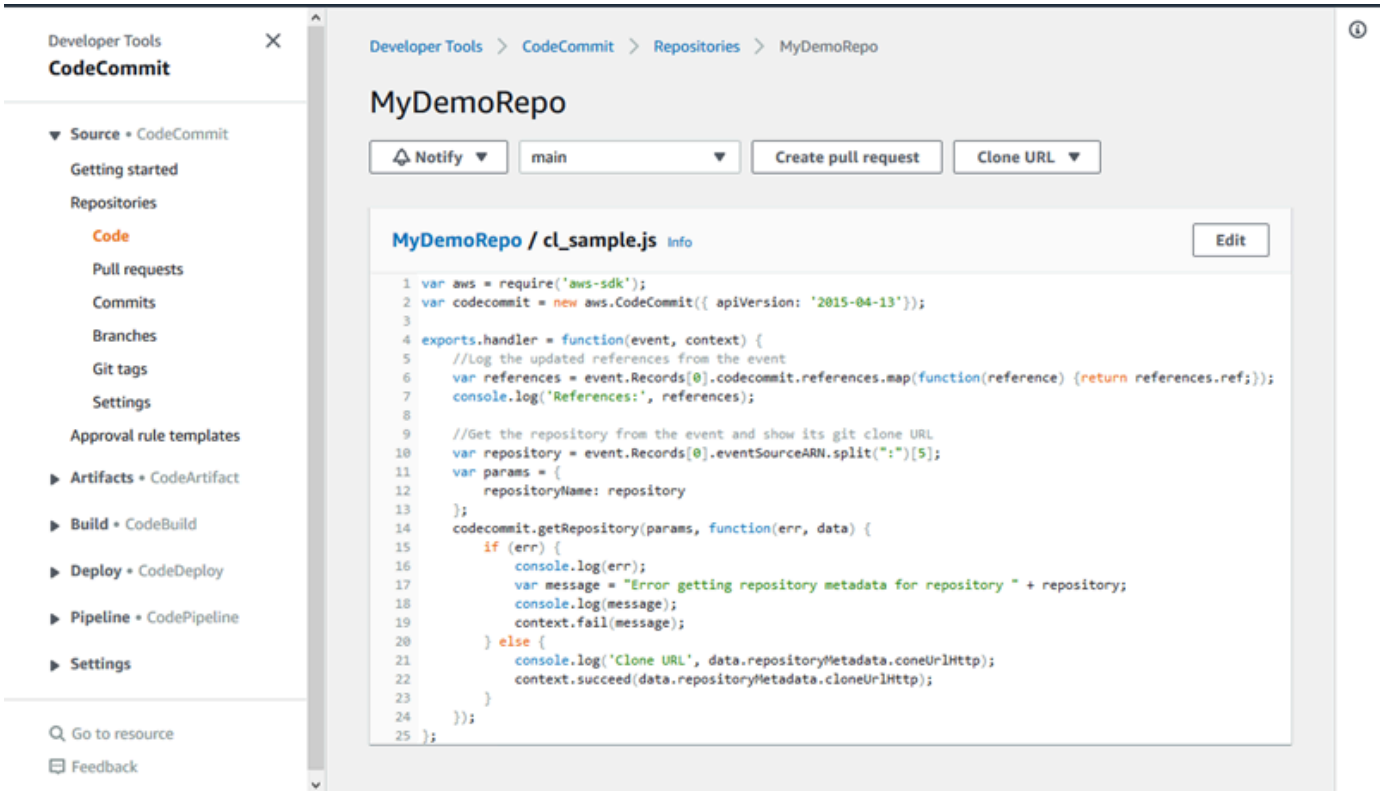
您可以使用 CodeCommit 控制台查看存储库中的文件或快速读取文件的内容。这可以帮助您确定要签出的分支或者确定是否需要创建存储库的本地副本。

浏览存储库

1. 从“存储库”中选择 MyDemoRepo。
2. 页面显示了存储库的默认分支中的内容。要查看其他分支或查看特定标签处的代码，请从列表中选择要查看的分支或标签。在下面的屏幕截图中，视图设置为 main 分支。

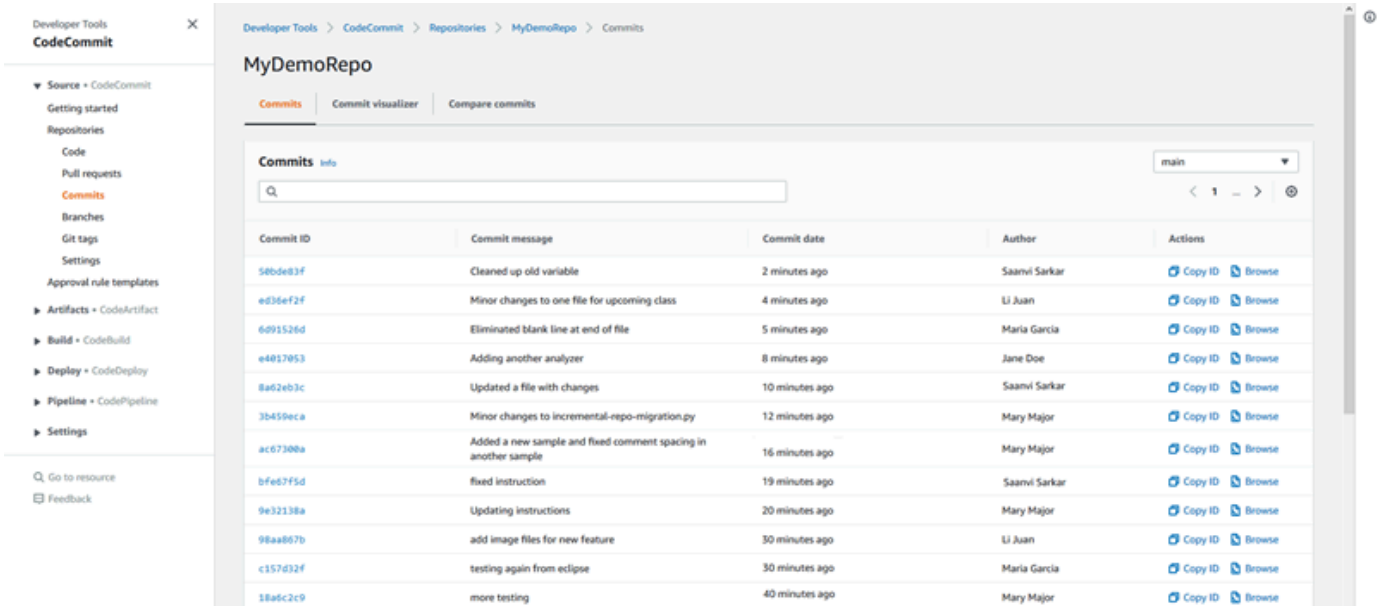


3. 要查看存储库中某个文件的内容，请从列表中选择该文件。要更改所显示代码的颜色，请选择设置图标。



有关更多信息，请参阅 浏览存储库中的文件。

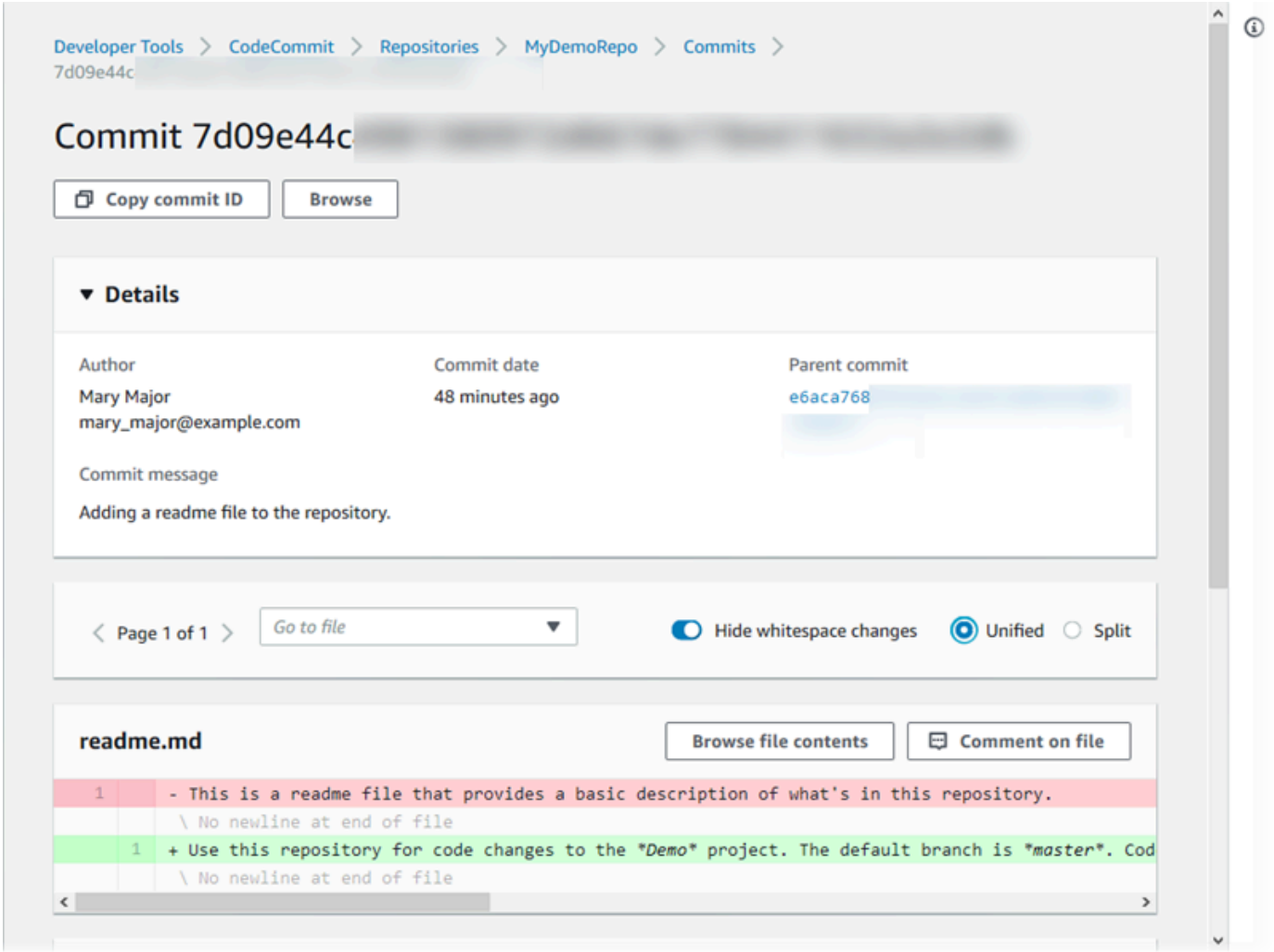
4. 要浏览存储库的提交历史记录，请选择提交。控制台按时间倒序显示默认分支的提交历史记录。可以按作者、日期等查看提交详细信息。



5. 要按分支或 Git 标签查看提交历史记录，请从列表中选择要查看的分支或标签。

6. 要查看提交与其父项之间的差异，请选择缩写的提交 ID。您可以选择更改的显示方式，包括显示或隐藏空格更改，以及是以内联方式（统一视图）还是并排方式（拆分视图）查看更改。

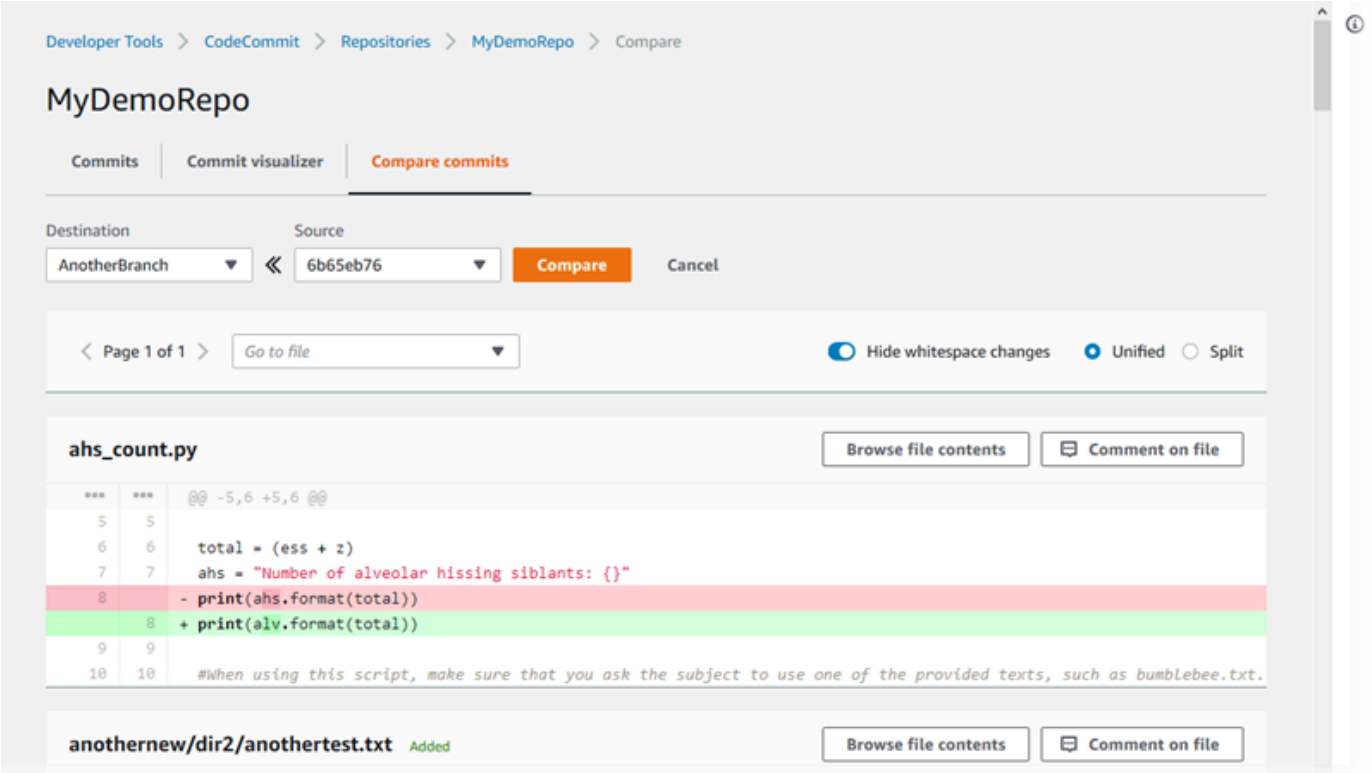
> **Note**
>
> 每当您更改用于查看代码和其他控制台设置的首选项时，都会将它们保存为浏览器 Cookie。有关更多信息，请参阅 使用用户首选项。



7. 要查看对提交的所有评论，请选择提交，然后滚动显示更改以便通过内联方式查看更改。您还可以添加自己的评论并回复他人发表的评论。

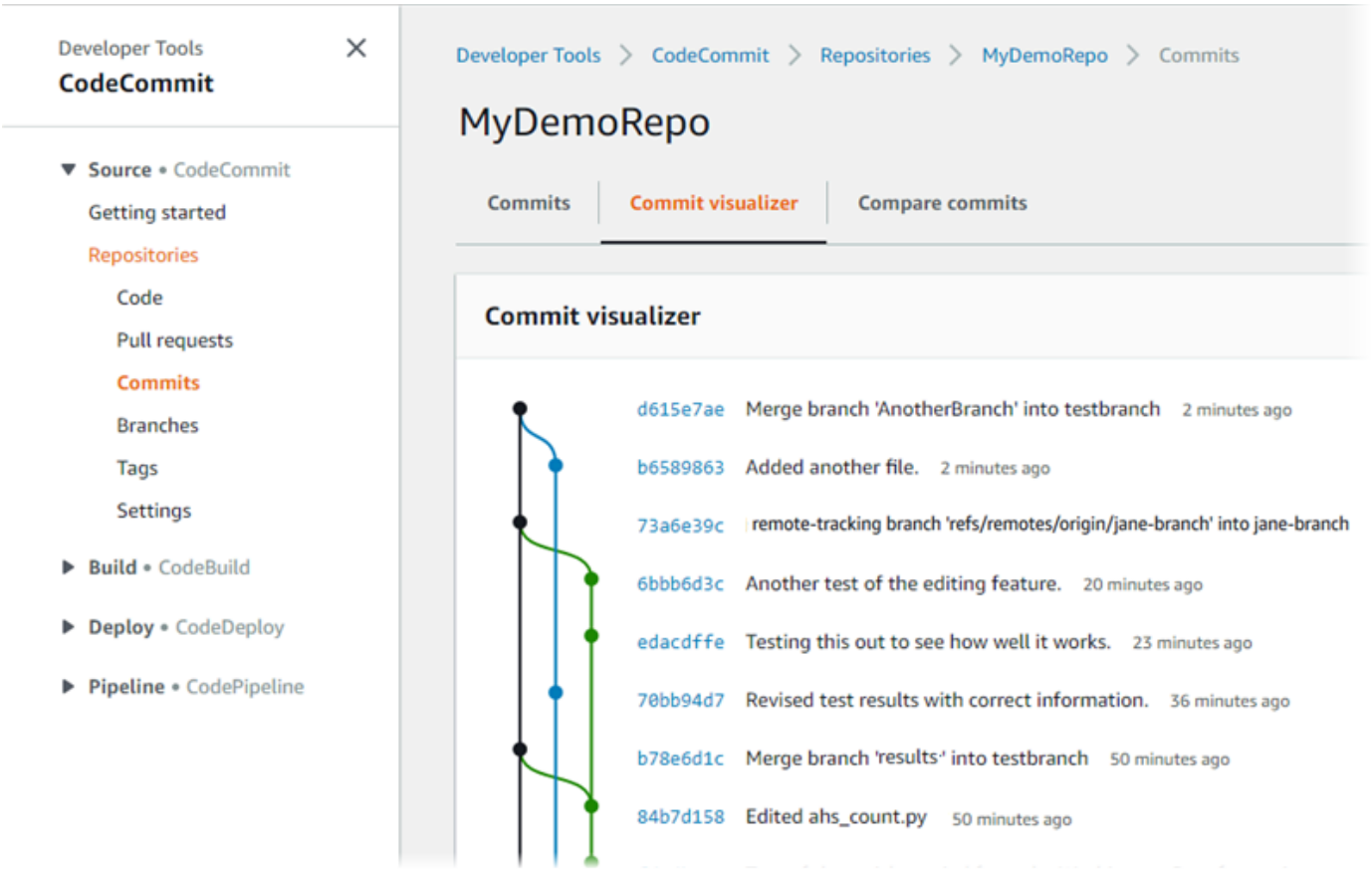
   有关更多信息，请参阅 评论提交。

8. 要查看任意两个提交说明符（包括标签、分支和提交 ID）之间的差异，请在导航窗格中选择提交，然后选择比较提交。



有关更多信息，请参阅 浏览存储库的提交历史记录 和 比较提交。

9. 在提交中，选择提交可视化工具选项卡。



此时会显示提交图，图中每个提交点的旁边显示其主题行。主题行显示限制为 80 个字符。

10. 要查看有关提交的更多详细信息，请选择其缩写的提交 ID。要呈现特定提交的图表，请在图表中选择该点。有关更多信息，请参阅 查看存储库的提交历史记录图表 。

步骤 4：在拉取请求中进行创建和协作

当您与其他用户在存储库中一起工作时，您可以协作编写代码并审核更改。您可以创建拉取请求，以便其他用户可以在分支中审核并注释您的代码更改。您还可以为拉取请求创建一个或多个审批规则。例如，您可以创建一个审批规则，要求在拉取请求合并之前，至少由另外两名用户对其进行审批。拉取请求经审批后，您可以将这些更改合并到其目标分支中。如果您为存储库设置了通知，则存储库用户可以收到有关存储库事件的电子邮件（例如，针对拉取请求，或在有人对代码评论时）。有关更多信息，请参阅 在 AWS CodeCommit 存储库中配置事件通知。

> **Important**
>
> 在创建拉取请求之前，必须先创建一个分支，其中包含要查看的代码更改。有关更多信息，请参阅 创建分支。

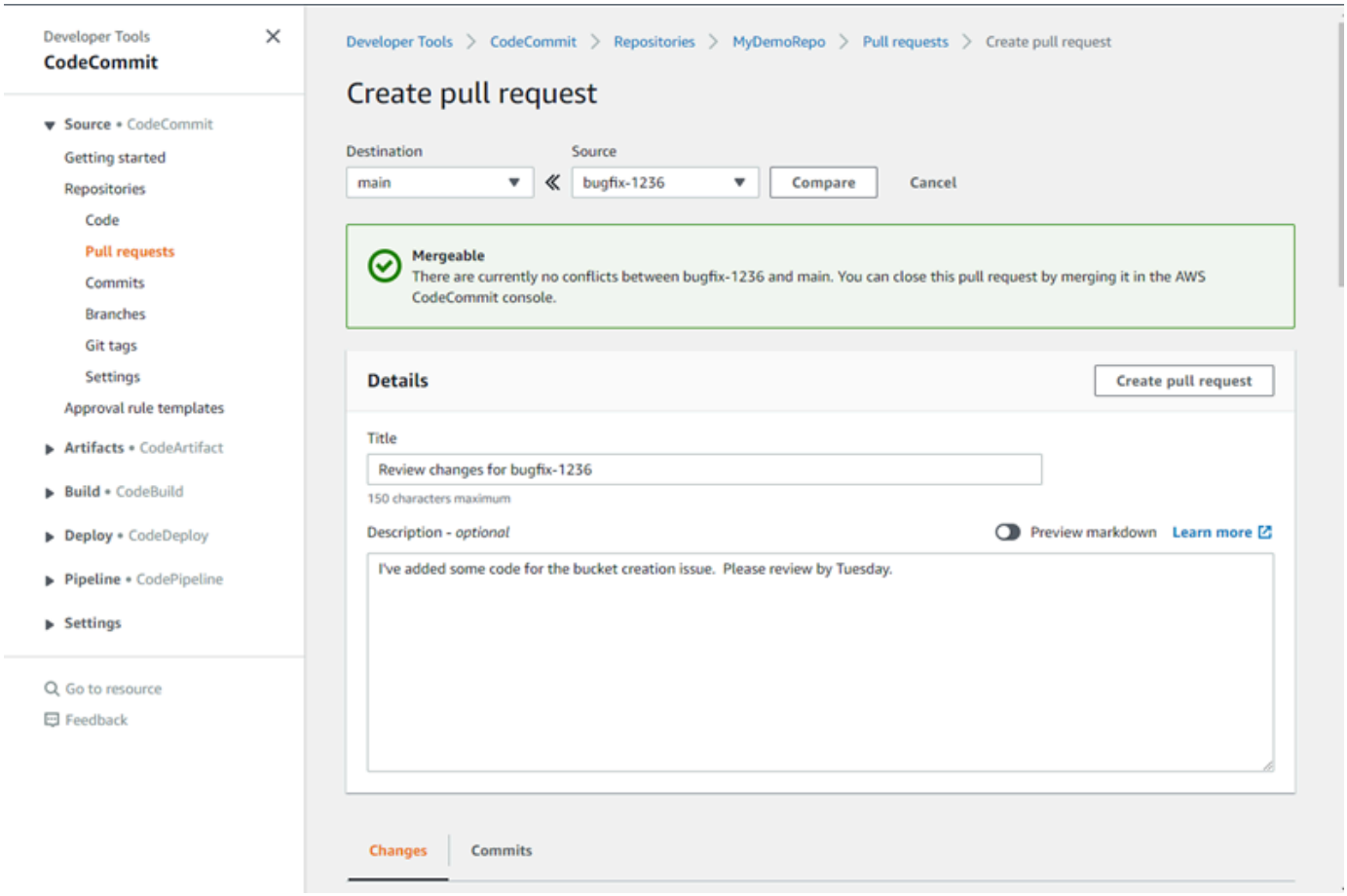
在拉取请求中进行创建和协作

1. 在导航窗格中，选择拉取请求。
2. 在拉取请求中，选择创建拉取请求。

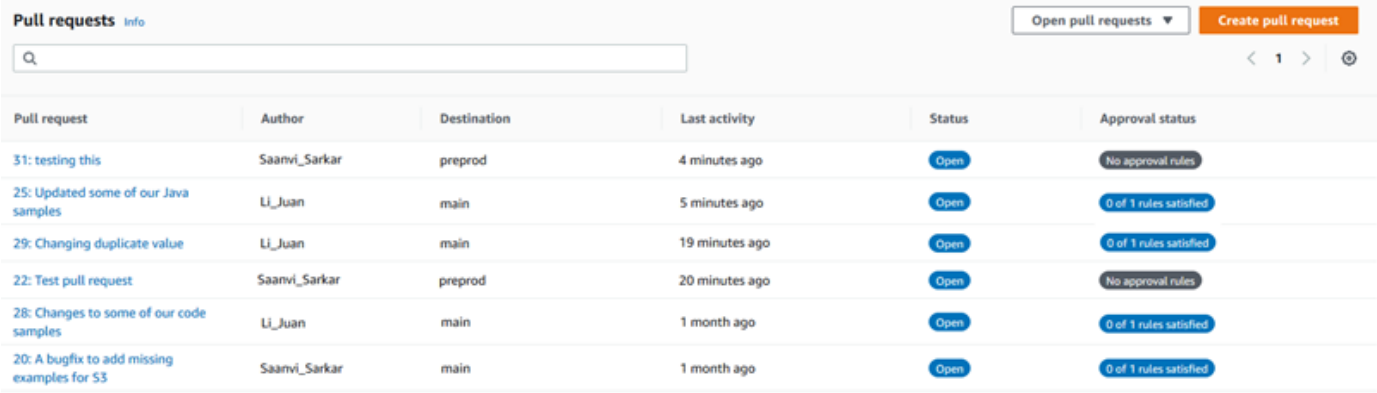
> **Tip**
>
> 您也可以从分支和代码创建拉取请求。

在 Create pull request 中，在 Source 中选择包含要审核的更改的分支。在 Destination (目标) 中，选择要在拉取请求关闭后将已审核的代码合并到的分支。选择 Compare。

3. 审核合并详细信息和更改，以确认拉取请求包含要审核的更改和提交。如果是这样，请在标题中输入此审核的标题。此标题显示在存储库的拉取请求列表中。在描述中，输入有关此审核的详细信息及对审核者有用的任何其他信息。选择创建。

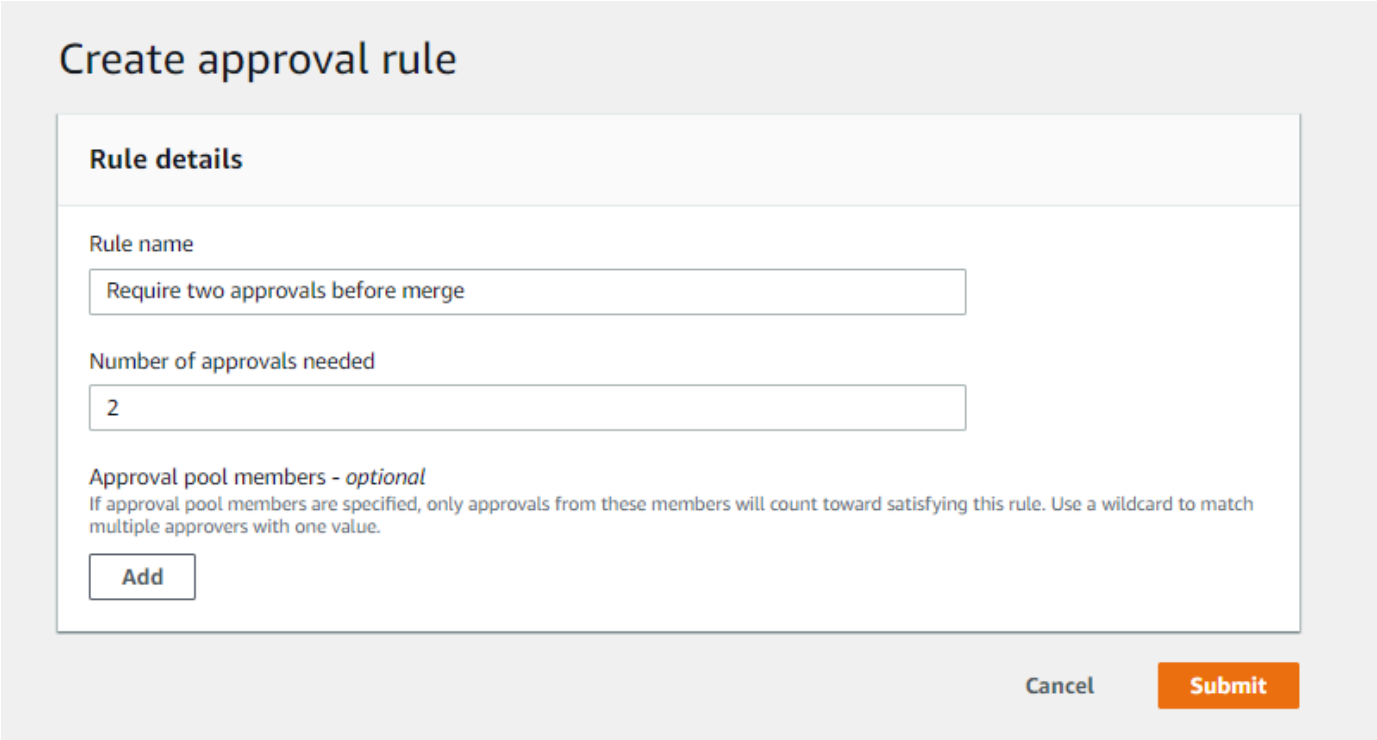


4. 拉取请求将显示在存储库的拉取请求列表中。您可以筛选视图，使其只显示打开的请求、关闭的请求、您创建的请求等。

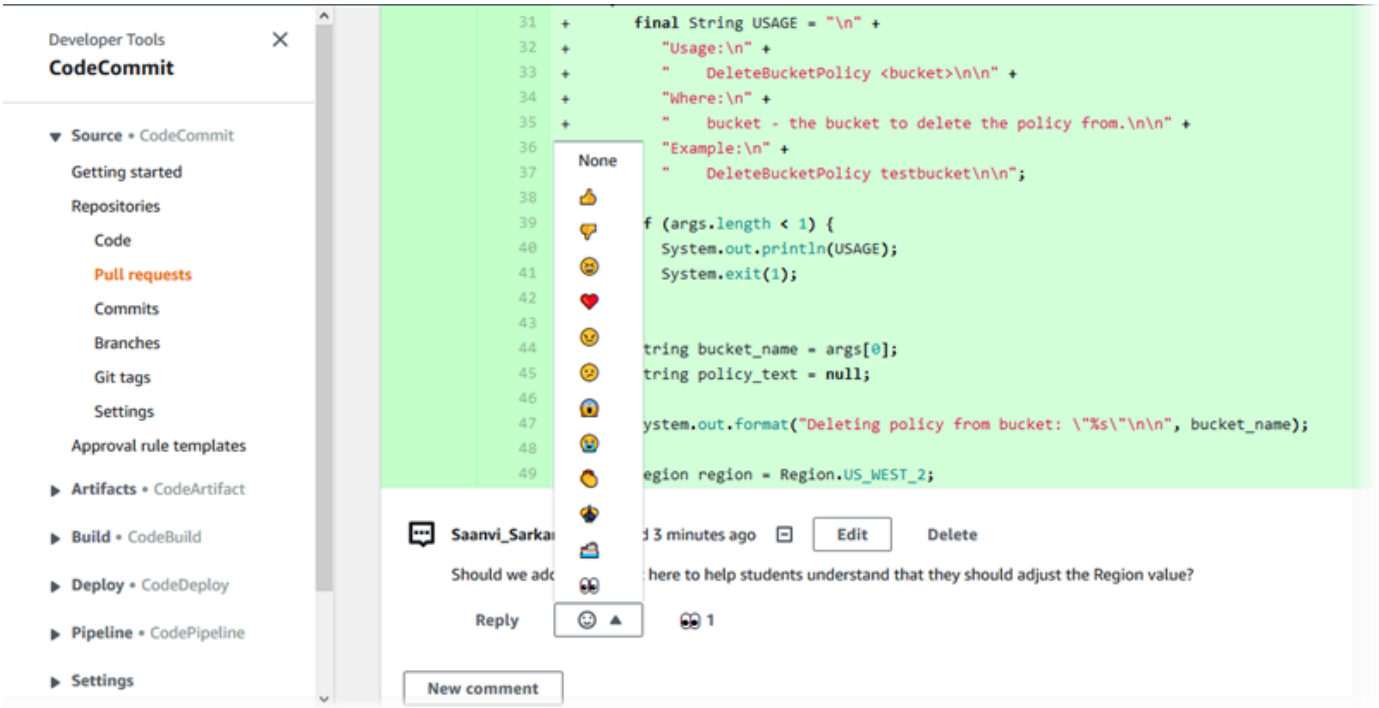


5. 您可以向拉取请求添加审批规则，以确保该请求在合并之前满足某些条件。要向拉取请求添加审批规则，请从列表中选择拉取请求。在 Approvals (审批) 选项卡上，选择 Create approval rule (创建审批规则)。

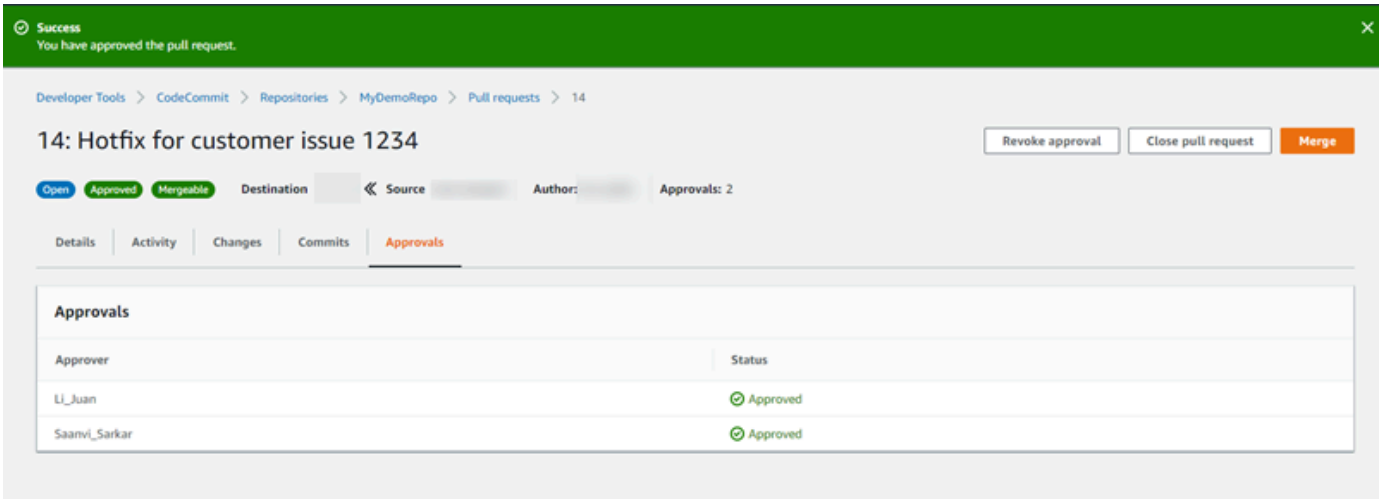
6. 在 Rule name (规则名称) 中，为规则指定一个描述性名称。例如，如果您希望拉取请求在合并之前，必须由两人对其进行审批，那么可以将该规则命名为 **`Require two approvals before merge`**。在 Number of approvals needed (需要的审批数量) 中，输入所需的数量 **2**。默认为 1。选择提交。要了解有关审批规则和审批池成员的更多信息，请参阅为拉取请求创建审批规则。



7. 如果已为存储库配置通知并选择向用户通知拉取请求事件，用户会收到有关新拉取请求的电子邮件。用户可以查看针对特定代码行、文件和拉取请求自身的更改和评论。他们还可以使用文字和表情符号回复评论。如有必要，可以将更改推送到拉取请求分支，此操作会更新拉取请求。



8. 如果您对请求中所做的更改感到满意，请选择 Approve (批准)。即使该拉取请求未配置审批规则，您也可以选择批准拉取请求。这样可清楚地记录您已审核拉取请求并批准更改。如果您改变主意，还可以选择撤销批准。

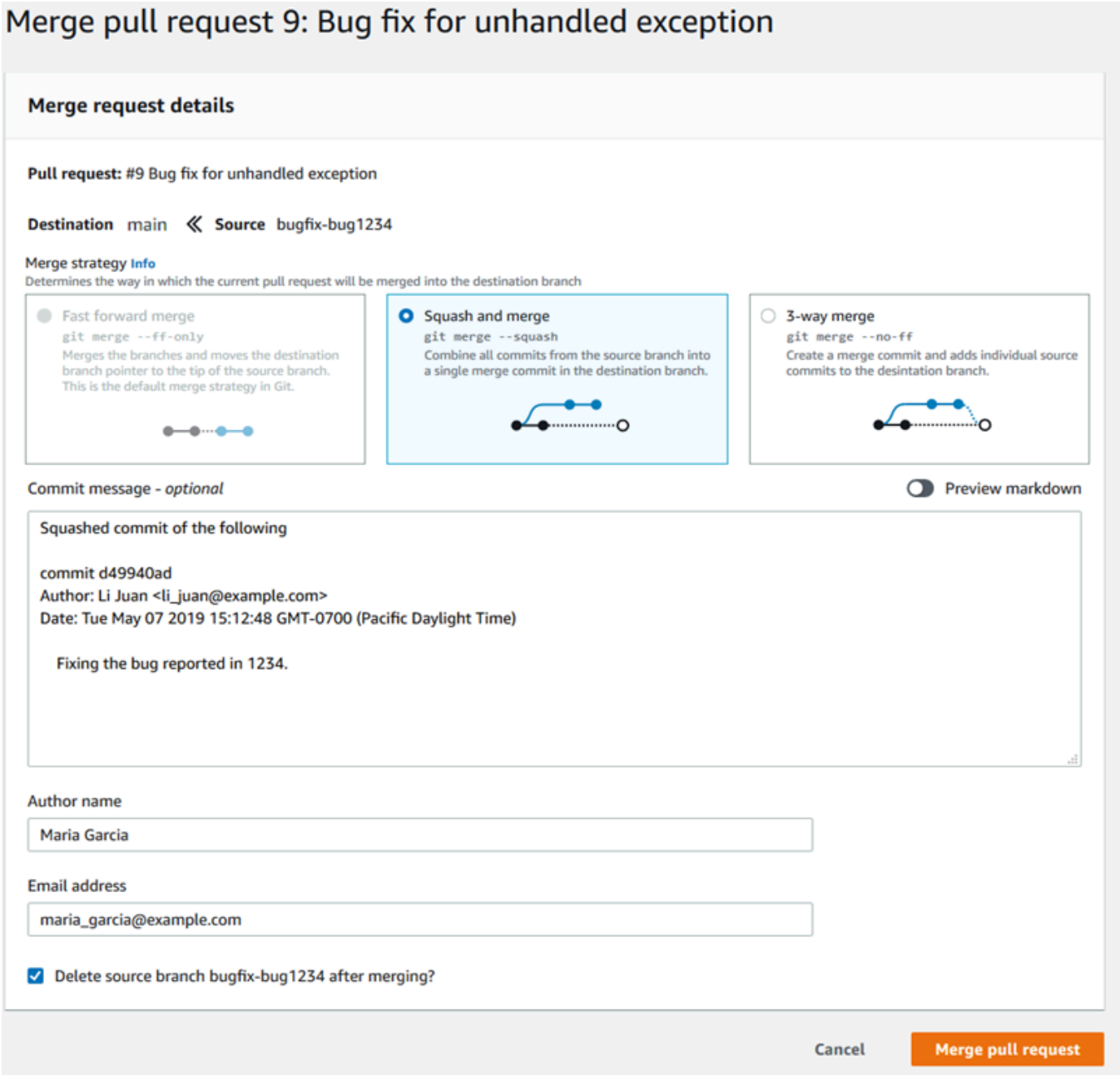


> **Note**
>
> 如果拉取请求由您创建，那么您无法批准该请求。

9. 当您对在拉取请求中审核并同意的所有代码更改感到满意后，执行以下操作之一：

   - 如果要关闭拉取请求而不合并分支，请选择 Close pull request (关闭拉取请求)。

- 如果要合并分支并关闭拉取请求，请选择 Merge (合并)。您可以在代码的可用合并策略之间进行选择，这些策略取决于源和目标分支之间的差异，以及在合并完成后是否自动删除源分支。在进行选择后，请选择 Merge pull request (合并拉取请求) 以完成合并。



- 如果分支中存在无法自动解决的合并冲突，则可以在 CodeCommit 控制台中解决这些冲突，也可以使用本地 Git 客户端合并分支，然后推送合并。有关更多信息，请参阅 解决 AWS CodeCommit 存储库中的拉取请求中的冲突。

> **Note**
>
> 您始终可以在本地存储库中使用 git merge 命令并推送更改以手动合并分支，包括拉取请求分支。

有关更多信息，请参阅 使用拉取请求 和 使用审批规则模板。

第 5 步：清理

如果您不再需要 CodeCommit 存储库，则应删除您在本练习中使用的 CodeCommit 存储库和其他资源，这样就不会继续向您收取存储空间费用。

> **Important**
>
> 并且无法撤消。删除该存储库后，您就无法再将其克隆到任何本地存储库或共享存储库，也无法再从任何本地存储库或共享存储库向其推送数据、从其拉取数据或执行任何 Git 操作。
> 如果您为存储库配置了通知，则删除存储库也会删除为存储库创建的 Amazon CloudWatch Events 规则。但不会删除用作该规则目标的 Amazon SNS 主题。
> 如果为存储库配置了触发器，则删除存储库不会删除您配置为这些触发器目标的 Amazon SNS 主题或 Lambda 函数。如果不需要这些资源，请务必将其删除。有关更多信息，请参阅 从存储库中删除触发器。

删除 CodeCommit 存储库

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要删除的存储库。如果您遵循本主题中的命名惯例，则将其命名MyDemoRepo。
3. 在导航窗格中，选择 Settings（设置）。
4. 在 Settings 页面上的 Delete repository 中，选择 Delete repository。
5. 键入 **`delete`**，然后选择删除。存储库将被永久删除。

步骤 6：后续步骤

既然您已经熟悉了它 CodeCommit 及其一些功能，可以考虑执行以下操作：

- 如果您不熟悉 Git， CodeCommit 或者想查看使用 Git 的示例 CodeCommit，请继续阅读本Git 和 CodeCommit 入门教程。
- 如果您想与 CodeCommit 仓库中的其他人合作，请参阅共享存储库。（如果您需要与其他 Amazon Web Services 账户中的用户共享您的存储库，请参阅使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限。）
- 如果要将存储库迁移到 CodeCommit，请按照中的步骤操作迁移到 CodeCommit。
- 如果需要将存储库添加到持续交付管道，请按照简单管道演练中的步骤操作。
- 如果您想进一步了解与之集成的产品和服务 CodeCommit，包括来自社区的示例，请参阅产品和服务集成。

Git 和 AWS CodeCommit 入门

如果您刚刚开始使用 Git 和 CodeCommit，本教程可帮助您学习一些简单的入门命令。如果您已熟悉 Git，可以跳过本教程并转到 入门 CodeCommit 。

在本教程中，您将创建一个存储库，该存储库代表 CodeCommit 存储库的本地副本，我们将该存储库称作本地存储库。

创建本地存储库后，您将对其进行一些更改，然后将更改发送（推送）到 CodeCommit 存储库。

您还将模拟团队环境：两名用户各自向其本地存储库提交更改，并将他们的更改推送到 CodeCommit 存储库。然后，这两名用户将更改从 CodeCommit 存储库拉取到各自的本地存储库中，查看对方做出的更改。

您还将在 CodeCommit 存储库中创建分支和标签，并管理某些访问权限。

完成本教程后，您应已完成有关 Git 和 CodeCommit 核心概念的充分练习，可以开始在自己的项目中使用它们。

完成前提条件和设置，包括：

- 向 IAM 用户分配权限。
- 将 CodeCommit 设置为使用 HTTPS、SSH 或 git-remote-codecommit 连接到存储库。有关这些选择的更多信息，请参阅 对 AWS CodeCommit 进行设置。
- 如果您想要使用命令行或终端完成所有操作，包括创建存储库，则还需要配置 AWS CLI。

主题

步骤 1：创建 CodeCommit 存储库

在此步骤中，您将使用 CodeCommit 控制台创建存储库。

如果您已有要使用的 CodeCommit 存储库，则可跳过该步骤。

> **Note**
>
> 根据您的使用情况，您可能需要为创建或访问存储库付费。有关更多信息，请参阅 CodeCommit 产品信息页面上的定价。

创建 CodeCommit 存储库

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 使用区域选择器选择要在其中创建存储库的 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 在存储库页面上，选择创建存储库。
4. 在 Create repository (创建存储库) 页面上的 Repository name (存储库名称)中，输入存储库的名称（例如，**MyDemoRepo**）。

   > **Note**
   >
   > 存储库名称区分大小写，且不能超过 100 个字符。有关更多信息，请参阅限制。

5. （可选）在描述中，输入描述（例如，**My demonstration repository**）。这可以帮助您及其他用户了解存储库的用途。
6. （可选）选择添加标签，向您的存储库添加一个或多个存储库标签（自定义属性标签，可帮助您组织和管理您的 AWS 资源）。有关更多信息，请参阅在中标记存储库 AWS CodeCommit。
7. （可选）展开其他配置，以指定是使用默认 AWS 托管式密钥还是您自己的客户托管密钥来加密和解密此存储库中的数据。如果您选择使用自己的客户托管密钥，则必须确保该密钥在您创建存储库的 AWS 区域可用，并且该密钥处于活跃状态。有关更多信息，请参阅AWS Key Management Service 和 AWS CodeCommit 存储库加密。
8. （可选）如果此存储库将包含 Java 或 Python 代码，并且您希望使用 CodeGuru Reviewer 对其进行分析，请选择启用适用于 Java 和 Python 的 Amazon CodeGuru Reviewer。CodeGuru Reviewer 使用多个机器学习模型来查找代码缺陷，并自动在拉取请求中提供改进和修复建议。有关更多信息，请参阅《Amazon CodeGuru Reviewer 用户指南》。
9. 选择创建。

> **Note**
>
> 本教程中的剩余步骤使用 `MyDemoRepo` 作为 CodeCommit 存储库的名称。如果您选择其他名称，请确保在本教程中通篇使用它。

有关创建存储库 (包括如何从终端或命令行创建存储库) 的更多信息，请参阅创建存储库。

步骤 2：创建本地存储库

在此步骤中，需要在本地计算机上设置一个本地存储库，以连接到您的存储库。为此，请在本地计算机上选择一个代表本地存储库的目录，然后使用 Git 在该目录中克隆并初始化一个空 CodeCommit 存储库的副本，并指定用于为您的提交添加注释的 Git 用户名和电子邮件地址。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 从列表中找到您要连接的存储库并选择此存储库。选择 Clone URL (克隆 URL)，然后选择克隆或连接到存储库时要使用的协议。此时将复制克隆 URL。

   - 如果您使用的是 IAM 用户的 Git 凭证或 AWS CLI 附带的凭证助手，请复制 HTTPS URL。
   - 如果您在本地计算机上使用 git-remote-codecommit 命令，请复制 HTTPS (GRC) URL。

- 如果您使用的是 IAM 用户的 SSH 公有密钥/私有密钥对，请复制 SSH URL。

> **Note**
>
> 如果看到欢迎页面，而不是存储库列表，说明在您登录的 AWS 区域中没有存储库与您的 AWS 账户关联。要创建存储库，请参阅the section called “创建存储库”或按照Git 和 CodeCommit 入门教程中的步骤进行操作。

4. （可选）我们建议您将本地 Git 客户端配置为使用 **`main`** 作为存储库默认分支的名称。本指南所有示例中的默认分支均使用此名称。如果您在控制台中进行首次提交，CodeCommit 也会使用此默认分支名称。运行以下命令为您的系统全局配置默认分支名称：

```
git config --global init.defaultBranch main
```

如果您更喜欢为所有存储库使用不同的默认分支名称，请将 **`main`** 替换为您的首选名称。本教程假定您的默认分支名称为 main。

如果您希望为不同的存储库使用不同的默认分支名称，则可以在本地 (--local) 设置此属性，而不是全局 (--global) 设置。

5. 在终端或命令提示符处，使用 git clone 命令并提供您在步骤 3 中复制的克隆 URL 来克隆存储库。您的克隆 URL 具体取决于您使用的协议和配置。例如，如果您要在美国东部（俄亥俄州）区域中使用 HTTPS 和 Git 凭证来克隆名为 *MyDemoRepo* 的存储库，则运行以下命令：

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

如果您将 HTTPS 与 git-remote-codecommit 结合使用：

```
git clone codecommit://MyDemoRepo my-demo-repo
```

如果您正在使用 SSH：

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

> **Note**
>
> 如果您在尝试克隆存储库时看到错误，则可能尚未完成本地计算机所需的设置。有关更多信息，请参阅对 AWS CodeCommit 进行设置。

步骤 3：创建您的第一个提交

在此步骤中，您将在本地存储库中创建第一个提交。为此，您需要在本地存储库中创建两个示例文件。使用 Git 暂存更改，然后将更改提交到您的本地存储库。

1. 使用文本编辑器在您的目录中创建以下两个示例文本文件。将这两个文件命名为 `cat.txt` 和 `dog.txt`：

```
cat.txt
-------
The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry,
 domesticated, and carnivorous mammal.
```

```
dog.txt
-------
The domestic dog (Canis lupus familiaris) is a canid that is known as man's best
 friend.
```

2. 运行 git config 以将占位符 *your-user-name* 和 *your-email-address* 表示的用户名和电子邮件地址添加到本地存储库。这样，就可以更轻松地标识您所做的提交：

```
git config --local user.name "your-user-name"
git config --local user.email your-email-address
```

3. 如果您在创建本地存储库时没有全局设置默认分支名称，请运行以下命令，将默认分支名称设置为 **`main`**：

```
git config --local init.defaultBranch main
```

4. 运行 git add 暂存更改：

```
git add cat.txt dog.txt
```

5. 运行 git commit 提交更改：

```
git commit -m "Added cat.txt and dog.txt"
```

> **Tip**
>
> 要查看刚刚创建的提交的详细信息，请运行 git log。

步骤 4：推送您的第一个提交

在此步骤中，您将把提交从本地存储库推送到 CodeCommit 存储库。

运行 git push，通过 Git 用于 CodeCommit 存储库的默认远程存储库名称 (`origin`) 从本地存储库的默认分支 (`main`) 推送提交：

```
git push -u origin main
```

> **Tip**
>
> 将文件推送到 CodeCommit 存储库后，可以使用 CodeCommit 控制台查看内容。有关更多信息，请参阅浏览存储库中的文件。

步骤 5：共享 CodeCommit 存储库并推送和拉取另一个提交

在此步骤中，您将与团队成员共享有关 CodeCommit 存储库的信息。团队成员利用这些信息获取本地副本，对其进行一些更改，然后将修改后的本地副本推送到 CodeCommit 存储库。然后，您将更改从 CodeCommit 存储库拉取到本地存储库。

在本教程中，您可以通过使用 Git 另外创建一个目录 (独立于您在步骤 2 中创建的目录) 来模拟其他用户。（该目录通常位于其他计算机上。）此新目录是您的 CodeCommit 存储库的副本。您对现有目录或该新目录所做的任何更改都是独立进行的。确定对这些目录所做的更改的唯一方法是从 CodeCommit 存储库拉取。

我们将现有目录称作本地存储库，将新目录称作共享存储库 (即使它们位于同一台本地计算机上)。

在新目录中，您将获取 CodeCommit 存储库的一个单独副本。然后，添加一个新的示例文件，将更改提交到共享存储库，然后将共享存储库中的提交推送到您的 CodeCommit 存储库。

最后，将存储库中的更改拉取到本地存储库，然后浏览它以查看其他用户提交的更改。

1. 切换到 /tmp 目录或 c:\temp 目录。
2. 运行 git clone 在共享存储库中提取一份存储库副本：

   对于 HTTPS：

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

   对于将 HTTPS 与 git-remote-codecommit 一起使用：

   ```
   git clone codecommit://MyDemoRepo shared-demo-repo
   ```

   对于 SSH：

   ```
   git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo shared-
   demo-repo
   ```

   > **Note**
   >
   > 在 Windows 操作系统上使用 SSH 克隆存储库时，您可能需要将 SSH 密钥 ID 添加到连接字符串，如下所示：
   >
   > ```
   > git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
   > repos/MyDemoRepo my-demo-repo
   > ```
   >
   > 有关更多信息，请参阅适用于 Windows 上的 SSH 连接。

   在此命令中，MyDemoRepo 是您的 CodeCommit 存储库的名称，shared-demo-repo 是 Git 在 /tmp 目录或 c:\temp 目录中创建的目录的名称。Git 创建该目录后，会向 shared-demo-repo 目录中提取一份存储库副本。
3. 切换到 shared-demo-repo 目录：

   ```
   (For Linux, macOS, or Unix) cd /tmp/shared-demo-repo
   (For Windows) cd c:\temp\shared-demo-repo
   ```

4. 运行 git config 以添加由占位符 *other-user-name* 和 *other-email-address* 表示的其他用户名和电子邮件地址。这样，就可以更轻松地标识其他用户进行的提交：

```
git config --local user.name "other-user-name"
git config --local user.email other-email-address
```

5. 使用文本编辑器在 shared-demo-repo 目录中创建以下示例文本文件。将该文件命名为 horse.txt：

```
horse.txt
-------
The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

6. 运行 git add 将更改暂存到共享存储库：

```
git add horse.txt
```

7. 运行 git commit 将更改提交到共享存储库：

```
git commit -m "Added horse.txt"
```

8. 运行 git push，通过 Git 用于 CodeCommit 存储库的默认远程存储库名称 (origin) 从本地存储库的默认分支 (main) 推送初始提交：

```
git push -u origin main
```

9. 切换到本地存储库并运行 git pull，将共享存储库对 CodeCommit 存储库所做的提交拉取到本地存储库。然后运行 git log 查看从共享存储库发起的提交。

步骤 6：创建并共享分支

在此步骤中，您将在本地存储库中创建一个分支，进行一些更改，然后将分支推送到 CodeCommit 存储库。然后，将该分支从 CodeCommit 存储库拉取到共享存储库。

分支允许您独立开发存储库内容的不同版本。例如，在不影响团队成员工作的情况下使用新的软件功能。当该功能稳定时，可以将分支合并到软件的更稳定的分支中。

使用 Git 创建分支，然后将其指向您的第一个提交。您可以使用 Git 将该分支推送到 CodeCommit 存储库。然后，切换到您的共享存储库，使用 Git 将新分支拉取到共享本地存储库并查看该分支。

1. 在本地存储库中运行 git checkout，指定该分支的名称（例如 MyNewBranch）和您在本地存储库中的第一个提交的 ID。

   如果不知道提交 ID，可以运行 git log 获取它。确保提交使用的是您的用户名和电子邮件地址，而不是其他用户的用户名和电子邮件地址。这是为了模拟 main 是 CodeCommit 存储库的稳定版本，而 MyNewBranch 是用于某种新的、相对不稳定的功能的分支：

   ```
   git checkout -b MyNewBranch commit-ID
   ```

2. 运行 git push，将新分支从本地存储库发送到 CodeCommit 存储库：

   ```
   git push origin MyNewBranch
   ```

3. 现在，将分支提取到共享存储库并检查结果：

   1. 切换到共享存储库目录 (shared-demo-repo)。
   2. 提取新分支 (git fetch origin)。
   3. 确认已提取分支（git branch --all 显示存储库所有分支的列表）。
   4. 切换到新分支 (git checkout MyNewBranch)。
   5. 通过运行 git status 或 git branch 确认您已切换到 MyNewBranch 分支。输出将显示您所在的分支。在本示例中，输出应为 MyNewBranch。
   6. 查看分支中的提交列表 (git log)。

   以下是要调用的 Git 命令的列表：

   ```
   git fetch origin
   git branch --all
   git checkout MyNewBranch
   git branch or git status
   git log
   ```

4. 切换回 main 分支，查看其提交列表。Git 命令应如下所示：

   ```
   git checkout main
   git log
   ```

5. 切换到本地存储库中的 `main` 分支。您可以运行 git status 或 git branch。输出将显示您所在的分支。在本示例中，输出应为 `main`。Git 命令应如下所示：

```
git checkout main
git branch or git status
```

步骤 7：创建并共享标签

在此步骤中，您将在本地存储库中创建两个标签，将标签与提交相关联，并将标签推送到 CodeCommit 存储库。然后，将更改从 CodeCommit 存储库拉取到共享存储库。

标签用于给提交 (或分支，甚至是另一个标签) 起一个易于阅读的名称。例如，如果要将提交标记为 `v2.1`，则可如此操作。提交、分支或标签可以关联任意数量的标签，但一个标签只能关联一个提交、分支或标签。在本教程中，将一个提交标记为 `release`，将另一个提交标记为 `beta`。

您使用 Git 创建标签，将 `release` 标签指向您进行的第一个提交，并将 `beta` 标签指向其他用户进行的提交。然后，使用 Git 将标签推送到 CodeCommit 存储库。接下来，切换到您的共享存储库，使用 Git 将标签拉取到共享本地存储库并查看标签。

1. 在本地存储库中运行 git tag，指定新标签的名称（例如 `release`）和您在本地存储库中的第一个提交的 ID。

   如果不知道提交 ID，可以运行 git log 获取它。确保提交使用的是您的用户名和电子邮件地址，而不是其他用户的用户名和电子邮件地址。这是为了模拟您的提交是 CodeCommit 存储库的稳定版本：

   ```
   git tag release commit-ID
   ```

   再次运行 git tag，将其他用户的提交与 `beta` 标签关联。这是模拟该提交用于某种新的、相对不稳定的功能的情况：

   ```
   git tag beta commit-ID
   ```

2. 运行 git push --tags，将标签发送到 CodeCommit 存储库。
3. 现在，将标签提取到共享存储库并检查结果：

   1. 切换到共享存储库目录 (shared-demo-repo)。

2. 提取新标签 (git fetch origin)。
3. 确认已提取标签（git tag 显示存储库的标签列表）。
4. 查看有关每个标签（git log release 和 git log beta）的信息。

以下是要调用的 Git 命令的列表：

```
git fetch origin
git tag
git log release
git log beta
```

4. 也请尝试在本地存储库中执行该操作：

```
git log release
git log beta
```

步骤 8：设置访问权限

在此步骤中，您将向用户授予将共享存储库与 CodeCommit 存储库同步的权限。此为可选步骤。建议以下用户执行该步骤：有兴趣了解当用户使用 Git 凭证或结合使用 SSH 密钥对和 IAM 用户来访问 CodeCommit 存储库时如何控制对 CodeCommit 存储库的访问。

> **Note**
>
> 如果您使用的是联合访问、临时凭证或 Web 身份提供者验证（例如 IAM Identity Center），请为您的身份提供者设置用户、访问权限和权限，然后使用 git-remote-codecommit。有关更多信息，请参阅 使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤 和 使用轮换凭证连接到 AWS CodeCommit 存储库。

为此，您可以使用 IAM 控制台创建用户，默认情况下，该用户没有将共享存储库与 CodeCommit 存储库同步的权限。您可以运行 git pull 来验证这一点。如果新用户没有同步权限，则该命令不起作用。然后，返回到 IAM 控制台，应用允许该用户使用 git pull 的策略。随后，您可以再次运行 git pull 进行验证。

此步骤假定您具备在 Amazon Web Services 账户中创建 IAM 用户的权限。如果您不具备这些权限，则无法执行此步骤中的过程。请跳到步骤 9：清除并清理学习教程时所使用的资源。

1. 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/。

   请务必用您在设置 中使用的用户名和密码登录。

2. 在导航窗格中选择用户，然后选择创建新用户。

3. 在第一个输入用户名称框中，输入示例用户名（例如，**JaneDoe-CodeCommit**）。选中为每个用户生成访问密钥框，然后选择创建。

4. 选择显示用户安全凭证。记下访问密钥 ID 和秘密访问密钥，或选择 Download Credentials。

5. 按照适用于使用 Git 凭证的 HTTPS 用户中的说明进行操作以生成并提供 IAM 用户的凭证。

   如果需要使用 SSH，请按照SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥或步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥中的说明操作，设置用户及公钥和私钥。

6. 运行 git pull。此时会出现下面的错误：

   对于 HTTPS：

   ```
   fatal: unable to access 'https://git-codecommit.us-
   east-2.amazonaws.com/v1/repos/repository-name/': The requested URL
   returned error: 403.
   ```

   对于 SSH：

   ```
   fatal: unable to access 'ssh://git-codecommit.us-east-2.amazonaws.com/
   v1/repos/repository-name/': The requested URL returned error: 403.
   ```

   出现此错误的原因是，新用户没有将共享存储库与 CodeCommit 存储库同步的权限。

7. 返回到 IAM 控制台。在导航窗格中选择 Policies，然后选择 Create Policy。（如果 Get Started 按钮出现，选择此按钮，然后选择 Create Policy。）

8. 在 Create Your Own Policy 旁，选择 Select。

9. 在策略名称框中，输入名称（例如 **CodeCommitAccess-GettingStarted**）。

10. 在策略文档框中，输入以下内容，它允许 IAM 用户从与该 IAM 用户关联的任何存储库中拉取内容：

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Tip**
>
> 如果您希望该 IAM 用户能够将提交推送到与之关联的任何存储库，请改为输入：
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Action": [
>         "codecommit:GitPull",
>         "codecommit:GitPush"
>       ],
>       "Resource": "*"
>     }
>   ]
> }
> ```
>
> 有关您可以向用户授予的其他 CodeCommit 操作及资源权限的信息，请参阅 AWS CodeCommit 的身份验证和访问控制。

11. 在导航窗格中，选择用户。
12. 选择要附加策略的示例用户名（例如，**JaneDoe-CodeCommit**）。
13. 选择权限选项卡。
14. 在 Managed Policies 中，选择 Attach Policy。
15. 选择您刚刚创建的 **CodeCommitAccess-GettingStarted** 策略，然后选择 Attach Policy (附加策略)。
16. 运行 git pull。这一次，命令应能起作用，并显示 `Already up-to-date` 消息。

17. 如果您使用 HTTPS，请切换到您的原始 Git 凭证；或者，如果使用 git-remote-codecommit，请切换到您的常用配置文件。有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户的设置或使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤中的说明。

    如果使用的是 SSH，请切换到您的原始密钥。有关更多信息，请参阅 SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥或 步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥。

本教程到此结束。

步骤 9：清除

在此步骤中，您将删除在本教程中使用的 CodeCommit 存储库，以免产生存储空间使用费。

另外还需要删除您本地计算机上的本地存储库和共享存储库，删除 CodeCommit 存储库后就不需要它们了。

> **Important**
>
> 删除该存储库后，您就无法再将其克隆到任何本地存储库或共享存储库，也无法再从任何本地存储库或共享存储库向其推送数据或从其拉取数据。并且无法撤消。

删除 CodeCommit 存储库（控制台）

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在控制面板页面的存储库列表中，选择 MyDemoRepo。
3. 在导航窗格中，选择设置。
4. 在 Settings 页面上的 Delete repository 中，选择 Delete repository。
5. 在 Type the name of the repository to confirm deletion (键入存储库名称以确认删除) 旁边的框中，输入 **MyDemoRepo**，然后选择 Delete (删除)。

删除 CodeCommit 存储库 (AWS CLI)

运行 delete-repository 命令：

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

删除本地存储库和共享存储库

对于 Linux、macOS 或 Unix：

```
cd /tmp
rm -rf /tmp/my-demo-repo
rm -rf /tmp/shared-demo-repo
```

对于 Windows：

```
cd c:\temp
rd /s /q c:\temp\my-demo-repo
rd /s /q c:\temp\shared-demo-repo
```

产品和服务与 AWS CodeCommit

默认情况下， CodeCommit 已与许多 AWS 服务集成。您也可以 CodeCommit 与之外的产品和服务一起使用 AWS。以下信息可以帮助您进行配置 CodeCommit以与您使用的产品和服务集成。

> **Note**
>
> 通过与集成，您可以自动生成提交并将其部署到 CodeCommit 存储库 CodePipeline。要了解更多信息，请按照《 DevOps 入门指南》中的步骤操作。AWS

主题

与其他 AWS 服务集成

CodeCommit 已与以下 AWS 服务集成：

AWS Amplify	借助 AWS Amplify，您可以轻松创建、配置和实施由 AWS提供支持的可扩展移动应用程序。Amplify 不仅可以无缝预置和管理移动后端，还能提供简单的框架来轻松将后端与 iOS、Android、Web 和 React Native 前端集成。另外，Amplify 还可以自动执行前端和后端的应用程序发布流程，这可让您更快地交付功能。 您可以在 Amplify 控制台中连接您的 CodeCommit 仓库。在您授权 Amplify 控制台后，Amplify 会从存储库提供者那里获取访问令牌，但它不会将该令牌存储在服务器上。 AWS Amplify 仅使用安装在特定存储库中的部署密钥访问存储库。 了解更多：

	• AWS Amplify 用户指南 • 开始使用
AWS Cloud9	AWS Cloud9 包含一套工具，可用于在云中对软件进行编码、生成、运行、测试、调试和发布。此工具集合称为 AWS Cloud9 集成开发环境或 IDE。 您可以通过 Web 浏览器访问 AWS Cloud9 IDE。 IDE 提供丰富的代码编辑体验，对多种编程语言和运行时调试程序的支持以及内置终端。 了解更多： • AWS Cloud9 用户指南 • AWS CodeCommit 的示例 AWS Cloud9 • 将 AWS Cloud9 与 AWS CodeCommit 集成
AWS CloudFormation	AWS CloudFormation是一项服务，可帮助您对 AWS 资源进行建模和设置，这样您就可以减少管理这些资源的时间，将更多的时间集中在应用程序上。您可以创建一个描述资源（包括 CodeCommit 存储库）的模板，并 AWS CloudFormation 负责为您配置和配置这些资源。 了解更多： • AWS CodeCommit 存储库资源页面

AWS CloudTrail	CloudTrail捕获 Amazon Web Services 账户或代表其发起的 AWS API 调用和相关事件，并将日志文件传输到您指定的亚马逊 S3 存储桶。您可以配置 CloudTrail 为捕获来自 AWS CodeCommit 控制台的 API 调用、来自本 AWS CLI地 Git 客户端和 CodeCommit API 的 CodeCommit 命令。 了解更多： • 使用 AWS CodeCommit 记录 AWS CloudTrail API 调用
亚马逊 CloudWatch 活动	CloudWatch E@@ v ents 提供近乎实时的系统事件流，这些事件描述了 AWS 资源的变化。使用可以快速设置的简单规则，您可以匹配事件并将它们路由到一个或多个目标函数或流。CloudWatch 事件在发生时就会意识到操作变化。 CloudWatch 事件通过发送消息以响应环境、激活函数、进行更改和捕获状态信息来响应这些操作变化并在必要时采取行动。 您可以将 CloudWatch 事件配置为监控 CodeCommit 存储库并响应存储库事件，方法是定位其他 AWS 服务（例如 Amazon Simple Queue Service、Amazon Kinesis 等）中的流、函数 AWS Lambda、任务或其他进程。 了解更多： • CloudWatch 活动用户指南 • AWS CodeCommit 事件 • 博客文章：使用 Amazon Ev CloudWatch ents 和 jGit 构建无服务器 AWS CodeCommit 工作流程

AWS CodeBuild	CodeBuild是云端完全托管的构建服务，它可以编译您的源代码、运行单元测试并生成随时可以部署的工件。您可以将要生成的源代码和构建规范存储在 CodeCommit存储库中。您可以 CodeBuild 直接与一起使用 CodeCommit，也可以将两者合 CodeBuild CodeCommit 并在持续交付管道中 CodePipeline。 了解更多： • 计划构建 • 创建构建项目 • CodePipeline 与一起使用 AWS CodeBuild 来运行构建
Amazon CodeGuru Reviewer	Amazon CodeGuru Reviewer 是一项自动代码审查服务，它使用程序分析和机器学习来检测 Java 或 Python 代码中的常见问题并推荐修复方法。您可以将亚马逊 Web Services 账户中的存储库与 CodeGuru Reviewer 关联起来。当你这样做时， CodeGuru Reviewer 会创建一个服务相关角色，允许 CodeGuru Reviewer 分析在建立关联后创建的所有拉取请求中的代码。 了解更多： • 将 AWS CodeCommit 存储库与 Amazon CodeGuru Reviewer 关联或取消关联 • Amazon CodeGuru Reviewer 用户指南

AWS CodePipeline	CodePipeline是一项持续交付服务，可用于对发布软件所需的步骤进行建模、可视化和自动化。您可以配置 CodePipeline 为在管道中使用 CodeCommit 存储库作为源操作，并自动构建、测试和部署您的更改。 了解更多： • 使用和进行简单的管道演练 CodePipeline AWS CodeCommit • 迁移到带有 CodeCommit存储库的管道的 Amazon CloudWatch 事件变更检测 • 用于自动启动管道的更改检测方法
AWS CodeStar	AWS CodeStar是一项基于云的服务，用于创建、管理和处理软件开发项目 AWS。您可以使用 AWS CodeStar 项目快速开发、构建和部署应用程序。 AWS AWS CodeStar 项目为您的项目开发工具链创建和集成 AWS 服务，包括项目的 CodeCommit 存储库。 AWS CodeStar 还会向团队成员分配该项目的权限。这些权限是自动应用的，包括访问 CodeCommit、创建和管理 Git 凭证的权限等。 您可以像配置任何其他存储库一样配置为 AWS CodeStar 项目创建的 CodeCommit 存储库，方法是使用 AWS CodeCommit 控制台 AWS CLI、本地 Git 客户端和 CodeCommit API 中的 CodeCommit 命令。 了解更多： • 使用存储库 • 处理 AWS CodeStar 项目 • 与 AWS CodeStar 团队协作

AWS Elastic Beanstalk	Elasti c Beanstalk 是一项托管服务，可以轻松地 AWS 在云端部署和管理应用程序，而不必担心运行这些应用程序的基础架构。您可以使用 Elastic Beanstalk 命令行界面 (EB CLI) 直接从新的或现有存储库部署应用程序。 CodeCommit 了解更多： • 将 EB CLI 与 AWS CodeCommit配合使用 • 使用现有 AWS CodeCommit 存储库 • eb codesource (EB CLI 命令)
AWS Key Management Service	AWS KMS 是一项托管服务，可让您轻松创建和控制加密您的数据所用的加密密钥。默认情况下， CodeCommit AWS KMS 用于加密存储库。 了解更多： • AWS KMS 和加密
AWS Lambda	利用 Lambda，您可以运行代码而无需预置或管理服务器。您可以为调用 Lambda 函数以响应 CodeCommit 存储库事件的存储库配置触发器。 了解更多： • 为 Lambda 函数创建触发器 • AWS Lambda 开发人员指南

Amazon Simple Notification Service	Amazon SNS 是一项 Web 服务，可让应用程序、终端用户和设备即时发送和接收云通知。您可以为发送 Amazon SNS 通知以响应 CodeCommit 存储库事件的存储库配置触发器。您还可以使用 Amazon SNS 通知与其他 AWS 服务集成。例如，您可以使用 Amazon SNS 通知向 Amazon Simple Queue Service 队列发送消息。 了解更多： • 为 Amazon SNS 主题创建触发器 • Amazon Simple Notification Service 开发人员指南

来自社区的集成示例

以下各部分提供的链接指向博客文章、文章和社区提供的示例。

> **Note**
>
> 这些链接仅供参考，不应被视为全面的清单或对示例内容的认可。 AWS 对外部内容的内容或准确性概不负责。

主题

- 博客文章
- 代码示例

博客文章

- 以拉取请求批准者的 SonarQube 身份集成 AWS CodeCommit

  了解如何创建需要成功进行 SonarQube 质量分析才能合并拉取请求的 CodeCommit 存储库。

  发布时间：2019 年 12 月 12 日

- 迁移到 AWS CodeCommitAWS CodePipeline、迁 AWS CodeBuild 出和迁出 GitLab

  了解如何使用 AWS CodePipeline 和将多个存储库迁移 AWS CodeCommit 到 GitLab并设置 CI/CD 管道。 AWS CodeBuild

  发布时间：2019 年 11 月 22 日

- GitFlow 使用 AWS CodePipeline、 AWS CodeCommitAWS CodeBuild、和 AWS CodeDeploy

  学习如何 GitFlow 使用 AWS CodePipeline、 AWS CodeCommit AWS CodeBuild、和来实现 AWS CodeDeploy。

  发布时间：2019 年 2 月 22 日

- AWS CodeCommit 在多个 AWS 账户中使用 Git

  了解如何跨多个 Amazon Web Services 账户管理 Git 配置。

  发布时间：2019 年 2 月 12 日

- 使用和验证 AWS CodeCommit 拉取 AWS CodeBuild请求 AWS Lambda

  了解如何使用 AWS CodeCommit AWS CodeBuild、和验证拉取请求 AWS Lambda。通过在将提议的更改合并到默认分支之前对其进行测试，您可以帮助确保拉取请求的高质量，捕捉任何潜在的问题，并提高开发人员对其更改的信心。

  发布时间：2019 年 2 月 11 日

- 将联合身份与配合使用 AWS CodeCommit

  了解如何 AWS CodeCommit 使用企业中使用的身份访问存储库。

  发布日期：2018 年 10 月 5 日

- 完善对分支机构的访问权限 AWS CodeCommit

  了解如何通过创建和应用使用上下文键的 IAM policy 来限制对存储库分支的提交。

  发布时间：2018 年 5 月 16 日

- 使用 AWS Fargate 在区域之间复制 AWS CodeCommit 存储库

  了解如何使用无服务器架构设置 CodeCommit 存储库从一个 AWS 区域到另一个区域的连续复制。

  发布时间：2018 年 4 月 11 日

- 分发您的 AWS OpsWorks for Chef Automate 基础架构

了解如何使用 CodePipeline、 CodeCommit CodeBuild、和 AWS Lambda 来确保食谱和其他配置一致地部署在一台或多台 Chef 服务器上的两台或多台 Chef 服务器上。 AWS 区域

发布时间：2018 年 3 月 9 日

- 花生酱和巧克力：具有 AWS CodeCommit的 Azure 函数 CI/CD 管道

  学习如何创建 PowerShell基于 Azure Functions 的 CI/CD 管道，将代码存储在存储库中。 CodeCommit

  发布时间：2018 年 2 月 19 日

- 使用 AWS CodePipeline、、、、 AWS CodeCommitAWS CodeBuild Amazon ECR 和 Kubernetes 持续部署到 Kubernetes AWS Lambda

  学习如何使用 Kubernetes 并 AWS 结合使用，为基于容器的应用程序创建完全托管的持续部署管道。

  发布时间：2018 年 1 月 11 日

- 使用 P AWS CodeCommit ull Requests 请求代码审查和讨论代码

  学习如何使用拉取请求来审查、评论和交互式迭代仓库中的代码更改。 CodeCommit

  发布时间：2017 年 11 月 20 日

- 使用 Amazon CloudWatch Events 和 jGit 构建无服务器 AWS CodeCommit 工作流程

  学习如何创建 CloudWatch 事件规则，这些规则使用存储 CodeCommit库事件和其他 AWS 服务中的目标操作来处理存储库中的更改。示例包括对提交强制执行 Git 提交消息策略、复制存储库以及将 CodeCommit存储库备份到 Amazon S3 的 AWS Lambda 函数。 CodeCommit

  发布时间：2017 年 8 月 3 日

- 迁移到 AWS CodeCommit

  在从使用另一个 Git 存储库迁移到使用 CodeCommit 时，学习如何将代码推送到两个存储库 SourceTree。

  发布时间：2016 年 9 月 6 日

- 使用 Appium、 AWS CodeCommit、Jenkins 和 AWS Device Farm

  了解如何使用 Appium、、 CodeCommit Jenkins 和 Device Farm 为移动设备创建持续测试流程。

发布时间：2016 年 2 月 2 日

- AWS CodeCommit 使用多个亚马逊 Web Services 账户中的 Git 存储库

  了解如何克隆您的 CodeCommit 存储库，并在一个命令中将凭证助手配置为使用特定的 IAM 角色连接到该存储库。

  发布时间：2015 年 11 月

- 整合 AWS OpsWorks 和 AWS CodeCommit

  了解 AWS OpsWorks 如何自动从 CodeCommit中获取 Apps 和 Chef 食谱。

  发布时间：2015 年 8 月 25 日

- 使用 AWS CodeCommit 和 GitHub 凭证助手

  了解如何配置您的 gitconfig 文件以同时使用 CodeCommit 和 GitHub凭据助手。

  发布时间：2015 年 9 月

- AWS CodeCommit 从 Eclipse 中使用

  学习如何使用 Eclipse 中的 eGit 工具来使用。 CodeCommit

  发布时间：2015 年 8 月

- AWS CodeCommit with Amazon EC2 Role Credentials

  了解在配置对 CodeCommit 存储库的自动代理访问权限时如何使用 Amazon EC2 的实例配置文件。

  发布时间：2015 年 7 月

- AWS CodeCommit 与 Jenkins 集成

  学习如何使用 CodeCommit 和 Jenkins 来支持两个简单的持续集成 (CI) 场景。

  发布时间：2015 年 7 月

- AWS CodeCommit 与审查委员会集成

  学习如何使用审查委员会代码审查系统 CodeCommit 集成到开发工作流程中。

  发布时间：2015 年 7 月

代码示例

以下是 CodeCommit 用户可能感兴趣的代码示例。

- 用于定期删除 OS X 证书存储中的缓存凭证的 Mac OS X 脚本

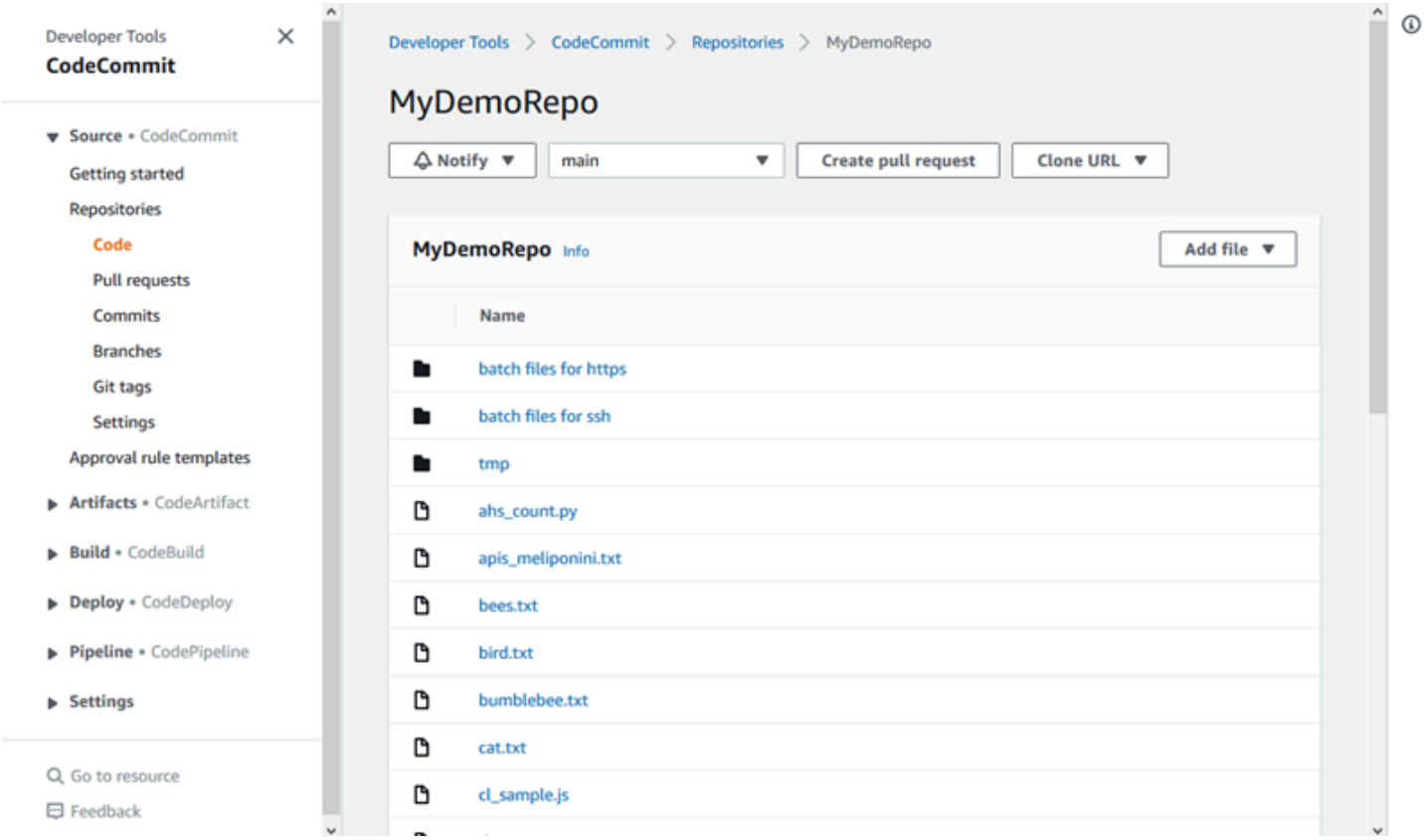
  如果您在 Mac OS X CodeCommit 上使用凭据助手，则可能已经熟悉了缓存凭据的问题。该脚本演示了一种解决方案。

  作者：Nico Coetzee

  发布时间：2016 年 2 月

使用中的存储库 AWS CodeCommit

存储库是中的基本版本控制对象 CodeCommit。您的项目代码和文件安全地存储在这里。它还储存您的从首次提交到最新更改的项目历史记录。您可以与其他用户共享存储库，从而进行项目协作。如果您向仓库添加 AWS 标签，则可以设置通知，以便仓库用户收到有关事件的电子邮件（例如，其他用户评论代码）。您也可以更改存储库的默认设置，浏览其内容等。您可以为存储库创建触发器，以使代码推送或其他事件能够触发电子邮件或代码函数等操作。您甚至可以在本地计算机上配置存储库 (本地存储库)，以便将更改推送到多个存储库。



在将更改推送到 CodeCommit 存储库之前，您必须在 Amazon Web Services 账户中配置 IAM 用户，或者设置联合访问权限或临时凭证的访问权限。有关更多信息，请参阅 步骤 1：的初始配置 CodeCommit 和 使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

有关在中使用存储库其他方面的信息 CodeCommit，请参阅使用文件使用拉取请求、使用提交、使用分支、和使用用户首选项。有关迁移到的信息 CodeCommit，请参阅迁移到 CodeCommit。

主题

- 创建 AWS CodeCommit 存储库
- Connect 连接到 AWS CodeCommit 存储库

- 共享存储 AWS CodeCommit 库
- 在 AWS CodeCommit 存储库中配置事件通知
- 在中标记存储库 AWS CodeCommit
- 管理 AWS CodeCommit 仓库的触发器
- 将 AWS CodeCommit 存储库与 Amazon CodeGuru Reviewer 关联或取消关联
- 查看 CodeCommit 存储库详情
- 更改 AWS CodeCommit 存储库设置
- 在本地存储库和 AWS CodeCommit 存储库之间同步更改
- 将提交推送到其他 Git 存储库
- 使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限
- 删除存储 AWS CodeCommit 库

创建 AWS CodeCommit 存储库

使用 AWS CodeCommit 控制台或 AWS Command Line Interface (AWS CLI) 创建空 CodeCommit 存储库。要在创建存储库后为其添加标签，请参阅为存储库添加标签。

以下说明假定您已完成设置 中的步骤。

> **Note**
>
> 根据您的使用情况，您可能需要为创建或访问存储库付费。有关更多信息，请参阅 CodeCommit 产品信息页面上的定价。

主题

- 创建存储库（控制台）
- 创建存储库 (AWS CLI)

创建存储库（控制台）

创建存储 CodeCommit 库

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。

2. 在区域选择器中，选择要创建存储库 AWS 区域 的位置。有关更多信息，请参阅 区域和 Git 连接端点。
3. 在存储库页面上，选择创建存储库。
4. 在创建存储库页面上的存储库名称中，为存储库输入名称。

> **Note**
>
> 存储库名称区分大小写。名称在 AWS 区域 中对于 Amazon Web Services 账户必须唯一。

5. （可选）在描述中，输入存储库的描述。这可以帮助您及其他用户了解存储库的用途。

> **Note**
>
> “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的应用程序开发人员，并且计划在 Web 浏览器中显示存储库描述字段，请参阅 CodeCommit API 参考。

6. （可选）选择 Add t ag，向存储库添加一个或多个存储库标签（可帮助您组织和管理 AWS 资源的自定义属性标签）。有关更多信息，请参阅 在中标记存储库 AWS CodeCommit。
7. （可选）展开其他配置以指定是使用默认密钥 AWS 托管式密钥 还是您自己的客户托管密钥来加密和解密此存储库中的数据。如果您选择使用自己的客户托管密钥，则必须确保该密钥在您创建存储库 AWS 区域 的地方可用，并且该密钥处于活动状态。有关更多信息，请参阅 AWS Key Management Service 和 AWS CodeCommit 存储库加密。
8. （可选）如果此存储库包含 Java 或 Python 代码，并且您想让 CodeGuru Reviewer 对其进行分析，请选择 “启用 Java 和 Python 版 Ama zon CodeGuru Reviewer”。 CodeGuru Reviewer 使用多个机器学习模型来查找代码缺陷，并对拉取请求提出改进和修复建议。有关更多信息，请参阅 Amazon CodeGuru Reviewer 用户指南。
9. 选择创建。

创建存储库后，您可以通过 CodeCommit 控制台或本地 Git 客户端，或者通过将 CodeCommit 仓库与您最喜欢的 IDE 集成来连接到该仓库并开始添加代码。有关更多信息，请参阅 对 AWS CodeCommit 进行设置。您也可以将存储库添加到持续交付管道中。有关更多信息，请参阅简单管道演练。

要获取有关新 CodeCommit 存储库的信息，例如克隆存储库时要使用的 URL，请从列表中选择存储库的名称，或者直接在存储库名称旁边选择要使用的连接协议。

要与其他用户共享该存储库，您必须向其发送用于克隆该存储库的 HTTPS 或 SSH 链接。确保他们具备访问该存储库所需的权限。有关更多信息，请参阅 共享存储库 和 AWS CodeCommit 的身份验证和访问控制。

创建存储库 (AWS CLI)

您可以使用 AWS CLI 来创建 CodeCommit 存储库。与控制台不同，如果使用 AWS CLI创建存储库，您可以为其添加标签。

1. 确保您已使用存储库 AWS CLI 的存在 AWS 区域 位置进行配置。要验证区域，请在命令行或终端中运行以下命令，并查看默认区域名称的信息。

   ```
   aws configure
   ```

   中存储库的默认区域名称必须与中存储库 AWS 区域 的名称相匹配 CodeCommit。有关更多信息，请参阅 区域和 Git 连接端点。

2. 运行 create-repository 命令，并指定：

   - 唯一标识 CodeCommit 存储库的名称（带`--repository-name`选项）。

     > **Note**
     >
     > 该名称必须在 Amazon Web Services 账户间保持唯一。

   - 关于 CodeCommit 存储库的可选评论（带`--repository-description`选项）。
   - 一个或多个可选的键值对，用作 CodeCommit 存储库的标签（带`--tags`选项）。
   - 加密和解密此存储库时使用的可选客户托管密钥。所有存储库都使用 AWS KMS中的密钥进行传输中和静态加密。如果未指定密钥，`aws/codecommit`则使用默认的 AWS 托管密钥。

   例如，要创建一个名为描述的 CodeCommit 存储库，`"My demonstration repository"`以及一个MyDemoRepo带有名为 Tea *m* 且值为 *Saanvi* 的密钥的标签，请使用此命令。

   ```
   aws codecommit create-repository --repository-name MyDemoRepo --repository-
   description "My demonstration repository" --tags Team=Saanvi
   ```

> **Note**
>
> “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的应用程序开发人员，并且计划在 Web 浏览器中显示存储库描述字段，请参阅 CodeCommit API 参考。

3. 如果成功，此命令会输出一个包含以下信息的 `repositoryMetadata` 对象：

   - 说明 (`repositoryDescription`)。
   - 系统生成的唯一 ID (`repositoryId`)。
   - 名称 (`repositoryName`)。
   - 与 CodeCommit 存储库关联的 Amazon Web Services 账户的 ID (`accountId`)。

   以下是基于上述示例命令的示例输出。

```
{
    "repositoryMetadata": {
        "repositoryName": "MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "lastModifiedDate": 1446071622.494,
        "repositoryDescription": "My demonstration repository",
        "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
        "defaultBranch": main,
        "kmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "creationDate": 1446071622.494,
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
        "accountId": "111111111111"
    }
}
```

> **Note**
>
> 不会在输出中返回创建存储库时添加的标签。要查看与存储库关联的标签列表，请运行 list-tags-for-resource 命令。

4. 记下 CodeCommit 存储库的名称和 ID。您需要它们来监视和更改有关 CodeCommit 存储库的信息，尤其是在您使用时 AWS CLI。

   如果您忘记了名称或 ID，请按照查看 CodeCommit 存储库详细信息 (AWS CLI)中的说明操作。

创建存储库后，您可以连接到该存储库并开始添加代码。有关更多信息，请参阅 连接存储库。您也可以将存储库添加到持续交付管道中。有关更多信息，请参阅简单管道演练。

Connect 连接到 AWS CodeCommit 存储库

首次连接到 CodeCommit 存储库时，通常会将其内容克隆到本地计算机。您也可以直接从 CodeCommit 控制台向存储库中添加文件和编辑存储库中的文件。或者，如果您已经有本地存储库，则可以将存储库添加为远程 CodeCommit存储库。本主题提供连接到 CodeCommit 存储库的说明。如果要将现有存储库迁移到 CodeCommit，请参阅迁移到 CodeCommit。

> **Note**
>
> 根据您的使用情况，您可能需要为创建或访问存储库付费。有关更多信息，请参阅 CodeCommit 产品信息页面上的定价。

主题

- 连接到 CodeCommit存储库的先决条件
- 通过克隆 CodeCommit 存储库来连接存储库
- 将本地存储库连接到存储库 CodeCommit

连接到 CodeCommit存储库的先决条件

在克隆存储 CodeCommit 库或将本地存储库连接到存储库之前，请执行以下操作： CodeCommit

- 您必须已为本地计算机配置了连接所需的软件和设置 CodeCommit。这包括安装和配置 Git。有关更多信息，请参阅 设置 和 Git 和 AWS CodeCommit 入门。
- 您必须拥有要连接的 CodeCommit 存储库的克隆 URL。有关更多信息，请参阅 查看存储库详细信息。

  如果您尚未创建 CodeCommit 存储库，请按照中的说明进行操作创建存储库，复制 CodeCommit 存储库的克隆 URL，然后返回此页面。

如果您有 CodeCommit 存储库但不知道其名称，请按照中的说明进行操作查看存储库详细信息。

- 您必须在本地计算机上有一个位置才能存储所连接 CodeCommit 存储库的本地副本。（此 CodeCommit 存储库的本地副本称为本地存储库。）然后，您可以切换到该位置并运行 Git 命令。例如，如果您出于测试目的而进行临时克隆，可以使用 `/tmp`（对于 Linux、macOS 或 Unix）或 `c:\temp`（对于 Windows）。这是这些示例中使用的目录路径。

> **Note**
>
> 您可以使用所需的任意目录。如果您要克隆存储库以供长期使用，请考虑从不是用于临时文件的工作目录中创建克隆。如果使用 `/tmp` 或 `c:\temp` 以外的目录，在按照以下说明操作时，请务必将示例中的目录替换成您自己指定的目录。

通过克隆 CodeCommit 存储库来连接存储库

如果您还没有本地存储库，请按照此过程中的步骤将 CodeCommit 存储库克隆到本地计算机。

1. 完成前提条件，包括设置 。

> **Important**
>
> 如果您尚未完成设置，则无法连接或克隆存储库。

2. 从 `/tmp` 目录或 `c:\temp` 目录，使用 Git 来运行 clone 命令。以下示例说明如何克隆在美国东部（俄亥俄州）地区命名的*MyDemoRepo*存储库。

   对于使用 Git 凭证的 HTTPS 或 AWS CLI随附的凭证辅助程序：

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

   对于使用 git-remote-codecommit 的 HTTPS，假设在 AWS CLI中配置了默认配置文件和 AWS 区域 ：

   ```
   git clone codecommit://MyDemoRepo my-demo-repo
   ```

   对于 SSH：

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

在此示例中，`git-codecommit.us-east-2.amazonaws.com`是存储库所在的美国东部（俄亥俄州）地区的 Git 连接点，`MyDemoRepo`代表您的 CodeCommit 仓库名称，并`my-demo-repo`表示 Git 在目录或目录中创建的`/tmpc:\temp`目录的名称。有关该支持 CodeCommit 以及这些支持 AWS 区域 的 Git 连接的更多信息 AWS 区域，请参阅区域和 Git 连接端点。

> **Note**
>
> 当您在 Windows 操作系统上使用 SSH 克隆存储库时，您可能需要将 SSH 密钥 ID 添加到连接字符串中，如下所示：
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 有关更多信息，请参阅 适用于 Windows 上的 SSH 连接 和 故障排除。

Git 创建目录后，会将 CodeCommit 仓库的副本拉到新创建的目录中。

如果 CodeCommit 存储库是新的或空的，您会看到一条消息，提示您正在克隆一个空存储库。这是预期行为。

> **Note**
>
> 如果您收到 Git 找不到 CodeCommit 存储库或您无权连接仓 CodeCommit 库的错误消息，请确保您已完成先决条件，包括向 IAM 用户分配权限以及为 Git 和 CodeCommit 本地计算机设置您的 IAM 用户证书。此外，请确保您指定了正确的存储库名称。

成功将本地存储库连接到 CodeCommit 仓库后，您现在可以开始从本地存储库中运行 Git 命令来创建提交、分支和标签，然后向仓库推送和拉取存储库了。 CodeCommit

将本地存储库连接到存储库 CodeCommit

如果您已经有本地存储库并且想要添加 CodeCommit存储库作为远程存储库，请完成以下步骤。如果您已经有一个远程存储库，并且想要将提交推送到 CodeCommit 另一个远程存储库，请按照中的步骤操作将提交推送到两个存储库。

1. 完成前提条件。
2. 在命令提示符或终端中，切换到本地 repo 目录并运行git remote add命令将 CodeCommit 存储库添加为本地存储库的远程存储库。

   例如，以下命令将昵称的遥控器添加**origin**到 https://git-codecommit.us-east-2.amazonaws.com/v1/repos/：MyDemoRepo

   对于 HTTPS：

   ```
   git remote add origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   对于 SSH：

   ```
   git remote add origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
   MyDemoRepo
   ```

   该命令不返回任何内容。
3. 要验证您是否已将 CodeCommit 存储库添加为本地存储库的远程存储库，请运行该git remote -v命令，该命令应创建类似于以下内容的输出：

   对于 HTTPS：

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   对于 SSH：

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)

origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

成功将本地存储库连接到 CodeCommit 仓库后，就可以开始从本地存储库运行 Git 命令来创建提交、分支和标签，以及向仓库推送和拉取存储库了。 CodeCommit

共享存储 AWS CodeCommit 库

创建 CodeCommit 存储库后，您可以与其他用户共享。首先，确定您在访问时是使用联合访问权限、临时证书还是网络身份提供商（例如 IAM Identity Center） CodeCommit，或者是否要对 IAM 用户使用 Git 证书或 SSH 密钥对。如果您使用的是前者，则需要为身份提供商设置用户、访问权限和权限，然后为用户提供使用 git-remote-codecommit 的说明。有关更多信息，请参阅 使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤 和 使用轮换凭证连接到 AWS CodeCommit 存储库。

您不能将 Git 凭证或 SSH 密钥对用于联合访问或身份提供商，但许多 IDE 在使用这些凭证时效果最好。在这种情况下，请确定在用户克隆并使用 Git 客户端或 IDE 连接您的存储库时向用户推荐哪个协议（HTTPS 或 SSH）。然后，将 URL 和连接信息发送给您要与之共享存储库的用户。根据您的安全要求，共享存储库时还可能需要创建 IAM 组、向该组应用托管式策略、编辑 IAM 策略以改进访问，或者创建并使用 IAM 角色。

> **Note**
>
> 在您向用户授予对存储库的控制台访问权限后，他们便可直接在控制台中添加或编辑文件，而无需设置 Git 客户端或其他连接。有关更多信息，请参阅 在 AWS CodeCommit 存储库中创建或添加文件 和 编辑 AWS CodeCommit 存储库中文件的内容。

编写这些说明时，假定您已完成设置 和创建存储库中的步骤。

> **Note**
>
> 根据您的使用情况，您可能需要为创建或访问存储库付费。有关更多信息，请参阅 CodeCommit 产品信息页面上的定价。

主题

选择与用户共享的连接协议

在中创建存储库时 CodeCommit，会生成两个端点：一个用于 HTTPS 连接，另一个用于 SSH 连接。两者都能提供安全的网络连接。您的用户可以使用这两种协议中的任何一种。不管您向用户推荐哪种协议，这两种终端节点都保持有效。

HTTPS 连接需要：

- Git 凭证，IAM 用户可在 IAM 中为自己生成该凭证。对于您的存储库用户来说，Git 凭证是最容易设置和使用的方法。
- 要担任的 AWS 访问密钥或角色，您的存储库用户必须在其凭据配置文件中对其进行配置。您可以配置 git-remote-codecommit（建议）或 AWS CLI中包含的凭证辅助程序。这些是根账户或联合身份用户可用的仅有的方法。

SSH 连接需要您的用户：

- 生成公有-私有密钥对。
- 存储公有密钥。
- 将公有密钥与其 IAM 用户相关联。
- 在本地计算机上配置已知主机文件。
- 在本地计算机上创建并维护配置文件。

由于这是一个更复杂的配置过程，因此我们建议您选择 HTTPS 和 Git 凭据进行连接 CodeCommit。

有关 HTTPS、SSH、Git、git-remote-codecommit 和远程存储库的更多信息，请参阅设置 、使用轮换凭证连接到 AWS CodeCommit 存储库或查阅 Git 文档。有关通信协议的一般概述以及每种协议如何与远程存储库通信的信息，请参阅服务器上的 Git - 协议。

> **Note**
>
> 尽管 Git 支持多种连接协议，但 CodeCommit 不支持使用不安全的协议（例如本地协议或通用 HTTP）的连接。

为存储库创建 IAM 策略

AWS 在 IAM 中为提供了三个托管策略 CodeCommit。这些策略无法编辑，应用于与您的 Amazon Web Services 账户关联的所有存储库。不过，您可以使用这些策略作为模板来创建自定义管理的策略，只将它们应用于要共享的存储库。您的客户管理策略可专门应用于要共享的存储库。有关更多信息，请参阅托管式策略和 IAM 用户和组。

> **Tip**
>
> 要对存储库的访问进行更精细的控制，您可以创建多个客户管理型策略，并向不同的 IAM 用户和组应用策略。

有关查看托管策略的内容和使用策略创建和应用权限的信息，请参阅AWS CodeCommit 的身份验证和访问控制。

为存储库创建客户托管策略

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在 Dashboard 导航区域中选择 Policies，然后选择 Create Policy。
3. 在创建策略页面上，选择导入管理型策略。
4. 在导入管理型策略页面的筛选策略中，输入 **AWSCodeCommitPowerUser**。选择策略名称旁的按钮，然后选择导入。
5. 在创建策略页面上，选择 JSON。将 CodeCommit 操作`Resource`行的 “*” 部分替换为 CodeCommit 存储库的 Amazon 资源名称 (ARN)，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> 要查找 CodeCommit 仓库的 ARN，请转到 CodeCommit 控制台，从列表中选择仓库名称，然后选择 “设置”。有关更多信息，请参阅 查看存储库详细信息。

若要将该策略应用到多个存储库，请通过指定其 ARN 将各个存储库添加为资源。在每个资源语句之间加上逗号，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

完成编辑后，选择查看策略。

6. 在 “查看策略” 页面的 “名称” 中，输入策略的新名称（例如，*AWSCodeCommitPowerUser-MyDemoRepo*）。（可选）提供此策略的描述。
7. 选择创建策略。

为存储库用户创建 IAM 组

要管理对您的存储库的访问，请为其用户创建一个 IAM 组、向该组添加 IAM 用户，然后附加在上一步中创建的客户管理型策略。或者，您也可以使用附加的客户管理型策略创建一个角色，然后让用户代入该角色。

如果您使用 SSH，则必须将另一个托管策略附加到 iamusersshKeys 组，该组是 IAM 托管策略，允许用户上传他们的 SSH 公钥并将其与他们用来连接的 IAM 用户关联。 CodeCommit

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在 Dashboard 导航区域中选择 Groups，然后选择 Create New Group。
3. 在 “设置组名” 页面的 “组名” 中，输入组的名称（例如 *MyDemoRepoGroup*），然后选择 “下一步”。请考虑在组名称中包含存储库名称。

> **Note**
>
> 该名称必须在 Amazon Web Services 账户间保持唯一。

4. 选中您在上一节中创建的客户托管策略旁边的复选框（例如，AWSCodeCommitPowerUser-MyDemoRepo）。
5. 在 Review 页面上，选择 Create Group。IAM 将使用已附加的指定策略创建此组。此组会显示在与您的 Amazon Web Services 账户关联的组列表中。
6. 从列表中选择您的组。
7. 在组摘要页面上，选择用户 选项卡，然后选择向组添加多个用户。在显示与您的 Amazon Web Services 账户关联的所有用户的列表中，选中要允许其访问 CodeCommit 存储库的用户旁边的复选框，然后选择添加用户。

> **Tip**
>
> 您可以使用搜索框快速地按名称查找用户。

8. 添加用户后，关闭 IAM 控制台。

与用户共享连接信息

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择存储库的创建 AWS 区域 位置。存储库特定于 AWS 区域。有关更多信息，请参阅 区域和 Git 连接端点。
3. 在 Repositories (存储库) 页面上，选择要共享的存储库。
4. 在克隆 URL 中，选择您希望用户使用的协议。这会复制连接协议的克隆 URL。
5. 向您的用户发送克隆 URL 以及任何其他说明，例如安装 AWS CLI、配置配置文件或安装 Git。请确保包含连接协议（例如 HTTPS）的配置信息。

以下示例电子邮件为在美国东部（俄亥俄州）(us-east-2) 地区使用 HTTPS 连接协议和 Git 凭据连接到 MyDemoRepo 存储库的用户提供信息。编写该电子邮件时假定用户已安装 Git 并能够熟练地使用它。

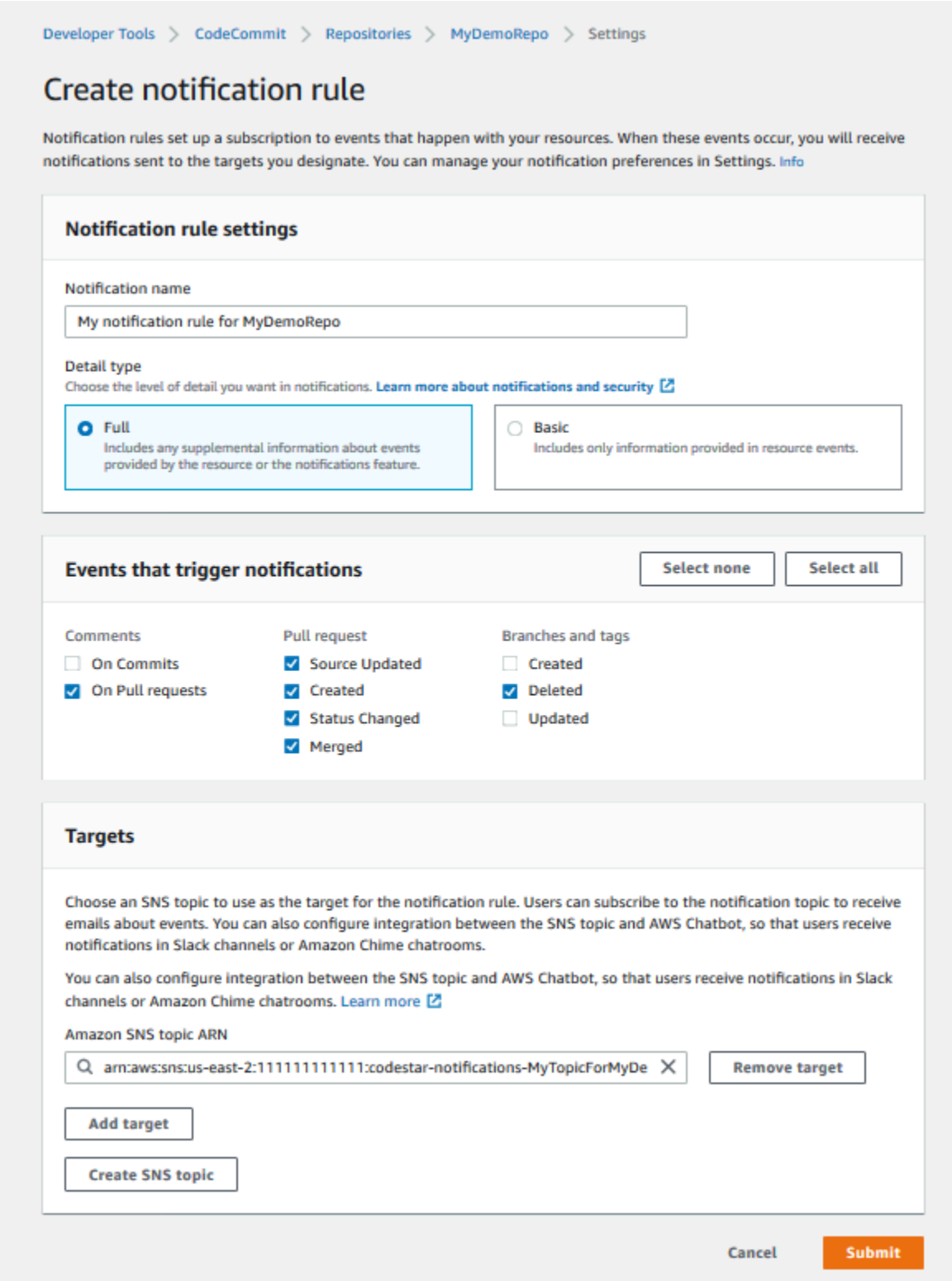
```
I've created a CodeCommit repository for us to use while working on our project.
The name of the repository is MyDemoRepo, and
```

```
it is in the  US East (Ohio) (us-east-2) region.
Here's what you need to do in order to get started using it:

1. Make sure that your version of Git on your local computer is 1.7.9 or later.
2. Generate Git credentials for your IAM user by signing into the IAM console
 here: https://console.aws.amazon.com/iam/  .
Switch to the Security credentials tab for your IAM user and choose the Generate button
 in HTTPS Git credentials for CodeCommit.
Make sure to save your credentials in a secure location!
3. Switch to a directory of your choice and clone the CodeCommit repository to your
 local machine by running the following command:
    git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-
demo-repo
4. When prompted for user name and password, use the Git credentials you just saved.


That's it!  If you'd like to learn more about using CodeCommit, you can start with the
 tutorial here.
```

您可以在设置 中找到完整的设置说明。

在 AWS CodeCommit 存储库中配置事件通知

您可以为存储库设置通知规则，以便存储库用户可以收到有关您指定的存储库事件类型的电子邮件。当事件与通知规则设置匹配时，将发送通知。您可以创建一个 Amazon SNS 主题以供通知使用，也可以使用您的 Amazon Web Services 账户中的现有主题。您可以使用 CodeCommit 控制台和 AWS CLI 来配置通知规则。



主题

- 使用存储库通知规则
- 创建通知规则

- 更改或禁用通知
- 删除通知

使用存储库通知规则

配置通知规则可在有人执行影响其他用户的操作时发送电子邮件，从而帮助您的存储库用户了解情况。例如，您可以将通知规则配置为在有人对提交发表评论时发送通知。在此配置中，当一个存储库用户对某个提交中的一行代码发表评论时，其他存储库用户就会收到电子邮件。他们可以登录并查看评论。对评论的响应也会生成电子邮件，以便存储库用户随时了解。

通知规则与仓库触发器不同，也不同于您可以在 2019 年 11 月 5 日之前在 CodeCommit 控制台中配置的通知。

- 虽然您可以配置触发器来使用 Amazon SNS 发送有关某些存储库事件的电子邮件，但这些事件仅限于操作事件，例如创建分支和将代码推送到分支。触发器不使用 CloudWatch 事件规则来评估存储库事件。其范围更加局限。有关如何使用触发器的更多信息，请参阅管理存储库触发器。
- 2019 年 11 月 5 日之前配置的通知的可用事件类型更少，并且无法配置为与 Amazon Chime 聊天室或 Slack 通道集成。您可以继续使用于 2019 年 11 月 5 日之前配置的通知，但不能创建此类型的通知。相反，可以创建和使用通知规则。我们建议使用通知规则并禁用或删除于 2019 年 11 月 5 日之前创建的通知。有关更多信息，请参阅 创建通知规则 和 删除通知。

创建通知规则

您可以使用通知规则来通知用户重要更改，例如在存储库中创建推送请求时。通知规则指定用于发送通知的事件和 Amazon SNS 主题。有关更多信息，请参阅什么是通知？

> **Note**
>
> 此功能在欧洲地区（米兰）区域不可用。要了解如何以该区域提供的体验配置通知，请参阅配置存储库通知。

您可以使用控制台或 AWS CLI 为创建通知规则 AWS CodeCommit。

创建通知规则（控制台）

1. 登录 AWS Management Console 并打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codecommit/。
2. 选择 Repositories (存储库)，然后选择要在其中添加通知规则的存储库。
3. 在存储库页面上，选择通知，然后选择 Create notification rule (创建通知规则)。您也可以转到存储库的设置页面，然后选择 Create notification rule (创建通知规则)。
4. 在 Notification name（通知名称）中，输入规则的名称。
5. 如果您只想在通知中 EventBridge 包含提供给 Amazon 的信息，请在 “详情类型” 中选择 “基本”。如果您想包括提供给 Amazon 的信息 EventBridge 以及可能由 CodeCommit 或通知管理器提供的信息，请选择 “全部”。

   有关更多信息，请参阅了解通知内容和安全性。
6. 在 Events that trigger notifications（触发通知的事件）中，选择要为其发送通知的事件。有关更多信息，请参阅存储库上的通知规则的事件。
7. 在目标中，执行下列操作之一：

   - 如果您已将资源配置为与通知一起使用，请在选择目标类型中，选择 AWS Chatbot (Slack) 或 SNS 主题。在选择目标中，选择客户端名称（对于中配置的 Slack 客户端 AWS Chatbot）或亚马逊 SNS 主题的亚马逊资源名称 (ARN)（对于已经配置了通知所需策略的 Amazon SNS 主题）。
   - 如果您尚未将资源配置为与通知一起使用，请选择创建目标，然后选择 SNS 主题。在 codestar-notifications- 之后提供主题的名称，然后选择创建。

> **Note**
>
> - 如果您在创建通知规则的过程中创建 Amazon SNS 主题，则为您应用允许通知功能将事件发布到主题的策略。使用为通知规则创建的主题有助于确保您仅订阅要接收有关此资源的通知的那些用户。
> - 您不能在创建通知规则的过程中创建 AWS Chatbot 客户端。如果您选择 AWS Chatbot (Slack)，则会看到一个按钮，指示您在中 AWS Chatbot配置客户端。选择该选项将打开 AWS Chatbot 控制台。有关更多信息，请参阅配置通知和 AWS Chatbot之间的集成。

- 如果要使用现有 Amazon SNS 主题作为目标，则在该主题可能存在的任何其他策略之外，您还必须为 AWS CodeStar 通知添加所需的策略。有关更多信息，请参阅为通知配置 Amazon SNS 主题以及了解通知内容和安全性。

8. 要完成规则创建，请选择提交。
9. 您必须为用户订阅规则的 Amazon SNS 主题，然后他们才能接收通知。有关更多信息，请参阅为用户订阅作为目标的 Amazon SNS 主题。您还可以设置通知之间的集成以及 AWS Chatbot 向 Amazon Chime 聊天室发送通知。有关更多信息，请参阅配置通知和之间的集成 AWS Chatbot。

创建通知规则（AWS CLI）

1. 在终端或命令提示符处，运行 create-notification rule 命令以生成 JSON 骨架：

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

您可以将此文件命名为所需的任意名称。在本示例中，文件命名为 *rule.json*。

2. 在纯文本编辑器中打开 JSON 文件，然后对其进行编辑，以包括该规则所需的资源、事件类型和目标。以下示例显示了名为 1234567890 *12* 的 AWS 账户*MyDemoRepo*中名**MyNotificationRule**为存储库的通知规则。具有完整详细信息的通知将发送到创建分支和标签*MyNotificationTopic*时命名的 Amazon SNS 主题：

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codecommit-repository-branches-and-tags-created"
    ],
    "Resource": "arn:aws:codecommit:us-east-1:123456789012:MyDemoRepo",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-
east-1:123456789012:MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

保存该文件。

3. 通过使用您刚编辑的文件，在终端或命令行上，再次运行 create-notification-rule 命令以创建通知规则：

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 如果成功，该命令将返回通知规则的 ARN，类似于以下内容：

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

更改或禁用通知

您可以使用 AWS CodeCommit 控制台更改 2019 年 11 月 5 日之前创建的通知的配置方式，包括向用户发送电子邮件的事件类型和用于发送有关存储库的电子邮件的 Amazon SNS 主题。您还可以使用 CodeCommit控制台管理订阅该主题的电子邮件地址和终端节点列表或禁用通知。

更改通知设置

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要在其中配置于 2019 年 11 月 5 日之前创建的通知的存储库名称。
3. 在导航窗格中，选择设置，然后选择通知。如果您看到一个横幅，告知您有通知而不是通知规则，请选择 Manage existing notifications (管理现有通知)。
4. 选择编辑。
5. 进行更改，然后选择 Save。

禁用通知是临时阻止用户接收有关存储库事件的电子邮件的简便方法。

要永久删除于 2019 年 11 月 5 日之前创建的通知，请按照 删除通知 中的步骤进行操作。

禁用通知

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中禁用通知的存储库的名称。
3. 在导航窗格中，选择设置，然后选择通知。选择 Manage existing notifications (管理现有通知)。
4. 选择编辑，然后在 事件状态 中，使用滑块以关闭 Enable notifications (启用通知)。选择保存。
5. 事件状态将更改为已禁用。将不会发送关于事件的电子邮件。禁用通知后，存储库 CloudWatch 的事件规则将自动禁用。请勿在 CloudWatch 事件控制台中手动更改其状态。

删除通知

如果您不想再使用在 2019 年 11 月 5 日之前为存储库创建的通知，则可以删除与该通知关联的 Amazon CloudWatch Events 规则。这将自动删除通知。这不会删除任何订阅或用于通知的 Amazon SNS 主题。

> **Note**
>
> 如果您从控制台更改了存储库的名称，2019 年 11 月 5 日之前创建的通知将继续工作，无需修改。但是，如果您从命令行或使用 API 更改了存储库的名称，通知将不再有效。还原通知的最简单的方法是删除通知设置并再次配置。

删除通知设置

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中删除于 2019 年 11 月 5 日之前创建的通知的存储库名称。
3. 在导航窗格中，选择设置，然后选择通知。如果您看到一个横幅，告知您有通知而不是通知规则，请选择 Manage existing notifications (管理现有通知)。
4. 在CloudWatch 事件规则中，复制为通知创建的规则的名称。
5. 登录 AWS Management Console 并打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
6. 在事件中，选择规则。在名称中，粘贴为通知创建的规则的名称。选择规则，然后在操作中，选择删除。

7. （可选）要在删除通知设置后更改或删除用于通知的亚马逊 SNS 主题，请转到 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的清除。

在中标记存储库 AWS CodeCommit

标签是您或 AWS 分配给 AWS 资源的自定义属性标签。 AWS 标签不同于 Git 标签，后者可以应用于提交。每个 AWS 标签分为两部分：

- 标签键 （例如，CostCenter、Environment、Project 或 Secret）。标签键区分大小写。
- 一个称为标签值的可选字段（例如，111122223333、Production 或团队名称）。省略标签值与使用空字符串效果相同。与标签键一样，标签值区分大小写。

这些被统称为键-值对。有关存储库可拥有的标签数量限制以及标签键和值的限制，请参阅限制。

标签可帮助您识别和整理 AWS 资源。许多 AWS 服务都支持标记，因此您可以为来自不同服务的资源分配相同的标签，以表明这些资源是相关的。例如，您可以为 CodeCommit存储库分配与分配给 Amazon S3 存储桶相同的标签。有关标记策略的更多信息，请参阅为资源添加标签。 AWS

在中 CodeCommit，主要资源是存储库。您可以使用 CodeCommit 控制台、 AWS CLI、 CodeCommit API 或 AWS 软件开发工具包为存储库添加、管理和移除标签。除了使用标签标识、整理和跟踪存储库以外，您还可以在 IAM 策略中使用标签以帮助控制哪些用户可以查看并与存储库交互。有关基于标签的访问策略示例，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

主题

- 为存储库添加标签
- 查看存储库的标签
- 编辑存储库的标签
- 从存储库中移除标签

为存储库添加标签

向存储库添加标签可以帮助您识别和组织 AWS 资源并管理对资源的访问权限。首先，为存储库添加一个或多个标签（键值对）。请记住，存储库可以拥有的标签数量有限制。键和值字段中可以使用的字符有限制。有关更多信息，请参阅限制。有了标签后，您可以创建 IAM 策略以根据这些标签管理对存储库的访问。您可以使用 CodeCommit 控制台或 AWS CLI 向存储库添加标签。

> **Important**
>
> 为存储库添加标签会影响对该存储库的访问。为存储库添加标签之前，请务必查看是否存在任何 IAM 策略可能使用标签来控制对资源（如存储库）的访问。有关基于标签的访问策略示例，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

有关在创建存储库时为其添加标签的更多信息，请参阅创建存储库（控制台）。

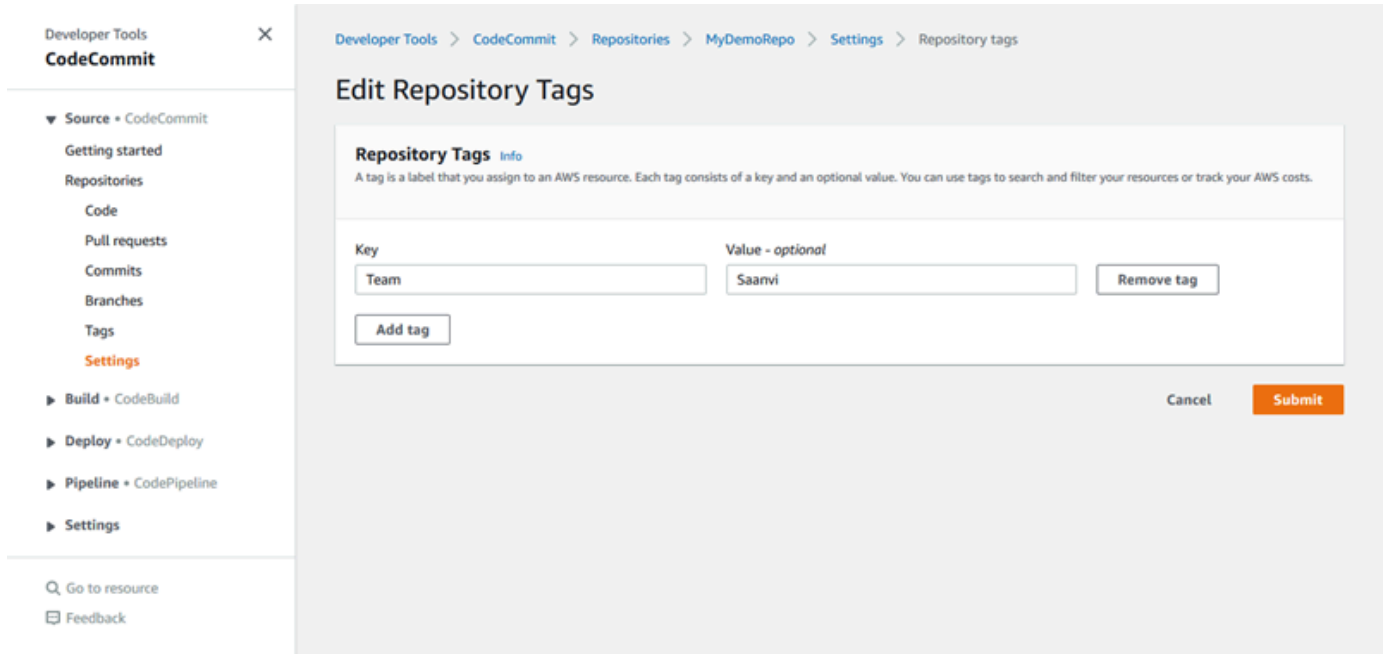
主题

- 为存储库添加标签（控制台）
- 为存储库添加标签 (AWS CLI)

为存储库添加标签（控制台）

您可以使用 CodeCommit 控制台向 CodeCommit 存储库添加一个或多个标签。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中添加标签的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。选择 Repository tags (存储库标签)。
4. 如果此存储库中尚未添加标签，请选择 Add tag (添加标签)。反之，请选择编辑，然后选择添加标签。
5. 在键中，输入标签的名称。您可以在值中添加可选的标签值。



6. （可选）要添加其他标签，请再次选择添加标签。
7. 添加完标签后，选择提交。

为存储库添加标签 (AWS CLI)

按照以下步骤使用 AWS CLI 向 CodeCommit 存储库添加标签。要在创建存储库为其添加标签，请参阅创建存储库 (AWS CLI)。

在这些步骤中，我们假设您已安装最新版本的 AWS CLI 或已更新到当前版本。有关更多信息，请参阅安装 AWS Command Line Interface。

在终端或命令行中运行 tag-resource 命令，并指定要添加标签的存储库的 Amazon 资源名称 (ARN) 以及要添加的标签的键和值。您可以将多个标签添加到一个存储库中。例如，要*MyDemoRepo*使用两个标签标记名为存储库，一个名为 *Stat* us 的标签键的标签值为 S *ecret*，一个名为 Tea *m* 的标签键的标签值为 *Saan* vi：

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Status=Secret,Team=Saanvi
```

如果成功，该命令不返回任何内容。

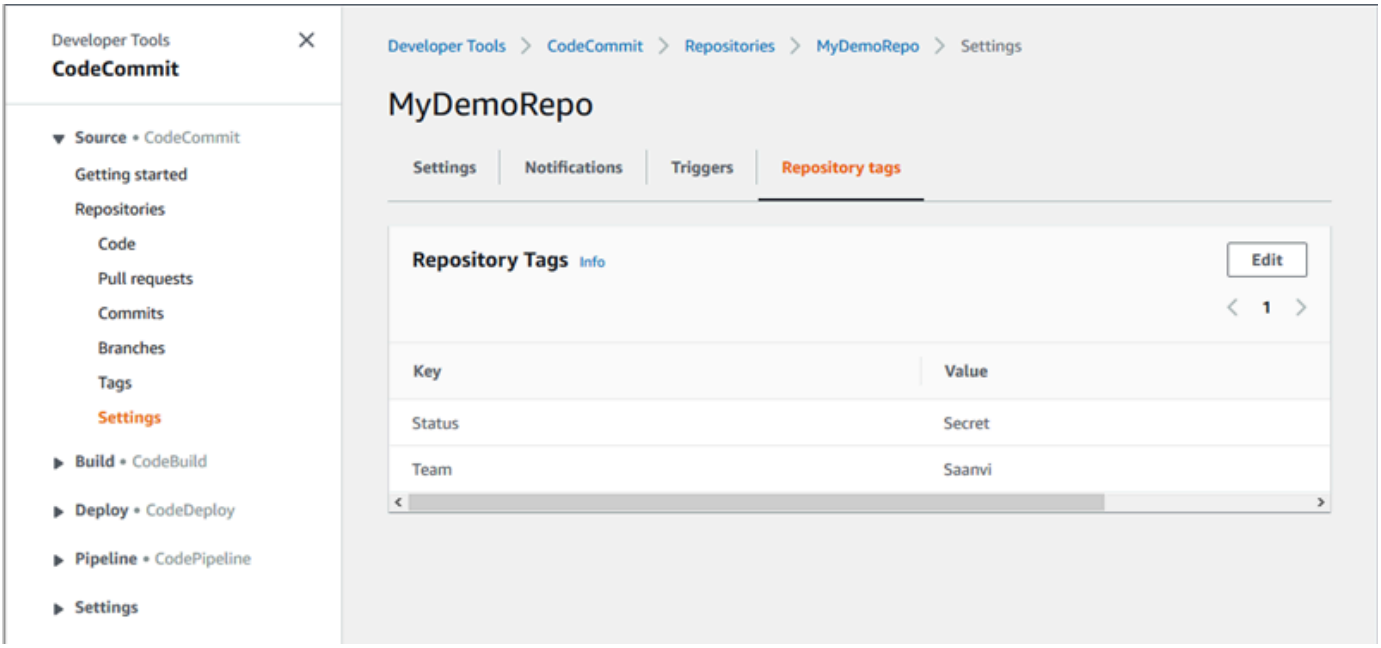
查看存储库的标签

标签可以帮助您识别和整理 AWS 资源并管理对资源的访问权限。有关标记策略的更多信息，请参阅为资源添加标签。 AWS有关基于标签的访问策略示例，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

查看存储库的标签（控制台）

您可以使用 CodeCommit 控制台查看与 CodeCommit 仓库关联的标签。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中查看标签的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。选择 Repository tags (存储库标签)。



查看存储库的标签 (AWS CLI)

按照以下步骤使用 AWS CLI 来查看 CodeCommit 存储库的 AWS 标签。如果尚未添加标签，则返回的列表为空。

在终端或命令行中，运行 list-tags-for-resource 命令。例如，要查看*MyDemoRepo*以 ARN 命名的存储库的标签键和标签值列表 arn: a *ws: codecommit: us-east-2:11111111* 111111:: MyDemoRepo

```
aws codecommit list-tags-for-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo
```

如果成功，该命令返回类似以下内容的信息：

```
{
    "tags": {
        "Status": "Secret",
        "Team": "Saanvi"
    }
}
```

编辑存储库的标签

您可以更改与存储库关联的标签值。您也可以更改标签键的名称，这相当于删除当前的标签并使用新名称和相同的值添加一个不同的标签。请记住，键和值字段中可以使用的字符有限制。有关更多信息，请参阅限制。

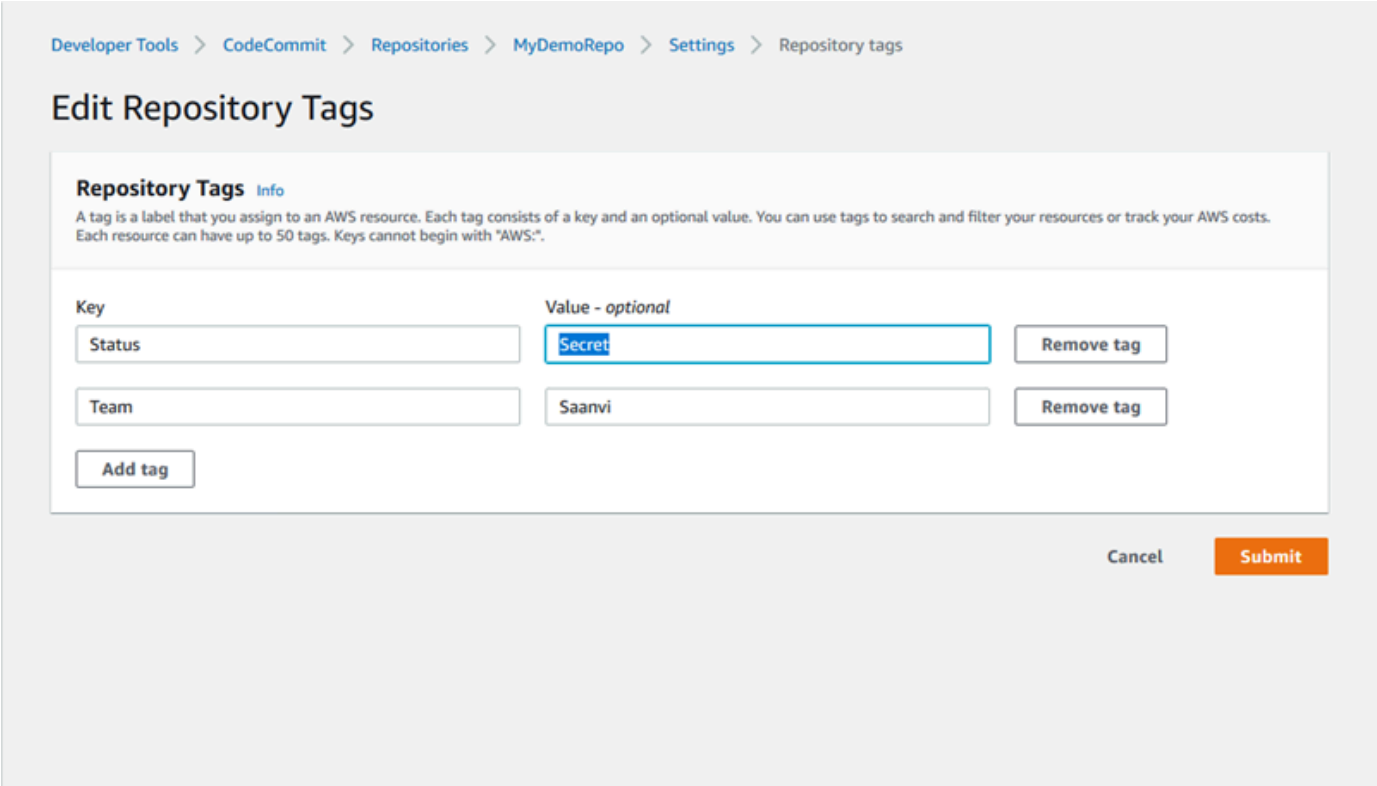
> **Important**
>
> 编辑存储库的标签会影响对该存储库的访问。编辑存储库的标签名称（键）或值之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如存储库）的访问。有关基于标签的访问策略示例，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

编辑存储库的标签（控制台）

您可以使用 CodeCommit 控制台编辑与 CodeCommit 仓库关联的标签。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要为其编辑标签的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。选择 Repository tags (存储库标签)。
4. 选择编辑。

5.



请执行以下操作之一：

- 要更改标签，则在键中输入新名称。更改标签的名称相当于删除标签并使用新的键名添加新标签。
- 要更改标签的值，则输入新值。如果您想将标签值清空，请删除当前的值并将字段保留为空白。

6. 编辑完标签后，选择提交。

编辑存储库的标签 (AWS CLI)

按照以下步骤使用 AWS CLI 来更新 CodeCommit 存储库的标签。您可以更改现有键的值或添加另一个键。

在终端或命令行中运行 tag-resource 命令，并指定要更新标签的存储库的 Amazon 资源名称 (ARN) 以及标签键和标签值：

```
aws codecommit tag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tags Team=Li
```

从存储库中移除标签

您可以移除与存储库关联的一个或多个标签。移除标签不会从与该标签关联的其他 AWS 资源中删除该标签。

> **Important**
>
> 删除存储库的标签会影响对该存储库的访问。从存储库中移除标签之前，请务必查看是否存在任何 IAM 策略可能使用标签的键或值来控制对资源（如存储库）的访问。有关基于标签的访问策略示例，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

从存储库中移除标签（控制台）

您可以使用 CodeCommit 控制台删除标签和 CodeCommit 存储库之间的关联。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要移除其标签的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。选择 Repository tags (存储库标签)。
4. 选择编辑。
5. 找到要移除的标签，然后选择移除标签。
6. 移除标签之后，选择提交。

从存储库中移除标签 (AWS CLI)

按照以下步骤使用从 AWS CLI CodeCommit 存储库中移除标签。移除标签不会将其删除，而只是删除标签和存储库之间的关联。

> **Note**
>
> 如果删除 CodeCommit 存储库，则会从已删除的存储库中移除所有标签关联。您无需在删除存储库之前移除标签。

在终端或命令行中运行 untag-resource 命令，并指定要移除标签的存储库的 Amazon 资源名称 (ARN) 以及要移除的标签的标签键。例如，要删除名为 *Stat* us 标签的仓库上的标签 *MyDemoRepo*，请执行以下操作：

```
aws codecommit untag-resource --resource-arn arn:aws:codecommit:us-
west-2:111111111111:MyDemoRepo --tag-keys Status
```

如果成功，该命令不返回任何内容。要验证与存储库关联的标签，请运行 list-tags-for-resource 命令。

管理 AWS CodeCommit 仓库的触发器

您可以配置 CodeCommit 存储库，以便代码推送或其他事件触发操作，例如从亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 发送通知或在中调用函数。 AWS Lambda您最多可以为每个 CodeCommit 存储库创建 10 个触发器。

触发器通常配置为：

- 有人向存储库推送内容时，向订阅用户发送电子邮件。
- 有人向存储库的主分支推送内容后，通知外部构建系统启动构建。

对于通知外部构建系统等场景，需要编写 Lambda 函数来与其他应用程序交互。对于电子邮件场景，只需创建一个 Amazon SNS 主题即可。

本主题向您展示如何设置允许在 Amazon SNS CodeCommit 和 Lambda 中触发操作的权限。此外，它还包括创建、编辑、测试和删除触发器的示例的链接。

主题

创建资源并添加权限 CodeCommit

您可以将 Amazon SNS 主题和 Lambda 函数与中的触发器集成 CodeCommit，但必须先创建资源，然后使用授予与这些资源交互 CodeCommit 的权限的策略配置资源。您必须在与 CodeCommit 存储库 AWS 区域 相同的位置创建资源。例如，如果存储库位于美国东部（俄亥俄州）(us-east-2)，则 Amazon SNS 主题或 Lambda 函数必须位于美国东部（俄亥俄州）。

- 对于 Amazon SNS 主题，如果使用与存储库相同的账户创建 Amazon SNS 主题，则无需配置其他 IAM 策略或权限。 CodeCommit 您可以在创建并订阅 Amazon SNS 主题后立即创建 CodeCommit 触发器。
  - 有关在 Amazon SNS 中创建主题的更多信息，请参阅 Amazon SNS 文档。
  - 有关使用 Amazon SNS 向 Amazon SQS 队列发送消息的信息，请参阅《Amazon SNS 开发人员指南》中的向 Amazon SQS 队列发送消息。
  - 有关使用 Amazon SNS 调用 Lambda 函数的信息，请参阅《Amazon SNS 开发人员指南》中的调用 Lambda 函数。
- 如果您想将触发器配置为在另一个 AWS 账户中使用某个 Amazon SNS 主题，则必须先使用允许 CodeCommit 向该主题发布内容的策略配置该主题。有关更多信息，请参阅 示例 1：创建允许对 Amazon SNS 主题进行跨账户存取的策略。
- 您可以通过在 Lambda 控制台中创建作为 Lambda 函数一部分的触发器来对该函数进行配置。这是最简单的方法，因为在 Lambda 控制台中创建的触发器会自动包含调用 Lambda 函数所需的 CodeCommit 权限。如果您在中创建触发器 CodeCommit，则必须包含 CodeCommit 允许调用该函数的策略。有关更多信息，请参阅 为现有的 Lambda 函数创建触发器 和 示例 3：为 AWS Lambda 与 CodeCommit 触发器的集成创建策略。

示例：为 Amazon SNS 主题创建 AWS CodeCommit 触发器

您可以为存储库创建触发器，以便该 CodeCommit 存储库中的事件触发来自亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题的通知。您可能需要为 Amazon SNS 主题创建触发器，使用户能够订阅有关存储库事件（如删除分支）的通知。您还可以利用亚马逊 SNS 主题与其他服务的集成，例如亚马逊简单队列服务 (Amazon SQS) Simple Queue Service 和。 AWS Lambda

> **Note**
>
> - 您必须将触发器指向现有的 Amazon SNS 主题，后者将作为响应存储库事件所执行的操作。有关创建和订阅 Amazon SNS 主题的更多信息，请参阅开始使用 Amazon Simple Notification Service。

- 触发器不支持 Amazon SNS FIFO（先入先出）主题。 CodeCommit

主题

为 CodeCommit存储库的 Amazon SNS 主题创建触发器（控制台）

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从存储库 中，选择要创建存储库事件触发器的存储库。
3. 在存储库的导航窗格中，选择设置，然后选择触发器。
4. 选择创建触发器，然后执行以下操作：

   - 在触发器名称中，输入触发器的名称（例如 *MyFirstTrigger*）。
   - 在事件中，选择将触发 Amazon SNS 主题以发送通知的存储库事件。

     如果选择 All repository events，则无法选择任何其他事件。要选择事件的子集，请删除 All repository events，然后从列表中选择一个或多个事件。例如，如果您希望触发器仅在用户在 CodeCommit存储库中创建分支或标签时运行，请移除所有存储库事件，然后选择创建分支或标记。
   - 如果希望触发器应用于存储库的所有分支，请在分支中，将选定内容保留为空，因为此默认选项会自动将触发器应用于所有分支。如果您希望此触发器仅应用于特定分支，请从存储库分支列表中选择最多 10 个分支名称。
   - 在选择要使用的服务中，选择 Amazon SNS。
   - 在 Amazon SNS 中，从列表中选择主题名称或输入主题的 ARN。

   > **Note**
   >
   > 触发器不支持 Amazon SNS FIFO（先入先出）主题。 CodeCommit 您必须选择类型设置为“标准”的 Amazon SNS 主题。如果您想使用 Amazon SNS FIFO 主题，则必须为将 SNS FIFO 主题配置为目标 CodeCommit 的事件配置 Amazon Eventbridge 规则。

- 在自定义数据中，提供您希望包含在 Amazon SNS 主题发送的通知中的任何信息（例如，开发人员在讨论该存储库中的开发工作时使用的 IRC 通道名称）。该字段是一个字符串。它不能用于传递任何动态参数。

5. （可选）选择测试触发器。此步骤可帮助您确认是否正确配置了与 Amazon SNS 主题 CodeCommit 之间的访问权限。它通过 Amazon SNS 主题使用您存储库中的数据（如果可用）发送测试通知。如果没有真实数据可用，测试通知将包含示例数据。
6. 选择创建触发器以完成触发器的创建操作。

为 CodeCommit 存储库的 Amazon SNS 主题创建触发器 ()AWS CLI

您还可以使用命令行为 Amazon SNS 主题创建触发器，以响应 CodeCommit 存储库事件，例如有人向您的存储库推送提交时。

为 Amazon SNS 主题创建触发器

1. 打开纯文本编辑器，创建一个 JSON 文件，在其中指定：

   - Amazon SNS 主题名称。

     > **Note**
     >
     > 触发器不支持 Amazon SNS FIFO（先入先出）主题。 CodeCommit 您必须选择类型设置为“标准”的 Amazon SNS 主题。如果您想使用 Amazon SNS FIFO 主题，则必须为将 SNS FIFO 主题配置为目标 CodeCommit 的事件配置 Amazon Eventbridge 规则。

   - 要用该触发器监控的存储库和分支。（如果没有指定任何分支，触发器将应用到存储库中的所有分支。）
   - 激活该触发器的事件。

   保存该文件。

   ####### main # preprod ######### MyDemoRepo############## mysnStopic # Amazon S N S ###

   ```
   {
       "repositoryName": "MyDemoRepo",
   ```

```
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

存储库中的每个触发器在该 JSON 文件中都有一个对应的触发块。要为存储库创建多个触发器，请在 JSON 文件中包含多个触发块。请记住，在该文件中创建的所有触发器都用于指定的存储库。不能在一个 JSON 文件中为多个存储库创建触发器。例如，如果要为某个存储库创建两个触发器，您可以创建一个包含两个触发块的 JSON 文件。在下面的示例中，没有为第二个触发器指定任何分支，因此，该触发器将应用到所有分支：

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyFirstTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        },
        {
            "name": "MySecondTrigger",
            "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic2",
            "customData": "",
            "branches": [],
            "events": [
```

```
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

您可以为指定的事件 (例如，向存储库推送提交时) 创建触发器。事件类型包括：

- `all`：指定存储库和分支中的所有事件。
- `updateReference`：向指定存储库和分支推送提交时。
- `createReference`：在指定存储库中创建新的分支或标签时。
- `deleteReference`：在指定存储库中删除分支或标签时。

> **Note**
>
> 您可以在触发器中使用多种事件类型。但如果指定 `all`，就无法再指定其他事件。

要查看有效事件类型的完整列表，请在终端或命令提示符处输入 aws codecommit put-repository-triggers help。

此外，您还可以在 `customData` 中包含一个字符串 (例如，开发人员在讨论该存储库的开发时使用的 IRC 通道的名称)。该字段是一个字符串。它不能用于传递任何动态参数。此字符串作为属性附加到为响应触发器而返回的 CodeCommit JSON 中。

2. （可选）在终端或命令提示符处，运行 test-repository-triggers 命令。该测试使用存储库中的示例数据（或者，如果没有数据可用，则生成示例数据）向该 Amazon SNS 主题的订阅用户发送通知。例如，以下内容用于测试名为 t*rigger.json ####### JSON* 是否有效，以及该文件是否 CodeCommit 可以发布到 Amazon SNS 主题：

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

如果成功，该命令返回类似以下内容的信息：

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
```

```
    "failedExecutions": []
}
```

3. 在终端或命令提示符处，运行put-repository-triggers命令以在中创建触发器 CodeCommit。例如，要使用名为 *trigger.json* 的 JSON 文件创建触发器：

   ```
   aws codecommit put-repository-triggers --cli-input-json
   file://trigger.json
   ```

   该命令将返回与以下示例类似的配置 ID：

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE"
   }
   ```

4. 要查看触发器的配置，请运行 get-repository-triggers 命令，并指定存储库的名称：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   该命令返回为存储库配置的所有触发器的结构，类似于以下内容：

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE",
       "triggers": [
           {
               "events": [
                   "all"
               ],
               "destinationArn": "arn:aws:sns:us-east-2:111122223333:MySNSTopic",
               "branches": [
                   "main",
                   "preprod"
               ],
               "name": "MyFirstTrigger",
               "customData": "Project ID 12345"
           }
       ]
   }
   ```

5. 要测试触发器本身的功能，请生成并向配置该触发器的存储库推送一个提交。您应该会看到来自 Amazon SNS 主题的响应。例如，如果您将 Amazon SNS 主题配置为发送电子邮件，您应该会在订阅了该主题的电子邮件账户中看到来自 Amazon SNS 的电子邮件。

以下是 Amazon SNS 为响应仓库推送而发送的电子邮件的输出示例： CodeCommit

```
{
  "Records":[
     {
        "awsRegion":"us-east-2",
        "codecommit":{
           "references" : [
              {
                    "commit":"317f8570EXAMPLE",
                    "created":true,
                    "ref":"refs/heads/NewBranch"
              },
              {
                    "commit":"4c925148EXAMPLE",
                    "ref":"refs/heads/preprod",
              }
            ]
          },
        "eventId":"11111-EXAMPLE-ID",
        "eventName":"ReferenceChange",
        "eventPartNumber":1,
        "eventSource":"aws:codecommit",
        "eventSourceARN":"arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
        "eventTime":"2016-02-09T00:08:11.743+0000",
        "eventTotalParts":1,
        "eventTriggerConfigId":"0123456-I-AM-AN-EXAMPLE",
        "eventTriggerName":"MyFirstTrigger",
        "eventVersion":"1.0",
        "customData":"Project ID 12345",
        "userIdentityARN":"arn:aws:iam::111122223333:user/JaneDoe-CodeCommit",
     }
  ]
}
```

示例：为 AWS Lambda 函数创建 AWS CodeCommit 触发器

您可以为仓库创建触发器，以便 CodeCommit 存储库中的事件调用 Lambda 函数。在此示例中，您将创建一个 Lambda 函数，该函数将用于将存储库克隆到亚马逊 CloudWatch 日志的 URL 返回。

主题

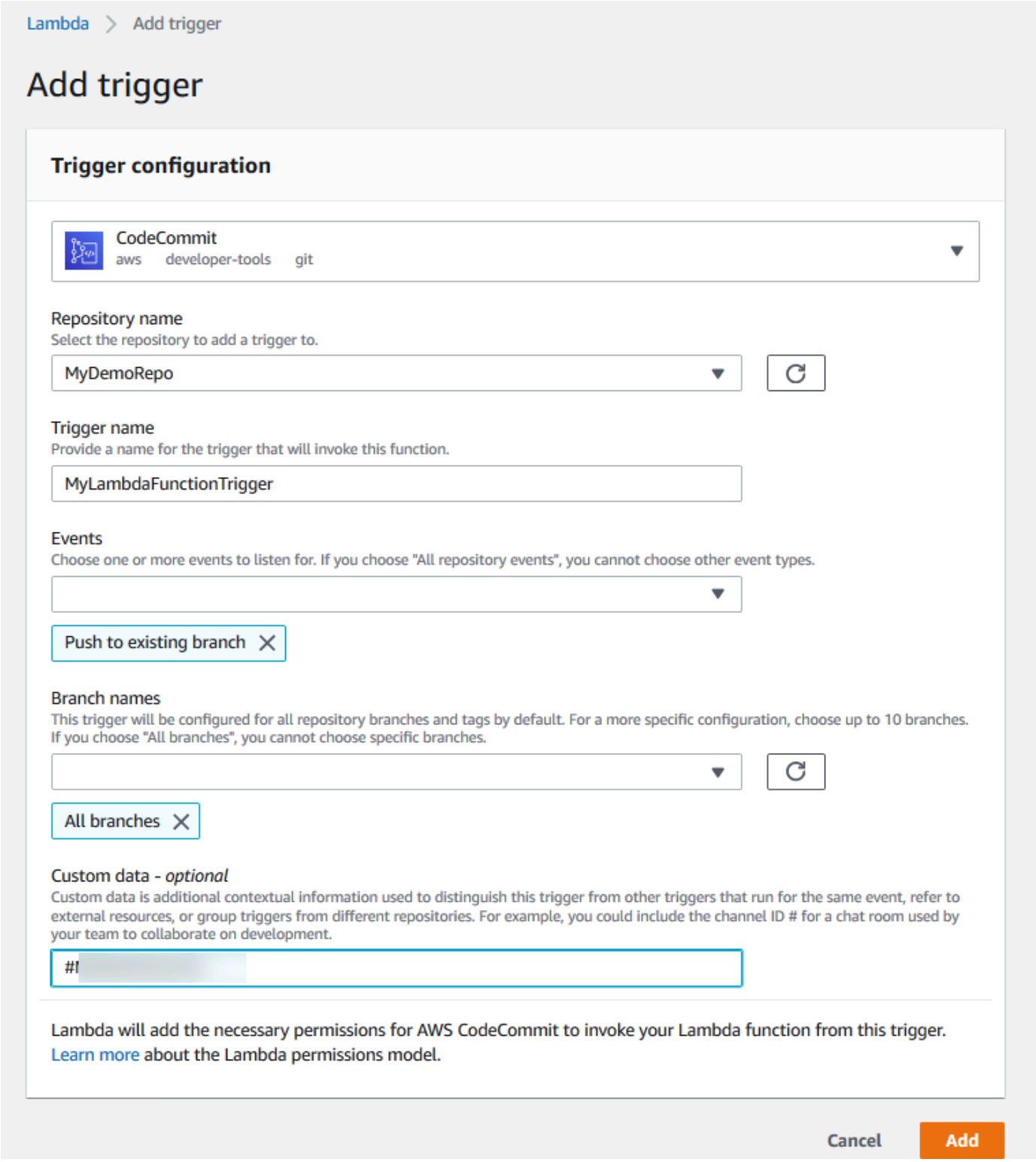
- 创建 Lambda 函数
- 在 AWS CodeCommit 存储库中查看 Lambda 函数的触发器

创建 Lambda 函数

当您使用 Lambda 控制台创建函数时，也可以为 Lambda 函数创建 CodeCommit触发器。以下步骤包含一个示例 Lambda 函数。该示例有两种语言版本： JavaScript 和 Python。该函数返回用于将存储库克隆到 CloudWatch 日志的 URL。

使用 Lambda 蓝图创建 Lambda 函数

1. 登录 AWS Management Console 并打开 AWS Lambda 控制台，网址为 https://console.aws.amazon.com/lambda/。
2. 在 Lambda 函数页面上，选择创建函数。（如果以前未使用过 Lambda，请选择立即开始使用。）
3. 在创建函数页面上，选择从头开始创作。例如，在函数名称中，提供函数的名称*MyLambdaFunctionforCodeCommit*。在 Runtime (运行时) 中，选择要用于编写函数的语言，然后选择 Create function (创建函数)。
4. 在 Configuration (配置) 选项卡中，选择 Add trigger (添加触发器)。
5. 在触发器配置中，CodeCommit从服务下拉列表中进行选择。



- 在存储库名称中，选择要配置触发器（该触发器使用 Lambda 函数响应存储库事件）的存储库的名称。

- 在触发器名称中，输入触发器的名称（例如 *MyLambdaFunctionTrigger*）。
- 在事件中，选择触发 Lambda 函数的存储库事件。如果选择 All repository events，则无法选择任何其他事件。如果需要选择事件的子集，请清除 All repository events，然后从列表中选择所需的事件。例如，如果您希望触发器仅在用户在 AWS CodeCommit 存储库中创建标签或分支时运行，请移除所有存储库事件，然后选择创建分支或标记。
- 如果希望将触发器应用于存储库的所有分支，请在 Branches 中选择 All branches。否则，请选择 Specific branches。默认情况下将添加存储库的默认分支。您可以保留或从列表中删除该分支。最多可从存储库分支列表中选择 10 个分支名称。
- （可选）在自定义数据中，输入要包含在 Lambda 函数中的信息（例如，开发人员用于讨论存储库中的开发工作的 IRC 通道的名称）。该字段是一个字符串。它不能用于传递任何动态参数。

选择添加。

6. 在 Configuration (配置) 页面的 Function Code (函数代码) 中的 “Code entry (代码条码)”类型中，选择“Edit code inline (内联编辑代码)”。在 Runtime (运行时) 中，选择 Node.js。如果需要创建示例 Python 函数，请选择 Python。
7. 在 Code entry type 中，选择 Edit code inline，然后使用以下两个示例之一替换 hello world 代码。

   对于 Node.js：

```
import {
  CodeCommitClient,
  GetRepositoryCommand,
} from "@aws-sdk/client-codecommit";

const codecommit = new CodeCommitClient({ region: "your-region" });

/**
 * @param {{ Records: { codecommit: { references: { ref: string }[] },
 eventSourceARN: string  }[]} event
 */
export const handler = async (event) => {
  // Log the updated references from the event
  const references = event.Records[0].codecommit.references.map(
    (reference) => reference.ref,
  );
  console.log("References:", references);
```

```
  // Get the repository from the event and show its git clone URL
  const repository = event.Records[0].eventSourceARN.split(":")[5];
  const params = {
    repositoryName: repository,
  };

  try {
    const data = await codecommit.send(new GetRepositoryCommand(params));
    console.log("Clone URL:", data.repositoryMetadata.cloneUrlHttp);
    return data.repositoryMetadata.cloneUrlHttp;
  } catch (error) {
    console.error("Error:", error);
    throw new Error(
      `Error getting repository metadata for repository ${repository}`,
    );
  }
};
```

对于 Python：

```
import json
import boto3

codecommit = boto3.client("codecommit")


def lambda_handler(event, context):
    # Log the updated references from the event
    references = {
        reference["ref"]
        for reference in event["Records"][0]["codecommit"]["references"]
    }
    print("References: " + str(references))

    # Get the repository from the event and show its git clone URL
    repository = event["Records"][0]["eventSourceARN"].split(":")[5]
    try:
        response = codecommit.get_repository(repositoryName=repository)
        print("Clone URL: " + response["repositoryMetadata"]["cloneUrlHttp"])
        return response["repositoryMetadata"]["cloneUrlHttp"]
    except Exception as e:
        print(e)
```

```
        print(
            "Error getting repository {}. Make sure it exists and that your
repository is in the same region as this function.".format(
                repository
            )
        )
        raise e
```

8. 在权限选项卡的执行角色中，选择角色以在 IAM 控制台中打开。编辑附加的策略以为要使用触发器的存储库添加 `GetRepository` 权限。

在 AWS CodeCommit 存储库中查看 Lambda 函数的触发器

创建 Lambda 函数后，可以在 AWS CodeCommit中查看和测试触发器。测试触发器将运行函数以响应您指定的存储库事件。

查看和测试 Lambda 函数的触发器

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从存储库中，选择要查看其触发器的存储库。
3. 在存储库的导航窗格中，选择设置，然后选择触发器。
4. 查看存储库的触发器列表。您应该会看到您在 Lambda 控制台中创建的触发器。从列表中选择该触发器，然后选择测试触发器。该选项将尝试使用有关您的存储库的示例数据（包括存储库的最新提交 ID）调用该函数。（如果不存在提交历史记录，则将生成由零组成的示例值。）这有助于您确认您已正确配置 AWS CodeCommit 和 Lambda 函数之间的访问权限。
5. 要进一步验证触发器的功能，请生成并向配置该触发器的存储库推送一个提交。在 Lambda 控制台中该函数的监控选项卡上，您应该会看到 Lambda 函数的响应。从 “监控” 选项卡中，选择 “查看登录信息” CloudWatch。 CloudWatch 控制台将在新选项卡中打开，并显示您的函数的事件。从与您推送提交的时间相对应的列表中选择日志流。您应会看到类似以下内容的事件数据：

```
START RequestId: 70afdc9a-EXAMPLE Version: $LATEST
2015-11-10T18:18:28.689Z 70afdc9a-EXAMPLE References: [ 'refs/heads/main' ]
2015-11-10T18:18:29.814Z 70afdc9a-EXAMPLE Clone URL: https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
END RequestId: 70afdc9a-EXAMPLE
```

```
REPORT RequestId: 70afdc9a-EXAMPLE Duration: 1126.87 ms Billed Duration: 1200 ms
 Memory Size: 128 MB Max Memory Used: 14 MB
```

示例：在中 AWS CodeCommit 为现有 AWS Lambda 函数创建触发器

创建调用 Lambda 函数的触发器的最简单的方法是在 Lambda 控制台中创建该触发器。这种内置集成可确保 CodeCommit 拥有运行该函数所需的权限。要为现有 Lambda 函数添加触发器，请转到 Lambda 控制台，然后选择该函数。在该函数的触发器选项卡上，按照添加触发器中的步骤操作。这些步骤类似于创建 Lambda 函数中的步骤。

您还可以在存储库中为 Lambda 函数创建触发器。 CodeCommit 为此，您需要选择要调用的现有 Lambda 函数。它还要求您手动配置运行该函数 CodeCommit 所需的权限。

主题

- 手动配置权限 CodeCommit 以允许运行 Lambda 函数
- 在 CodeCommit存储库（控制台）中为 Lambda 函数创建触发器
- 为 CodeCommit存储库创建 Lambda 函数的触发器 ()AWS CLI

手动配置权限 CodeCommit 以允许运行 Lambda 函数

如果您在中 CodeCommit 创建调用 Lambda 函数的触发器，则必须手动配置允许 CodeCommit 运行 Lambda 函数的权限。要避免进行此手动配置，可以考虑改为在 Lambda 控制台中为该函数创建触发器。

CodeCommit 允许运行 Lambda 函数

1. 打开纯文本编辑器并创建一个 JSON 文件，该文件指定 Lambda 函数名称、 CodeCommit 存储库的详细信息以及您希望在 Lambda 中允许的操作，如下所示：

```
{
    "FunctionName": "MyCodeCommitFunction",
    "StatementId": "1",
    "Action": "lambda:InvokeFunction",
    "Principal": "codecommit.amazonaws.com",
    "SourceArn": "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo",
    "SourceAccount": "111122223333"
}
```

2. 将文件另存为 JSON 文件，其名称便于您记住（例如 *AllowAccessfromMyDemoRepo*.json）。

3. 在终端（Linux、macOS 或 Unix）或命令行 (Windows) 中，使用刚刚创建的 JSON 文件运行 aws lambda add-permissions 命令，向与 Lambda 函数关联的资源策略添加权限：

```
aws lambda add-permission --cli-input-json file://AllowAccessfromMyDemoRepo.json
```

该命令返回刚添加的策略语句（JSON 格式），内容如下：

```
{
    "Statement": "{\"Condition\":{\"StringEquals\":{\"AWS:SourceAccount
\":\"111122223333\"},\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo\"}},\"Action
\":[\"lambda:InvokeFunction\"],\"Resource\":\"arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction\",\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"codecommit.amazonaws.com\"},\"Sid\":\"1\"}"
}
```

有关 Lambda 函数资源策略的更多信息，请参阅用户指南AddPermission中的拉取/推送事件模型。AWS Lambda

4. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
5. 在 Dashboard 导航窗格中，选择 Roles，然后在角色列表中选择 *lambda_basic_execution*。
6. 在角色的摘要页面上，选择权限选项卡，然后在内联策略中选择创建角色策略。
7. 在 Set Permissions 页面上，选择 Policy Generator，然后选择 Select。
8. 在编辑权限页面上，执行以下操作：

   - 在 Effect 中选择 Allow。
   - 在 AWS 服务中，选择 AWS CodeCommit。
   - 在 “操作” 中，选择GetRepository。
   - 在 Amazon 资源名称 (ARN) 中输入存储库的 ARN（例如，`arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo`）。

   选择 Add Statement，然后选择 Next Step。
9. 在 Review Policy 页面上，选择 Apply Policy。

策略语句应如下例所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt11111111",
            "Effect": "Allow",
            "Action": [
                "codecommit:GetRepository"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo"
            ]
        }
    ]
}
```

在 CodeCommit存储库（控制台）中为 Lambda 函数创建触发器

创建 Lambda 函数后，您可以在中创建一个触发器 CodeCommit ，该触发器运行该函数以响应您指定的存储库事件。

> **Note**
>
> 在成功测试或运行示例的触发器之前，必须配置允许 CodeCommit 调用函数和 Lambda 函数以获取存储库相关信息的策略。有关更多信息，请参阅 CodeCommit 允许运行 Lambda 函数。

为 Lambda 函数创建触发器

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从存储库 中，选择要创建存储库事件触发器的存储库。
3. 在存储库的导航窗格中，选择设置，然后选择触发器。
4. 选择创建触发器。
5. 在创建触发器中，执行以下操作：

- 在触发器名称中，输入触发器的名称（例如 *MyLambdaFunctionTrigger*）。
- 在事件中，选择触发 Lambda 函数的存储库事件。

  如果选择 All repository events，则无法选择任何其他事件。如果需要选择事件的子集，请清除 All repository events，然后从列表中选择所需的事件。例如，如果您希望触发器仅在用户在 CodeCommit 存储库中创建标签或分支时运行，请移除所有存储库事件，然后选择创建分支或标记。
- 如果您希望触发器应用于存储库中的所有分支，请在分支中，将选定内容保留为空，因为此默认选项会自动将触发器应用于所有分支。如果您希望此触发器仅应用于特定分支，请从存储库分支列表中选择最多 10 个分支名称。
- 在选择要使用的服务中，选择 AWS Lambda。
- 在 Lambda 函数中，从列表中选择函数名称，或输入函数的 ARN。
- （可选）在自定义数据中，输入要包含在 Lambda 函数中的信息（例如，开发人员用于讨论存储库中的开发工作的 IRC 通道的名称）。该字段是一个字符串。它不能用于传递任何动态参数。

6. （可选）选择测试触发器。该选项将尝试使用有关您的存储库的示例数据（包括存储库的最新提交 ID）调用该函数。（如果不存在提交历史记录，则将生成由零组成的示例值。）这有助于您确认您已正确配置 CodeCommit 和 Lambda 函数之间的访问权限。
7. 选择创建触发器以完成触发器的创建操作。
8. 要验证触发器的功能，请生成并向配置该触发器的存储库推送一个提交。在 Lambda 控制台中该函数的监控选项卡上，您应该会看到 Lambda 函数的响应。

为 CodeCommit存储库创建 Lambda 函数的触发器 ()AWS CLI

您还可以使用命令行为 Lambda 函数创建触发器，以响应 CodeCommit 存储库事件，例如有人向您的仓库推送提交时。

为 Lambda 函数创建触发器

1. 打开纯文本编辑器，创建一个 JSON 文件，在其中指定：

   - Lambda 函数名称。
   - 要用该触发器监控的存储库和分支。（如果没有指定任何分支，触发器将应用到存储库中的所有分支。）
   - 激活该触发器的事件。

保存该文件。

########## main # preprod ######### MyDemoRepo###############
#MyCodeCommitFunction######main # preprod#### Lambda ###

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

存储库中的每个触发器在该 JSON 文件中都有一个对应的触发块。要为存储库创建多个触发器，请在 JSON 中包含更多的触发块。请记住，在该文件中创建的所有触发器都用于指定的存储库。不能在一个 JSON 文件中为多个存储库创建触发器。例如，如果要为某个存储库创建两个触发器，您可以创建一个包含两个触发块的 JSON 文件。在下面的示例中，第二个触发块中未指定任何分支，因此，该触发器将应用到所有分支：

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
```

```
            ],
            "events": [
                "all"
            ]
        },
        {
            "name": "MyOtherLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyOtherCodeCommitFunction",
            "customData": "",
            "branches": [],
            "events": [
                "updateReference", "deleteReference"
            ]
        }
    ]
}
```

您可以为指定的事件 (例如，向存储库推送提交时) 创建触发器。事件类型包括：

- `all`：指定存储库和分支中的所有事件。
- `updateReference`：向指定存储库和分支推送提交时。
- `createReference`：在指定存储库中创建新的分支或标签时。
- `deleteReference`：在指定存储库中删除分支或标签时。

> **Note**
>
> 您可以在触发器中使用多种事件类型。但如果指定 `all`，就无法再指定其他事件。

要查看有效事件类型的完整列表，请在终端或命令提示符处输入 aws codecommit put-repository-triggers help。

此外，您还可以在 `customData` 中包含一个字符串 (例如，开发人员在讨论该存储库的开发时使用的 IRC 通道的名称)。该字段是一个字符串。它不能用于传递任何动态参数。此字符串作为属性附加到为响应触发器而返回的 CodeCommit JSON 中。

2. （可选）在终端或命令提示符处，运行 test-repository-triggers 命令。例如，以下内容用于测试名为 t*rigger.json # JSON ############ CodeCommit ####* Lambda 函数。如果没有实际数据可用，该测试使用示例数据来触发函数。

```
aws codecommit test-repository-triggers --cli-input-json file://trigger.json
```

如果成功，该命令返回类似以下内容的信息：

```
{
    "successfulExecutions": [
        "MyLambdaFunctionTrigger"
    ],
    "failedExecutions": []
}
```

3. 在终端或命令提示符处，运行put-repository-triggers命令以在中创建触发器 CodeCommit。例如，要使用名为 *trigger.json* 的 JSON 文件创建触发器：

   ```
   aws codecommit put-repository-triggers --cli-input-json
   file://trigger.json
   ```

   该命令将返回与以下示例类似的配置 ID：

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE"
   }
   ```

4. 要查看触发器的配置，请运行 get-repository-triggers 命令，并指定存储库的名称：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   该命令返回为存储库配置的所有触发器的结构，类似于以下内容：

   ```
   {
       "configurationId": "0123456-I-AM-AN-EXAMPLE",
       "triggers": [
           {
               "events": [
                   "all"
               ],
               "destinationArn": "arn:aws:lambda:us-
   east-1:111122223333:MyCodeCommitFunction",
               "branches": [
                   "main",
                   "preprod"
   ```

```
            ],
            "name": "MyLambdaFunctionTrigger",
            "customData": "Project ID 12345"
        }
    ]
}
```

5. 要测试触发器的功能，请生成并向配置该触发器的存储库推送一个提交。在 Lambda 控制台中该函数的监控选项卡上，您应该会看到 Lambda 函数的响应。

编辑 AWS CodeCommit 仓库的触发器

您可以编辑为 CodeCommit 存储库创建的触发器。您可以更改触发器的事件和分支、响应事件时采取的操作以及其他设置。

主题

- 编辑存储库的触发器（控制台）
- 编辑存储库的触发器 (AWS CLI)

编辑存储库的触发器（控制台）

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从存储库 中，选择要编辑其存储库事件触发器的存储库。
3. 在存储库的导航窗格中，选择设置，然后选择触发器。
4. 从存储库的触发器列表中选择要编辑的触发器，然后选择编辑。
5. 对触发器进行所需的更改，然后选择保存。

编辑存储库的触发器 (AWS CLI)

1. 在终端（Linux、macOS 或 Unix）或命令提示符 (Windows) 处，运行 get-repository-triggers 命令创建一个 JSON 文件，其中包含为您的存储库配置的所有触发器的结构。例如，要创建一个名为 *MyTriggers.json # JSON* 文件，其中包含为名*MyDemoRepo*为的存储库配置的所有触发器的结构：

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >MyTriggers.json
```

此命令不返回任何内容，但是在您运行该命令的目录中创建了一个名为 *MyTriggers.json* 的文件。

2. 在纯文本编辑器中编辑该 JSON 文件，更改要编辑的触发器的触发块。用 repositoryName 对替换 configurationId 对。保存该文件。

   例如，如果要编辑存储库*MyFirstTrigger*中名为的触发器，*MyDemoRepo*使其适用于所有分支，请configurationId替repositoryName换为######中的指定main和preprod分支，然后将其删除。默认情况下，如果未指定分支，触发器将应用到存储库中的所有分支：

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. 在终端或命令行中，运行 put-repository-triggers 命令。这将更新存储库的所有触发器，包括您对*MyFirstTrigger*触发器所做的更改：

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

该命令将返回与以下示例类似的配置 ID：

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
}
```

测试 AWS CodeCommit 仓库的触发器

您可以测试为 CodeCommit 存储库创建的触发器。测试涉及使用您的存储库中的示例数据运行触发器，包括最新的提交 ID。如果存储库不存在提交历史记录，则生成由零组成的示例值。测试触发器可帮助您确认您是否已正确配置触发器目标 CodeCommit 之间的访问权限，无论是 AWS Lambda 函数通知还是亚马逊简单通知服务通知。

主题

- 测试存储库的触发器（控制台）
- 测试存储库的触发器 (AWS CLI)

测试存储库的触发器（控制台）

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从存储库中，选择要测试其存储库事件触发器的存储库。
3. 在存储库的导航窗格中，选择设置，然后选择触发器。
4. 选择要测试的触发器，然后选择测试触发器。您应看到指示成功或失败的消息。如果成功，您还应看到来自 Lambda 函数或 Amazon SNS 主题的相应操作响应。

测试存储库的触发器 (AWS CLI)

1. 在终端（Linux、macOS 或 Unix）或命令提示符 (Windows) 处，运行 get-repository-triggers 命令创建一个 JSON 文件，其中包含为您的存储库配置的所有触发器的结构。例如，要创建一个名为 *TestTrigger.json # JSON* 文件，其中包含为名为的存储库配置的所有触发器的结构，请执行以下操作： MyDemoRepo

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >TestTrigger.json
```

此命令将在您运行该命令的目录中创建一个名为 *TestTriggers.json* 的文件。

2. 在纯文本编辑器中编辑 JSON 文件并更改触发器语句。用 repositoryName 对替换 configurationId 对。保存该文件。

   例如，如果要测试存储库*MyFirstTrigger*中名为的触发器以使其适用于所有分支，请将替换为 *MyDemoRepo*，repositoryName然后将外观类似于以下内容的文件另存为 *TestTrigger.json*：configurationId

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. 在终端或命令行中，运行 test-repository-triggers 命令。这将更新存储库的所有触发器，包括您对*MyFirstTrigger*触发器所做的更改：

```
aws codecommit test-repository-triggers --cli-input-json file://TestTrigger.json
```

   该命令将返回与以下示例类似的响应：

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
    ],
    "failedExecutions": []
}
```

从 AWS CodeCommit 存储库中删除触发器

您可能想要删除不再使用的触发器。您无法撤消触发器删除操作，但可以重新创建一个触发器。

> **Note**
>
> 如果为存储库配置了一个或多个触发器，删除存储库不会删除您配置为这些触发器目标的 Amazon SNS 主题或 Lambda 函数。如果不再需要这些资源，请务必将它们也删除。

主题

从存储库中删除触发器（控制台）

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 从 Repositories (存储库) 中，选择要删除其存储库事件触发器的存储库。
3. 在存储库的导航窗格中，选择 Settings。在设置中，选择触发器。
4. 从触发器列表中选择要删除的触发器，然后选择删除。
5. 在对话框中，键入 delete 进行确认。

从存储库中删除触发器 (AWS CLI)

1. 在终端（Linux、macOS 或 Unix）或命令提示符 (Windows) 处，运行 get-repository-triggers 命令创建一个 JSON 文件，其中包含为您的存储库配置的所有触发器的结构。例如，要创建一个名为 *MyTriggers.json # JSON* 文件，其中包含为名为的存储库配置的所有触发器的结构，请执行以下操作： MyDemoRepo

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
    >MyTriggers.json
   ```

   此命令将在您运行该命令的目录中创建一个名为 *MyTriggers.json* 的文件。
2. 在纯文本编辑器中编辑 JSON 文件并删除您要删除的触发器的触发器块。用 `repositoryName` 对替换 `configurationId` 对。保存该文件。

例如，如果要*MyFirstTrigger*从名为的存储库中移除名为的触发器 *MyDemoRepo*，则应configurationId使用repositoryName######替换并删除该语句：

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "destinationArn": "arn:aws:sns:us-
east-2:111122223333:MyCodeCommitTopic",
            "branches": [
                "main",
                "preprod"
            ],
            "name": "MyFirstTrigger",
            "customData": "",
            "events": [
                "all"
            ]
        },
        {
            "destinationArn": "arn:aws:lambda:us-
east-2:111122223333:function:MyCodeCommitJSFunction",
            "branches": [],
            "name": "MyLambdaTrigger",
            "events": [
                "all"
            ]
        }
    ]
}
```

3. 在终端或命令行中，运行 put-repository-triggers 命令。这将更新存储库的触发器并删除*MyFirstTrigger*触发器：

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

该命令将返回与以下示例类似的配置 ID：

```
{
    "configurationId": "0123456-I-AM-AN-EXAMPLE"
```

```
}
```

> **Note**
>
> 要删除名为的存储库的所有触发器 *MyDemoRepo*，您的 JSON 文件将如下所示：
>
> ```
> {
>     "repositoryName": "MyDemoRepo",
>     "triggers": []
> }
> ```

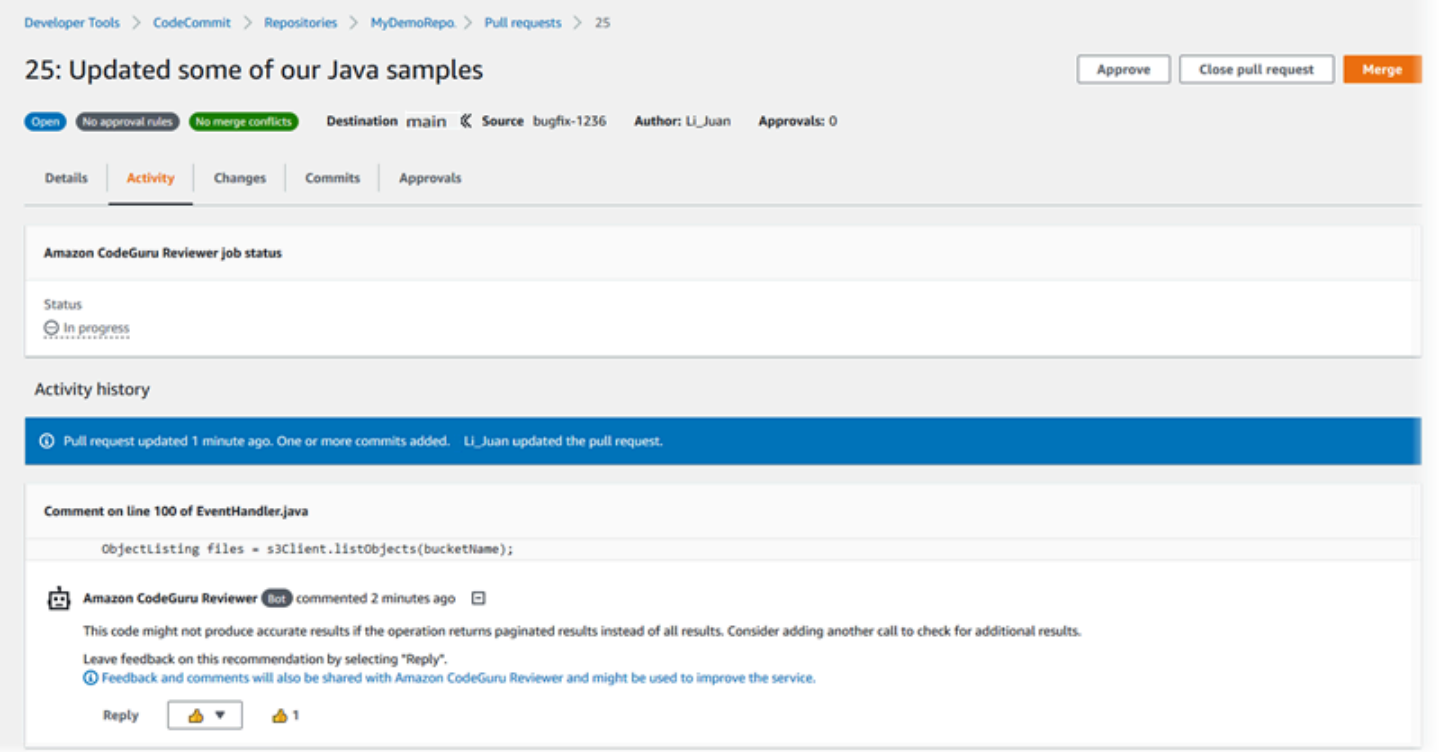
将 AWS CodeCommit 存储库与 Amazon CodeGuru Reviewer 关联或取消关联

Amazon CodeGuru Reviewer 是一项自动代码审查服务，它使用程序分析和机器学习来检测 Java 或 Python 代码中的常见问题并推荐修复方法。您可以将亚马逊 Web Services 账户中的存储库与 CodeGuru Reviewer 关联起来。当你这样做时， CodeGuru Reviewer 会创建一个服务相关角色，允许 CodeGuru Reviewer 分析在建立关联后创建的所有拉取请求中的代码。

关联存储库后， CodeGuru Reviewer 会对创建拉取请求时发现的任何问题进行分析和评论。每条评论都明确标记为来自 CodeGuru 审阅者，名称为 Amazon CodeGuru Revie wer。您可以像对拉取请求中的任何其他注释一样回复这些注释，也可以提供有关建议质量的反馈。此反馈将与 CodeGuru Reviewer 共享，有助于改进服务及其建议。

> **Note**
>
> 在存储库与其关联之前创建的拉取请求中，您将看不到 CodeGuru 来自 Reviewer 的评论。在关联之后创建的拉取请求中，可能也看不到注释，原因如下：
>
> - 拉取请求不包含 Java 或 Python 代码。
> - CodeGuru Reviewer 没有足够的时间运行和查看拉取请求中的代码。这一过程耗时最多 30 分钟。评论可以随着审查的进行而出现，但在任务状态显示为已完成之前，评论不会完成。
> - CodeGuru 审阅者未在拉取请求中的 Java 或 Python 代码中发现任何问题。
> - 代码审核作业运行失败。要查看拉取请求的审查状态，请参阅拉取请求的活动选项卡。

- 您正在更改选项卡中查看拉取请求的更改，拉取请求已更新，Amazon CodeGuru Reviewer 未在更改中发现任何问题。Amazon CodeGuru Reviewer 评论只有在对拉取请求的最新修订版发表评论时，才会显示在 “更改” 选项卡中。它们始终显示在活动选项卡中。



有关更多信息，请参阅使用 AWS CodeCommit 存储库中的拉取请求审核拉取请求、和 Amazon CodeGuru Reviewer 用户指南。

> **Note**
>
> 您必须使用具有足够权限才能将存储库与 CodeGuru Reviewer 关联或取消关联的 IAM 用户或角色登录。有关包含这些权限 CodeCommit 的托管策略的信息，请参阅适用于 CodeCommit 的 AWS 托管式策略和AWS CodeCommit 托管式策略和 Amazon CodeGuru Reviewer。有关 CodeGuru 审阅者权限和安全性的信息，请参阅 Amazon CodeGuru Reviewer 用户指南。

主题

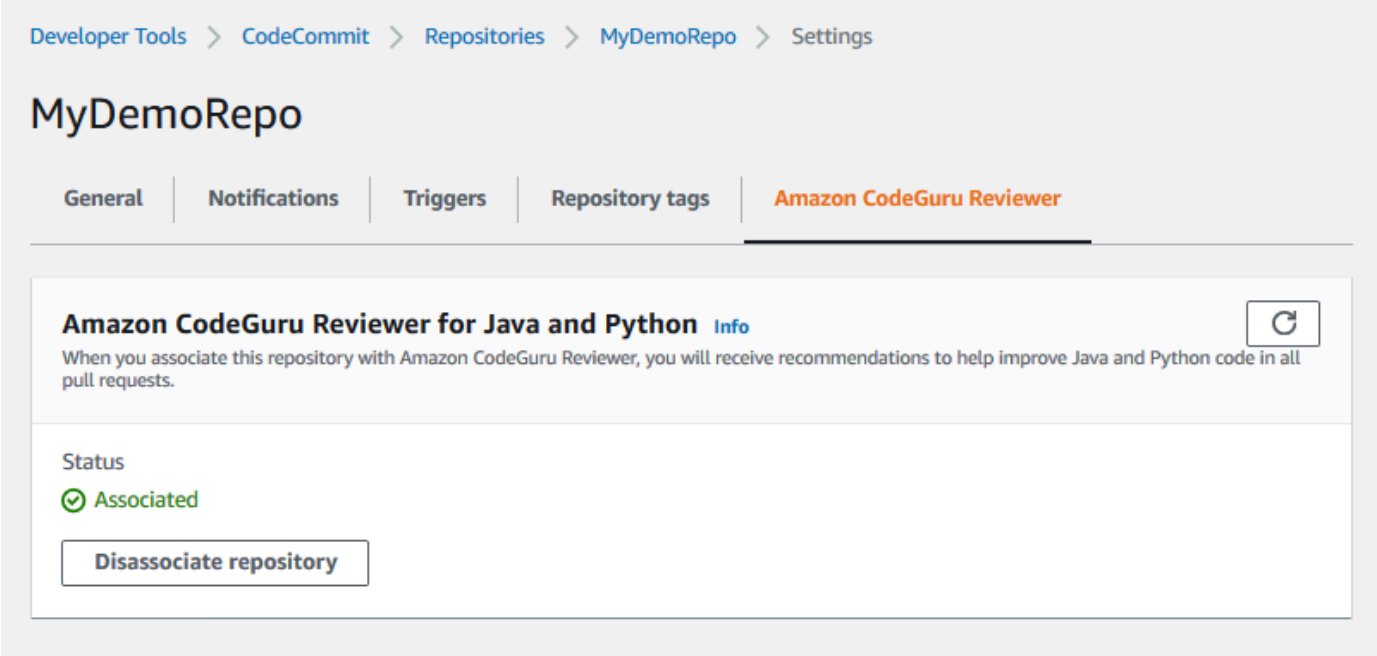
- 将存储库与 CodeGuru Reviewer 关联
- 取消仓库与 Reviewer 的 CodeGuru 关联

将存储库与 CodeGuru Reviewer 关联

使用 AWS CodeCommit 控制台快速将存储库与 CodeGuru Reviewer 关联起来。有关其他方法，请参阅 Amazon CodeGuru Reviewer 用户指南。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要与 CodeGuru Reviewer 关联的存储库的名称。
3. 选择 “设置”，然后选择 Amazon CodeGuru Reviewer。
4. 选择 Associate repository (关联存储库)。

> **Note**
>
> 将存储库与 CodeGuru Reviewer 完全关联可能需要 10 分钟。状态不会自动更新。要查看当前状态，请选择刷新按钮。



取消仓库与 Reviewer 的 CodeGuru 关联

使用 AWS CodeCommit 控制台快速解除存储库与 CodeGuru Reviewer 的关联。有关其他方法，请参阅 Amazon CodeGuru Reviewer 用户指南。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要取消与 CodeGuru Reviewer 关联的存储库的名称。
3. 选择 “设置”，然后选择 Amazon CodeGuru Reviewer。
4. 选择 Disassociate repository (取消关联存储库)。

查看 CodeCommit 存储库详情

您可以使用连接到 CodeCommit 存储库的本地存储库中的 AWS CodeCommit 控制台 AWS CLI、或 Git 来查看有关可用存储库的信息。

在按照这些说明进行操作之前，请完成设置 中的步骤。

主题

- 查看存储库详细信息（控制台）
- 查看 CodeCommit 仓库详情 (Git)
- 查看 CodeCommit 存储库详细信息 (AWS CLI)

查看存储库详细信息（控制台）

使用 AWS CodeCommit 控制台快速查看使用您的亚马逊 Web Services 账户创建的所有存储库。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，查看您登录的 AWS 区域 中的存储库相关详细信息。使用区域选择器选择不同的 AWS 区域，以查看相应区域中的存储库。
3. 选择您要查看更多详情的存储库的名称，然后执行以下操作之一：
   - 要查看 URL 来克隆存储库，请选择 Clone URL (克隆 URL)，然后选择克隆存储库时要使用的协议。此时将复制克隆 URL。要查看它，请将其粘贴到纯文本编辑器中。
   - 要查看存储库的可配置选项以及存储库 ARN 和存储库 ID 等详细信息，请在导航窗格中选择设置。

> **Note**
>
> 如果您以 IAM 用户身份登录，则可配置并保存用于查看代码的首选项和其他控制台设置。有关更多信息，请参阅 使用用户首选项。

查看 CodeCommit 仓库详情 (Git)

要使用本地存储库中的 Git 来查看 CodeCommit 存储库的详细信息，请运行git remote show命令。

在执行这些步骤之前，请将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

1. 运行git remote show ***remote-name***命令，其中 *remote-name* 是 CodeCommit 存储库的别名（默认为 ）。`origin`

   > **Tip**
   >
   > 要获取 CodeCommit 仓库名称及其 URL 的列表，请运行该git remote -v命令。

   例如，要查看别名为 CodeCommit 存储库的详细信息，请执行`origin`以下操作：

   ```
   git remote show origin
   ```

2. 对于 HTTPS：

   ```
   * remote origin
     Fetch URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     Push  URL: https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
     HEAD branch: (unknown)
     Remote branches:
       MyNewBranch tracked
       main tracked
     Local ref configured for 'git pull':
       MyNewBranch merges with remote MyNewBranch (up to date)
     Local refs configured for 'git push':
       MyNewBranch pushes to MyNewBranch (up to date)
       main pushes to main (up to date)
   ```

   对于 SSH：

```
* remote origin
  Fetch URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
  Push  URL: ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
  HEAD branch: (unknown)
  Remote branches:
    MyNewBranch tracked
    main tracked
  Local ref configured for 'git pull':
    MyNewBranch merges with remote MyNewBranch (up to date)
  Local refs configured for 'git push':
    MyNewBranch pushes to MyNewBranch (up to date)
    main pushes to main (up to date)
```

> **Tip**
>
> 要查找您的 IAM 用户的 SSH 密钥 ID，请打开 IAM 控制台并在 IAM 用户详细信息页面上展开安全凭证。SSH 密钥 ID 可以在的 SSH 密钥中找到 AWS CodeCommit。

有关更多选项，请参阅 Git 文档。

查看 CodeCommit 存储库详细信息 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

要使用查看存储库的详细信息，请运行以下命令： AWS CLI

- 要查看 CodeCommit 仓库名称及其对应 ID 的列表，请运行列表存储库。
- 要查看有关单个 CodeCommit 存储库的信息，请运行 get- repository。
- 要查看有关中多个存储库的信息 CodeCommit，请运行batch-get-repositories。

查看 CodeCommit 存储库列表

1. 运行 list-repositories 命令：

```
aws codecommit list-repositories
```

您可以使用可选的 `--sort-by` 或 `--order` 选项来更改返回信息的顺序。

2. 如果成功，此命令将输出一个`repositories`对象，其中包含与 Amazon Web Services 账户 CodeCommit 关联的所有存储库的名称和 ID。

   下面是前面命令的一些示例输出：

```
{
    "repositories": [
        {
           "repositoryName": "MyDemoRepo",
           "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
        },
        {
           "repositoryName": "MyOtherDemoRepo",
           "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
        }
    ]
}
```

查看有关单个 CodeCommit 存储库的详细信息

1. 运行get-repository命令，使用`--repository-name`选项指定 CodeCommit 存储库的名称。

   > **Tip**
   >
   > 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

   例如，要查看名为的 CodeCommit 存储库的详细信息`MyDemoRepo`：

```
aws codecommit get-repository --repository-name MyDemoRepo
```

2. 如果成功，此命令会输出一个包含以下信息的 `repositoryMetadata` 对象：

   - 存储库的名称 (`repositoryName`)。
   - 存储库说明 (`repositoryDescription`)。
   - 系统生成的存储库的唯一 ID (`repositoryId`)。
   - 与存储库关联的 Amazon Web Services 账户的 ID (`accountId`)。

下面是前面示例命令的一些示例输出：

```
{
        "repositoryMetadata": {
            "creationDate": 1429203623.625,
            "defaultBranch": "main",
            "repositoryName": "MyDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "lastModifiedDate": 1430783812.0869999,
            "repositoryDescription": "My demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "accountId": "111111111111"
        }
}
```

查看有关多个 CodeCommit 存储库的详细信息

1. 使用 batch-get-repositories 选项运行 `--repository-names` 命令。在每个 CodeCommit 存储库名称之间添加一个空格。

   > **Tip**
   >
   > 要获取存储库的名称，请运行 list- CodeCommit repositories 命令。

   例如，要查看名为MyDemoRepo和的两个 CodeCommit 存储库的详细信息，请执行MyOtherDemoRepo以下操作：

   ```
   aws codecommit batch-get-repositories --repository-names MyDemoRepo MyOtherDemoRepo
   ```

2. 如果成功，此命令会输出一个包含以下信息的对象：

   - 找不到的所有 CodeCommit 存储库的列表 (`repositoriesNotFound`)。
   - CodeCommit 存储库列表 (`repositories`)。每个 CodeCommit 存储库名称后面都有：

- 存储库说明 (repositoryDescription)。
- 系统生成的存储库的唯一 ID (repositoryId)。
- 与存储库关联的 Amazon Web Services 账户的 ID (accountId)。

下面是前面示例命令的一些示例输出：

```
{
        "repositoriesNotFound": [],
        "repositories": [
             {
                "creationDate": 1429203623.625,
                "defaultBranch": "main",
                "repositoryName": "MyDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "lastModifiedDate": 1430783812.0869999,
                "repositoryDescription": "My demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
                "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
                "accountId": "111111111111"
            },
            {
                "creationDate": 1429203623.627,
                "defaultBranch": "main",
                "repositoryName": "MyOtherDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "lastModifiedDate": 1430783812.0889999,
                "repositoryDescription": "My other demonstration repository",
                "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
                "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
                "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo",
                "accountId": "111111111111"
            }
        ],
        "repositoriesNotFound": []
    }
```

更改 AWS CodeCommit 存储库设置

您可以使用 AWS CLI 和 AWS CodeCommit 控制台来更改 CodeCommit 存储库的设置，例如其描述或名称。

> **Important**
>
> 更改存储库的名称可能导致在其远程 URL 中使用旧名称的所有本地存储库连接中断。运行 git remote set-url 命令可以更新远程 URL，使其使用新的存储库名称。

主题

更改存储库设置（控制台）

要使用 AWS CodeCommit 控制台在中更改 CodeCommit 存储库的设置 AWS CodeCommit，请按照以下步骤操作。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要更改设置的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。
4. 要更改存储库的名称，请在存储库名称中的名称文本框中输入新名称，然后选择保存。在出现提示时，确认您的选择。

   > **Important**
   >
   > 更改 AWS CodeCommit 存储库的名称将更改用户连接到存储库所需的 SSH 和 HTTPS 网址。在更新连接设置之前，用户无法连接到此存储库。此外，由于存储库的 ARN 会发生更改，更改存储库名称会使依赖该存储库 ARN 的所有 IAM 用户策略失效。
   > 更改名称后，所有用户都必须使用 git remote set-url 命令并指定要使用的新 URL，然后才能连接到存储库。例如，如果您将仓库名称从 MyDemoRepo 更改为 MyRenamedDemoRepo，则使用 HTTPS 连接仓库的用户将运行以下 Git 命令：

```
git remote set-url origin https://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyRenamedDemoRepo
```

使用 SSH 连接该存储库的用户需要运行下面的 Git 命令：

```
git remote set-url origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyRenamedDemoRepo
```

有关更多选项，请参阅 Git 文档。

5. 要更改存储库的说明，请修改 Description 文本框中的文本，然后选择 Save 。

> **Note**
>
> “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的应用程序开发人员，并且计划在 Web 浏览器中显示存储库描述字段，请参阅 CodeCommit API 参考。

6. 要更改默认分支，请在默认分支中选择分支下拉列表，然后选择其他分支。选择保存。
7. 要更改用于 AWS KMS 加密和解密存储库中数据的加密密钥，请在存储库加密密钥中，选择其中一个AWS 托管式密钥或客户托管密钥以指定要使用的密钥类型。如果选择客户管理的密钥，请输入密钥的 ARN。选择保存。
8. 要删除存储库，请选择 Delete repository。在 Type the name of the repository to confirm deletion (键入存储库名称以确认删除) 旁边的框中，输入 **`delete`**，然后选择 Delete (删除)。

> **Important**
>
> 在中删除此存储库后 AWS CodeCommit，您将无法再将其克隆到任何本地存储库或共享存储库。也无法再从任何本地存储库或共享存储库向其推送数据或从其提取数据。并且无法撤消。

更改 AWS CodeCommit 存储库设置 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

AWS CLI 要使用在中更改 CodeCommit 存储库的设置 AWS CodeCommit，请运行以下一个或多个命令：

- update-repository-description更改 CodeCommit 存储库的描述。
- update-repository-name更改 CodeCommit 存储库的名称。

更改 CodeCommit 仓库的描述

1. 运行 update-repository-description 命令，并指定：

   - CodeCommit 存储库的名称（带`--repository-name`选项）。

     > **Tip**
     >
     > 要获取 CodeCommit 存储库的名称，请运行list-repositories命令。

   - 新存储库说明 (使用 `--repository-description` 选项)。

     > **Note**
     >
     > “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的应用程序开发人员，并且计划在 Web 浏览器中显示存储库描述字段，请参阅 CodeCommit API 参考。

   例如，要将名为的 CodeCommit 存储库的描述更改`MyDemoRepo`为`This description was changed`：

   ```
   aws codecommit update-repository-description --repository-name MyDemoRepo --
   repository-description "This description was changed"
   ```

   该命令只在出现错误时生成输出。

2. 要验证更改后的描述，请运行get-repository命令，指定使用该`--repository-name`选项更改其描述的 CodeCommit 存储库的名称。

   该命令的输出会在 `repositoryDescription` 中显示已更改的文本。

更改 CodeCommit 仓库的名称

1. 运行 update-repository-name 命令，并指定：

   - CodeCommit 存储库的当前名称（带--old-name选项）。

     > **Tip**
     >
     > 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

   - CodeCommit 存储库的新名称（带--new-name选项）。

   例如，要将名为 MyDemoRepo 的存储库更改为 MyRenamedDemoRepo：

   ```
   aws codecommit update-repository-name --old-name MyDemoRepo --new-name
    MyRenamedDemoRepo
   ```

   该命令只在出现错误时生成输出。

   > **Important**
   >
   > 更改 AWS CodeCommit 存储库的名称会更改用户连接到存储库所需的 SSH 和 HTTPS 网址。在更新连接设置之前，用户无法连接到此存储库。此外，由于存储库的 ARN 会发生更改，更改存储库名称会使依赖该存储库 ARN 的任何 IAM 用户策略失效。

2. 要验证更改的名称，请运行 list-repositories 命令并查看存储库名称列表。

在本地存储库和 AWS CodeCommit 存储库之间同步更改

您可以使用 Git 在本地存储库和连接到本地 CodeCommit 存储库的存储库之间同步更改。

要将更改从本地存储库推送到 CodeCommit 存储库，请运行git push ***remote-name branch-name***。

要从存储库中提取对本地 CodeCommit 存储库的更改，请运行git pull ***remote-name branch-name***。

对于推送和拉取，*remote-name* 是本地存储库使用的昵称。 CodeCommit *branch-nam* e 是 CodeCommit 存储库中要推送或从中提取的分支的名称。

> **Tip**
>
> 要获取本地存储库用于 CodeCommit 存储库的昵称，请运行git remote。要获取分支名称列表，请运行 git branch。当前分支的名称旁边会显示星号 (*)。（您也可以运行 git status 来显示当前分支名称。）

> **Note**
>
> 如果您克隆了存储库，则从本地存储库的角度来看，re *mote-name* 不是存储库的名称。CodeCommit 在克隆存储库时，*remote-name* 会自动设为 `origin`。

例如，要将更改从本地存储库推送到 CodeCommit 存储库中使用昵称`origin`的`main`分支，请执行以下操作：

```
git push origin main
```

同样，要从存储库中使用昵称`origin`的`main`分支中提取对本地 CodeCommit 存储库的更改，请执行以下操作：

```
git pull origin main
```

> **Tip**
>
> 如果向 git push 添加 -u 选项，则会设置上游跟踪信息。例如，如果您运行了 git push -u origin main，则日后只需运行 git push 和 git pull，无需指定 *remote-name branch-name*。要获取上游跟踪信息，请运行 git remote show ***remote-name***（例如，git remote show origin）。

有关更多选项，请参阅 Git 文档。

将提交推送到其他 Git 存储库

您可以将本地存储库配置为将更改推送到两个远程存储库。例如，您可能想要在尝试使用 AWS CodeCommit时继续使用现有的 Git 存储库解决方案。按照以下基本步骤将本地存储库中的更改推送到 CodeCommit 单独的 Git 存储库。

> **Tip**
>
> 如果您没有 Git 存储库，则可以在除之外的服务上创建一个空仓库， CodeCommit然后将仓库迁移到该存储 CodeCommit 库。您应按照迁移到 CodeCommit 中的步骤进行操作。

1. 在命令提示符或终端中，切换到本地存储库目录，然后运行 git remote -v 命令。您应该可以看到类似于如下所示的输出内容：

   对于 HTTPS：

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

   对于 SSH：

   ```
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
   ```

2. 运行git remote set-url --add --push origin ***git-repository-name***命令，其中*git-repository-name*是你要托管代码的 Git 存储库的 URL 和名称。这会将 origin 的推送目标更改为该 Git 存储库。

   > **Note**
   >
   > git remote set-url --add --push 会覆盖推送的默认 URL，因此您必须运行该命令两次，如后面的步骤所示。

   例如，以下命令将 Origin 的推送更改为 s *ome-URL/*：MyDestinationRepo

   ```
   git remote set-url --add --push origin some-URL/MyDestinationRepo
   ```

   该命令不返回任何内容。

> **Tip**
>
> 如果要推送到需要凭证的 Git 存储库，请确保在凭证辅助程序或 *some-URL* 字符串配置中配置这些凭证。否则，到该存储库的推送操作将失败。

3. 再次运行 git remote -v 命令，它应产生类似以下内容的输出：

   对于 HTTPS：

   ```
   origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  some-URL/MyDestinationRepo (push)
   ```

   对于 SSH：

   ```
   origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
   origin  some-URL/MyDestinationRepo (push)
   ```

4. 现在添加 CodeCommit 存储库。git remote set-url --add --push origin再次运行，这次使用仓库的 URL 和存储 CodeCommit库名称。

   例如，以下命令将 O rigin 的推送添加到 https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo：

   对于 HTTPS：

   ```
   git remote set-url --add --push origin https://git-codecommit.us-
   east-2.amazonaws.com/v1/repos/MyDemoRepo
   ```

   对于 SSH：

   ```
   git remote set-url --add --push origin ssh://git-codecommit.us-
   east-2.amazonaws.com/v1/repos/MyDemoRepo
   ```

   该命令不返回任何内容。

5. 再次运行 git remote -v 命令，它应产生类似以下内容的输出：

   对于 HTTPS：

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

对于 SSH：

```
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  some-URL/MyDestinationRepo (push)
origin  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

你现在有两个 Git 存储库作为推送的目的地，但你的推送会先到####/MyDestinationRepo 。如果到该存储库的推送失败，则您的提交不会被推送到任何一个存储库。

> **Tip**
>
> 如果另一个存储库需要您要手动输入的凭据，请考虑更改推送顺序，以便 CodeCommit 先推送到。运行 git remote set-url --delete 删除首先推送到的存储库，然后运行 git remote set-url --add 再次添加它，使其成为列表中的第二个推送目标。
> 有关更多选项，请参阅 Git 文档。

6. 要验证您是否正在推送到两个远程存储库，请使用文本编辑器在本地存储库中创建下面的文本文件：

```
bees.txt
-------
Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
```

7. 运行 git add 将更改暂存在本地存储库中：

```
git add bees.txt
```

8. 运行 git commit 提交本地存储库中的更改：

```
git commit -m "Added bees.txt"
```

9. 要将本地存储库的提交推送到远程存储库，请运行 git push -u ***remote-name branch-name***，其中 *remote-name* 是本地存储库用于远程存储库的别名，*branch-name* 是要推送到存储库的分支的名称。

> **Tip**
>
> 您只需在第一次推送时使用 -u 选项。然后将设置上游跟踪信息。

例如，运行 git push -u origin main 会显示推送到两个远程存储库中的预期分支，并产生类似以下内容的输出：

对于 HTTPS：

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   a5ba4ed..250f6c3  main -> main
```

对于 SSH：

```
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To some-URL/MyDestinationRepo
   a5ba4ed..250f6c3  main -> main
Counting objects: 5, done.
Delta compression using up to 4 threads.
```

```
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 5.61 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
   a5ba4ed..250f6c3  main -> main
```

有关更多选项，请参阅 Git 文档。

使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限

您可以为其他 AWS 账户中的 IAM 用户和群组配置对 CodeCommit 存储库的访问权限。此过程通常称为跨账户访问。本节提供示例和 step-by-step 说明，说明如何为属于 AWS 另一个账户（称为 AccountB）*MySharedDemoRepo*中的 IAM 群组 *DevelopersWithCrossAccountRepositoryAccess*（称为 AccountA）的 IAM 用户配置跨 AWS 账户访问权限，该存储库位于美国东部（俄亥俄州）地区的账户（称为 AccountA）。

本节分为三个部分：

- AccountA 中管理员的操作。
- AccountB 中管理员的操作。
- AccountB 中存储库用户的操作。

要配置跨账户访问，请执行以下操作：

- AccountA 中的管理员以 IAM 用户身份登录，该用户拥有在 IAM 中创建和管理存储库以及在 IAM 中 CodeCommit 创建角色所需的权限。如果您使用的是托管策略，请将 IAM FullAccess 和应用 AWSCodeCommitFullAccess 于此 IAM 用户。

  AccountA 的示例账户 ID 为 *111122223333*。
- AccountB 中的管理员以 IAM 用户身份登录，该用户具有创建和管理 IAM 用户和组以及为用户和组配置策略所需的权限。如果您使用托管策略，请将 IAM 应用FullAccess 于此 IAM 用户。

  AccountB 的示例账户 ID 为 *888888888888*。
- 为了模拟开发者的活动，AccountB 中的存储库用户以 IAM 用户身份登录，该用户是为允许访问 AccountA 中的仓库而创建的 IAM 群组的成员。 CodeCommit 必须为此账户配置：
  - AWS 管理控制台访问权限。

- 连接 AWS 资源时使用的访问密钥和私有密钥，以及访问 AccountA 中的仓库时要扮演的角色的 ARN。
- 在其上克隆存储库的本地计算机上的 git-remote-codecommit 实用程序。此实用程序需要 Python 及其安装程序 pip。您可以从 Python 程序包索引网站上通过 git-remote-codecommit 下载该实用程序。

有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤和 IAM 用户。

主题

跨账户存储库访问：AccountA 中管理员的操作

要允许 AccountB 中的用户或组访问 AccountA 中的存储库，AccountA 管理员必须：

- 在 AccountA 中创建一个授予对存储库的访问权限的策略。
- 在 AccountA 中创建一个可由 AccountB 中的 IAM 用户和组代入的角色。
- 将 策略附加到该角色。

以下各节提供了步骤和示例。

主题

步骤 1：在 AccountA 中创建用于存储库访问的策略

您可以在 AccountA 中创建一个授予对 AccountB 中存储库的访问权限的策略。根据您希望允许的访问级别，执行以下操作之一：

- 配置策略以允许 AccountB 用户访问特定存储库，但不允许这些用户查看 AccountA 中所有存储库的列表。

- 配置额外访问权限以允许 AccountB 用户从 AccountA 的所有存储库列表中选择存储库。

创建用于存储库访问的策略

1. 以有权在 AccountA 中创建策略的 IAM 用户身份登录 AWS 管理控制台。
2. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择策略。
4. 选择创建策略。
5. 选择 JSON 选项卡，并将以下 JSON 策略文档粘贴到 JSON 文本框中。将 *us-east-2* 替换为存储库的，将 111122223333 替换为 accountA 的账户 ID#将 accountA 中的仓库名称替换为：AWS 区域 *MySharedDemoRepo* CodeCommit

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:BatchGet*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Describe*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:Test*",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [
            "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
        ]
    }
]
}
```

如果您希望担任此角色的用户能够在 CodeCommit 控制台主页上查看存储库列表，请在策略中添加其他声明，如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:BatchGet*",
                "codecommit:Create*",
                "codecommit:DeleteBranch",
                "codecommit:Get*",
                "codecommit:List*",
                "codecommit:Describe*",
                "codecommit:Put*",
                "codecommit:Post*",
                "codecommit:Merge*",
                "codecommit:Test*",
                "codecommit:Update*",
                "codecommit:GitPull",
                "codecommit:GitPush"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-2:111122223333:MySharedDemoRepo"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*"
        }
    ]
}
```

利用该访问权限，使用此策略代入此角色的用户能够更轻松地找到其有权访问的存储库。他们可以从列表中选择存储库的名称，并定位到共享存储库的主页 (Code)。虽然用户无法访问列表中显示的任何其他存储库，但他们可以在控制面板页面上查看 AccountA 中的存储库。

如果您不想让担任该角色的用户能够查看 AccountA 中所有仓库的列表，请使用第一个策略示例，但要确保在控制台中向这些用户发送指向共享仓库主页的直接链接。 CodeCommit

6. 选择查看策略。策略验证器会报告语法错误（例如，如果您忘记将示例 Amazon Web Services 账户 ID 和存储库名称替换为您的 Amazon Web Services 账户 ID 和存储库名称）。
7. 在查看策略页面上，输入策略的名称（例如 *CrossAccountAccessForMySharedDemoRepo*）。您也可以提供此策略的可选描述。选择 创建策略。

步骤 2：在 AccountA 中创建用于存储库访问的角色

配置策略后，创建 AccountB 中的 IAM 用户和组可代入的角色，然后向此角色附加策略。

创建用于存储库访问的角色

1. 在 IAM 控制台中，选择角色。
2. 选择 创建角色。
3. 选择另一个 Amazon Web Services 账户。
4. 在账户 ID 中，输入 AccountB 的 Amazon Web Services 账户 ID（例如，*888888888888*）。选择下一步：权限。
5. 在附加权限策略中，选择您在上一个过程中创建的策略（*CrossAccountAccessForMySharedDemoRepo*）。选择 下一步: 审核。
6. 在角色名称中，输入角色的名称（例如 *MyCrossAccountRepositoryContributorRole*）。您还可以输入可选描述，帮助他人了解角色的用途。
7. 选择 创建角色。
8. 打开您刚刚创建的角色，并复制角色 ARN (例如，arn:aws:iam::*111122223333*:role/*MyCrossAccountRepositoryContributorRole*)。您需要向 AccountB 管理员提供此 ARN。

跨账户存储库访问：AccountB 中管理员的操作

要允许 AccountB 中的用户或组访问 AccountA 中的存储库，AccountB 管理员必须在 AccountB 中创建一个组。必须为此组配置一个策略，该策略允许组成员代入由 AccountA 管理员创建的角色。

以下各节提供了步骤和示例。

主题

步骤 1：为 AccountB 用户创建用于存储库访问的 IAM 组

管理 AccountB 中的哪些 IAM 用户可访问 AccountA 存储库的最简单方法是，在 AccountB 中创建一个有权代入 AccountA 中的角色的 IAM 组，然后将 IAM 用户添加到该组。

创建用于跨账户存储库访问的组

1. 以具有创建 IAM 群组和策略以及 AWS 管理 AccountB 中的 IAM 用户所需权限的 IAM 用户身份登录管理控制台。
2. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 在 IAM 控制台中，选择组。
4. 选择 Create New Group (创建新组)。
5. 在组名中，输入该组的名称（例如，*DevelopersWithCrossAccountRepositoryAccess*）。选择下一步。
6. 在 Attach Policy 中，选择 Next Step。您在下一个过程中创建跨账户策略。完成组的创建。

步骤 2：创建策略并将用户添加到 IAM 组

现在您已拥有一个组，请创建策略来允许此组的成员代入允许其访问 AccountA 中的存储库的角色。然后，向该组添加 AccountB 中的 IAM 用户，您希望这些用户能够访问 AccountA 中的存储库。

为组创建策略并向组添加用户

1. 在 IAM 控制台中，选择群组，然后选择您刚刚创建的群组的名称（例如 *DevelopersWithCrossAccountRepositoryAccess*）。
2. 选择权限选项卡。展开内联策略，然后选择用于创建内联策略的链接。(如果您配置的是已拥有内联策略的组，请选择创建组策略。)
3. 选择 Custom Policy，然后选择 Select。
4. 在策略名称中，输入策略的名称（例如 *AccessPolicyForSharedRepository*）。
5. 在 Policy Document (策略文档) 中，粘贴以下策略。***##Resource## ARN ######## # A ####### ARN####arn: aws: iam:: 111122223333: role/########## ##MyCrossAccountRepositoryContributorRole***有关管理员在 AccountA 中创建的策略的更多信息，请参阅步骤 1：在 AccountA 中创建用于存储库访问的策略。

```
{
  "Version": "2012-10-17",
  "Statement": {
```

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":
 "arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole"
  }
}
```

6. 选择用户选项卡。选择向组添加用户，然后添加 AccountB IAM 用户。例如，您可以将用户名为 *Saanvi_Sarkar* 的 IAM 用户添加到组。

> **Note**
>
> AccountB 中的用户必须具有编程访问权限，包括访问密钥和密钥，才能配置其本地计算机以访问共享 CodeCommit 存储库。如果您要创建 IAM 用户，请确保保存访问密钥和私有密钥。为确保您的 AWS 账户的安全，私有访问密钥仅在您创建它时可用。

跨账户存储库访问：AccountB 中存储库用户的操作

要访问 AccountA 中的存储库，AccountB 组中的用户必须配置其本地计算机以访问存储库。以下各节提供了步骤和示例。

主题

- 步骤 1：配置 AWS CLI 和 Git，让 AccountB 用户能够访问 AccountA 中的仓库
- 第 2 步：在 AccountA 中克隆并访问 CodeCommit 仓库

步骤 1：配置 AWS CLI 和 Git，让 AccountB 用户能够访问 AccountA 中的仓库

您无法使用 SSH 密钥或 Git 凭证来访问另一个 Amazon Web Services 账户中的存储库。AccountB 用户必须将其计算机配置为使用git-remote-codecommit（推荐）或凭据助手来访问 AccountA CodeCommit 中的共享存储库。不过，您在访问 AccountB 中的存储库时，可以继续使用 SSH 密钥或 Git 凭证。

请按照以下步骤使用 git-remote-codecommit 配置访问权限。如果尚未安装 git-remote-codecommit，请在 Python 程序包索引网站上通过 git-remote-codecommit 下载它。

配置 AWS CLI 和 Git 以实现跨账户访问

1. 在本地计算机 AWS CLI 上安装。请参阅安装 AWS CLI 中有关您的操作系统的说明。

2. 在本地计算机上安装 Git。要安装 Git，建议访问 Git 下载或 Git for Windows 等网站。

> **Note**
>
> CodeCommit 支持 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。Git 是一个不断发展的平台，会定期进行更新。有时，功能更改可能会影响其工作方式 CodeCommit。如果您在使用特定版本的 Git 时遇到问题 CodeCommit，请查看中的信息故障排除。

3. 从终端或命令行中，在要克隆存储库的目录位置，运行 git config --local user.name 和 git config --local user.email 命令可为您将对存储库做出的提交设置用户名和电子邮件。例如：

```
git config --local user.name "Saanvi Sarkar"
git config --local user.email saanvi_sarkar@example.com
```

这些命令未返回任何内容，但您指定的电子邮件和用户名将与您对 AccountA 中的存储库进行的提交关联。

4. 运行 aws configure --profile 命令以配置在连接到 AccountB 中的资源时要使用的默认配置文件。在系统提示时，请提供您的 IAM 用户的访问密钥和私有密钥。

> **Note**
>
> 如果您已经安装 AWS CLI 并配置了配置文件，则可以跳过此步骤。

例如，运行以下命令创建用于访问美国东部（俄亥俄州）AccountB (us-east-2) 中 AWS 资源的默认 AWS CLI 配置文件：

```
aws configure
```

当系统提示时，请提供以下信息：

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

5. 再次运行 aws configure --profile 命令可配置在连接到 AccountA 中的存储库时要使用的命名配置文件。在系统提示时，请提供您的 IAM 用户的访问密钥和私有密钥。例如，运行以下命令创建一个名为的 AWS CLI 配置文件 *MyCrossAccountAccessProfile*，用于访问美国东部（俄亥俄州）AccountA（us-east-2）中的存储库：

```
aws configure --profile MyCrossAccountAccessProfile
```

当系统提示时，请提供以下信息：

```
AWS Access Key ID [None]: Your-IAM-User-Access-Key
AWS Secret Access Key ID [None]: Your-IAM-User-Secret-Access-Key
Default region name ID [None]: us-east-2
Default output format [None]: json
```

6. 在纯文本编辑器中，打开 config 文件（也称为 AWS CLI 配置文件）。根据您的操作系统，此文件可能位于 ~/.aws/config（Linux、macOS 或 Unix），或者位于 *drive*:\Users*USERNAME*\.aws\config (Windows)。

7. 在文件中，找到与您为访问 AccountB 中的存储库而配置的默认配置文件相对应的条目。如下所示：

```
[default]
region = us-east-2
output = json
```

将 account 添加到配置文件配置。提供 AccountB 的 AWS 账户 ID。例如：

```
[default]
account = 888888888888
region = us-east-2
output = json
```

8. 在文件中，找到与您刚刚创建的*MyCrossAccountAccessProfile*配置文件对应的条目。如下所示：

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
output = json
```

将 account、role_arn 和 source_profile 添加到配置文件配置。提供 AccountA 的 Amazon Web Services 账户 ID、AccountA 中您将代入（以访问其他账户中的存储库）的角色的 ARN，以及您在 AccountB 中的默认 AWS CLI 配置文件的名称。例如：

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
account = 111122223333
role_arn = arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole
source_profile = default
output = json
```

保存更改并关闭纯文本编辑器。

第 2 步：在 AccountA 中克隆并访问 CodeCommit 仓库

运行git clonegit push、和git pull以克隆、推送到跨账户存储库和从中拉取跨账户 CodeCommit存储库。您还可以登录 AWS 管理控制台、切换角色并使用 CodeCommit 控制台与其他账户中的存储库进行交互。

> **Note**
>
> 根据 IAM 角色的配置方式，您可能可以在默认页面上查看的存储库 CodeCommit。如果您无法查看存储库，请仓库管理员通过电子邮件向您发送指向 CodeCommit 控制台中共享仓库代码页面的 URL 链接。该 URL 类似于以下内容：
>
> ```
> https://console.aws.amazon.com/codecommit/home?region=us-east-2#/
> repository/MySharedDemoRepo/browse/HEAD/--/
> ```

将跨账户存储库克隆到本地计算机

1. 在命令行或终端，在要克隆存储库的目录中，运行带 HTTPS (GRC) 克隆 URL 的 git clone 命令。例如：

   ```
   git clone codecommit://MyCrossAccountAccessProfile@MySharedDemoRepo
   ```

   除非另行指定，否则会将存储库克隆到与存储库同名的子目录中。

2. 将目录更改为克隆的存储库，并添加或更改文件。例如，您可以添加一个名为 *NewFile.txt* 的文件。
3. 将文件添加到本地存储库的跟踪更改中，提交更改，然后将文件推送到 CodeCommit 存储库。例如：

```
git add NewFile.txt
git commit -m "Added a file to test cross-account access to this repository"
git push
```

有关更多信息，请参阅 Git 和 AWS CodeCommit 入门。

现在您已经添加了文件，请前往 CodeCommit 控制台查看您的提交、查看其他用户对存储库的更改、参与拉取请求等。

在控制台中访问跨账户存储库 CodeCommit

1. 以 AccountA AWS Management Console 中被授予跨账户访问权限的 IAM 用户身份登录账户 B (*888888888888*)。
2. 在导航栏中选择您的用户名，然后在下拉列表中选择切换角色。

> **Note**
>
> 如果这是您首次选择此选项，请查看该页面上的信息，然后再次选择切换角色。

3. 在 Switch Role (切换角色) 页面上，执行以下操作：
   - 在账户中，输入 AccountA 的账户 ID（例如，*111122223333*）。
   - 在“角色”中，输入您要代入的 accountA 中访问存储库的角色名称（例如）。*MyCrossAccountRepositoryContributorRole*
   - 在 Display Name (显示名称) 中，输入此角色的友好名称。当您代入此角色时，该名称将显示在控制台中。此外，当您下次想在控制台中切换角色时，该名称会显示在已代入角色的列表中。
   - (可选) 在颜色中，选择显示名称的颜色标签。
   - 选择 Switch Role。

有关更多信息，请参阅切换到角色（AWS Management Console）。

4. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。

   如果已代入角色有权查看 AccountA 中的存储库的名称，您会看到一个存储库列表，并且会显示一条错误消息，告知您无权查看其状态。这是预料之中的行为。从该列表中选择共享存储库的名称。

   如果已代入角色无权查看 AccountA 中的存储库的名称，您会看到一条错误消息和一个没有存储库的空白列表。粘贴指向存储库的 URL 链接，或修改控制台链接并将 `/list` 更改为共享存储库的名称 (例如，`/MySharedDemoRepo`)。

5. 在代码中，找到您已从本地计算机中添加的文件的名称。选择该名称以浏览文件中的代码，然后浏览存储库的其余内容并开始使用其功能。

   有关更多信息，请参阅 入门 AWS CodeCommit。

删除存储 AWS CodeCommit 库

您可以使用 CodeCommit 控制台或 AWS CLI 删除 CodeCommit 存储库。

> **Note**
>
> 删除存储库不会删除该存储库的任何本地副本（本地存储库）。要删除本地存储库，请使用您的本地计算机的目录和文件管理工具。

主题

删除存储 CodeCommit 库（控制台）

按照以下步骤使用 CodeCommit 控制台删除 CodeCommit 存储库。

> **Important**
>
> 删除 CodeCommit 存储库后，您将无法再将其克隆到任何本地存储库或共享存储库。也无法再从任何本地存储库或共享存储库中拉取数据或将数据推送到其中。并且无法撤消。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要删除的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。
4. 在 General (常规) 选项卡上的 Delete repository (删除存储库) 中，选择 Delete repository (删除存储库)。输入 **delete**，然后选择删除。存储库将被永久删除。

> **Note**
>
> 删除中的存储库 CodeCommit 不会删除任何本地存储库。

删除本地存储库

使用您的本地计算机的目录和文件管理工具删除包含本地存储库的目录。

删除本地存储库不会删除它可能与之连接的任何 CodeCommit 存储库。

删除存储 CodeCommit 库 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

要使用删除 CodeCommit 存储库，请运行delete-repository命令，指定要删除的 CodeCommit 存储库的名称（使用`--repository-name`选项）。 AWS CLI

> **Important**
>
> 删除 CodeCommit 存储库后，您将无法再将其克隆到任何本地存储库或共享存储库。也无法再从任何本地存储库或共享存储库中拉取数据或将数据推送到其中。并且无法撤消。

> **Tip**
>
> 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

例如，要删除名为 MyDemoRepo 的存储库：

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

如果成功，则永久删除的 CodeCommit 存储库的 ID 将显示在输出中：

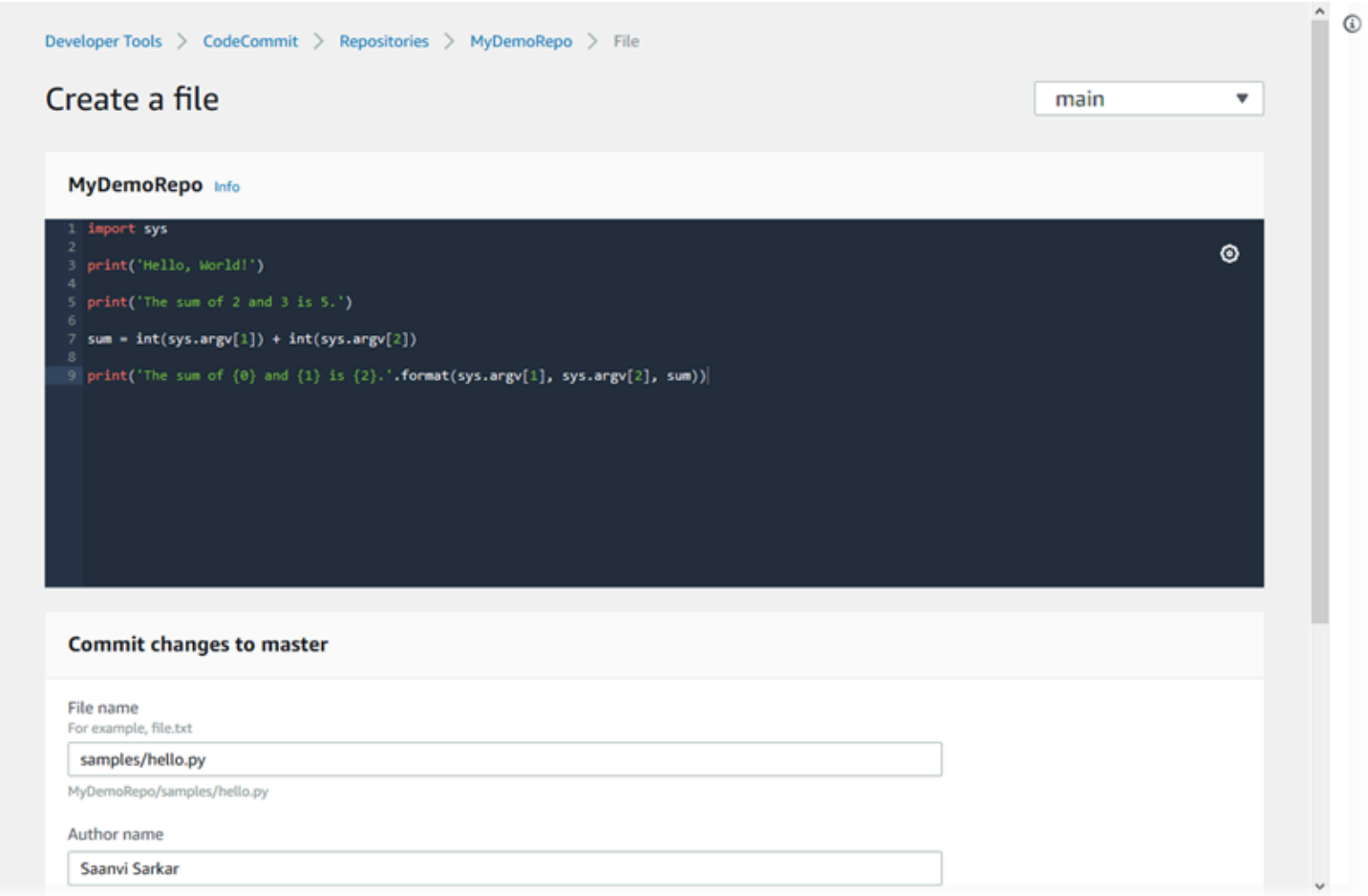
```
{
    "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

删除 CodeCommit 存储库不会删除任何可能与其连接的本地存储库。

使用 AWS CodeCommit 存储库中的文件

在 CodeCommit 中，文件是一种受版本控制的自包含式信息片段，可向您和其他用户提供存储文件的存储库和分支。您可以使用目录结构整理您的存储库文件，就如同在您的计算机上一样。与您的计算机不同，CodeCommit 会自动跟踪对文件所做的每个更改。您可以比较文件的版本，并将文件的不同版本存储在不同的存储库分支中。

要在存储库中添加或编辑文件，您可以使用 Git 客户端。您也可以使用 CodeCommit 控制台、AWS CLI 或 CodeCommit API。



有关使用 CodeCommit 中的存储库的其他方面的信息，请参阅 使用存储库、使用拉取请求、使用分支、使用提交 和 使用用户首选项。

主题

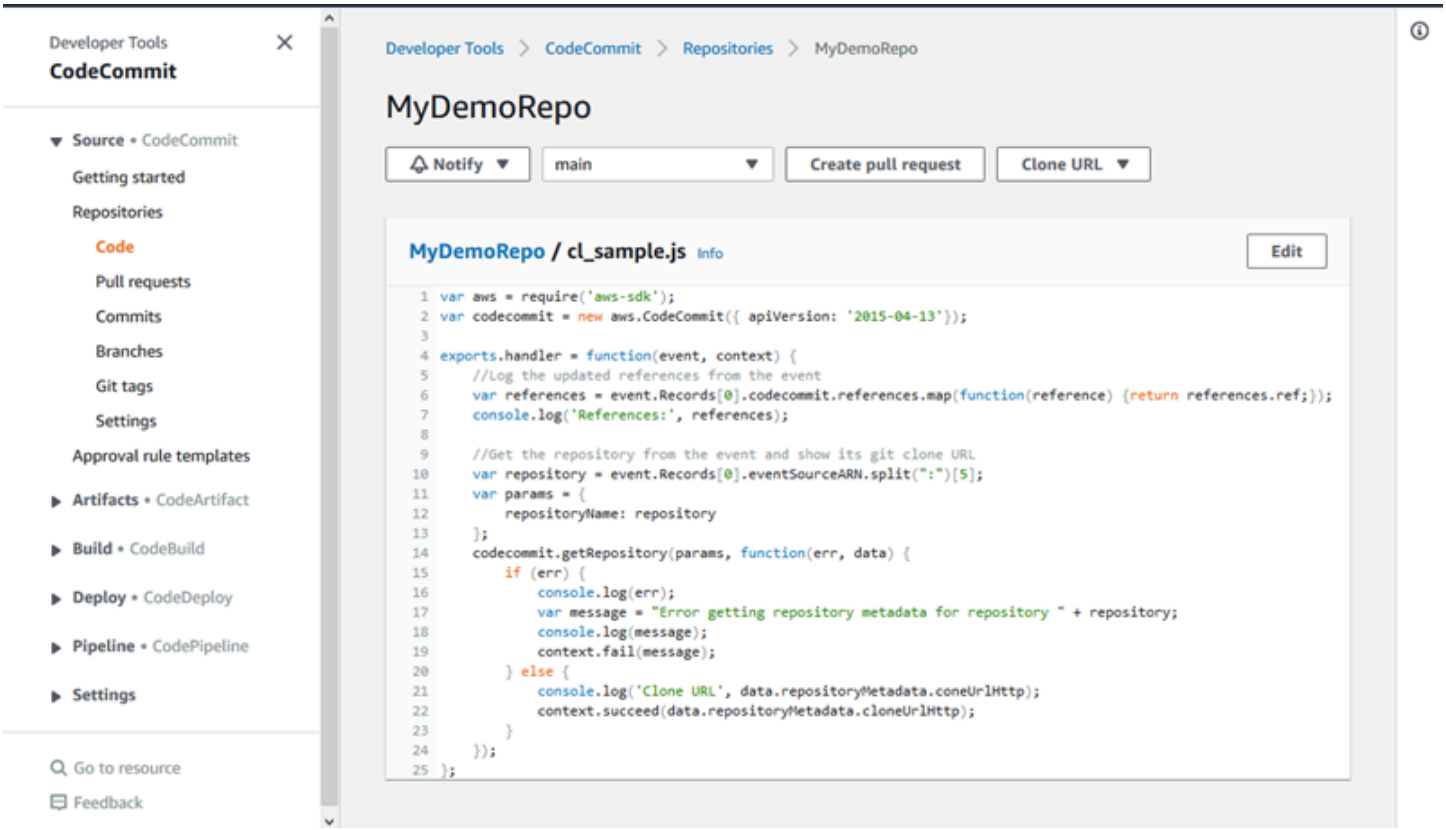
- 浏览 AWS CodeCommit 存储库中的文件
- 在 AWS CodeCommit 存储库中创建或添加文件

- 编辑 AWS CodeCommit 存储库中文件的内容

浏览 AWS CodeCommit 存储库中的文件

连接到 CodeCommit 存储库后，您可以将其克隆到本地存储库或使用 CodeCommit 控制台浏览其内容。本主题介绍如何使用 CodeCommit 控制台浏览 CodeCommit 存储库的内容。

> **Note**
>
> 对于有效的 CodeCommit 用户，从 CodeCommit 控制台浏览代码不会产生费用。有关何时可能产生费用的信息，请参阅定价。



浏览 CodeCommit 存储库

您可以使用 CodeCommit 控制台查看存储库中包含的文件或快速读取文件的内容。

浏览存储库的内容

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库页面上，从存储库列表中选择要浏览的存储库。
3. 在代码视图中，浏览存储库的默认分支的内容。

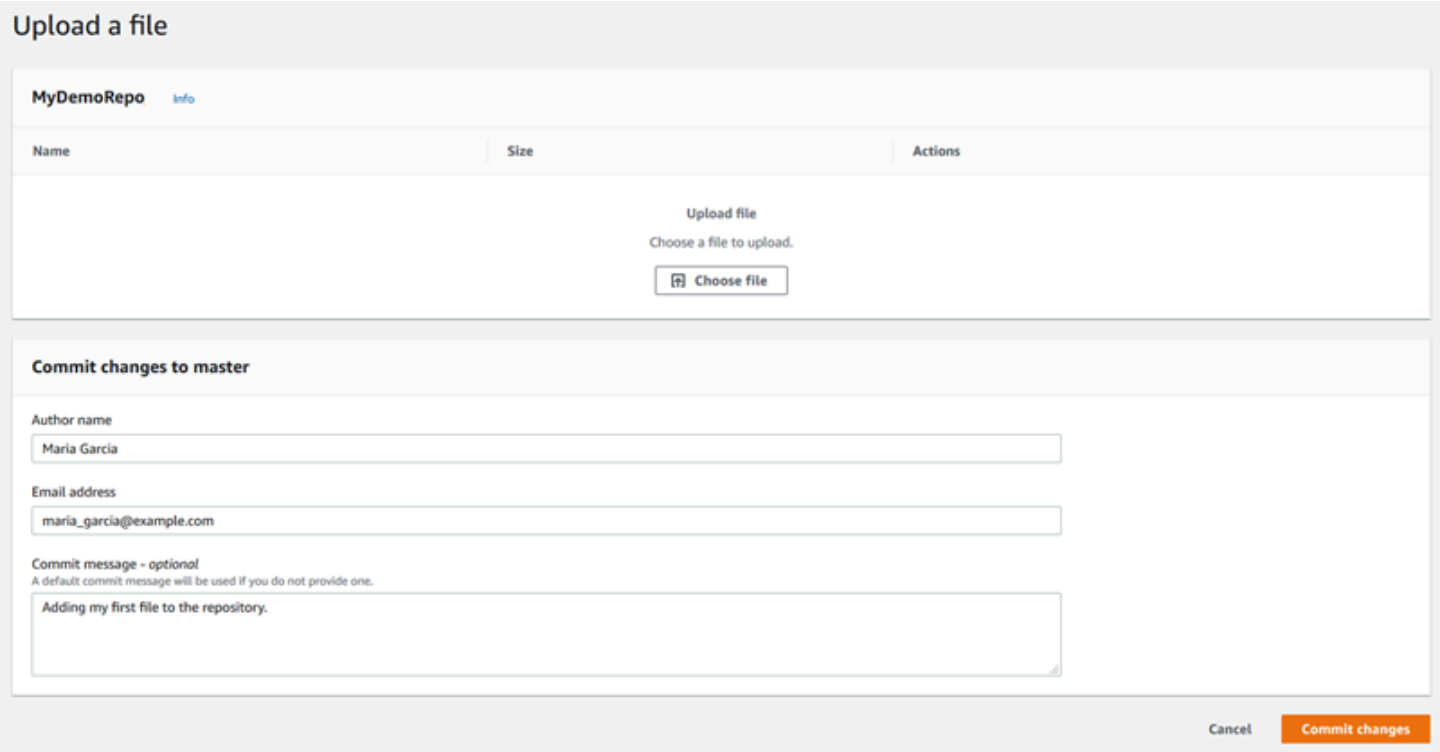
   要将视图更改到不同的分支或标签，请选择视图选择器按钮。从下拉列表中选择分支或标签名称，或者在筛选器框中输入分支或标签名称，然后从列表中选择该名称。
4. 请执行下列操作之一：

   - 要查看某个目录的内容，请从列表中选择该目录。您可以从导航列表中选择任何目录以返回到该目录的视图。也可以使用目录列表顶部的向上箭头。
   - 要查看某个文件的内容，请从列表中选择该文件。如果文件超出提交对象限制，则无法在控制台中显示，只能在本地存储库中查看。有关更多信息，请参阅配额。要退出文件视图，请从代码导航栏选择要查看的目录。

> **Note**
>
> 并非所有的二进制文件都可以在控制台中查看。如果您选择某个可能支持查看的二进制文件，将显示一条警告消息，要求您确认要显示其内容。要查看该文件，请选择 Show file contents。如果您不想查看该文件，请从代码导航栏选择要查看的目录。
> 如果您选择 markdown 文件 (.md)，请使用 Rendered Markdown 和 Markdown Source 按钮在呈现和语法视图间切换。有关更多信息，请参阅在控制台中使用 Markdown。

在 AWS CodeCommit 存储库中创建或添加文件

您可以使用 CodeCommit 控制台、AWS CLI 或 Git 客户端将文件添加到存储库。可以从您的本地计算机将文件上传到存储库，或者可以使用控制台中的代码编辑器创建文件。编辑器是一种快速简便的方法，可将单个文件 (例如 readme.md 文件) 添加到存储库中的分支。



主题

- 创建或上传文件（控制台）
- 添加文件 (AWS CLI)
- 添加文件 (Git)

创建或上传文件（控制台）

您可以使用 CodeCommit 控制台创建文件，并将其添加到 CodeCommit 存储库中的分支。在创建文件过程中，可以提供您的用户名和电子邮件地址。您还可以添加提交消息，以便其他用户了解谁添加了文件以及原因。您还可以直接从您的本地计算机将文件上传到存储库中的分支。

向存储库添加文件

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要将文件添加到其中的存储库。
3. 在代码视图中，选择要将文件添加到的分支。默认情况下，在您打开代码视图时，将显示默认分支的内容。

要将视图更改到不同的分支，请选择视图选择器按钮。从下拉列表中选择分支名称，或者在筛选器框中输入分支名称，然后从列表中选择该名称。

4. 选择添加文件，然后选择下列选项之一：
   - 要使用代码编辑器创建一个文件的内容并将该文件添加到存储库，请选择创建文件。
   - 要将文件从本地计算机上传到存储库，请选择上传文件。
5. 向其他用户提供有关谁将此文件添加到存储库以及添加原因的信息。
   - 在作者姓名中，输入您的姓名。此姓名同时用作提交信息中的作者姓名和提交者姓名。CodeCommit 默认使用您的 IAM 用户名作为作者姓名，或者从您的控制台登录名派生。
   - 在电子邮件地址中，输入一个电子邮件地址，以便其他存储库用户可以就此更改与您联系。
   - 在提交消息中，输入简要描述。这是可选的，但强烈建议您这样做。否则，将使用默认的提交消息。
6. 请执行下列操作之一：
   - 如果正在上传文件，请从您的本地计算机选择文件。
   - 如果正在创建文件，请输入您要在代码编辑器中添加的内容，并提供文件的名称。
7. 选择提交更改。

添加文件 (AWS CLI)

您可以使用 AWS CLI 和 put-file 命令在 CodeCommit 存储库中添加文件。您还可以使用 put-file 命令为该文件添加目录或路径结构。

> **Note**
>
> 要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

向存储库添加文件

1. 在本地计算机上，创建要添加到 CodeCommit 存储库中的文件。
2. 在终端或命令行中，运行 put-file 命令，并指定：
   - 您要将文件添加到的存储库。

- 您要将文件添加到的分支。
- 该分支的最近提交的完整提交 ID，也称为最前端提交或 HEAD 提交。
- 文件的本地位置。用于此位置的语法取决于您的本地操作系统。
- 您要添加的文件的名称，包括更新的文件在存储库中的存储路径 (如果有)。
- 您希望与此文件关联的用户名和电子邮件。
- 一条提交消息，说明您为什么添加此文件。

用户名、电子邮件地址和提交消息是可选的，但可帮助其他用户知道谁执行的更改以及原因。如果您没有提供用户名，CodeCommit 默认使用您的 IAM 用户名或控制台登录名的派生名称作为作者名。

例如，要将 *ExampleSolution.py* 文件添加到 *MyDemoRepo* 存储库中的 *feature-randomizationfeature* 分支，并且其最新提交的 ID 为 *4c925148EXAMPLE*，请执行以下操作：

```
aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
 García" --email "maría_garcía@example.com" --commit-message "I added a third
 randomization routine."
```

> **Note**
>
> 当您添加二进制文件时，请确保使用 `fileb://` 指定文件的本地位置。

如果成功，该命令返回类似以下内容的输出：

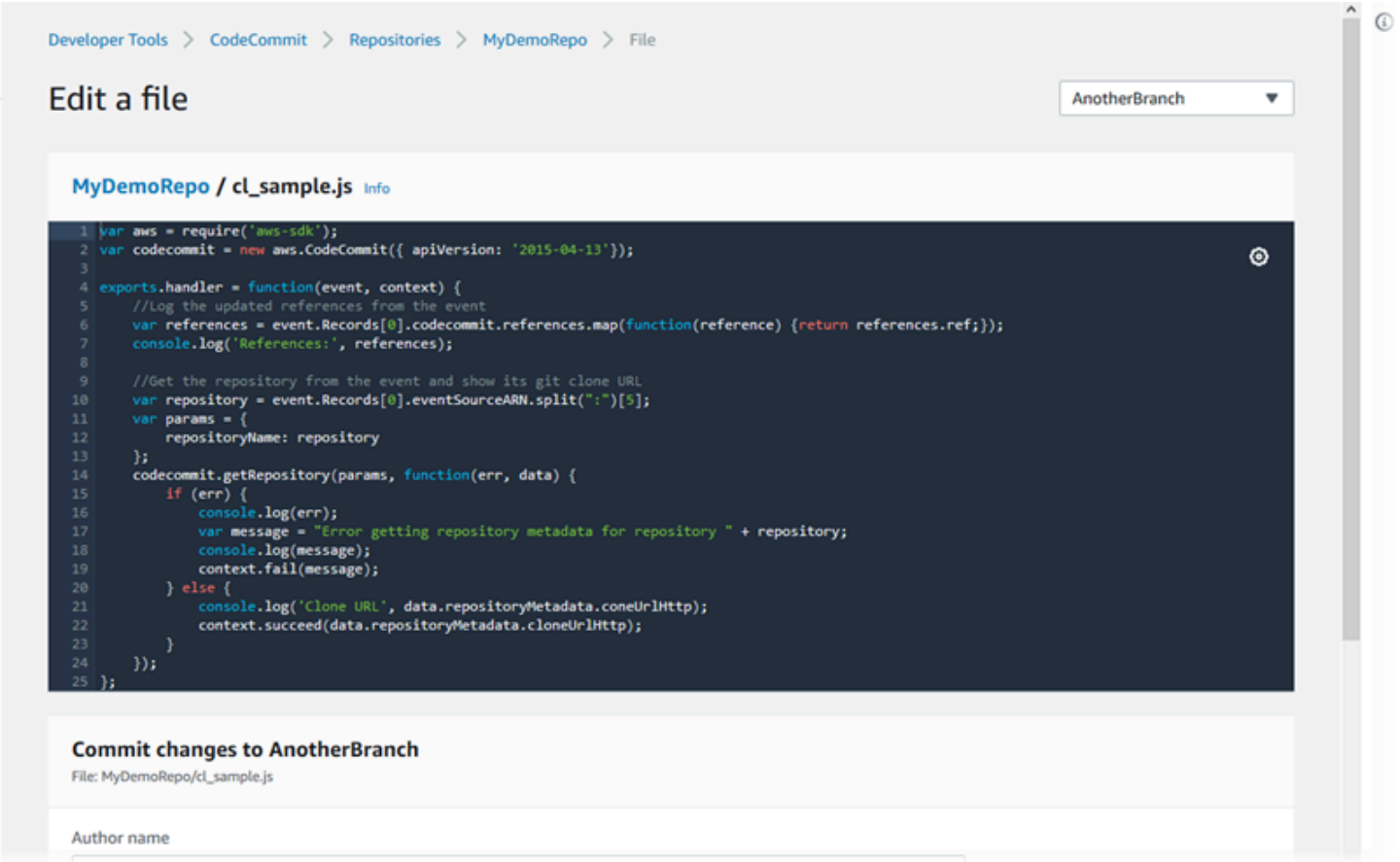
```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

添加文件 (Git)

您可以在本地存储库中添加文件，并将您的更改推送到 CodeCommit 存储库。有关更多信息，请参阅Git 和 AWS CodeCommit 入门。

编辑 AWS CodeCommit 存储库中文件的内容

您可以使用 CodeCommit 控制台、AWS CLI 或 Git 客户端编辑 CodeCommit 存储库中文件的内容。



主题

- 编辑文件（控制台）
- 编辑或删除文件 (AWS CLI)
- 编辑文件 (Git)

编辑文件（控制台）

您可以使用 CodeCommit 控制台编辑已添加到 CodeCommit 存储库中分支的文件。在编辑文件过程中，可以提供您的用户名和电子邮件地址。您还可以添加提交消息，以便其他用户了解谁执行了更改以及原因。

编辑存储库中的文件

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要编辑其中的文件的存储库。
3. 在代码视图中，选择要编辑文件的分支。默认情况下，在您打开代码视图时，将显示默认分支的内容。

   要将视图更改到不同的分支，请选择视图选择器按钮。从下拉列表中选择分支名称，或者在筛选器框中输入分支名称，然后从列表中选择该名称。
4. 导航分支内容并选择要编辑的文件。在文件视图中，选择编辑。

   > **Note**
   >
   > 如果您选择二进制文件，将显示一条警告消息，要求您确认要显示其内容。您不应使用 CodeCommit 控制台编辑二进制文件。
5. 编辑该文件，并向其他用户提供有关谁执行了此更改以及原因的信息。
   - 在作者姓名中，输入您的姓名。此姓名同时用作提交信息中的作者姓名和提交者姓名。CodeCommit 默认使用您的 IAM 用户名作为作者姓名，或者从您的控制台登录名派生。
   - 在电子邮件地址中，输入一个电子邮件地址，以便其他存储库用户可以就此更改与您联系。
   - 在提交消息中，输入您的更改的简要描述。
6. 选择 Commit changes (提交更改) 以保存对文件所做的更改并将更改提交到存储库。

编辑或删除文件 (AWS CLI)

您可以使用 AWS CLI 和 put-file 命令对 CodeCommit 存储库中的文件进行更改。如果您希望将更改的文件存储在一个不同于原始位置的位置，还可以使用 put-file 命令为更改的文件添加目录或路径结构。如果您要完全删除文件，您可以使用 delete-file 命令。

> **Note**
>
> 要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

编辑存储库中的文件

1. 使用文件的本地副本，执行您要添加到 CodeCommit 存储库的更改。
2. 在终端或命令行中，运行 put-file 命令，并指定：

   - 您要将编辑后的文件添加到的存储库。
   - 您要将编辑后的文件添加到的分支。
   - 该分支的最近提交的完整提交 ID，也称为最前端提交或 HEAD 提交。
   - 文件的本地位置。
   - 您要添加的更新的文件的名称，包括更新的文件在存储库中的存储路径 (如果有)。
   - 您希望与此文件更改关联的用户名和电子邮件。
   - 一条提交消息，说明您所做的更改。

   用户名、电子邮件地址和提交消息是可选的，但可帮助其他用户知道谁执行的更改以及原因。如果您未提供用户名，CodeCommit 默认使用您的 IAM 用户名，或者从您的控制台登录名派生。

   例如，要将对 *ExampleSolution.py* 文件所做的编辑添加到 *MyDemoRepo* 存储库中的 *feature-randomizationfeature* 分支，并且其最新提交的 ID 为 *4c925148EXAMPLE*，请执行以下操作：

   ```
   aws codecommit put-file --repository-name MyDemoRepo --branch-name feature-
   randomizationfeature --file-content file://MyDirectory/ExampleSolution.py --file-
   path /solutions/ExampleSolution.py --parent-commit-id 4c925148EXAMPLE --name "María
    García" --email "maría_garcía@example.com" --commit-message "I fixed the bug Mary
    found."
   ```

   > **Note**
   >
   > 如果您要添加更改后的二进制文件，请确保结合使用 `--file-content` 与表示法 **fileb**://*MyDirectory*/*MyFile*.raw。

如果成功，该命令返回类似以下内容的输出：

```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

要删除文件，请使用 delete-file 命令。例如，要在 *MyDemoRepo* 存储库中名为 *main*、最新提交 ID 为 *c5709475EXAMPLE* 的分支中删除名为 *README.md* 的文件，请运行以下命令：

```
aws codecommit delete-file --repository-name MyDemoRepo --branch-name main --file-
path README.md --parent-commit-id c5709475EXAMPLE
```

如果成功，该命令返回类似以下内容的输出：

```
{
  "blobId":"559b44fEXAMPLE",
  "commitId":"353cf655EXAMPLE",
  "filePath":"README.md",
  "treeId":"6bc824cEXAMPLE"
}
```

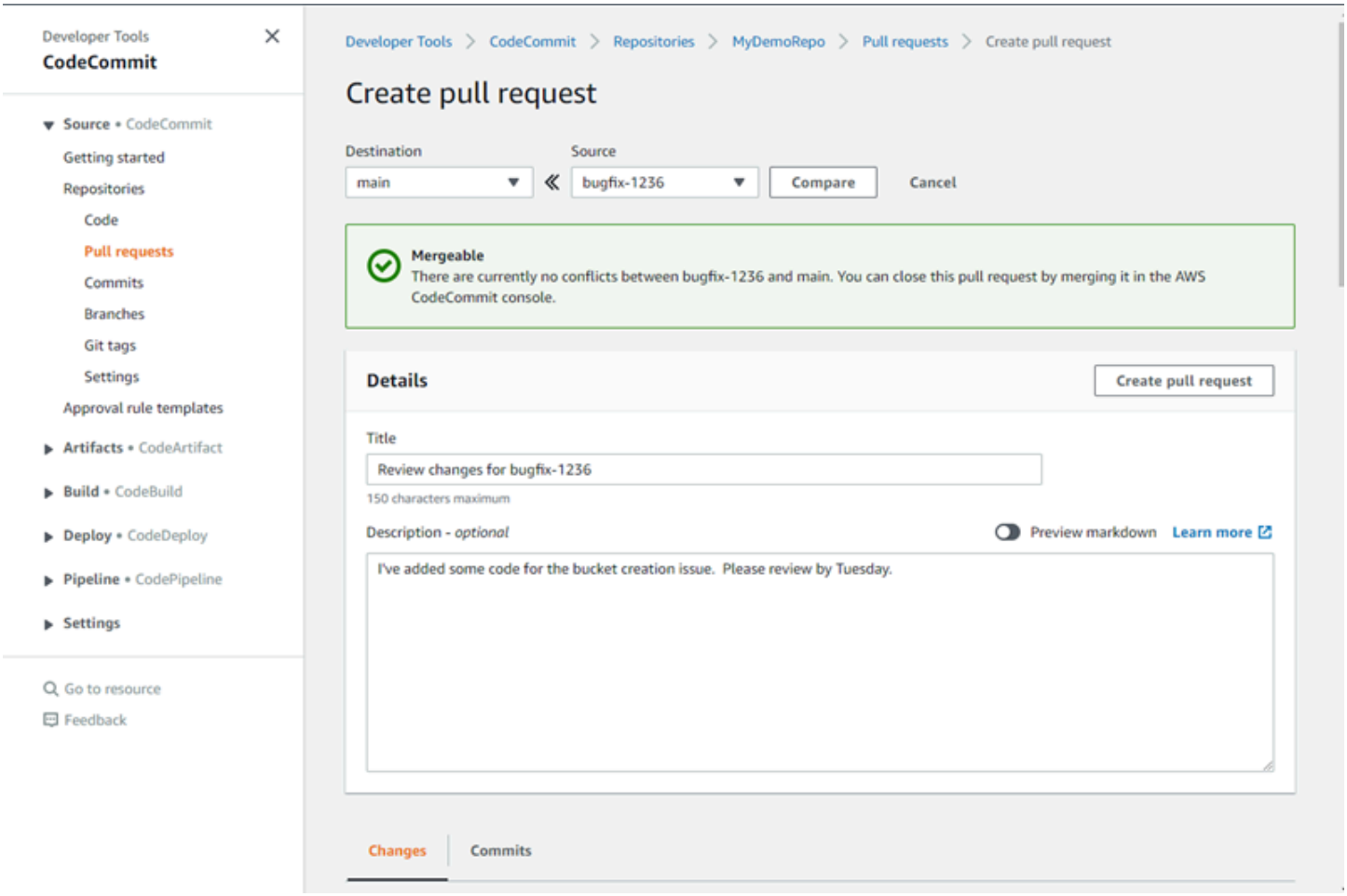
编辑文件 (Git)

您可以在本地存储库中编辑文件，并将您的更改推送到 CodeCommit 存储库。有关更多信息，请参阅Git 和 AWS CodeCommit 入门。

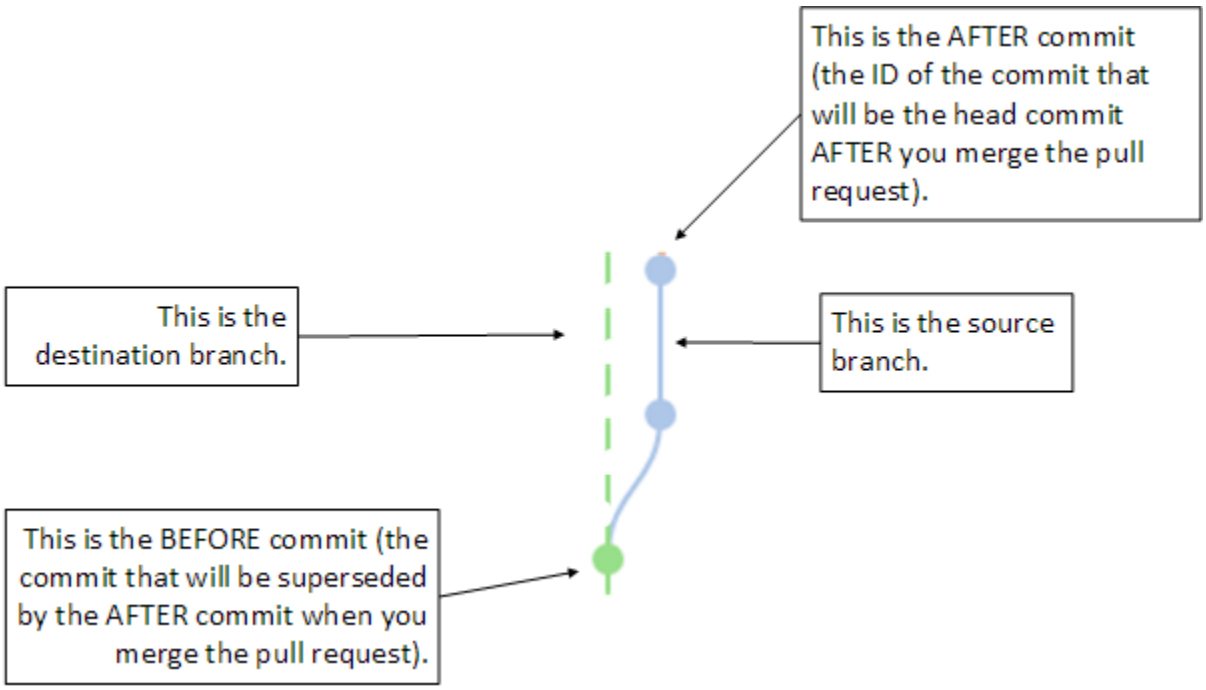
使用 AWS CodeCommit 存储库中的拉取请求

拉取请求是您和其他存储库用户用于审查、评论和合并分支间的代码更改的主要方式。您可以使用拉取请求与他人协作审查代码更改中次要的更改或修复、主要功能增加或已发布软件的新版本。以下是一个适用于拉取请求的可能的工作流程：

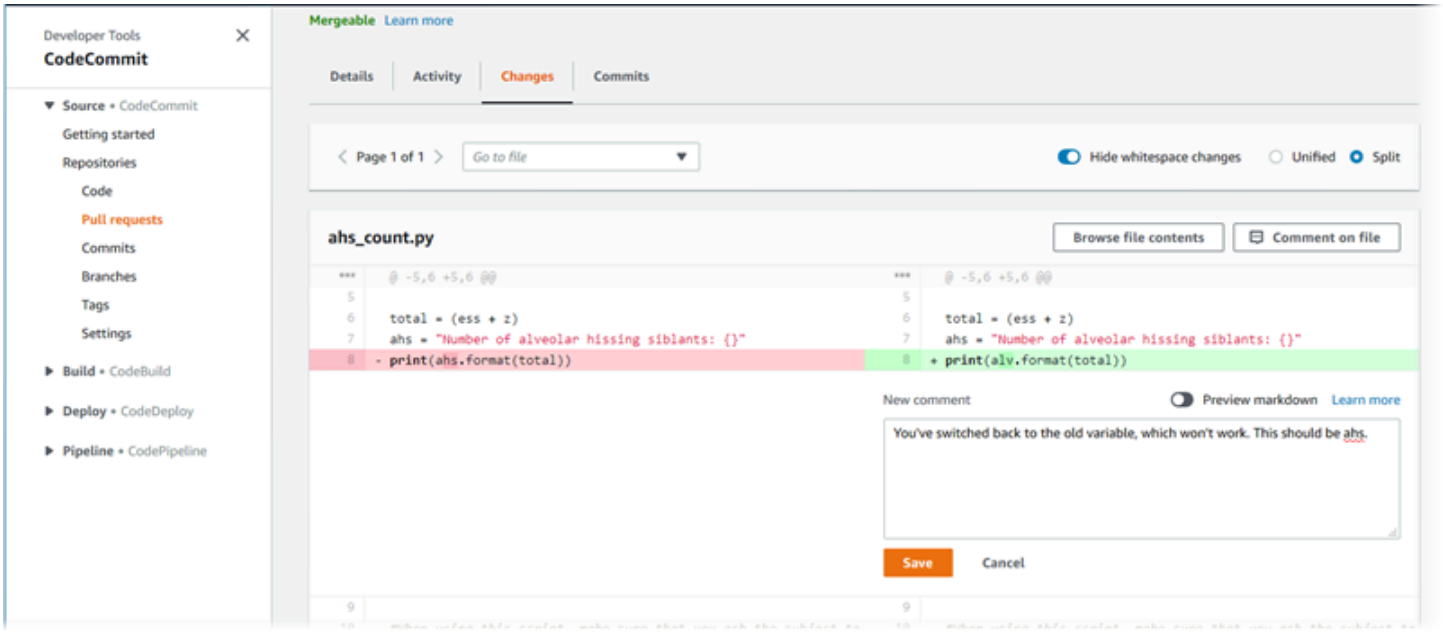
Li Juan 是一位开发人员，使用名为 MyDemoRepo 的存储库工作。她需要在即将推出的产品版本中开发一项新功能。为使她的工作与生产就绪代码相独立，她独立于默认分支创建了一个分支，将其命名为 *feature-randomizationfeature*。她编写代码、进行提交并将新功能代码推送到此分支中。在将更改合并到默认分支中之前，她希望其他存储库用户审查代码质量。为执行此操作，她创建了一个拉取请求。拉取请求包含她的工作分支与她打算合并更改的代码分支 (在本示例中为默认分支) 之间的比较。她还可以创建审批规则，要求指定数量的用户对其拉取请求进行审批。她甚至可以指定用户审批池。其他用户会审查她的代码和更改，同时添加评论和建议。她可能会多次根据评论更改代码并更新其工作分支。每当她将更改推送到 CodeCommit 中的该分支时，这些更改都将并入拉取请求。当拉取请求处于打开状态时，她还可能并入在预期目标分支中所做的更改，以使用户可以确保他们在上下文中审查所有提议的更改。在她及其审核人感到满意并且满足了审批规则条件（如果有）时，她或一位审核人可以合并代码并关闭拉取请求。



拉取请求需要两个分支：源分支 (包含要审查的代码) 和目标分支 (被审查的代码将合并到其中)。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。选择的合并策略会影响在 CodeCommit 控制台中如何在源和目标分支之间合并提交的详细信息。有关 CodeCommit 中的合并策略的更多信息，请参阅[合并拉取请求（控制台）](#)。



拉取请求显示创建拉取请求时源分支的提示与目标分支中的最新提交之间的差异，以便于用户查看并评论更改。通过将更改提交和推送到源分支，可以更新拉取请求以响应评论。



当代码经过审核并满足审批规则要求（如果有）时，您可以通过以下几种方式关闭拉取请求：

- 本地合并分支并推送更改。如果使用快进合并策略，这会自动关闭请求，并且不会产生任何合并冲突。

- 使用 AWS CodeCommit 控制台关闭拉取请求而不进行合并、解决合并冲突，如果没有冲突，则使用可用的合并策略关闭及合并分支。
- 请使用 AWS CLI。

在创建拉取请求之前，请：

- 确保您已将要审核的代码更改提交并推送至分支（源分支）。
- 为存储库设置通知，这样其他用户可以收到有关拉取请求及其更改的通知。(虽然此步骤是可选的，但我们建议您这样做。)
- 创建审批规则模板并将其与存储库关联，以便为拉取请求自动创建审批规则，从而帮助确保代码质量。有关更多信息，请参阅使用审批规则模板。

如果已为 Amazon Web Services 账户中的存储库用户设置 IAM 用户，则拉取请求会更有效。可以更轻松地确定哪个用户发布了什么注释。另一个优点是 IAM 用户可以使用 Git 凭证访问存储库。有关更多信息，请参阅步骤 1：的初始配置 CodeCommit。您可以对其他类型的用户（包括联合访问用户）使用拉取请求。

有关使用 CodeCommit 中的存储库的其他方面的信息，请参阅使用存储库、使用审批规则模板、使用文件、使用提交、使用分支、和使用用户首选项。

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。在将这些更改提交并推送到存储库之后，您可以创建拉取请求将该分支 (源分支) 的内容与要在拉取请求关闭后将更改合并到的分支 (目的分支) 进行比较。

您可以使用 AWS CodeCommit 控制台或 AWS CLI 为存储库创建拉取请求。

主题

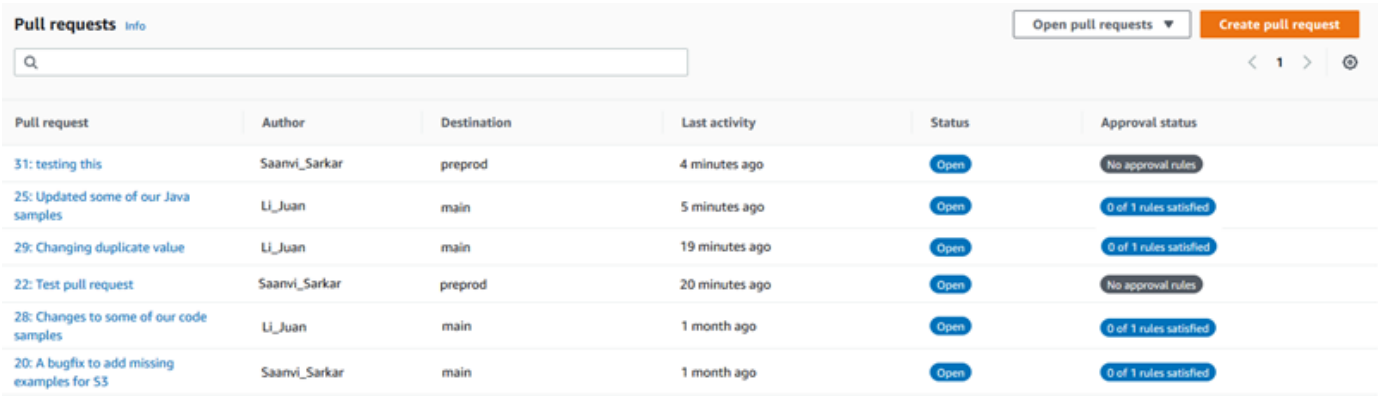
创建拉取请求（控制台）

您可以使用 CodeCommit 控制台在 CodeCommit 存储库中创建拉取请求。如果您的存储库配置了通知，则订阅用户将在您创建拉取请求时收到电子邮件。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要在其中创建拉取请求的存储库的名称。
3. 在导航窗格中，选择拉取请求。

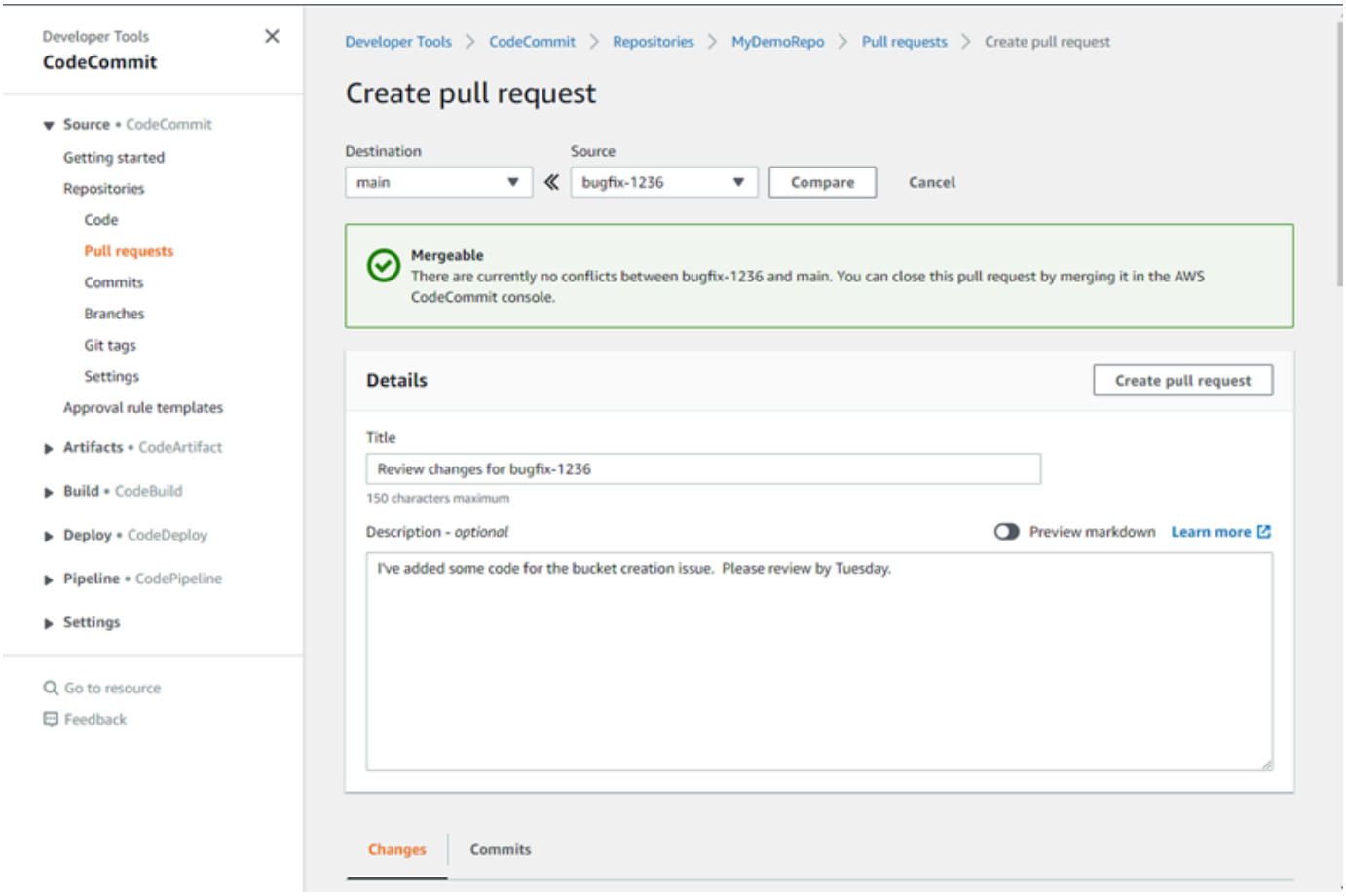
> **Tip**
>
> 您也可以从分支和代码创建拉取请求。

4. 选择创建拉取请求。



5. 在 Create pull request 中，在 Source 中选择包含要审核的更改的分支。

6. 在 Destination (目标) 中，选择要在拉取请求关闭后将代码更改合并到的分支。
7. 选择 Compare。将对两个分支进行比较，并显示它们之间的差异。还会执行分析以确定在拉取请求关闭时两个分支能否自动合并。
8. 审核比较详细信息和更改以确保拉取请求包含您要审核的更改和提交。如果不包含，请调整您的源分支和目标分支选择，然后再次选择 Compare。
9. 在您对拉取请求的比较结果感到满意以后，在标题中为此审核输入一个简短的描述性标题。此标题显示在存储库的拉取请求列表中。
10. （可选）在描述中，输入有关此审核的详细信息以及对审核者有用的任何其他信息。
11. 选择 Create（创建）。



拉取请求将显示在存储库的拉取请求列表中。如果您配置了通知，则该 Amazon SNS 主题的订阅者会收到电子邮件，通知他们新创建了拉取请求。

创建拉取请求 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

使用 AWS CLI 在 CodeCommit 存储库中创建拉取请求

1. 运行 create-pull-request 命令，并指定：

   - 拉取请求的名称（使用 --title 选项）。
   - 拉取请求的描述（使用 --description 选项）。
   - create-pull-request 命令的目标列表，包括：
     - 将在其中创建拉取请求的 CodeCommit 存储库的名称(使用 repositoryName 属性)。
     - 包含您要审核的代码更改的分支的名称，也称为源分支（使用 sourceReference 属性）。
     - （可选）如果您不想合并到默认分支，还需要提供要将代码更改合并到的分支（也称为目标分支）的名称（使用 destinationReference 属性）。
   - 客户端生成的唯一等幂令牌（使用 --client-request-token 选项）。

   此示例创建一个名为 *Pronunciation difficulty analyzer* 的拉取请求，其描述为 *Please review these changes by Tuesday*，并以 *jane-branch* 源分支为目标。拉取请求将合并到名为 MyDemoRepo 的 CodeCommit 存储库的默认分支 *main* 中：

```
aws codecommit create-pull-request --title "Pronunciation difficulty analyzer"
 --description "Please review these changes by Tuesday" --client-request-token
 123Example --targets repositoryName=MyDemoRepo,sourceReference=jane-branch
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd3d22fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
```

```
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::111111111111:user/Jane_Doe",
        "description": "Please review these changes by Tuesday",
        "title": "Pronunciation difficulty analyzer",
        "pullRequestTargets": [
            {
                "destinationCommit": "5d036259EXAMPLE",
                "destinationReference": "refs/heads/main",
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "317f8570EXAMPLE",
                "sourceReference": "refs/heads/jane-branch",
                "mergeMetadata": {
                    "isMerged": false
                }
            }
        ],
        "lastActivityDate": 1508962823.285,
        "pullRequestId": "42",
        "clientRequestToken": "123Example",
        "pullRequestStatus": "OPEN",
        "creationDate": 1508962823.285
    }
}
```

为拉取请求创建审批规则

为拉取请求创建审批规则，要求用户在将代码合并到目标分支之前对拉取请求进行审批，从而有助于确保代码的质量。您可以指定必须审批拉取请求的用户数量。您还可以为规则指定用户审批池。如果这样做，则只有这些用户的审批计入规则的所需审批数量。

> **Note**
>
> 您还可以创建审批规则模板，帮助您跨存储库自动创建要应用于每个拉取请求的审批规则。有关更多信息，请参阅使用审批规则模板。

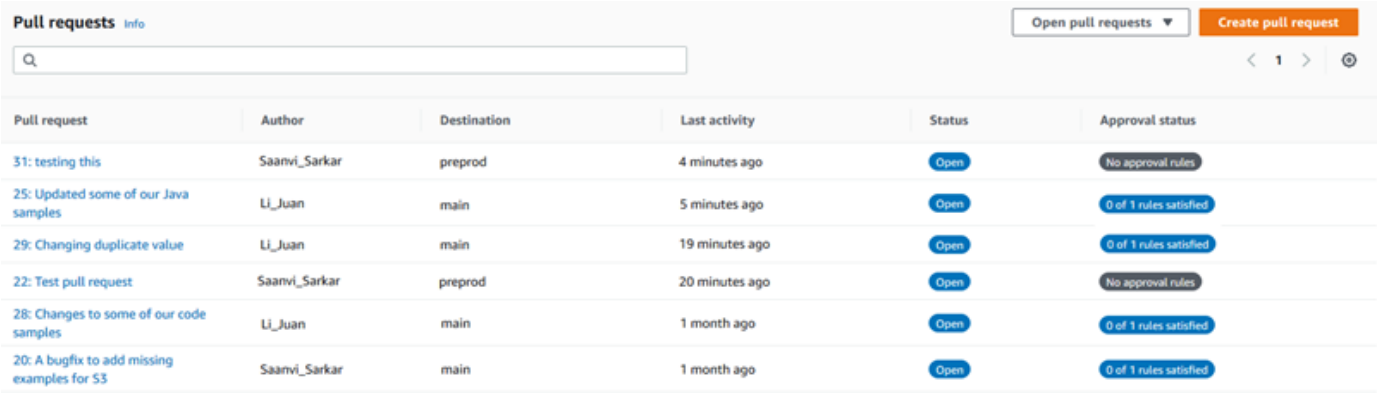
您可以使用 AWS CodeCommit 控制台或 AWS CLI 为存储库创建审批规则。

主题

为拉取请求创建审批规则（控制台）

您可以使用 CodeCommit 控制台为 CodeCommit 存储库中的拉取请求创建审批规则。

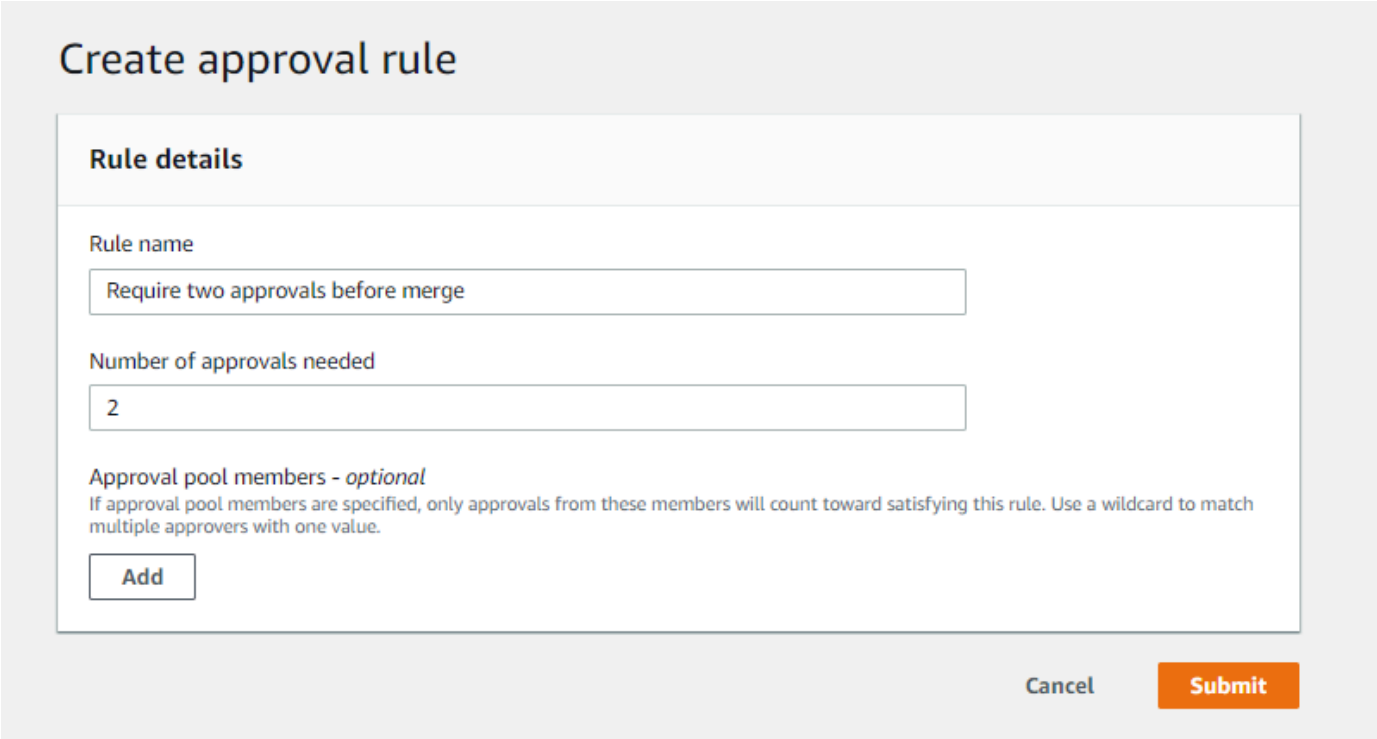
1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中为拉取请求创建审批规则的存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 从列表中选择要为其创建审批规则的拉取请求。您只能为处于打开状态的拉取请求创建审批规则。



5. 在拉取请求中，选择 Approvals (审批)，然后选择 Create approval rule (创建审批规则)。
6. 在 Rule name (规则名称) 中，为规则指定一个描述性名称，以便您知道其用途。例如，如果您希望拉取请求在合并之前，必须由两人对其进行审批，那么可以将该规则命名为 **`Require two approvals before merge`**。

> **Note**
>
> 审批规则在创建之后，其名称无法更改。

在 Number of approvals needed (需要的审批数量) 中，输入所需的数量。默认值为 1。



7. （可选）如果您希望必须由特定用户组对拉取请求进行审批，那么可以在 Approval rule members (审批规则成员) 中，选择 Add (添加)。在 Approver type (审批人类型) 中，选择以下选项之一：

   - IAM 用户名或代入的角色：此选项会预先填入您用于登录的账户的 AWS 账户 ID，并且只需要一个名称。它可以用于 IAM 用户，以及名称与所提供名称相匹配的联合访问用户。这是一个非常强大的选项，提供了极大的灵活性。例如，如果您使用 Amazon Web Services 账户 123456789012 登录并选择了此选项，而且指定了 **`Mary_Major`**，那么以下所有用户都将计为来自该用户的审批：

     - 账户中的 IAM 用户 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中标识为 Mary_Major 的联合用户 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非包含通配符 (`*Mary_Major`)，否则此选项无法识别代入 **`CodeCommitReview`** 角色且角色会话名称为 Mary_Major (`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`) 的某人的活动会话。您还可以显式指定角色名称 (`CodeCommitReview/Mary_Major`)。

   - 完全限定的 ARN：此选项允许您指定 IAM 用户或角色的完全限定 Amazon 资源名称 (ARN)。此选项还支持由其他 AWS 服务（如 AWS Lambda 和 AWS CodeBuild）使用的代入角色。对于代入的角色，ARN 格式应为 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`RoleName`*（适

用于角色）和 arn:aws:sts::*AccountID*:assumed-role/*FunctionName*（适用于函数）。

如果选择 IAM 用户名或代入的角色作为审批人类型，那么请在值中输入 IAM 用户或角色的名称，或者输入用户或角色的完全限定 ARN。再次选择 Add (添加) 可添加多个用户或角色，直到您已添加了其审批计入所需审批数量的所有用户或角色。

这两种审批人类型都允许在其值中使用通配符 (*)。例如，如果选择 IAM 用户名或代入的角色选项，并且指定 **CodeCommitReview/***，那么代入 **CodeCommitReview** 角色的所有用户都将计入审批池中。他们各自的角色会话名称将计入所需的审批人数量。按照此方法，Mary_Major 和 Li_Juan 在登录并代入 CodeCommitReview 角色时，都计为审批。有关 IAM ARN、通配符和格式的更多信息，请参阅 IAM 标识符。

> **Note**
>
> 审批规则不支持跨账户审批。

8. 完成审批规则的配置之后，选择 Submit (提交)。

为拉取请求创建审批规则 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

为 CodeCommit 存储库中的拉取请求创建审批规则

1. 运行 create-pull-request-approval-rule 命令，并指定：
   - 拉取请求的 ID（使用 --id 选项）。
   - 审批规则的名称（使用 --approval-rule-name 选项）。
   - 审批规则的内容（使用 --approval-rule-content 选项）。

   创建审批规则时，可以按照以下两种方式之一指定审批池中的审批人：

   - CodeCommitApprovers：此选项仅需要 Amazon Web Services 账户和资源。它可以用于 IAM 用户，以及名称与所提供资源名称相匹配的联合访问用户。这是一个非常强大的选项，提供了极大的灵活性。例如，如果指定 Amazon Web Services 账户 123456789012 和 **Mary_Major**，那么以下所有用户都将计为来自该用户的审批：

- 账户中的 IAM 用户 (arn:aws:iam::123456789012:user/Mary_Major)
- 在 IAM 中标识为 Mary_Major 的联合用户 (arn:aws:sts::123456789012:federated-user/Mary_Major)

除非包含通配符 (*Mary_Major)，否则此选项无法识别代入 **CodeCommitReview** 角色且角色会话名称为 Mary_Major (arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major) 的某人的活动会话。

- 完全限定的 ARN：此选项允许您指定 IAM 用户或角色的完全限定 Amazon 资源名称 (ARN)。

有关 IAM ARN、通配符和格式的更多信息，请参阅 IAM 标识符。

以下示例为 ID 为 27 的拉取请求创建名为 Require two approved approvers 的审批规则。该规则指定审批池中需要两个审批。池中包括 123456789012 Amazon Web Services 账户中访问 CodeCommit 并代入 **CodeCommitReview** 角色的所有用户，还包括同一 Amazon Web Services 账户中的 IAM 用户或名为 Nikhil_Jayashankar 的联合用户：

```
aws codecommit create-pull-request-approval-rule --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-
rule-content "{\"Version\": \"2018-11-08\",\"Statements\": [{\"Type\":
 \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "approvalRule": {
        "approvalRuleName": "Require two approved approvers",
        "lastModifiedDate": 1570752871.932,
        "ruleContentSha256": "7c44e6ebEXAMPLE",
        "creationDate": 1570752871.932,
        "approvalRuleId": "aac33506-EXAMPLE",
        "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements\":
 [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"CodeCommitApprovers:123456789012:Nikhil_Jayashankar\",
 \"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major"
    }
}
```

查看 AWS CodeCommit 存储库中的拉取请求

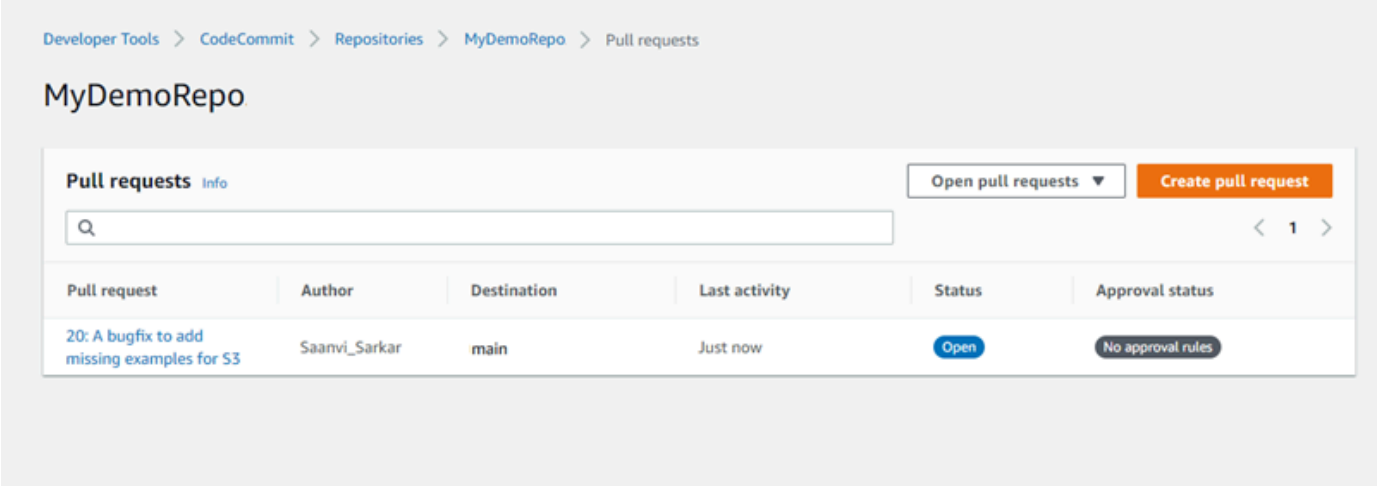
您可以使用 AWS CodeCommit 控制台或 AWS CLI 查看存储库的拉取请求。默认情况下，虽然您只会看到处于打开状态的拉取请求，但您可以更改筛选器以查看所有拉取请求、仅查看关闭的请求、仅查看您创建的拉取请求等。

主题

查看拉取请求（控制台）

您可以使用 AWS CodeCommit 控制台查看 CodeCommit 存储库中的拉取请求列表。通过更改筛选器，可以更改列表显示，使其仅显示一组特定的拉取请求。例如，您可以查看您创建的状态为 Open 的拉取请求列表，也可以选择不同的筛选器并查看您创建的状态为 Closed 的拉取请求。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中查看拉取请求的存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。



5. 要更改显示筛选器，请从可用筛选器列表中选择：
   - 已打开的拉取请求（默认）：显示状态为 Open (打开) 的所有拉取请求。
   - All pull requests (所有拉取请求)：显示所有拉取请求。
   - Closed pull requests (已关闭的拉取请求)：显示状态为 Closed (关闭) 的所有拉取请求。

- My pull requests (我的拉取请求)：显示您创建的所有拉取请求，而不考虑状态。这不会显示您注释过或参与的评论。
- My open pull requests (我的打开的拉取请求)：显示您创建的状态为 Open (打开) 的所有拉取请求。
- My closed pull requests (我的已关闭拉取请求)：显示您创建的状态为 Closed (已关闭) 的所有拉取请求。

6. 在显示列表中找到要查看的拉取请求时，将其选中。

查看拉取请求 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

按照以下步骤来使用 AWS CLI 查看 CodeCommit 存储库中的拉取请求。

1. 要查看存储库中的拉取请求列表，请运行 list-pull-requests 命令，并且指定：

   - 要查看拉取请求的 CodeCommit 存储库的名称（使用 --repository-name 选项）。
   - （可选）拉取请求的状态（使用 --pull-request-status 选项）。
   - （可选）创建拉取请求的 IAM 用户的 Amazon 资源名称 (ARN)（使用 --author-arn 选项）。
   - （可选）可用于返回批量结果的枚举令牌（使用 --next-token 选项）。
   - （可选）对每个请求返回结果数的限制（使用 --max-results 选项）。

   例如，要列出在名为 MyDemoRepo 的 CodeCommit 存储库中由 ARN 为 *arn:aws:iam::111111111111:user/Li_Juan* 的 IAM 用户创建的状态为 *CLOSED* 的拉取请求，请运行以下命令：

   ```
   aws codecommit list-pull-requests --author-arn arn:aws:iam::111111111111:user/
   Li_Juan --pull-request-status CLOSED --repository-name MyDemoRepo
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
      "nextToken": "",
      "pullRequestIds": ["2","12","16","22","23","35","30","39","47"]
   }
   ```

拉取请求 ID 按照最近活动的顺序显示。

2. 要查看某个拉取请求的详细信息，请运行带 --pull-request-id 选项的 get-pull-request 命令，并指定该拉取请求的 ID。例如，要查看 ID 为 *27* 的拉取请求的有关信息，请运行以下命令：

```
aws codecommit get-pull-request --pull-request-id 27
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "lastActivityDate": 1562619583.565,
        "pullRequestTargets": [
            {
                "sourceCommit": "ca45e279EXAMPLE",
                "sourceReference": "refs/heads/bugfix-1234",
                "mergeBase": "a99f5ddbEXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false
                },
                "destinationCommit": "2abfc6beEXAMPLE",
                "repositoryName": "MyDemoRepo"
            }
        ],
        "revisionId": "e47def21EXAMPLE",
        "title": "Quick fix for bug 1234",
        "authorArn": "arn:aws:iam::123456789012:user/Nikhil_Jayashankar",
        "clientRequestToken": "d8d7612e-EXAMPLE",
```

```
        "creationDate": 1562619583.565,
        "pullRequestId": "27",
        "pullRequestStatus": "OPEN"
    }
}
```

3. 要查看拉取请求的审批状态，请运行 get-pull-request-approval-state 命令，并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 拉取请求的修订 ID（使用 --revision-id option)）。您可以使用 get-pull-request 命令获取拉取请求的当前修订 ID。

   例如，要查看 ID 为 *8* 且修订 ID 为 *9f29d167EXAMPLE* 的拉取请求的审批状态，请运行以下命令：

   ```
   aws codecommit get-pull-request-approval-state --pull-request-id 8 --revision-
   id 9f29d167EXAMPLE
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "approvals": [
           {
               "userArn": "arn:aws:iam::123456789012:user/Mary_Major",
               "approvalState": "APPROVE"
           }
       ]
   }
   ```

4. 要查看某个拉取请求中的事件，请运行带 --pull-request-id 选项的 describe-pull-request-events 命令，并指定该拉取请求的 ID。例如，要查看 ID 为 *8* 的拉取请求的事件，请执行以下操作：

   ```
   aws codecommit describe-pull-request-events --pull-request-id 8
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "pullRequestEvents": [
           {
   ```

```
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_CREATED",
            "eventDate": 1510341779.53,
            "actor": "arn:aws:iam::111111111111:user/Zhang_Wei"
        },
        {
            "pullRequestStatusChangedEventMetadata": {
                "pullRequestStatus": "CLOSED"
            },
            "pullRequestId": "8",
            "pullRequestEventType": "PULL_REQUEST_STATUS_CHANGED",
            "eventDate": 1510341930.72,
            "actor": "arn:aws:iam::111111111111:user/Jane_Doe"
        }
    ]
}
```

5. 要查看拉取请求是否存在合并冲突，请运行 get-merge-conflicts 命令，并指定：

   - CodeCommit 存储库的名称（使用 --repository-name 选项）。
   - 要在合并评估中使用的变更源的分支、标签、HEAD 或其他完全限定的引用（使用 --source-commit-specifier 选项）。
   - 要在合并评估中使用的变更目标的分支、标签、HEAD 或其他完全限定的引用（使用 --destination-commit-specifier 选项）。
   - 要使用的合并选项（使用 --merge-option 选项）

   例如，要查看在名为 MyDemoRepo 的存储库中的名为 *my-feature-branch* 的源分支的提示和名为 *main* 的目标分支的提示之间是否存在合并冲突，请运行以下命令：

   ```
   aws codecommit get-merge-conflicts --repository-name MyDemoRepo --source-commit-
   specifier my-feature-branch --destination-commit-specifier main --merge-option
    FAST_FORWARD_MERGE
   ```

   如果成功，该命令返回类似以下内容的输出：

   ```
   {
       "destinationCommitId": "fac04518EXAMPLE",
       "mergeable": false,
       "sourceCommitId": "16d097f03EXAMPLE"
   }
   ```

审核拉取请求

您可以使用 AWS CodeCommit 控制台查看拉取请求中包含的更改。您可以向请求、文件和特定代码行添加评论。您还可以回复其他用户所做的评论。如果存储库已配置通知，您将在用户回复您的评论或用户评论拉取请求时收到电子邮件。

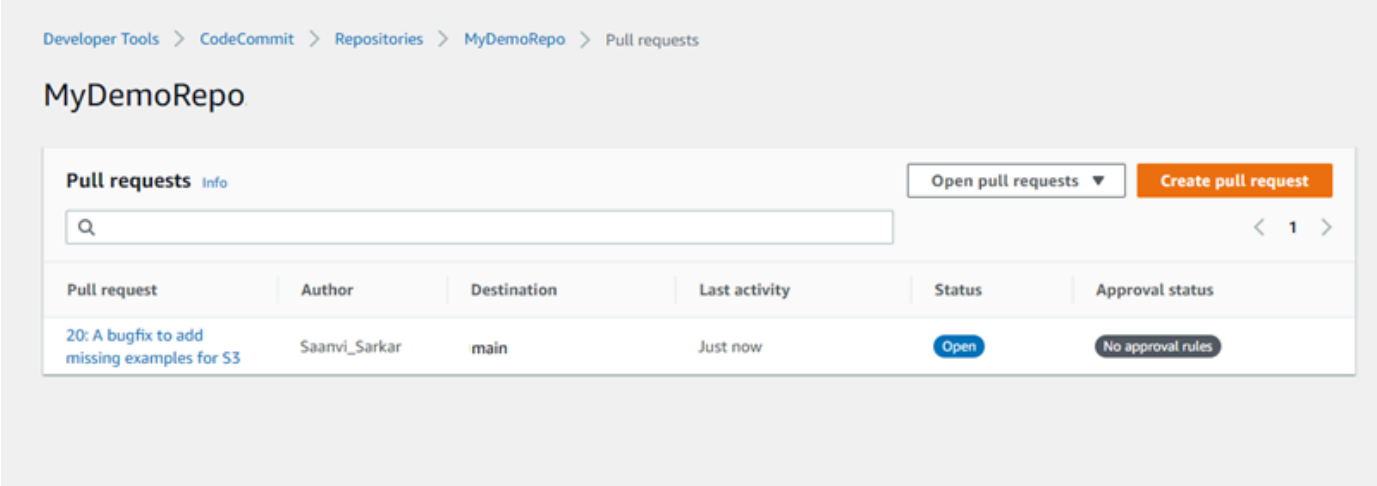
您可以使用 AWS CLI 对拉取请求发表评论和回复评论。要查看更改，必须使用 CodeCommit 控制台、git diff命令或差异工具。

主题

审核拉取请求（控制台）

您可以使用 CodeCommit 控制台查看 CodeCommit 存储库中的拉取请求。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。选择要审核的处于打开状态的拉取请求。



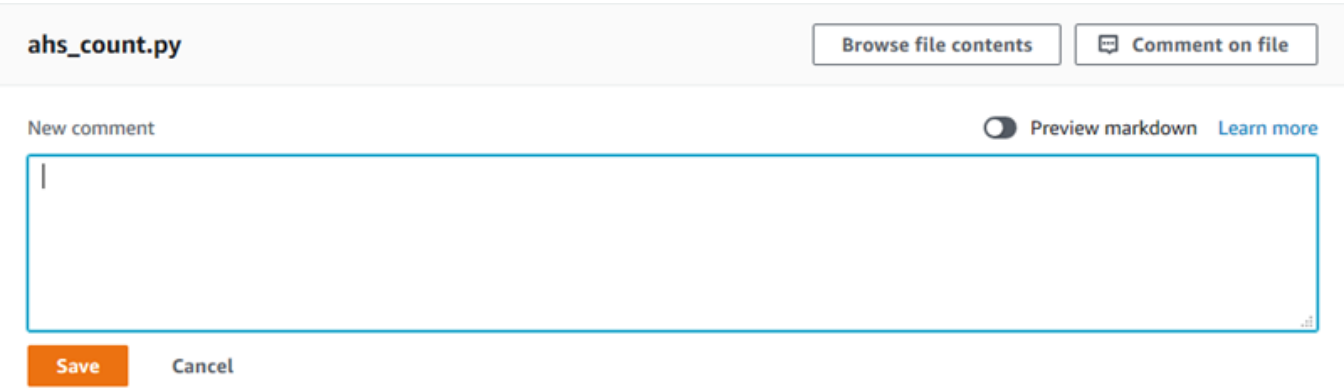
> **Note**
>
> 您可以对已关闭或已合并的拉取请求进行注释，但不能合并或重新打开该请求。

5. 在拉取请求中，选择 Changes。
6. 请执行以下操作之一：

   - 要为整个拉取请求添加一般注释，请在 Comments on changes (更改注释) 的 New comment (新建注释) 中，输入注释，然后选择 Save (保存)。您可以使用 Markdown，也可以纯文本格式输入评论。

   - 要向提交中的文件添加评论，请在 Changes 中找到该文件的名称。选择显示在文件名旁的注释图标

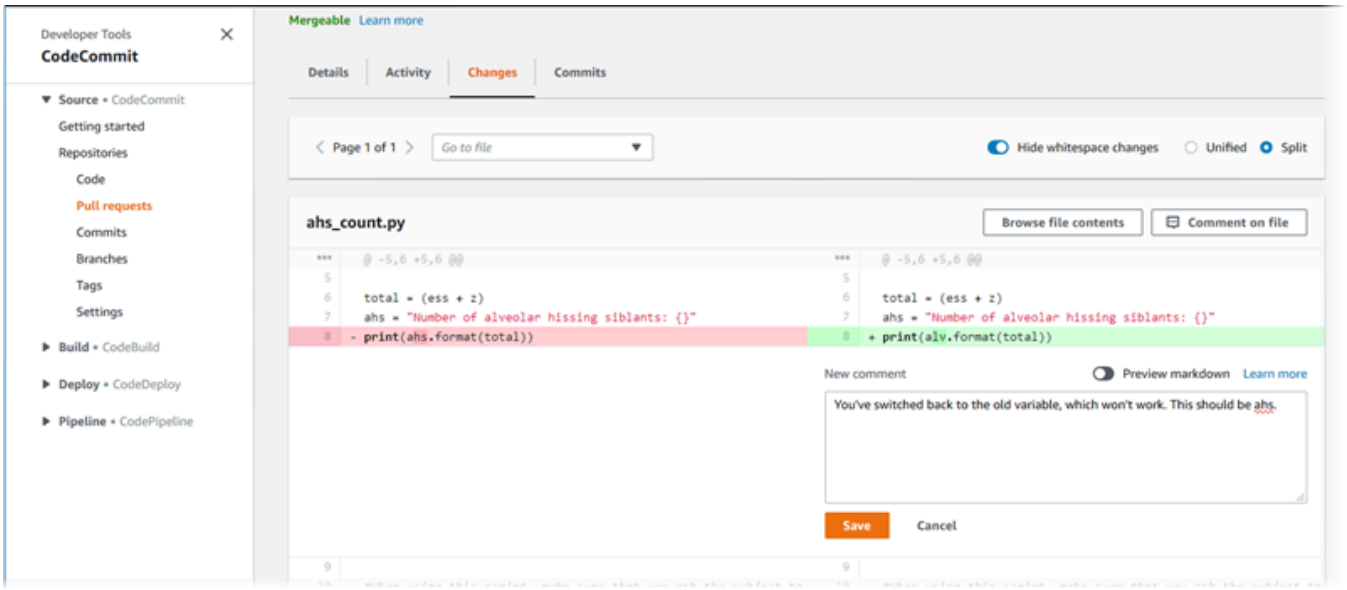
     ，输入注释，然后选择 Save (保存)。



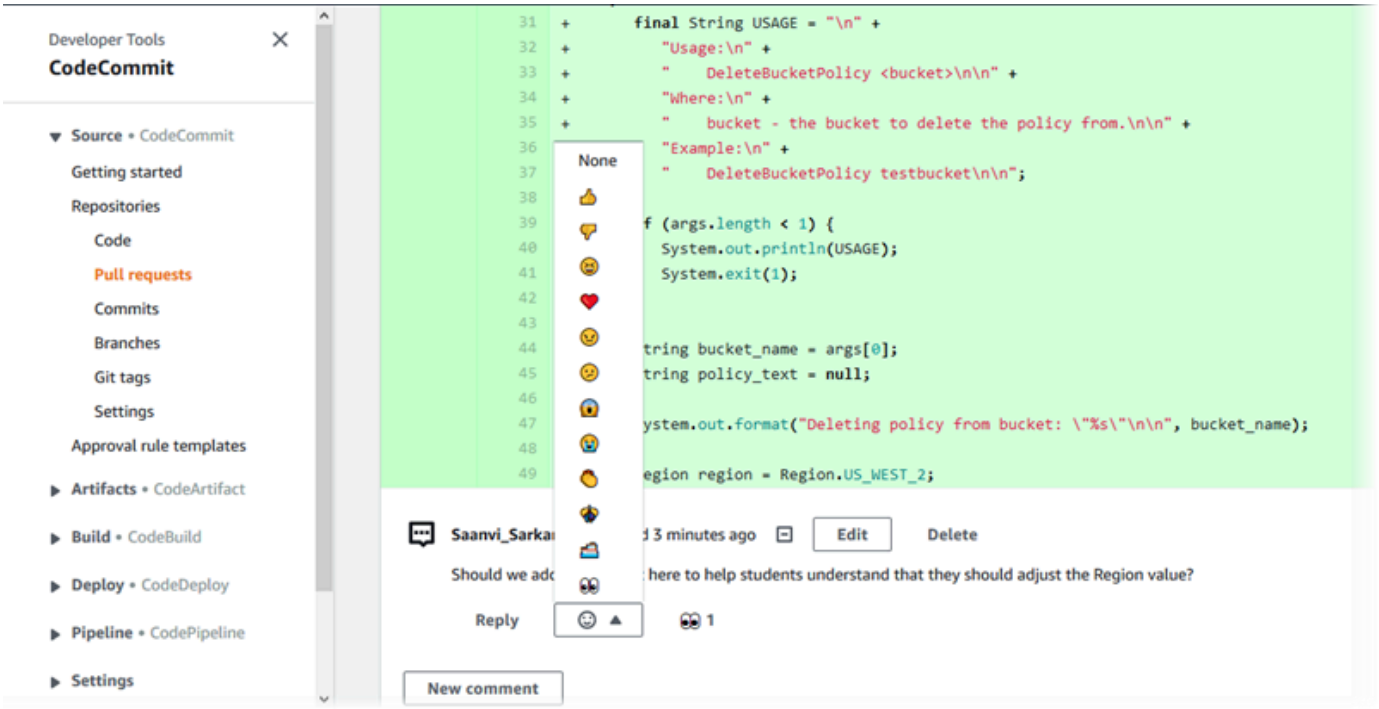
   - 要向拉取请求中的已更改行添加评论，请在 Changes 中转到要评论的行。选择为该行显示的注释图标

，

输入注释，然后选择 Save (保存)。

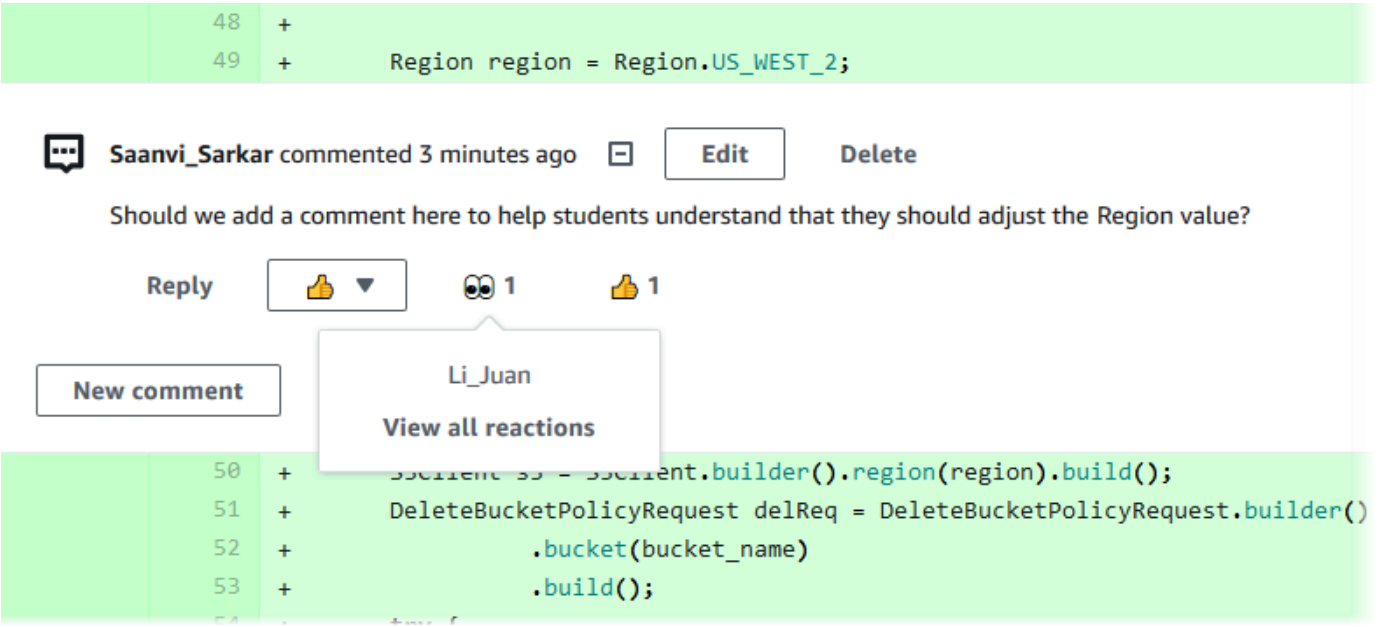


7. 要回复对提交的评论，请在 Changes 或 Activity 中，选择 Reply。您可以使用文字和表情符号进行回复。



您可以通过选择特定表情符号反应回复来查看使用其进行回复的人员的姓名。要查看所有表情符号反应以及有关谁使用了哪些表情符号进行回复的信息，请选择查看所有反应。如果您使用表情符号回复了评论，则您的回复将显示在表情符号反应按钮的图标中。

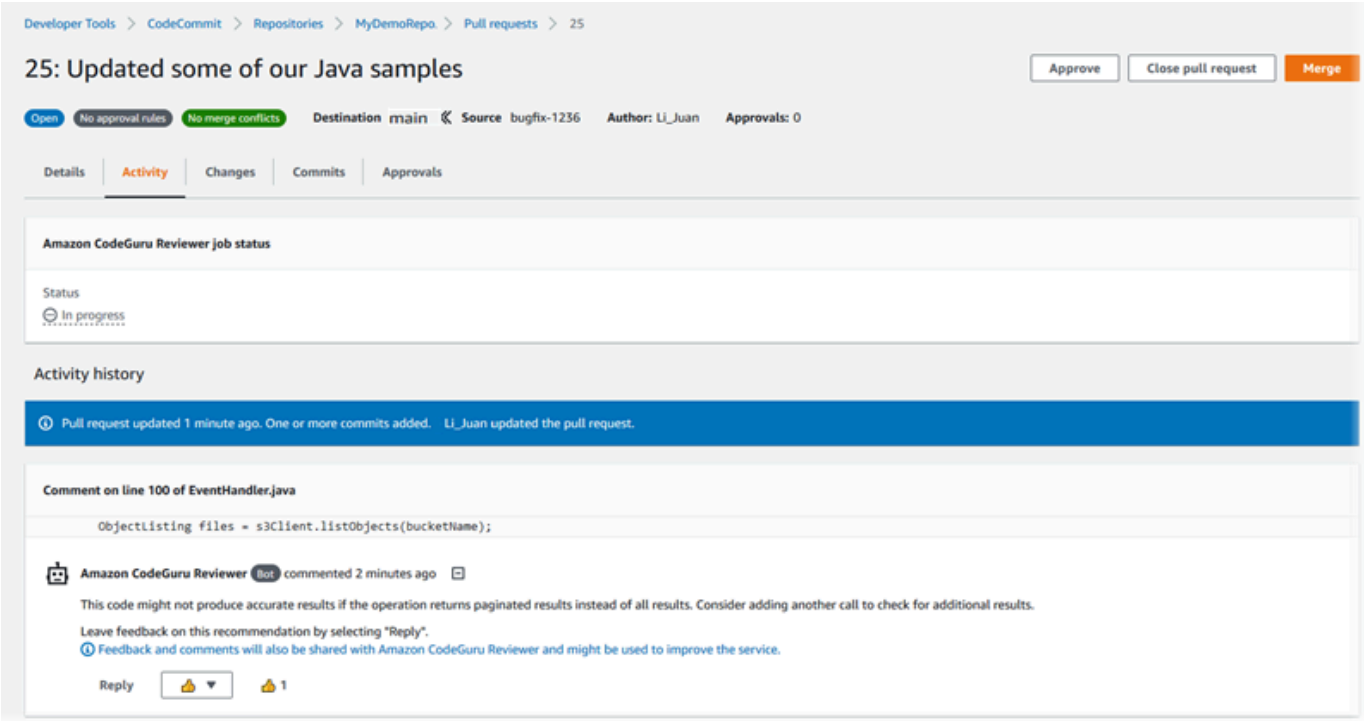
> **Note**
>
> 控制台中显示的反应计数在页面加载时是准确的。要了解有关表情符号反应计数的最新信息，请刷新页面，或选择查看所有反应。



8. （可选）要回复 Amazon CodeGuru Reviewer 创建的推荐，包括提供有关推荐质量的反馈，请选择回复。可使用反应按钮，提供有关您是批准还是不批准建议的一般信息。您可使用注释字段，提供有关反应的详细信息。

> **Note**
>
> Amazon CodeGuru Reviewer 是一项自动代码审查服务，它使用程序分析和机器学习来检测 Java 或 Python 代码中的常见问题并推荐修复方法。
>
> - 只有当您已将存储库与 Amazon CodeGuru Reviewer 关联起来、分析已完成以及拉取请求中的代码是 Java 或 Python 代码时，您才能看到 Amazon CodeGuru Reviewer 评论。有关更多信息，请参阅 将 AWS CodeCommit 存储库与 Amazon CodeGuru Reviewer 关联或取消关联。
> - Amazon CodeGuru Reviewer 评论只有在对拉取请求的最新修订版发表评论时，才会显示在 “更改” 选项卡中。它们始终显示在活动选项卡中。

- 虽然您可以对 Amazon CodeGuru Reviewer 推荐使用任何可用的表情符号反应进行回应，但只能使用竖起大拇指和竖起大拇指的表情符号反应来评估推荐的用处。

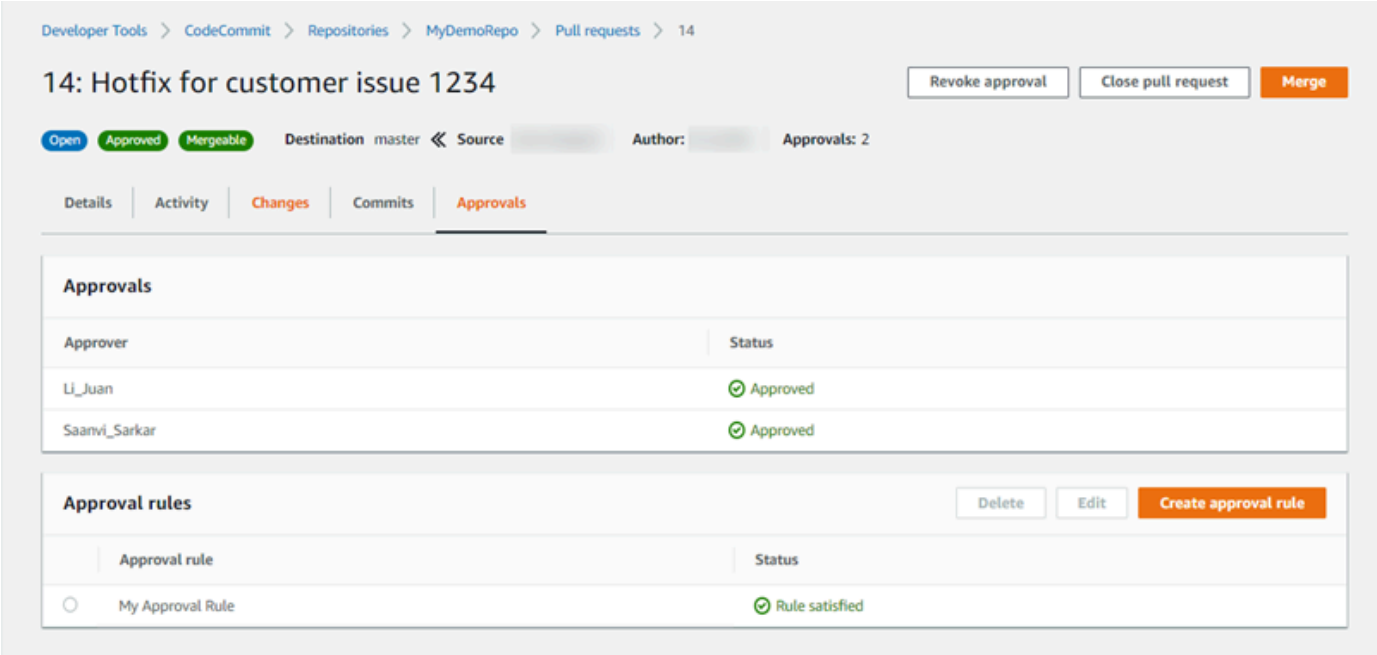


9. 要审批拉取请求中所做的更改，请选择 Approve (审批)。

> **Note**
>
> 您无法批准自己创建的拉取请求。

您可以在 Approvals (审批) 中，查看审批、拉取请求的审批规则以及由审批规则模板创建的审批规则。如果您最终决定不希望审批拉取请求，则可选择 Revoke approval (撤销审批)。

> **Note**
>
> 您只能对处于打开状态的拉取请求进行审批或撤销审批。不能对状态为“已合并”或“已关闭”的拉取请求进行审批或撤销审批。



审核拉取请求 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

您可以使用以下 AWS CLI 命令查看拉取请求：

- post-comment-for-pull-request，用于向拉取请求添加评论
- get-comments-for-pull-request，用于查看拉取请求中留下的评论
- update-pull-request-approval-state，用于审批或撤销对拉取请求的批准
- post-comment-reply，用于回复拉取请求中的评论

您还可以运行以下命令，对拉取请求中的评论使用表情符号：

- 要使用表情符号回复评论，请运行 put-comment-reaction。
- 要查看对评论的表情符号反应，请运行 get-comment-reactions。

使用 AWS CLI 查看 CodeCommit 存储库中的拉取请求

1. 要向存储库中的拉取请求添加评论，请运行 post-comment-for-pull-request 命令，并且指定：

- 拉取请求的 ID（使用 --pull-request-id 选项）。
- 包含拉取请求的存储库的名称（使用 --repository-name 选项）。
- 将在其中合并拉取请求的目标分支中的提交的完整提交 ID（使用 --before-commit-id 选项）。
- 源分支中的提交的完整提交 ID，它在发布评论时作为拉取请求的分支的当前提示（使用 --after-commit-id 选项）。
- 客户端生成的唯一等幂令牌（使用 --client-request-token 选项）。
- 您的评论的内容（使用 --content 选项）。
- 有关评论放置位置信息的列表，包括：
  - 所比较文件的名称，包括其扩展名和子目录（如果有，则使用 filePath 属性）。
  - 在比较文件中更改的行号（使用 filePosition 属性）。
  - 对更改的评论在源分支与目标分支之间比较“之前”还是“之后”（使用 relativeFileVersion 属性）。

例如，使用以下命令添加评论 *"These don't appear to be used anywhere. Can we remove them?"*。关于在名为的存储库中 ID 为 *47* 的拉取请求中对 *ahs_count.py* 文件的更改*MyDemoRepo*。

```
aws codecommit post-comment-for-pull-request --pull-request-id "47" --
repository-name MyDemoRepo --before-commit-id 317f8570EXAMPLE --after-
commit-id 5d036259EXAMPLE --client-request-token 123Example --content
 "These don't appear to be used anywhere. Can we remove them?" --location
 filePath=ahs_count.py,filePosition=367,relativeFileVersion=AFTER
```

如果成功，该命令产生类似以下内容的输出。

```
{
        "afterBlobId": "1f330709EXAMPLE",
        "afterCommitId": "5d036259EXAMPLE",
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "317f8570EXAMPLE",
        "comment": {
              "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
              "clientRequestToken": "123Example",
              "commentId": "abcd1234EXAMPLEb5678efgh",
              "content": "These don't appear to be used anywhere. Can we remove
 them?",
```

```
                "creationDate": 1508369622.123,
                "deleted": false,
                "lastModifiedDate": 1508369622.123,
                "callerReactions": [],
                "reactionCounts": []
             }
              "location": {
                "filePath": "ahs_count.py",
                "filePosition": 367,
                "relativeFileVersion": "AFTER"
              },
          "repositoryName": "MyDemoRepo",
          "pullRequestId": "47"
 }
```

2. 要查看拉取请求的评论，请运行 get-comments-for-pull-request 命令，并且指定：

   - CodeCommit 存储库的名称（带--repository-name选项）。
   - 系统生成的拉取请求的 ID（使用 --pull-request-id 选项）。
   - （可选）要返回下一批结果的枚举令牌（使用 --next-token 选项）。
   - (可选) 一个用于限制返回的结果数的非负整数 (使用 --max-results 选项)。

   例如，可使用此命令查看 ID 为 42 的拉取请求的注释。

   ```
   aws codecommit get-comments-for-pull-request --pull-request-id 42
   ```

   如果成功，该命令产生类似以下内容的输出。

   ```
   {
      "commentsForPullRequestData": [
         {
            "afterBlobId": "1f330709EXAMPLE",
            "afterCommitId": "5d036259EXAMPLE",
            "beforeBlobId": "80906a4cEXAMPLE",
            "beforeCommitId": "317f8570EXAMPLE",
            "comments": [
               {
                  "authorArn": "arn:aws:iam::111111111111:user/Saanvi_Sarkar",
                  "clientRequestToken": "",
                  "commentId": "abcd1234EXAMPLEb5678efgh",
   ```

```
                "content": "These don't appear to be used anywhere. Can we remove
 them?",
                "creationDate": 1508369622.123,
                "deleted": false,
                "lastModifiedDate": 1508369622.123,
                "callerReactions": [],
                "reactionCounts":
                 {
                   "THUMBSUP" : 6,
                   "CONFUSED" : 1
                 }
            },
            {
                "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
                "clientRequestToken": "",
                "commentId": "442b498bEXAMPLE5756813",
                "content": "Good catch. I'll remove them.",
                "creationDate": 1508369829.104,
                "deleted": false,
                "lastModifiedDate": 150836912.273,
                "callerReactions": ["THUMBSUP"]
                "reactionCounts":
                 {
                   "THUMBSUP" : 14
                 }
             }
        ],
        "location": {
           "filePath": "ahs_count.py",
           "filePosition": 367,
           "relativeFileVersion": "AFTER"
        },
        "repositoryName": "MyDemoRepo",
        "pullRequestId": "42"
      }
   ],
   "nextToken": "exampleToken"
}
```

3. 要对拉取请求进行审批或撤销审批，请运行 update-pull-request-approval-state 命令，并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。

- 拉取请求的修订 ID（使用 --revision-id option））。您可以使用get-pull-request命令获取拉取请求的当前修订版 ID。
- 要应用的审批状态（使用 --approval-state 选项）。有效的审批状态包括 APPROVE 和 REVOKE。

例如，可使用此命令对 ID 为 *27* 且修订 ID 为 *9f29d167EXAMPLE* 的拉取请求进行审批。

```
aws codecommit update-pull-request-approval-state --pull-request-id 27 --revision-
id 9f29d167EXAMPLE --approval-state "APPROVE"
```

如果成功，该命令不返回任何内容。

4. 要在拉取请求中发布对评论的回复，请运行 post-comment-reply 命令，并且指定：

- 要回复的评论的系统生成的 ID（使用 --in-reply-to 选项）。
- 客户端生成的唯一等幂令牌（使用 --client-request-token 选项）。
- 您的回复的内容（使用 --content 选项）。

例如，使用以下命令将回复 *"Good catch. I'll remove them."* 添加到系统生成的 ID 为 *abcd1234EXAMPLEb5678efgh* 的评论中。

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

如果成功，该命令产生类似以下内容的输出。

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
```

```
}
```

更新拉取请求

您可以将提交推送到处于打开状态的拉取请求的源分支，从而通过进一步的代码更改来更新拉取请求。有关更多信息，请参阅在中创建提交 AWS CodeCommit。

您可以使用 AWS CodeCommit 控制台或 AWS CLI 更新拉取请求的标题或描述。如果存在以下情况，您可能需要更新拉取请求的标题或描述：

- 其他用户不了解描述或原始标题产生误导。
- 您希望标题或描述反映您对处于打开状态的拉取请求的源分支所做的更改。

更新拉取请求（控制台）

您可以使用 CodeCommit 控制台更新 CodeCommit 存储库中的拉取请求的标题和描述。要更新拉取请求中的代码，请将提交推送到处于打开状态的拉取请求的源分支。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中更新拉取请求的存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。选择要更新的处于打开状态的拉取请求。
5. 在拉取请求中，选择 Details (详细信息)，然后选择 Edit details (编辑详细信息) 以编辑标题或描述。

> **Note**
>
> 无法更新已关闭或已合并的拉取请求的标题或描述。

更新拉取请求 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

您可能还对以下命令感兴趣：

- update-pull-request-approval-state，用于审批或撤销对拉取请求的审批。
- create-pull-request-approval-rule，用于为拉取请求创建审批规则。
- delete-pull-request-approval-rule，用于删除拉取请求的审批规则。
- 使用创建提交 AWS CLI 或 使用 Git 客户端创建提交，用于创建其他代码更改并将其推送到处于打开状态的拉取请求的源分支。

使用 AWS CLI 更新 CodeCommit 存储库中的拉取请求

1. 要更新存储库中拉取请求的标题，请运行 update-pull-request-title 命令，并且指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 拉取请求的标题（使用 --title 选项）。

   例如，要更新 ID 为 *47* 的拉取请求的标题：

   ```
   aws codecommit update-pull-request-title --pull-request-id 47 --title
    "Consolidation of global variables - updated review"
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "pullRequest": {
           "approvalRules": [
               {
                   "approvalRuleContent": "{\"Version\": \"2018-11-08\",
   \"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
   \": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
    [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                   "approvalRuleId": "dd8b17fe-EXAMPLE",
                   "approvalRuleName": "2-approver-rule-for-main",
                   "creationDate": 1571356106.936,
                   "lastModifiedDate": 571356106.936,
                   "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                   "originApprovalRuleTemplate": {
                       "approvalRuleTemplateId": "dd8b26gr-EXAMPLE",
                       "approvalRuleTemplateName": "2-approver-rule-for-main"
                   },
                   "ruleContentSha256": "4711b576EXAMPLE"
               }
   ```

```
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.12,
        "description": "Review the latest changes and updates to the global
 variables. I have updated this request with some changes, including removing some
 unused variables.",
        "lastActivityDate": 1508372657.188,
        "pullRequestId": "47",
        "pullRequestStatus": "OPEN",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables - updated review"
    }
}
```

2. 要更新拉取请求的描述，请运行 update-pull-request-description 命令，并且指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 描述（使用 --description 选项）。

   例如，要更新 ID 为 *47* 的拉取请求的描述：

   ```
   aws codecommit update-pull-request-description --pull-request-id 47 --description
    "Updated the pull request to remove unused global variable."
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
      "pullRequest": {
         "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
         "clientRequestToken": "",
   ```

```
      "creationDate": 1508530823.155,
      "description": "Updated the pull request to remove unused global variable.",
      "lastActivityDate": 1508372423.204,
      "pullRequestId": "47",
      "pullRequestStatus": "OPEN",
      "pullRequestTargets": [
         {
            "destinationCommit": "9f31c968EXAMPLE",
            "destinationReference": "refs/heads/main",
            "mergeMetadata": {
               "isMerged": false,
            },
            "repositoryName": "MyDemoRepo",
            "sourceCommit": "99132ab0EXAMPLE",
            "sourceReference": "refs/heads/variables-branch"
         }
      ],
      "title": "Consolidation of global variables"
   }
}
```

编辑或删除拉取请求的审批规则

如果拉取请求具有审批规则，则该拉取请求无法合并，直到其条件得以满足。您可以更改拉取请求的审批规则，以使其条件更容易满足，或提高审核的严谨性。您可以更改必须审批拉取请求的用户数量。还可以在规则的用户审批池中添加、删除或更改成员资格。最后，如果您不想再对拉取请求使用审批规则，可以将其删除。

> **Note**
>
> 您还可以覆盖拉取请求的审批规则。有关更多信息，请参阅覆盖拉取请求的审批规则。

您可以使用 AWS CodeCommit 控制台或 AWS CLI 编辑和删除存储库的审批规则。

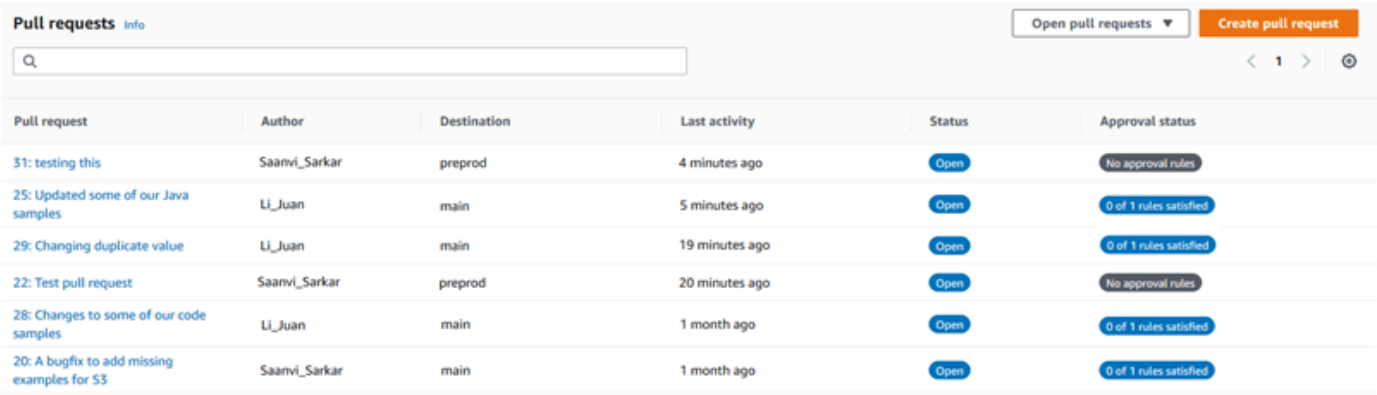
主题

- 编辑或删除拉取请求的审批规则（控制台）
- 编辑或删除拉取请求的审批规则 (AWS CLI)

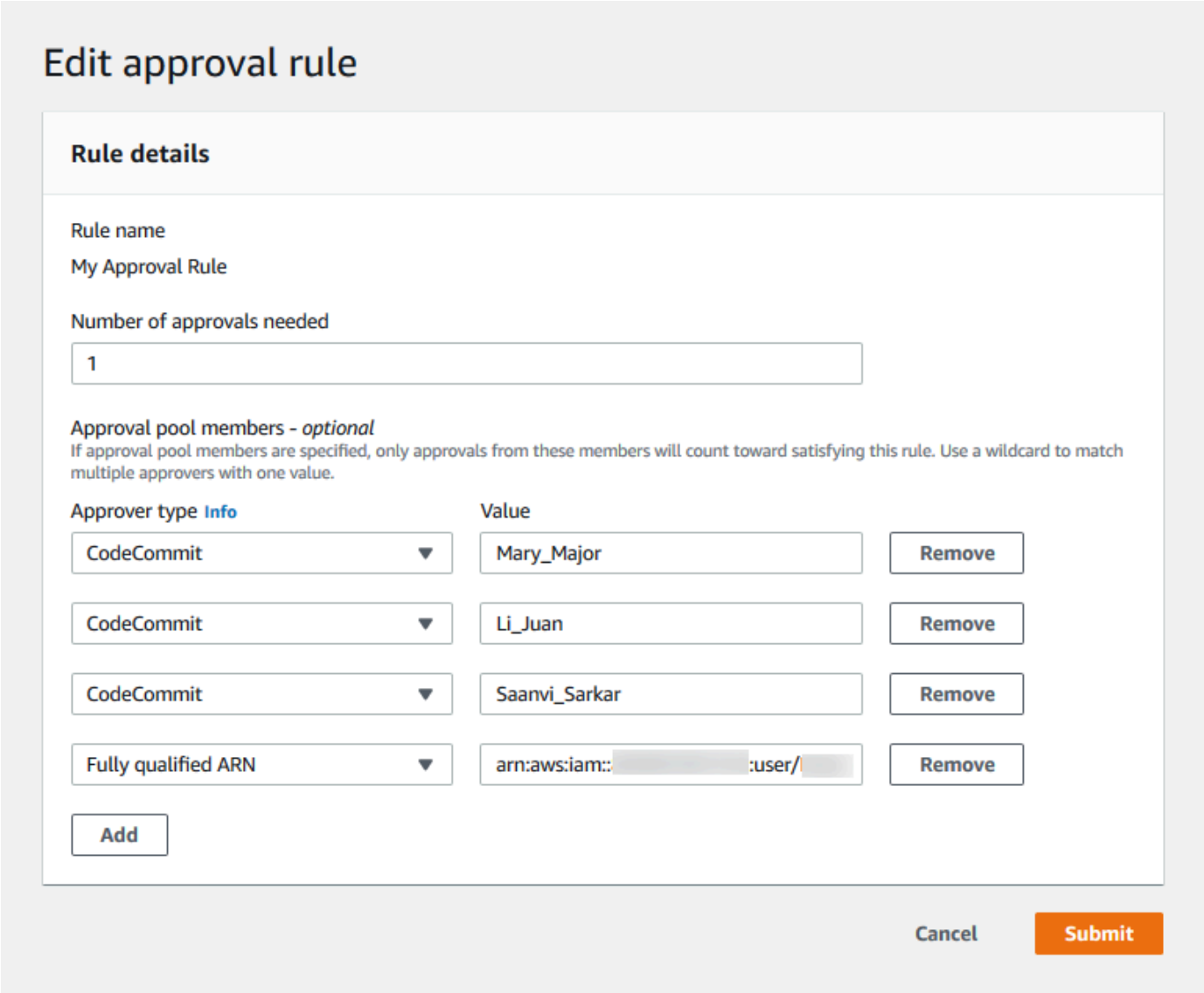
编辑或删除拉取请求的审批规则（控制台）

您可以使用 CodeCommit 控制台为 CodeCommit 存储库中的拉取请求编辑或删除审批规则。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中编辑或删除拉取请求的审批规则的存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 选择要编辑或删除审批规则的拉取请求。您只能编辑和删除处于打开状态的拉取请求的审批规则。



5. 在拉取请求中，选择 Approvals (审批)，然后从列表中选择要编辑或删除的规则。请执行下列操作之一：
   - 如果要编辑规则，请选择 Edit (编辑)。
   - 如果要删除规则，请选择 Delete (删除)，然后按照说明进行操作，以验证规则是否删除。
6. 在 Edit approval rule (编辑审批规则) 中，对规则进行所需的更改，然后选择 Submit (提交)。



7. 完成审批规则的配置之后，选择 Submit (提交)。

编辑或删除拉取请求的审批规则 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

您可以使用 AWS CLI 编辑审批规则的内容并删除审批规则。

> **Note**
>
> 您可能还对以下命令感兴趣：
>
> - update-pull-request-approval-state，用于审批或撤销对拉取请求的审批。

- get-pull-request-approval-states，用于查看拉取请求的审批。
- evaluate-pull-request-approval-rules，用于确定拉取请求的审批规则是否满足了其条件。

使用 AWS CLI 为 CodeCommit 存储库中的拉取请求编辑或删除审批规则

1. 要编辑审批规则，请运行 update-pull-request-approval-rule-content 命令，并指定：

   - 拉取请求的 ID（使用 --id 选项）。
   - 审批规则的名称（使用 --approval-rule-name 选项）。
   - 审批规则的内容（使用 --approval-rule-content 选项）。

   此示例更新了 ID 为 *27* 的拉取请求的名为 *Require two approved approvers* 的审批规则。该规则要求一个用户获得批准池的批准，该批准池包括 *123456789012* Amazon Web Services 账户中的任何 IAM 用户：

```
aws codecommit update-pull-request-approval-rule-content --pull-request-id 27
 --approval-rule-name "Require two approved approvers" --approval-rule-content
 "{Version: 2018-11-08, Statements: [{Type: \"Approvers\", NumberOfApprovalsNeeded:
 1, ApprovalPoolMembers:[\"CodeCommitApprovers:123456789012:user/*\"]}]}"
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "approvalRule": {
        "approvalRuleContent": "{Version: 2018-11-08, Statements:
 [{Type: \"Approvers\", NumberOfApprovalsNeeded: 1, ApprovalPoolMembers:
[\"CodeCommitApprovers:123456789012:user/*\"]}]}",
        "approvalRuleId": "aac33506-EXAMPLE",
        "originApprovalRuleTemplate": {},
        "creationDate": 1570752871.932,
        "lastModifiedDate": 1570754058.333,
        "approvalRuleName": Require two approved approvers",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "ruleContentSha256": "cd93921cEXAMPLE",
    }
}
```

3.

要删除审批规则，请运行 delete-pull-request-approval-rule 命令，并指定：

- 拉取请求的 ID（使用 --id 选项）。
- 审批规则的名称（使用 --approval-rule-name 选项）。

例如，要删除 ID 为 *15* 的拉取请求的名为 *My Approval Rule* 的审批规则，请运行以下命令：

```
aws codecommit delete-pull-request-approval-rule --pull-request-id 15 --approval-
rule-name "My Approval Rule"
```

如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleId": "077d8e8a8-EXAMPLE"
}
```

覆盖拉取请求的审批规则

在正常开发过程中，您希望用户在合并拉取请求之前满足审批规则的条件。但是，有时您可能需要加快合并拉取请求。例如，您可能希望在生产中放置错误修复，但审批池中无人可审批拉取请求。在这种情况下，您可以选择覆盖拉取请求的审批规则。您可以覆盖拉取请求的所有审批规则，包括那些专门为拉取请求创建并从审批规则模板生成的规则。您不能选择性地覆盖特定审批规则，只能覆盖所有规则。通过覆盖规则预留审批规则要求之后，可以将拉取请求合并到其目标分支中。

当您覆盖拉取请求的审批规则时，有关覆盖这些规则的用户的信息将记录在拉取请求的活动中。这样一来，您可以返回拉取请求的历史记录，查看是谁覆盖了规则。如果拉取请求仍处于打开状态，您也可以选择撤销覆盖。拉取请求在合并之后，便无法再撤销覆盖。

主题

- 覆盖审批规则（控制台）
- 覆盖审批规则 (AWS CLI)

覆盖审批规则（控制台）

作为拉取请求审核的一部分，您可以在控制台中覆盖拉取请求的审批规则要求。如果您改变主意，可以撤销覆盖，并重新应用审批规则要求。只有当拉取请求仍处于打开状态时，才能覆盖审批规则或撤销覆盖。如果拉取请求已合并或关闭，则无法更改其覆盖状态。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称。
3. 在导航窗格中，选择拉取请求。选择要覆盖其审批规则要求，或撤销覆盖的拉取请求。
4. 在 Approvals (审批) 选项卡上，选择 Override approval rules (覆盖审批规则)。这些要求将预留，且按钮文本更改为 Revoke override (撤销覆盖)。要重新应用审批规则要求，请选择 Revoke override (撤销覆盖)。

覆盖审批规则 (AWS CLI)

您可以使用 AWS CLI 覆盖审批规则要求。您还可以使用它来查看拉取请求的覆盖状态。

覆盖拉取请求的审批规则要求

1. 在终端或命令行中，运行 override-pull-request-approval-rules 命令，并指定：

   - 系统生成的拉取请求的 ID。
   - 拉取请求的最新修订 ID。要查看此信息，请使用 get-pull-request。
   - 您希望的覆盖状态，OVERRIDE 或 REVOKE。REVOKE 状态会删除 OVERRIDE 状态，但不会保存。

   例如，要覆盖 ID 为 **34** 且修订 ID 为 **927df8d8EXAMPLE** 的拉取请求的审批规则，请运行以下命令：

   ```
   aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
   revision-id 927df8d8dEXAMPLE --override-status OVERRIDE
   ```

2. 如果成功，该命令不返回任何内容。
3. 要撤销 ID 为 **34** 且修订 ID 为 **927df8d8EXAMPLE** 的拉取请求的覆盖，请运行以下命令：

   ```
   aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
   revision-id 927df8d8dEXAMPLE --override-status REVOKE
   ```

获取有关拉取请求覆盖状态的信息

1. 在终端或命令行中，运行 get-pull-request-override-state 命令，并指定：

   - 系统生成的拉取请求的 ID。
   - 拉取请求的最新修订 ID。要查看此信息，请使用 get-pull-request。

   例如，要查看 ID 为 **34** 且修订 ID 为 **927df8d8EXAMPLE** 的拉取请求的覆盖状态，请运行以下命令：

   ```
   aws codecommit get-pull-request-override-state --pull-request-id 34 --revision-
   id 927df8d8dEXAMPLE
   ```

2. 如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "overridden": true,
       "overrider": "arn:aws:iam::123456789012:user/Mary_Major"
   }
   ```

合并 AWS CodeCommit 存储库中的拉取请求

审核完代码并满足拉取请求的所有审批规则（如果有）后，您可以通过以下几种方式之一合并拉取请求：

- 您可以在控制台中，使用可用的合并策略之一，将源分支合并到目标分支中，这也会关闭拉取请求。您也可以在控制台中解决合并冲突。控制台将显示一条消息，指示拉取请求是否可合并，或者是否必须解决冲突。当所有冲突都得以解决并选择 Merge (合并) 后，将使用您所选合并策略执行合并。快进是默认的合并策略，这是 Git 的默认选项。根据源和目标分支中的代码的状态，可能无法使用该策略，但可以使用其他选项，例如压缩或三向。
- 在 AWS CLI 中，您可以使用快进、压缩或三向合并策略合并和关闭拉取请求。
- 在本地计算机上，可以使用 git merge 命令将源分支合并到目标分支中，然后将合并的代码推送到目标分支。这种方法有缺点，你应该仔细考虑。无论对拉取请求的审批规则要求是否得以满足，它都会合并拉取请求，从而绕过这些控件。如果使用快进合并策略合并拉取请求，则合并和推送目标分支也会自动关闭拉取请求。这种方法的一个优点是，git merge 命令允许您选择 CodeCommit 控制台中不可用的合并选项或策略。有关 git merge 和合并选项的更多信息，请参阅 git-merge 或 Git 文档。

如果拉取请求的源分支或目标分支已删除，CodeCommit 将自动关闭拉取请求。

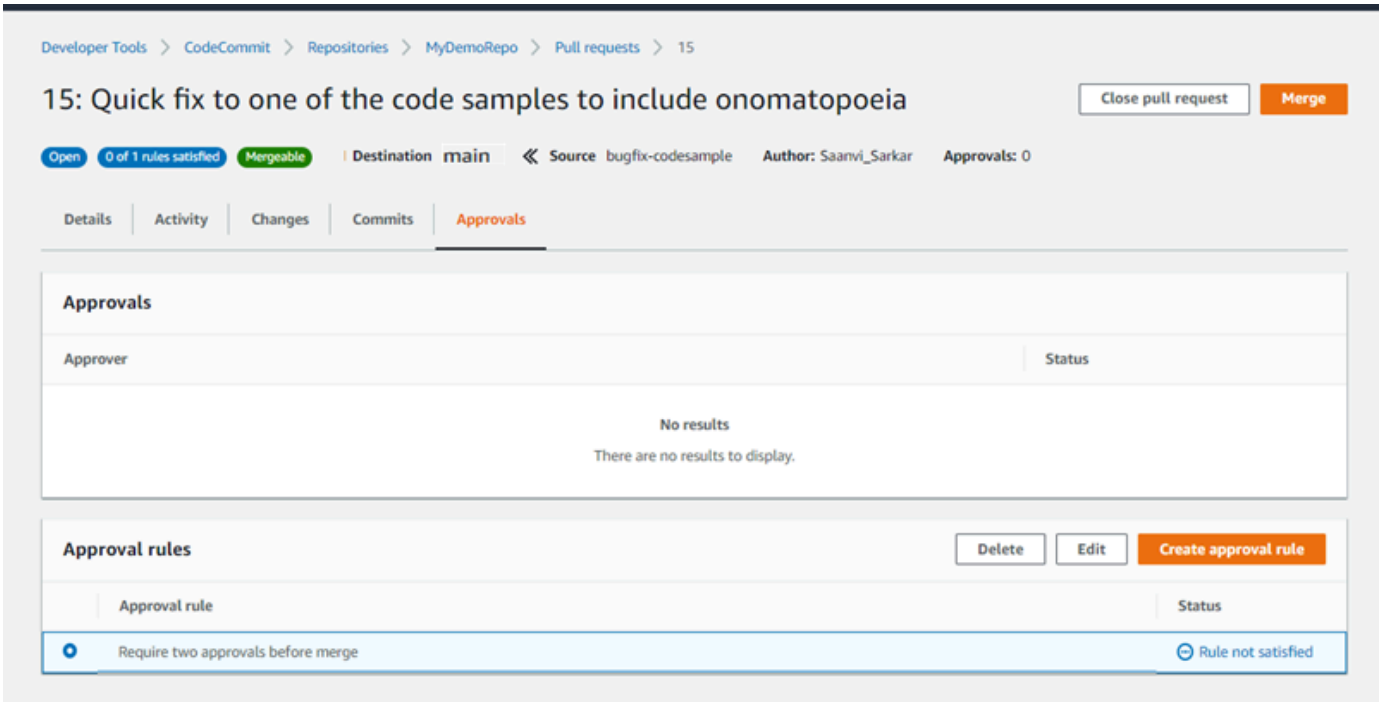
主题

合并拉取请求（控制台）

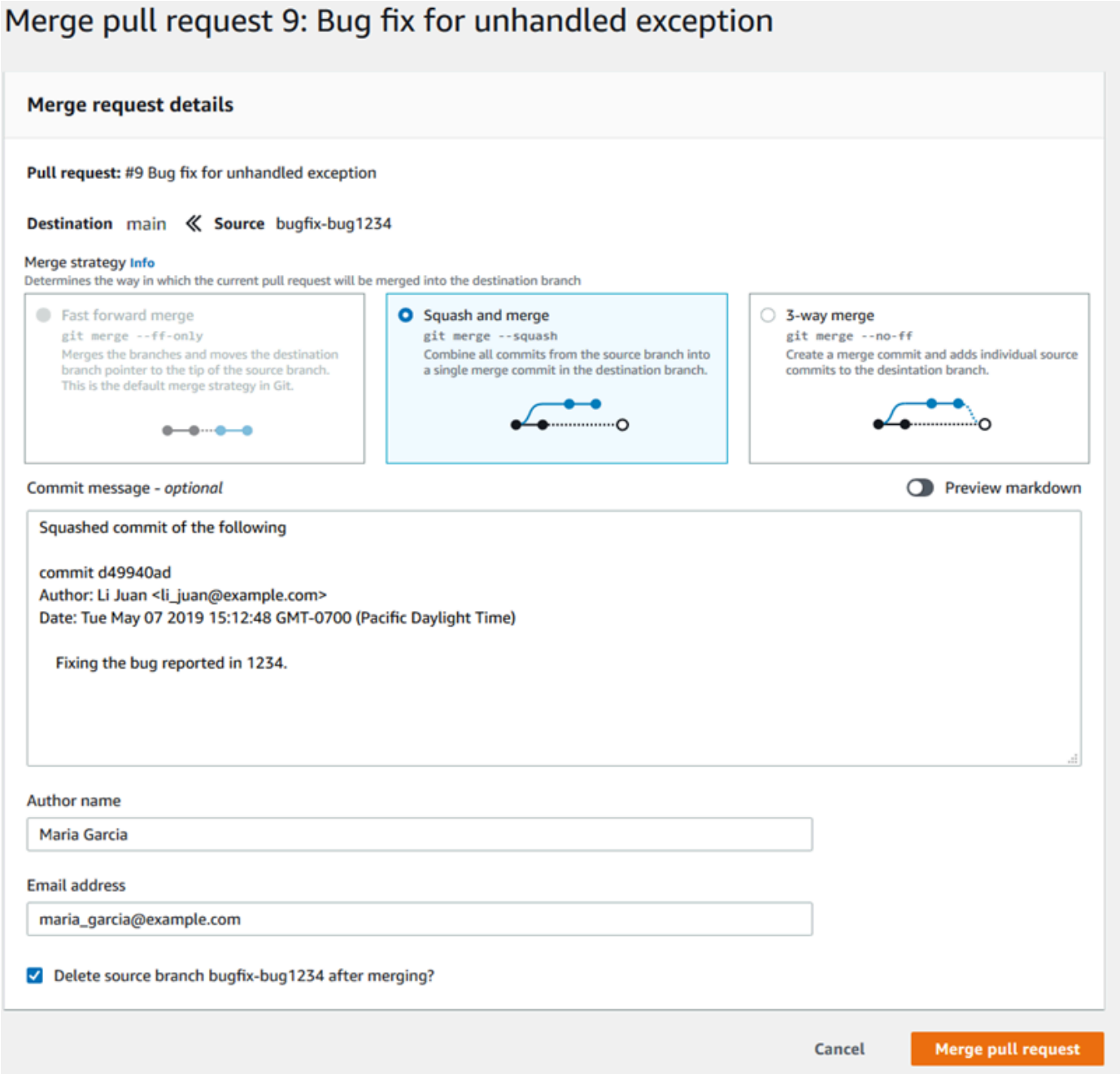
您可以使用 CodeCommit 控制台合并 CodeCommit 存储库中的拉取请求。在拉取请求的状态变为 Merged (已合并) 后，它不再显示在处于打开状态的拉取请求列表中。合并的拉取请求划分为 closed (已关闭)。无法将其改回到 Open (打开)，但用户仍然可以对更改发表评论和回复评论。合并或关闭拉取请求后，您无法对其进行审批、撤销其审批，或覆盖应用于拉取请求的审批规则。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。选择要合并的打开拉取请求。
5. 在拉取请求中，选择 Approvals (审批)。查看审批人列表，并验证所有审批规则（如果有）都已满足其条件。如果一个或多个审批规则的状态为 Rule not satisfied (不满足规则)，则无法合并拉取请求。如果没有人审批拉取请求，请考虑是否要合并该请求，或者是否要等待审批。

> **Note**
>
> 如果为拉取请求创建了审批规则，您可以对其进行编辑或删除以取消阻止合并。如果审批规则是使用审批规则模板创建的，则无法编辑或删除该规则。您只能选择覆盖要求。有关更多信息，请参阅 覆盖拉取请求的审批规则。



6. 选择 Merge (合并)。
7. 在拉取请求中，在可用的合并策略之间进行选择。无法应用的合并策略将灰显。如果没有可用的合并策略，您可以选择在 CodeCommit 控制台中手动解决冲突，也可以使用 Git 客户端在本地解决冲突。有关更多信息，请参阅 解决 AWS CodeCommit 存储库中的拉取请求中的冲突。



- 快进合并会将目标分支引用向前移动到源分支的最新提交。如果可能，这是 Git 的默认行为。不会创建合并提交，但会保留源分支中的所有提交历史记录，就好像它是在目标分支中执行的一样。快进合并在目标分支历史记录的提交可视化视图中不会显示为分支合并，因为没有创建合并提交。源分支提示会快进到目标分支提示。
- 压缩合并会创建一个包含源分支更改的提交，并将该单个压缩提交应用于目标分支。默认情况下，该压缩提交的提交消息包含源分支中的更改的所有提交消息。不会保留分支更改的各个提交历史记录。这可能有助于简化存储库历史记录，同时仍在目标分支历史记录的提交可视化工具视图中保留合并的图形表示形式。

- 三向合并会为目标分支中的合并创建合并提交，但还会保留在源分支中执行的各个提交，以作为目标分支历史记录的一部分。这有助于保留存储库更改的完整历史记录。

8. 如果选择压缩或三向合并策略，请查看自动生成的提交消息，并在要更改该信息时对其进行修改。为提交历史记录添加您的姓名和电子邮件地址。
9. （可选）清除在合并期间删除源分支的选项。默认情况下，在合并拉取请求时会删除源分支。
10. 选择 Merge pull request (合并拉取请求) 以完成合并。

合并拉取请求 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

使用 AWS CLI 合并 CodeCommit 存储库中的拉取请求

1. 要评估拉取请求是否已满足其所有审批规则并准备好合并，请运行 evaluate-pull-request-approval-rules 命令，并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 拉取请求的修订 ID（使用 --revision-id option)）。您可以使用 get-pull-request 命令获取拉取请求的当前修订 ID。

   例如，要对 ID 为 *27* 且修订 ID 为 *9f29d167EXAMPLE* 的拉取请求评估审批规则的状态，请运行以下命令：

   ```
   aws codecommit evaluate-pull-request-approval-rules --pull-request-id 27 --
   revision-id 9f29d167EXAMPLE
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "evaluation": {
           "approved": false,
           "approvalRulesNotSatisfied": [
               "Require two approved approvers"
           ],
           "overridden": false,
           "approvalRulesSatisfied": []
       }
   ```

```
}
```

> **Note**
>
> 此输出指示拉取请求不可合并，因为未满足审批规则的要求。要合并此拉取请求，您可以让审核人审批该请求，以满足规则的条件。根据您的权限和规则的创建方式，您也可以编辑、覆盖或删除规则。有关更多信息，请参阅审核拉取请求、覆盖拉取请求的审批规则和编辑或删除拉取请求的审批规则。

2. 要使用快进合并策略合并和关闭拉取请求，请运行 merge-pull-request-by-fast-forward 命令并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 源分支提示的完整提交 ID（使用 --source-commit-id 选项）。
   - 存储库的名称（使用 --repository-name 选项）。

   例如，要在名为 *MyDemoRepo* 的存储库中合并并关闭 ID 为 *47* 并且源提交 ID 为 *99132ab0EXAMPLE* 的拉取请求，请运行以下命令：

   ```
   aws codecommit merge-pull-request-by-fast-forward --pull-request-id 47 --source-commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "pullRequest": {
           "approvalRules": [
               {
                   "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                   "approvalRuleId": "dd8b17fe-EXAMPLE",
                   "approvalRuleName": "I want one approver for this pull request",
                   "creationDate": 1571356106.936,
                   "lastModifiedDate": 571356106.936,
                   "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                   "ruleContentSha256": "4711b576EXAMPLE"
               }
           ],
   ```

```
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

3. 要使用压缩合并策略合并和关闭拉取请求，请运行 merge-pull-request-by-squash 命令并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 源分支提示的完整提交 ID（使用 --source-commit-id 选项）。
   - 存储库的名称（使用 --repository-name 选项）。
   - 要使用的冲突详细信息级别（使用 --conflict-detail-level 选项）。如果未指定，则使用默认值 **FILE_LEVEL**。
   - 要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。如果未指定，它默认为 NONE，并且必须手动解决冲突。
   - 要包括的提交消息（使用 --commit-message 选项）。
   - 要用于提交的姓名（使用 --author-name 选项）。
   - 要用于提交的电子邮件地址（使用 --email 选项）。
   - 是否保留任何空文件夹（使用 --keep-empty-folders 选项）。

以下示例展示了在名为 *MyDemoRepo* 的存储库中合并并关闭 ID 为 *47* 并且源提交 ID 为 *99132ab0EXAMPLE* 的拉取请求。它使用 LINE_LEVEL 冲突详细信息和 ACCEPT_SOURCE 冲突解决策略：

```
aws codecommit merge-pull-request-by-squash --pull-request-id 47 --source-commit-
id 99132ab0EXAMPLE --repository-name MyDemoRepo --conflict-detail-level LINE_LEVEL
 --conflict-resolution-strategy ACCEPT_SOURCE --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging pull request 47 by squash and
 accepting source in merge conflicts"
```

如果成功，该命令生成与快进合并相同类型的输出，类似于以下内容：

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
```

```
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

4. 要使用三向合并策略合并和关闭拉取请求，请运行 merge-pull-request-by-three-way 命令并指定：

   - 拉取请求的 ID（使用 --pull-request-id 选项）。
   - 源分支提示的完整提交 ID（使用 --source-commit-id 选项）。
   - 存储库的名称（使用 --repository-name 选项）。
   - 要使用的冲突详细信息级别（使用 --conflict-detail-level 选项）。如果未指定，则使用默认值 **FILE_LEVEL**。
   - 要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。如果未指定，它默认为 NONE，并且必须手动解决冲突。
   - 要包括的提交消息（使用 --commit-message 选项）。
   - 要用于提交的姓名（使用 --author-name 选项）。
   - 要用于提交的电子邮件地址（使用 --email 选项）。
   - 是否保留任何空文件夹（使用 --keep-empty-folders 选项）。

   以下示例展示了在名为 *MyDemoRepo* 的存储库中合并并关闭 ID 为 *47* 并且源提交 ID 为 *99132ab0EXAMPLE* 的拉取请求。它使用冲突详细信息和冲突解决策略的默认选项：

   ```
   aws codecommit merge-pull-request-by-three-way --pull-request-id 47 --source-
   commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo --author-name "Maria Garcia"
    --email "maria_garcia@example.com" --commit-message "Merging pull request 47 by
    three-way with default options"
   ```

如果成功，该命令生成与快进合并相同类型的输出，类似于以下内容：

```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approver-rule-for-main",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "originApprovalRuleTemplate": {
                    "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
                    "approvalRuleTemplateName": "2-approver-rule-for-main"
                },
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.142,
        "description": "Review the latest changes and updates to the global
 variables",
        "lastActivityDate": 1508887223.155,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": true,
                    "mergedBy": "arn:aws:iam::123456789012:user/Mary_Major"
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
            }
        ],
```

```
        "title": "Consolidation of global variables"
    }
}
```

解决 AWS CodeCommit 存储库中的拉取请求中的冲突

如果拉取请求存在冲突而无法合并，您可以尝试使用几种方法之一解决冲突：

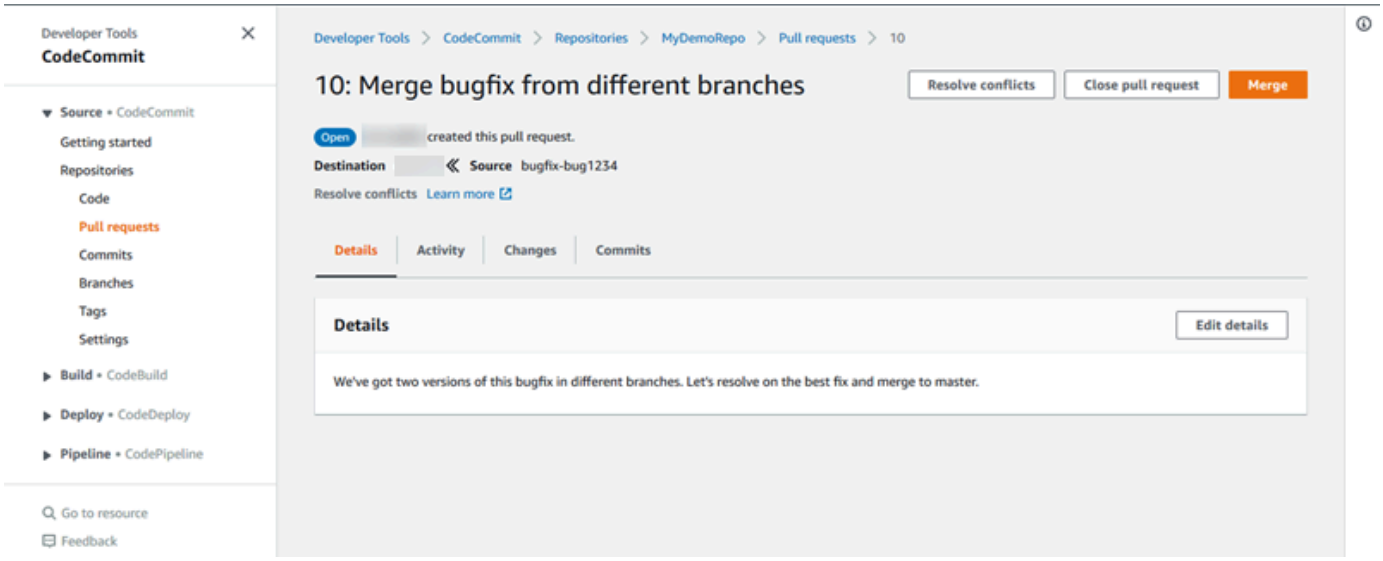
- 在本地计算机上，您可以使用 git diff 命令查找两个分支之间的冲突，并进行更改以解决这些冲突。您还可以使用差异工具或其他软件帮助查找和解决差异。在解决这些冲突并感到满意后，您可以推送具有更改（包含解决的冲突）的源分支，这会更新拉取请求。有关 git diff 和 git difftool 的更多信息，请参阅 Git 文档。
- 在控制台中，您可以选择 Resolve conflicts (解决冲突)。这会打开一个纯文本编辑器，以使用与 git diff 命令类似的方式显示冲突。您可以在包含冲突的每个文件中手动查看这些冲突，进行更改，然后使用更改更新拉取请求。
- 在 AWS CLI 中，您可以使用 AWS CLI 获取有关合并冲突的信息，并创建未引用的合并提交以测试合并。

主题

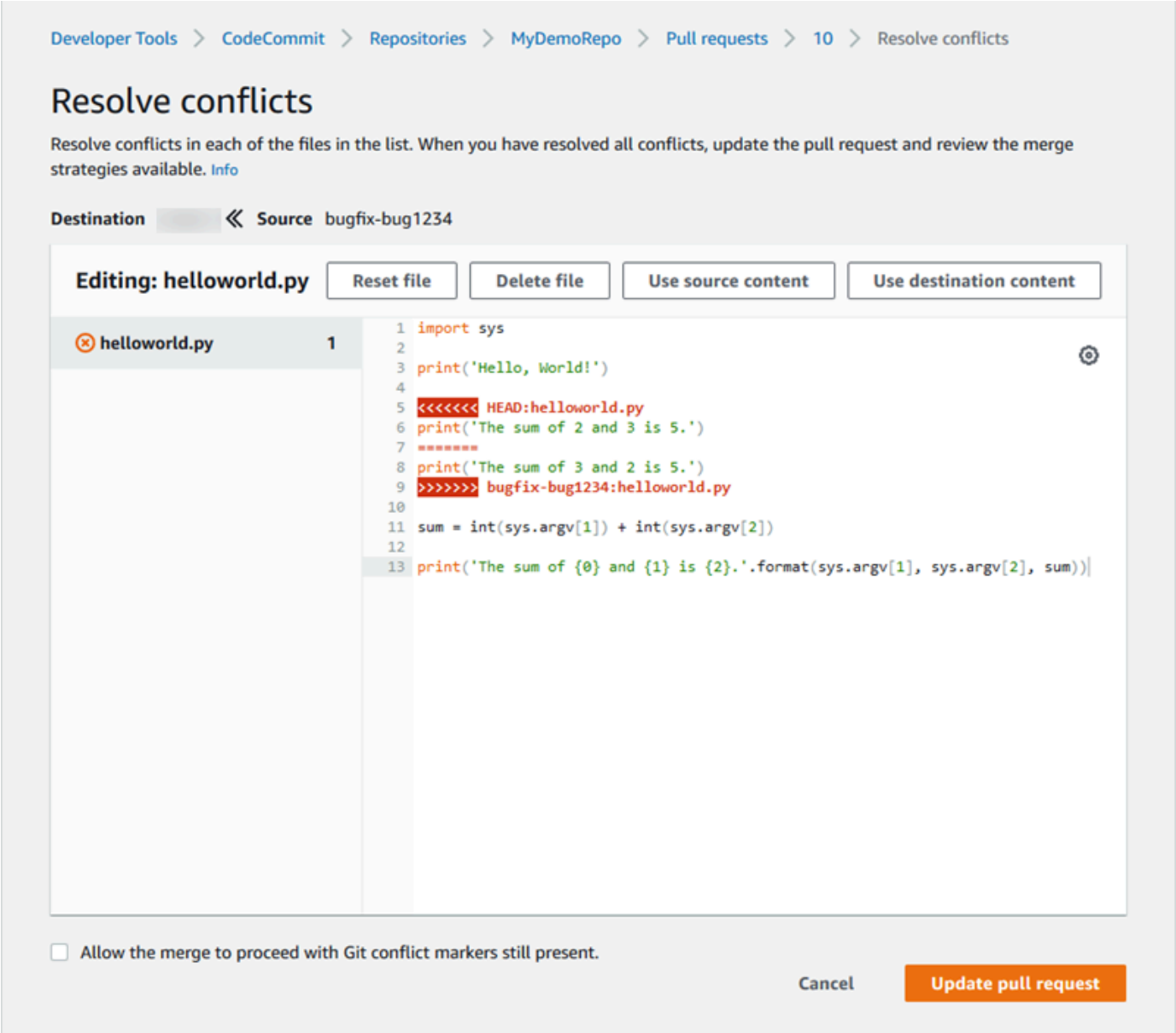
解决拉取请求中的冲突（控制台）

您可以使用 CodeCommit 控制台解决 CodeCommit 存储库中的拉取请求冲突。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。选择要合并但包含冲突的打开拉取请求。
5. 在拉取请求中，选择 Resolve conflicts (解决冲突)。只有在具有必须解决后才能合并拉取请求的冲突时，才会显示该选项。



6. 将打开冲突解决窗口，其中列出具有必须解决的冲突的每个文件。在列表中选择每个文件以查看冲突，并进行所需的任何更改，直到解决了所有冲突。



- 您可以选择使用源文件内容、目标文件内容，或者如果文件不是二进制文件，则手动编辑文件内容以使其仅包含所需的更改。标准 git diff 标记用于显示文件中的目标 (HEAD) 和源分支之间的冲突。
- 如果文件是二进制文件、Git 子模块，或者存在文件/文件夹名称冲突，您必须选择使用源文件或目标文件以解决这些冲突。您无法在 CodeCommit 控制台中查看或编辑二进制文件。
- 如果存在文件模式冲突，您将会看到在源文件的文件模式和目标文件的文件模式之间选择以解决该冲突的选项。
- 如果您决定要放弃对文件的更改并将其恢复为冲突状态，请选择 Reset file (重置文件)。这样，您就可以使用不同的方式解决冲突。

7. 如果对更改感到满意，请选择 Update pull request (更新拉取请求)。

> **Note**
>
> 您必须先解决所有文件中的所有冲突，然后才能使用更改成功更新拉取请求。

8. 将使用更改更新拉取请求，并且可以合并该请求。您将会看到合并页面。您可以选择此时合并拉取请求，也可以返回到拉取请求列表。

解决拉取请求中的冲突 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

您无法使用单个 AWS CLI 命令解决拉取请求中的冲突并合并该请求。不过，您可以使用单个命令查找冲突，尝试解决冲突，以及测试是否可以合并拉取请求。您可以使用：

- get-merge-options，用于找出可以使用哪些合并选项合并两个提交说明符。
- get-merge-conflicts，用于返回具有两个提交说明符之间的合并冲突的文件列表。
- batch-describe-merge-conflicts，用于获取有关文件中的所有合并冲突的信息，这些冲突是在使用指定的合并策略合并两个提交时产生的。
- describe-merge-conflicts，用于获取有关特定文件的合并冲突的详细信息，这些冲突是在使用指定的合并策略合并两个提交时产生的。
- create-unreferenced-merge-commit，用于测试使用指定的合并策略合并两个提交说明符的结果。

1. 要查找可用于合并两个提交说明符的合并选项，请运行 get-merge-options 命令并指定：

   - 合并源的提交说明符（使用 --source-commit-specifier 选项）。
   - 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
   - 存储库的名称（使用 --repository-name 选项）。
   - （可选）要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。
   - （可选）任何冲突的所需详细信息级别（使用 --conflict-detail-level 选项）。

   例如，要确定可用于在名为 *MyDemoRepo* 的存储库中合并名为 *bugfix-1234* 的源分支与名为 *main* 的目标分支的合并策略，请运行以下命令：

```
aws codecommit get-merge-options --source-commit-specifier bugfix-1234 --
destination-commit-specifier main --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "mergeOptions": [
        "FAST_FORWARD_MERGE",
        "SQUASH_MERGE",
        "THREE_WAY_MERGE"
    ],
    "sourceCommitId": "d49940adEXAMPLE",
    "destinationCommitId": "86958e0aEXAMPLE",
    "baseCommitId": "86958e0aEXAMPLE"
}
```

2. 要获取包含两个提交说明符的合并冲突的文件列表，请运行 get-merge-conflicts 命令并指定：

- 合并源的提交说明符（使用 --source-commit-specifier 选项）。
- 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
- 存储库的名称（使用 --repository-name 选项）。
- 要使用的合并选项（使用 --merge-option 选项）。
- （可选）任何冲突的所需详细信息级别（使用 --conflict-detail-level 选项）。
- （可选）要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。
- （可选）要返回的包含冲突的最大文件数（使用 --max-conflict-files 选项）。

例如，要获取包含在名为 MyDemoRepo 的存储库中使用三向合并策略合并名为 feature-randomizationfeature 的源分支和名为 main 的目标分支时产生的冲突的文件列表，请运行以下命令：

```
aws codecommit get-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "mergeable": false,
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE",
    "conflictMetadataList": [
        {
            "filePath": "readme.md",
            "fileSizes": {
                "source": 139,
                "destination": 230,
                "base": 85
            },
            "fileModes": {
                "source": "NORMAL",
                "destination": "NORMAL",
                "base": "NORMAL"
            },
            "objectTypes": {
                "source": "FILE",
                "destination": "FILE",
                "base": "FILE"
            },
            "numberOfConflicts": 1,
            "isBinaryFile": {
                "source": false,
                "destination": false,
                "base": false
            },
            "contentConflict": true,
            "fileModeConflict": false,
            "objectTypeConflict": false,
            "mergeOperations": {
                "source": "M",
                "destination": "M"
            }
        }
    ]
}
```

3. 要获取有关在所有文件或部分文件中包含的两个提交说明符的合并冲突的信息，请运行 batch-describe-merge-conflicts 命令并指定：

- 合并源的提交说明符（使用 --source-commit-specifier 选项）。
- 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
- 要使用的合并选项（使用 --merge-option 选项）。
- 存储库的名称（使用 --repository-name 选项）。
- （可选）要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。
- （可选）任何冲突的所需详细信息级别（使用 --conflict-detail-level 选项）。
- （可选）要返回的最大合并块数（使用 --max-merge-hunks 选项）。
- （可选）要返回的包含冲突的最大文件数（使用 --max-conflict-files 选项）。
- （可选）用于描述冲突的目标文件的路径（使用 --file-paths 选项）。

例如，要确定在名为 *MyDemoRepo* 的存储库中使用 *THREE_WAY_MERGE* 策略将名为 *feature-randomizationfeature* 的源分支与名为 *main* 的目标分支合并时产生的合并冲突，请运行以下命令：

```
aws codecommit batch-describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "conflicts": [
        {
            "conflictMetadata": {
                "filePath": "readme.md",
                "fileSizes": {
                    "source": 139,
                    "destination": 230,
                    "base": 85
                },
                "fileModes": {
                    "source": "NORMAL",
                    "destination": "NORMAL",
                    "base": "NORMAL"
                },
                "objectTypes": {
                    "source": "FILE",
```

```
                    "destination": "FILE",
                    "base": "FILE"
                },
                "numberOfConflicts": 1,
                "isBinaryFile": {
                    "source": false,
                    "destination": false,
                    "base": false
                },
                "contentConflict": true,
                "fileModeConflict": false,
                "objectTypeConflict": false,
                "mergeOperations": {
                    "source": "M",
                    "destination": "M"
                }
            },
            "mergeHunks": [
                {
                    "isConflict": true,
                    "source": {
                        "startLine": 0,
                        "endLine": 3,
                        "hunkContent": "VGhpcyBpEXAMPLE=="
                    },
                    "destination": {
                        "startLine": 0,
                        "endLine": 1,
                        "hunkContent": "VXNlIHRoEXAMPLE="
                    }
                }
            ]
        }
    ],
    "errors": [],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b6958EXAMPLE"
}
```

4. 要获取有关特定文件中包含的两个提交说明符的任何合并冲突的详细信息，请运行 describe-merge-conflicts 命令并指定：

- 合并源的提交说明符（使用 --source-commit-specifier 选项）。
- 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
- 要使用的合并选项（使用 --merge-option 选项）。
- 用于描述冲突的目标文件的路径（使用 --file-path 选项）。
- 存储库的名称（使用 --repository-name 选项）。
- （可选）要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。
- （可选）任何冲突的所需详细信息级别（使用 --conflict-detail-level 选项）。
- （可选）要返回的最大合并块数（使用 --max-merge-hunks 选项）。
- （可选）要返回的包含冲突的最大文件数（使用 --max-conflict-files 选项）。

例如，要确定在名为 *MyDemoRepo* 的存储库中使用 *THREE_WAY_MERGE* 策略将名为 *feature-randomizationfeature* 的源分支中名为 *readme.md* 的文件与名为 *main* 的目标分支合并时产生的合并冲突，请运行以下命令：

```
aws codecommit describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --file-path readme.md --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "conflictMetadata": {
        "filePath": "readme.md",
        "fileSizes": {
            "source": 139,
            "destination": 230,
            "base": 85
        },
        "fileModes": {
            "source": "NORMAL",
            "destination": "NORMAL",
            "base": "NORMAL"
        },
        "objectTypes": {
            "source": "FILE",
            "destination": "FILE",
            "base": "FILE"
```

```
        },
        "numberOfConflicts": 1,
        "isBinaryFile": {
            "source": false,
            "destination": false,
            "base": false
        },
        "contentConflict": true,
        "fileModeConflict": false,
        "objectTypeConflict": false,
        "mergeOperations": {
            "source": "M",
            "destination": "M"
        }
    },
    "mergeHunks": [
        {
            "isConflict": true,
            "source": {
                "startLine": 0,
                "endLine": 3,
                "hunkContent": "VGhpcyBpEXAMPLE=="
            },
            "destination": {
                "startLine": 0,
                "endLine": 1,
                "hunkContent": "VXNlIHRoEXAMPLE="
            }
        }
    ],
    "destinationCommitId": "86958e0aEXAMPLE",
    "sourceCommitId": "6ccd57fdEXAMPLE",
    "baseCommitId": "767b69580EXAMPLE"
}
```

5. 要创建表示两个提交说明符的合并结果的未引用提交，请运行 create-unreferenced-merge-commit 命令并指定：

   - 合并源的提交说明符（使用 --source-commit-specifier 选项）。
   - 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
   - 要使用的合并选项（使用 --merge-option 选项）。
   - 存储库的名称（使用 --repository-name 选项）。

- （可选）要使用的冲突解决策略（使用 --conflict-resolution-strategy 选项）。
- （可选）任何冲突的所需详细信息级别（使用 --conflict-detail-level 选项）。
- （可选）要包含的提交消息（使用 --commit-message 选项）。
- （可选）要用于提交的姓名（使用 --name 选项）。
- （可选）要用于提交的电子邮件地址（使用 --email 选项）。
- （可选）是否保留任何空文件夹（使用 --keep-empty-folders 选项）。

例如，要确定在名为 *MyDemoRepo* 的存储库中使用 ACCEPT_SOURCE 策略将名为 *bugfix-1234* 的源分支与名为 *main* 的目标分支合并时产生的合并冲突，请运行以下命令：

```
aws codecommit create-unreferenced-merge-commit --source-commit-
specifier bugfix-1234 --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo --name "Maria Garcia" --email
 "maria_garcia@example.com" --commit-message "Testing the results of this merge."
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

关闭 AWS CodeCommit 存储库中的拉取请求

如果要关闭拉取请求而不合并代码，您可以通过以下几种方法之一完成该操作：

- 在控制台中，可以关闭拉取请求而不合并代码。如果要使用 git merge 命令手动合并分支，或者拉取请求源分支中的代码不是要合并到目标分支中的代码，则需要执行此操作。
- 您可以删除拉取请求中指定的源分支。如果拉取请求的源分支或目标分支已删除，CodeCommit 将自动关闭拉取请求。
- 在 AWS CLI 中，您可以将拉取请求的状态从 OPEN 更新为 CLOSED。这会关闭拉取请求而不合并代码。

主题

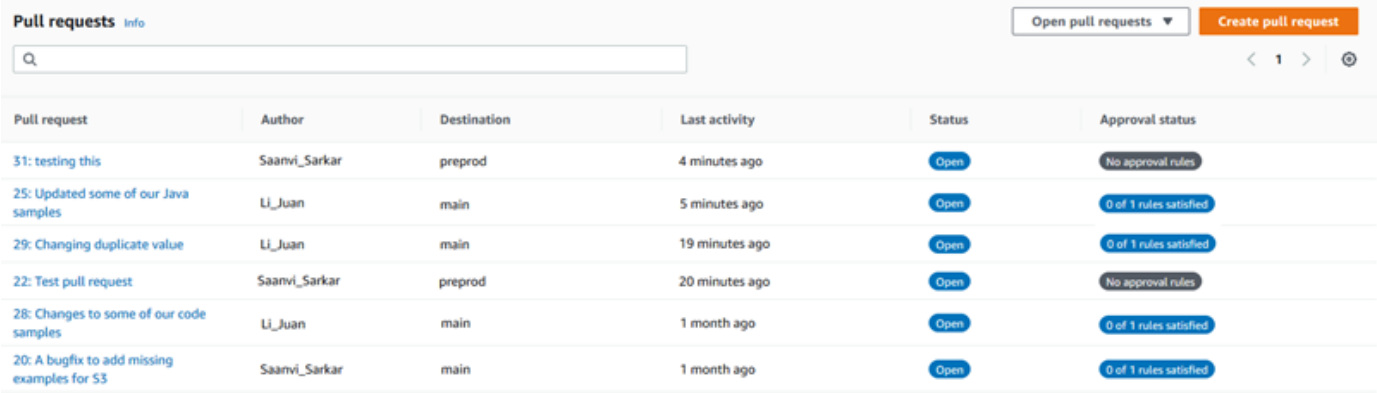
- 关闭拉取请求（控制台）

- 关闭拉取请求 (AWS CLI)

关闭拉取请求（控制台）

您可以使用 CodeCommit 控制台关闭 CodeCommit 存储库中的拉取请求。在拉取请求的状态更改为 Closed (已关闭) 后，就不能再将其更改回 Open (打开)，但用户仍然可以评论更改和回复评论。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称。
3. 在导航窗格中，选择拉取请求。
4. 默认情况下，会显示所有处于打开状态的拉取请求的列表。选择要关闭的处于打开状态的拉取请求。



5. 在拉取请求中，选择 Close pull request (关闭拉取请求)。该选项关闭拉取请求，而不尝试将源分支合并到目标分支中。此选项不提供在关闭拉取请求时删除源分支的方法，但您可以在关闭请求之后自行执行此操作。

关闭拉取请求 (AWS CLI)

要使用 AWS CLI 命令操作 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅命令行参考。

使用 AWS CLI 关闭 CodeCommit 存储库中的拉取请求

- 要将存储库中拉取请求的状态从 `OPEN` 更新为 `CLOSED`，请运行 update-pull-request-status 命令，并且指定：
  - 拉取请求的 ID（使用 --pull-request-id 选项）。
  - 拉取请求的状态（使用 --pull-request-status 选项）。

例如，要在名为 MyDemoRepo 的 CodeCommit 存储库中将 ID 为 *42* 的拉取请求的状态更新为状态 *CLOSED*，请运行以下命令：

```
aws codecommit update-pull-request-status --pull-request-id 42 --pull-request-
status CLOSED
```

如果成功，该命令产生类似以下内容的输出：

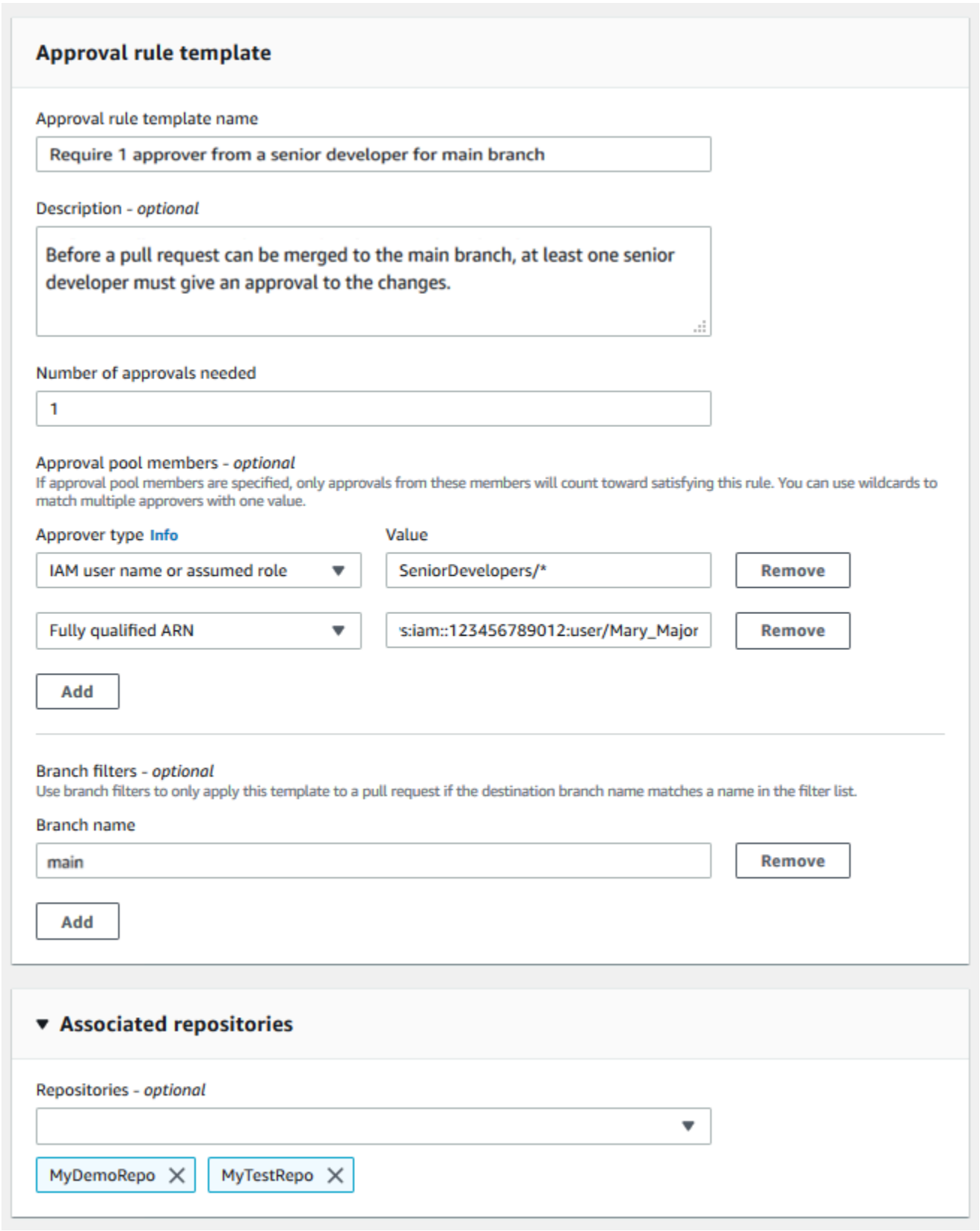
```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "approvalRuleId": "dd8b17fe-EXAMPLE",
                "approvalRuleName": "2-approvers-needed-for-this-change",
                "creationDate": 1571356106.936,
                "lastModifiedDate": 571356106.936,
                "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
                "ruleContentSha256": "4711b576EXAMPLE"
            }
        ],
        "authorArn": "arn:aws:iam::123456789012:user/Li_Juan",
        "clientRequestToken": "",
        "creationDate": 1508530823.165,
        "description": "Updated the pull request to remove unused global
 variable.",
        "lastActivityDate": 1508372423.12,
        "pullRequestId": "47",
        "pullRequestStatus": "CLOSED",
        "pullRequestTargets": [
            {
                "destinationCommit": "9f31c968EXAMPLE",
                "destinationReference": "refs/heads/main",
                "mergeMetadata": {
                    "isMerged": false,
                },
                "repositoryName": "MyDemoRepo",
                "sourceCommit": "99132ab0EXAMPLE",
                "sourceReference": "refs/heads/variables-branch"
```

```
            }
        ],
        "title": "Consolidation of global variables"
    }
}
```

使用审批规则模板

您可以为拉取请求创建审批规则。要将审批规则自动应用于在存储库中创建的部分或全部拉取请求，可使用审批规则模板。审批规则模板可帮助您自定义跨存储库的开发工作流程，以便不同分支拥有适当的审批和控制级别。您可以为生产和开发分支定义不同的规则。每次创建与规则条件匹配的拉取请求时，都会应用这些规则。有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

您可以将审批规则模板与创建时所处的 AWS 区域中的一个或多个存储库关联。模板与存储库关联后，当该存储库中创建拉取请求时，模板会为其自动创建审批规则。与单个审批规则一样，审批规则模板定义了审批规则结构，包括所需审批的数量以及必须获得其审批的可选用户池。与单个审批规则不同的是，您还可以定义目标引用（一个或多个分支），也称为分支筛选条件。如果定义了目标引用，那么仅当拉取请求的目标分支名称与模板中指定的分支名称（目标引用）匹配时，才会为其创建规则。例如，如果指定 **`refs/heads/main`** 为目标引用，则模板中定义的审批规则仅应用于目标分支为 `main` 的拉取请求。


Approval rule template
Approval rule template name
Require 1 approver from a senior developer for main branch
Description - optional
Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.
Number of approvals needed
1
Approval pool members - optional
If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.
Approver type Info
Value
IAM user name or assumed role
SeniorDevelopers/*
Remove
Fully qualified ARN
s:iam::123456789012:user/Mary_Major
Remove
Add
Branch filters - optional
Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.
Branch name
main
Remove
Add
Associated repositories
Repositories - optional
MyDemoRepo
MyTestRepo

主题

创建审批规则模板

您可以创建一个或多个审批规则模板，以帮助自定义跨存储库的开发工作流程。通过创建多个模板，您可以配置自动创建审批规则，以便不同分支拥有适当的审批和控制级别。例如，您可以为生产和开发分支创建不同模板，并将这些模板应用到一个或多个存储库。当用户在这些存储库中创建拉取请求时，将根据这些模板对请求进行评估。如果请求与所应用模板中的条件匹配，则会为该拉取请求创建审批规则。

您可以使用控制台或 AWS CLI 创建审批规则模板。有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

主题

创建审批规则模板（控制台）

默认情况下，审批规则模板不与任何存储库关联。您可以在创建模板时，在模板与一个或多个存储库之间建立关联，也可以之后再添加关联。

创建审批规则模板（控制台）

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 选择 Approval rule templates (审批规则模板)，然后选择 Create template (创建模板)。
3. 在 Approval rule template name (审批规则模板名称) 中，为模板指定一个描述性名称，以便您知道其用途。例如，如果您希望拉取请求在合并之前，必须由一组高级开发人员中的一位对其进行审批，那么可以将该规则命名为 **`Require 1 approver from a senior developer`**。
4. （可选）在 Description (描述) 中，提供有关此模板用途的描述。这可以帮助其他人决定此模板是否适合其存储库。

5. 在 Number of approvals needed (需要的审批数量) 中，输入所需的数量。默认值为 1。
6. （可选）如果您希望必须由特定用户组对拉取请求进行审批，那么可以在 Approval rule members (审批规则成员) 中，选择 Add (添加)。在 Approver type (审批人类型) 中，选择以下选项之一：

   - IAM 用户名或代入的角色：此选项会预先填入您用于登录的账户的 Amazon Web Services 账户 ID，并且只需要一个名称。它可以用于 IAM 用户，以及名称与所提供名称相匹配的联合访问用户。这是一个非常强大的选项，提供了极大的灵活性。例如，如果您选择了此选项并使用 Amazon Web Services 账户 123456789012 登录，而且指定了 **Mary_Major**，那么以下所有用户都将计为来自该用户的审批：

     - 账户中的 IAM 用户 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中标识为 Mary_Major 的联合用户 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非包含通配符 (`*Mary_Major`)，否则此选项无法识别代入 **CodeCommitReview** 角色且角色会话名称为 Mary_Major (`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`) 的某人的活动会话。您还可以显式指定角色名称 (`CodeCommitReview/Mary_Major`)。
   - 完全限定的 ARN：此选项允许您指定 IAM 用户或角色的完全限定 Amazon 资源名称 (ARN)。此选项还支持由其他 AWS 服务（如 AWS Lambda 和 AWS CodeBuild）使用的代入角色。对于代入的角色，ARN 格式应为 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`RoleName`*（适用于角色）和 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`FunctionName`*（适用于函数）。

   如果选择 IAM 用户名或代入的角色作为审批人类型，那么请在值中输入 IAM 用户或角色的名称，或者输入用户或角色的完全限定 ARN。再次选择 Add (添加) 可添加多个用户或角色，直到您已添加了其审批计入所需审批数量的所有用户或角色。

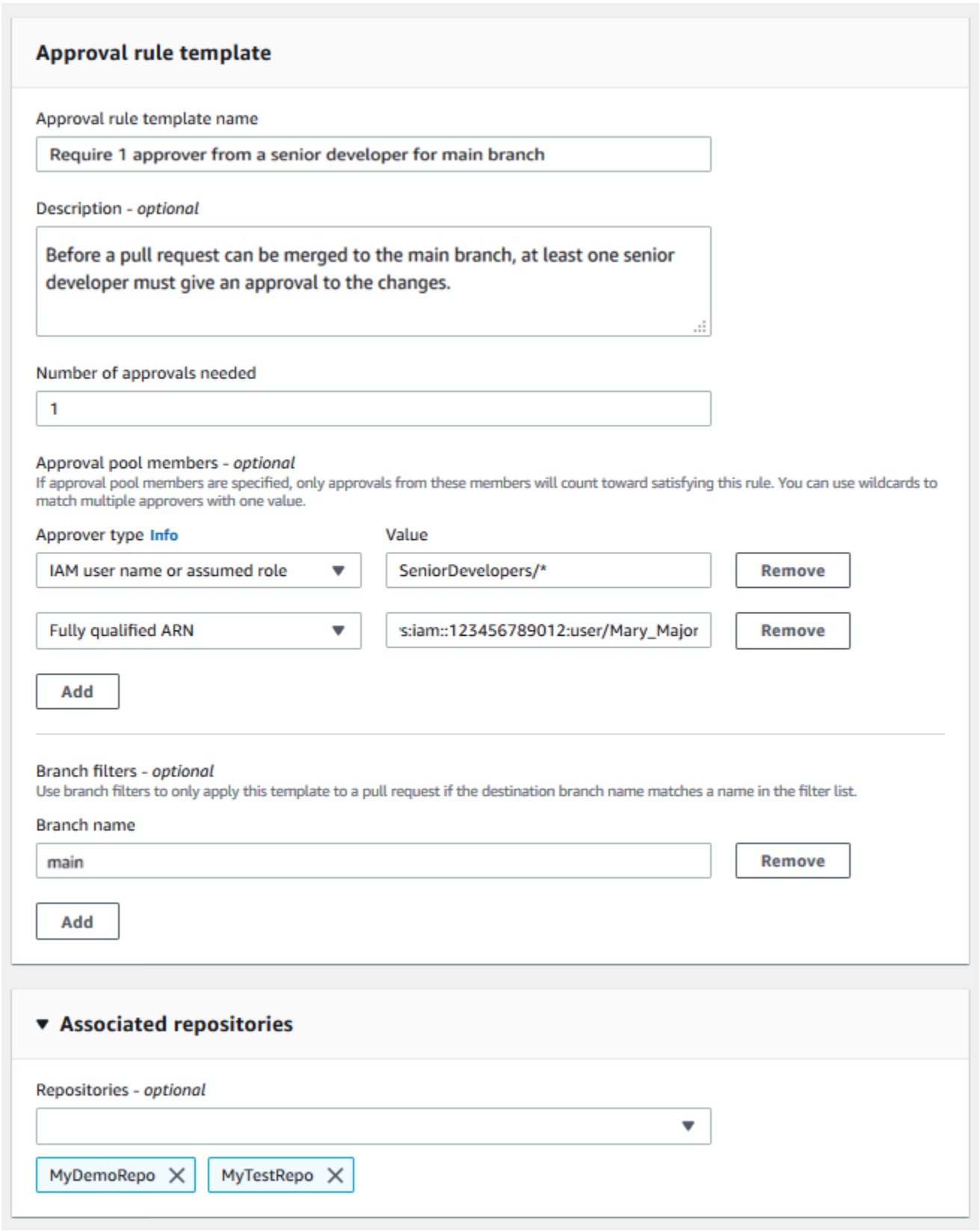
   这两种审批人类型都允许在其值中使用通配符 (*)。例如，如果选择 IAM 用户名或代入的角色选项，并且指定 **CodeCommitReview/***，那么代入 **CodeCommitReview** 角色的所有用户都将计入审批池中。他们各自的角色会话名称将计入所需的审批人数量。按照此方法，Mary_Major 和 Li_Juan 在登录并代入 `CodeCommitReview` 角色时，都计为审批。有关 IAM ARN、通配符和格式的更多信息，请参阅 IAM 标识符。

   > **Note**
   >
   > 审批规则不支持跨账户审批。

7. （可选）在 Branch filters (分支筛选条件) 中，输入要用于筛选审批规则创建的目标分支名称。例如，如果指定 `main`，那么仅当拉取请求的目标分支名为 `main` 时，才会为关联存储库中的该拉取请求创建审批规则。您可以在分支名称中使用通配符 (*)，将审批规则应用于名称与通配符匹配的所有分支。不过，通配符不能用在分支名称的开头。最多可指定 100 个分支名称。如果未指定任何筛选条件，则模板将应用于关联存储库中的所有分支。
8. （可选）在已关联的存储库中，在存储库列表中，选择要与审批规则关联的此 AWS 区域中的存储库。

> **Note**
>
> 您可以选择在创建模板后关联存储库。有关更多信息，请参阅将审批规则模板与存储库关联。

9. 选择 Create（创建）。


Approval rule template
Approval rule template name
Require 1 approver from a senior developer for main branch
Description - optional
Before a pull request can be merged to the main branch, at least one senior developer must give an approval to the changes.
Number of approvals needed
1
Approval pool members - optional
If approval pool members are specified, only approvals from these members will count toward satisfying this rule. You can use wildcards to match multiple approvers with one value.
Approver type Info
Value
IAM user name or assumed role
SeniorDevelopers/*
Remove
Fully qualified ARN
s:iam::123456789012:user/Mary_Major
Remove
Add
Branch filters - optional
Use branch filters to only apply this template to a pull request if the destination branch name matches a name in the filter list.
Branch name
main
Remove
Add
Associated repositories
Repositories - optional
MyDemoRepo
MyTestRepo

创建审批规则模板 (AWS CLI)

您可以使用 AWS CLI 创建审批规则模板。使用 AWS CLI 时，可以为模板指定目标引用，以便模板仅适用于其目标分支与模板中的目标分支匹配的拉取请求。

创建审批规则模板 (AWS CLI)

1. 在终端或命令行中，运行 create-approval-rule-template 命令，并指定：

   - 审批规则模板的名称。请考虑使用可描述其用途的名称。
   - 审批规则模板的描述。与名称一样，请考虑提供详细说明。
   - 审批规则模板的 JSON 结构。此结构可包括对目标引用的要求，即要对其应用审批规则的拉取请求的目标分支，以及审批池成员，即其审批将计入所需审批数量的用户。

   创建审批规则的内容时，可通过以下两种方式之一指定审批池中的审批人：

   - CodeCommitApprovers：此选项仅需要 Amazon Web Services 账户和资源。它可以用于 IAM 用户，以及名称与所提供资源名称相匹配的联合访问用户。这是一个非常强大的选项，提供了极大的灵活性。例如，如果指定 AWS 账户 123456789012 和 **Mary_Major**，那么以下所有用户都将计为来自该用户的审批：
     - 账户中的 IAM 用户 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中标识为 Mary_Major 的联合用户 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非包含通配符 (`*Mary_Major`)，否则此选项无法识别代入 *SeniorDevelopers* 角色且角色会话名称为 *Mary_Major* (`arn:aws:sts::123456789012:assumed-role/`*SeniorDevelopers*`/`*Mary_Major*) 的人员的活动会话。
   - 完全限定的 ARN：此选项允许您指定 IAM 用户或角色的完全限定 Amazon 资源名称 (ARN)。

   有关 IAM ARN、通配符和格式的更多信息，请参阅 IAM 标识符。

   以下示例创建名为 **2-approver-rule-for-main** 的审批规则模板，且其描述为 **Requires two developers from the team to approve the pull request if the destination branch is main**。模板需要两个代入 **CodeCommitReview** 角色的用户，在任何拉取请求合并到 **main** 分支之前对其进行审批：

```
aws codecommit create-approval-rule-template --approval-rule-template-name 2-
approver-rule-for-main --approval-rule-template-description "Requires two
 developers from the team to approve the pull request if the destination branch
 is main" --approval-rule-template-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "2-approver-rule-for-main",
        "creationDate": 1571356106.936,
        "approvalRuleTemplateId": "dd8b17fe-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
        "approvalRuleTemplateDescription": "Requires two developers from the team
 to approve the pull request if the destination branch is main",
        "lastModifiedDate": 1571356106.936,
        "ruleContentSha256": "4711b576EXAMPLE"
    }
}
```

将审批规则模板与存储库关联

审批规则模板是在特定 AWS 区域中创建的，但在与存储库关联之前，不会对该 AWS 区域中的任何存储库造成影响。要将模板应用到一个或多个存储库，必须将该模板与一个或多个存储库关联。您可以将单个模板应用于 AWS 区域中的多个存储库。这可帮助您通过创建一致的条件以审批及合并拉取请求，从而实现存储库中开发工作流程的自动化和标准化。

您只能将审批规则模板与创建该模板时所在的 AWS 区域中的存储库相关联。

有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

主题

关联审批规则模板（控制台）

您可能在创建审批规则模板时即关联了存储库。（此为可选步骤。）您可以通过编辑模板来添加或删除关联。

将审批规则模板与存储库关联

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 选择 Approval rule templates (审批规则模板)。选择模板，然后选择 Edit (编辑)。
3. 在 Associated Repositories (关联存储库) 中，从 Repositories (存储库) 列表中选择存储库。每个关联存储库均显示在列表框下。
4. 选择 Save（保存）。审批规则现在将应用于在这些关联存储库中创建的任何拉取请求。

关联审批规则模板 (AWS CLI)

您可以使用 AWS CLI 将审批规则模板与一个或多个存储库关联。

将模板与单个存储库关联

1. 在终端或命令行中，运行 associate-approval-rule-template-with-repository 命令，并指定：

   - 要与存储库关联的审批规则模板的名称。
   - 要与审批规则模板关联的存储库的名称。

   例如，要将名为 *2-approver-rule-for-main* 的审批规则模板与名为 *MyDemoRepo* 的存储库关联，请运行以下命令：

   ```
   aws codecommit associate-approval-rule-template-with-repository --repository-
   name MyDemoRepo --approval-rule-template-name 2-approver-rule-for-main
   ```

2. 如果成功，该命令不返回任何内容。

将模板与多个存储库关联

1. 在终端或命令行中，运行 batch-associate-approval-rule-template-with-repositories 命令，并指定：

   - 要与存储库关联的审批规则模板的名称。
   - 要与审批规则模板关联的存储库的名称。

   例如，要将名为 **2-approver-rule-for-main** 的审批规则模板与名为 **MyDemoRepo** 和 **MyOtherDemoRepo** 的存储库关联：

   ```
   aws codecommit batch-associate-approval-rule-template-with-repositories --
   repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 2-
   approver-rule-for-main
   ```

2. 如果成功，该命令返回类似以下内容的输出：

   ```
   {
       "associatedRepositoryNames": [
           "MyDemoRepo",
           "MyOtherDemoRepo"
       ],
       "errors": []
   }
   ```

管理审批规则模板

您可以管理 AWS 区域中的审批规则模板，以帮助了解其使用方式和用途。例如，您可以编辑审批规则模板的名称和描述，以帮助其他人了解其用途。您可以列出 AWS 区域中的所有审批规则模板，并获取有关模板内容和结构的信息。您可以查看哪些模板与存储库关联，哪些存储库与模板关联。

有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

管理审批规则模板（控制台）

您可以在 CodeCommit 控制台中查看和管理审批规则模板。

管理审批规则模板

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 选择审批规则模板，以查看您登录的 AWS 区域中的审批规则模板列表。

> **Note**
>
> 审批规则模板仅在创建时所在的 AWS 区域中可用。

3. 如果要对模板进行更改，请从列表中选择该模板，然后选择 Edit (编辑)。
4. 进行更改，然后选择 Save。

管理审批规则模板 (AWS CLI)

您可以使用以下 AWS CLI 命令管理审批规则模板：

- list-approval-rule-templates，用于查看 AWS 区域中所有审批规则模板的列表
- get-approval-rule-template，用于查看审批规则模板的内容
- update-approval-rule-template-content，用于更改审批规则模板的内容
- update-approval-rule-template-name，用于更改审批规则模板的名称
- update-approval-rule-template-description，用于更改审批规则模板的描述
- list-repositories-for-approval-rule-template，用于查看与审批规则模板关联的所有存储库
- list-associated-approval-rule-templates-for-repository，用于查看与存储库关联的所有审批规则模板

列出 AWS 区域中的所有审批规则模板

1. 在终端或命令行中，运行 list-approval-rule-templates 命令。例如，要列出美国东部（俄亥俄州）区域中的所有审批规则模板，请运行以下命令：

```
aws codecommit list-approval-rule-templates --region us-east-2
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
```

```
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

获取审批规则模板的内容

1. 在终端或命令行中，运行 get-approval-rule-template 命令，并指定审批规则模板的名称：

```
aws codecommit get-approval-rule-template --approval-rule-template-name 1-approver-
rule-for-all-pull-requests
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "ruleContentSha256": "621181bbEXAMPLE",
        "lastModifiedDate": 1571356106.936,
        "creationDate": 1571356106.936,
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan",
        "approvalRuleTemplateId": "a29abb15-EXAMPLE",
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team."
    }
}
```

更新审批规则模板的内容

1. 在终端或命令提示符处，运行 update-approval-rule-template-content 命令，并指定模板的名称和更改的内容。例如，要更改名为 **1-approver-rule** 的审批规则模板的内容，以将审批池重新定义为代入 **CodeCommitReview** 角色的用户，请运行以下命令：

```
aws codecommit update-approval-rule-template-content --approval-rule-template-
name 1-approver-rule --new-rule-content "{\"Version\": \"2018-11-08\",
\"DestinationReferences\": [\"refs/heads/main\"],\"Statements\": [{\"Type
\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers\":
 [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}"
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

更新审批规则模板的名称

1. 在终端或命令提示符处，运行 update-approval-rule-template-name 命令，并指定当前名称和更改后的名称。例如，要将审批规则模板的名称从 **1-approver-rule** 更改为 **1-approver-rule-for-all-pull-requests**，请运行以下命令：

```
aws codecommit update-approval-rule-template-name --old-approval-rule-template-name
 "1-approver-rule" --new-approval-rule-template-name "1-approver-rule-for-all-pull-
requests"
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplate": {
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "lastModifiedDate": 1571358241.619,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "creationDate": 1571352720.773,
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Mary_Major",
```

```
        "approvalRuleTemplateDescription": "All pull requests must be approved by
 one developer on the team.",
        "ruleContentSha256": "2f6c21a5cEXAMPLE"
    }
}
```

更新审批规则模板的描述

1. 在终端或命令行中，运行 update-approval-rule-template-description 命令，并指定审批规则模板的名称和新描述：

```
aws codecommit update-approval-rule-template-description --approval-rule-template-
name "1-approver-rule-for-all-pull-requests" --approval-rule-template-description
 "Requires 1 approval for all pull requests from the CodeCommitReview pool"
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "approvalRuleTemplate": {
        "creationDate": 1571352720.773,
        "approvalRuleTemplateDescription": "Requires 1 approval for all pull
 requests from the CodeCommitReview pool",
        "lastModifiedDate": 1571358728.41,
        "approvalRuleTemplateId": "41de97b7-EXAMPLE",
        "approvalRuleTemplateContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 1,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
        "approvalRuleTemplateName": "1-approver-rule-for-all-pull-requests",
        "ruleContentSha256": "2f6c21a5EXAMPLE",
        "lastModifiedUser": "arn:aws:iam::123456789012:user/Li_Juan"
    }
}
```

列出与模板关联的所有存储库

1. 在命令行或终端上，运行 list-repositories-for-approval-rule-template 命令，并指定模板的名称：

```
aws codecommit list-repositories-for-approval-rule-template --approval-rule-
template-name 2-approver-rule-for-main
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "repositoryNames": [
        "MyDemoRepo",
        "MyClonedRepo"
    ]
}
```

列出与存储库关联的所有模板

1. 在命令行或终端上，运行 list-associated-approval-rule-templates-for-repository 命令，并指定存储库的名称：

```
aws codecommit list-associated-approval-rule-templates-for-repository --repository-name MyDemoRepo
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

取消关联审批规则模板

如果审批规则模板生成的审批规则对您团队在存储库中的工作流程不再有意义，您可以取消该模板与该存储库的关联。取消关联模板不会删除在将模板与存储库关联时所创建的任何审批规则。

有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

取消关联审批规则模板（控制台）

您可以使用控制台删除存储库与审批规则模板之间的关联。

取消审批规则模板与存储库的关联

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 选择 Approval rule templates (审批规则模板)。选择要与一个或多个存储库取消关联的模板，然后选择 Edit (编辑)。
3. 在 Associated repositories (关联存储库) 中，选中要取消关联的存储库旁的 X。存储库名称将不再显示。
4. 选择 Save（保存）。审批规则将不再适用于在这些存储库中创建的拉取请求。这些规则仍适用于关联存在时发出的拉取请求。

取消关联审批规则模板 (AWS CLI)

您可以使用 AWS CLI 取消一个或多个存储库与审批规则模板的关联。

取消审批规则模板与存储库的关联

1. 在终端或命令行中，运行 disassociate-approval-rule-template-from-repository 命令，并指定：
   - 审批规则模板的名称。
   - 存储库的名称。

   例如，要取消名为 **1-approver-rule-for-all-pull-requests** 的审批规则模板与名为 **MyDemoRepo** 的存储库的关联，请运行以下命令：

   ```
   aws codecommit disassociate-approval-rule-template-from-repository --repository-name MyDemoRepo --approval-rule-template-name 1-approver-rule-for-all-pull-requests
   ```
2. 如果成功，该命令不返回任何内容。

取消审批规则模板与多个存储库的关联

1. 在终端或命令行中，运行 batch-disassociate-approval-rule-template-from-repositories 命令，并指定：
   - 审批规则模板的名称。
   - 存储库的名称。

例如，要取消名为 **1-approver-rule-for-all-pull-requests** 的审批规则模板与名为 **MyDemoRepo** 的存储库和名为 **MyOtherDemoRepo** 的存储库的关联，请运行以下命令：

```
aws codecommit batch-disassociate-approval-rule-template-from-repositories --
repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 1-
approver-rule-for-all-pull-requests
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
    "disassociatedRepositoryNames": [
        "MyDemoRepo",
        "MyOtherDemoRepo"
    ],
    "errors": []
}
```

删除审批规则模板

如果您未在任何存储库中使用审批规则模板，则可删除该模板。删除未使用的审批规则模板有助于保持模板井然有序，并且更轻松地查找对工作流程有意义的模板。

有关托管策略和审批规则模板权限的更多信息，请参阅针对审批规则模板的操作所需的权限和适用于 CodeCommit 的 AWS 托管式策略。

主题

- 删除审批规则模板（控制台）
- 删除审批规则模板 (AWS CLI)

删除审批规则模板（控制台）

如果审批规则模板与您的开发工作不再相关，则可删除该模板。在使用控制台删除审批规则模板时，将在删除过程中取消该模板与任何存储库的关联。

删除审批规则模板

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。

2. 选择 Approval rule templates (审批规则模板)。选择要删除的模板，然后选择 Delete (删除)。

删除审批规则模板 (AWS CLI)

如果审批规则已与所有存储库取消关联，则可使用 AWS CLI 删除该规则。有关更多信息，请参阅取消关联审批规则模板 (AWS CLI)。

删除审批规则模板

1. 在终端或命令行中，运行 delete-approval-rule-template 命令，并指定要删除的审批规则模板的名称：

```
aws codecommit delete-approval-rule-template --approval-rule-template-name 1-approver-for-all-pull-requests
```

2. 如果成功，该命令返回类似以下内容的输出。如果该审批规则模板已经删除，则此命令不会返回任何内容。

```
{
    "approvalRuleTemplateId": "41de97b7-EXAMPLE"
}
```

处理 AWS CodeCommit 仓库中的提交

提交是存储库内容以及对该内容进行的更改的快照。每次用户提交和推送更改时，系统都会保存和存储该信息。同样保存和存储的信息还包括：提交更改的用户、提交的日期和时间以及作为提交的一部分所做的更改。您还可以向提交添加标签，以方便识别特定的提交。在中 CodeCommit，你可以：

- 审查提交。
- 查看图表中提交的历史记录。
- 将某个提交与其父级或另一个说明符进行比较。
- 向您的提交中添加评论并回复他人发表的评论。

必须先将本地计算机设置为连接到 CodeCommit 存储库，然后才能将提交推送到存储库。有关最简单的方法的说明，请参阅适用于使用 Git 凭证的 HTTPS 用户。

有关在中使用存储库其他方面的信息 CodeCommit，请参阅使用存储库使用文件、使用拉取请求、使用分支、和使用用户首选项。

主题

- 在中创建提交 AWS CodeCommit
- 在中查看提交详情 AWS CodeCommit
- 比较中的提交 AWS CodeCommit

- 评论中的提交 AWS CodeCommit
- 在中创建 Git 标签 AWS CodeCommit
- 在中查看 Git 标签的详细信息 AWS CodeCommit
- 删除中的 Git 标签 AWS CodeCommit

在中创建提交 AWS CodeCommit

在为新存储库创建第一个提交时，使用 AWS CLI 和put-file命令。这将创建第一个提交，并让您可以为新存储库创建和指定默认分支。您可以使用 Git 或 AWS CLI 在 CodeCommit 仓库中创建提交。如果本地存储库已连接到 CodeCommit 存储库，则可以使用 Git 将提交从本地存储库推送到存储库。CodeCommit 要直接在 CodeCommit 控制台中创建提交，请参阅在 AWS CodeCommit 存储库中创建或添加文件和编辑 AWS CodeCommit 存储库中文件的内容。

> **Note**
>
> 作为最佳实践，我们建议您使用支持的最新版本的 AWS CLI、Git 和其他软件。如果您使用 AWS CLI，请确保安装了最新版本，以确保您使用的版本包含该`create-commit`命令。

主题

- 使用创建仓库的第一个提交 AWS CLI
- 使用 Git 客户端创建提交
- 使用创建提交 AWS CLI

使用创建仓库的第一个提交 AWS CLI

您可以使用 AWS CLI 和`put-file`命令为存储库创建您的第一个提交。使用 put-file 会创建第一个提交，该分支会将文件添加到空存储库，并创建一个具有您指定的名称的分支。它会将新分支指定为存储库的默认分支。

> **Note**
>
> 要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

要为仓库创建第一个提交，请使用 AWS CLI

1. 在本地计算机上，创建要作为第一个文件添加到 CodeCommit存储库的文件。一个常见做法是创建一个 README.md markdown 文件，向其他存储库用户说明此存储库的用途。如果您包含README.md文件，则该文件的内容将自动显示在 CodeCommit控制台中仓库代码页面的底部。
2. 在终端或命令行中，运行 put-file 命令，并指定：

   - 要将第一个文件添加到的存储库的名称。
   - 要创建为默认分支的分支的名称。
   - 文件的本地位置。用于此位置的语法取决于您的本地操作系统。
   - 要添加的文件的名称，包括更新的文件在存储库中的存储路径。
   - 您希望与此文件关联的用户名和电子邮件。
   - 一条提交消息，说明您为什么添加此文件。

   用户名、电子邮件地址和提交消息是可选的，但可帮助其他用户了解执行更改的人员以及原因。如果您不提供用户名，则 CodeCommit 默认使用您的 IAM 用户名或控制台登录名的派生形式作为作者姓名。

   例如，要将名为 *README.md*、内容为“Welcome to our team repository!”的文件添加到 到名为 develop *MyDemoRepo*ment 的分支命名的*##*库：

   ```
   aws codecommit put-file --repository-name MyDemoRepo --branch-name development --
   file-path README.md --file-content "Welcome to our team repository!" --name "Mary
    Major" --email "mary_major@example.com" --commit-message "I added a quick readme
    for our new team repository."
   ```

   如果成功，该命令返回类似以下内容的输出：

   ```
   {
       "commitId": "724caa36EXAMPLE",
       "blobId": "a8a94062EXAMPLE",
       "treeId": "08b2fc73EXAMPLE"
   }
   ```

使用 Git 客户端创建提交

您可以使用安装在本地计算机上的 Git 客户端创建提交，然后将这些提交推送到 CodeCommit 仓库。

1. 完成前提条件，包括设置。

   > **Important**
   >
   > 如果您尚未完成设置，则无法使用 Git 连接或提交到存储库。

2. 确保将在正确的分支中创建提交。要查看可用分支的列表并确认您当前设置使用的分支，请运行 git branch。这会显示所有分支。当前分支旁边会显示星号 (*)。要切换到其他分支，请运行 git checkout ***branch-name***。如果这是您的第一个提交，请运行 git config 命令，将 Git 客户端配置为使用您要用于该分支的名称创建初始分支。例如，如果您希望默认分支的名称为 *development*：

   ```
   git config --local init.defaultBranch development
   ```

   > **Tip**
   >
   > 此命令仅在 Git v.2.28 及更高版本中可用。
   > 您也可以运行以下命令，将所有新创建的存储库的默认分支名称设置为 **development**：
   >
   > ```
   > git config --global init.defaultBranch development
   > ```

3. 对分支做出更改 (例如，添加、修改或删除文件)。

   例如，在本地存储库中创建名为 `bird.txt` 的文件并写入以下文本：

   ```
   bird.txt
   --------
   Birds (class Aves or clade Avialae) are feathered, winged, two-legged, warm-
   blooded, egg-laying vertebrates.
   ```

4. 运行 git status，这应会指示 `bird.txt` 尚未包含在任何待处理提交中：

   ```
   ...
   Untracked files:
     (use "git add <file>..." to include in what will be committed)

           bird.txt
   ```

5. 运行 git add bird.txt 将新文件包含在待处理提交中。

6. 如果您再次运行 git status，应显示与以下内容类似的输出。它指示 `bird.txt` 现已加入到待处理提交中，或已暂存并等待提交：

```
...
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   bird.txt
```

7. 要完成提交，请运行 git commit 并指定 `-m` 选项（例如， git commit -m "***Adding bird.txt to the repository.***"）。`-m` 选项用于创建提交消息。
8. 如果您再次运行 git status，应显示与以下内容类似的输出。它表示提交已准备好从本地存储库推送到 CodeCommit 存储库：

```
...
nothing to commit, working directory clean
```

9. 在将已完成的提交从本地存储库推送到 CodeCommit 存储库之前，您可以通过运行来查看正在推送的内容git diff --stat ***remote-name*/*branch-name***，其中 rem *ote-name* 是本地 CodeCommit 存储库使用的昵称，b *ranch-name* *#######*的名称。

> **Tip**
>
> 要获取别名，请运行 git remote。要获取分支名称列表，请运行 git branch。当前分支旁边会显示星号 (*)。您也可以运行 git status 来获取当前分支的名称。

> **Note**
>
> 如果您克隆了存储库，则从本地存储库的角度来看，re *mote-name* 不是存储库的名称。CodeCommit 在克隆存储库时，*remote-name* 会自动设为 `origin`。

例如，git diff --stat origin/main 将显示类似以下内容的输出：

```
bird.txt | 1 +
1 file changed, 1 insertion(+)
```

输出假设您已经将本地存储库连接到 CodeCommit 存储库。（有关说明，请参阅连接存储库。）

10. 当你准备好将提交从本地存储库推送到 CodeCommit 存储库时，运行git push ***remote-name branch-name***，其中 rem *ote-name* 是本地存储库使用的昵称，b *ranch-name* 是要推送到 CodeCommit 存储库的分支的名称。 CodeCommit

    例如，运行 git push origin main 将显示类似以下内容的输出：

    对于 HTTPS：

    ```
    Counting objects: 7, done.
    Delta compression using up to 4 threads.
    Compressing objects: 100% (4/4), done.
    Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
    Total 5 (delta 2), reused 0 (delta 0)
    remote:
    To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
        b9e7aa6..3dbf4dd main -> main
    ```

    对于 SSH：

    ```
    Counting objects: 7, done.
    Delta compression using up to 4 threads.
    Compressing objects: 100% (4/4), done.
    Writing objects: 100% (5/5), 516 bytes | 0 bytes/s, done.
    Total 5 (delta 2), reused 0 (delta 0)
    remote:
    To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo
        b9e7aa6..3dbf4dd main -> main
    ```

    > **Tip**
    >
    > 如果向 git push 添加 -u 选项（例如，git push -u origin main），则以后只需运行 git push，因为已设置了上游跟踪信息。要获取上游跟踪信息，请运行 git remote show ***remote-name***（例如，git remote show origin）。

有关更多选项，请参阅 Git 文档。

使用创建提交 AWS CLI

您可以使用 AWS CLI 和`create-commit`命令在指定分支的尖端为存储库创建提交。您还可以创建未引用的合并提交来表示合并两个提交说明符的结果。有关更多信息，请参阅创建非引用提交。

> **Note**
>
> 要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

创建提交

1. 在本地计算机上，进行要提交到 CodeCommit存储库的更改。
2. 在终端或命令行中，运行 create-commit 命令，并指定：
   - 要将更改提交到的存储库。
   - 要将更改提交到的分支。
   - 对该分支进行的最新提交的完整提交 ID，也称为顶端或头提交或父提交 ID。
   - 如果所做的更改删除了这些文件夹的内容，是否保留任何空文件夹。默认情况下，此值为 false。
   - 有关要添加、更改或删除的文件的信息。
   - 希望与这些更改关联的用户名和电子邮件。
   - 一条提交消息，说明您进行这些更改的原因。

   用户名、电子邮件地址和提交消息是可选的，但可帮助其他用户知道更改的执行者及原因。如果您不提供用户名，则 CodeCommit 默认使用您的 IAM 用户名或控制台登录名的派生形式作为作者姓名。

   例如，为存储库创建提交，将README.md文件添加到*#*分支*MyDemoRepo*中名为的存储库中。该文件采用 Base64 格式，内容为“Welcome to our team repository!”：

```
aws codecommit create-commit --repository-name MyDemoRepo --
branch-name main --parent-commit-id 4c925148EXAMPLE --put-files
 "filePath=README.md,fileContent=V2VsY29tZSB0byBvdXIgdGVhbSByZXBvc2l0b3J5IQo="
```

> **Tip**
>
> 要获取父提交 ID，请运行 get-branch 命令。

如果成功，该命令返回类似以下内容的输出：

```
{
    "commitId": "4df8b524-EXAMPLE",
    "treeId": "55b57003-EXAMPLE",
    "filesAdded": [
        {
            "blobId": "5e1c309dEXAMPLE",
            "absolutePath": "meeting.md",
            "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [],
    "filesUpdated": []
}
```

file1.py # file2.txt ################ picture.png #### image1.png######## p ic tures ######## im ag es #########.py ######## # ExampleSolution.py ####MyFeatureBranch#####MyDemoRepo########## ID # 4c925148Example#

```
aws codecommit create-commit --repository-name MyDemoRepo --branch-
name MyFeatureBranch --parent-commit-id 4c925148EXAMPLE --name "Saanvi Sarkar"
 --email "saanvi_sarkar@example.com" --commit-message "I'm creating this commit to
 update a variable name in a number of files."
 --keep-empty-folders false  --put-files '{"filePath": "file1.py", "fileMode":
 "EXECUTABLE", "fileContent": "bucket_name = sys.argv[1] region = sys.argv[2]"}'
'{"filePath": "file2.txt", "fileMode": "NORMAL", "fileContent": "//Adding a comment
 to explain the variable changes in file1.py"}' '{"filePath": "images/image1.png",
"fileMode": "NORMAL", "sourceFile": {"filePath": "pictures/picture.png", "isMove":
 true}}' --delete-files filePath="ExampleSolution.py"
```

> **Note**
>
> 根据您的操作系统，--put-files 段的语法略有不同。以上示例针对 Linux、macOS 或 Unix 用户和具有 Bash 仿真器的 Windows 用户进行了优化。使用命令行或 Powershell 的 Windows 用户应使用适合这些系统的语法。

如果成功，该命令返回类似以下内容的输出：

```
{
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE",
   "filesAdded": [
        {
        "absolutePath": "images/image1.png",
        "blobId": "d68ba6ccEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesUpdated": [
        {
        "absolutePath": "file1.py",
        "blobId": "0a4d55a8EXAMPLE",
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "file2.txt",
        "blobId": "915766bbEXAMPLE",
        "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [
        {
        "absolutePath": "ExampleSolution.py",
        "blobId": "4f9cebe6aEXAMPLE",
        "fileMode": "EXECUTABLE"
        },
        {
        "absolutePath": "pictures/picture.png",
        "blobId": "fb12a539EXAMPLE",
        "fileMode": "NORMAL"
```

```
        }
    ]
}
```

在中查看提交详情 AWS CodeCommit

您可以使用 AWS CodeCommit 控制台浏览仓库中的提交历史记录。这可以帮助您确定在存储库中进行的更改，包括：

- 做出更改的时间和更改者。
- 特定提交合并到一个分支的时间。

查看分支的提交历史记录也许可帮助您了解分支之间的差异。如果您使用了标记，还可以快速查看带有标签的提交及该提交的父级。在命令行中，您可以使用 Git 查看有关本地存储库或仓 CodeCommit 库中提交的详细信息。

浏览存储库中的提交

您可以使用 AWS CodeCommit 控制台浏览仓库的提交历史记录。您还可以按时间顺序查看存储库及其分支中提交的图表。这可以帮助您了解存储库的历史记录，包括做出更改的时间。

> **Note**
>
> 使用 git rebase 命令对存储库执行变基操作会更改存储库的历史记录，这可能会导致提交出现乱序。有关更多信息，请参阅 Git Branching-Rebasing 或 Git 文档。

主题

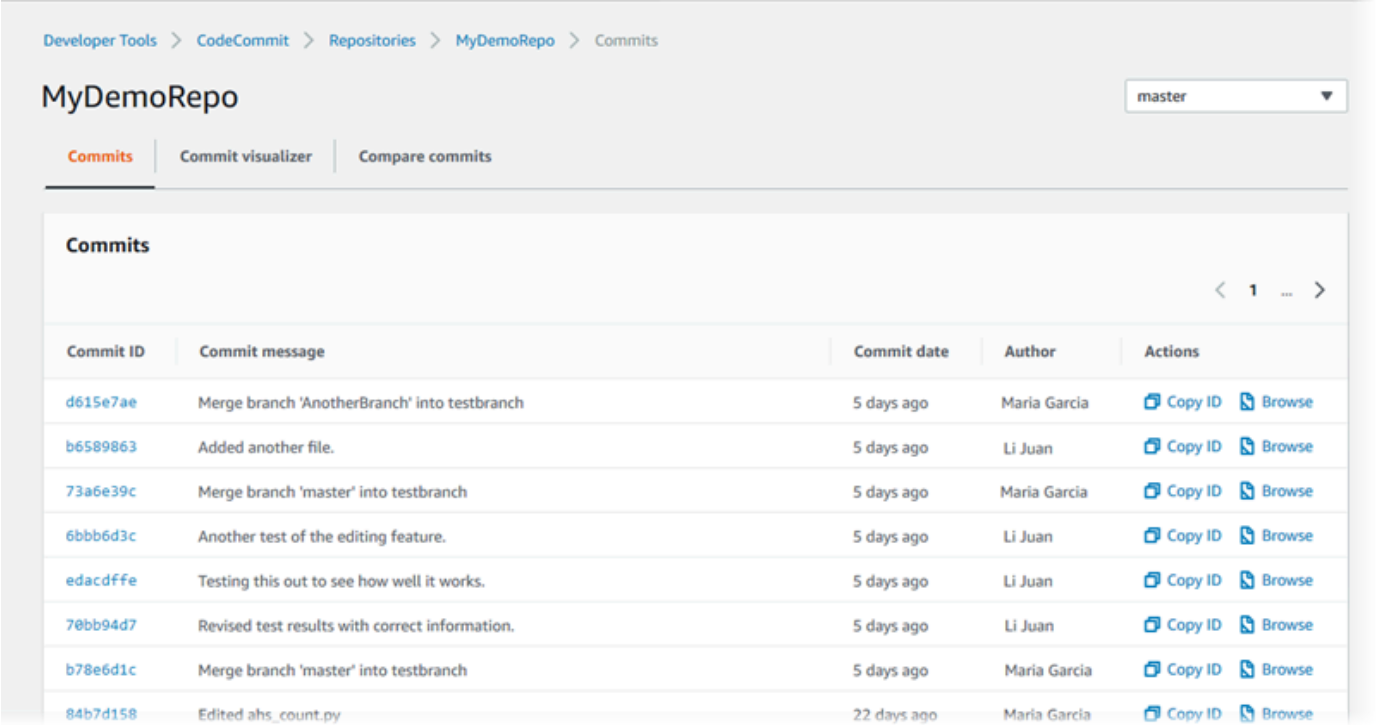
- 浏览存储库的提交历史记录
- 查看存储库的提交历史记录图表

浏览存储库的提交历史记录

您可以浏览存储库特定分支或标签的提交历史记录，包括有关提交者和提交消息的信息。您也可以查看提交的代码。

浏览提交历史记录

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库)中，选择要查看其提交历史记录的存储库。
3. 在导航窗格中，选择 Commits。在提交历史记录视图中，系统按提交日期的反向时间顺序显示存储库默认分支中的提交的历史记录。日期和时间用协调世界时 (UTC) 表示。您可以按照如下方式查看不同分支的提交历史记录：选择视图选择器按钮，然后从列表中选择分支。如果您在存储库中使用了标签，则可通过在视图选择器按钮中选择特定标签来查看具有该标签的提交及其父级。



4. 要查看提交与其父项之间的差异并查看对更改的评论，请选择缩写的提交 ID。有关更多信息，请参阅 比较提交与其父级 和 评论提交。要查看提交及分支、标签或提交 ID 等任何其他提交说明符之间的差异，请参阅比较任意两个提交说明符。
5. 执行以下一个或多个操作：

   - 要查看进行更改的日期和时间，请将鼠标指针悬停在提交日期上方。
   - 要查看完整的提交 ID，请复制然后将其粘贴到文本编辑器或其他位置。要复制它，请选择 Copy ID (复制 ID)。
   - 要查看提交时的代码，请选择 Browse (浏览)。Code 视图中显示做出该提交时存储库中的内容。视图选择器按钮显示缩写的提交 ID，而不是分支或标签。

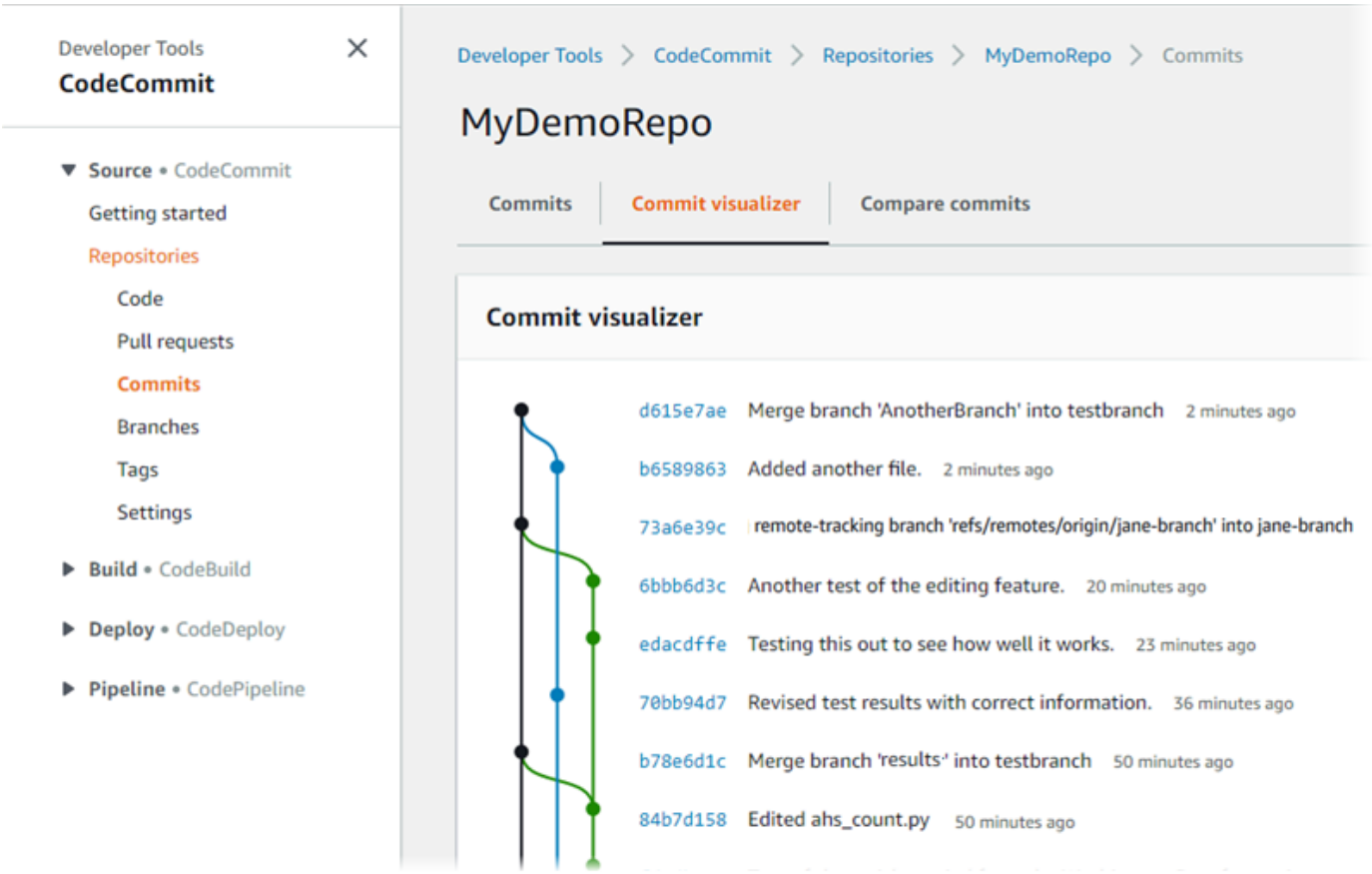
查看存储库的提交历史记录图表

您可以查看对存储库做出的提交的图表。Commit Visualizer 视图是一个有向无环图 (DAG)，它表示对存储库的某个分支做出的所有提交。该图形表示可以帮助您了解添加或合并提交及相关功能的时间。它还能帮助您确定某项更改相对其他更改的执行时间。

> **Note**
>
> 在提交图表中，使用快速转发方法合并的提交不会以单独的行显示。

查看提交图表

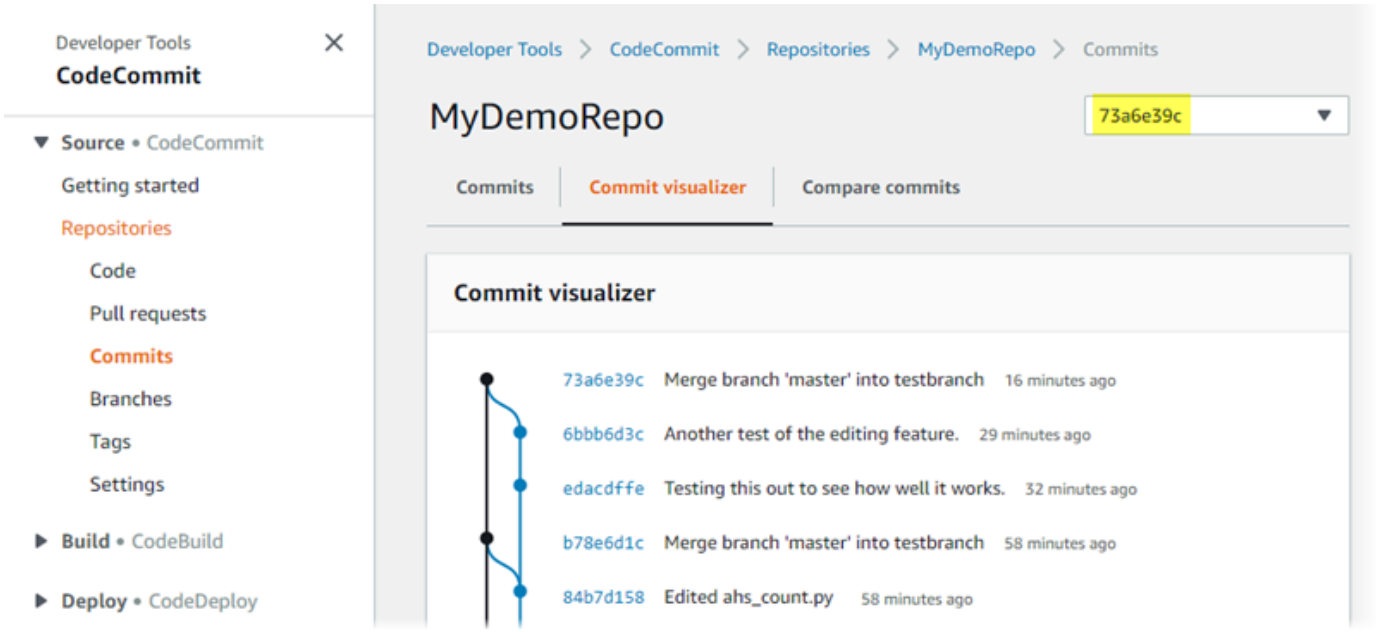
1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库)中，选择要查看其提交图表的存储库。
3. 在导航窗格中，选择 Commits (提交)，然后选择 Commit visualizer (提交可视化工具) 选项卡。



在提交图表中，每个提交消息的缩写提交 ID 和主题都显示在图表中所示点的旁边。

> **Note**
>
> 图表在一页上最多可显示 35 个分支。如果分支数超过 35 个，图表会变得太过复杂而无法显示。您可通过以下两种方法简化视图：
>
> - 使用视图选择器按钮显示特定分支的图表。
> - 将完整的提交 ID 粘贴到搜索框中以呈现该提交的图表。

4. 要呈现提交的新图表，请从图表中选择与该提交对应的点。视图选择器按钮更改为缩写的提交 ID。



查看提交详细信息 (AWS CLI)

Git 可让您查看提交的详细信息。您还可以通过运行以下命令 AWS CLI 来查看有关本地存储库或 CodeCommit 存储库中提交的详细信息：

- 要查看有关提交的信息，请运行 aws codecommit get-commit。
- 要查看有关多个提交的信息，请运行 aws codecommit batch-get-commits。
- 要查看有关合并提交的信息，请运行 aws codecommit get-merge-commit。
- 要查看有关提交说明符（分支、标签、HEAD 或其他完全限定的引用，如提交 ID）更改的信息，请运行 aws codecommit get-differences。
- 要查看存储库中单个 Git blob 对象的 base64 编码内容，请运行 aws codecommit get-blob。

查看有关提交的信息

1. 运行 aws codecommit get-commit 命令，并指定：

   - CodeCommit 存储库的名称（带--repository-name选项）。
   - 完整的提交 ID。

   例如，要在名为MyDemoRepo：的 CodeCommit 存储库317f8570EXAMPLE中查看 ID 为的提交的信息：

   ```
   aws codecommit get-commit  --repository-name MyDemoRepo  --commit-id
    317f8570EXAMPLE
   ```

2. 如果成功，该命令的输出包括以下内容：

   - 有关提交作者的信息（如 Git 中所配置），包括时间戳格式的日期和协调世界时 (UTC) 偏移量。
   - 有关提交者的信息（如 Git 中所配置），包括时间戳格式的日期和 UTC 偏移量。
   - 提交所在的 Git 树的 ID。
   - 父提交的提交 ID。
   - 提交消息。

   下面是前面示例命令的一些示例输出：

   ```
   {
       "commit": {
           "additionalData": "",
           "committer": {
               "date": "1484167798 -0800",
               "name": "Mary Major",
               "email": "mary_major@example.com"
           },
           "author": {
               "date": "1484167798 -0800",
               "name": "Mary Major",
               "email": "mary_major@example.com"
           },
           "treeId": "347a3408EXAMPLE",
           "parents": [
   ```

```
            "4c925148EXAMPLE"
        ],
        "message": "Fix incorrect variable name"
    }
}
```

查看有关合并提交的信息

1. 运行 get-merge-commit 命令，并指定：

   - 合并源的提交说明符（使用 --source-commit-specifier 选项）。
   - 合并目标的提交说明符（使用 --destination-commit-specifier 选项）。
   - 要使用的合并选项（使用 --merge-option 选项）。
   - 存储库的名称（使用 --repository-name 选项）。

   例如，要在名为 bugfix-bug1234 的存储库中使用 TH *REE_WAY_* MERGE 策略查看名为 *bugfix-bug1234* 的源分支与名为 *main* 的目标分支的合并提交的信息：*MyDemoRepo*

   ```
   aws codecommit get-merge-commit --source-commit-specifier bugfix-bug1234 --
   destination-commit-specifier main --merge-option THREE_WAY_MERGE --repository-
   name MyDemoRepo
   ```

2. 如果成功，该命令的输出返回类似以下内容的信息：

   ```
   {
       "sourceCommitId": "c5709475EXAMPLE",
       "destinationCommitId": "317f8570EXAMPLE",
       "baseCommitId": "fb12a539EXAMPLE",
       "mergeCommitId": "ffc4d608eEXAMPLE"
   }
   ```

查看有关多个提交的信息

1. 运行 batch-get-commits 命令，并指定：

   - CodeCommit 存储库的名称（带`--repository-name`选项）。
   - 要查看其信息的每个提交的完整提交 ID 的列表。

例如，要在名为MyDemoRepo：的 CodeCommit 存储库4c925148EXAMPLE中查看有关带有 ID 317f8570EXAMPLE 的提交的信息：

```
aws codecommit batch-get-commits  --repository-name MyDemoRepo  --commit-ids
 317f8570EXAMPLE 4c925148EXAMPLE
```

2. 如果成功，该命令的输出包括以下内容：

   - 有关提交作者的信息（如 Git 中所配置），包括时间戳格式的日期和协调世界时 (UTC) 偏移量。
   - 有关提交者的信息（如 Git 中所配置），包括时间戳格式的日期和 UTC 偏移量。
   - 提交所在的 Git 树的 ID。
   - 父提交的提交 ID。
   - 提交消息。

   下面是前面示例命令的一些示例输出：

```
{
    "commits": [
      {
        "additionalData": "",
        "committer": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "author": {
            "date": "1508280564 -0800",
            "name": "Mary Major",
            "email": "mary_major@example.com"
        },
        "commitId": "317f8570EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "6e147360EXAMPLE"
        ],
        "message": "Change variable name and add new response element"
    },
    {
```

```
        "additionalData": "",
        "committer": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "author": {
            "date": "1508280542 -0800",
            "name": "Li Juan",
            "email": "li_juan@example.com"
        },
        "commitId": "4c925148EXAMPLE",
        "treeId": "1f330709EXAMPLE",
        "parents": [
            "317f8570EXAMPLE"
        ],
        "message": "Added new class"
    }
}
```

查看提交说明符更改的相关信息

1. 运行 aws codecommit get-differences 命令，并指定：

   - CodeCommit 存储库的名称（带--repository-name选项）。
   - 要获取其信息的提交说明符。只有 --after-commit-specifier 是必需的。如果未指定 --before-commit-specifier，则显示自 --after-commit-specifier 起当前的所有文件。

   例如，要查看有关带有 ID 的提交317f8570EXAMPLE和名为MyDemoRepo：的 CodeCommit 存储库4c925148EXAMPLE中的提交之间的区别信息：

   ```
   aws codecommit get-differences  --repository-name MyDemoRepo  --before-commit-
   specifier 317f8570EXAMPLE --after-commit-specifier 4c925148EXAMPLE
   ```

2. 如果成功，该命令的输出包括以下内容：

   - 差异列表，包括更改类型 (A 表示已添加、D 表示已删除、M 表示已修改)。
   - 文件更改类型的模式。

- 包含更改的 Git blob 对象的 ID。

下面是前面示例命令的一些示例输出：

```
{
    "differences": [
        {
            "afterBlob": {
                "path": "blob.txt",
                "blobId": "2eb4af3bEXAMPLE",
                "mode": "100644"
            },
            "changeType": "M",
            "beforeBlob": {
                "path": "blob.txt",
                "blobId": "bf7fcf28fEXAMPLE",
                "mode": "100644"
            }
        }
    ]
}
```

查看有关 Git blob 对象的信息

1. 运行 aws codecommit get-blob 命令，并指定：

   - CodeCommit 存储库的名称（带--repository-name选项）。
   - Git Blob 的 ID（使用 --blob-id 选项）。

   例如，要在名MyDemoRepo为的 CodeCommit 仓库2eb4af3bEXAMPLE中查看 ID 为的 Git blob 的信息：

   ```
   aws codecommit get-blob  --repository-name MyDemoRepo  --blob-id 2eb4af3bEXAMPLE
   ```

2. 如果成功，该命令的输出包括以下内容：

   - blob 的 base64 编码内容，通常是一个文件。

例如，上一个命令的输出可能类似于以下内容：

```
{
    "content": "QSBCaW5hcnkgTGFyToEXAMPLE="
}
```

查看提交详细信息 (Git)

在执行这些步骤之前，您应该已经将本地存储库连接到 CodeCommit 存储库并提交了更改。有关说明，请参阅连接存储库。

要显示到存储库的最新提交的更改，请运行 git show 命令。

```
git show
```

该命令生成类似于下述信息的输出：

```
commit 4f8c6f9d
Author: Mary Major <mary.major@example.com>
Date:   Mon May 23 15:56:48 2016 -0700

    Added bumblebee.txt

diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
\ No newline at end of file
```

> **Note**
>
> 在此示例及以下示例中，提交 ID 为缩写格式。未显示完整的提交 ID。

要查看发生的更改，请使用 git show 命令并指定提交 ID：

```
git show 94ba1e60

commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..080f68f
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus.
```

要查看两个提交之间的差异，请运行 git diff 命令并提供两个提交 ID。

```
git diff ce22850d 4f8c6f9d
```

在本例中，两个提交之间的差异是添加了两个文件。该命令生成类似于下述信息的输出：

```
diff --git a/bees.txt b/bees.txt
new file mode 100644
index 0000000..cf57550
--- /dev/null
+++ b/bees.txt
@@ -0,0 +1 @@
+Bees are flying insects closely related to wasps and ants, and are known for their
 role in pollination and for producing honey and beeswax.
diff --git a/bumblebee.txt b/bumblebee.txt
new file mode 100644
index 0000000..443b974
--- /dev/null
+++ b/bumblebee.txt
@@ -0,0 +1 @@
+A bumblebee, also written bumble bee, is a member of the bee genus Bombus, in the
 family Apidae.
\ No newline at end of file
```

要使用 Git 查看有关本地存储库中的提交的详细信息，请运行 git log 命令：

```
git log
```

如果成功，该命令产生类似以下内容的输出：

```
commit 94ba1e60
Author: John Doe <johndoe@example.com>
Date:   Mon May 23 15:39:14 2016 -0700

    Added horse.txt

commit 4c925148
Author: Jane Doe <janedoe@example.com>
Date:   Mon May 22 14:54:55 2014 -0700

    Added cat.txt and dog.txt
```

要只显示提交 ID 和消息，请运行 git log --pretty=oneline 命令：

```
git log --pretty=oneline
```

如果成功，该命令产生类似以下内容的输出：

```
94ba1e60 Added horse.txt
4c925148 Added cat.txt and dog.txt
```

有关更多选项，请参阅 Git 文档。

比较中的提交 AWS CodeCommit

您可以使用 CodeCommit 控制台查看 CodeCommit存储库中提交说明符之间的区别。您可以快速查看提交与其父级之间的差异。还可以比较任意两个参考信息，包括提交 ID。

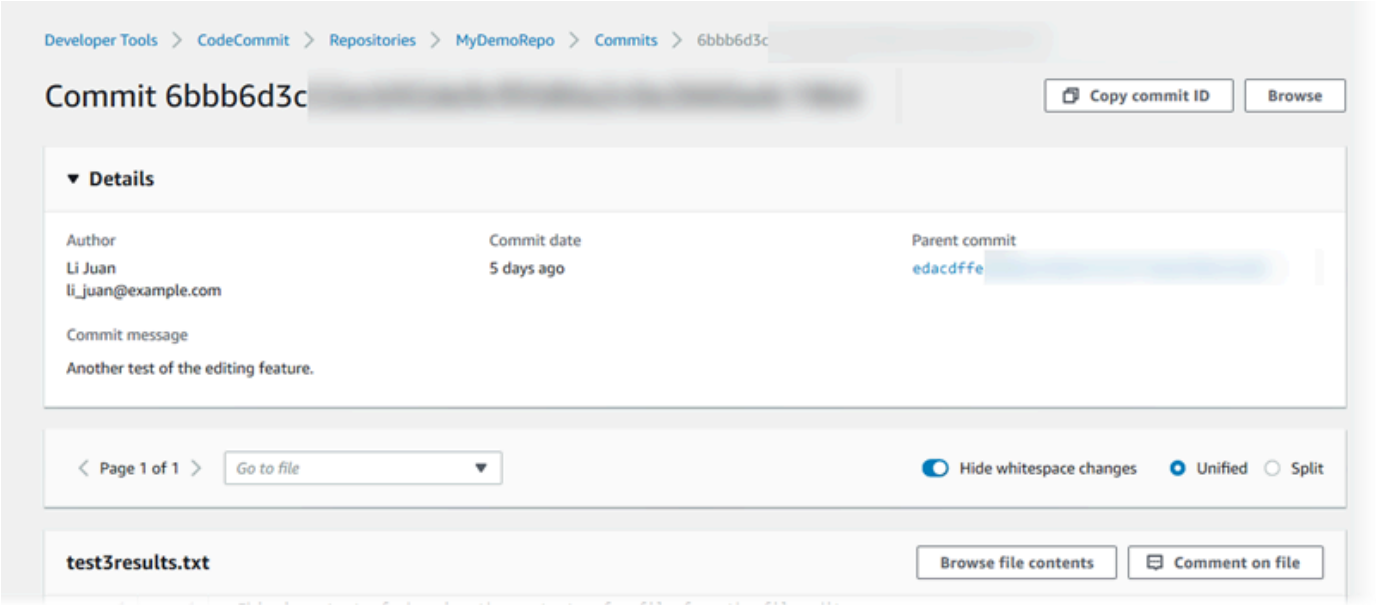
主题

- 比较提交与其父级
- 比较任意两个提交说明符

比较提交与其父级

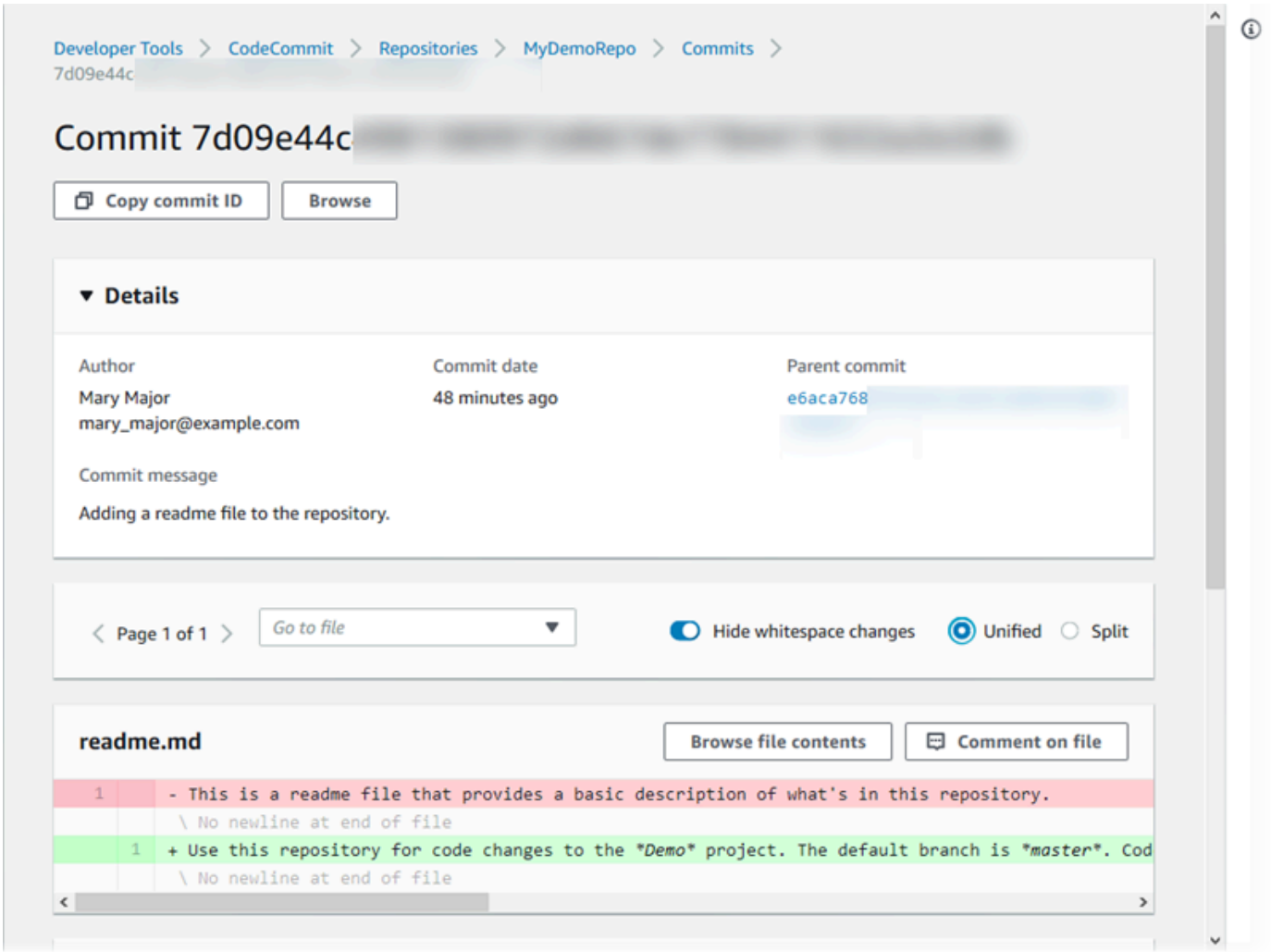
您可以快速查看提交与其父级之间的差异，从而审核提交消息、提交者和更改的内容。

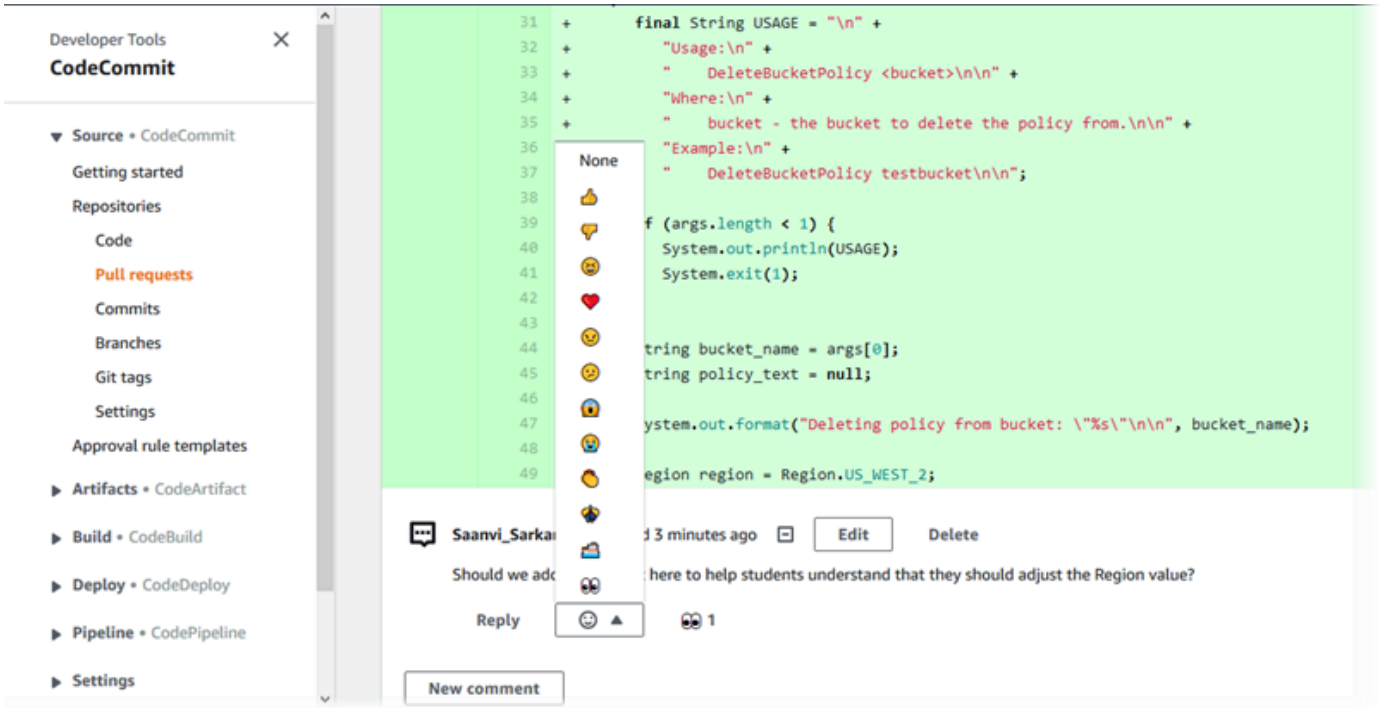
1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库页面上，选择您要查看提交与其父级之间差异的存储库。
3. 在导航窗格中，选择 Commits。
4. 选择列表中任意提交的缩写提交 ID。视图会发生变化以显示此提交的详细信息，包括它与其父提交之间的差异。



您可以通过并排 (Split 视图) 或内联 (Unified 视图) 方式显示更改。也可以隐藏或显示空格更改。您还可以添加评论。有关更多信息，请参阅 评论提交。

> **Note**
>
> 每当您更改用于查看代码和其他控制台设置的首选项时，都会将它们保存为浏览器 Cookie。有关更多信息，请参阅 使用用户首选项。


Developer Tools > CodeCommit > Repositories > MyDemoRepo > Commits >
7d09e44c
Commit 7d09e44c
Copy commit ID
Browse
Details
Author
Mary Major
mary_major@example.com
Commit date
48 minutes ago
Parent commit
e6aca768
Commit message
Adding a readme file to the repository.
Page 1 of 1
Go to file
Hide whitespace changes
Unified
Split
readme.md
Browse file contents
Comment on file
1 - This is a readme file that provides a basic description of what's in this repository.
\ No newline at end of file
1 + Use this repository for code changes to the *Demo* project. The default branch is *master*. Cod
\ No newline at end of file



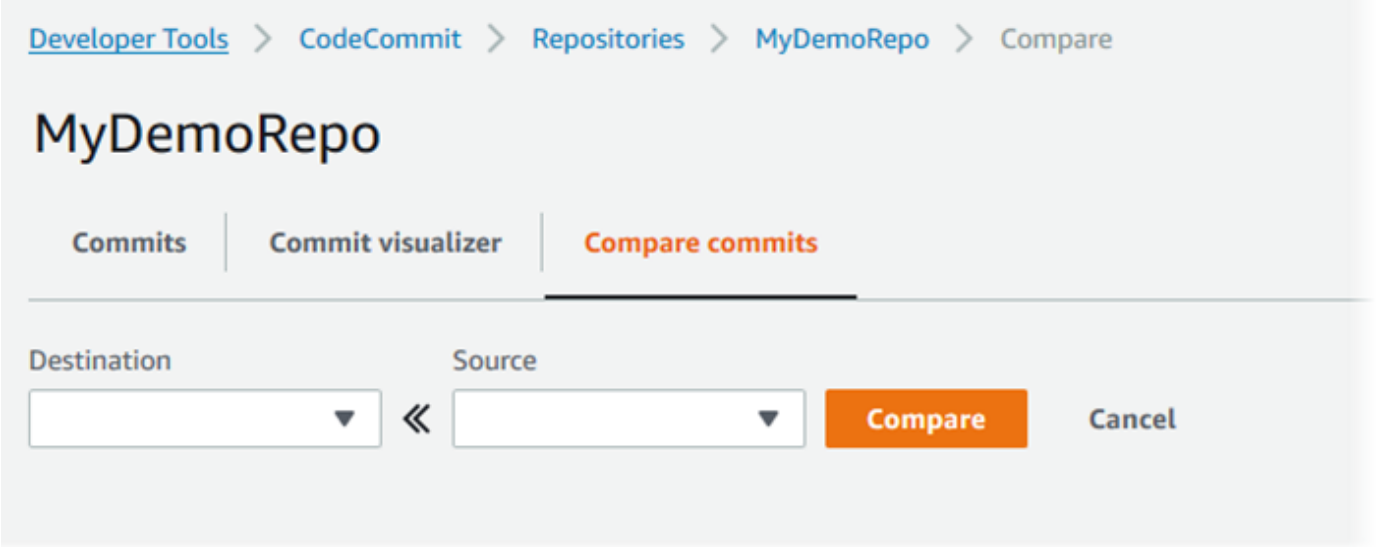
> **Note**
>
> 根据行结束样式、代码编辑器及其他因素的不同，您可能会看到整行的添加或删除，而不是一行中的具体更改。详细程度与 git show 或 git diff 命令的返回内容一致。

5. 要将提交与其父级进行比较，请从提交可视化工具选项卡中选择缩写的提交 ID。将显示提交详细信息，包括提交与其父级之间的更改。

比较任意两个提交说明符

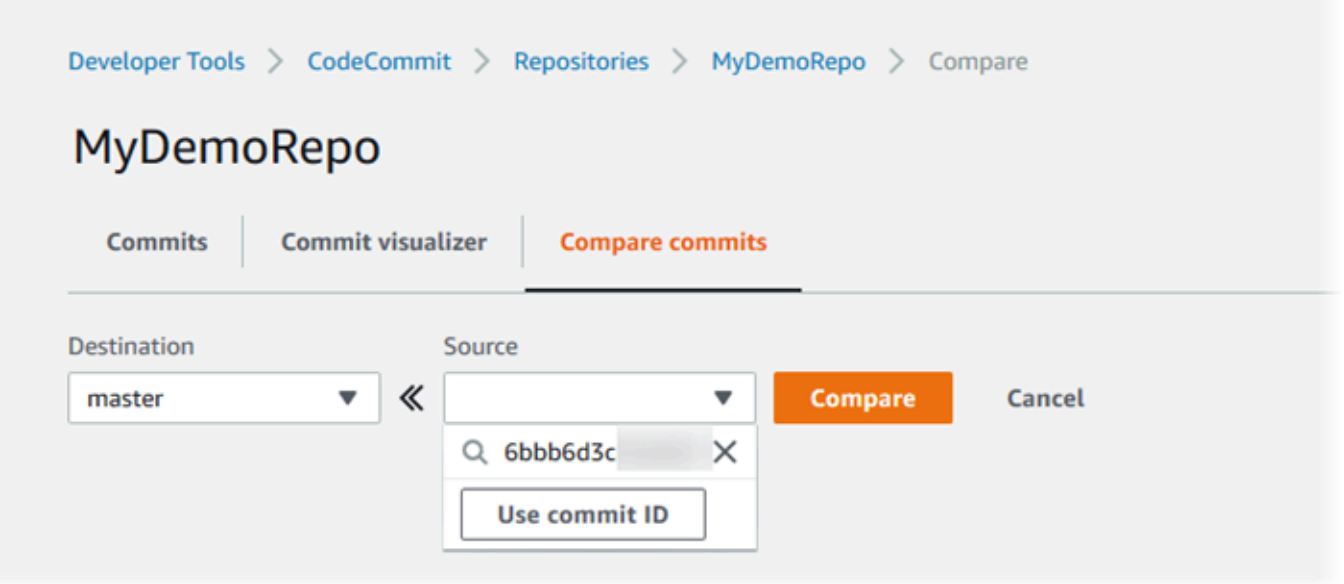
您可以在 CodeCommit 控制台中查看任意两个提交说明符之间的区别。提交说明符是参考信息，如分支、标签和提交 ID。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库页面上，选择要比较提交、分支或已标记提交的存储库。
3. 在导航窗格中，选择提交，然后选择比较提交。

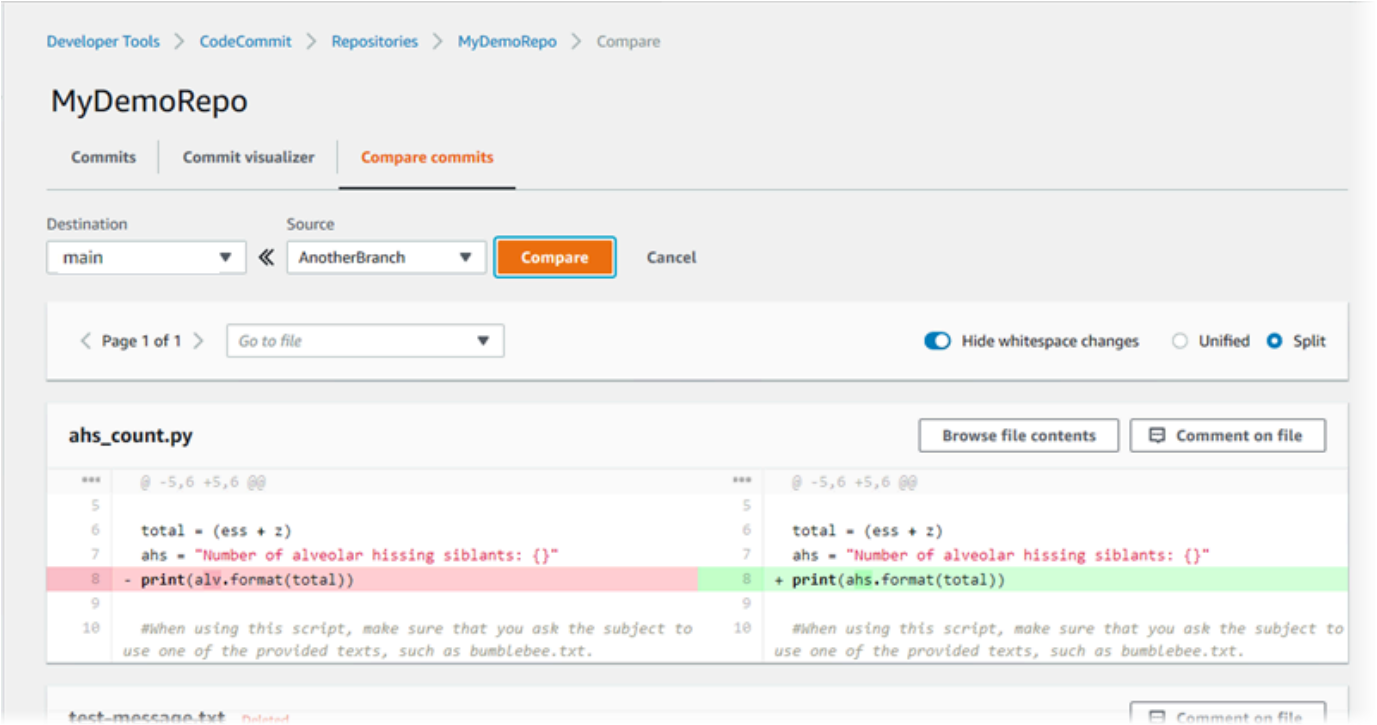


4. 使用框比较两个提交说明符。

   - 要比较分支的提示，请从列表中选择分支名称。这会从分支中选择最近的提交进行比较。
   - 要比较与特定标签关联的提交，请从列表中选择标签名称（如果有）。这会选择已标记的提交进行比较。
   - 要比较特定提交，请在框中输入或粘贴提交 ID。要获取完整的提交 ID，请在导航栏中选择 Commits，然后从列表中复制提交 ID。在 Compare commits (比较提交) 页面上，将完整的提交 ID 粘贴在文本框中，然后选择 Use commit ID (使用提交 ID)。



5. 选中说明符后，选择 Compare。



您可以通过并排 (Split 视图) 或内联 (Unified 视图) 方式显示差别。也可以隐藏或显示空格更改。

6. 要清除您的比较选择，请选择取消。

评论中的提交 AWS CodeCommit

您可以使用 CodeCommit 控制台对仓库中的提交进行评论，以及查看和回复其他用户对提交的评论。这可以帮助您讨论在存储库中进行的更改，包括：

- 进行更改的原因。
- 是否需要进行多处更改。
- 是否应将更改合并到其他分支中。

您可以对整个提交、提交中的某个文件或文件中特定的行或更改发表评论。您也可以通过以下方式链接到一行代码：选择一行，然后在浏览器中复制生成的 URL。

> **Note**
>
> 为了获得最佳结果，请在以 IAM 用户身份登录时使用评论功能。评论功能并未针对用根账户凭证、联合访问或临时凭证登录的用户进行优化。

主题

查看对存储库中的提交的评论

您可以使用 CodeCommit 控制台查看对提交的评论。

查看对提交进行的评论

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要查看对其提交进行的评论的存储库。
3. 在导航窗格中，选择 Commits。选择您要查看任何评论的提交的提交 ID。

   该提交的页面将与任何评论一起显示。

在存储库中添加和回复对提交的评论

您可以使用 CodeCommit 控制台为提交和父提交的比较或两个指定提交之间的比较添加注释。您还可以使用表情符号和/或您自己的评论来回复评论。

添加和回复对提交的评论（控制台）

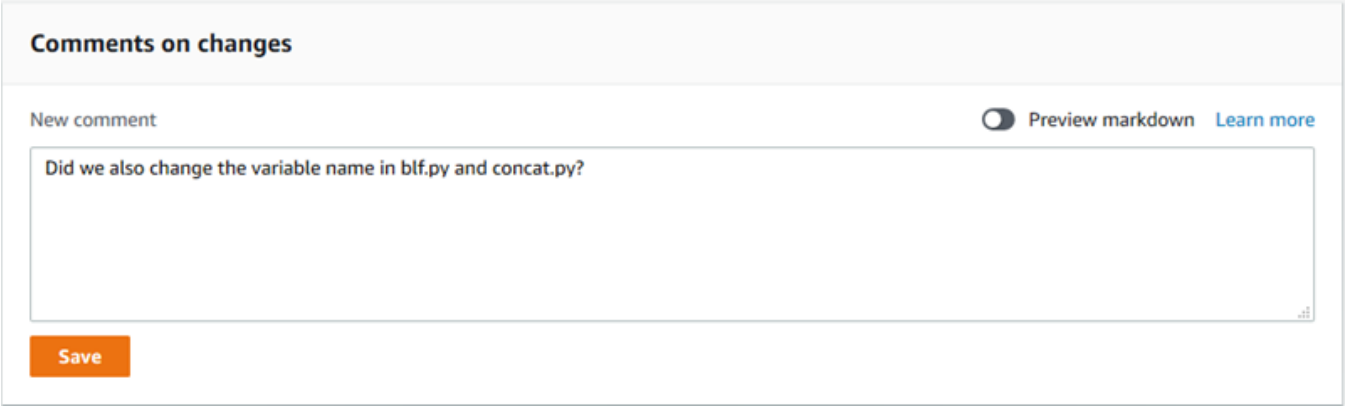
您可以使用文本和表情符号添加和回复对提交的评论。您的评论和表情符号将标记为属于用于登录控制台的 IAM 用户或角色。

添加和回复对提交进行的评论

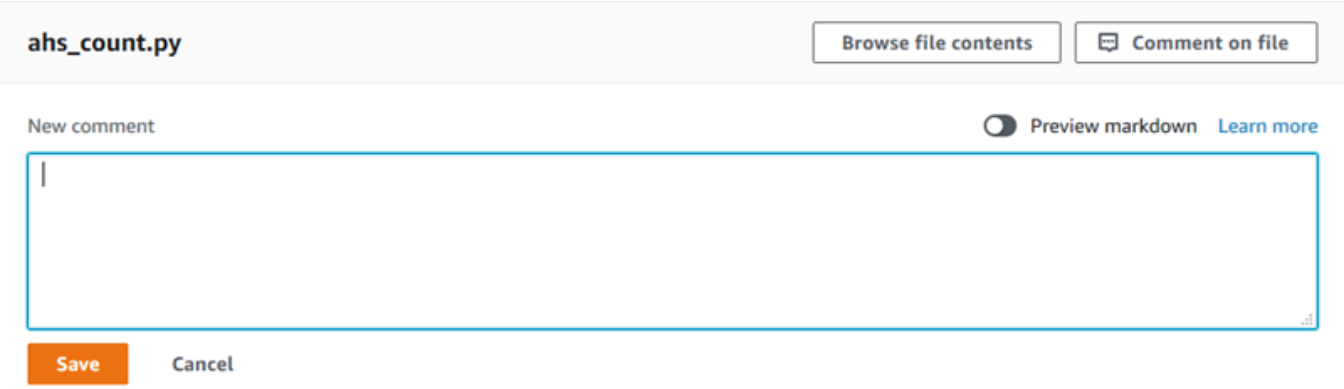
1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要在其中对提交进行评论的存储库。
3. 在导航窗格中，选择 Commits。选择您要在其中添加或回复评论的提交的提交 ID。

   该提交的页面将与任何评论一起显示。
4. 要添加评论，请执行以下操作之一：

- 要添加一般评论，请在对更改的评论中输入评论，然后选择保存。您可以使用 Markdown，也可以纯文本格式输入评论。



- 要向提交中的文件添加评论，请找到该文件的名称。选择对文件的评论，输入评论，然后选择保存。



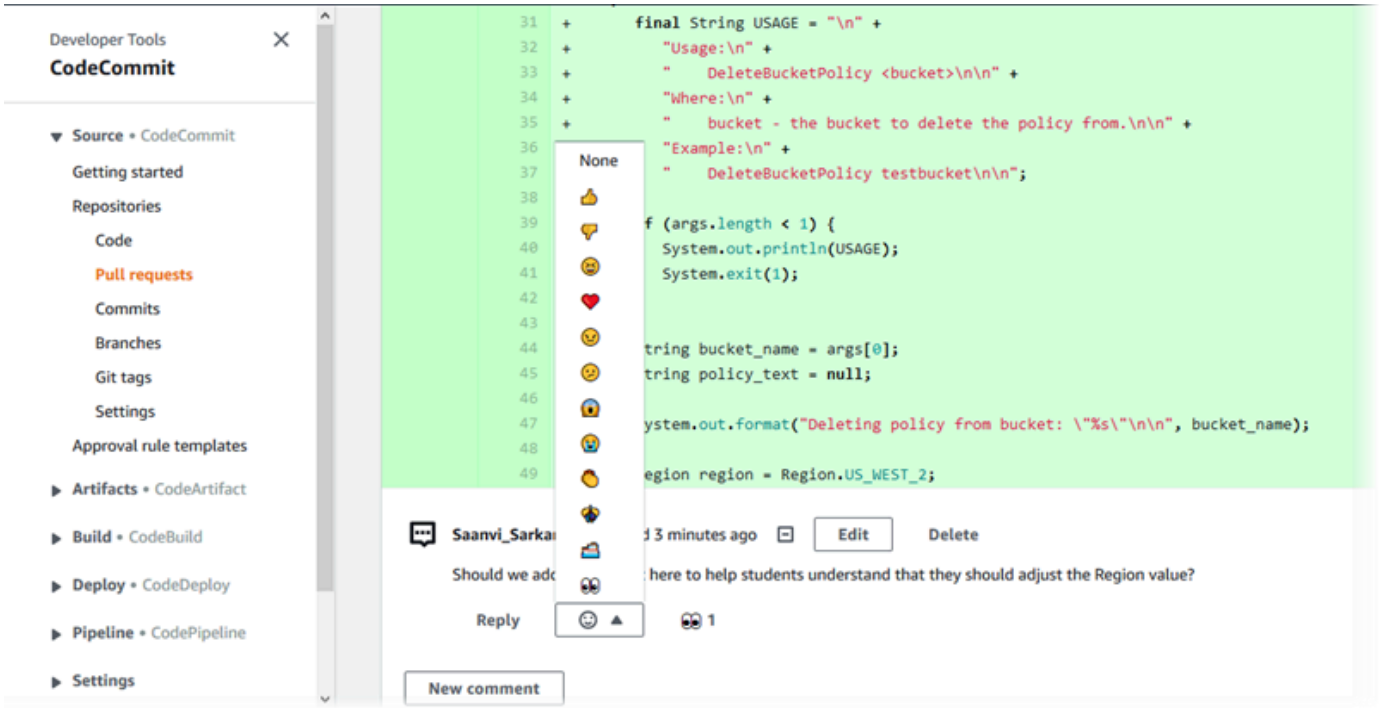
- 要向提交中的已更改行添加评论，请转到显示更改的行。选择评论气泡

，

输入评论，然后选择保存。

> **Note**
>
> 您可以在保存评论后编辑它。您也可以删除其内容。评论将保留一条消息，说明内容已被删除。请考虑在保存您的评论之前为该评论使用预览 markdown 模式。

5. 要回复对提交的评论，请选择回复。要使用表情符号回复评论，请从列表中选择所需的表情符号。每条评论只能选择一个表情符号。如果要更改表情符号反应，请从列表中选择其他表情符号，或选择无以删除反应。

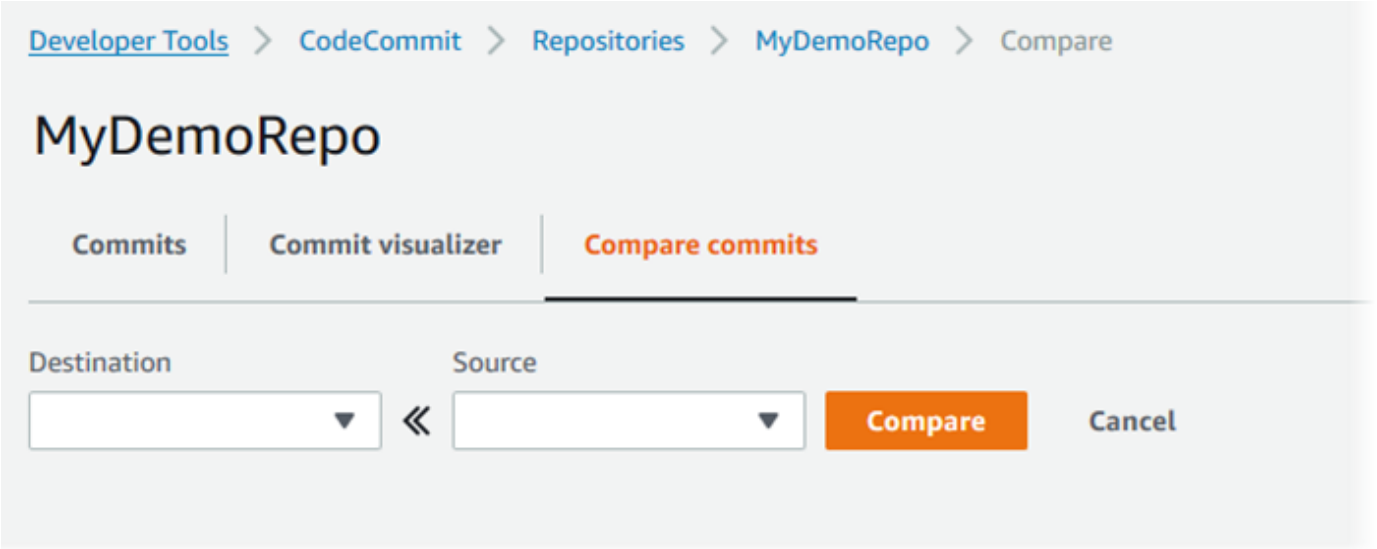


在比较两个提交说明符时添加和回复评论

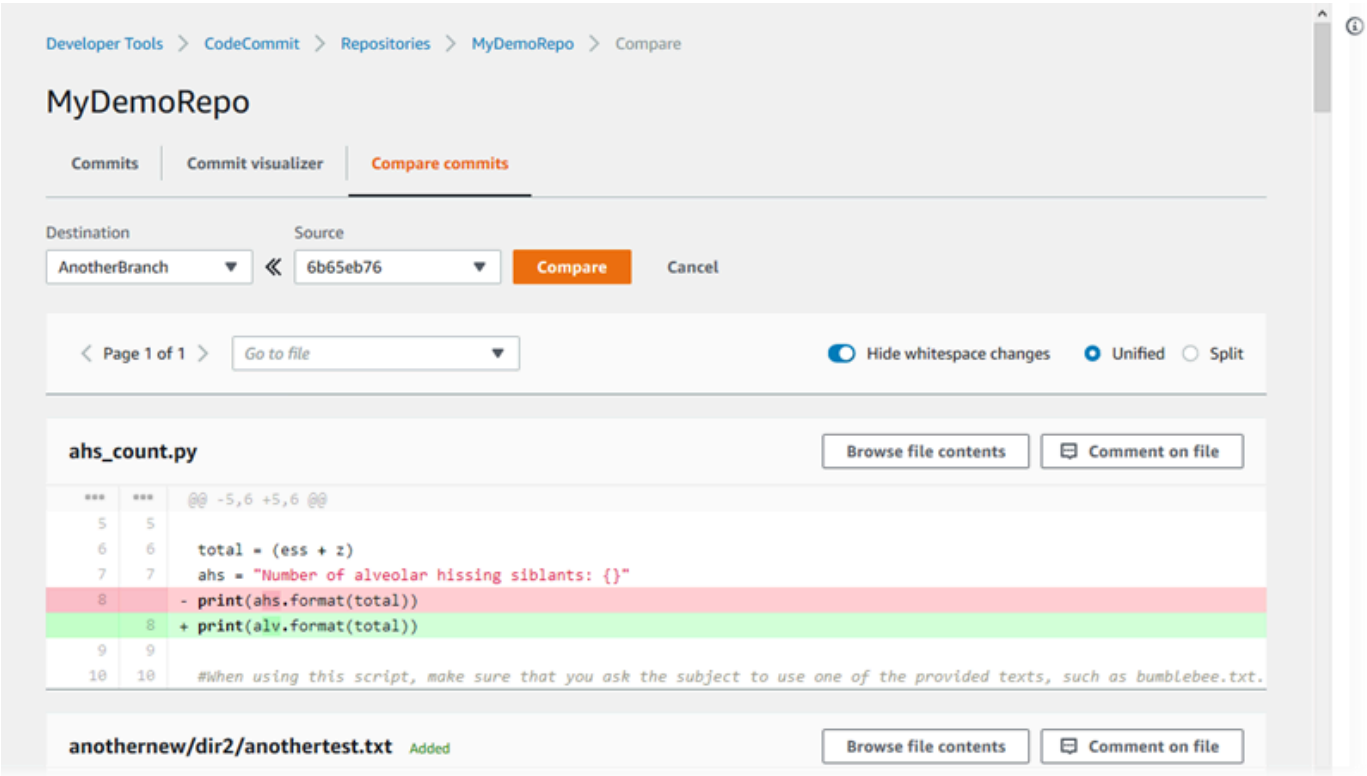
您可以向两个分支、标签或提交之间的比较添加评论。

在比较提交说明符时添加或回复评论

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要比较提交、分支或已标记提交的存储库。
3. 在导航窗格中，选择提交，然后选择比较提交选项卡。



4. 使用目标和源字段来比较两个提交说明符。使用下拉列表或粘贴提交 ID。选择 Compare。



5. 执行以下一个或多个操作：

   - 要添加对文件或行的评论，请选择评论气泡

     。

   - 要添加对所比较更改的一般评论，请转到 Comments on changes。

查看、添加、更新和回复评论 (AWS CLI)

您可以通过运行以下命令来查看、添加、回复、更新和删除评论内容：

- 要查看对两个提交之间的比较进行的评论，请运行 get-comments-for-compared-commit。
- 要查看有关评论的详细信息，请运行 get-comment。
- 要删除您创建的评论的内容，请运行 delete-comment-content。
- 要创建对两个提交之间的比较进行的评论，请运行 post-comment-for-compared-commit。
- 要更新评论，请运行 update-comment。
- 要回复评论，请运行 post-comment-reply。
- 要使用表情符号回复评论，请运行 put-comment-reaction。
- 要查看对评论的表情符号反应，请运行 get-comment-reactions。

查看对提交进行的评论

1. 运行 get-comments-for-compared-commit 命令，并指定：

   - CodeCommit 存储库的名称（带`--repository-name`选项）。
   - “after”提交的完整提交 ID，用于建立比较的方向性（使用 `--after-commit-id option`）。
   - “before”提交的完整提交 ID，用于建立比较的方向性（使用 `--before-commit-id` 选项）。
   - （可选）要返回下一批结果的枚举令牌（使用 `--next-token` 选项）。
   - (可选) 一个用于限制返回的结果数的非负整数 (使用 `--max-results` 选项)。

   例如，要查看对名为的存储库中两个提交之间的比较所做的评论 *MyDemoRepo*：

   ```
   aws codecommit get-comments-for-compared-commit --repository-name MyDemoRepo --
   before-commit-ID 6e147360EXAMPLE --after-commit-id 317f8570EXAMPLE
   ```

2. 如果成功，该命令产生类似以下内容的输出：

   ```
   {
      "commentsForComparedCommitData": [
         {
            "afterBlobId": "1f330709EXAMPLE",
            "afterCommitId": "317f8570EXAMPLE",
            "beforeBlobId": "80906a4cEXAMPLE",
   ```

```
        "beforeCommitId": "6e147360EXAMPLE",
        "comments": [
           {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "123Example",
              "commentId": "ff30b348EXAMPLEb9aa670f",
              "content": "Whoops - I meant to add this comment to the line, not
the file, but I don't see how to delete it.",
              "creationDate": 1508369768.142,
              "deleted": false,
              "CommentId": "123abc-EXAMPLE",
              "lastModifiedDate": 1508369842.278,
              "callerReactions": [],
              "reactionCounts":
               {
                 "SMILE" : 6,
                 "THUMBSUP" : 1
               }
           },
           {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "123Example",
              "commentId": "553b509bEXAMPLE56198325",
              "content": "Can you add a test case for this?",
              "creationDate": 1508369612.240,
              "deleted": false,
              "commentId": "456def-EXAMPLE",
              "lastModifiedDate": 1508369612.240,
              "callerReactions": [],
              "reactionCounts":
               {
                 "THUMBSUP" : 2
               }
            }
        ],
        "location": {
           "filePath": "cl_sample.js",
           "filePosition": 1232,
           "relativeFileVersion": "after"
        },
        "repositoryName": "MyDemoRepo"
     }
  ],
  "nextToken": "exampleToken"
```

```
}
```

查看提交评论的详细信息

1. 运行 get-comment 命令，并指定系统生成的评论 ID。例如：

```
aws codecommit get-comment --comment-id ff30b348EXAMPLEb9aa670f
```

2. 如果成功，该命令返回类似以下内容的输出：

```
{
   "comment": {
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "content": "Whoops - I meant to add this comment to the line, but I don't see
 how to delete it.",
      "creationDate": 1508369768.142,
      "deleted": false,
      "commentId": "",
      "lastModifiedDate": 1508369842.278,
      "callerReactions": [],
      "reactionCounts":
         {
           "SMILE" : 6,
           "THUMBSUP" : 1
         }
   }
}
```

删除提交评论的内容

1. 运行 delete-comment-content 命令，并指定系统生成的评论 ID。例如：

```
aws codecommit delete-comment-content --comment-id ff30b348EXAMPLEb9aa670f
```

> **Note**
>
> 只有在应用了 AWSCodeCommitFullAccess 策略或将`DeleteCommentContent`权限设置为 “允许” 的情况下，您才能删除评论的内容。

2. 如果成功，该命令产生类似以下内容的输出：

```
{
   "comment": {
      "creationDate": 1508369768.142,
      "deleted": true,
      "lastModifiedDate": 1508369842.278,
      "clientRequestToken": "123Example",
      "commentId": "ff30b348EXAMPLEb9aa670f",
      "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
      "callerReactions": [],
      "reactionCounts":
         {
           "CLAP" : 1
         }
   }
}
```

创建提交评论

1. 运行 post-comment-for-compared-commit 命令，并指定：

   - CodeCommit 存储库的名称（带`--repository-name`选项）。
   - “after”提交的完整提交 ID，用于建立比较的方向性（使用 `--after-commit-id` 选项）。
   - “before”提交的完整提交 ID，用于建立比较的方向性（使用 `--before-commit-id` 选项）。
   - 客户端生成的唯一等幂令牌（使用 --client-request-token 选项）。
   - 您的评论的内容（使用 --content 选项）。
   - 有关评论放置位置信息的列表，包括：
     - 所比较文件的名称，包括其扩展名和子目录（如果有，则使用 filePath 属性）。
     - 在比较文件中更改的行号（使用 filePosition 属性）。

- 对更改的评论在源分支与目标分支之间比较“之前”还是“之后”（使用 relativeFileVersion 属性）。

例如，要添加评论 *"Can you add a test case for this?"* 关于比较名为 *MyDemoRepo* 的存储库中两次提交时对 *cl_sample.js* 文件的更改：

```
aws codecommit post-comment-for-compared-commit --repository-name MyDemoRepo
 --before-commit-id 317f8570EXAMPLE --after-commit-id 5d036259EXAMPLE --client-
request-token 123Example --content "Can you add a test case for this?" --location
 filePath=cl_sample.js,filePosition=1232,relativeFileVersion=AFTER
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
        "afterBlobId": "1f330709EXAMPLE",
        "afterCommitId": "317f8570EXAMPLE",
        "beforeBlobId": "80906a4cEXAMPLE",
        "beforeCommitId": "6e147360EXAMPLE",
        "comment": {
              "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
              "clientRequestToken": "",
              "commentId": "553b509bEXAMPLE56198325",
              "content": "Can you add a test case for this?",
              "creationDate": 1508369612.203,
              "deleted": false,
              "commentId": "abc123-EXAMPLE",
              "lastModifiedDate": 1508369612.203,
              "callerReactions": [],
              "reactionCounts": []
            },
            "location": {
              "filePath": "cl_sample.js",
              "filePosition": 1232,
              "relativeFileVersion": "AFTER"
            },
        "repositoryName": "MyDemoRepo"
 }
```

更新提交评论

1. 运行 update-comment 命令，并指定要替换任何现有内容的系统生成的评论 ID 和内容。

   例如，要将内容 *"Fixed as requested. I'll update the pull request."* 添加到 ID 为 *442b498bEXAMPLE5756813* 的评论，请运行以下命令：

   ```
   aws codecommit update-comment --comment-id 442b498bEXAMPLE5756813 --content "Fixed
    as requested. I'll update the pull request."
   ```

2. 如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "comment": {
           "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
           "clientRequestToken": "",
           "commentId": "442b498bEXAMPLE5756813",
           "content": "Fixed as requested. I'll update the pull request.",
           "creationDate": 1508369929.783,
           "deleted": false,
           "lastModifiedDate": 1508369929.287,
           "callerReactions": [],
           "reactionCounts":
             {
               "THUMBSUP" : 2
             }
       }
    }
   ```

回复提交评论

1. 要在拉取请求中发布对评论的回复，请运行 post-comment-reply 命令，并且指定：

   - 要回复的评论的系统生成的 ID（使用 --in-reply-to 选项）。
   - 客户端生成的唯一等幂令牌（使用 --client-request-token 选项）。
   - 您的回复的内容（使用 --content 选项）。

   例如，要将回复 *"Good catch. I'll remove them."* 添加到系统生成的 ID 为 *abcd1234EXAMPLEb5678efgh* 的评论，请运行以下命令：

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "comment": {
        "authorArn": "arn:aws:iam::111111111111:user/Li_Juan",
        "clientRequestToken": "123Example",
        "commentId": "442b498bEXAMPLE5756813",
        "content": "Good catch. I'll remove them.",
        "creationDate": 1508369829.136,
        "deleted": false,
        "CommentId": "abcd1234EXAMPLEb5678efgh",
        "lastModifiedDate": 150836912.221,
        "callerReactions": [],
        "reactionCounts": []
    }
 }
```

使用表情符号回复对提交的评论

1. 要使用表情符号回复拉取请求中的评论，或者要更改表情符号反应的值，请运行 put-comment-reaction 命令，指定以下内容：

   - 要使用表情符号回复的评论的系统生成的 ID。
   - 要添加或更新的反应的值。可接受的值包括支持的表情符号、简码和 Unicode 值。

   中的表情符号支持以下值： CodeCommit

表情符号	短代码	Unicode
#	:thumbsup:	U+1F44D
#	:thumbsdown:	U+1F44E
#	:smile:	U+1F604

表情符号	短代码	Unicode
❤	:heart:	U+2764
#	:angry:	U+1F620
#	:confused:	U+1F615
#	:scream:	U+1F631
#	:sob:	U+1F62D
#	:clap:	U+1F44F
#	:confetti_ball:	U+1F38A
#	:ship:	U+1F6A2
#	:eyes:	U+1F440
	none	U+0000

例如，要将表情符号 *:thumbsup:* 添加到系统生成的 ID 为 *abcd1234EXAMPLEb5678efgh* 的评论，请运行以下命令：

```
aws codecommit put-comment-reaction --comment-id abcd1234EXAMPLEb5678efgh --
reaction-value :thumbsup:
```

2. 如果成功，此命令不会产生任何输出。

查看对评论的表情符号反应

1. 要查看对评论的表情符号反应（包括使用这些表情符号做出了反应的用户），请运行 get-comment-reactions 命令，指定评论的系统生成的 ID。

   例如，要将表情符号反应添加到系统生成的 ID 为 *abcd1234EXAMPLEb5678efgh* 的评论，请运行以下命令：

   ```
   aws codecommit get-comment-reactions --comment-id abcd1234EXAMPLEb5678efgh
   ```

2. 如果成功，该命令产生类似以下内容的输出：

```
{
    "reactionsForComment": {
        [
           {
                "reaction": {
                    "emoji:"#",
                    "shortCode": "thumbsup",
                    "unicode": "U+1F44D"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Li_Juan",
                    "arn:aws:iam::123456789012:user/Mary_Major",
                    "arn:aws:iam::123456789012:user/Jorge_Souza"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "thumbsdown",
                    "unicode": "U+1F44E"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Nikhil_Jayashankar"
                ]
            },
            {
                "reaction": {
                    "emoji": "#",
                    "shortCode": "confused",
                    "unicode": "U+1F615"
                },
                "users": [
                    "arn:aws:iam::123456789012:user/Saanvi_Sarkar"
                ]
            }
        ]
    }
}
```

在中创建 Git 标签 AWS CodeCommit

您可以使用 Git 标签通过标签来标记提交，以帮助其他存储库用户了解其重要性。要在 CodeCommit 仓库中创建 Git 标签，您可以使用连接到仓库的本地 CodeCommit 存储库中的 Git。在本地存储库中创建 Git 标签后，您可以使用将其推送git push --tags到 CodeCommit 存储库。

有关更多信息，请参阅 查看标签详细信息。

使用 Git 创建标签

按照以下步骤使用本地存储库中的 Git 在 CodeCommit 仓库中创建 Git 标签。

在这些步骤中，我们假设您已经将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

1. 运行git tag ***new-tag-name commit-id***命令，其中*new-tag-name*是新 Git 标签的名称，commit *-id* 是要与 Git 标签关联的提交 ID。

   例如，下面的命令创建一个名为 `beta` 的 Git 标签，并将其关联到提交 ID `dc082f9a...af873b88`：

   ```
   git tag beta dc082f9a...af873b88
   ```

2. 要将新的 Git 标签从本地存储库推送到 CodeCommit 存储库，请运行git push ***remote-name new-tag-name***命令，其中 rem *ote-name* 是 CodeCommit 存储库的名称，*new-tag-name*也是新 Git 标签的名称。

   例如，要将名为的新 Git 标签推送`beta`到名为的 CodeCommit 仓库`origin`：

   ```
   git push origin beta
   ```

> **Note**
>
> 要将所有新的 Git 标签从本地存储库推送到 CodeCommit 存储库，请运行git push --tags。要确保使用仓库中的所有 Git 标签更新您的本地 CodeCommit 存储库，请运行，git fetch然后执行。git fetch --tags

有关更多选项，请参阅 Git 文档。

在中查看 Git 标签的详细信息 AWS CodeCommit

在 Git 中，标签是可以应用于引用 (如提交) 的标记，用于为引用标记对其他存储库用户可能很重要的信息。例如，您可以使用 **beta** 标签为项目标记 beta 版本的提交。有关更多信息，请参阅 使用 Git 创建标签。Git 标签与存储库标签不同。有关如何使用存储库标签的更多信息，请参阅为存储库添加标签。

您可以使用 AWS CodeCommit 控制台查看仓库中 Git 标签的相关信息，包括每个 Git 标签引用的提交日期和提交消息。从控制台中，您可以将标签引用的提交与存储库默认分支的标头进行比较。与其他任何提交一样，您还可以查看该 Git 标签点的代码。

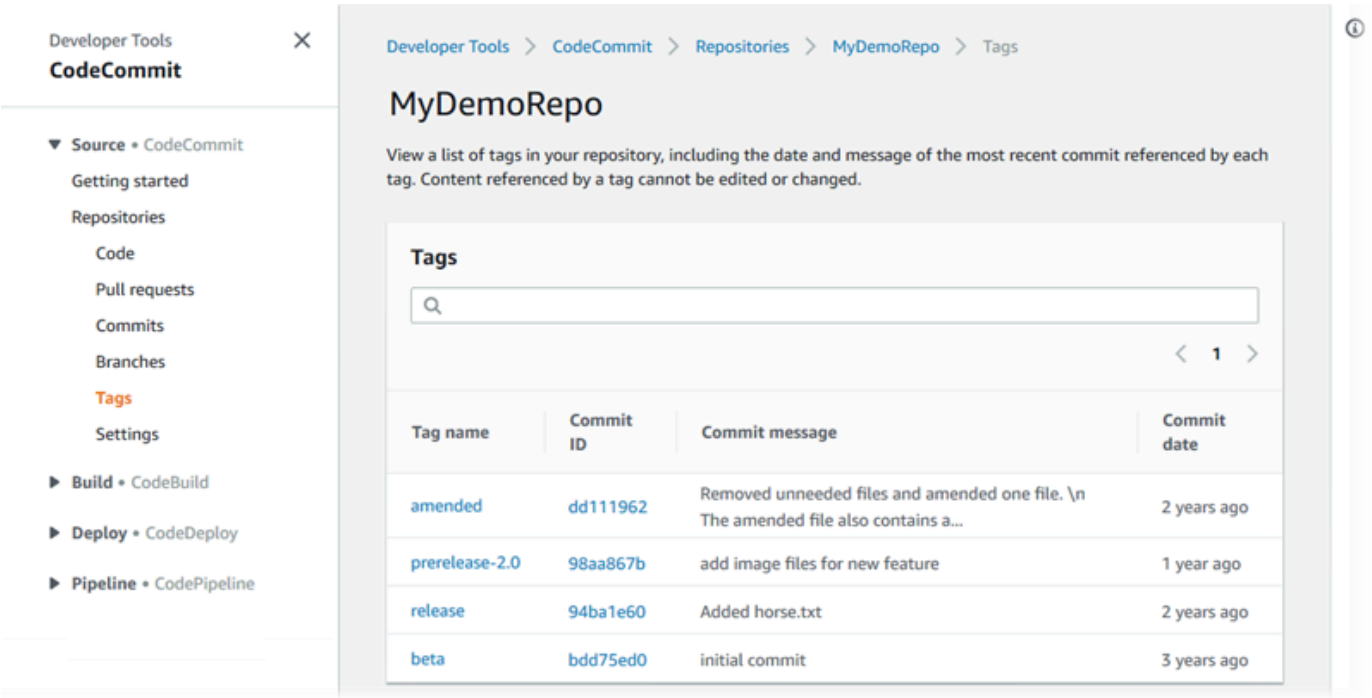
您还可以从您的终端或命令行使用 Git 查看有关本地存储库中 Git 标签的详细信息。

主题

查看标签详细信息（控制台）

使用 AWS CodeCommit 控制台快速查看仓库的 Git 标签列表以及有关 Git 标签引用的提交的详细信息。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中查看标签的存储库的名称。
3. 在导航窗格中，选择 Git tags (Git 标签)。



4. 请执行以下操作之一：

   - 要查看标签在该提交处的代码，请选择 Git 标签名称。
   - 要查看提交的详细信息，包括完整提交消息、提交者和作者，请选择缩写的提交 ID。

查看 Git 标签详细信息 (Git)

要使用 Git 查看中本地存储库中 Git 标签的详细信息，请运行以下命令之一：

- git tag：查看 Git 标签名称的列表。
- git show：查看特定 Git 标签的信息。
- git ls-remot e 用于查看仓库中有关 Git 标签的信息。 CodeCommit

> **Note**
>
> 要确保使用仓库中的所有 Git 标签更新您的本地 CodeCommit 存储库，请运行，git fetch然后执行。git fetch --tags

在以下步骤中，我们假设您已经将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

查看本地存储库中 Git 标签的列表

1. 运行 git tag 命令：

```
git tag
```

2. 如果成功，该命令产生类似以下内容的输出：

```
beta
release
```

> **Note**
>
> 如果尚未定义任何标签，git tag 不返回任何内容。

有关更多选项，请参阅 Git 文档。

查看本地存储库中 Git 标签的信息

1. 运行 git show ***tag-name*** 命令。例如，要查看有关名为 beta 的 Git 标签的信息，请运行：

```
git show beta
```

2. 如果成功，该命令产生类似以下内容的输出：

```
commit 317f8570...ad9e3c09
Author: John Doe <johndoe@example.com>
Date:   Tue Sep 23 13:49:51 2014 -0700

    Added horse.txt

diff --git a/horse.txt b/horse.txt
new file mode 100644
index 0000000..df42ff1
--- /dev/null
+++ b/horse.txt
@@ -0,0 +1 @@
+The horse (Equus ferus caballus) is one of two extant subspecies of Equus ferus
\ No newline at end of file
```

> **Note**
>
> 要退出 Git 标签信息输出，请键入 :q。

有关更多选项，请参阅 Git 文档。

查看 CodeCommit 仓库中 Git 标签的相关信息

1. 运行 git ls-remote --tags 命令。

```
git ls-remote --tags
```

2. 如果成功，此命令将生成 CodeCommit 存储库中 Git 标签的列表作为输出：

```
129ce87a...70fbffba    refs/tags/beta
785de9bd...59b402d8    refs/tags/release
```

   如果尚未定义任何 Git 标签，git ls-remote --tags 将返回一个空白行。

有关更多选项，请参阅 Git 文档。

删除中的 Git 标签 AWS CodeCommit

要删除 CodeCommit 仓库中的 Git 标签，请使用连接到仓库的本地 CodeCommit 存储库中的 Git。

使用 Git 删除 Git 标签

按照以下步骤使用本地存储库中的 Git 删除 CodeCommit 仓库中的 Git 标签。

编写这些步骤时假设您已经将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

1. 要从本地存储库中删除 Git 标签，请运行 git tag -d ***tag-name*** 命令，其中 *tag-name* 是要删除的 Git 标签的名称。

> **Tip**
>
> 要获取标 Git 签名称列表，请运行 git tag。

例如，要删除本地存储库中名为 `beta` 的 Git 标签，请运行以下命令：

```
git tag -d beta
```

2. 要从 CodeCommit 仓库中删除 Git 标签，请运行git push ***remote-name*** --delete ***tag-name***命令，其中 *remote-name* 是本地 CodeCommit 存储库使用的昵称，*tag-name* 是要从仓库中删除的 Git 标签的名称。 CodeCommit

> **Tip**
>
> 要获取 CodeCommit 仓库名称及其 URL 的列表，请运行该git remote -v命令。

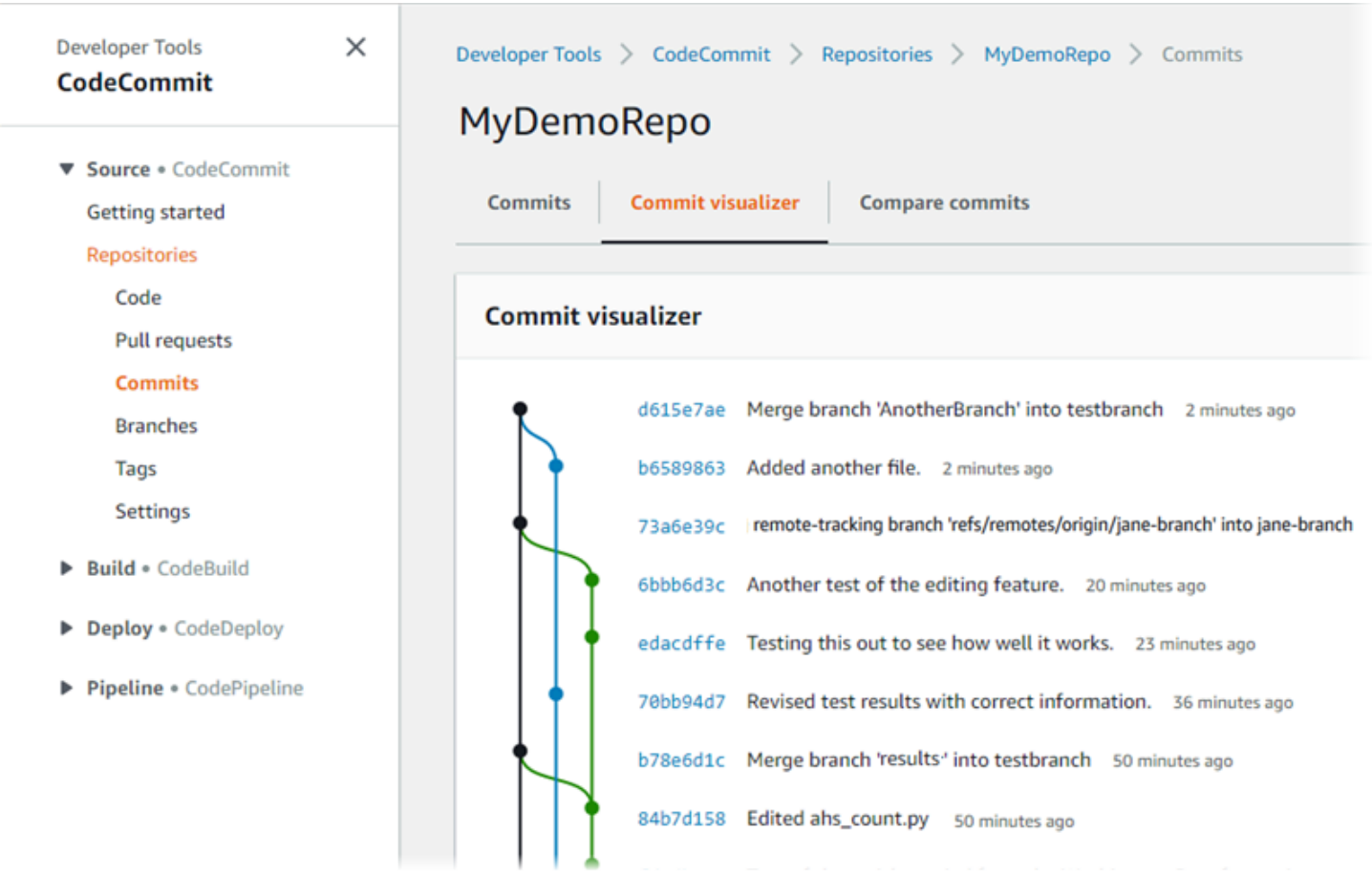
例如，要删除 CodeCommit 存储库`beta`中名为的 Git 标签，名为`origin`：

```
git push origin --delete beta
```

使用 AWS CodeCommit 存储库中的分支

什么是分支？ 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。您可以使用分支开发新功能、从特定提交存储项目的特定版本等。当您创建第一个提交时，系统将为您创建一个默认分支。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。此默认分支的名称因创建第一个提交的方式而异。如果您使用 CodeCommit 控制台、或其中一个 SDK 将第一个文件添加到存储库中，则该默认分支的名称为 main。 AWS CLI这是本指南的示例中使用的默认分支名称。如果您使用 Git 客户端推送第一个提交，则默认分支的名称就是 Git 客户端指定为其默认名称的名称。请考虑将 Git 客户端配置为使用 main 作为初始分支的名称。

在中 CodeCommit，您可以更改存储库的默认分支。您还可以创建和删除分支并查看有关分支的详细信息。您可以快速比较某一分支与默认分支之间 (或任意两个分支之间) 的区别。要查看存储库中分支和合并的历史记录，可以使用提交可视化工具，如下图所示。



有关在中使用存储库其他方面的信息 CodeCommit，请参阅使用存储库使用文件、使用拉取请求、使用提交、和使用用户首选项。

主题

在中创建分支 AWS CodeCommit

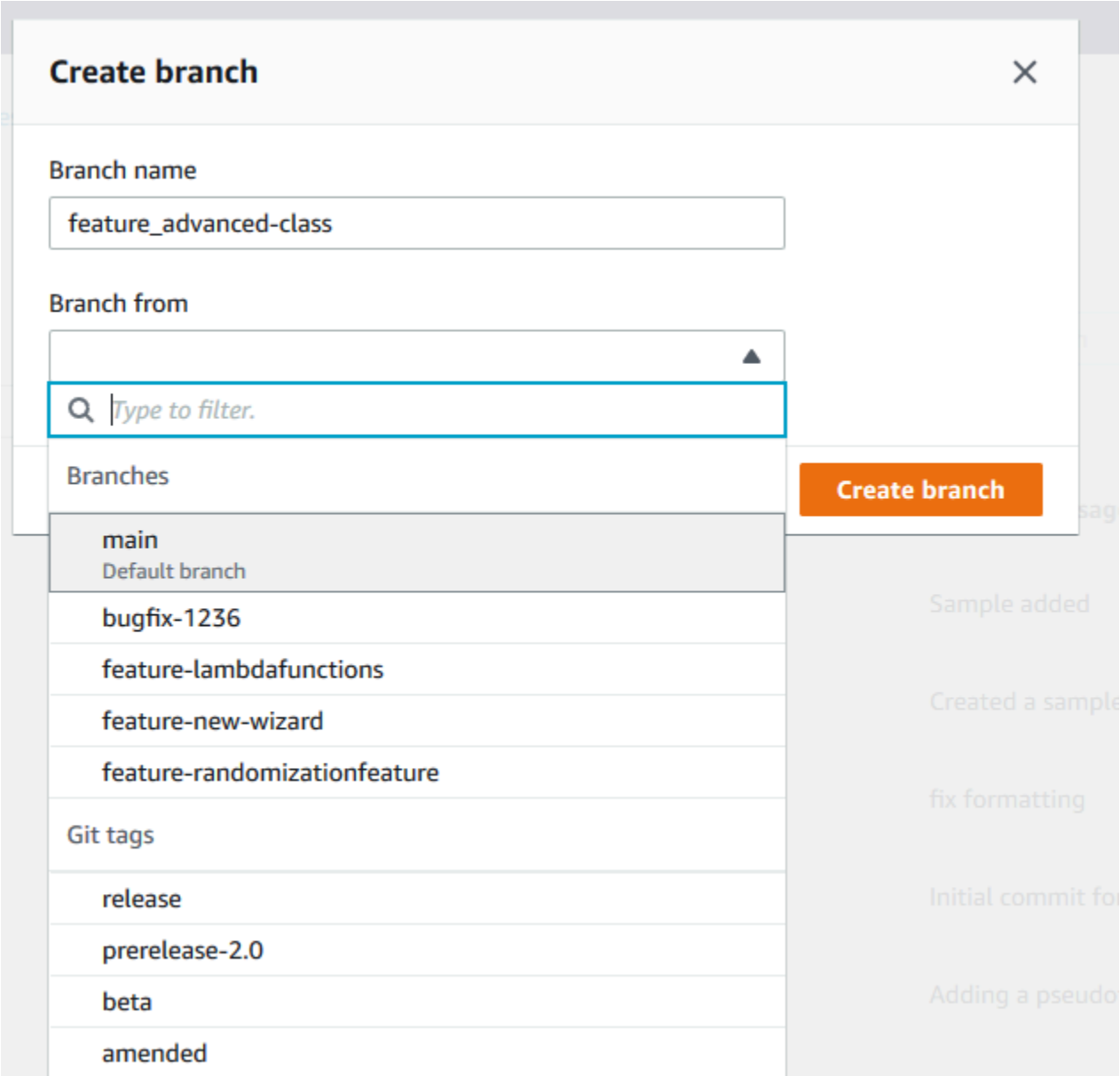
您可以使用 CodeCommit 控制台或为您的存储库创建分支。 AWS CLI 这是分离新的或不同版本文件的工作而不影响默认分支中的工作的快速方式。在 CodeCommit 控制台中创建分支后，必须将该更改拉到本地存储库。或者，你可以在本地创建一个分支，然后使用连接到存储库的本地 CodeCommit 存储库中的 Git 来推送该更改。

主题

创建分支（控制台）

您可以使用 CodeCommit 控制台在 CodeCommit 存储库中创建分支。当用户下次从该存储库中拉取更改时，将看到新分支。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要在其中创建分支的存储库的名称。
3. 在导航窗格中，选择 Branches。
4. 选择创建分支。



在分支名称中，输入分支的名称。在分支来源中，从列表中选择分支或标签，或粘贴提交 ID。选择创建分支。

创建分支 (Git)

按照以下步骤使用本地存储库中的 Git 在本地存储库中创建分支，然后将该分支推送到存储库。CodeCommit

编写这些步骤时假设您已经将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

1. 通过运行git checkout -b ***new-branch-name***命令在本地存储库中创建一个分支，其中*new-branch-name*是新分支的名称。

   例如，下面的命令在本地存储库中创建一个名为 MyNewBranch 的分支：

   ```
   git checkout -b MyNewBranch
   ```

2. 要将新分支从本地存储库推送到 CodeCommit 存储库，请运行git push命令，同时指定***remote-name***和。***new-branch-name***

   例如，要将本地存储库中名为的新分支推送MyNewBranch到带有昵称origin的 CodeCommit 存储库：

   ```
   git push origin MyNewBranch
   ```

> **Note**
>
> 如果向 git push 添加 -u 选项（例如，git push -u origin main），则日后只需运行 git push，而不必指定 *remote-name branch-name*。将设置上游跟踪信息。要获取上游跟踪信息，请运行 git remote show ***remote-name***（例如，git remote show origin）。
> 要查看所有本地和远程跟踪分支的列表，请运行 git branch --all。
> 要在本地存储库中设置与 CodeCommit 存储库中分支相连的分支，请运行git checkout ***remote-branch-name***。

有关更多选项，请参阅 Git 文档。

创建分支 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

按照以下步骤使用在 AWS CLI CodeCommit 存储库中创建分支，然后将该分支推送到 CodeCommit 存储库。有关创建初始提交并为空存储库指定默认分支的名称的步骤，请参阅使用 AWS CLI为存储库创建第一个提交。

1. 运行 create-branch 命令，并指定：

   - 创建分支的 CodeCommit 存储库的名称（带--repository-name选项）。

> **Note**
>
> 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

- 新分支的名称（使用 --branch-name 选项）。
- 新分支指向的提交的 ID（使用 --commit-id 选项）。

例如，要在名为的 CodeCommit 存储库中创建一个`MyNewBranch`指向提交 ID `317f8570EXAMPLE` 的名为的分支`MyDemoRepo`：

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-name MyNewBranch
 --commit-id 317f8570EXAMPLE
```

该命令只在出现错误时生成输出。

2. 要使用新的远程分支名称更新本地 CodeCommit 存储库中可用存储库分支的列表，请运行git remote update ***remote-name***。

   例如，要使用昵称更新 CodeCommit 存储库的可用分支列表，请执行`origin`以下操作：

   ```
   git remote update origin
   ```

   > **Note**
   >
   > 或者，您可以运行 git fetch 命令。您还可以通过运行 git branch --all 来查看所有远程分支，但在更新本地存储库列表之前，您创建的远程分支不会出现在该列表中。
   > 有关更多选项，请参阅 Git 文档。

3. 要在本地存储库中设置与 CodeCommit 存储库中新分支相连的分支，请运行git checkout ***remote-branch-name***。

> **Note**
>
> 要获取 CodeCommit 仓库名称及其 URL 的列表，请运行该git remote -v命令。

限制推送和合并到中的分支 AWS CodeCommit

默认情况下，任何拥有足够权限将代码推送到存储 CodeCommit 库的存储库用户都可以为该仓库中的任何分支做出贡献。无论是使用控制台、命令行或 Git，您以何种方式在存储库中添加分支情况都是如此。但是您可能希望对某一分支进行配置，只允许存储库的一些用户向该分支推送或合并代码。例如，您可能希望配置一个生产代码所用的分支，只有一部分高级开发人员才能在该分支上推送或合并更改。其他开发人员仍可从该分支拉取内容，生成他们自己的分支并创建拉取请求，但他们不能将更改推送或合并到该分支。您可以在 IAM 中创建条件策略，针对一个或多个分支使用上下文键，从而配置此访问权限。

> **Note**
>
> 要完成本主题中的一些步骤，您必须作为管理员用户登录，该用户需拥有配置并应用 IAM 策略的足够权限。有关更多信息，请参阅创建 IAM 管理员用户和组。

主题

配置 IAM 策略以限制针对分支的推送和合并

您可以在 IAM 中创建一个策略，阻止用户更新分支，包括向分支推送提交以及将拉取请求合并到分支。为此，您的策略需使用条件语句，只在满足条件时才应用 Deny 语句的效果。Deny 语句中包含的 API 将决定不允许执行的操作。您可以将此策略配置为只应用于一个存储库中的一个分支，应用于一个存储库中的多个分支，或应用于一个 Amazon Web Services 账户的所有存储库中满足条件的所有分支。

为分支创建条件策略

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 选择创建策略。

4. 选择 JSON，然后粘贴以下示例策略。将 Resource 的值替换为存储库的 ARN，其中包含您希望限制访问权限的分支。将 codecommit:References 的值替换为您希望限制访问权限的一个或多个分支的引用。例如，该策略拒绝针对名为 *main* 的分支和名为 *prod* 的分支（位于名为 *MyDemoRepo* 的存储库中）推送提交、合并分支、删除分支、合并拉取请求和添加文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:PutFile",
                "codecommit:MergeBranchesByFastForward",
                "codecommit:MergeBranchesBySquash",
                "codecommit:MergeBranchesByThreeWay",
                "codecommit:MergePullRequestByFastForward",
                "codecommit:MergePullRequestBySquash",
                "codecommit:MergePullRequestByThreeWay"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main",
                        "refs/heads/prod"
                     ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
        }
    ]
}
```

Git 中的分支只是 HEAD 提交 SHA-1 值的指针 (引用)，这就是条件使用 References 的原因。如果策略的效果是 Deny，而且其中一个操作是 GitPush，则 Null 语句是必需的，这是因为 Git 和将更改从本地存储库推送到时git-receive-pack的工作方式是必需的。 CodeCommit

> **Tip**
>
> 要创建一个策略来应用于 Amazon Web Services 账户的所有存储库中所有名为 main 的分支，请将 `Resource` 的值从存储库 ARN 改为星号 (*)。

5. 选择查看策略。更正策略语句中的所有错误，然后继续创建策略。
6. JSON 经过验证后，将显示创建策略页面。摘要部分出现一条警告，告知此策略不会授予权限。这是预期行为。

   - 在名称中，输入此策略的名称，例如 **`DenyChangesToMain`**。
   - 在描述中，输入策略用途的描述。您可以自由选择，但我们建议您这样做。
   - 选择 创建策略。

将 IAM 策略应用于 IAM 组或角色

您已创建一个策略来限制针对某一分支的推送和合并，但该策略在应用于 IAM 用户、组或角色后才会生效。作为最佳实践，请考虑将该策略应用于 IAM 组或角色。将策略应用于单个 IAM 用户无法很好地扩展。

将条件策略应用于组或角色

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在导航窗格中，如果您希望针对 IAM 组应用策略，请选择组。如果您希望针对用户代入的角色应用策略，请选择角色。选择组或角色的名称。
3. 在权限选项卡上，选择附加策略。
4. 从策略列表中选择您创建的条件策略，然后选择附加策略。

有关更多信息，请参阅附加和分离 IAM 策略。

测试策略

您应测试策略针对组或角色应用的效果，确保与预期效果相同。您可以使用多种方法进行测试。例如，要测试与上述策略类似的策略，您可以：

- 使用 IAM 用户登录 CodeCommit 控制台，该用户要么是已应用策略的 IAM 群组的成员，要么担任已应用该策略的角色。在控制台中，向受限分支添加文件。您在尝试向该分支保存或上传文件时，应看到一条错误消息。将文件添加到其他分支。这次操作应该会成功。
- 使用 IAM 用户登录 CodeCommit 控制台，该用户要么是已应用策略的 IAM 群组的成员，要么担任已应用该策略的角色。创建将合并到受限分支的拉取请求。您应该能够创建拉取请求，但在尝试合并时会出现错误。
- 在终端或命令行中，在适用限制的分支上创建提交，然后将该提交推送到 CodeCommit 存储库。您应该看到一条错误消息。从其他分支进行的提交和推送应该是正常的。

在中查看分行详情 AWS CodeCommit

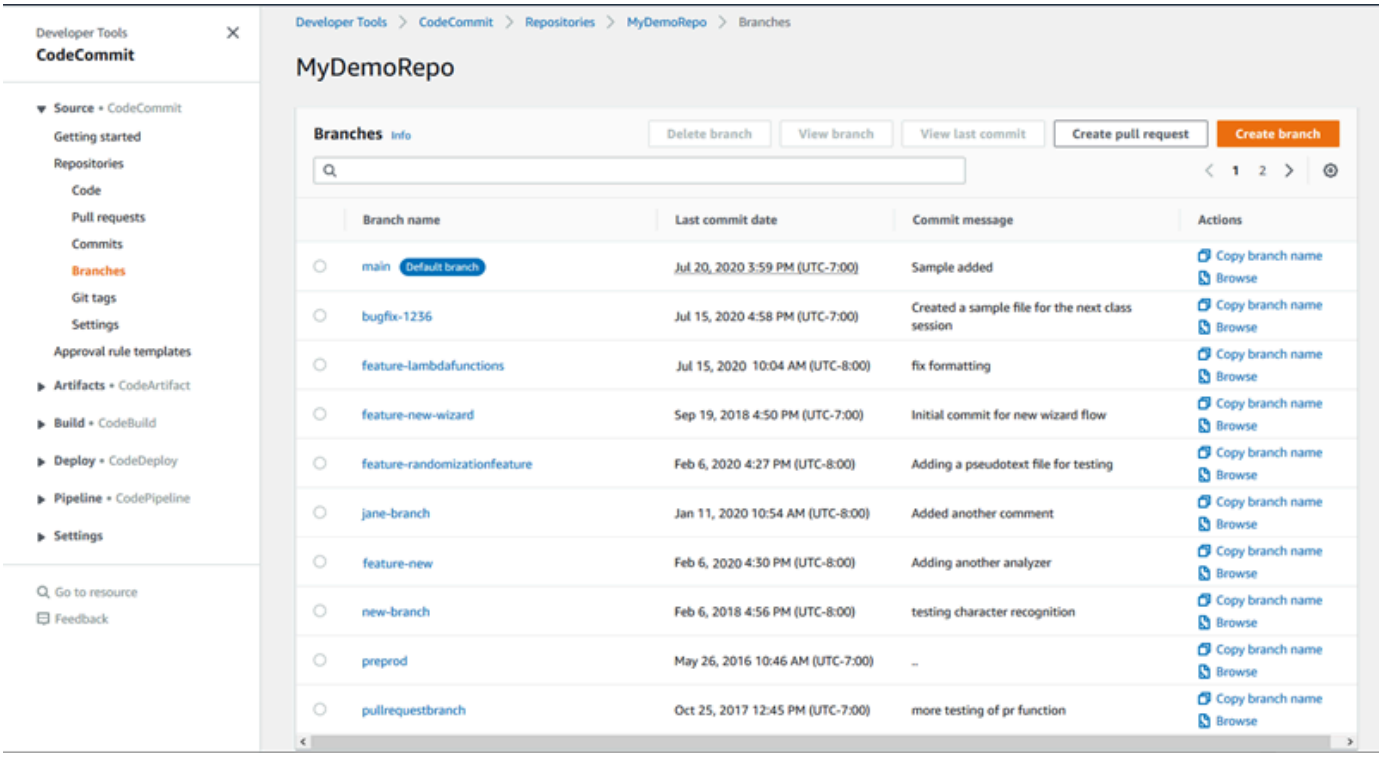
您可以使用 CodeCommit 控制台查看有关 CodeCommit 存储库中分支的详细信息。您可以查看上次提交到分支的日期、提交消息等。您也可以使用连接到存储库的本地 CodeCommit 存储库中的 AWS CLI 或 Git。

主题

查看分支详细信息（控制台）

使用 CodeCommit 控制台快速查看仓库的分支列表以及有关分支的详细信息。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (存储库) 中，选择要在其中查看分支详细信息的存储库的名称。
3. 在导航窗格中，选择 Branches。



4. 存储库默认分支的名称显示在默认分支旁边。要查看有关分支的最近提交的详细信息，请选择该分支，然后选择 View last commit (查看最近的提交)。要查看分支中的文件和代码，请选择分支名称。

查看分支详细信息 (Git)

要使用本地存储库中的 Git 来查看 CodeCommit 仓库的本地和远程跟踪分支的详细信息，请运行git branch命令。

以下步骤是在假设您已经将本地存储库连接到 CodeCommit 存储库的情况下编写的。有关说明，请参阅连接存储库。

1. 运行 git branch 命令，同时指定 --all 选项：

```
git branch --all
```

2. 如果成功，该命令返回类似以下内容的输出：

```
  MyNewBranch
* main
  remotes/origin/MyNewBranch
```

```
  remotes/origin/main
```

当前打开的分支旁边显示有星号 (*)。之后的条目是远程跟踪引用。

> **Tip**
>
> git branch 显示本地分支。
> git branch -r 显示远程分支。
> git checkout ***existing-branch-name*** 切换到指定的分支名称，如果随后立即运行 git branch，则其旁边会显示星号 (*)。
> git remote update ***remote-name***使用可用 CodeCommit 存储库分支列表更新您的本地存储库。（要获取 CodeCommit 存储库名称及其 URL 的列表，请运行git remote -v命令。）

有关更多选项，请参阅 Git 文档。

查看分支详细信息 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

要使用查看有关 CodeCommit 存储库中分支的详细信息，请运行以下一个或多个命令： AWS CLI

- 要查看分支名称列表，请运行 list-branches。
- 要查看有关特定分支的信息，请运行 get-branch。

查看分支名称列表

1. 运行list-branches命令，指定 CodeCommit 存储库的名称（使用`--repository-name`选项）。

   > **Tip**
   >
   > 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

   例如，要查看名为`MyDemoRepo`：的 CodeCommit 存储库中有关分支的详细信息：

   ```
   aws codecommit list-branches --repository-name MyDemoRepo
   ```

2. 如果成功，该命令输出一个 branchNameList 对象以及每个分支的条目。

   下面是前面示例命令的一些示例输出：

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

查看有关分支的信息

1. 运行 get-branch 命令，并指定：

   - 存储库名称（使用 --repository-name 选项）。
   - 分支名称（使用 --branch-name 选项）。

   例如，要查看名为：的 CodeCommit 存储库MyNewBranch中名为的分支的相关信息MyDemoRepo：

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

2. 如果成功，该命令输出分支的名称和推送到该分支的最后一个提交的 ID。

   下面是前面示例命令的一些示例输出：

```
{
    "branch": {
          "branchName": "MyNewBranch",
          "commitID": "317f8570EXAMPLE"
    }
}
```

比较和合并 AWS CodeCommit中的分支

您可以使用 CodeCommit 控制台比较 CodeCommit 存储库中的分支。通过比较分支可帮助您快速查看某分支与默认分支之间的差别，或查看任意两个分支之间的差别。

主题

比较分支与默认分支

使用 CodeCommit 控制台快速查看仓库的分支和默认分支之间的区别。

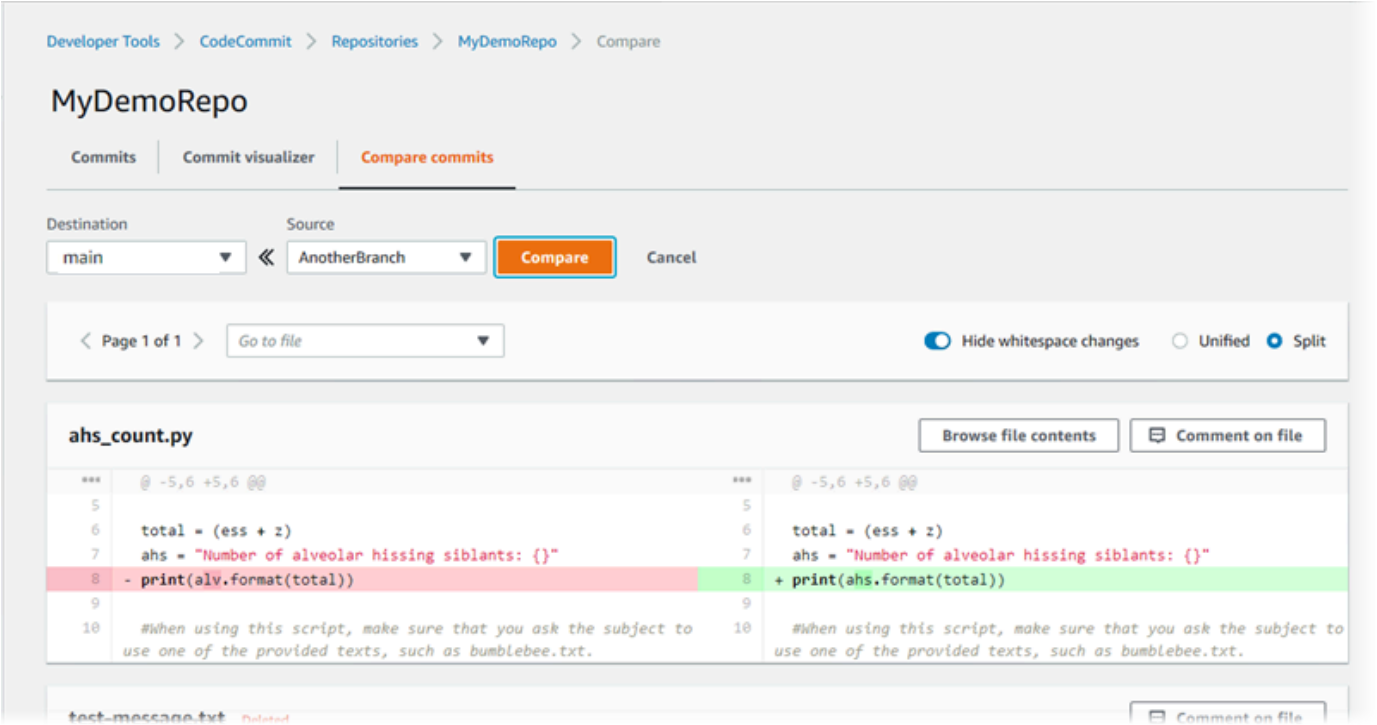
1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要比较其中的分支的存储库的名称。
3. 在导航窗格中，选择提交，然后选择比较提交选项卡。
4. 在目标中，选择默认分支的名称。在源中，选择要与默认分支比较的分支。选择 Compare。

比较两个特定分支

使用 CodeCommit 控制台查看要比较的两个分支之间的差异。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要比较其中的分支的存储库的名称。
3. 在导航窗格中，选择提交，然后选择比较提交选项卡。
4. 在目标和源中，选择要比较的两个分支，然后选择比较。要查看已更改文件的列表，请展开已更改文件列表。您可以通过并排 (拆分视图) 或内联 (统一视图) 方式查看文件中的更改。

> **Note**
>
> 如果您以 IAM 用户身份登录，则可配置并保存用于查看代码的首选项和其他控制台设置。有关更多信息，请参阅 使用用户首选项。



合并两个分支 (AWS CLI)

您可以通过运行以下命令之一，使用可用的合并策略之一来合并 CodeCommit 存储库中的两个分支：AWS CLI

- 要使用快速转发合并策略合并两个分支，请运行 merge-branches-by-fast-forward 命令。
- 要使用 squash 合并策略合并两个分支，请运行 merge-branches-by-squash 命令。
- 要使用三向合并策略合并两个分支，请运行 merge-branches-by-three-way 命令。

您还可以通过运行 create-unreferenced-merge-commit 命令来测试合并。有关更多信息，请参阅解决拉取请求冲突。

> **Note**
>
> 要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

使用合并 CodeCommit 存储库中的两个分支 AWS CLI

1. 要使用快速转发合并策略合并两个分支，请运行 merge-branches-by-fast-forward 命令，并指定：

   - 包含要合并的更改的源分支的名称（使用 --source-commit-specifier 选项）。
   - 要合并更改的目标分支的名称（使用 --destination-commit-specifier 选项）。
   - 存储库的名称（使用 --repository-name 选项）。

   例如，要将名为 *bugfix-1234* 的源分支合并到名为 *prepro* d 的存储库中的目标分支中，名为：*MyDemoRepo*

   ```
   aws codecommit merge-branches-by-fast-forward --source-commit-specifier bugfix-
   bug1234 --destination-commit-specifier preprod --repository-name MyDemoRepo
   ```

   如果成功，该命令产生类似以下内容的输出：

   ```
   {
       "commitId": "4f178133EXAMPLE",
       "treeId": "389765daEXAMPLE"
   }
   ```

2. 要使用 squash 合并策略合并两个分支，请运行 merge-branches-by-squash 命令，并指定：

   - 包含要合并的更改的源分支的名称（使用 --source-commit-specifier 选项）。
   - 要合并更改的目标分支的名称（使用 --destination-commit-specifier 选项）。
   - 存储库的名称（使用 --repository-name 选项）。
   - 要包括的提交消息（使用 --commit-message 选项）。
   - 要用于提交的姓名（使用 --name 选项）。
   - 要用于提交的电子邮件地址（使用 --email 选项）。

   例如，要将名为 bugfix-bug *1234 ###### bugfix-* q *u* artry 的目标分支合并在名为：*MyDemoRepo*

   ```
   aws codecommit merge-branches-by-squash --source-commit-specifier bugfix-bug1234 --
   destination-commit-specifier bugfix-quarterly --author-name "Maria Garcia" --email
   ```

```
 "maria_garcia@example.com" --commit-message "Merging in fix branches to prepare
 for a general patch." --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

3. 要使用三向合并策略合并两个分支，请运行 merge-branches-by-three-way 命令，并指定：

- 包含要合并的更改的源分支的名称（使用 --source-commit-specifier 选项）。
- 要合并更改的目标分支的名称（使用 --destination-commit-specifier 选项）。
- 存储库的名称（使用 --repository-name 选项）。
- 要包括的提交消息（使用 --commit-message 选项）。
- 要用于提交的姓名（使用 --name 选项）。
- 要用于提交的电子邮件地址（使用 --email 选项）。

例如，要将名为 *main* 的源分支与名为 *bugfix-1234* 的目标分支合并在名为：*MyDemoRepo*

```
aws codecommit merge-branches-by-three-way --source-commit-specifier main --
destination-commit-specifier bugfix-bug1234 --author-name "Jorge Souza" --email
 "jorge_souza@example.com" --commit-message "Merging changes from main to bugfix
 branch before additional testing."  --repository-name MyDemoRepo
```

如果成功，该命令产生类似以下内容的输出：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

在中更改分支设置 AWS CodeCommit

您可以在 AWS CodeCommit 控制台中更改哪个分支用作默认分支，也可以通过使用 AWS CLI。例如，如果您使用将默认分支设置为 master 的 Git 客户端创建了第一个提交，则可以创建一个名为 main 的分支，然后更改分支设置，以便将新分支设置为存储库的默认分支。要更改其他分支设置，可以从连接到存储库的本地 CodeCommit 存储库中使用 Git。

主题

更改默认分支（控制台）

您可以在 AWS CodeCommit 控制台中指定哪个分支是 CodeCommit 存储库中的默认分支。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要更改设置的存储库的名称。
3. 在导航窗格中，选择 Settings（设置）。
4. 在默认分支中，选择分支下拉列表，然后选择其他分支。选择保存。

> **Tip**
>
> - 如果在下拉列表中未看到其他分支，则表示您尚未创建任何其他分支。如果存储库只有一个分支，则无法更改存储库的默认分支。有关更多信息，请参阅 在中创建分支 AWS CodeCommit。
> - 如果您没有看到默认分支部分，而是看到通知规则和连接的项目，则说明您位于控制台的常规设置菜单中。存储库的设置菜单列在存储库下，与代码和拉取请求处于同一级别。

更改默认分支 (AWS CLI)

要将 AWS CLI 命令与一起使用 CodeCommit，请安装 AWS CLI。有关更多信息，请参阅 命令行参考。

要使用更改存储库中存储库的分支设置，请运行以下命令： AWS CLI CodeCommit

- update-default-branch，用于更改默认分支。

更改默认分支

1. 运行 update-default-branch 命令，并指定：

   - 更新默认分支的 CodeCommit 存储库的名称（带--repository-name选项）。

     > **Tip**
     > 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

   - 新的默认分支的名称（使用 --default-branch-name 选项）。

     > **Tip**
     > 要获取分支名称，请运行 list-branches 命令。

2. 例如，要将名为的 CodeCommit 存储库MyNewBranch中的默认分支更改为MyDemoRepo：

   ```
   aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-name MyNewBranch
   ```

   该命令只在出现错误时生成输出。

有关更多选项，请参阅 Git 文档。

删除中的分支 AWS CodeCommit

您可以使用 CodeCommit 控制台删除存储库中的分支。删除中的分支 CodeCommit不会删除本地存储库中的该分支，因此用户可能会继续拥有该分支的副本，直到下次拉取更改为止。要在本地删除分支并将更改推送到 CodeCommit 存储库，请使用连接到该仓库的本地 CodeCommit 存储库中的 Git。

删除分支不会删除任何提交，但会删除分支中所有对提交的引用。如果删除了某个分支，该分支包含尚未合并到存储库中另一分支的提交，则除非有完整的提交 ID，否则无法检索这些提交。

> **Note**
>
> 您不能使用本主题中的说明删除存储库的默认分支。如果要删除默认分支，必须先创建一个分支并使新分支成为默认分支，然后才能删除旧分支。有关更多信息，请参阅 创建分支 和 更改分支设置。

主题

删除分支（控制台）

您可以使用 CodeCommit 控制台删除 CodeCommit 存储库中的分支。

1. 打开 CodeCommit 控制台，网址为 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择要在其中删除分支的存储库的名称。
3. 在导航窗格中，选择 Branches。
4. 找到要删除的分支的名称，选择删除分支，然后确认您的选择。

删除分支 (AWS CLI)

如果存储库中的分支不是 CodeCommit 存储库的默认分支，则可以使用删除该分支。 AWS CLI 有关安装和使用的更多信息 AWS CLI，请参阅命令行参考。

1. 在终端或命令行中，运行 delete-branch 命令，并指定：
   - 要删除分支的 CodeCommit 存储库的名称（带--repository-name选项）。

     > **Tip**
     >
     > 要获取 CodeCommit 存储库的名称，请运行列表存储库命令。

   - 要删除的分支的名称（使用 branch-name 选项）。

> **Tip**
>
> 要获取分支名称，请运行 list-branches 命令。

2. 例如，要删除名为的 CodeCommit 存储库MyNewBranch中名为的分支MyDemoRepo：

```
aws codecommit delete-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

此命令返回有关所删除分支的信息，包括所删除分支的名称和作为该分支标头的提交的完整提交 ID。例如：

```
"deletedBranch": {
    "branchName": "MyNewBranch",
    "commitId": "317f8570EXAMPLE"
}
```

删除分支 (Git)

按照以下步骤使用本地存储库中的 Git 删除 CodeCommit 仓库中的分支。

编写这些步骤时假设您已经将本地存储库连接到 CodeCommit 存储库。有关说明，请参阅连接存储库。

1. 要从本地存储库中删除分支，请运行 git branch -D ***branch-name*** 命令，其中的 *branch-name* 是要删除的分支的名称。

> **Tip**
>
> 要获取分支名称列表，请运行 git branch --all。

例如，要删除本地存储库中名为 MyNewBranch 的分支，请运行以下命令：

```
git branch -D MyNewBranch
```

2. 要从 CodeCommit 存储库中删除分支，请运行git push ***remote-name*** --delete ***branch-name***命令，其中 *remote-name* 是本地 CodeCommit 存储库使用的昵称，分###是您要从存储库中删除的分支的名称。 CodeCommit

> **Tip**
>
> 要获取 CodeCommit 仓库名称及其 URL 的列表，请运行该git remote -v命令。

例如，要删除 CodeCommit 存储库MyNewBranch中名为的分支，名为origin：

```
git push origin --delete MyNewBranch
```

> **Tip**
>
> 如果是默认分支，则该命令不会删除分支。

有关更多选项，请参阅 Git 文档。

使用用户首选项

您可以使用 AWS CodeCommit 控制台配置一些默认设置。例如，您可以更改您的首选项，即以内联方式还是在分隔视图中查看代码更改。当您更改其中一个设置时，AWS CodeCommit 控制台会在您的浏览器中设置一个 Cookie，每次您使用控制台时它将存储并应用您的选择。每当您使用该浏览器访问 AWS CodeCommit 控制台时，这些首选项都将应用于所有区域中的所有存储库。这些设置首选项不是存储库或区域特定的。它们对于您与 AWS CLI、AWS CodeCommit API 或（与 AWS CodeCommit 交互的）其他服务之间的交互没有任何影响。

> **Note**
>
> 用户首选项 Cookie 是浏览器特定的。如果您从浏览器中清除 Cookie，则会清除您的首选项。同样，如果您使用不同的浏览器访问存储库，该浏览器将无法访问其他浏览器的 Cookie。您的首选项不会保留。

用户首选项包括：

- 在查看代码更改时，使用 Unified 还是 Split 视图，以及显示还是隐藏空格更改。
- 当查看、编辑或编写代码时，在代码编辑器窗口中使用浅色背景还是黑色背景。

没有用于设置首选项的页面。相反，无论您在控制台中的什么位置更改首选项（例如，您查看代码更改的方式），该更改都将在适当位置保存并应用。

迁移到 AWS CodeCommit

您可以通过多种方法将 Git 存储库迁移到 CodeCommit 存储库：克隆、镜像、迁移所有或部分分支等。您也可以将计算机上的本地、非版本控制内容迁移到 CodeCommit。

以下主题演示了迁移存储库的一些方法。您的步骤可能会有所不同，具体取决于存储库的类型、样式或复杂性以及您针对迁移内容和方式所做的决定。对于非常大的存储库，您可能需要考虑增量迁移。

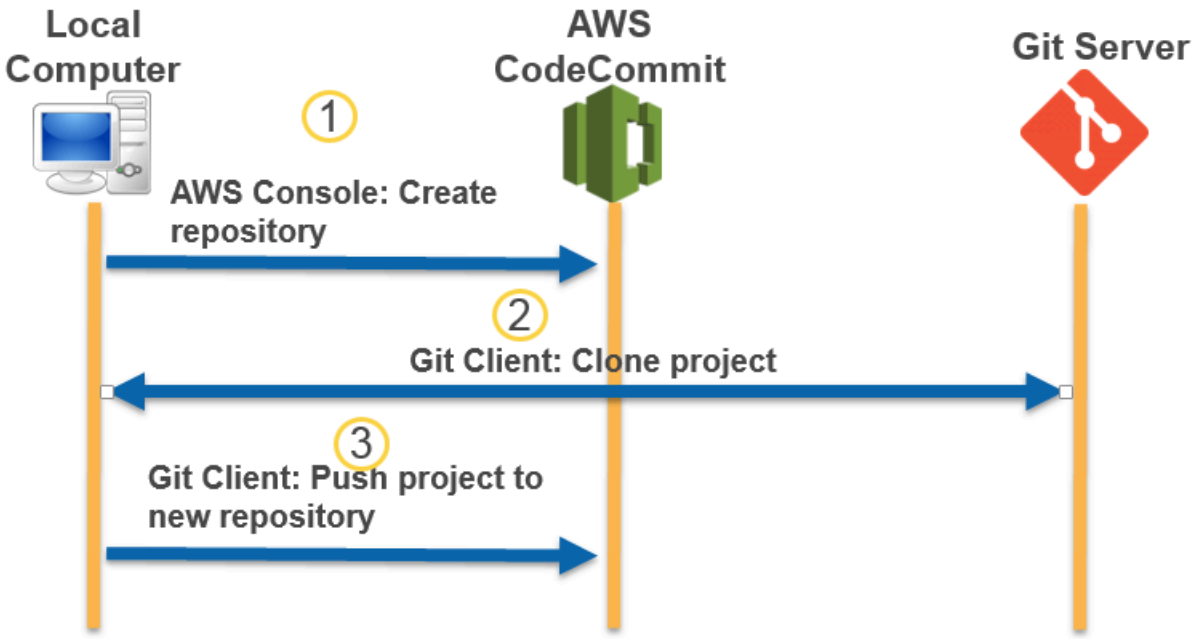
> **Note**
>
> 您可以从其他版本控制系统（例如 Perforce、Subversion、TFS 等）迁移到 CodeCommit，但必须先迁移到 Git。
> 有关更多选项，请参阅 Git 文档。
> 或者，您可以阅读 Scott Chacon 和 Ben Straub 编写的 Pro Git 图书中的迁移到 Git 部分。

主题

将 Git 存储库迁移到 AWS CodeCommit

您可以将现有的 Git 存储库迁移到 CodeCommit 存储库。本主题中的过程将演示如何将托管在另一个 Git 存储库上的项目迁移到 CodeCommit。在该过程中，您将：

- 完成 CodeCommit 所需的初始设置。
- 创建 CodeCommit 存储库。
- 克隆存储库并将其推送到 CodeCommit。
- 查看 CodeCommit 存储库中的文件。
- 与您的团队共享 CodeCommit 存储库。



主题

步骤 0：访问 CodeCommit 所需的设置

在将存储库迁移到 CodeCommit 之前，您必须为 CodeCommit 创建和配置 IAM 用户，并配置您的本地计算机以进行访问。您还应安装 AWS CLI 来管理 CodeCommit。虽然您可以在没有 AWS CLI 的情况下执行大多数 CodeCommit 任务，但它为在命令行或终端中使用 Git 提供了灵活性。

如果您已完成 CodeCommit 所需的设置，请跳到步骤 1：创建 CodeCommit 存储库。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。

2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅 AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

安装和配置 AWS CLI

1. 在本地计算机上，下载并安装 AWS CLI。这是从命令行与 CodeCommit 进行交互的先决条件。我们建议您安装 AWS CLI 版本 2。它是 AWS CLI 的最新主版本，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书与 git-remote-codecommit 的 AWS CLI 版本。

   有关更多信息，请参阅使用 AWS 命令行界面进行设置。

   > **Note**
   >
   > CodeCommit 仅适用于 AWS CLI 1.7.38 及更高版本。作为最佳做法，请安装 AWS CLI 或将其升级到可用的最新版本。要确定您安装的 AWS CLI 的版本，请运行 aws --version 命令。

要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证 AWS CLI 中的 CodeCommit 命令是否已安装：

```
aws codecommit help
```

该命令返回 CodeCommit 命令的列表。

3. 使用 configure 命令通过配置文件来配置 AWS CLI，如下所示：

```
aws configure
```

出现提示时，指定要用于 CodeCommit 的 IAM 用户的 AWS 访问密钥和 AWS 秘密访问密钥。此外，请务必指定存储库所在的 AWS 区域，例如 us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

有关创建和配置相关配置文件以与 AWS CLI 配合使用的详细信息，请参阅以下内容：

- 命名配置文件
- 在 AWS CLI 中使用 IAM 角色
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到另一个 AWS 区域中的存储库或资源，您必须使用默认区域名称重新配置 AWS CLI。CodeCommit 支持的默认区域名称包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2

- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

有关 CodeCommit 和 AWS 区域的更多信息，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

接下来，您必须安装 Git。

- 对于 Linux、macOS 或 Unix：

  要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

  要安装 Git，建议您访问 Git 下载等网站。

  > **Note**
  >
  > Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

- 对于 Windows：

  要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

  要安装 Git，建议访问 Git for Windows 等网站。如果您使用此链接安装 Git，则可以接受除以下设置之外的所有安装默认设置：

  - 在调整 PATH 环境步骤中出现提示时，从命令行中选择使用 Git 的选项。
  - （可选）如果您打算将 HTTPS 与 AWS CLI 中包含的凭证助手一起使用，而不是为 CodeCommit 配置 Git 凭证，请在配置额外选项页面上，确保清除启用 Git Credential Manager 选项。仅当 IAM 用户配置 Git 凭证时，Git Credential Manager 才与 CodeCommit 兼容。有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户 和 Windows 版 Git：我安装了 Windows 版 Git，但在访问我的存储库时被系统拒绝 (403)：

  > **Note**
  >
  > Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

CodeCommit 支持 HTTPS 和 SSH 身份验证。要完成设置，您必须配置用于 CodeCommit 的 Git 凭证（建议大多数用户使用 HTTPS）、访问 CodeCommit 时使用的 SSH 密钥对 (SSH)、git-remote-codecommit（建议使用联合身份访问的用户采用）或 AWS CLI 中包含的凭证助手 (HTTPS)。

- 有关在所有支持的操作系统上设置 Git 凭证的信息，请参阅步骤 3：创建 Git 凭据，以便通过 HTTPS 连接到 CodeCommit。
- 有关在 Linux、macOS 或 Unix 上设置 SSH 的信息，请参阅SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥。
- 有关在 Windows 上设置 SSH 的信息，请参阅步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥。
- 对于 git-remote-codecommit，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。
- 有关 Linux、macOS 或 Unix 上的凭证助手，请参阅设置凭证助手（Linux、macOS 或 Unix）。
- 有关在 Windows 上设置凭证辅助程序的信息，请参阅设置凭证辅助程序 (Windows)。

步骤 1：创建 CodeCommit 存储库

在本部分中，您将使用 CodeCommit 控制台创建在本教程的其余部分中使用的 CodeCommit 存储库。要使用 AWS CLI 创建存储库，请参阅创建存储库 (AWS CLI)。

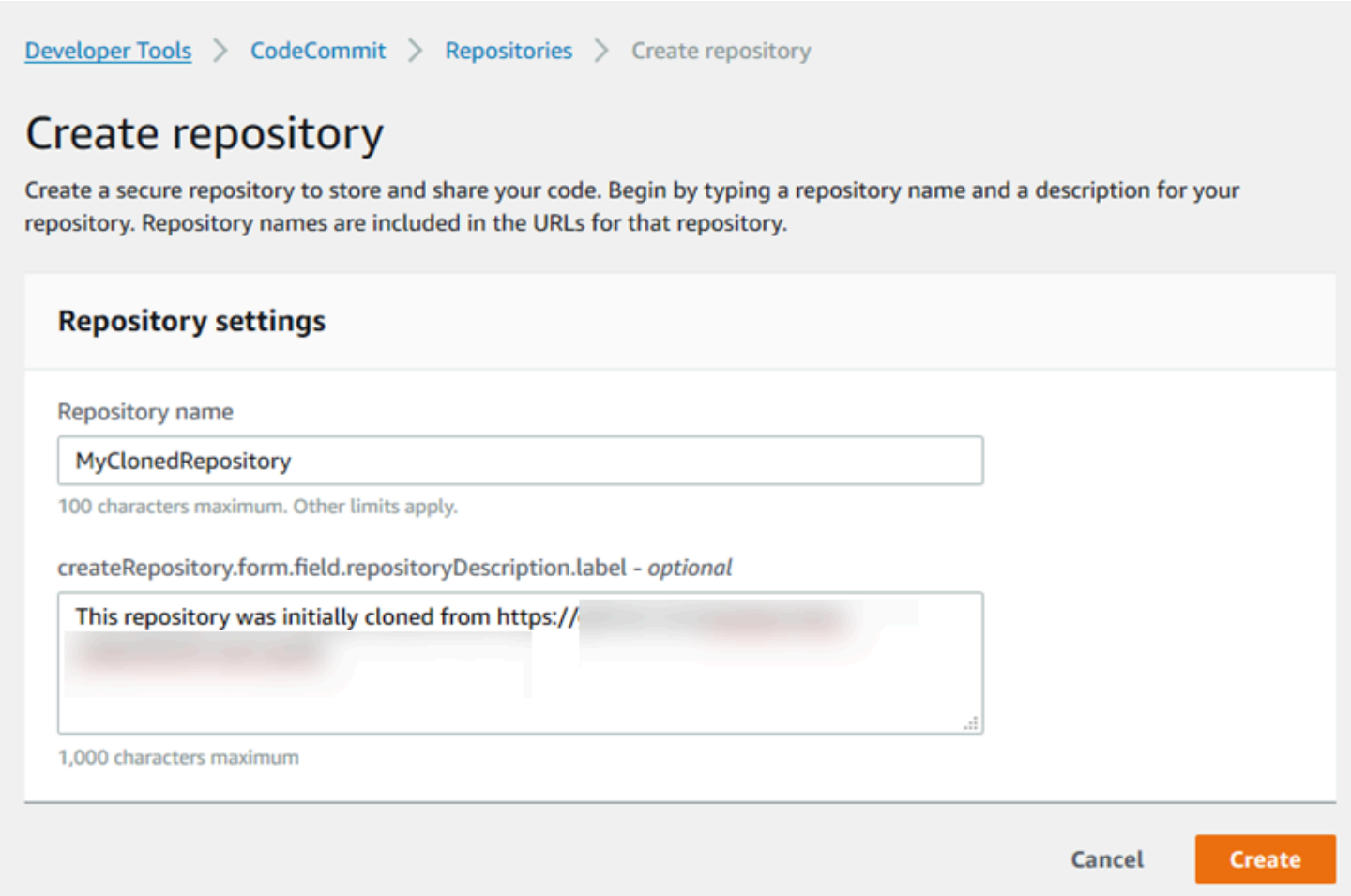
1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择要在其中创建存储库的 AWS 区域。有关更多信息，请参阅 区域和 Git 连接端点。
3. 在存储库页面上，选择创建存储库。
4. 在创建存储库页面上的存储库名称中，为存储库输入名称。

> **Note**
>
> 存储库名称区分大小写。名称在 AWS 区域中对于 Amazon Web Services 账户必须唯一。

5. （可选）在描述中，输入存储库的描述。这可以帮助您及其他用户了解存储库的用途。

> **Note**
>
> “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用 `GetRepository` 或 `BatchGetRepositories` API 的应用程序开发人员，并且打算在 Web 浏览器中显示“存储库描述”字段，请参阅 CodeCommit API 参考。

6. （可选）选择添加标签，向您的存储库添加一个或多个存储库标签（自定义属性标签，可帮助您组织和管理您的 AWS 资源）。有关更多信息，请参阅在中标记存储库 AWS CodeCommit。
7. （可选）展开其他配置，以指定是使用默认 AWS 托管式密钥还是您自己的客户托管密钥来加密和解密此存储库中的数据。如果您选择使用自己的客户托管密钥，则必须确保该密钥在您创建存储库的 AWS 区域可用，并且该密钥处于活跃状态。有关更多信息，请参阅AWS Key Management Service 和 AWS CodeCommit 存储库加密。
8. （可选）如果此存储库包含 Java 或 Python 代码，并且您希望 CodeGuru Reviewer 对其进行分析，请选择启用适用于 Java 和 Python 的 Amazon CodeGuru Reviewer。CodeGuru Reviewer 使用多个机器学习模型来查找代码缺陷，并对拉取请求提供改进和修复建议。有关更多信息，请参阅Amazon CodeGuru Reviewer 用户指南。
9. 选择创建。



创建后，存储库将显示在存储库列表中。在 URL 列中，选择复制图标，然后选择用于连接到 CodeCommit 的协议（SSH 或 HTTPS）。复制 URL。

例如，如果将存储库命名为 *MyClonedRepository*，并且在美国东部（俄亥俄州）区域中结合使用 Git 凭证和 HTTPS，则 URL 如下所示：

```
https://git-codecommit.us-east-2.amazonaws.com/MyClonedRepository
```

稍后在步骤 2：克隆存储库并将其推送到 CodeCommit 存储库中您将需要该 URL。

步骤 2：克隆存储库并将其推送到 CodeCommit 存储库

在本部分中，您将 Git 存储库克隆到本地计算机，从而创建所谓的本地存储库。然后，您将把本地存储库的内容推送到您之前创建的 CodeCommit 存储库。

1. 在本地计算机上的终端或命令提示符中，运行带 `--mirror` 选项的 git clone 命令，以将远程存储库的裸副本克隆到名为 *aws-codecommit-demo* 的新文件夹中。这是仅用于迁移的裸存储库。它

不是用于与 CodeCommit 中的已迁移存储库交互的本地存储库。稍后，在完成到 CodeCommit 的迁移后，您可以创建该本地存储库。

以下示例将 GitHub (*https://github.com/awslabs/aws-demo-php-simple-app.git*) 上托管的演示应用程序克隆到名为 *aws-codecommit-demo* 的目录中的本地存储库。

```
git clone --mirror https://github.com/awslabs/aws-demo-php-simple-app.git aws-
codecommit-demo
```

2. 切换到您执行克隆操作的目录。

```
cd aws-codecommit-demo
```

3. 运行 git push 命令，指定目标 CodeCommit 存储库的 URL 和名称以及 --all 选项。（这是您在步骤 1：创建 CodeCommit 存储库中复制的 URL）。

例如，如果将存储库命名为 *MyClonedRepository* 并设置为使用 HTTPS，则将运行以下命令：

```
git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
 --all
```

> **Note**
>
> --all 选项仅推送该存储库的所有分支。它不会推送其他引用，例如标签。如果您想要推送标签，请等待初始推送完成，然后再次推送，这次使用 --tags 选项：
>
> ```
> git push ssh://git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyClonedRepository --tags
> ```
>
> 有关更多信息，请参阅 Git 网站上的 Git 推送。有关推送大型存储库的信息，特别是一次性推送所有引用的情况（例如，使用 --mirror 选项），请参阅以增量方式迁移存储库。

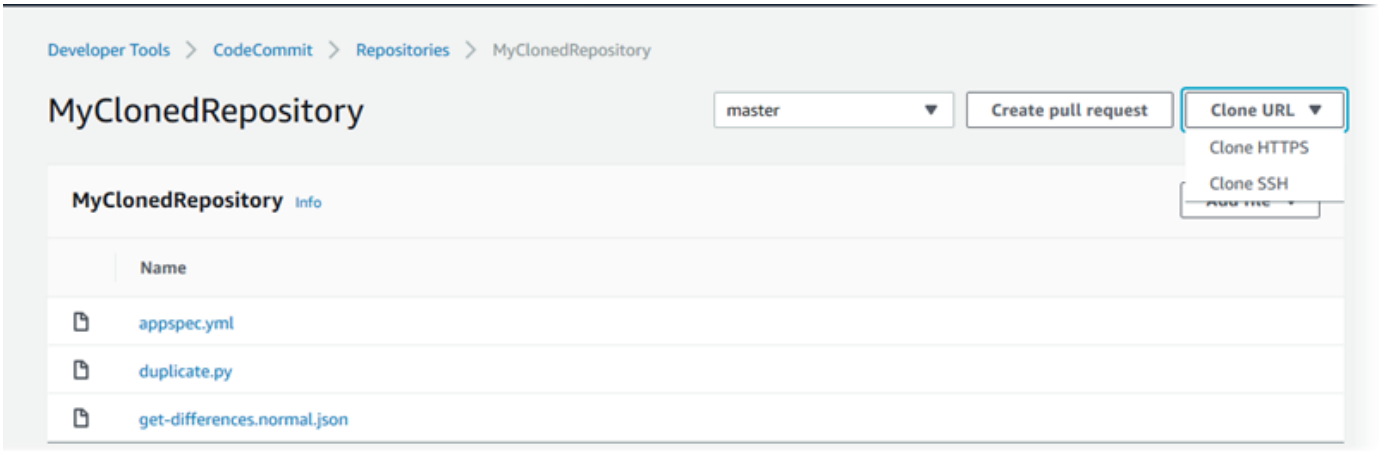
将存储库迁移到 CodeCommit 之后，便可删除 *aws-codecommit-demo* 文件夹以及其中的内容。要创建本地存储库，在其中包含用于使用 CodeCommit 中的存储库的所有正确引用，请运行不带 `--mirror` 选项的 `git clone` 命令：

```
git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
```

步骤 3：查看 CodeCommit 中的文件

推送完目录内容后，可以使用 CodeCommit 控制台快速查看存储库中的所有文件。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，选择存储库的名称（例如，*MyClonedRepository*）。
3. 查看存储库中的文件，以了解分支、克隆 URL、设置等信息。



步骤 1：共享 CodeCommit 存储库

在 CodeCommit 中创建存储库时，会生成两个端点：一个用于 HTTPS 连接，一个用于 SSH 连接。两者都能提供安全的网络连接。您的用户可以使用这两种协议中的任何一种。不管您向用户推荐哪种协议，这两种终端节点都保持有效。在与他人共享您的存储库之前，您必须创建允许其他用户访问该存储库的 IAM policy。向您的用户提供下述访问说明。

为存储库创建客户托管策略

1. 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/。
2. 在 Dashboard 导航区域中选择 Policies，然后选择 Create Policy。
3. 在创建策略页面上，选择导入管理型策略。
4. 在导入管理型策略页面的筛选策略中，输入 **AWSCodeCommitPowerUser**。选择策略名称旁的按钮，然后选择导入。
5. 在创建策略页面上，选择 JSON。将 CodeCommit 操作的 `Resource` 行的“*”部分替换为 CodeCommit 存储库的 Amazon 资源名称 (ARN)，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> 要查找 CodeCommit 存储库的 ARN，请转到 CodeCommit 控制台，从列表中选择存储库名称，然后选择设置。有关更多信息，请参阅查看存储库详细信息。

若要将该策略应用到多个存储库，请通过指定其 ARN 将各个存储库添加为资源。在每个资源语句之间加上逗号，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

完成编辑后，选择查看策略。

6. 在查看策略页面上的名称中，输入策略的新名称（例如 *AWSCodeCommitPowerUser-MyDemoRepo*）。（可选）提供此策略的描述。
7. 选择创建策略。

要管理对您的存储库的访问，请为其用户创建一个 IAM 组、向该组添加 IAM 用户，然后附加在上一步中创建的客户管理型策略。附加访问所需的任何其他策略，例如 IAMUserSSHKeys 或 IAMSelfManageServiceSpecificCredentials。

1. 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/。
2. 在 Dashboard 导航区域中选择 Groups，然后选择 Create New Group。
3. 在设置组名页面上的组名中，为组输入名称（例如 *MyDemoRepoGroup*），然后选择下一步。请考虑在组名称中包含存储库名称。

> **Note**
>
> 该名称必须在 Amazon Web Services 账户间保持唯一。

4. 选中您在前一部分中创建的客户管理型策略旁边的复选框（例如 AWSCodeCommitPowerUser-MyDemoRepo）。
5. 在 Review 页面上，选择 Create Group。IAM 将使用已附加的指定策略创建此组。此组会显示在与您的 Amazon Web Services 账户关联的组列表中。
6. 从列表中选择您的组。
7. 在组摘要页面上，选择用户 选项卡，然后选择向组添加多个用户。在显示与您的 Amazon Web Services 账户关联的所有用户的列表中，选中要允许其访问 CodeCommit 存储库的用户旁边的复选框，然后选择添加用户。

> **Tip**
>
> 您可以使用搜索框快速地按名称查找用户。

8. 添加用户后，关闭 IAM 控制台。

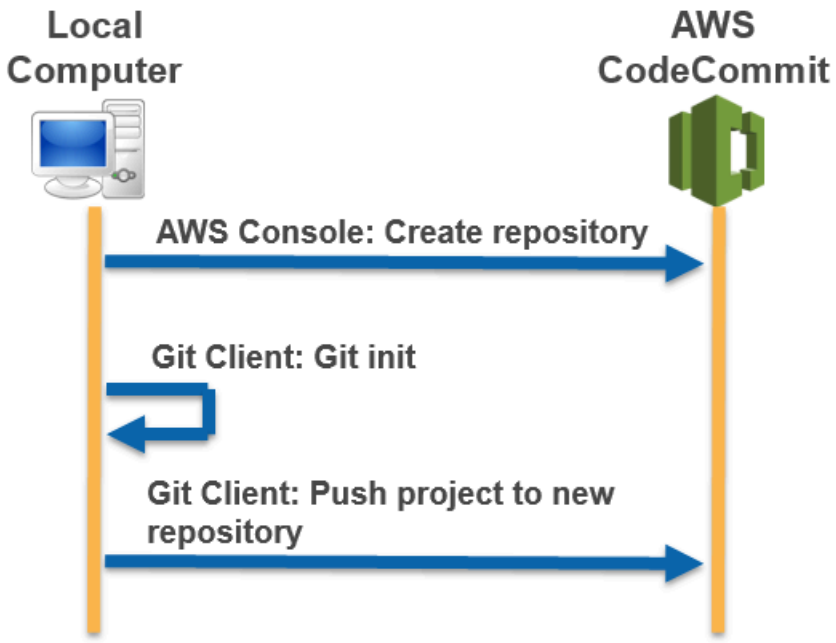
在创建 IAM 用户以使用您配置的策略组和策略访问 CodeCommit 后，向该用户发送连接到存储库所需的信息。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 在 Repositories (存储库) 页面上，选择要共享的存储库。
4. 在克隆 URL 中，选择您希望用户使用的协议。这会复制连接协议的克隆 URL。
5. 向您的用户发送克隆 URL 以及任何其他说明，如安装 AWS CLI、对配置文件进行配置或安装 Git 的说明。请确保包含连接协议（例如 HTTPS）的配置信息。

将本地或非版本控制内容迁移到 AWS CodeCommit

本主题中的过程演示如何将计算机上的现有项目或本地内容迁移到 CodeCommit 存储库。在该过程中，您将：

- 完成 CodeCommit 所需的初始设置。
- 创建 CodeCommit 存储库。
- 使某个本地文件夹置于 Git 版本控制之下，并将该文件夹的内容推送到 CodeCommit 存储库。
- 查看 CodeCommit 存储库中的文件。
- 与您的团队共享 CodeCommit 存储库。



主题

步骤 0：访问 CodeCommit 所需的设置

在将本地内容迁移到 CodeCommit 之前，您必须为 CodeCommit 创建和配置 IAM 用户，并配置您的本地计算机以进行访问。您还应安装 AWS CLI 来管理 CodeCommit。虽然您可以在没有 AWS CLI 的情况下执行大多数 CodeCommit 任务，但它为使用 Git 提供了灵活性。

如果您已完成 CodeCommit 所需的设置，请跳到步骤 1：创建 CodeCommit 存储库。

创建和配置用于访问 CodeCommit 的 IAM 用户

1. 前往 http://aws.amazon.com，并选择注册，创建一个 Amazon Web Services 账户。
2. 创建 IAM 用户或使用您的 Amazon Web Services 账户中的现有用户。确保您具有与该 IAM 用户关联的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅在 Amazon Web Services 账户中创建 IAM 用户。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果使用现有的 IAM 用户，请确保未向该用户附加明确拒绝 CodeCommit 所需的 AWS KMS 操作的策略。有关更多信息，请参阅 AWS KMS 和加密。

3. 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/。
4. 在 IAM 控制台的导航窗格中，选择用户，然后选择要配置进行 CodeCommit 访问的 IAM 用户。
5. 在 Permissions 选项卡上，选择 Add Permissions。
6. 在 Grant permissions (授予权限) 中，选择 Attach existing policies directly (直接附加现有策略)。
7. 从策略列表中选择 AWSCodeCommitPowerUser 或用于 CodeCommit 访问的其他托管策略。有关更多信息，请参阅适用于 CodeCommit 的 AWS 托管式策略。

   选择要附加的策略后，选择下一步：审核以审核要附加到 IAM 用户的策略列表。如果列表正确，选择 Add permissions。

   有关 CodeCommit 托管策略以及与其他组和用户共享访问存储库的更多信息，请参阅共享存储库和 AWS CodeCommit 的身份验证和访问控制。

安装和配置 AWS CLI

1. 在本地计算机上，下载并安装 AWS CLI。这是从命令行与 CodeCommit 进行交互的先决条件。我们建议您安装 AWS CLI 版本 2。它是 AWS CLI 的最新主版本，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书与 git-remote-codecommit 的 AWS CLI 版本。

   有关更多信息，请参阅使用 AWS 命令行界面进行设置。

   > **Note**
   >
   > CodeCommit 仅适用于 AWS CLI 1.7.38 及更高版本。作为最佳做法，请安装 AWS CLI 或将其升级到可用的最新版本。要确定您安装的 AWS CLI 的版本，请运行 aws --version 命令。
   > 要将旧版本的 AWS CLI 升级到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证 AWS CLI 中的 CodeCommit 命令是否已安装：

   ```
   aws codecommit help
   ```

   该命令返回 CodeCommit 命令的列表。

3. 使用 configure 命令通过配置文件来配置 AWS CLI，如下所示：

   ```
   aws configure
   ```

   出现提示时，指定要用于 CodeCommit 的 IAM 用户的 AWS 访问密钥和 AWS 秘密访问密钥。此外，请务必指定存储库所在的 AWS 区域，例如 us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

   ```
   AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
    Enter
   AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
    then press Enter
   Default region name [None]: Type a supported region for CodeCommit here, and then
    press Enter
   Default output format [None]: Type json here, and then press Enter
   ```

   有关创建和配置相关配置文件以与 AWS CLI 配合使用的详细信息，请参阅以下内容：

   - 命名配置文件

- 在 AWS CLI 中使用 IAM 角色
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到另一个 AWS 区域中的存储库或资源，您必须使用默认区域名称重新配置 AWS CLI。CodeCommit 支持的默认区域名称包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1
- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1

- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

有关 CodeCommit 和 AWS 区域的更多信息，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

接下来，您必须安装 Git。

- 对于 Linux、macOS 或 Unix：

  要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

  要安装 Git，建议您访问 Git 下载等网站。

  > **Note**
  >
  > Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

- 对于 Windows：

  要使用 CodeCommit 存储库中的文件、提交和其他信息，您必须在本地计算机上安装 Git。CodeCommit 支持 Git 1.7.9 及更高版本。Git 版本 2.28 支持为初始提交配置分支名称。我们建议使用最新版本的 Git。

  要安装 Git，建议访问 Git for Windows 等网站。如果您使用此链接安装 Git，则可以接受除以下设置之外的所有安装默认设置：

  - 在调整 PATH 环境步骤中出现提示时，从命令行中选择使用 Git 的选项。
  - （可选）如果您打算将 HTTPS 与 AWS CLI 中包含的凭证助手一起使用，而不是为 CodeCommit 配置 Git 凭证，请在配置额外选项页面上，确保清除启用 Git Credential Manager 选项。仅当 IAM 用户配置 Git 凭证时，Git Credential Manager 才与 CodeCommit 兼容。有关更多信息，请参阅

适用于使用 Git 凭证的 HTTPS 用户 和 Windows 版 Git：我安装了 Windows 版 Git，但在访问我的存储库时被系统拒绝 (403)：

> **Note**
>
> Git 是一个不断发展的平台，会定期进行更新。有时，功能上的更改可能会影响到它与 CodeCommit 协作的方式。如果在使用特定版本的 Git 和 CodeCommit 时遇到问题，请参阅故障排除中的信息。

CodeCommit 支持 HTTPS 和 SSH 身份验证。要完成设置，您必须配置用于 CodeCommit 的 Git 凭证（建议大多数用户使用 HTTPS）、访问 CodeCommit 时使用的 SSH 密钥对 (SSH)、git-remote-codecommit（建议使用联合身份访问的用户采用）或 AWS CLI 中包含的凭证助手。

- 有关在所有支持的操作系统上设置 Git 凭证的信息，请参阅步骤 3：创建 Git 凭据，以便通过 HTTPS 连接到 CodeCommit。
- 有关在 Linux、macOS 或 Unix 上设置 SSH 的信息，请参阅SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥。
- 有关在 Windows 上设置 SSH 的信息，请参阅步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥。
- 对于 git-remote-codecommit，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。
- 有关 Linux、macOS 或 Unix 上的凭证助手，请参阅设置凭证助手（Linux、macOS 或 Unix）。
- 有关在 Windows 上设置凭证辅助程序的信息，请参阅设置凭证辅助程序 (Windows)。

步骤 1：创建 CodeCommit 存储库

在本部分中，您将使用 CodeCommit 控制台创建在本教程的其余部分中使用的 CodeCommit 存储库。要使用 AWS CLI 创建存储库，请参阅创建存储库 (AWS CLI)。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择要在其中创建存储库的 AWS 区域。有关更多信息，请参阅 区域和 Git 连接端点。
3. 在存储库页面上，选择创建存储库。
4. 在创建存储库页面上的存储库名称中，为存储库输入名称。

> **Note**
>
> 存储库名称区分大小写。名称在 AWS 区域中对于 Amazon Web Services 账户必须唯一。

5. （可选）在描述中，输入存储库的描述。这可以帮助您及其他用户了解存储库的用途。

> **Note**
>
> “描述”字段在控制台中显示“Markdown”，并接受所有 HTML 字符和有效的 Unicode 字符。如果您是使用 `GetRepository` 或 `BatchGetRepositories` API 的应用程序开发人员，并且打算在 Web 浏览器中显示“存储库描述”字段，请参阅 CodeCommit API 参考。

6. （可选）选择添加标签，向您的存储库添加一个或多个存储库标签（自定义属性标签，可帮助您组织和管理您的 AWS 资源）。有关更多信息，请参阅在中标记存储库 AWS CodeCommit。
7. （可选）展开其他配置，以指定是使用默认 AWS 托管式密钥还是您自己的客户托管密钥来加密和解密此存储库中的数据。如果您选择使用自己的客户托管密钥，则必须确保该密钥在您创建存储库的 AWS 区域可用，并且该密钥处于活跃状态。有关更多信息，请参阅AWS Key Management Service 和 AWS CodeCommit 存储库加密。
8. （可选）如果此存储库包含 Java 或 Python 代码，并且您希望 CodeGuru Reviewer 对其进行分析，请选择启用适用于 Java 和 Python 的 Amazon CodeGuru Reviewer。CodeGuru Reviewer 使用多个机器学习模型来查找代码缺陷，并对拉取请求提供改进和修复建议。有关更多信息，请参阅Amazon CodeGuru Reviewer 用户指南。
9. 选择创建。

创建后，存储库将显示在存储库列表中。在 URL 列中，选择复制图标，然后选择用于连接到 CodeCommit 的协议（HTTPS 或 SSH）。复制 URL。

例如，如果将存储库命名为 *MyFirstRepo* 并且使用 HTTPS，则 URL 将如下所示：

```
https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo
```

稍后在步骤 2：将本地内容迁移到 CodeCommit 存储库中您将需要该 URL。

步骤 2：将本地内容迁移到 CodeCommit 存储库

现在您有了一个 CodeCommit 存储库，可以在本地计算机上选择一个目录，将其转换为本地 Git 存储库。可以使用 git init 命令将现有的非版本控制内容转化为 Git 存储库，或在尚无任何文件或内容的情况下初始化一个新的空存储库。

1. 在本地计算机上的终端或命令行中，进入要用作存储库源的目录。
2. 运行以下命令将 Git 配置为使用名为 **main** 的默认分支：

   ```
   git config --local init.defaultBranch main
   ```

   您也可以运行以下命令，将所有新创建的存储库的默认分支名称设置为 **main**：

   ```
   git config --global init.defaultBranch main
   ```
3. 运行 git init 命令在该目录中初始化 Git 版本控制。这会在启用版本控制跟踪的目录的根目录中创建一个 .git 子目录。.git 文件夹还包含存储库的所有必需元数据。

   ```
   git init
   ```
4. 运行以下命令，检查初始化目录的状态：

   ```
   git status
   ```

   添加要置于版本控制之下的文件。在本教程中，将运行带 . 说明符的 `git add` 命令来添加该目录中的所有文件。有关其他选项的信息，请参阅 Git 文档。

   ```
   git add .
   ```
5. 为添加的文件创建一个带有提交消息的提交。

   ```
   git commit -m "Initial commit"
   ```
6. 运行 git push 命令，指定目标 CodeCommit 存储库的 URL 和名称以及 `--all` 选项。（这是您在步骤 1：创建 CodeCommit 存储库中复制的 URL。）

   例如，如果将存储库命名为 *MyFirstRepo* 并设置为使用 HTTPS，则将运行以下命令：

   ```
   git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo --all
   ```

步骤 3：查看 CodeCommit 中的文件

推送完目录内容后，可以使用 CodeCommit 控制台快速查看存储库中的所有文件。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存储库中，从列表中选择存储库的名称（例如，*MyFirstRepository*）。
3. 查看存储库中的文件，以了解分支、克隆 URL、设置等信息。

步骤 1：共享 CodeCommit 存储库

在 CodeCommit 中创建存储库时，会生成两个端点：一个用于 HTTPS 连接，一个用于 SSH 连接。两者都能提供安全的网络连接。您的用户可以使用这两种协议中的任何一种。不管您向用户推荐哪种协议，这两种终端节点都保持有效。在与他人共享您的存储库之前，您必须创建允许其他用户访问该存储库的 IAM policy。向您的用户提供下述访问说明。

为存储库创建客户托管策略

1. 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/。
2. 在 Dashboard 导航区域中选择 Policies，然后选择 Create Policy。
3. 在创建策略页面上，选择导入管理型策略。
4. 在导入管理型策略页面的筛选策略中，输入 **AWSCodeCommitPowerUser**。选择策略名称旁的按钮，然后选择导入。
5. 在创建策略页面上，选择 JSON。将 CodeCommit 操作的 `Resource` 行的“*”部分替换为 CodeCommit 存储库的 Amazon 资源名称 (ARN)，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
 ]
```

> **Tip**
>
> 要查找 CodeCommit 存储库的 ARN，请转到 CodeCommit 控制台，从列表中选择存储库名称，然后选择设置。有关更多信息，请参阅查看存储库详细信息。

若要将该策略应用到多个存储库，请通过指定其 ARN 将各个存储库添加为资源。在每个资源语句之间加上逗号，如下所示：

```
"Resource": [
 "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
 "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
 ]
```

完成编辑后，选择查看策略。

6. 在查看策略页面上的名称中，输入策略的新名称（例如 *AWSCodeCommitPowerUser-MyDemoRepo*）。（可选）提供此策略的描述。
7. 选择创建策略。

要管理对您的存储库的访问，请为其用户创建一个 IAM 组、向该组添加 IAM 用户，然后附加在上一步中创建的客户管理型策略。附加访问所需的任何其他策略，例如 `IAMSelfManageServiceSpecificCredentials` 或 `IAMUserSSHKeys`。

1. 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/。
2. 在 Dashboard 导航区域中选择 Groups，然后选择 Create New Group。
3. 在设置组名页面上的组名中，为组输入名称（例如 *MyDemoRepoGroup*），然后选择下一步。请考虑在组名称中包含存储库名称。

> **Note**
>
> 该名称必须在 Amazon Web Services 账户间保持唯一。

4. 选中您在前一部分中创建的客户管理型策略旁边的复选框（例如 AWSCodeCommitPowerUser-MyDemoRepo）。
5. 在 Review 页面上，选择 Create Group。IAM 将使用已附加的指定策略创建此组。此组会显示在与您的 Amazon Web Services 账户关联的组列表中。
6. 从列表中选择您的组。
7. 在组摘要页面上，选择用户 选项卡，然后选择向组添加多个用户。在显示与您的 Amazon Web Services 账户关联的所有用户的列表中，选中要允许其访问 CodeCommit 存储库的用户旁边的复选框，然后选择添加用户。

> **Tip**
>
> 您可以使用搜索框快速地按名称查找用户。

8. 添加用户后，关闭 IAM 控制台。

在创建 IAM 用户以使用您配置的策略组和策略访问 CodeCommit 后，向该用户发送连接到存储库所需的信息。

1. 打开 CodeCommit 控制台：https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在区域选择器中，选择创建存储库的 AWS 区域。存储库特定于 AWS 区域。有关更多信息，请参阅区域和 Git 连接端点。
3. 在 Repositories (存储库) 页面上，选择要共享的存储库。
4. 在克隆 URL 中，选择您希望用户使用的协议。这会复制连接协议的克隆 URL。
5. 向您的用户发送克隆 URL 以及任何其他说明，如安装 AWS CLI、对配置文件进行配置或安装 Git 的说明。请确保包含连接协议（例如 HTTPS）的配置信息。

以增量方式迁移存储库

在迁移到 AWS CodeCommit 时，请考虑以增量或数据块方式推送存储库，以减少因间歇性网络问题或网络性能下降而导致整个推送操作失败的几率。通过使用类似此处包含的脚本进行增量推送，您可以重新启动迁移，并只推送先前推送失败的提交。

本主题中的过程向您展示如何创建和运行以增量方式迁移存储库的脚本，该脚本将只重新推送那些推送失败的增量提交，直到迁移完成。

编写这些说明时，假定您已完成设置 和创建存储库中的步骤。

主题

步骤 0：确定是否需要增量迁移

要确定存储库的整体大小以及是否需要增量迁移，可以考虑以下几个因素。首先要考虑的当然是存储库中项目的整体大小。存储库的累积历史记录等因素也会对大小产生影响。就算存储库的各个资产并不大，但如果它包含多年的历史记录和分支，其整体大小也可能非常大。您可以通过多种策略来更简单、更高效地迁移这些存储库。例如，可以在克隆具有长期开发历史的存储库时使用浅克隆策略，也可以关闭大型二进制文件的增量压缩。您可以通过查阅 Git 文档来研究选项，也可以选择使用本主题中包含的示例脚本 (`incremental-repo-migration.py`) 来设置和配置以增量推送方式迁移存储库。

如果您满足以下一个或多个条件，则可能需要配置增量推送：

- 您要迁移的存储库具有五年以上的历史。
- 您的 Internet 连接存在间歇性中断、丢包、响应缓慢或其他服务中断问题。
- 存储库的整体大小大于 2 GB，并且您打算迁移整个存储库。
- 存储库包含压缩率不高的大型项目或二进制文件，例如具有超过五个跟踪版本的大型映像文件。
- 您以前尝试过迁移到 CodeCommit，但收到“内部服务错误”消息。

就算上述条件都不成立，您也可以选择增量推送。

步骤 1：安装必备组件并将 CodeCommit 存储库添加为远程存储库

您可以创建自己的自定义脚本，它有自己的必备组件。如果您使用本主题中包括的示例，则必须：

- 安装其必备组件。
- 将存储库克隆到本地计算机。
- 将 CodeCommit 存储库添加为要迁移的存储库的远程存储库。

设置运行 incremental-repo-migration.py

1. 在本地计算机上，安装 Python 2.6 或更高版本。有关更多信息以及最新版本的信息，请参阅 Python 网站。
2. 在同一台计算机上，安装 GitPython，这是用于与 Git 存储库交互的 Python 库。有关更多信息，请参阅 GitPython 文档。
3. 使用 git clone --mirror 命令克隆要迁移到本地计算机的存储库。在终端（Linux、macOS 或 Unix）或命令提示符 (Windows) 中，使用 git clone --mirror 命令为该存储库创建一个本地存储

库，包括要在其中创建本地存储库的目录。例如，要将 URL 为 *https://example.com/my-repo/* 的名为 *MyMigrationRepo* 的 Git 存储库克隆到名为 *my-repo* 的目录：

```
git clone --mirror https://example.com/my-repo/MyMigrationRepo.git my-repo
```

您应会看到类似以下内容的输出，这表示存储库已被克隆到名为 my-repo 的本地空存储库中：

```
Cloning into bare repository 'my-repo'...
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 20 (delta 5), reused 15 (delta 3)
Unpacking objects: 100% (20/20), done.
Checking connectivity... done.
```

4. 将目录切换到您刚刚克隆的存储库的本地存储库 (例如，*my-repo*)。在该目录中，使用 git remote add ***DefaultRemoteName RemoteRepositoryURL*** 命令将 CodeCommit 存储库添加为本地存储库的远程存储库。

> **Note**
>
> 在推送大型存储库时，请考虑使用 SSH 而不是 HTTPS。在推送较大的更改、大量更改或大型存储库时，长时间运行的 HTTPS 连接通常会因为网络问题或防火墙设置而提前终止。有关为 CodeCommit 设置 SSH 连接的更多信息，请参阅适用于 Linux、macOS 或 Unix 上的 SSH 连接或适用于 Windows 上的 SSH 连接。

例如，使用以下命令来为名为 MyDestinationRepo 的 CodeCommit 存储库（它是名为 `codecommit` 的远程存储库的远程存储库）添加 SSH 端点：

```
git remote add codecommit ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDestinationRepo
```

> **Tip**
>
> 由于这是克隆存储库，已使用默认的远程存储库名称 (`origin`)。您必须使用其他的远程存储库名称。虽然示例使用了 `codecommit`，但您可以使用任意名称。使用 git remote show 命令查看为本地存储库设置的远程存储库的列表。

5. 使用 git remote -v 命令显示本地存储库的提取和推送设置，确认它们设置正确。例如：

```
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (fetch)
codecommit  ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDestinationRepo
 (push)
```

> **Tip**
>
> 如果您仍看到其他远程存储库的提取和推送条目（例如，origin 的条目），请使用 git remote set-url --delete 命令删除它们。

步骤 2：创建用于增量迁移的脚本

编写这些步骤时，假定您使用的是 `incremental-repo-migration.py` 示例脚本。

1. 打开一个文本编辑器，将示例脚本的内容粘贴到一个空文档中。
2. 将文档保存到某个文档目录中 (而不是本地存储库的工作目录)，并将其命名为 `incremental-repo-migration.py`。确保选择的目录是在本地环境或路径变量中配置的目录，以便您能够在命令行或终端中运行该 Python 脚本。

步骤 3：运行脚本并增量迁移到 CodeCommit

现在您已创建 `incremental-repo-migration.py` 脚本，可以使用它将本地存储库增量迁移到 CodeCommit 存储库。默认情况下，该脚本以 1000 个提交为批次推送提交，并尝试将运行该脚本时所在目录的 Git 设置用作本地存储库和远程存储库的设置。如果需要，您可以使用 `incremental-repo-migration.py` 中包含的选项配置其他设置。

1. 在终端或命令提示符中，切换到要迁移的本地存储库的目录。
2. 从该目录运行以下命令：

```
python incremental-repo-migration.py
```

3. 脚本运行并在终端或命令提示符中显示进度。一些大型存储库在显示进度时会有延迟。如果单次推送失败三次，脚本将停止运行。然后，您可以重新运行脚本，它会从失败的批次继续。您可以重新运行脚本，直到所有推送成功并且迁移完成。

Tip

您可以在任意目录中运行 `incremental-repo-migration.py`，前提是您用 `-l` 和 `-r` 选项指定了要使用的本地和远程存储库设置。例如，要在任意目录中使用该脚本将位于 /tmp/*my-repo* 的本地存储库迁移到别名为 *codecommit* 的远程存储库：

```
python incremental-repo-migration.py -l "/tmp/my-repo" -r "codecommit"
```

您可能还需要使用 `-b` 选项来更改增量推送时使用的默认批处理大小。例如，如果您定期推送具有经常发生更改的超大二进制文件的存储库，并且在网络带宽受限的位置执行操作，则您可能需要使用 `-b` 选项将批处理大小更改为 500 而不使用 1000。例如：

```
python incremental-repo-migration.py -b 500
```

这将以 500 个提交为批次增量推送本地存储库。如果您在迁移存储库时决定再次更改批次大小（例如，在推送失败后，您决定减小批次大小），请记得使用 `-c` 选项删除批次标签，然后再使用 `-b` 重置批次大小：

```
python incremental-repo-migration.py -c
python incremental-repo-migration.py -b 250
```

Important

推送失败后重新运行脚本时，请勿使用 `-c` 选项。`-c` 选项会删除用于对提交进行批处理的标签。仅当您想要更改批处理大小并重新开始时，或者您决定不再使用该脚本时，才能使用 `-c` 选项。

附录：示例脚本 **incremental-repo-migration.py**

为方便您参考，我们开发了一个用于增量推送存储库的示例 Python 脚本 `incremental-repo-migration.py`。该脚本是一个开源代码示例，按原样提供。

```
# Copyright 2015 Amazon.com, Inc. or its affiliates. All Rights Reserved. Licensed
 under the Amazon Software License (the "License").
# You may not use this file except in compliance with the License. A copy of the
 License is located at
```

```
#    http://aws.amazon.com/asl/
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, express or implied. See the License for
# the specific language governing permissions and limitations under the License.

#!/usr/bin/env python

import os
import sys
from optparse import OptionParser
from git import Repo, TagReference, RemoteProgress, GitCommandError


class PushProgressPrinter(RemoteProgress):
    def update(self, op_code, cur_count, max_count=None, message=""):
        op_id = op_code & self.OP_MASK
        stage_id = op_code & self.STAGE_MASK
        if op_id == self.WRITING and stage_id == self.BEGIN:
            print("\tObjects: %d" % max_count)


class RepositoryMigration:
    MAX_COMMITS_TOLERANCE_PERCENT = 0.05
    PUSH_RETRY_LIMIT = 3
    MIGRATION_TAG_PREFIX = "codecommit_migration_"

    def migrate_repository_in_parts(
        self, repo_dir, remote_name, commit_batch_size, clean
    ):
        self.next_tag_number = 0
        self.migration_tags = []
        self.walked_commits = set()
        self.local_repo = Repo(repo_dir)
        self.remote_name = remote_name
        self.max_commits_per_push = commit_batch_size
        self.max_commits_tolerance = (
            self.max_commits_per_push * self.MAX_COMMITS_TOLERANCE_PERCENT
        )

        try:
            self.remote_repo = self.local_repo.remote(remote_name)
            self.get_remote_migration_tags()
        except (ValueError, GitCommandError):
            print(
```

```
                "Could not contact the remote repository. The most common reasons for
this error are that the name of the remote repository is incorrect, or that you do not
have permissions to interact with that remote repository."
            )
            sys.exit(1)

        if clean:
            self.clean_up(clean_up_remote=True)
            return

        self.clean_up()

        print("Analyzing repository")
        head_commit = self.local_repo.head.commit
        sys.setrecursionlimit(max(sys.getrecursionlimit(), head_commit.count()))

        # tag commits on default branch
        leftover_commits = self.migrate_commit(head_commit)
        self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # tag commits on each branch
        for branch in self.local_repo.heads:
            leftover_commits = self.migrate_commit(branch.commit)
            self.tag_commits([commit for (commit, commit_count) in leftover_commits])

        # push the tags
        self.push_migration_tags()

        # push all branch references
        for branch in self.local_repo.heads:
            print("Pushing branch %s" % branch.name)
            self.do_push_with_retries(ref=branch.name)

        # push all tags
        print("Pushing tags")
        self.do_push_with_retries(push_tags=True)

        self.get_remote_migration_tags()
        self.clean_up(clean_up_remote=True)

        print("Migration to CodeCommit was successful")

    def migrate_commit(self, commit):
        if commit in self.walked_commits:
```

```
            return []

        pending_ancestor_pushes = []
        commit_count = 1

        if len(commit.parents) > 1:
            # This is a merge commit
            # Ensure that all parents are pushed first
            for parent_commit in commit.parents:
                pending_ancestor_pushes.extend(self.migrate_commit(parent_commit))
        elif len(commit.parents) == 1:
            # Split linear history into individual pushes
            next_ancestor, commits_to_next_ancestor = self.find_next_ancestor_for_push(
                commit.parents[0]
            )
            commit_count += commits_to_next_ancestor
            pending_ancestor_pushes.extend(self.migrate_commit(next_ancestor))

        self.walked_commits.add(commit)

        return self.stage_push(commit, commit_count, pending_ancestor_pushes)

    def find_next_ancestor_for_push(self, commit):
        commit_count = 0

        # Traverse linear history until we reach our commit limit, a merge commit, or
an initial commit
        while (
            len(commit.parents) == 1
            and commit_count < self.max_commits_per_push
            and commit not in self.walked_commits
        ):
            commit_count += 1
            self.walked_commits.add(commit)
            commit = commit.parents[0]

        return commit, commit_count

    def stage_push(self, commit, commit_count, pending_ancestor_pushes):
        # Determine whether we can roll up pending ancestor pushes into this push
        combined_commit_count = commit_count + sum(
            ancestor_commit_count
            for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
        )
```

```
        if combined_commit_count < self.max_commits_per_push:
            # don't push anything, roll up all pending ancestor pushes into this
pending push
            return [(commit, combined_commit_count)]

        if combined_commit_count <= (
            self.max_commits_per_push + self.max_commits_tolerance
        ):
            # roll up everything into this commit and push
            self.tag_commits([commit])
            return []

        if commit_count >= self.max_commits_per_push:
            # need to push each pending ancestor and this commit
            self.tag_commits(
                [
                    ancestor
                    for (ancestor, ancestor_commit_count) in pending_ancestor_pushes
                ]
            )
            self.tag_commits([commit])
            return []

        # push each pending ancestor, but roll up this commit
        self.tag_commits(
            [ancestor for (ancestor, ancestor_commit_count) in pending_ancestor_pushes]
        )
        return [(commit, commit_count)]

    def tag_commits(self, commits):
        for commit in commits:
            self.next_tag_number += 1
            tag_name = self.MIGRATION_TAG_PREFIX + str(self.next_tag_number)

            if tag_name not in self.remote_migration_tags:
                tag = self.local_repo.create_tag(tag_name, ref=commit)
                self.migration_tags.append(tag)
            elif self.remote_migration_tags[tag_name] != str(commit):
                print(
                    "Migration tags on the remote do not match the local tags. Most
likely your batch size has changed since the last time you ran this script. Please run
this script with the --clean option, and try again."
                )
```

```
                sys.exit(1)

    def push_migration_tags(self):
        print("Will attempt to push %d tags" % len(self.migration_tags))
        self.migration_tags.sort(
            key=lambda tag: int(tag.name.replace(self.MIGRATION_TAG_PREFIX, ""))
        )
        for tag in self.migration_tags:
            print(
                "Pushing tag %s (out of %d tags), commit %s"
                % (tag.name, self.next_tag_number, str(tag.commit))
            )
            self.do_push_with_retries(ref=tag.name)

    def do_push_with_retries(self, ref=None, push_tags=False):
        for i in range(0, self.PUSH_RETRY_LIMIT):
            if i == 0:
                progress_printer = PushProgressPrinter()
            else:
                progress_printer = None

            try:
                if push_tags:
                    infos = self.remote_repo.push(tags=True, progress=progress_printer)
                elif ref is not None:
                    infos = self.remote_repo.push(
                        refspec=ref, progress=progress_printer
                    )
                else:
                    infos = self.remote_repo.push(progress=progress_printer)

                success = True
                if len(infos) == 0:
                    success = False
                else:
                    for info in infos:
                        if (
                            info.flags & info.UP_TO_DATE
                            or info.flags & info.NEW_TAG
                            or info.flags & info.NEW_HEAD
                        ):
                            continue
                        success = False
                        print(info.summary)
```

```
                if success:
                    return
            except GitCommandError as err:
                print(err)

        if push_tags:
            print("Pushing all tags failed after %d attempts" %
(self.PUSH_RETRY_LIMIT))
        elif ref is not None:
            print("Pushing %s failed after %d attempts" % (ref, self.PUSH_RETRY_LIMIT))
            print(
                "For more information about the cause of this error, run the following
command from the local repo: 'git push %s %s'"
                % (self.remote_name, ref)
            )
        else:
            print(
                "Pushing all branches failed after %d attempts"
                % (self.PUSH_RETRY_LIMIT)
            )
        sys.exit(1)

    def get_remote_migration_tags(self):
        remote_tags_output = self.local_repo.git.ls_remote(
            self.remote_name, tags=True
        ).split("\n")
        self.remote_migration_tags = dict(
            (tag.split()[1].replace("refs/tags/", ""), tag.split()[0])
            for tag in remote_tags_output
            if self.MIGRATION_TAG_PREFIX in tag
        )

    def clean_up(self, clean_up_remote=False):
        tags = [
            tag
            for tag in self.local_repo.tags
            if tag.name.startswith(self.MIGRATION_TAG_PREFIX)
        ]

        # delete the local tags
        TagReference.delete(self.local_repo, *tags)

        # delete the remote tags
```

```
        if clean_up_remote:
            tags_to_delete = [":" + tag_name for tag_name in
 self.remote_migration_tags]
            self.remote_repo.push(refspec=tags_to_delete)


parser = OptionParser()
parser.add_option(
    "-l",
    "--local",
    action="store",
    dest="localrepo",
    default=os.getcwd(),
    help="The path to the local repo. If this option is not specified, the script will
 attempt to use current directory by default. If it is not a local git repo, the script
 will fail.",
)
parser.add_option(
    "-r",
    "--remote",
    action="store",
    dest="remoterepo",
    default="codecommit",
    help="The name of the remote repository to be used as the push or migration
 destination. The remote must already be set in the local repo ('git remote add ...').
 If this option is not specified, the script will use 'codecommit' by default.",
)
parser.add_option(
    "-b",
    "--batch",
    action="store",
    dest="batchsize",
    default="1000",
    help="Specifies the commit batch size for pushes. If not explicitly set, the
 default is 1,000 commits.",
)
parser.add_option(
    "-c",
    "--clean",
    action="store_true",
    dest="clean",
    default=False,
    help="Remove the temporary tags created by migration from both the local repo
 and the remote repository. This option will not do any migration work, just cleanup.
```

```
 Cleanup is done automatically at the end of a successful migration, but not after a
 failure so that when you re-run the script, the tags from the prior run can be used to
 identify commit batches that were not pushed successfully.",
)

(options, args) = parser.parse_args()

migration = RepositoryMigration()
migration.migrate_repository_in_parts(
    options.localrepo, options.remoterepo, int(options.batchsize), options.clean
)
```

AWS CodeCommit 中的安全性

AWS 的云安全性具有优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 AWS Compliance Programs 的一部分。要了解适用于 AWS CodeCommit 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。您还需要对其它因素负责，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。

本文档将帮助您了解如何在使用 CodeCommit 时应用责任共担模式。以下主题介绍了如何配置 CodeCommit 以实现您的安全性和合规性目标。您还将了解如何使用其他 AWS 服务以帮助您监控 CodeCommit 资源并确保其安全。

主题

AWS CodeCommit 中的数据保护

作为一项托管式服务，受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实操设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络进行访问。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（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

CodeCommit 存储库会自动执行静态加密。客户无需执行任何操作。CodeCommit 还会加密传输中存储库数据。您可以将 HTTPS 协议和/或 SSH 协议用于 CodeCommit 存储库。有关更多信息，请参阅对 AWS CodeCommit 进行设置。您还可以配置对 CodeCommit 存储库的跨账户存取。

主题

AWS Key Management Service 和 AWS CodeCommit 存储库加密

CodeCommit 存储库中的数据在传输过程中和静态时都经过加密。当数据被推送到 CodeCommit 存储库时（例如，通过调用git push），CodeCommit 会像存储在存储库中一样对收到的数据进行加密。从 CodeCommit 存储库中提取数据时（例如，通过调用git pull），会 CodeCommit 解密数据，然后将其发送给调用方。上述过程假定与推送或拉取请求关联的 IAM 用户已经过 AWS 的身份验证。发送或接收的数据使用 HTTPS 或 SSH 加密网络协议进行传输。

您可以使用 AWS 托管式密钥或客户托管密钥来加密和解密存储库中的数据。有关客户托管密钥与 AWS 托管式密钥之间的不同之处的更多信息，请参阅客户托管密钥和 AWS 托管式密钥。如果您未指定客户托管密钥，则 CodeCommit AWS 托管式密钥将使用加密和解密存储库中的数据。系统会自动在 AWS 账户中为您创建此 AWS 托管式密钥。首次在您的 Amazon Web Services 账户的新 CodeCommit 存储库AWS 区域中创建存储库时，如果您未指定客户托管密钥，则会在 AWS 托管式密钥 () 的同AWS 区域一个存储库中 CodeCommit 创建一个AWS Key Management Service（`aws/codecommit`密钥AWS KMS）。此`aws/codecommit`密钥仅供使用 CodeCommit。它存储在您的 Amazon Web Services 账户中。根据您指定的内容，CodeCommit 要么使用客户管理的密钥，要么使用AWS 托管式密钥来加密和解密存储库中的数据。

> **Important**
>
> CodeCommit 对存储库中用于加密和解密数据的密AWS KMS钥执行以下AWS KMS操作。如果您使用 AWS 托管式密钥，用户无需获得下述操作的显式权限，但您也不得向其附加拒绝对 `aws/codecommit` 密钥执行这些操作的任何策略。如果您使用的客户托管密钥将 AWS 账户 ID 设置为该密钥的策略委托人，则必须将这些权限明确设置为`allow`。具体而言，在创建第一个存储库时，如果您更新存储库的密钥，且使用的是 AWS 托管式密钥，则不得将以下任何权限设置为 `deny`；如果您使用的是具有策略主体的客户托管密钥，则必须将所有权限设置为 `allow`：

- "kms:Encrypt"
- "kms:Decrypt"
- "kms:ReEncrypt"（视情况而定，这可能需要 kms:ReEncryptFrom、kms:ReEncryptTo 或者 kms:ReEncrypt* 不设为 deny）
- "kms:GenerateDataKey"
- "kms:GenerateDataKeyWithoutPlaintext"
- "kms:DescribeKey"

如果您想使用自己的客户托管密钥，则该密钥必须在存储库AWS 区域所在的地方可用。 CodeCommit 支持同时使用单区域客户托管密钥和多区域客户托管密钥。虽然支持所有密钥材料源类型，但建议使用默认 KMS 选项。使用外部密钥存储选项的客户可能会遇到因存储提供程序而发生的延迟。此外。CodeCommit 对客户托管密钥有以下要求：

- CodeCommit 仅支持使用对称密钥。
- 密钥使用类型必须设置为加密和解密。

有关创建客户托管密钥的更多信息，请参阅概念和创建密钥。

要查看有关AWS 托管式密钥生成者的信息 CodeCommit，请执行以下操作：

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Key Management Service (AWS KMS) 控制台：https://console.aws.amazon.com/kms。
2. 要更改 AWS 区域，请使用页面右上角的区域选择器。
3. 在服务导航窗格中，选择 AWS 托管式密钥。请确保您已登录要在其中查看密钥的 AWS 区域。
4. 在加密密钥列表中，选择别名为 aws/codecommit 的 AWS 托管式密钥。有关 AWS 拥有的密钥的基本信息将会显示。

您无法更改或删除此 AWS 托管式密钥。

如何使用加密算法加密存储库数据

CodeCommit 使用两种不同的方法来加密数据。6 MB 以下的单个 Git 对象使用 AES-GCM-256 进行加密，该方法提供数据完整性验证。对于单个 blob 来说，介于 6 MB 到最大 2 GB 之间的对象使用 AES-CBC-256 进行加密。 CodeCommit 始终验证加密上下文。

加密上下文

与 AWS KMS 集成的每项服务都会为加密和解密操作指定加密上下文。加密上下文是 AWS KMS 检查数据完整性时使用的额外的身份验证信息。如果为加密操作指定了加密上下文，则也必须在解密操作中指定它。否则，解密将失败。 CodeCommit 使用 CodeCommit 存储库 ID 作为加密上下文。您可以使用get-repository命令或 CodeCommit 控制台来查找存储库 ID。在AWS CloudTrail日志中搜索 CodeCommit 存储库 ID，以了解对哪个密钥进行了哪些加密操作AWS KMS来加密或解密存储库中的 CodeCommit 数据。

有关 AWS KMS 的更多信息，请参阅《AWS Key Management Service 开发人员指南》。

使用轮换凭证连接到 AWS CodeCommit 存储库

您可以授予用户访问您的 AWS CodeCommit 存储库的权限，而无需为其配置 IAM 用户或使用访问密钥和秘密密钥。要向联合身份分配权限，您可以创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。您还可以配置基于角色的访问权限，以便 IAM 用户访问不同 Amazon Web Services 账户中的 CodeCommit 存储库（一种称为跨账户存取的技术）。有关配置对存储库的跨账户存取的演练，请参阅使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限。

当用户想要或必须通过以下方式进行身份验证时，您可以为这些用户配置访问权限：

- 安全断言标记语言 (SAML)
- 多重身份验证 (MFA)
- 联合身份验证
- Login with Amazon
- Amazon Cognito
- Facebook
- Google
- OpenID Connect (OIDC) 兼容身份提供商

> **Note**
>
> 以下信息仅适用于使用 git-remote-codecommit 或 AWS CLI 凭证助手连接到 CodeCommit 存储库的情况。由于对 CodeCommit 进行临时访问或联合访问的建议方法是设置 git-remote-codecommit，因此本主题提供了使用该实用程序的示例。有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。
> 您无法使用 SSH 或 Git 凭证及 HTTPS 通过轮换或临时访问凭证连接到 CodeCommit 存储库。

如果满足以下所有要求，则不需要完成这些步骤：

- 您已登录 Amazon EC2 实例。
- 您结合使用 Git 和 HTTPS 与 AWS CLI 凭证助手从 Amazon EC2 实例连接到 CodeCommit 存储库。
- Amazon EC2 实例附加了 IAM 实例配置文件，其中包含适用于在 Linux、macOS 或 Unix 上使用 AWS CLI 凭证助手进行 HTTPS 连接或适用于在 Windows 上使用 AWS CLI 凭证助手进行 HTTPS 连接中所述的访问权限。
- 您已按照适用于在 Linux、macOS 或 Unix 上使用 AWS CLI 凭证助手进行 HTTPS 连接或适用于在 Windows 上使用 AWS CLI 凭证助手进行 HTTPS 连接中所述，在 Amazon EC2 实例上安装和配置了 Git 凭证助手。

已将满足上述要求的 Amazon EC2 实例设为代表您将临时访问凭证传达给 CodeCommit。

> **Note**
>
> 您可以在 Amazon EC2 实例上配置和使用 git-remote-codecommit。

要允许用户临时访问您的 CodeCommit 存储库，请完成以下步骤。

步骤 1：完成先决条件

完成设置步骤，为用户提供使用轮换凭证访问您的 CodeCommit 存储库的权限：

- 有关跨账户存取的信息，请参阅演练：使用 IAM 角色委派跨 Amazon Web Services 账户的访问权限和使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限。

- 有关 SAML 和联合身份验证的信息，请参阅使用贵组织的身份验证系统授予访问 AWS 资源的权限和关于 AWS STS 基于 SAML 2.0 的联合身份验证。
- 有关 MFA 的信息，请参阅使用 AWS 多重身份验证 (MFA) 设备和创建临时安全凭证，为 IAM 用户启用访问权限。
- 有关 Login with Amazon、Amazon Cognito、Facebook、Google 或任何与 OIDC 兼容的身份提供者的信息，请参阅关于 AWS STS Web 身份联合验证。

使用AWS CodeCommit 的身份验证和访问控制中的信息指定要授予用户的 CodeCommit 权限。

步骤 2：获取角色名称或访问凭证

如果您希望您的用户通过代入角色来访问存储库，请为您的用户提供该角色的 Amazon 资源名称 (ARN)。否则，根据您设置访问权限的方式，您的用户可以通过以下方式之一获得轮换凭证：

- 对于跨账户访问，请调用 AWS CLI assume-role 命令或调用 AWS STS AssumeRole API。
- 对于 SAML，请调用 AWS CLI assume-role-with-saml 命令或 AWS STS AssumeRoleWithSAML API。
- 对于联合身份验证，请调用 AWS CLI assume-role 或 get-federation-token 命令或者 AWS STS AssumeRole 或 GetFederationToken API。
- 对于 MFA，请调用 AWS CLI get-session-token 命令或 AWS STS GetSessionToken API。
- 对于 Login with Amazon、Amazon Cognito、Facebook、Google 或任何与 OIDC 兼容的身份提供者，请调用 AWS CLI assume-role-with-web-identity 命令或 AWS STS AssumeRoleWithWebIdentity API。

步骤 3：安装 git-remote-codecommit 并配置 AWS CLI

您必须通过安装 git-remote-codecommit 并在 AWS CLI 中配置一个配置文件来配置本地计算机以使用访问凭证。

1. 按照设置 中的说明设置 AWS CLI。使用 aws configure 命令配置一个或多个配置文件。考虑创建一个命名配置文件，以便在使用轮换凭证连接到 CodeCommit 存储库时使用。
2. 您可以使用下列方法之一将凭证与用户的 AWS CLI 命名配置文件关联。

    - 如果您通过代入角色来访问 CodeCommit，请使用代入该角色所需的信息配置一个命名配置文件。例如，如果您希望在 Amazon Web Services 账户 111111111111 中代入一个名为 *CodeCommitAccess* 的角色，则可以配置一个默认配置文件（在使用其他 AWS 资源时使用）

和一个命名配置文件（在代入该角色时使用）。以下命令创建一个名为 *CodeAccess* 的命名配置文件，该配置文件代入一个名为 *CodeCommitAccess* 的角色。用户名 *Maria_Garcia* 与会话关联，默认配置文件设置为其 AWS 凭证的来源：

```
aws configure set role_arn arn:aws:iam::111111111111:role/CodeCommitAccess --
profile CodeAccess
aws configure set source_profile default --profile CodeAccess
aws configure set role_session_name "Maria_Garcia" --profile CodeAccess
```

如果要验证更改，请手动查看或编辑 ~/.aws/config 文件（适用于 Linux）或 %UserProfile%.aws\config 文件（适用于 Windows），并查看命名配置文件下的信息。例如，您的文件可能如下所示：

```
[default]
region = us-east-1
output = json

[profile CodeAccess]
source_profile = default
role_session_name = Maria_Garcia
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
```

配置您的命名配置文件后，您可以使用此命名配置文件通过 git-remote-codecommit 实用程序克隆 CodeCommit 存储库。例如，要克隆名为 *MyDemoRepo* 的存储库，请运行以下命令：

```
git clone codecommit://CodeAccess@MyDemoRepo
```

- 如果您使用的是 Web 联合身份验证和 OpenID Connect (OIDC)，请配置代表您进行 AWS Security Token Service (AWS STS) AssumeRoleWithWebIdentity API 调用的命名配置文件以刷新临时凭证。使用 aws configure set 命令或手动编辑 ~/.aws/credentials 文件（适用于 Linux）或 %UserProfile%.aws\credentials 文件（适用于 Windows），以添加具有所需设置值的 AWS CLI 命名配置文件。例如，要创建一个代入 *CodeCommitAccess* 角色且使用 Web 身份令牌文件 ~/*my-credentials/my-token-file* 的配置文件，请运行以下命令：

```
[CodeCommitWebIdentity]
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
web_identity_token_file=~/my-credentials/my-token-file
role_session_name = Maria_Garcia
```

有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS Command Line Interface 和使用 AWS CLI 中的 IAM 角色。

步骤 4：访问 CodeCommit 存储库

如果您的用户已按照连接存储库中的说明连接到 CodeCommit 存储库，那么该用户就可以使用 git-remote-codecommit 提供的扩展功能和 Git 调用 git clone、git push 和 git pull 来对他或她有权访问的 CodeCommit 存储库执行克隆、推送和提取操作。例如，要克隆存储库：

```
git clone codecommit://CodeAccess@MyDemoRepo
```

Git 提交、推送和拉取命令使用常规 Git 语法。

如果用户使用 AWS CLI 并指定与轮换访问凭证关联的 AWS CLI 命名配置文件，则所返回结果的范围限定为该配置文件。

适用于 AWS CodeCommit 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。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 CodeCommit 资源的人员。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- AWS CodeCommit 的身份验证和访问控制
- AWS CodeCommit 如何与 IAM 协同工作
- CodeCommit 基于资源的策略
- 基于 CodeCommit 标签的授权
- CodeCommit IAM 角色
- AWS CodeCommit 基于身份的策略示例
- 对 AWS CodeCommit 身份和访问进行故障排除

受众

使用 AWS Identity and Access Management (IAM) 的方式因您可以在 CodeCommit 中执行的操作而异。

服务用户：如果您使用 CodeCommit 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 CodeCommit 功能来完成工作时，您可能需要更多权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 CodeCommit 中的功能，请参阅对 AWS CodeCommit 身份和访问进行故障排除。

服务管理员：如果您在公司负责管理 CodeCommit 资源，则您可能具有 CodeCommit 的完全访问权限。您有责任确定您的服务用户应访问哪些 CodeCommit 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 CodeCommit 搭配使用的更多信息，请参阅 AWS CodeCommit 如何与 IAM 协同工作。

IAM 管理员：如果您是 IAM 管理员，您可能希望详细了解如何编写策略以管理对 CodeCommit 的访问。要查看您可在 IAM 中使用的 CodeCommit 基于身份的策略示例，请参阅 AWS CodeCommit 基于身份的策略示例。

使用身份进行身份验证

身份验证是使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过担任 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您都可能需要提供其它安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS 中使用多重身份验证（MFA）。

AWS 账户 根用户

创建 AWS 账户 时，最初使用的是一个对账户中所有 AWS 服务 和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

IAM 用户和群组

IAM 用户是 AWS 账户 内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用群组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用群组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人担任。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时担任 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户或角色可代入 IAM 角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户存取权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到

资源（而不是使用角色作为座席）。要了解用于跨账户存取的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

- 跨服务访问 – 某些 AWS 服务 使用其他 AWS 服务 中的特征。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的政策详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色 - 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 - 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM policy。然后，管理员可以向角色添加 IAM policy，并且用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、群组或角色中。托管策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》 中的访问控制列表 (ACL) 概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型授予的最大权限。

- 权限边界 - 权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可以为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边

界的交集。在 `Principal` 字段中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。

- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅《IAM 用户指南》中的策略评估逻辑。

AWS CodeCommit 的身份验证和访问控制

访问 AWS CodeCommit 需要凭证。这些凭证必须有权访问 AWS 资源（如 CodeCommit 存储库）和您的 IAM 用户（您通过该用户来管理 Git 凭证或用于进行 Git 连接的 SSH 公有密钥）。下列各节详述了如何使用 AWS Identity and Access Management (IAM) 和 CodeCommit 来帮助确保对您资源的访问安全：

- 身份验证
- 访问控制

身份验证

由于 CodeCommit 存储库基于 Git 并且支持包括 Git 凭证在内的 Git 基本功能，建议通过 IAM 用户使用 CodeCommit。您可以使用其他身份类型访问 CodeCommit，但其他身份类型存在以下限制。

身份类型：

- IAM 用户：IAM 用户是您的 Amazon Web Services 中具有特定自定义权限的身份。例如，IAM 用户可以有权创建和管理用于访问 CodeCommit 存储库的 Git 凭证。这是使用 CodeCommit 时

建议的用户类型。您可以使用 IAM 用户名和密码登录以保护 AWS 网页（如AWS Management Console、AWS 开发论坛或 AWS Support 中心）。

您可以生成 Git 凭证或将 SSH 公有密钥与 IAM 用户关联，也可以安装和配置 git-remote-codecommit。这些是设置 Git 以使用 CodeCommit 存储库的最简单方法。您可以使用 Git 凭证在 IAM 中生成静态用户名和密码。然后在使用 Git 的 HTTPS 连接和支持 Git 用户名及密码身份验证的任何第三方工具中使用这些凭证。借助 SSH 连接，您可以在本地计算机上创建公有和私有密钥文件，以供 Git 和 CodeCommit 进行 SSH 身份验证。将公有密钥与 IAM 用户关联，然后将私有密钥存储在本地计算机上。git-remote-codecommit 扩展 Git 本身，不要求为用户设置 Git 凭证。

此外，还可以为每个用户生成访问密钥。在通过 AWS 软件开发工具包之一AWS或使用 (AWS Command Line Interface)AWS CLI 以编程方式访问 服务时，可以使用这些访问密钥。软件开发工具包和 CLI 工具使用访问密钥对您的请求进行加密签名。如果不使用 AWS 工具，则必须自行对请求签名。CodeCommit 支持签名版本 4，这是用于对入站 API 请求进行身份验证的协议。有关验证请求的更多信息，请参阅《AWS 一般参考》 中的签名版本 4 签名流程。

- Amazon Web Services 账户根用户：注册 AWS 时，您需要提供与您的 Amazon Web Services 账户关联的电子邮件地址和密码。这些是您的根凭证，它们提供对您所有 AWS 资源的完全访问权限。某些 CodeCommit 功能对根账户用户不可用。此外，将 Git 与根账户结合使用的唯一方法是安装和配置 git-remote-codecommit（推荐）或配置 AWS 凭证辅助程序（随 AWS CLI 附带）。您不能对根账户用户使用 Git 凭证或 SSH 公有-私有密钥对。出于这些原因，不建议您在与 CodeCommit 交互时使用根账户用户。

  > **Important**
  >
  > 出于安全考虑，我们建议您仅使用根凭证创建管理员用户，该用户是对您的AWS账户具有完全访问权的 IAM 用户。随后，您可以使用此管理员用户来创建具有有限权限的其他 IAM 用户和角色。有关更多信息，请参阅 IAM 用户指南中的 IAM 最佳实践和创建管理员用户和组。

- IAM Identity Center 和 IAM Identity Center 中的用户：AWS IAM Identity Center 扩展了 AWS Identity and Access Management 的功能，让您能够在一个集中的位置管理用户及其对 AWS 账户和云应用程序的访问。虽然 IAM Identity Center 是建议大多数用户使用 AWS 的最佳实践，但目前并未提供针对 Git 凭证或 SSH 密钥对的机制。这些用户可以安装和配置 git-remote-codecommit 来本地克隆 CodeCommit 存储库，但并非所有集成开发环境 (IDE) 都支持使用 git-remote-codecommit 进行克隆、推送或拉取。

  作为最佳实操，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、网络身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户时，他们担任角色，而角色提供临时凭证。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？

- IAM 角色：与 IAM 用户类似，IAM 角色是可在账户中创建以授予特定权限的 IAM 身份。

  IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时担任 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

  具有临时凭证的 IAM 角色在以下情况下很有用：

  - 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
  - 临时 IAM 用户权限 – IAM 用户或角色可代入 IAM 角色，以暂时获得针对特定任务的不同权限。
  - 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户存取权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为座席）。要了解用于跨账户存取的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
  - 跨服务访问 – 某些 AWS 服务 使用其他 AWS 服务 中的特征。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
    - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的政策详情，请参阅转发访问会话。

- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
- 服务相关角色 - 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 Amazon EC2 上运行的应用程序 - 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

> **Note**
>
> 您不能对联合身份用户使用 Git 凭证或 SSH 公有-私有密钥对。此外，用户偏好对于联合身份用户不可用。有关如何使用联合访问设置连接的信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 CodeCommit 资源。例如，您必须拥有权限才能查看存储库、推送代码、创建和管理 Git 凭证等。

下面几节介绍了如何管理 CodeCommit 的权限。我们建议您先阅读概述。

- 管理 CodeCommit 资源的访问权限的概述
- 将基于身份的策略 (IAM policy) 用于 CodeCommit
- CodeCommit 权限参考

管理 CodeCommit 资源的访问权限的概述

每个 AWS 资源归一个 Amazon Web Services 账户拥有。创建或访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略。一些服务（例如 AWS Lambda）还支持向资源附加权限策略。

> **Note**
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅《IAM 用户指南》中的 IAM 最佳实操。

在授予权限时，您确定谁获得权限、获得对哪些资源的权限以及允许对这些资源执行的具体操作。

主题

CodeCommit 资源和操作

在 CodeCommit 中，主要资源是存储库。每种资源均有相关联的唯一 Amazon 资源名称 (ARN)。在策略中，您可以使用 Amazon Resource Name（ARN）标识策略应用到的资源。有关 ARN 的详细信息，请参阅 AWS 中的 Amazon 资源名称 (ARN) 和 Amazon Web Services 一般参考 服务命名空间。CodeCommit 目前不支持称作子资源的其他资源类型。

下表介绍了如何指定 CodeCommit 资源。

资源类型	ARN 格式
存储库	arn:aws:codecommit:*region*:*account-id* :*repository-name*
所有 CodeCommit 存储库	arn:aws:codecommit:*

资源类型	ARN 格式
指定 AWS 区域中指定账户拥有的所有 CodeCommit 存储库	arn:aws:codecommit:*region*:*account-id* :*

> **Note**
>
> 大多数 AWS 服务将 ARN 中的冒号 (:) 或正斜杠 (/) 视为相同字符。不过，CodeCommit 在资源模式和规则中要求精确匹配。在创建事件模式时，请务必使用正确的 ARN 字符，以使其与资源中的 ARN 语法匹配。

例如，您可以使用特定存储库 (*MyDemoRepo*) 的 ARN 在语句中指定它，如下所示：

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo"
```

要指定属于某一特定账户的所有存储库，请使用通配符 (*)，如下所示：

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:*"
```

要指定所有资源，或者，如果特定 API 操作不支持 ARN，请在 `Resource` 元素中使用通配符 (*)，如下所示：

```
"Resource": "*"
```

您还可以使用通配符 (*) 指定与某一存储库名称部分匹配的所有资源。例如，以下 ARN 指定了以 `MyDemo` 开头的 CodeCommit 存储库，该存储库注册到 `us-east-2` AWS 区域的 Amazon Web Services 帐户 `111111111111`：

```
arn:aws:codecommit:us-east-2:111111111111:MyDemo*
```

有关 CodeCommit 资源的可用操作的列表，请参阅CodeCommit 权限参考。

了解资源所有权

Amazon Web Services 账户拥有在该账户中创建的资源，无论这些资源由谁创建。具体来说，资源所有者是验证资源创建请求的主体实体（即根账户、IAM 用户或 IAM 角色）的 Amazon Web Services 账户。以下示例说明了它的工作原理：

- 如果您在 Amazon Web Services 账户中创建一个 IAM 用户，并授予该用户创建 CodeCommit 资源的权限，则该用户可以创建 CodeCommit 资源。但是，该用户所属的 Amazon Web Services 账户拥有 CodeCommit 资源。
- 如果您使用 Amazon Web Services 账户的根账户凭证创建规则，则您的 Amazon Web Services 账户是 CodeCommit 资源的拥有者。
- 如果您在 Amazon Web Services 账户中创建一个 IAM 角色，并赋予其创建 CodeCommit 资源的权限，那么任何可以代入该角色的人都可以创建 CodeCommit 资源。该角色所属的 Amazon Web Services 账户拥有 CodeCommit 资源。

管理对资源的访问

要管理对 AWS 资源的访问，可以使用权限策略。权限策略规定谁可以访问哪些内容。下一节介绍权限策略创建选项。

> **Note**
>
> 本节讨论如何在 CodeCommit 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关 IAM 的更多信息，请参阅《IAM 用户指南》中的什么是 IAM？。有关 IAM 策略语法和说明的信息，请参阅《IAM 用户指南》中的 IAM 策略参考。

附加到 IAM 身份的权限策略称作基于身份的策略（IAM policy）。附加到资源的权限策略称作基于资源的策略。目前，CodeCommit 仅支持基于身份的策略（IAM policy）。

主题

- 基于身份的策略（IAM 策略）
- 基于资源的策略

基于身份的策略（IAM 策略）

要管理对 AWS 资源的访问，可以将权限策略附加到 IAM 身份。在 CodeCommit 中，可以使用基于身份的策略控制对存储库的访问。例如，您可以执行以下操作：

- 将权限策略附加到账户中的用户或组：要授予用户查看 CodeCommit 控制台中的 CodeCommit 资源的权限，可以将基于身份的权限策略附加到用户或用户所属的组。
- 将权限策略附加到角色（授予跨账户权限）：委托（例如，要授予跨账户存取权限时）需要在拥有资源的账户（信任账户）与包含需要访问资源的用户的账户（可信账户）之间建立信任。权限策略授予角色用户对资源执行预期任务所需的权限。信任策略指定允许哪些可信账户授予其用户代入角色的权限。有关更多信息，请参阅 IAM 术语和概念。

  要授予跨账户权限，可将基于身份的权限策略附加到 IAM 角色。例如，账户 A 中的管理员可以创建一个角色，以向其他 Amazon Web Services 账户（如账户 B）或某项 AWS 服务授予跨账户权限，如下所述：

  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以把信任策略附加至用来标识账户 B 的角色，账户 B 由此可以作为主体代入该角色。
  3. 之后，账户 B 管理员可以委派权限，指派账户 B 中的任何用户担任该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 AWS 服务权限来代入该角色，则信任策略中的委托人也可以是 AWS 服务委托人。有关更多信息，请参阅 IAM 术语和概念中的“委托”。

  有关使用 IAM 委托权限的更多信息，请参阅 IAM 用户指南中的访问权限管理。

以下示例策略允许用户在名为 *MyDemoRepo* 的存储库中创建一个分支：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:CreateBranch"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

要限制账户中的用户有权访问的调用和资源，可以创建特定的 IAM policy，然后将这些策略附加到 IAM 用户。有关创建 IAM 角色和探索适用于 CodeCommit 的 IAM policy 语句的更多信息，请参阅客户管理型身份策略示例。

基于资源的策略

一些服务（如 Amazon S3）支持基于资源的权限策略。例如，您可以将基于资源的策略附加到 S3 桶以管理对该桶的访问权限。CodeCommit 不支持基于资源的策略，但您可以使用标签来识别资源，然后在 IAM policy 中使用。有关基于标签的策略示例，请参阅基于身份的策略（IAM 策略）。

CodeCommit 中的资源范围界定

在 CodeCommit 中，可以将基于身份的策略和权限的范围限定到资源，如 CodeCommit 资源和操作中所述。但是，不能将 `ListRepositories` 权限的范围限定为某一资源，而必须将其范围设置为所有资源 (使用通配符 *)。否则，该操作将会失败。

所有其他 CodeCommit 权限的范围都可以限定到资源。

指定策略元素：资源、操作、效果和委托方

您可以创建策略来允许或拒绝用户访问资源，或者允许或拒绝用户对这些资源执行特定操作。CodeCommit 定义一组公有 API 操作来确定用户如何使用该服务：是通过 CodeCommit 控制台、软件开发工具包、AWS CLI 还是直接调用这些 API。为了向这些 API 操作授予权限，CodeCommit 定义了一组您可以在策略中指定的操作。

某些 API 操作可能需要执行多个操作的权限。有关资源和 API 操作的更多信息，请参阅 CodeCommit 资源和操作 和 CodeCommit 权限参考。

下面是基本的策略元素：

- 资源：要标识策略所适用的资源，可以使用 Amazon 资源名称 (ARN)。有关更多信息，请参阅CodeCommit 资源和操作。
- 操作：要标识要允许或拒绝的资源操作，可以使用操作关键字。例如，根据指定的 `Effect`，`codecommit:GetBranch` 权限允许或拒绝用户执行 `GetBranch` 操作，该操作可获取有关 CodeCommit 存储库中某个分支的详细信息。
- 效果：您可以指定当用户请求特定操作时的效果（允许或拒绝）。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。也可显式拒绝对资源的访问，这样即使有其他策略授予了访问权限，也可确保用户无法访问该资源。
- 主体：在基于身份的策略（IAM policy，这是 CodeCommit 支持的唯一一种策略类型）中，策略所关联的用户是隐式主体。

有关 IAM policy 语法的更多信息，请参阅《IAM 用户指南》中的 IAM policy 参考。

有关显示所有 CodeCommit API 操作及其适用的资源的表，请参阅 CodeCommit 权限参考。

在策略中指定条件

授予权限时，可以使用 IAM 的访问策略语言指定策略生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅《IAM 用户指南》中的条件和策略语法。

要表示条件，您可以使用预定义的条件键。没有特定于 CodeCommit 的条件键。但有 AWS 范围内的条件密钥，您可以根据需要使用。有关 AWS 范围内的键的完整列表，请参阅《IAM 用户指南》中的条件的可用键。

将基于身份的策略 (IAM policy) 用于 CodeCommit

以下基于身份的策略示例演示了账户管理员如何将权限策略附加到 IAM 身份（用户、组和角色），以授予对 CodeCommit 资源执行操作的权限。

> **Important**
>
> 我们建议您首先阅读以下介绍性主题，这些主题说明了可用于管理 CodeCommit 资源访问的基本概念和选项。有关更多信息，请参阅管理 CodeCommit 资源的访问权限的概述。

主题

- 使用控制台所需的权限
- 在控制台中查看资源
- 适用于 CodeCommit 的 AWS 托管式策略
- 客户管理型策略示例

下面是基于身份的权限策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:BatchGetRepositories"
      ],
      "Resource" : [
        "arn:aws:codecommit:us-east-2:111111111111:MyDestinationRepo",
        "arn:aws:codecommit:us-east-2:111111111111:MyDemo*"
```

```
      ]
    }
  ]
}
```

该策略有一条语句，允许用户获取 **us-east-2** 区域中名为 MyDestinationRepo 的 CodeCommit 存储库和所有以 MyDemo 开头的 CodeCommit 存储库的信息。

使用控制台所需的权限

要查看每个 CodeCommit API 操作所需的权限以及有关 CodeCommit 操作的更多信息，请参阅 CodeCommit 权限参考。

要允许用户使用 CodeCommit 控制台，管理员必须授予他们执行 CodeCommit 操作的权限。例如，您可以将 AWSCodeCommitPowerUser 托管策略或其等效策略附加到用户或组。

除了通过基于身份的策略授予用户的权限外，CodeCommit 还需要执行 AWS Key Management Service (AWS KMS) 操作的权限。IAM 用户不需要这些操作的显式 Allow 权限，但该用户不能附加有任何将以下权限设为 Deny 的策略：

```
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:DescribeKey"
```

有关加密和 CodeCommit 的更多信息，请参阅 AWS KMS 和加密。

在控制台中查看资源

CodeCommit 控制台需要 ListRepositories 权限，以显示您的 Amazon Web Services 账户在所登录的 AWS 区域中的存储库列表。该控制台还包括一个转到资源功能，可对资源快速执行不区分大小写的搜索。此搜索通过您的 Amazon Web Services 账户，在您登录的 AWS 区域中执行。将显示以下服务中的以下资源：

- AWS CodeBuild：构建项目
- AWS CodeCommit：存储库
- AWS CodeDeploy：应用程序
- AWS CodePipeline：管道

要在所有服务中跨资源执行此搜索，您必须具有如下权限：

- CodeBuild：`ListProjects`
- CodeCommit：`ListRepositories`
- CodeDeploy：`ListApplications`
- CodePipeline：`ListPipelines`

如果您没有针对某个服务的权限，搜索将不会针对该服务的资源返回结果。即使您有权限查看资源，但如果特定资源明确 `Deny` 查看，搜索也不会返回这些资源。

适用于 CodeCommit 的 AWS 托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 AWS 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户 中使用。有关 AWS 托管策略的更多信息，请参阅《IAM 用户指南》中的AWS 托管策略。

AWS 服务负责维护和更新 AWS 托管式策略。您无法更改 AWS 托管式策略中的权限。服务偶尔会向 AWS 托管策略添加额外权限以支持新特征。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新特征或新操作可用时，服务最有可能会更新 AWS 托管策略。服务不会从 AWS 托管策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，AWS 还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管式策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动新特征时，AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的 AWS 托管策略。

AWS 通过提供由 AWS 创建和管理的独立 IAM 策略来满足许多常用案例的要求。这些 AWS 托管策略将授予针对常用案例的必要权限。CodeCommit 的托管式策略还提供在其他服务（如 IAM、Amazon SNS 和 Amazon CloudWatch Events）中执行操作的权限，这是授予了相关策略的用户职责所必需的。例如，AWSCodeCommitFullAccess 策略是管理级用户策略，允许具有此策略的用户为存储库创建和管理 CloudWatch Events 规则（名称前缀为 `codecommit` 的规则），并为存储库相关事件通知创建和管理 Amazon SNS 主题（名称前缀为 `codecommit` 的主题），以及在 CodeCommit 中管理存储库。

以下 AWS 托管式策略（可附加到账户中的用户）专门用于 CodeCommit。

主题

- AWS 托管式策略：AWSCodeCommitFullAccess

- AWS 托管式策略：AWSCodeCommitPowerUser
- AWS 托管式策略：AWSCodeCommitReadOnly
- CodeCommit 托管式策略和通知
- AWS CodeCommit 托管式策略和 Amazon CodeGuru Reviewer
- CodeCommit 对 AWS 托管式策略的更新

AWS 托管式策略：AWSCodeCommitFullAccess

您可以将 AWSCodeCommitFullAccess 策略附加到 IAM 身份。此策略授予对 CodeCommit 的完全访问权限。仅将此策略应用于您希望向其授予对 CodeCommit 存储库及您的 Amazon Web Services 账户中相关资源的完全控制权限（包括删除存储库的能力）的管理级用户。

AWSCodeCommitFullAccess 策略包含以下策略声明：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchEventsCodeCommitRulesAccess",
      "Effect": "Allow",
      "Action": [
        "events:DeleteRule",
        "events:DescribeRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "events:ListTargetsByRule"
      ],
      "Resource": "arn:aws:events:*:*:rule/codecommit*"
    },
    {
      "Sid": "SNSTopicAndSubscriptionAccess",
```

```
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:DeleteTopic",
        "sns:Subscribe",
        "sns:Unsubscribe",
        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:*:*:codecommit*"
    },
    {
      "Sid": "SNSTopicAndSubscriptionReadAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:ListSubscriptionsByTopic",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "LambdaReadOnlyListAccess",
      "Effect": "Allow",
      "Action": [
        "lambda:ListFunctions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMReadOnlyListAccess",
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMReadOnlyConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:ListAccessKeys",
        "iam:ListSSHPublicKeys",
        "iam:ListServiceSpecificCredentials"
      ],
```

```
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "IAMUserSSHKeys",
      "Effect": "Allow",
      "Action": [
        "iam:DeleteSSHPublicKey",
        "iam:GetSSHPublicKey",
        "iam:ListSSHPublicKeys",
        "iam:UpdateSSHPublicKey",
        "iam:UploadSSHPublicKey"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "IAMSelfManageServiceSpecificCredentials",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceSpecificCredential",
        "iam:UpdateServiceSpecificCredential",
        "iam:DeleteServiceSpecificCredential",
        "iam:ResetServiceSpecificCredential"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "CodeStarNotificationsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:DeleteNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
        }
      }
    },
    {
```

```
          "Sid": "CodeStarNotificationsListAccess",
          "Effect": "Allow",
          "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
          "Effect": "Allow",
          "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
          ],
          "Resource": "arn:aws:sns:*:*:codestar-notifications*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerFullAccess",
          "Effect": "Allow",
          "Action": [
            "codeguru-reviewer:AssociateRepository",
            "codeguru-reviewer:DescribeRepositoryAssociation",
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DisassociateRepository",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
```

```
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
          "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
      ]
    }
```

AWS 托管式策略：AWSCodeCommitPowerUser

您可以将 AWSCodeCommitPowerUser 策略附加到您的 IAM 身份。此策略允许用户访问 CodeCommit 的所有功能和存储库相关资源，但不允许用户删除 CodeCommit 存储库或在其他 AWS 服务（如 Amazon CloudWatch Events）中创建或删除存储库相关资源。建议对大多数用户应用此策略。

AWSCodeCommitPowerUser 策略包含以下策略声明：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:AssociateApprovalRuleTemplateWithRepository",
        "codecommit:BatchAssociateApprovalRuleTemplateWithRepositories",
        "codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories",
        "codecommit:BatchGet*",
        "codecommit:BatchDescribe*",
        "codecommit:Create*",
        "codecommit:DeleteBranch",
        "codecommit:DeleteFile",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:Get*",
        "codecommit:List*",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Put*",
        "codecommit:Post*",
        "codecommit:TagResource",
        "codecommit:Test*",
        "codecommit:UntagResource",
        "codecommit:Update*",
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchEventsCodeCommitRulesAccess",
      "Effect": "Allow",
      "Action": [
        "events:DeleteRule",
        "events:DescribeRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:PutRule",
        "events:PutTargets",
```

```
            "events:RemoveTargets",
            "events:ListTargetsByRule"
          ],
          "Resource": "arn:aws:events:*:*:rule/codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionAccess",
          "Effect": "Allow",
          "Action": [
            "sns:Subscribe",
            "sns:Unsubscribe"
          ],
          "Resource": "arn:aws:sns:*:*:codecommit*"
        },
        {
          "Sid": "SNSTopicAndSubscriptionReadAccess",
          "Effect": "Allow",
          "Action": [
            "sns:ListTopics",
            "sns:ListSubscriptionsByTopic",
            "sns:GetTopicAttributes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "LambdaReadOnlyListAccess",
          "Effect": "Allow",
          "Action": [
            "lambda:ListFunctions"
          ],
          "Resource": "*"
        },
        {
          "Sid": "IAMReadOnlyListAccess",
          "Effect": "Allow",
          "Action": [
            "iam:ListUsers"
          ],
          "Resource": "*"
        },
        {
          "Sid": "IAMReadOnlyConsoleAccess",
          "Effect": "Allow",
          "Action": [
```

```
            "iam:ListAccessKeys",
            "iam:ListSSHPublicKeys",
            "iam:ListServiceSpecificCredentials"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
          "Sid": "IAMUserSSHKeys",
          "Effect": "Allow",
          "Action": [
            "iam:DeleteSSHPublicKey",
            "iam:GetSSHPublicKey",
            "iam:ListSSHPublicKeys",
            "iam:UpdateSSHPublicKey",
            "iam:UploadSSHPublicKey"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
          "Sid": "IAMSelfManageServiceSpecificCredentials",
          "Effect": "Allow",
          "Action": [
            "iam:CreateServiceSpecificCredential",
            "iam:UpdateServiceSpecificCredential",
            "iam:DeleteServiceSpecificCredential",
            "iam:ResetServiceSpecificCredential"
          ],
          "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
          "Sid": "CodeStarNotificationsReadWriteAccess",
          "Effect": "Allow",
          "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
          ],
          "Resource": "*",
          "Condition": {
            "StringLike": {
              "codestar-notifications:NotificationsForResource": "arn:aws:codecommit:*"
            }
```

```
          }
        },
        {
          "Sid": "CodeStarNotificationsListAccess",
          "Effect": "Allow",
          "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerFullAccess",
          "Effect": "Allow",
          "Action": [
            "codeguru-reviewer:AssociateRepository",
            "codeguru-reviewer:DescribeRepositoryAssociation",
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DisassociateRepository",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
          ],
          "Resource": "*"
        },
        {
          "Sid": "AmazonCodeGuruReviewerSLRCreation",
          "Action": "iam:CreateServiceLinkedRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CloudWatchEventsManagedRules",
          "Effect": "Allow",
          "Action": [
            "events:PutRule",
            "events:PutTargets",
```

```
            "events:DeleteRule",
            "events:RemoveTargets"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
            }
          }
        },
        {
          "Sid": "CodeStarNotificationsChatbotAccess",
          "Effect": "Allow",
          "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
          "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:ListConnections",
                "codestar-connections:GetConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*"
        }
      ]
    }
```

AWS 托管式策略：AWSCodeCommitReadOnly

您可以将 AWSCodeCommitReadOnly 策略附加到 IAM 身份。该策略授予对 CodeCommit 和其他 AWS 服务中存储库相关资源的只读访问权限，以及创建和管理他们自己的 CodeCommit 相关资源（如供 IAM 用户访问存储库时使用的 Git 凭证和 SSH 密钥）的权限。针对希望向其授予读取存储库内容的能力但不能对内容进行任何更改的用户，应用此策略。

AWSCodeCommitReadOnly 策略包含以下策略声明：

```
    {
       "Version":"2012-10-17",
       "Statement":[
```

```
        {
           "Effect":"Allow",
           "Action":[
              "codecommit:BatchGet*",
              "codecommit:BatchDescribe*",
              "codecommit:Describe*",
              "codecommit:EvaluatePullRequestApprovalRules",
              "codecommit:Get*",
              "codecommit:List*",
              "codecommit:GitPull"
           ],
           "Resource":"*"
        },
        {
           "Sid":"CloudWatchEventsCodeCommitRulesReadOnlyAccess",
           "Effect":"Allow",
           "Action":[
              "events:DescribeRule",
              "events:ListTargetsByRule"
           ],
           "Resource":"arn:aws:events:*:*:rule/codecommit*"
        },
        {
           "Sid":"SNSSubscriptionAccess",
           "Effect":"Allow",
           "Action":[
              "sns:ListTopics",
              "sns:ListSubscriptionsByTopic",
              "sns:GetTopicAttributes"
           ],
           "Resource":"*"
        },
        {
           "Sid":"LambdaReadOnlyListAccess",
           "Effect":"Allow",
           "Action":[
              "lambda:ListFunctions"
           ],
           "Resource":"*"
        },
        {
           "Sid":"IAMReadOnlyListAccess",
           "Effect":"Allow",
           "Action":[
```

```
                "iam:ListUsers"
            ],
            "Resource":"*"
         },
         {
            "Sid":"IAMReadOnlyConsoleAccess",
            "Effect":"Allow",
            "Action":[
               "iam:ListAccessKeys",
               "iam:ListSSHPublicKeys",
               "iam:ListServiceSpecificCredentials",
               "iam:GetSSHPublicKey"
            ],
            "Resource":"arn:aws:iam::*:user/${aws:username}"
         },
         {
            "Sid":"CodeStarNotificationsReadOnlyAccess",
            "Effect":"Allow",
            "Action":[
               "codestar-notifications:DescribeNotificationRule"
            ],
            "Resource":"*",
            "Condition":{
               "StringLike":{
                  "codestar-
notifications:NotificationsForResource":"arn:aws:codecommit:*"
               }
            }
         },
         {
            "Sid":"CodeStarNotificationsListAccess",
            "Effect":"Allow",
            "Action":[
               "codestar-notifications:ListNotificationRules",
               "codestar-notifications:ListEventTypes",
               "codestar-notifications:ListTargets"
            ],
            "Resource":"*"
         },
         {
            "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
               "codeguru-reviewer:DescribeRepositoryAssociation",
```

```
                "codeguru-reviewer:ListRepositoryAssociations",
                "codeguru-reviewer:DescribeCodeReview",
                "codeguru-reviewer:ListCodeReviews"
            ],
            "Resource": "*"
        },
        {
          "Sid": "CodeStarConnectionsReadOnlyAccess",
          "Effect": "Allow",
          "Action": [
              "codestar-connections:ListConnections",
              "codestar-connections:GetConnection"
          ],
          "Resource": "arn:aws:codestar-connections:*:*:connection/*"
      }
    ]
}
```

CodeCommit 托管式策略和通知

AWS CodeCommit 支持通知功能，可以向用户通知存储库的重要更改。CodeCommit 的托管式策略包含通知功能的策略语句。有关更多信息，请参阅什么是通知？。

完全访问托管策略中的通知的相关权限

AWSCodeCommitFullAccess 托管策略包含以下语句，以允许对通知进行完全访问。应用此托管策略的用户还可以创建和管理用于通知的 Amazon SNS 主题、为用户订阅和取消订阅主题、列出要选择作为通知规则目标的主题，以及列出为 Slack 配置的 AWS Chatbot 客户端。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
```

```
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource,"
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

只读托管策略中的通知的相关权限

AWSCodeCommitReadOnlyAccess 托管策略包含以下语句，以允许对通知进行只读访问。应用此托管策略的用户可以查看资源的通知，但无法创建、管理或订阅这些通知。

```
  {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
```

```
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codecommit:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
        "Resource": "*"
    }
```

其他托管策略中的通知的相关权限

AWSCodeCommitPowerUser 托管策略包含以下语句，以允许用户创建、编辑和订阅通知。用户无法删除通知规则或管理资源的标签。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codecommit*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
```

```
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

有关 IAM 和通知的更多信息，请参阅 AWS CodeStar 通知的身份和访问管理。

AWS CodeCommit 托管式策略和 Amazon CodeGuru Reviewer

CodeCommit 支持 Amazon CodeGuru Reviewer，后者是一项自动代码审查服务，它使用程序分析和机器学习来检测 Java 或 Python 代码中的常见问题并提供修复建议。CodeCommit 托管式策略包含针对 CodeGuru Reviewer 功能的策略语句。有关更多信息，请参阅什么是 Amazon CodeGuru Reviewer。

AWSCodeCommitFullAccess 中与 CodeGuru Reviewer 相关的权限

`AWSCodeCommitFullAccess` 托管式策略包含以下语句，以允许将 CodeGuru Reviewer 与 CodeCommit 存储库关联和取消关联。应用此托管式策略的用户还可以查看 CodeCommit 存储库与 CodeGuru Reviewer 之间的关联状态，并查看拉取请求的审核作业状态。

```
    {
```

```
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
        "codeguru-reviewer:DisassociateRepository",
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
      }
    }
```

AWSCodeCommitPowerUser 中与 CodeGuru Reviewer 相关的权限

AWSCodeCommitPowerUser 托管式策略包含以下语句，以允许用户将存储库与 CodeGuru 关联和取消关联，并查看拉取请求的审核作业状态。

```
    {
      "Sid": "AmazonCodeGuruReviewerFullAccess",
      "Effect": "Allow",
      "Action": [
        "codeguru-reviewer:AssociateRepository",
        "codeguru-reviewer:DescribeRepositoryAssociation",
        "codeguru-reviewer:ListRepositoryAssociations",
        "codeguru-reviewer:DisassociateRepository",
        "codeguru-reviewer:DescribeCodeReview",
        "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonCodeGuruReviewerSLRCreation",
      "Action": "iam:CreateServiceLinkedRole",
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:role/aws-service-role/codeguru-
reviewer.amazonaws.com/AWSServiceRoleForAmazonCodeGuruReviewer",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "codeguru-reviewer.amazonaws.com"
        }
      }
    },
    {
      "Sid": "CloudWatchEventsManagedRules",
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:ManagedBy": "codeguru-reviewer.amazonaws.com"
        }
```

```
      }
    }
```

AWSCodeCommitReadOnly 中与 CodeGuru Reviewer 相关的权限

AWSCodeCommitReadOnlyAccess 托管式策略包括以下语句，允许对 CodeGuru Reviewer 关联状态进行只读访问并查看拉取请求的审核作业状态。应用了此托管策略的用户无法关联或取消关联存储库。

```
    {
     "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
     "Effect": "Allow",
     "Action": [
           "codeguru-reviewer:DescribeRepositoryAssociation",
           "codeguru-reviewer:ListRepositoryAssociations",
           "codeguru-reviewer:DescribeCodeReview",
           "codeguru-reviewer:ListCodeReviews"
     ],
     "Resource": "*"
    }
```

Amazon CodeGuru Reviewer 服务相关角色

当您将存储库与 CodeGuru Reviewer 关联时，系统会创建一个服务相关角色，以便 CodeGuru Reviewer 能检测拉取请求中 Java 或 Python 代码的问题并提供修复建议。该服务相关角色命名为 AWSServiceRoleForAmazonCodeGuruReviewer。有关更多信息，请参阅使用面向 Amazon CodeGuru Reviewer 的服务相关角色。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

CodeCommit 对 AWS 托管式策略的更新

查看有关自此服务开始跟踪这些更改起，CodeCommit 的 AWS 托管式策略更新的详细信息。要获得有关此页面更改的自动提示，请订阅 AWS CodeCommit 用户指南文档历史记录 的 RSS 源。

更改	说明	日期
AWS 托管式策略：AWS CodeCommitFullAccess 和 AWS 托管式策略：AWS	CodeCommit 为这些策略增加了一项权限，以支持使用 AWS Chatbot 的额外通知类型。	2023 年 5 月 16 日

更改	说明	日期
CodeCommitPowerUser：对现有策略的更新	AWSCodeCommitPowerUser 和 AWSCodeCommitFullAccess 策略已更改，增加了一项权限 `chatbot:ListMicrosoftTeamsChannelConfigurations` 。	
AWS 托管式策略：AWS CodeCommitReadOnly – 现有策略更新	CodeCommit 从策略中删除了一个重复的权限。 AWSCodeCommitReadOnly 已更改，删除了一个重复权限 `"iam:ListAccessKeys"` 。	2021 年 8 月 18 日
CodeCommit 开始跟踪更改	CodeCommit 为其 AWS 托管式策略开启了跟踪更改。	2021 年 8 月 18 日

客户管理型策略示例

此外，您还可以创建您自己的自定义 IAM policy，以向 CodeCommit 操作和资源授予相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。您还可以创建自己的自定义 IAM policy，以便集成 CodeCommit 和其他 AWS 服务。

主题

- 客户管理型身份策略示例
- 客户管理型集成策略示例

客户管理型身份策略示例

以下 IAM policy 示例为各种 CodeCommit 操作授予权限。使用它们来限制 IAM 用户和角色对 CodeCommit 的访问。这些策略控制使用 CodeCommit 控制台、API、AWS 软件开发工具包或 AWS CLI 执行操作的能力。

> **Note**
>
> 所有示例都使用 美国西部（俄勒冈）区域 (us-west-2) 和虚构的账户 ID。

示例

- 示例 1：允许用户在单个 AWS 区域中执行 CodeCommit 操作
- 示例 2：允许用户对单个存储库使用 Git
- 示例 3：允许从指定 IP 地址范围连接的用户访问存储库
- 示例 4：拒绝或允许对分支执行操作
- 示例 5：使用标签拒绝或允许对存储库执行操作

示例 1：允许用户在单个 AWS 区域中执行 CodeCommit 操作

以下权限策略使用通配符 (`"codecommit:*"`) 以允许用户在 us-east-2 区域中（而不是在其他 AWS 区域中）执行所有 CodeCommit 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codecommit:*",
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "codecommit:ListRepositories",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-2"
                }
            }
```

```
        }
    ]
}
```

示例 2：允许用户对单个存储库使用 Git

在 CodeCommit 中，GitPull IAM policy 权限适用于从 CodeCommit 检索数据的任何 Git 客户端命令，包括 git fetch、git clone 等。同样，GitPush IAM policy 权限适用于将数据发送到 CodeCommit 的任何 Git 客户端命令。例如，如果 GitPush IAM policy 权限设置为 Allow，则用户可以使用 Git 协议推送分支删除。针对该 IAM 用户的 DeleteBranch 操作应用的任何权限都不会影响该推送。DeleteBranch 权限适用于通过控制台、AWS CLI、软件开发工具包和 API 执行的操作，但不适用于通过 Git 协议执行的操作。

下面的示例允许指定用户对名为 MyDemoRepo 的 CodeCommit 存储库执行拉取和推送操作：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

示例 3：允许从指定 IP 地址范围连接的用户访问存储库

您可以创建一个策略，仅允许 IP 地址在特定 IP 地址范围内的用户连接到 CodeCommit 存储库。可通过两种等效方法来实现此目的。一种是创建 Deny 策略，当用户 IP 地址不在特定块内时禁止 CodeCommit 操作；另一种是创建 Allow 策略，当用户 IP 地址在特定块内时允许 CodeCommit 操作。

您可以创建 Deny 策略，拒绝在特定 IP 范围之外的所有用户的访问。例如，您可以为所有需要访问存储库的用户附加 AWSCodeCommitPowerUser 托管式策略和客户管理型策略。以下示例策略拒绝向 IP 地址不在指定的 IP 地址块 203.0.113.0/16 内的用户授予所有 CodeCommit 权限：

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": [
            "codecommit:*"
         ],
         "Resource": "*",
         "Condition": {
            "NotIpAddress": {
               "aws:SourceIp": [
                  "203.0.113.0/16"
               ]
            }
         }
      }
   ]
}
```

下面的示例策略允许指定用户访问名为 MyDemoRepo 的 CodeCommit 存储库，只有当他们的 IP 地址位于指定的地址块 203.0.113.0/16 内时，才具有与 AWSCodeCommitPowerUser 托管式策略同等的权限：

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codecommit:BatchGetRepositories",
            "codecommit:CreateBranch",
            "codecommit:CreateRepository",
            "codecommit:Get*",
            "codecommit:GitPull",
            "codecommit:GitPush",
            "codecommit:List*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:TagResource",
            "codecommit:Test*",
            "codecommit:UntagResource",
            "codecommit:Update*"
```

```
         ],
         "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
         "Condition": {
            "IpAddress": {
               "aws:SourceIp": [
                  "203.0.113.0/16"
               ]
            }
         }
      }
   ]
}
```

示例 4：拒绝或允许对分支执行操作

您可以创建一条策略，拒绝用户在一个或多个分支上执行指定操作的权限。或者，您可以创建一条策略，允许在一个或多个分支上执行某些操作，但在该存储库的其他分支上则不允许执行这些操作。这些策略可与相应的托管 (预定义) 策略结合使用。有关更多信息，请参阅限制推送和合并到中的分支 AWS CodeCommit。

例如，您可以创建一个 Deny 策略，拒绝用户更改名为 *MyDemoRepo* 的存储库中名为 main 的分支，包括删除该分支。您可以将此策略与 AWSCodeCommitPowerUser 托管式策略一起使用。应用了这两个策略的用户可以创建和删除分支、创建拉取请求，还可以执行 AWSCodeCommitPowerUser 允许的所有其他操作，但不能将更改推送到名为 main 的分支、不能在 CodeCommit 控制台中添加或编辑 main 分支中的文件，也不能将分支或拉取请求合并到 main 分支。由于 Deny 应用于 GitPush，您必须在该策略中包含 Null 语句，当用户从本地存储库进行推送时，分析初始 GitPush 调用是否有效。

> **Tip**
>
> 如果您希望创建一个策略，应用于您的 Amazon Web Services 账户的所有存储库中名为 main 的所有分支，对于 Resource，请指定星号 (*) 而不是存储库 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
```

```
                "codecommit:GitPush",
                "codecommit:DeleteBranch",
                "codecommit:PutFile",
                "codecommit:Merge*"
            ],
            "Resource": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                },
                "Null": {
                    "codecommit:References": "false"
                }
            }
        }
    ]
}
```

以下示例策略允许用户对 Amazon Web Services 账户的所有存储库中名为 main 的分支进行更改。它不允许更改任何其他分支。可以将此策略与 AWSCodeCommitReadOnly 托管式策略结合使用，以允许自动推送到 main 分支中的存储库。由于效果为 `Allow`，所以此示例策略无法与 AWSCodeCommitPowerUser 这样的托管式策略结合使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:GitPush",
                "codecommit:Merge*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                }
            }
        }
```

```
    ]
}
```

示例 5：使用标签拒绝或允许对存储库执行操作

您可以根据与存储库相关联的 AWS 标签创建允许或拒绝对存储库执行操作的策略，然后将这些策略应用到您为管理 IAM 用户而配置的 IAM 组。例如，您可以创建一个策略，拒绝对 AWS 标签键为 Status 且键值为 Secret 的任何存储库执行所有 CodeCommit 操作，然后将该策略应用到您为普通开发人员创建的 IAM 组（####）。然后，您需要确保在标记的存储库上工作的开发人员不是一般####组的成员，而是属于另一个没有应用限制性策略的 IAM 组 (SecretDevelopers)。

以下示例拒绝对用键 Status 和键值 Secret 标记的存储库执行所有 CodeCommit 操作：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:DeleteRepository",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
```

```
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:Git*",
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Status": "Secret"
        }
      }
    }
  ]
}
```

您可以通过指定特定存储库（而不是所有存储库）作为资源以进一步优化该策略。您还可以创建策略，允许对未使用特定标签标记的所有存储库执行 CodeCommit 操作。例如，以下策略允许对 CodeCommit 的操作使用等同于 AWSCodeCommitPowerUser 的权限，但只允许对未使用指定标签标记的存储库执行 CodeCommit 操作：

> **Note**
>
> 此策略示例仅包括对 CodeCommit 的操作。它不包括对 AWSCodeCommitPowerUser 托管式策略中包含的其他 AWS 服务的操作。有关更多信息，请参阅AWS 托管式策略：AWSCodeCommitPowerUser。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:Associate*",
        "codecommit:Batch*",
        "codecommit:CancelUploadArchive",
        "codecommit:CreateBranch",
        "codecommit:CreateCommit",
        "codecommit:CreatePullRequest*",
        "codecommit:CreateRepository",
        "codecommit:CreateUnreferencedMergeCommit",
        "codecommit:DeleteBranch",
        "codecommit:DeleteCommentContent",
        "codecommit:DeleteFile",
        "codecommit:DeletePullRequest*",
        "codecommit:Describe*",
        "codecommit:DisassociateApprovalRuleTemplateFromRepository",
        "codecommit:EvaluatePullRequestApprovalRules",
        "codecommit:GetBlob",
        "codecommit:GetBranch",
        "codecommit:GetComment*",
        "codecommit:GetCommit",
        "codecommit:GetDifferences*",
        "codecommit:GetFile",
        "codecommit:GetFolder",
        "codecommit:GetMerge*",
        "codecommit:GetObjectIdentifier",
        "codecommit:GetPullRequest*",
        "codecommit:GetReferences",
        "codecommit:GetRepository*",
        "codecommit:GetTree",
        "codecommit:GetUploadArchiveStatus",
```

```
        "codecommit:Git*",
        "codecommit:ListAssociatedApprovalRuleTemplatesForRepository",
        "codecommit:ListBranches",
        "codecommit:ListPullRequests",
        "codecommit:ListTagsForResource",
        "codecommit:Merge*",
        "codecommit:OverridePullRequestApprovalRules",
        "codecommit:Post*",
        "codecommit:Put*",
        "codecommit:TagResource",
        "codecommit:TestRepositoryTriggers",
        "codecommit:UntagResource",
        "codecommit:UpdateComment",
        "codecommit:UpdateDefaultBranch",
        "codecommit:UpdatePullRequest*",
        "codecommit:UpdateRepository*",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/Status": "Secret",
          "aws:ResourceTag/Team": "Saanvi"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CreateApprovalRuleTemplate",
        "codecommit:GetApprovalRuleTemplate",
        "codecommit:ListApprovalRuleTemplates",
        "codecommit:ListRepositories",
        "codecommit:ListRepositoriesForApprovalRuleTemplate",
        "codecommit:UpdateApprovalRuleTemplateContent",
        "codecommit:UpdateApprovalRuleTemplateDescription",
        "codecommit:UpdateApprovalRuleTemplateName"
      ],
      "Resource": "*"
    }
  ]
}
```

客户管理型集成策略示例

本节提供客户管理型用户策略示例，它们为 CodeCommit 和其他 AWS 服务之间的集成授予权限。有关允许对 CodeCommit 存储库进行跨账户存取的特定策略示例，请参阅使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限。

> **Note**
>
> 所有示例在需要 AWS 区域 时都使用美国西部（俄勒冈州）区域 (us-west-2)，并包含一个虚构的账户 ID。

示例

- 示例 1：创建允许对 Amazon SNS 主题进行跨账户存取的策略
- 示例 2：创建 Amazon Simple Notification Service (Amazon SNS) 主题策略，允许 Amazon CloudWatch Events 向主题发布 CodeCommit 事件
- 示例 3：为 AWS Lambda 与 CodeCommit 触发器的集成创建策略

示例 1：创建允许对 Amazon SNS 主题进行跨账户存取的策略

您可以配置 CodeCommit 存储库，以便代码推送或其他事件触发操作，例如从 Amazon Simple Notification Service (Amazon SNS) 发送通知。如果使用用于创建 CodeCommit 存储库的同一账户创建 Amazon SNS 主题，则无需配置其他 IAM policy 或权限。您可以创建主题，然后为存储库创建触发器。有关更多信息，请参阅为 Amazon SNS 主题创建触发器。

但是，如果要将触发器配置为使用另一个 Amazon Web Services 账户中的 Amazon SNS 主题，则必须先用允许 CodeCommit 发布到该主题的策略配置该主题。从另一账户打开 Amazon SNS 控制台，从列表中选择主题，然后在其他主题操作中选择编辑主题策略。在高级选项卡上，修改该主题的策略，以允许 CodeCommit 向该主题发布内容。例如，如果策略是默认策略，则应按如下方式修改策略，更改######中的项目，使其与存储库、Amazon SNS 主题和账户的值相匹配：

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "sns:Subscribe",
        "sns:ListSubscriptionsByTopic",
        "sns:DeleteTopic",
        "sns:GetTopicAttributes",
        "sns:Publish",
        "sns:RemovePermission",
        "sns:AddPermission",        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "111111111111"
        }
      }
     },
     {
      "Sid": "CodeCommit-Policy_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "codecommit.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:111111111111:NotMySNSTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

示例 2：创建 Amazon Simple Notification Service (Amazon SNS) 主题策略，允许 Amazon CloudWatch Events 向主题发布 CodeCommit 事件

您可以将 CloudWatch Events 配置为在事件（包括 CodeCommit 事件）发生时发布到 Amazon SNS 主题。为此，您必须确保 CloudWatch Events 拥有向 Amazon SNS 主题发布事件的权限，方法是为主题创建策略或修改主题的现有策略，如下所示：

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "123456789012"
        }
      }
    },
    {
      "Sid": "Allow_Publish_Events",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

有关 CodeCommit 和 CloudWatch Events 的更多信息，请参阅来自受支持服务的 CloudWatch Events 事件示例。有关 IAM 和策略语言的更多信息，请参阅 IAM JSON 策略语言的语法。

示例 3：为 AWS Lambda 与 CodeCommit 触发器的集成创建策略

您可以对 CodeCommit 存储库进行配置，以使代码推送或其他事件能够触发操作，例如调用 AWS Lambda 中的函数。有关更多信息，请参阅为 Lambda 函数创建触发器。此信息特定于触发器而不是 CloudWatch Events。

如果您希望触发器直接运行 Lambda 函数（而不是使用 Amazon SNS 主题来调用 Lambda 函数），并且不在 Lambda 控制台中配置触发器，则必须在函数的基于资源的策略中包含类似下面的语句：

```
{
```

```
  "Statement":{
     "StatementId":"Id-1",
     "Action":"lambda:InvokeFunction",
     "Principal":"codecommit.amazonaws.com",
     "SourceArn":"arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
     "SourceAccount":"111111111111"
  }
}
```

手动配置调用 Lambda 函数的 CodeCommit 触发器时，还必须使用 Lambda AddPermission 命令授予 CodeCommit 调用函数的权限。有关示例，请参阅为现有的 Lambda 函数创建触发器的 CodeCommit 允许运行 Lambda 函数部分。

有关 Lambda 函数资源策略的更多信息，请参阅《AWS Lambda 开发人员指南》中的 AddPermission 和拉取/推送事件模型。

CodeCommit 权限参考

下表列出了每个 CodeCommit API 操作、可授予权限的相应操作以及授予权限时使用的资源 ARN 格式。表中的 CodeCommit API 是根据 API 允许的操作范围分组的。在设置访问控制和编写可附加到 IAM 身份的权限策略（基于身份的策略）时，请参考此表。

在创建权限策略时，可以在策略的 `Action` 字段中指定操作。在策略的 `Resource` 字段中以 ARN 的形式指定资源值，可以使用或不使用通配符 (*)。

要在 CodeCommit 策略中表示条件，可以使用 AWS 范围内的条件键。有关 AWS 范围内的键的完整列表，请参阅《IAM 用户指南》中的可用键。有关 IAM policy 中 CodeCommit 的操作、资源和条件键的完整信息，请参阅 AWS CodeCommit 的操作、资源和条件键。

> **Note**
>
> 要指定操作，请在 API 操作名称之前使用 `codecommit:` 前缀 (例如，`codecommit:GetRepository` 或 `codecommit:CreateRepository`)。

使用通配符

要指定多个操作或资源，可以在 ARN 中使用通配符 (*)。例如，`codecommit:*` 指定所有 CodeCommit 操作，`codecommit:Get*` 指定以单词 `Get` 开头的所有 CodeCommit 操作。以下示例授予对以 `MyDemo` 名称开头的所有存储库的访问权限。

```
arn:aws:codecommit:us-west-2:111111111111:MyDemo*
```

只能对下表中列出的 *repository-name* 资源使用通配符，不能对 *region* 或 *account-id* 资源使用通配符。有关通配符的更多信息，请参阅《IAM 用户指南》中的 IAM 标识符。

主题

- Git 客户端命令所需的权限
- 分支操作权限
- 针对合并的操作所需的权限
- 针对拉取请求的操作所需的权限
- 针对审批规则模板的操作所需的权限
- 针对单个文件的操作所需的权限
- 针对评论的操作所需的权限
- 针对已提交代码的操作所需的权限
- 针对存储库的操作所需的权限
- 针对标签的操作所需的权限
- 针对触发器的操作所需的权限
- 针对 CodePipeline 集成的操作所需的权限

Git 客户端命令所需的权限

在 CodeCommit 中，`GitPull` IAM policy 权限适用于从 CodeCommit 检索数据的任何 Git 客户端命令，包括 git fetch、git clone 等。同样，`GitPush` IAM policy 权限适用于将数据发送到 CodeCommit 的任何 Git 客户端命令。例如，如果 `GitPush` IAM policy 权限设置为 `Allow`，则用户可以使用 Git 协议推送分支删除。针对该 IAM 用户的 `DeleteBranch` 操作应用的任何权限都不会影响该推送。`DeleteBranch` 权限适用于通过控制台、AWS CLI、软件开发工具包和 API 执行的操作，但不适用于通过 Git 协议执行的操作。

`GitPull` 和 `GitPush` 是 IAM policy 权限，它们不是 API 操作。

CodeCommit 对 Git 客户端命令的操作所需的权限

GitPull

操作： `codecommit:GitPull`

将信息从 CodeCommit 存储库拉取到本地存储库时是必需的。这只是一种 IAM 策略权限，不是 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GitPush

操作： codecommit:Git Push

将信息从本地存储库推送到 CodeCommit 存储库时是必需的。这只是一种 IAM 策略权限，不是 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

分支操作权限

以下权限允许或拒绝对 CodeCommit 存储库中的分支执行操作。这些权限仅适用于在 CodeCommit 控制台和使用 CodeCommit API 执行的操作，以及使用 AWS CLI 执行的命令。它们与可以使用 Git 协议执行的类似操作无关。例如，git show-branch -r 命令使用 Git 协议显示存储库的远程分支及其提交的列表。它不受任何 CodeCommit ListBranches 操作权限的影响。

有关为分支创建策略的更多信息，请参阅限制推送和合并到中的分支 AWS CodeCommit和客户管理型策略示例。

针对分支的 CodeCommit API 操作和操作所需的权限

CreateBranch

操作： codecommit:CreateBranch

在 CodeCommit 存储库中创建分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DeleteBranch

操作： codecommit:DeleteBranch

从 CodeCommit 存储库中删除分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetBranch

操作： codecommit:GetBranch

获取有关 CodeCommit 存储库中某个分支的详细信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

ListBranches

操作： codecommit:ListBranches

获取 CodeCommit 存储库中分支的列表时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesByFastForward

操作： codecommit:MergeBranchesByFastForward

在 CodeCommit 存储库中使用快进合并策略合并两个分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesBySquash

操作： codecommit:ListBranches

在 CodeCommit 存储库中使用压缩合并策略合并两个分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesByThreeWay

操作： codecommit:ListBranches

在 CodeCommit 存储库中使用三向合并策略合并两个分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdateDefaultBranch

操作： codecommit:UpdateDefaultBranch

更改 CodeCommit 存储库中的默认分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对合并的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库中的合并执行操作。这些权限与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的

类似操作无关。有关分支的相关权限，请参阅分支操作权限。有关拉取请求的相关权限，请参阅针对拉取请求的操作所需的权限。

针对合并命令的 CodeCommit API 操作和操作所需的权限

BatchDescribeMergeConflicts

操作： codecommit:BatchDescribeMergeConflicts

返回有关 CodeCommit 存储库中的提交之间的合并冲突的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreateUnreferencedMergeCommit

操作： codecommit:CreateUnreferencedMergeCommit

在 CodeCommit 存储库中的两个分支或提交之间创建未引用的提交以进行比较和找出任何潜在冲突时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DescribeMergeConflicts

操作： codecommit:DescribeMergeConflicts

返回有关 CodeCommit 存储库中的潜在合并中的文件的基本、源和目标版本之间的合并冲突的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeCommit

操作： codecommit:GetMergeCommit

返回有关 CodeCommit 存储库中的源和目标提交之间的合并的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeOptions

操作： codecommit:GetMergeOptions

返回有关 CodeCommit 存储库中的两个分支或提交说明符之间的可用合并选项的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对拉取请求的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库中的拉取请求执行操作。这些权限与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的类似操作无关。有关相关评论权限，请参阅针对评论的操作所需的权限。

针对拉取请求的 CodeCommit API 操作和操作所需的权限

BatchGetPullRequests

操作： `codecommit:BatchGetPullRequests`

返回有关 CodeCommit 存储库中一个或多个拉取请求的信息时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

CreatePullRequest

操作： `codecommit:CreatePullRequest`

在 CodeCommit 存储库中创建拉取请求时是必需的。

资源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

CreatePullRequestApprovalRule

操作： `codecommit:CreatePullRequestApprovalRule`

为 CodeCommit 存储库中的拉取请求创建审批规则时是必需的。

资源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

DeletePullRequestApprovalRule

操作： `codecommit:DeletePullRequestApprovalRule`

删除 CodeCommit 存储库中拉取请求的审批规则时是必需的。

资源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

DescribePullRequestEvents

操作： `codecommit:DescribePullRequestEvents`

返回有关 CodeCommit 存储库中的一个或多个拉取请求事件的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

EvaluatePullRequestApprovalRules

操作： codecommit:EvaluatePullRequestApprovalRules

评估拉取请求是否满足 CodeCommit 存储库中关联的审批规则指定的所有条件时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentsForPullRequest

操作： codecommit:GetCommentsForPullRequest

返回对拉取请求的评论时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommitsFromMergeBase

操作： codecommit:GetCommitsFromMergeBase

返回潜在合并上下文中各提交之间的差异的相关信息时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeConflicts

操作： codecommit:GetMergeConflicts

返回有关拉取请求中源分支和目标分支之间的合并冲突的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequest

操作： codecommit:GetPullRequest

返回有关 CodeCommit 存储库中拉取请求的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequestApprovalStates

操作： codecommit:GetPullRequestApprovalStates

返回有关指定拉取请求的审批状态的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequestOverrideState

操作： codecommit:GetPullRequestOverrideState

返回有关是否已为拉取请求搁置（覆盖）审批规则的信息时是必需的，如果是，则返回覆盖规则的用户或身份的 Amazon 资源名称 (ARN) 及其对拉取请求的要求。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

ListPullRequests

操作： codecommit:ListPullRequests

列出存储库中的拉取请求时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestByFastForward

操作： codecommit:MergePullRequestByFastForward

关闭拉取请求并尝试使用快进合并策略将源分支合并到拉取请求的目标分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestBySquash

操作： codecommit:MergePullRequestBySquash

关闭拉取请求并尝试使用压缩合并策略将源分支合并到拉取请求的目标分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestByThreeWay

操作： codecommit:MergePullRequestByThreeWay

关闭拉取请求并尝试使用三向合并策略将源分支合并到拉取请求的目标分支时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

OverridePullRequestApprovalRules

操作： codecommit:OverridePullRequestApprovalRules

为 CodeCommit 存储库中的拉取请求搁置所有审批规则要求时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentForPullRequest

操作： codecommit:PostCommentForPullRequest

对 CodeCommit 存储库中的拉取请求发布评论时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestApprovalRuleContent

操作： codecommit:UpdatePullRequestApprovalRuleContent

更改 CodeCommit 存储库中拉取请求的审批规则结构时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestApprovalState

操作： codecommit:UpdatePullRequestApprovalState

更新 CodeCommit 存储库中拉取请求的审批状态时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestDescription

操作： codecommit:UpdatePullRequestDescription

更改 CodeCommit 存储库中拉取请求的说明时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestStatus

操作： codecommit:UpdatePullRequestStatus

更改 CodeCommit 存储库中拉取请求的状态时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestTitle

操作： codecommit:UpdatePullRequestTitle

更改 CodeCommit 存储库中拉取请求的标题时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

针对审批规则模板的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库中的审批规则模板执行操作。这些权限仅与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的类似操作无关。有关拉取请求的相关权限，请参阅针对拉取请求的操作所需的权限。

针对审批规则模板的 CodeCommit API 操作和操作所需的权限

AssociateApprovalRuleTemplateWithRepository

操作： `codecommit:AssociateApprovalRuleTemplateWithRepository`

将模板与 Amazon Web Services 账户中的指定存储库关联时是必需的。关联后，此操作会根据在指定存储库中创建的每个拉取请求的模板条件，自动创建与之匹配的审批规则。

资源：*

BatchAssociateApprovalRuleTemplateWithRepositories

操作： `codecommit:BatchAssociateApprovalRuleTemplateWithRepositories`

将模板与 Amazon Web Services 账户中的一个或多个指定存储库关联时是必需的。

资源：*

BatchDisassociateApprovalRuleTemplateFromRepositories

操作： `codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories`

取消模板与 Amazon Web Services 账户中的一个或多个指定存储库的关联时是必需的。

资源：*

CreateApprovalRuleTemplate

操作： `codecommit:CreateApprovalRuleTemplate`

创建随后可与 Amazon Web Services 账户中的一个或多个存储库关联的审批规则模板时是必需的。

资源：*

DeleteApprovalRuleTemplate

操作：codecommit:DeleteApprovalRuleTemplate

从 AWS 账户中删除审批规则模板时是必需的。

资源：*

DisassociateApprovalRuleTemplateFromRepository

操作：codecommit:DisassociateApprovalRuleTemplateFromRepository

取消指定模板与 Amazon Web Services 账户中的存储库的关联时是必需的。它不会删除已使用模板创建的拉取请求的审批规则。

资源：*

GetApprovalRuleTemplate

操作：codecommit:GetApprovalRuleTemplate

返回有关 Amazon Web Services 账户中审批规则模板的信息时是必需的。

资源：*

ListApprovalRuleTemplates

操作：codecommit:ListApprovalRuleTemplates

列出 Amazon Web Services 账户中的审批规则模板时是必需的。

资源：*

ListAssociatedApprovalRuleTemplatesForRepository

操作：codecommit:ListAssociatedApprovalRuleTemplatesForRepository

列出与 Amazon Web Services 账户中的指定存储库关联的所有审批规则模板时是必需的。

资源：*

ListRepositoriesForApprovalRuleTemplate

操作：codecommit:ListRepositoriesForApprovalRuleTemplate

列出与 Amazon Web Services 账户中的指定审批规则模板关联的所有存储库时是必需的。

资源：*

UpdateApprovalRuleTemplateContent

操作： codecommit:UpdateApprovalRuleTemplateContent

更新 Amazon Web Services 账户中的审批规则模板的内容时是必需的。

资源：*

UpdateApprovalRuleTemplateDescription

操作： codecommit:UpdateApprovalRuleTemplateDescription

更新 Amazon Web Services 账户中的审批规则模板的描述时是必需的。

资源：*

UpdateApprovalRuleTemplateName

操作： codecommit:UpdateApprovalRuleTemplateName

更新 AWS 账户中的审批规则模板的名称时是必需的。

资源：*

针对单个文件的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库中的单个文件执行操作。这些权限仅与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的类似操作无关。例如，git push 命令通过使用 Git 协议将新文件和更改后的文件推送到 CodeCommit 存储库。它不受任何 CodeCommit PutFile 操作权限的影响。

针对单个文件的 CodeCommit API 操作和操作所需的权限

DeleteFile

操作： codecommit:DeleteFile

在 CodeCommit 控制台中删除 CodeCommit 存储库中指定分支中的指定文件时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetBlob

操作： codecommit:GetBlob

在 CodeCommit 控制台中查看 CodeCommit 存储库中单个文件的编码内容时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetFile

操作： codecommit:GetFile

在 CodeCommit 控制台中查看 CodeCommit 存储库中指定文件及其元数据的编码内容时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetFolder

操作： codecommit:GetFolder

在 CodeCommit 控制台中查看 CodeCommit 存储库中指定文件夹的内容时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PutFile

操作： codecommit:PutFile

对于通过 CodeCommit 控制台、CodeCommit API 或 AWS CLI 将新文件或修改后的文件添加到 CodeCommit 存储库是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对评论的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库中的评论执行操作。这些权限与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。有关对拉取请求中的评论的相关权限，请参阅针对拉取请求的操作所需的权限。

针对存储库的 CodeCommit API 操作和操作所需的权限

DeleteCommentContent

操作： codecommit:DeleteCommentContent

删除对存储库中的更改、文件或提交做出的评论的内容时是必需的。评论无法删除，但如果用户拥有此权限，则可以删除评论内容。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetComment

操作： codecommit:GetComment

返回有关对 CodeCommit 存储库中的更改、文件或提交做出的评论的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentReactions

操作： codecommit:GetCommentReactions

返回有关对 CodeCommit 存储库中的更改、文件或提交做出的评论的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentsForComparedCommit

操作： codecommit:GetCommentsForComparedCommit

返回有关对某一 CodeCommit 存储库中的两个提交之间的对比做出的评论的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentForComparedCommit

操作： codecommit:PostCommentForComparedCommit

对某一 CodeCommit 存储库中的两个提交之间的对比进行评论时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentReply

操作： codecommit:PostCommentReply

对提交之间的对比或 CodeCommit 存储库中的拉取请求的评论创建回复时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PutCommentReaction

操作： codecommit:PutCommentReaction

对提交之间的对比或 CodeCommit 存储库中的拉取请求的评论创建回复时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdateComment

操作： codecommit:UpdateComment

对评论之间的比较或拉取请求的评论进行编辑时是必需的。只有评论作者可以编辑评论。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对已提交代码的操作所需的权限

以下权限允许或拒绝对已提交到 CodeCommit 存储库的代码执行操作。这些权限与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的类似操作无关。例如，git commit 命令使用 Git 协议为存储库中的分支创建提交。它不受任何 CodeCommit CreateCommit 操作权限的影响。

显式拒绝这些权限中的某些权限可能会在 CodeCommit 控制台中导致意外后果。例如，将 GetTree 设置为 Deny 将阻止用户在控制台中浏览存储库的内容，但不会阻止用户查看存储库中文件的内容 (例如，用户可以打开并浏览以电子邮件方式发来的文件链接)。将 GetBlob 设置为 Deny 将阻止用户查看文件的内容，但不会阻止用户浏览存储库的结构。将 GetCommit 设置为 Deny 将阻止用户检索有关提交的详细信息。将 GetObjectIdentifier 设置为 Deny 将阻止大部分代码浏览功能。如果在策略中将所有这三个操作都设为 Deny，则具有该策略的用户无法在 CodeCommit 控制台中浏览代码。

针对已提交代码的 CodeCommit API 操作和操作所需的权限

BatchGetCommits

操作： codecommit:BatchGetCommits

返回有关 CodeCommit 存储库中的一个或多个提交的信息时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreateCommit

操作： codecommit:CreateCommit

创建提交时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommit

操作： codecommit:GetCommit

返回有关提交的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommitHistory

操作： codecommit:GetCommitHistory

返回有关存储库中提交历史记录的信息时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetDifferences

操作： codecommit:GetDifferences

返回有关某个提交说明符 (如分支、标签、HEAD、提交 ID 或其他完全限定引用) 差异的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetObjectIdentifier

操作： codecommit:GetObjectIdentifier

将 blob、树和提交解析为其标识符时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetReferences

操作： codecommit:GetReferences

返回所有引用 (如分支和标签) 时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetTree

操作： `codecommit:GetTree`

在 CodeCommit 控制台中查看 CodeCommit 存储库中指定树的内容时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： `arn:aws:codecommit:region:account-id:repository-name`

针对存储库的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库执行操作。这些权限与使用 CodeCommit 控制台和 CodeCommit API 执行的操作以及使用 AWS CLI 执行的命令有关。它们与可以使用 Git 协议执行的类似操作无关。

针对存储库的 CodeCommit API 操作和操作所需的权限

BatchGetRepositories

操作： `codecommit:BatchGetRepositories`

获取有关 Amazon Web Services 账户中的多个 CodeCommit 存储库的信息时是必需的。在 `Resource` 中，必须指定允许（或拒绝）用户访问其中信息的所有 CodeCommit 存储库的名称。

资源： `arn:aws:codecommit:region:account-id:repository-name`

CreateRepository

操作： `codecommit:CreateRepository`

创建 CodeCommit 存储库时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

DeleteRepository

操作： `codecommit:DeleteRepository`

删除 CodeCommit 存储库时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

GetRepository

操作： `codecommit:GetRepository`

获取有关单个 CodeCommit 存储库的信息时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

ListRepositories

操作： `codecommit:ListRepositories`

获取 Amazon Web Services 账户的多个 CodeCommit 存储库的名称和系统 ID 列表时是必需的。对于该操作来说，`Resource` 唯一允许的值是所有存储库 (`*`)。

资源： `*`

UpdateRepositoryDescription

操作： `codecommit:UpdateRepositoryDescription`

更改 CodeCommit 存储库的说明时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

UpdateRepositoryName

操作： `codecommit:UpdateRepositoryName`

更改 CodeCommit 存储库的名称时是必需的。在 `Resource` 中，必须同时指定允许更改的 CodeCommit 存储库和新存储库的名称。

资源： `arn:aws:codecommit:region:account-id:repository-name`

针对标签的操作所需的权限

以下权限允许或拒绝对 CodeCommit 资源的 AWS 标签执行操作。

针对标签的 CodeCommit API 操作和操作所需的权限

ListTagsForResource

操作： `codecommit:ListTagsForResource`

返回有关为 CodeCommit 中的资源配置的 AWS 标签的信息时是必需的。

资源： `arn:aws:codecommit:region:account-id:repository-name`

TagResource

操作： codecommit:TagResource

为存储库添加或编辑 AWS 标签时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UntagResource

操作： codecommit:UntagResource

从 CodeCommit 中的资源删除 AWS 标签时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对触发器的操作所需的权限

以下权限允许或拒绝对 CodeCommit 存储库的触发器执行操作。

针对触发器的 CodeCommit API 操作和操作所需的权限

GetRepositoryTriggers

操作： codecommit:GetRepositoryTriggers

返回有关为存储库配置的触发器的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PutRepositoryTriggers

操作： codecommit:PutRepositoryTriggers

创建、编辑或删除存储库触发器时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

TestRepositoryTriggers

操作： codecommit:TestRepositoryTriggers

通过向为触发器配置的主题或函数发送数据来测试存储库触发器的功能时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

针对 CodePipeline 集成的操作所需的权限

为了使 CodePipeline 在源操作中使用 CodeCommit 存储库作为管道，您必须向 CodePipeline 的服务角色授予下表中列出的所有权限。如果未在服务角色中设置这些权限或将这些权限设为 **Deny**，则在对存储库进行更改时，管道不会自动运行，并且无法手动发布更改。

针对 CodePipeline 集成的 CodeCommit API 操作和操作所需的权限

GetBranch

操作： codecommit:GetBranch

获取有关 CodeCommit 存储库中某个分支的详细信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommit

操作： codecommit:GetCommit

返回有关提交的信息时是必需的。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UploadArchive

操作： codecommit:UploadArchive

允许 CodePipeline 服务角色将存储库更改上传到管道中时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetUploadArchiveStatus

操作： codecommit:GetUploadArchiveStatus

确定存档的上传状态：是正在进行、完成、已取消还是出现错误时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CancelUploadArchive

操作： `codecommit:CancelUploadArchive`

取消将存档上传到管道的操作时是必需的。这只是一种 IAM policy 权限，不是可调用的 API 操作。

资源： `arn:aws:codecommit:region:account-id:repository-name`

AWS CodeCommit 如何与 IAM 协同工作

在使用 IAM 管理对 CodeCommit 的访问之前，您应了解哪些 IAM 功能可与 CodeCommit 结合使用。要大致了解 CodeCommit 和其他 AWS 服务如何与 IAM 一起使用，请参阅《IAM 用户指南》中的与 IAM 配合使用的 AWS 服务。

主题

- 条件键
- 示例

条件键

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，您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` 元素中指定多个密钥，则 AWS 使用逻辑 AND 运算评估它们。如果您要为单个条件密钥指定多个值，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。

您也可以在指定条件时使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS 支持全局条件键和特定于服务的条件键。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。

CodeCommit 定义了自己的一组条件键，还支持使用一些全局条件键。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。

某些 CodeCommit 操作支持 `codecommit:References` 条件键。有关使用此键的示例策略，请参阅示例 4：拒绝或允许对分支执行操作。

要查看 CodeCommit 条件键的列表，请参阅《IAM 用户指南》中的 AWS CodeCommit 条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 AWS CodeCommit 定义的操作。

示例

要查看 CodeCommit 基于身份的策略示例，请参阅 AWS CodeCommit 基于身份的策略示例。

CodeCommit 基于资源的策略

CodeCommit 不支持基于资源的策略。

基于 CodeCommit 标签的授权

您可以将标签附加到 CodeCommit 资源或将请求中的标签传递到 CodeCommit。要基于标签控制访问，需要使用 `codecommit:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件密钥在策略的条件元素中提供标签信息。有关为 CodeCommit 资源添加标签的更多信息，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。有关标记策略的更多信息，请参阅标记 AWS 资源。

CodeCommit 还支持基于会话标签的策略。有关更多信息，请参阅会话标签。

在 CodeCommit 中使用标签提供标识信息

CodeCommit 支持使用会话标签，这些标签是您在代入 IAM 角色、使用临时凭证或在 AWS Security Token Service (AWS STS) 中对用户进行联合身份验证时传递的键值对属性。您还可以将标签与 IAM 用户关联。您可以使用这些标签中提供的信息，以便更轻松地识别谁进行了更改或导致了事件。CodeCommit 在 CodeCommit 事件中包含具有以下键名称的标签的值：

键名称	值
`displayName`	要显示并要与用户关联的便于阅读的名称（例如 Mary Major 或 Saanvi Sarkar）。
`emailAddress`	您希望为用户显示并与用户关联的电子邮件地址（例如 mary_major@example.com 或 saanvi_sarkar@example.com）。

如果提供了这些信息，CodeCommit 会将其包含在发送到 Amazon EventBridge 和 Amazon CloudWatch Events 的事件中。有关更多信息，请参阅在 Amazon EventBridge 和 Amazon CloudWatch Events 中监控 CodeCommit 事件。

要使用会话标记，角色必须具有包含设置为 Allow 的 sts:TagSession 权限的策略。如果使用联合访问，则可以在设置过程中配置显示名称和电子邮件标签信息。例如，如果您使用的是 Azure Active Directory，则可能会提供以下声明信息：

声明名称	值
https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayNam e	user.displayname
https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress	user.mail

您可以使用 AWS CLI 通过 AssumeRole 传递 displayName 和 emailAddress 的会话标签。例如，想要代入一个名为 *Developer* 的角色且希望关联其姓名 *Mary Major* 的用户可能使用类似于以下内容的 assume-role 命令：

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer \
--role-session-name Mary-Major \
--tags Key=displayName,Value="Mary Major"
 Key=emailAddress,Value="mary_major@example.com" \
--external-id Example987
```

有关详细信息，请参阅 AssumeRole。

您可以使用 AssumeRoleWithSAML 操作返回包含 displayName 和 emailAddress 标签的一组临时凭证。您可以在访问 CodeCommit 存储库时使用这些标签。这要求您的公司或组织已将您的第三方 SAML 解决方案与 AWS 集成。如果是这样，则可以将 SAML 属性作为会话标签传递。例如，如果您想将姓名为 *Saanvi Sarkar* 的用户的显示名称和电子邮件地址的标识属性作为会话标记传递：

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayName">
  <AttributeValue>Saanvi Sarkar</AttributeValue>
```

```
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress">
  <AttributeValue>saanvi_sarkar@example.com</AttributeValue>
</Attribute>
```

有关更多信息，请参阅使用 AssumeRoleWithSAML 传递会话标签。

您可以使用 AssumeRoleWithIdentity 操作返回包含 displayName 和 emailAddress 标签的一组临时凭证。您可以在访问 CodeCommit 存储库时使用这些标签。要从 OpenID Connect (OIDC) 传递会话标签，您必须在 JSON Web 令牌 (JWT) 中包含会话标签。例如，用于调用 AssumeRoleWithWebIdentity 的已解码 JWP 令牌，其中包括名为 *Li Juan* 的用户的 displayName 和 emailAddress 会话标签：

```
{
    "sub": "lijuan",
    "aud": "ac_oic_client",
    "jti": "ZYUCeREXAMPLE",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "displayName": ["Li Juan"],
            "emailAddress": ["li_juan@example.com"],
        },
        "transitive_tag_keys": [
            "displayName",
            "emailAddress"
        ]
    }
}
```

有关更多信息，请参阅使用 AssumeRoleWithWebIdentity 传递会话标签。

您可以使用 GetFederationToken 操作返回包含 displayName 和 emailAddress 标签的一组临时凭证。您可以在访问 CodeCommit 存储库时使用这些标签。例如，要使用 AWS CLI 获取包含 displayName 和 emailAddress 标签的联合令牌：

```
aws sts get-federation-token \
--name my-federated-user \
```

```
--tags key=displayName,value="Nikhil Jayashankar"
 key=emailAddress,value=nikhil_jayashankar@example.com
```

有关更多信息，请参阅使用 GetFederationToken 传递会话标签。

CodeCommit IAM 角色

IAM 角色是 Amazon Web Services 账户中具有特定权限的实体。

在 CodeCommit 中使用临时凭证

可以使用临时凭证进行联合身份验证登录，代入 IAM 角色或代入跨账户角色。可以通过调用AWS STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

CodeCommit 支持使用临时凭证。有关更多信息，请参阅使用轮换凭证连接到 AWS CodeCommit 存储库。

服务关联角色

服务相关角色允许AWS服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

CodeCommit 不使用服务相关角色。

服务角色

此功能允许服务代表您代入服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

CodeCommit 不使用服务角色。

AWS CodeCommit 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 CodeCommit 资源的权限。它们还无法使用AWS Management Console、AWS CLI或AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

有关策略示例，请参阅以下主题：

- 示例 1：允许用户在单个 AWS 区域中执行 CodeCommit 操作

- 示例 2：允许用户对单个存储库使用 Git
- 示例 3：允许从指定 IP 地址范围连接的用户访问存储库
- 示例 4：拒绝或允许对分支执行操作
- 示例 5：使用标签拒绝或允许对存储库执行操作
- 使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实操
- 使用 CodeCommit 控制台
- 允许用户查看他们自己的权限
- 基于标签查看 CodeCommit 存储库

策略最佳实操

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 CodeCommit 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- AWS 托管式策略及转向最低权限许可入门 – 要开始向用户和工作负载授予权限，请使用 AWS 托管式策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略或工作职能的 AWS 托管式策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy 语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，有助于制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。

- 需要多重身份验证（MFA）– 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 CodeCommit 控制台

要访问 AWS CodeCommit控制台，您必须拥有一组最低的权限。这些权限必须允许您列出和查看有关您 Amazon Web Services 账户中的 CodeCommit 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

要确保这些实体仍可使用 CodeCommit 控制台，也可向这些实体附加以下 AWS 托管策略。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限：

有关更多信息，请参阅将基于身份的策略 (IAM policy) 用于 CodeCommit。

对于只需要调用 AWS CLI 或 AWS API 的用户，无需为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
```

```
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

基于标签查看 CodeCommit

您可以在基于身份的策略中使用条件，以便基于标签控制对 CodeCommit 资源的访问。有关演示如何执行此操作的示例策略，请参阅示例 5：使用标签拒绝或允许对存储库执行操作。

有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

对 AWS CodeCommit 身份和访问进行故障排除

以下信息可帮助您诊断和修复在使用 CodeCommit 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 CodeCommit 中执行操作
- 我无权执行 iam:PassRole
- 我想要查看我的访问密钥
- 我是管理员并希望允许其他人访问 CodeCommit
- 我希望允许我的 Amazon Web Services 账户之外的用户访问我的 CodeCommit 资源

我无权在 CodeCommit 中执行操作

如果 AWS Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。管理员是向您提供登录凭证的人。

有关更多信息，请参阅使用控制台所需的权限。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 CodeCommit。

有些 AWS 服务 允许将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 CodeCommit 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系 AWS 管理员。管理员是向您提供登录凭证的人。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 AKIAIOSFODNN7EXAMPLE）和秘密访问密钥（例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的 AWS 账户 账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 CodeCommit

要允许其他人访问 CodeCommit，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 AWS。然后，您必须将策略附加到该实体，以便在 CodeCommit 中向其授予正确的权限。

要立即开始使用，请参阅《IAM 用户指南》中的创建您的第一个 IAM 委派用户和组。

我希望允许我的 Amazon Web Services 账户之外的用户访问我的 CodeCommit 资源

有关更多信息，请参阅使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限。

AWS CodeCommit 中的故障恢复能力

AWS 全球基础设施围绕 AWS 区域 和可用区构建。AWS 区域 提供多个在物理上独立且隔离的可用区，这些可用区与延迟率低、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。

CodeCommit 存储库或 CodeCommit 审批规则模板位于创建时所在的 AWS 区域。有关更多信息，请参阅的地区和 Git 连接终端节点 AWS CodeCommit。为了提高存储库的恢复能力，您可以将 Git 客户端配置为同时推送到两个存储库。有关更多信息，请参阅将提交推送到其他 Git 存储库。

有关AWS 区域和可用区的更多信息，请参阅AWS全球基础设施。

AWS CodeCommit 中的基础设施安全性

作为一项托管式服务，AWS CodeCommit 由Amazon Web Services：安全流程概览白皮书中所述的AWS全球网络安全程序提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 CodeCommit。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

必须使用访问密钥 ID 以及与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

您可以从任何网络位置调用这些 API 操作，但 CodeCommit 基于源 IP 地址应用了一些限制。您还可以使用 CodeCommit 策略来控制来自特定 Amazon Virtual Private Cloud (Amazon VPC) 端点或特定

VPC 的访问。事实上，这隔离了在 AWS 网络中仅从特定 VPC 到给定 CodeCommit 资源的网络访问。

有关更多信息，请参阅下列内容：

- 示例 1：允许用户在单个 AWS 区域中执行 CodeCommit 操作
- 示例 3：允许从指定 IP 地址范围连接的用户访问存储库
- AWS CodeCommit 与接口 VPC 终端节点一起使用

监控 AWS CodeCommit

监控是保持 CodeCommit 及您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS 提供了以下一些监控工具来监控 CodeCommit、在出现错误时进行报告并适时自动采取措施。

- 您可以使用 Amazon EventBridge 自动执行您的 AWS 服务并自动响应系统事件，例如应用程序可用性问题或资源更改。AWS 服务中的事件将近乎实时传输到 EventBridge。您可以编写简单的规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。有关更多信息，请参阅 Amazon EventBridge 用户指南和在 Amazon EventBridge 和 Amazon CloudWatch Events 中监控 CodeCommit 事件。
- Amazon CloudWatch Events 提供近乎实时的系统事件流，这些事件描述 AWS 资源的更改。CloudWatch Events 支持自动事件驱动型计算，因为您可以编写规则，以监控某些事件和在这些事件发生时在其他 AWS 服务中触发自动操作。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南和在 Amazon EventBridge 和 Amazon CloudWatch Events 中监控 CodeCommit 事件。
- Amazon CloudWatch Logs 可用于监控、存储和访问来自 CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- AWS CloudTrail 捕获由您的 Amazon Web Services 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 桶。您可以标识哪些用户和账户调用了 AWS、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 AWS CloudTrail 用户指南和使用 AWS CodeCommit 记录 AWS CloudTrail API 调用。

在 Amazon EventBridge 和 Amazon CloudWatch Events 中监控 CodeCommit 事件

您可以在 EventBridge 中监控 AWS CodeCommit 事件，这将从您自己的应用程序、软件即服务 (SaaS) 应用程序和 AWS 服务传输实时数据流。EventBridge 将该数据路由到 AWS Lambda 和 Amazon Simple Notification Service 等目标。这些事件与 Amazon CloudWatch Events 中出现的事件相同，可提供近乎实时的系统事件流，这些事件描述 AWS 资源的更改。

以下示例显示了 CodeCommit 的事件。

> **Note**
>
> CodeCommit 支持提供事件中的会话标签所包含的 `displayName` 和 `emailAddress` 信息（如果该信息可用）。有关更多信息，请参阅会话标签和在 CodeCommit 中使用标签提供标识信息。

主题

- pullRequestApprovalStateChanged 事件
- pullRequestApprovalRuleUpdated 事件
- reactionCreated 事件
- reactionUpdated 事件

referenceCreated 事件

在此示例事件中，已在名为 MyDemoRepo 的存储库中创建了一个名为 myBranch 的分支。

```
{
   "version": "0",
   "id": "01234567-EXAMPLE",
   "detail-type": "CodeCommit Repository State Change",
   "source": "aws.codecommit",
   "account": "123456789012",
   "time": "2019-06-12T10:23:43Z",
   "region": "us-east-2",
   "resources": [
     "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail": {
     "event": "referenceCreated",
     "repositoryName": "MyDemoRepo",
     "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
     "referenceType": "branch",
     "referenceName": "myBranch",
     "referenceFullName": "refs/heads/myBranch",
     "commitId": "3e5983DESTINATION"
   }
 }
```

referenceUpdated 事件

在此示例事件中，已在名为 MyDemoRepo 的存储库中更新了一个名为 myBranch 的分支。

```
{
   "version": "0",
   "id": "01234567-EXAMPLE",
   "detail-type": "CodeCommit Repository State Change",
   "source": "aws.codecommit",
```

```
    "account": "123456789012",
    "time": "2019-06-12T10:23:43Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
      "event": "referenceUpdated",
      "repositoryName": "MyDemoRepo",
      "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
      "referenceType": "branch",
      "referenceName": "myBranch",
      "referenceFullName": "refs/heads/myBranch",
      "commitId": "7f0103fMERGE",
      "oldCommitId": "3e5983DESTINATION",
      "baseCommitId": "3e5a9bf1BASE",
      "sourceCommitId": "26a8f2SOURCE",
      "destinationCommitId": "3e5983DESTINATION",
      "mergeOption": "THREE_WAY_MERGE",
      "conflictDetailsLevel": "LINE_LEVEL",
      "conflictResolutionStrategy": "AUTOMERGE"
    }
}
```

referenceDeleted 事件

在此示例事件中，已在名为 MyDemoRepo 的存储库中删除了一个名为 myBranch 的分支。

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "event": "referenceDeleted",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
```

```
    "referenceType": "branch",
    "referenceName": "myBranch",
    "referenceFullName": "refs/heads/myBranch",
    "oldCommitId": "26a8f2EXAMPLE"
  }
}
```

unreferencedMergeCommitCreated 事件

在此示例事件中，已在名为 MyDemoRepo 的存储库中创建了一个未引用的合并提交。

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "CodeCommit Repository State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "event": "unreferencedMergeCommitCreated",
    "repositoryName": "MyDemoRepo",
    "repositoryId": "12345678-1234-5678-abcd-12345678abcd",
    "commitId": "7f0103fMERGE",
    "baseCommitId": "3e5a9bf1BASE",
    "sourceCommitId": "26a8f2SOURCE",
    "destinationCommitId": "3e5983DESTINATION",
    "mergeOption": "SQUASH_MERGE",
    "conflictDetailsLevel": "LINE_LEVEL",
    "conflictResolutionStrategy": "AUTOMERGE"
  }
}
```

commentOnCommitCreated 事件

在此示例事件中，名为 Mary_Major 的联合用户对提交进行了评论。在此示例中，她的联合身份提供程序为 displayName 和 emailAddress 配置了会话标签。该信息包含在事件中。

```
{
```

```
  "version": "0",
  "id": "e9dce2e9-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-09-29T20:20:39Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following repository:
 MyDemoRepo. The display name for the user is Mary Major. The email address for
 the user is mary_major@example.com. The user arn:aws:sts::123456789012:federated-
user/Mary_Major made a comment. The comment was made on the following comment ID:
 463bEXAMPLE.... For more information, go to the AWS CodeCommit console at https://us-
east-2.console.aws.amazon.com/codecommit/home?region=us-east-2#/repository/MyDemoRepo/
compare/3c5dEXAMPLE...f4d5EXAMPLE#463bEXAMPLE....",
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnCommitCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Mary_Major",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnCommitUpdated 事件

在此示例事件中，代入名为 Admin 的角色（会话名称为 Mary_Major）的用户编辑了关于提交的注释。在此示例中，角色包括了为 displayName 和 emailAddress 配置的会话标签。该信息包含在事件中。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Commit",
  "source": "aws.codecommit",
  "account": "123456789012",
```

```
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "53812581",
    "beforeCommitId": "03314446",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "commentId": "a7e5471e-EXAMPLE",
    "event": "commentOnCommitUpdated",
    "inReplyTo": "bdb07d47-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Mary
 Major. The email address for the user is mary_major@example.com. The user
 arn:aws:sts::123456789012:federated-user/Mary_Major updated a comment or
 replied to a comment. The comment was made on the following comment ID:
 bdb07d47-6fe9-47b0-a839-b93cc743b2ac:468cd1cb-2dfb-4f68-9636-8de52431d1d6.
 For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/
compare/033144464617842958996982309670958225176 8/.../53812581502937833614716802771360172913 82?
region\u003dus-east-2",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo",
    "displayName": "Mary Major",
    "emailAddress": "mary_major@example.com"
  }
}
```

commentOnPullRequestCreated 事件

在此示例事件中，名为 Saanvi_Sarkar 的联合用户对拉取请求进行了评论。在此示例中，她的联合身份提供程序为 displayName 和 emailAddress 配置了会话标签。该信息包含在事件中。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
```

```
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "beforeCommitId": "3c5dEXAMPLE",
    "repositoryId": "7dd1EXAMPLE...",
    "inReplyTo": "695bEXAMPLE...",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar made a comment. The comment
 was made on the following Pull Request: 201. For more information, go to the AWS
 CodeCommit console https://us-east-2.console.aws.amazon.com/codecommit/home?region=us-
east-2#/repository/MyDemoRepo/pull-request/201/activity#3276EXAMPLE...",
    "commentId": "463bEXAMPLE...",
    "afterCommitId": "f4d5EXAMPLE",
    "event": "commentOnPullRequestCreated",
    "repositoryName": "MyDemoRepo",
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "pullRequestId": "201",
    "displayName": "Saanvi Sarkar",
    "emailAddress": "saanvi_sarkar@example.com"
  }
}
```

commentOnPullRequestUpdated 事件

在此示例事件中，名为 Saanvi_Sarkar 的联合用户编辑了对拉取请求的评论。在此示例中，她的联合身份提供程序为 displayName 和 emailAddress 配置了会话标签。该信息包含在事件中。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Comment on Pull Request",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "afterCommitId": "96814774EXAMPLE",
    "beforeCommitId": "6031971EXAMPLE",
```

```
    "callerUserArn": "arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar",
    "commentId": "40cb52f0-EXAMPLE",
    "event": "commentOnPullRequestUpdated",
    "inReplyTo": "1285e713-EXAMPLE",
    "notificationBody": "A comment event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. The display name for the user is Saanvi
 Sarkar. The email address for the user is saanvi_sarkar@example.com. The user
 arn:aws:sts::123456789012:federated-user/Saanvi_Sarkar updated a comment or
 replied to a comment. The comment was made on the following Pull Request:
 1. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1/activity#40cb52f0-aac7-4c43-b771-601eff02EXAMPLE",
    "pullRequestId": "1",
    "repositoryId": "12345678-1234-1234-1234-123456789012",
    "repositoryName": "MyDemoRepo"
  }
}
```

pullRequestCreated 事件

在此示例事件中，在名为 MyDemoRepo 的存储库中创建了一个拉取请求，创建者为代入名为 Admin 的角色（会话名称为 Mary_Major）的用户。未提供会话标记信息，因此该信息不包含在事件中。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT ",
    "description": "An example description.",
    "destinationCommit": "12241970EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestCreated",
    "isMerged": "False",
```

```
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major.
 Event: Created. The pull request was created with the following information: Pull
 Request ID as 1 and title as My Example Pull Request. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/1",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "2774290EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestSourceBranchUpdated 事件

在此示例事件中，代入名为 Admin 的角色（会话名称为 Mary_Major）的用户为 ID 为 1 的拉取请求更新了名为 test-branch 的源分支。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Feb 9 2019 10:18:42 PDT",
    "description": "An example description.",
    "destinationCommit": "7644990EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestSourceBranchUpdated",
    "isMerged": "False",
    "lastModifiedDate": "Tue Feb 9 2019 10:18:42 PDT",
```

```
    "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:sts::123456789012:assumed-role/
Admin/Mary_Major. Event: Updated.  The user updated the following pull request:
 1. The pull request was updated with one or more commits to the source branch:
 test-branch. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Open",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9b4EXAMPLE",
    "sourceCommit": "64875001EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestStatusChanged 事件

在此示例事件中，代入名为 Admin 的角色（会话名称为 Mary_Major）的用户结束了 ID 为 1 的拉取请求。拉取请求未合并。

```
{
  "version": "0",
  "id": "98377d67-EXAMPLE",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-02-09T07:15:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Tue Jun 18 10:34:20 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "95149731EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestStatusChanged",
    "isMerged": "False",
    "lastModifiedDate": "Tue Jun 18 10:34:20 PDT 2019",
```

```
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request status has been updated. The
 status is closed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9bEXAMPLE",
    "sourceCommit": "4409936EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

pullRequestMergeStatusUpdated 事件

在此示例事件中，代入名为 Admin 的角色（会话名称为 Mary_Major）的用户合并了 ID 为 1 的拉取请求。

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "CodeCommit Pull Request State Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-06-12T10:23:43Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
  ],
  "detail": {
    "author": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "callerUserArn": "arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major",
    "creationDate": "Mon Mar 11 14:42:31 PDT 2019",
    "description": "An example description.",
    "destinationCommit": "4376719EXAMPLE",
    "destinationReference": "refs/heads/main",
    "event": "pullRequestMergeStatusUpdated",
    "isMerged": "True",
    "lastModifiedDate": "Mon Mar 11 14:42:31 PDT 2019",
    "mergeOption": "FAST_FORWARD_MERGE",
```

```
    "notificationBody": "A pull request event occurred in the following AWS CodeCommit
 repository: MyDemoRepo. arn:aws:sts::123456789012:assumed-role/Admin/Mary_Major
 updated the following PullRequest 1. The pull request merge status has been updated.
 The status is merged. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/1?region\u003dus-east-2",
    "pullRequestId": "1",
    "pullRequestStatus": "Closed",
    "repositoryNames": ["MyDemoRepo"],
    "revisionId": "bdc0cb9beEXAMPLE",
    "sourceCommit": "0701696EXAMPLE",
    "sourceReference": "refs/heads/test-branch",
    "title": "My Example Pull Request"
  }
}
```

approvalRuleTemplateCreated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户创建了名为 2-approvers-required-for-main 的审批规则模板。

```
{
    "version": "0",
    "id": "f7702227-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateCreated",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
  Additional information: An approval rule template with the following name has been
 created: 2-approvers-required-for-main. The ID of the created template is: d7385967-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
```

```
        "repositories": {}
    }
}
```

approvalRuleTemplateUpdated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户编辑了名为 2-approvers-required-for-main 的审批规则模板。审批规则模板不与任何存储库关联。

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
  "detail-type": "CodeCommit Approval Rule Template Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [

  ],
  "detail": {
    "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
    "approvalRuleTemplateName": "2-approvers-required-for-main",
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateDeleted",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has been
 deleted: 2-approvers-required-for-main. The ID of the updated template is: c9d2b844-
EXAMPLE. For more information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateDeleted 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户删除了名为 2-approvers-required-for-main 的审批规则模板。审批规则模板不与任何存储库关联。

```
{
  "version": "0",
  "id": "66403118-EXAMPLE",
  "detail-type": "CodeCommit Approval Rule Template Change",
  "source": "aws.codecommit",
  "account": "123456789012",
  "time": "2019-11-12T23:03:30Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "approvalRuleTemplateContentSha256": "4f3de6632EXAMPLE",
    "approvalRuleTemplateId": "c9d2b844-EXAMPLE",
    "approvalRuleTemplateName": "2-approvers-required-for-main",
    "callerUserArn": "arn:aws:iam::123456789012:user\Mary_Major",
    "creationDate": "Tue Nov 12 23:03:06 UTC 2019",
    "event": "approvalRuleTemplateUpdated",
    "lastModifiedDate": "Tue Nov 12 23:03:20 UTC 2019",
    "notificationBody": "A approval rule template event occurred in the following AWS
 CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template with the following name has
 been updated: 2-approvers-required-for-main. The ID of the updated template is:
 c9d2b844-EXAMPLE. The after rule template content SHA256 is 4f3de663EXAMPLE. For more
 information, go to the AWS CodeCommit console.",
    "repositories": {}
  }
}
```

approvalRuleTemplateAssociatedWithRepository 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户将名为 2-approvers-required-for-main 的审批规则模板与名为 MyDemoRepo 的存储库关联。

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
```

```
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateAssociatedWithRepository",
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been associated with the
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
}
```

approvalRuleTemplateDisassociatedWithRepository 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户将名为 2-approvers-required-for-main 的审批规则模板与名为 MyDemoRepo 的存储库取消关联。

```
{
    "version": "0",
    "id": "ea1c6d73-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:02:27Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "d7385967-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:02:14 UTC 2019",
        "event": "approvalRuleTemplateDisassociatedFromRepository",
```

```
        "lastModifiedDate": "Wed Nov 06 19:02:14 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been disassociated from the
 following repository: [MyDemoRepo]. For more information, go to the AWS CodeCommit
 console.",
        "repositories": {
            "MyDemoRepo": "92ca7bf2-d878-49ed-a994-336a6cc7c574"
        }
    }
}
```

approvalRuleTemplateBatchAssociatedWithRepositories 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户将名为 2-approvers-required-for-main 的审批规则模板与名为 MyDemoRepo 的存储库以及名为 MyTestRepo 的存储库批量关联。

```
{
    "version": "0",
    "id": "0f861e5b-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-EXAMPLE",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchAssociateApprovalRuleTemplateWithRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user\Mary_Major.
 Additional information: An approval rule template has been batch associated with the
 following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to the
 AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
```

```
        }
    }
}
```

approvalRuleTemplateBatchDisassociatedFromRepositories 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户将名为 2-approvers-required-for-main 的审批规则模板与名为 MyDemoRepo 的存储库以及名为 MyTestRepo 的存储库批量取消关联。

```
{
    "version": "0",
    "id": "e08fc996-EXAMPLE",
    "detail-type": "CodeCommit Approval Rule Template Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-12T23:39:09Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleTemplateContentSha256": "f742eebbEXAMPLE",
        "approvalRuleTemplateId": "c71c1fe0-ff91-4db4-9a45-a86a7b6c474f",
        "approvalRuleTemplateName": "2-approvers-required-for-main",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Tue Nov 12 23:38:57 UTC 2019",
        "event": "batchDisassociateApprovalRuleTemplateFromRepositories",
        "lastModifiedDate": "Tue Nov 12 23:38:57 UTC 2019",
        "notificationBody": "A approval rule template event occurred in the following
 AWS CodeCommit account: 123456789012. User: arn:aws:iam::123456789012:user/Mary_Major.
 Additional information: An approval rule template has been batch disassociated from
 the following repository names: [MyDemoRepo, MyTestRepo]. For more information, go to
 the AWS CodeCommit console.",
        "repositories": {
            "MyDemoRepo": "MyTestRepo"
        }
    }
}
```

pullRequestApprovalRuleCreated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户为 ID 为 227 的拉取请求创建了名为 1-approver-needed 的审批规则。


```
{
    "version": "0",
    "id": "ad860f12-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleCreated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request: 227. Additional information: An approval rule has been
 created with the following name: 1-approver-needed. For more information, go to the
 AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
```

```
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleDeleted 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户为 ID 为 227 的拉取请求删除了名为 1-approver-needed 的审批规则。名称为 Saanvi_Sarkar 的 IAM 用户最初创建了该审批规则。

```
{
    "version": "0",
    "id": "c1c3509d-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Saanvi_Sarkar",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleDeleted",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Created. Pull request: 227. Additional information: An approval rule has been
 deleted: 1-approver-needed was deleted. For more information, go to the AWS CodeCommit
 console https://us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/
MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
```

```
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleOverridden 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户搁置 (OVERRIDE) 了一个拉取请求的审批规则要求。该拉取请求由 IAM 用户名为 Li_Juan 的用户创建。

```
{
    "version": "0",
    "id": "52d2cb73-EXAMPLE",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: An override
 event has occurred for the approval rules for this pull request. Override status:
 OVERRIDE. For more information, go to the AWS CodeCommit console https://us-
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "OVERRIDE",
        "pullRequestId": "227",
```

```
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

在此示例事件中，已恢复拉取请求的审批规则要求 (REVOKE)。

```
{
    "version": "0",
    "id": "2895482d-13eb-b783-270d-76588e6029fa",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleOverridden",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information: An
 override event has occurred for the approval rules for this pull request. Override
 status: REVOKE. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "overrideStatus": "REVOKE",
        "pullRequestId": "227",
```

```
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalStateChanged 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户批准了一个拉取请求。

```
{
    "version": "0",
    "id": "53e5d7e9-986c-1ebf-9d8b-ebef5596da0e",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "APPROVE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. Additional information:
 A user has changed their approval state for the pull request. State change:
 APPROVE. For more information, go to the AWS CodeCommit console https://us-
```

```
east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

在此示例事件中，IAM 用户名为 Mary_Major 的用户撤销了对一个拉取请求的批准。

```
{
    "version": "0",
    "id": "25e183d7-d01a-4e07-2bd9-b2d56ebecc81",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalStatus": "REVOKE",
        "author": "arn:aws:iam::123456789012:user/Li_Juan",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalStateChanged",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
        "notificationBody": "A pull request event occurred in the following AWS
 CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/Mary_Major.
 Event: Updated. Pull request name: 227. Additional information: A user has changed
 their approval state for the pull request. State change: REVOKE. For more information,
```

```
 go to the AWS CodeCommit console https://us-east-2.console.aws.amazon.com/codesuite/
codecommit/repositories/MyDemoRepo/pull-requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecabEXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

pullRequestApprovalRuleUpdated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户编辑了对一个拉取请求的审批规则。她也是撰写拉取请求的用户。

```
{
    "version": "0",
    "id": "21b1c819-2889-3528-1cb8-3861aacf9d42",
    "detail-type": "CodeCommit Pull Request State Change",
    "source": "aws.codecommit",
    "account": "123456789012",
    "time": "2019-11-06T19:12:19Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
    ],
    "detail": {
        "approvalRuleContentSha256": "f742eebbEXAMPLE",
        "approvalRuleId": "0a9b5dfc-EXAMPLE",
        "approvalRuleName": "1-approver-needed",
        "author": "arn:aws:iam::123456789012:user/Mary_Major",
        "callerUserArn": "arn:aws:iam::123456789012:user/Mary_Major",
        "creationDate": "Wed Nov 06 19:10:58 UTC 2019",
        "description": "An example description.",
        "destinationCommit": "194fdf00EXAMPLE",
        "destinationReference": "refs/heads/main",
        "event": "pullRequestApprovalRuleUpdated",
        "isMerged": "False",
        "lastModifiedDate": "Wed Nov 06 19:10:58 UTC 2019",
```

```
        "notificationBody": "A pull request event occurred in the following
 AWS CodeCommit repository: MyDemoRepo. User: arn:aws:iam::123456789012:user/
Mary_Major. Event: Updated. Pull request name: 227. The content of an approval
 rule has been updated for the pull request. The name of the updated rule is: 1-
approver-needed. For more information, go to the AWS CodeCommit console https://
us-east-2.console.aws.amazon.com/codesuite/codecommit/repositories/MyDemoRepo/pull-
requests/227?region=us-east-2",
        "pullRequestId": "227",
        "pullRequestStatus": "Open",
        "repositoryNames": [
            "MyDemoRepo"
        ],
        "revisionId": "3b8cecab3EXAMPLE",
        "sourceCommit": "29964a17EXAMPLE",
        "sourceReference": "refs/heads/test-branch",
        "title": "My example pull request"
    }
}
```

reactionCreated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户对一个评论添加了表情符号反应。

```
{
   "version":"0",
   "id":"59fcccd8-217a-32ce-2b05-561ed68a1c42",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-14T00:49:03Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionCreated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 made a comment reaction # to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":["#"],
      "reactionShortcodes":[":thumbsdown:"],
```

```
      "reactionUnicodes":["U+1F44E"],
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

reactionUpdated 事件

在此示例事件中，IAM 用户名为 Mary_Major 的用户更新了对一个评论的表情符号反应。用户只能更新自己的表情符号反应。

```
{
   "version":"0",
   "id":"0844ed99-a53f-3bdb-6048-4de315516889",
   "detail-type":"CodeCommit Comment Reaction Change",
   "source":"aws.codecommit",
   "account":"123456789012",
   "time":"2020-04-22T23:19:42Z",
   "region":"us-east-2",
   "resources":[
      "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo"
   ],
   "detail":{
      "callerUserArn":"arn:aws:iam::123456789012:user/Mary_Major",
      "commentId":"28930161-EXAMPLE",
      "event":"commentReactionUpdated",
      "notificationBody":"A comment reaction event occurred in the following AWS
 CodeCommit Repository: MyDemoRepo. The user: arn:aws:iam::123456789012:user/Mary_Major
 updated a reaction :smile: to the comment with comment ID: 28930161-EXAMPLE",
      "reactionEmojis":[
         "#"
      ],
      "reactionShortcodes":[
         ":smile:"
      ],
      "reactionUnicodes":[
         "U+1F604"
      ],
      "repositoryId":"12345678-1234-5678-abcd-12345678abcd",
      "repositoryName":"MyDemoRepo"
   }
}
```

使用 AWS CodeCommit 记录 AWS CloudTrail API 调用

CodeCommit 与 AWS CloudTrail 服务集成，后者提供某个用户、角色或 AWS 服务在 CodeCommit 中执行的操作的记录。CloudTrail 将对 CodeCommit 的所有 API 调用作为事件捕获，包括来自 CodeCommit 控制台的调用、来自您的 Git 客户端的调用和对 CodeCommit API 的代码调用。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 桶（包括 CodeCommit 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 CodeCommit 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

CloudTrail 中的 CcodCommit 信息

在您创建 Amazon Web Services 账户时，将在该账户上启用 CloudTrail。当 CodeCommit 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在事件历史记录中。您可以在 Amazon Web Services 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 Amazon Web Services 账户中的事件（包括 CodeCommit 的事件），请创建跟踪记录。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

在您的 Amazon Web Services 账户中启用 CloudTrail 日志记录后，系统将在 CloudTrail 日志文件中跟踪对 CodeCommit 操作发出的 API 调用，这些调用随其他 AWS 服务记录一起写入到这些文件中。CloudTrail 基于时间段和文件大小来确定何时创建并写入新文件。

CloudTrail 记录所有 CodeCommit 操作，包括 AWS CodeCommit API 参考中当前尚未记录但在 CodeCommit 权限参考中引述为访问权限的某些操作(例如 `GetObjectIdentifier`)。例如，对在 CloudTrail 日志文件中生成条目的 `ListRepositories`（在 AWS CLI 中为 `aws codecommit list-repositories`）、`CreateRepository`(`aws codecommit create-repository`) 和

`PutRepositoryTriggers`（`aws codecommit put-repository-triggers`）操作的调用，以及对 `GitPull` 和 `GitPush` 的 Git 客户端调用。此外，如果您有配置为 CodePipeline 中管道源的 CodeCommit 存储库，则还将看到对 CodeCommit 访问权限操作（例如从 CodePipeline 进行的 `UploadArchive`）的调用。由于 CodeCommit 使用 AWS Key Management Service 加密和解密存储库，您还将从 CloudTrail 日志中的 AWS KMS 看到从 CodeCommit 对 `Encrypt` 和 `Decrypt` 操作的调用。

每个日志条目都包含有关生成请求的人员的信息。日志条目中的用户身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的
- 请求是使用角色还是联合身份用户的临时安全凭证发出的，还是由代入的角色发出的
- 请求是否由其它 AWS 服务发出

有关更多信息，请参阅 CloudTrail userIdentity 元素。

日志文件可以在 Amazon S3 存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。默认情况下，系统将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

了解 CodeCommit 日志文件条目

CloudTrail 日志文件可以包含一个或多个日志条目。每个条目列出了多个 JSON 格式的事件。一个日志事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。日志条目不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

> **Note**
>
> 为提高可读性，已对该示例设置格式。在 CloudTrail 日志文件，所有条目和事件都连接成一行。此外，该示例限于单个 CodeCommit 条目。在实际的 CloudTrail 日志文件中，有来自多个 AWS 服务的条目和事件。

目录

示例：用于列出 CodeCommit 存储库的日志条目

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `ListRepositories` 操作。

> **Note**
>
> 虽然 `ListRepositories` 返回一个存储库列表，但 CloudTrail 日志不记录不可变的响应，因此在日志文件中，`responseElements` 显示为 `null`。

```
{
  "eventVersion":"1.05",
  "userIdentity": {
    "type":"IAMUser",
    "principalId":"AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::444455556666:user/Mary_Major",
    "accountId":"444455556666",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime":"2016-12-14T17:57:36Z",
  "eventSource":"codecommit.amazonaws.com",
  "eventName":"ListRepositories",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.12",
  "userAgent":"aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters":null,
  "responseElements":null,
  "requestID":"cb8c167e-EXAMPLE",
  "eventID":"e3c6f4ce-EXAMPLE",
  "readOnly":true,
  "eventType":"AwsApiCall",
  "apiVersion":"2015-04-13",
  "recipientAccountId":"444455556666"
}
```

示例：用于创建 CodeCommit 存储库的日志条目

下面的示例显示了一个 CloudTrail 日志条目，该条目记录了在美国东部（俄亥俄州）区域的 `CreateRepository` 操作。

```
{
```

```
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
  },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "CreateRepository",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "aws-cli/1.10.53 Python/2.7.9 Windows/8 botocore/1.4.43",
  "requestParameters": {
    "repositoryDescription": "Creating a demonstration repository.",
    "repositoryName": "MyDemoRepo"
  },
  "responseElements": {
    "repositoryMetadata": {
      "arn": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "creationDate": "Dec 14, 2016 6:19:14 PM",
      "repositoryId": "8afe792d-EXAMPLE",
      "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryName": "MyDemoRepo",
      "accountId": "111122223333",
      "cloneUrlHttp": "https://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
      "repositoryDescription": "Creating a demonstration repository.",
      "lastModifiedDate": "Dec 14, 2016 6:19:14 PM"
    }
  },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
    }
  ],
```

```
  "eventType": "AwsApiCall",
  "apiVersion": "2015-04-13",
  "recipientAccountId": "111122223333"
}
```

示例：有关对 CodeCommit 存储库的 Git 拉取调用的日志条目

下面的示例显示了一个 CloudTrail 日志条目，该条目记录了在本地存储库处于最新状态时执行的 `GitPull` 操作。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.11.0.windows.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
```

```
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

下面的示例显示了一个 CloudTrail 日志条目，该条目记录了在本地存储库不是最新状态（因此有数据从 CodeCommit 存储库传输到本地存储库）时执行的 `GitPull` 操作。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPull",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "multi_ack_detailed",
      "side-band-64k",
      "thin-pack"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    "shallow": false
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": true,
  "resources": [
    {
```

```
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

示例：有关对 CodeCommit 存储库的成功推送的日志条目

下面的示例显示了一个 CloudTrail 日志条目，该条目记录了一个成功的 GitPush 操作。对于一次成功推送，日志条目中会显示两个 GitPush 操作。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "dataTransferred": false,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
```

```
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/Mary_Major",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName":"Mary_Major"
    },
  "eventTime": "2016-12-14T18:19:15Z",
  "eventSource": "codecommit.amazonaws.com",
  "eventName": "GitPush",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "git/2.10.1",
  "requestParameters": {
    "references": [
      {
        "commit": "100644EXAMPLE",
        "ref": "refs/heads/main"
        }
      ]
    },
  "responseElements": null,
  "additionalEventData": {
    "protocol": "HTTP",
    "capabilities": [
      "report-status",
      "side-band-64k"
      ],
    "dataTransferred": true,
    "repositoryName": "MyDemoRepo",
    "repositoryId": "8afe792d-EXAMPLE",
    },
  "requestID": "d148de46-EXAMPLE",
```

```
  "eventID": "740f179d-EXAMPLE",
  "readOnly": false,
  "resources": [
    {
      "ARN": "arn:aws:codecommit:us-east-2:111122223333:MyDemoRepo",
      "accountId": "111122223333",
      "type": "AWS::CodeCommit::Repository"
      }
    ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

使用 AWS CloudFormation 创建 CodeCommit 资源

AWS CodeCommit 与 AWS CloudFormation 集成，后者是一项服务，可帮助您对 AWS 资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个描述所需的全部 AWS 资源的模板（例如存储库），AWS CloudFormation 将为您预置和配置这些资源。

使用 AWS CloudFormation 时，您可以重复使用您的模板来不断地重复设置您的 CodeCommit 资源。描述您的资源一次，然后在多个 AWS 账户 和区域中反复预置相同的资源。

CodeCommit 和 AWS CloudFormation 模板

要为 CodeCommit 和相关服务预置和配置资源，您必须了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 AWS CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅 AWS CloudFormation 用户指南中的什么是 AWS CloudFormation Designer？。

CodeCommit 支持在 AWS CloudFormation 中创建存储库。与通过控制台或命令行创建存储库不同，您可以使用 AWS CloudFormation 创建存储库并自动将代码从 Amazon S3 桶中的特定 .zip 文件提交到新创建的存储库。有关更多信息（包括存储库的 JSON 和 YAML 模板示例），请参阅 AWS::CodeCommit::Repository。

使用 AWS CloudFormation 创建 CodeCommit 存储库时，只要存档小于 20MB，您就可以选择在 AWS:CodeCommit::Repository 代码中配置属性，从而在创建过程中将代码提交到该存储库。您可以指定存储代码的 Amazon S3 桶，也可以选择使用 BranchName 属性指定默认分支的名称，该名称将在代码的初始提交中创建。这些属性仅用于初始存储库创建，在堆栈更新时会被忽略。您无法使用这些属性对存储库进行其他提交，也无法在初始提交完成后更改默认分支的名称。

> **Note**
>
> AWS 在 2021 年 1 月 19 日将 CodeCommit 中默认分支的名称从 master 改为了 main。在使用 CodeCommit 控制台、CodeCommit API、AWS SDK 和 AWS CLI 为存储库创建初始提交时，这一名称变更将影响 CodeCommit 的默认行为。这一变更适用于使用 AWS CloudFormation 或 AWS CDK 创建且在创建过程中完成代码初始提交的存储库，并于 2021 年 3 月 4 日开始生效。这一变更不会影响现有的存储库或分支。使用本地 Git 客户端创建初始提交的客户有一个默认分支名称，该名称遵循这些 Git 客户端的配置。有关更多信息，请参阅使用分支、创建提交和更改分支设置。

您还可以创建用于创建相关资源的模板，例如存储库通知规则、AWS CodeBuild 构建项目、AWS CodeDeploy 应用程序和 AWS CodePipeline 管道。

模板示例

以下示例创建一个名为 *MyDemoRepo* 的 CodeCommit 存储库。新创建的存储库使用存储在名为 *MySourceCodeBucket* 的 Amazon S3 存储桶中的代码填充，并放置在名为 *development* 的分支中，该分支是此存储库的默认分支。

> **Note**
>
> 对于包含内容将提交到新存储库的 ZIP 文件的 Amazon S3 桶，您可以使用 ARN 或 Amazon Web Services 账户中该桶的名称来指定该桶的名称。有关 Amazon S3 对象键的定义，请参阅 Amazon S3 开发人员指南。

JSON:

```
{
    "MyRepo": {
        "Type": "AWS::CodeCommit::Repository",
        "Properties": {
            "RepositoryName": "MyDemoRepo",
            "RepositoryDescription": "This is a repository for my project with code
 from MySourceCodeBucket.",
            "Code": {
                "BranchName": "development",
                "S3": {
                    "Bucket": "MySourceCodeBucket",
                    "Key": "MyKey",
                    "ObjectVersion": "1"
                }
            }
        }
    }
}
```

YAML：

```
MyRepo:
```

```
  Type: AWS::CodeCommit::Repository
  Properties:
    RepositoryName: MyDemoRepo
    RepositoryDescription: This is a repository for my project with code from
 MySourceCodeBucket.
    Code:
      BranchName: development
      S3:
        Bucket: MySourceCodeBucket,
        Key: MyKey,
        ObjectVersion: 1
```

有关更多示例，请参阅 AWS::CodeCommit::Repository。

AWS CloudFormation、CodeCommit 和 AWS Cloud Development Kit (AWS CDK)

使用 AWS CDK 创建的存储库在创建时使用 AWS CloudFormation 功能。了解 AWS CloudFormation 模板如何用于 CodeCommit 资源，可以帮助您创建和管理 AWS CDK 代码。有关 AWS CDK 的更多信息，请参阅 AWS Cloud Development Kit (AWS CDK) 开发人员指南和 AWS CDK API 参考。

以下 AWS CDK TypeScript 示例创建一个名为 *MyDemoRepo* 的 CodeCommit 存储库。新创建的存储库使用存储在名为 *MySourceCodeBucket* 的 Amazon S3 存储桶中的代码填充，并放置在名为 *development* 的分支中，该分支是此存储库的默认分支。

```
import * as cdk from '@aws-cdk/core';
import codecommit = require('@aws-cdk/aws-codecommit');
export class CdkCodecommitStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);
    // The code creates a CodeCommit repository with a default branch name development
      new codecommit.CfnRepository(this, 'MyRepoResource', {
            repositoryName: "MyDemoRepo",
            code: {
              "branchName": "development",
              "s3": {
                "bucket": "MySourceCodeBucket",
                "key": "MyKey"
              }
            },
        }
```

```
      );
  }
}
```

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation 命令行界面用户指南

故障排除 AWS CodeCommit

以下信息可帮助您处理 AWS CodeCommit 中的常见问题。

主题

Git 凭证和 HTTPS 与 AWS CodeCommit 的连接问题排查

以下信息可帮助您排查在使用 Git 凭证和 HTTPS 连接到 AWS CodeCommit 存储库时的常见问题。

主题

AWS CodeCommit 的 Git 凭证：在终端或命令行中连接到我的 CodeCommit 存储库时，系统总是提示我输入凭证

问题：尝试在终端或命令行中针对 CodeCommit 存储库执行推送、拉取操作或以其他方式与之交互时，系统提示您提供用户名和密码，而您必须提供您的 IAM 用户的 Git 凭证。

可能的修复措施：出现此错误的最常见原因是，您的本地计算机运行的操作系统不支持凭证管理、没有安装凭证管理实用程序或者您尚未将 IAM 用户的 Git 凭证保存到某个凭证管理系统中。根据您的操作系统和本地环境，您可能需要安装凭证管理器、配置操作系统随附的凭证管理器或自定义您的本地

环境以使用凭证存储。例如，如果您的计算机运行的是 macOS，您可以使用 Keychain Access 实用程序存储您的凭证。如果您的计算机运行的是 Windows，您可以使用随 Windows 版 Git 安装的 Git Credential Manager。有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户和 Git 文档中的凭证存储。

AWS CodeCommit 的 Git 凭证：我设置了 Git 凭证，但系统未使用这些凭证

问题：尝试通过 Git 客户端使用 CodeCommit 时，该客户端未使用您的 IAM 用户的 Git 凭证。

可能的修复措施：造成此错误的最常见原因是您之前将计算机设置为使用 AWS CLI 附带的凭证辅助程序。检查 .gitconfig 文件的配置部分 (与以下内容类似)，并将其删除：

```
[credential "https://git-codecommit.*.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

保存文件，打开一个新的命令行或终端会话，再次尝试连接。

计算机中可能还设置了多个凭证辅助程序或管理器，系统也可能默认使用其他配置。要重设默认使用的凭证辅助程序，可以在运行 git config 命令时使用 --system 选项取代 --global 或 --local 选项。

有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户和 Git 文档中的凭证存储。

git-remote-codecommit 和 AWS CodeCommit 疑难解答

以下信息可帮助您排查在连接 AWS CodeCommit 存储库时遇到的与 git-remote-codecommit 有关的错误。

主题

- 我看到一个错误：git: 'remote-codecommit' is not a git command
- 我看到一个错误：fatal: Unable to find remote helper for 'codecommit'
- 克隆错误：我无法从 IDE 克隆 CodeCommit 存储库
- 推送或拉取错误：我无法将提交从 IDE 推送或拉取到 CodeCommit 存储库

我看到一个错误：git: 'remote-codecommit' is not a git command

问题：尝试使用 git-remote-codecommit 时，您看到一个错误，显示“git-remote-codecommit is not a git command. See 'git --help'”。

可能的修复措施：出现此错误的最常见原因是，您没有将 git-remote-codecommit 可执行文件添加到 PATH 中，或者字符串包含语法错误。当 git 和 remote-codecommit 之间缺少连字符，或者在 git-remote-codecommit 之前添加一个额外的 git 时，就可能会发生这种情况。

有关设置和使用 git-remote-codecommit 的更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

我看到一个错误：fatal: Unable to find remote helper for 'codecommit'

问题：尝试使用 git-remote-codecommit 时，您看到一个错误，显示“fatal: Unable to find remote helper for 'codecommit'”。

可能的修复措施：出现此错误的最常见原因包括：

- git-remote-codecommit 的设置未完成
- 安装 git-remote-codecommit 的位置不在您的路径中或者没有配置为 `Path` 环境变量的一部分
- Python 不在您的路径中或者没有配置为 `Path` 环境变量的一部分
- 使用的终端或命令行窗口自安装 git-remote-codecommit 后一直未重新启动

有关设置和使用 git-remote-codecommit 的更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

克隆错误：我无法从 IDE 克隆 CodeCommit 存储库

问题：尝试在 IDE 中克隆 CodeCommit 存储库时，您看到一个错误，指出端点或 URL 无效。

可能的修复：并非所有 IDE 都支持 git-remote-codecommit 在克隆过程中使用的 URL。从终端或命令行本地克隆存储库，然后将该本地存储库添加到 IDE 中。有关更多信息，请参阅步骤 3：连接 CodeCommit 控制台并克隆存储库。

推送或拉取错误：我无法将提交从 IDE 推送或拉取到 CodeCommit 存储库

问题：当您尝试从 IDE 中提取或推送代码时，会看到连接错误。

可能的修复：此错误的最常见原因是 IDE 与 Git 远程辅助程序（如 git-remote-codecommit）不兼容。使用 Git 命令从命令行或终端手动更新本地存储库，而不是使用 IDE 功能提交、推送和拉取代码。

有关远程辅助程序和 Git 的更多信息，请参阅 Git 文档。

SSH 与 AWS CodeCommit 的连接问题排查

以下信息可帮助您排查在使用 SSH 连接到 CodeCommit 存储库时的常见问题。

主题

访问错误：已将公有密钥成功上传到 IAM，但在 Linux、macOS 或 Unix 系统上进行连接时失败

问题：尝试连接到 SSH 端点以与 CodeCommit 存储库通信时，测试连接或克隆存储库的操作出现连接失败或连接被拒绝。

可能的修复措施：分配给 IAM 中公有密钥的 SSH 密钥 ID 可能未关联到您尝试的连接。您可能尚未配置配置文件、您可能无权访问配置文件、其他设置可能导致系统无法成功读取配置文件、您可能提供了错误的密钥 ID 或者您可能提供了 IAM 用户 ID 而不是密钥 ID。

您可以通过 IAM 控制台在 IAM 用户的配置文件中找到 SSH 密钥 ID：

> **Note**
>
> 如果您上传了多个 SSH 密钥 ID，则按密钥 ID 的字母顺序（而不是按上传日期）列出密钥。请确保已复制与正确上传日期关联的密钥 ID。

尝试使用下面的命令测试连接：

```
ssh Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com
```

如果在确认连接后看到成功消息，说明您的 SSH 密钥 ID 是有效的。编辑配置文件，将您尝试的连接关联到 IAM 中的公有密钥。如果您不想编辑配置文件，可以在所有存储库连接前面加上 SSH 密钥 ID。例如，如果需要克隆名为 *MyDemoRepo* 的存储库，但不想修改与连接关联的配置文件，可以键入下面的命令：

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo my-demo-repo
```

有关更多信息，请参阅适用于 Linux、macOS 或 Unix 上的 SSH 连接。

访问错误：已将公有密钥成功上传到 IAM 并且 SSH 测试成功，但在 Windows 系统上进行连接时失败

问题：尝试使用 SSH 端点克隆 CodeCommit 存储库或与之通信时，出现一条错误消息，其中包含 `No supported authentication methods available` 字样。

可能的修复措施：导致该错误的最常见原因是您设置的某个 Windows 系统环境变量指示 Windows 在您尝试使用 SSH 时使用其他的程序。例如，您可能设置了 GIT_SSH 变量，将其指向了某个 PuTTY 工具集 (plink.exe)。这可能是一个旧式配置，或计算机上安装的一个或多个其他程序需要该配置。如果确定不需要该环境变量，可以打开系统属性来将其删除。

要解决这个问题，请打开 Bash 仿真器，再次尝试 SSH 连接，但包含 `GIT_SSH_COMMAND="SSH"` 作为前缀。例如，要使用 SSH 克隆存储库：

```
GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

如果您的 Windows 版本要求在 Windows 命令行中通过 SSH 进行连接时必须将 SSH 密钥 ID 包含在连接字符串中，也可能会出现类似问题。尝试再次连接，这次请在命令中包含从 IAM 复制的 SSH 密钥 ID。例如：

```
git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo my-demo-repo
```

身份验证质询：连接到 CodeCommit 存储库时，无法确认主机的真实性

问题：尝试使用 SSH 端点与 CodeCommit 存储库通信时，出现一条警告消息，其中包含 `The authenticity of host 'host-name' can't be established.` 字样。

可能的修复措施：您的凭证设置可能不正确。按照适用于 Linux、macOS 或 Unix 上的 SSH 连接 或适用于 Windows 上的 SSH 连接 中的说明操作。

如果按照上述步骤操作后，问题仍然存在，可能是有人正在尝试实施中间人攻击。如果看到下面的消息，请键入 no 并按 Enter。

```
Are you sure you want to continue connecting (yes/no)?
```

确保 CodeCommit 连接的指纹和公有密钥与 SSH 设置主题中记录的指纹和公有密钥匹配，再继续连接。

CodeCommit 的公有指纹

服务器	加密哈希类型	指纹
git-codecommit.us-east-2.amazonaws.com	MD5	a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e
git-codecommit.us-east-2.amazonaws.com	SHA256	3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ
git-codecommit.us-east-1.amazonaws.com	MD5	a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84

服务器	加密哈希类型	指纹
git-codecommit.us-east-1.amazonaws.com	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git-codecommit.ap-southeast-1.amazonaws.com	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno

服务器	加密哈希类型	指纹
git-codecommit.ap-southeast-2.amazonaws.com	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec
git-codecommit.ap-southeast-3.amazonaws.com	MD5	64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g
git-codecommit.me-central-1.amazonaws.com	MD5	bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76
git-codecommit.me-central-1.amazonaws.com	SHA256	grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA
git-codecommit.eu-central-1.amazonaws.com	MD5	74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d
git-codecommit.eu-central-1.amazonaws.com	SHA256	MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY
git-codecommit.ap-northeast-2.amazonaws.com	MD5	9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed

服务器	加密哈希类型	指纹
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko
git-codecommit.sa-east-1.amazonaws.com	MD5	74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51
git-codecommit.sa-east-1.amazonaws.com	SHA256	kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU
git-codecommit.us-west-1.amazonaws.com	MD5	3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1
git-codecommit.us-west-1.amazonaws.com	SHA256	gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg
git-codecommit.eu-west-2.amazonaws.com	MD5	a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3
git-codecommit.eu-west-2.amazonaws.com	SHA256	r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs
git-codecommit.ap-south-1.amazonaws.com	MD5	da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21
git-codecommit.ap-south-1.amazonaws.com	SHA256	hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE

服务器	加密哈希类型	指纹
git-codecommit.ap-south-2.amazonaws.com	MD5	bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce
git-codecommit.ap-south-2.amazonaws.com	SHA256	Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig
git-codecommit.ca-central-1.amazonaws.com	MD5	9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23
git-codecommit.ca-central-1.amazonaws.com	SHA256	Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w
git-codecommit.eu-west-3.amazonaws.com	MD5	1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76
git-codecommit.eu-west-3.amazonaws.com	SHA256	uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI
git-codecommit.us-gov-west-1.amazonaws.com	MD5	9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69
git-codecommit.us-gov-west-1.amazonaws.com	SHA256	djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4
git-codecommit.us-gov-east-1.amazonaws.com	MD5	00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2

服务器	加密哈希类型	指纹
git-codecommit.us-gov-east-1.amazonaws.com	SHA256	fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98
git-codecommit.eu-north-1.amazonaws.com	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
git-codecommit.eu-north-1.amazonaws.com	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git-codecommit.me-south-1.amazonaws.com	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git-codecommit.me-south-1.amazonaws.com	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git-codecommit.ap-east-1.amazonaws.com	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git-codecommit.ap-east-1.amazonaws.com	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git-codecommit.cn-north-1.amazonaws.com.cn	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git-codecommit.cn-north-1.amazonaws.com.cn	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8

服务器	加密哈希类型	指纹
git-codecommit.cn-northwest-1.amazonaws.com.cn	MD5	`2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6`
git-codecommit.cn-northwest-1.amazonaws.com.cn	SHA256	`wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ`
git-codecommit.eu-south-1.amazonaws.com	MD5	`b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11`
git-codecommit.eu-south-1.amazonaws.com	SHA256	`lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI`
git-codecommit.ap-northeast-3.amazonaws.com	MD5	`25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29`
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	`2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA`
git-codecommit.af-south-1.amazonaws.com	MD5	`21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29`
git-codecommit.af-south-1.amazonaws.com	SHA256	`C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM`
git-codecommit.il-central-1.amazonaws.com	MD5	`04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9`

服务器	加密哈希类型	指纹
git-codecommit.il-central-1.amazonaws.com	SHA256	uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8

IAM 错误：尝试向 IAM 添加公有密钥时，出现“格式无效”错误

问题：在 IAM 中，设置对 CodeCommit 使用 SSH 连接，在尝试添加公有密钥时，出现一条错误消息，其中包含 `Invalid format` 字样。

可能的修复措施：IAM 要求公有密钥必须采用 ssh-rsa 格式或 PEM 格式进行编码。它仅接受 OpenSSH 格式的公有密钥，并且具有《IAM 用户指南》中的将 SSH 密钥与 CodeCommit 配合使用所述的其他要求。如果提供其他格式的公有密钥，或者密钥包含的位数不够，就会出现此错误。

- 当您复制 SSH 公钥时，您的操作系统可能引入了换行符。确保添加到 IAM 的公有密钥中没有换行符。
- 某些 Windows 操作系统不使用 OpenSSH 格式。要生成密钥对并复制 IAM 要求的 OpenSSH 格式，请参阅the section called “步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥”。

有关 IAM 中的 SSH 密钥要求的更多信息，请参阅《IAM 用户指南》中的将 SSH 密钥与 CodeCommit 配合使用。

我需要使用 SSH 凭证访问多个 Amazon Web Services 账户中的 CodeCommit 存储库

问题：我想在多个 Amazon Web Services 账户中设置对 CodeCommit 存储库的 SSH 访问权限。

可能的修复措施：您可以为每个 Amazon Web Services 账户创建唯一的 SSH 公有/私有密钥对，并使用每个密钥配置 IAM。然后，您可以使用与公有密钥关联的每个 IAM 用户 ID 的相关信息来配置您的 ~/.ssh/config 文件。例如：

```
Host codecommit-1
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-1 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 1 (for example, APKAEIBAERJR2EXAMPLE1).
    IdentityFile ~/.ssh/codecommit_rsa # This is the path to the associated public key
 file, such as id_rsa.  We advise creating CodeCommit specific _rsa files.
```

```
Host codecommit-2
    Hostname git-codecommit.us-east-1.amazonaws.com
    User SSH-KEY-ID-2 # This is the SSH Key ID you copied from IAM in Amazon Web
 Services account 2 (for example, APKAEIBAERJR2EXAMPLE2).
    IdentityFile ~/.ssh/codecommit_2_rsa # This is the path to the other associated
 public key file.  We advise creating CodeCommit specific _rsa files.
```

在此配置中，您可以将“git-codecommit.us-east-1.amazonaws.com”替换为“codecommit-2”。例如，要克隆第二个 Amazon Web Services 账户中的存储库，请运行以下命令：

```
git clone ssh://codecommit-2/v1/repos/YourRepositoryName
```

要为存储库设置远程操作，请运行 git remote add。例如：

```
git remote add origin ssh://codecommit-2/v1/repos/YourRepositoryName
```

有关更多示例，请参阅这篇论坛帖子和 GitHub 上的这篇文章。

Windows 上的 Git：尝试使用 SSH 进行连接时，Bash 仿真器或命令行卡住

问题：为 Windows 配置 SSH 访问并在命令行或终端中确认连接后，当尝试在命令提示符或 Bash 仿真器中 git pull、git push 或 git clone 命令时，显示一条消息，指示服务器的主机密钥未缓存在注册表中，并且有关在缓存中存储密钥的提示卡住（不接受 y/n/return 输入）。

可能的修复措施：导致该错误的最常见原因是 Git 环境配置为使用 OpenSSH 以外的程序 (可能是 PuTTY) 进行身份验证。这会在某些配置中导致与缓存密钥有关的问题。要解决这个问题，请尝试以下操作之一：

- 打开 Bash 仿真器，在 Git 命令前添加 `GIT_SSH_COMMAND="ssh"` 参数。例如，在尝试推送到存储库时，不要键入 git push，而是键入：

  ```
  GIT_SSH_COMMAND="ssh" git push
  ```

- 如果已安装 PuTTY，请打开 PuTTY，在主机名（或 IP 地址）中，输入要访问的 CodeCommit 端点（例如 git-codecommit.us-east-2.amazonaws.com）。选择 Open（打开）。当出现 PuTTY 安全提醒的提示时，选择 Yes (是) 永久缓存密钥。
- 重命名或删除不再使用的 `GIT_SSH` 环境变量。然后，打开新的命令提示符或 Bash 仿真器会话，再次尝试您的命令。

有关其他解决方案的信息，请参阅 Stack Overflow 上的 Git clone/pull continually freezing at Store key in cache。

公有密钥格式在某些 Linux 发行版中需要规范化

问题：尝试配置公有-私有密钥对时，收到一个错误。

可能的修复措施：某些 Linux 发行版需要在 `~/.ssh/config` 文件中添加额外的配置行，以指定公有密钥的可接受类型。有关更多信息，请参阅有关 `PubkeyAcceptedKeyTypes` 的具体发行版文档。

访问错误：连接到 CodeCommit 存储库时，SSH 公有密钥被拒绝

问题：尝试使用 SSH 端点与 CodeCommit 存储库通信时，出现一条错误消息，其中包含 `Error: public key denied` 字样。

可能的修复措施：导致出现此错误的最常见原因是您尚未完成 SSH 连接设置。请配置公有和私有 SSH 密钥对，然后将公有密钥与您的 IAM 用户相关联。有关配置 SSH 的更多信息，请参阅适用于 Linux、macOS 或 Unix 上的 SSH 连接 和适用于 Windows 上的 SSH 连接。

凭证助手和 HTTPS 与 AWS CodeCommit 的连接问题排查

以下信息可帮助您排查在使用 AWS CLI 中包含的凭证助手和 HTTPS 连接到 CodeCommit 存储库时的常见问题。

> **Note**
>
> 虽然使用联合访问、身份提供者或临时凭证连接到 CodeCommit 时，凭证助手是一种受支持的方法，但建议方法是安装并使用 git-remote-codecommit 实用程序。有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。

主题

在运行 `git config` 命令来配置凭证助手时，收到错误

问题：尝试运行 git config 命令来配置凭证助手以与 CodeCommit 存储库通信时，显示一个错误，指示参数太少，或者显示一个用法提示，建议使用 Git config 命令和语法。

可能的修复措施：出现此错误的最常见原因是，在 Windows 操作系统中针对命令使用单引号，或者在 Linux、macOS 或 Unix 操作系统中针对命令使用双引号。正确的语法如下：

- Windows: `git config --global credential.helper "!aws codecommit credential-helper $@"`
- Linux、macOS 或 Unix：`git config --global credential.helper '!aws codecommit credential-helper $@'`

在 Windows 中使用凭证辅助程序时返回的“找不到命令”错误

问题：更新 AWS CLI 后，凭证助手与 CodeCommit 存储库的连接失败，并显示 `aws codecommit credential-helper $@ get: aws: command not found`。

原因：出现此错误的最常见原因是，您的 AWS CLI 版本已更新到使用 Python 3 的版本。MSI 程序包存在一个已知问题。要验证您是否有其中一个受影响的版本，请打开命令行并运行以下命令：`aws --version`

如果输出 Python 版本以 3 开头，则表示您拥有受影响的版本。例如：

```
aws-cli/1.16.62 Python/3.6.2 Darwin/16.7.0 botocore/1.12.52
```

可能的修复措施：您可以执行下列操作之一来解决此问题：

- 在 Windows 上使用 Python 和 pip 而不是 MSI 来安装和配置 AWS CLI。有关更多信息，请参阅在 Windows 上安装 Python、pip 和 AWS CLI。
- 手动编辑 `.gitconfig` 文件，更改 `[credential]` 部分以明确指向本地计算机上的 `aws.cmd`。例如：

```
[credential]
    helper = !"\C:\\Program Files\\Amazon\\AWSCLI\\bin\\aws.cmd\" codecommit
 credential-helper $@
    UseHttpPath = true
```

- 运行 git config 命令以更新 `.gitconfig` 文件来明确引用 `aws.cmd`，并手动更新 PATH 环境变量以便根据需要包含命令的路径。例如：

```
git config --global credential.helper "!aws.cmd codecommit credential-helper $@"
git config --global credential.UseHttpPath true
```

在连接到 CodeCommit 存储库时，系统提示我输入用户名

问题：尝试使用凭证助手与 CodeCommit 存储库通信时，显示一条消息，提示您输入您的用户名。

可能的修复措施：配置您的 AWS 配置文件或确保所使用的配置文件是为使用 CodeCommit 而配置的。有关设置的更多信息，请参阅使用 AWS CLI 凭证助手在 Linux、macOS 或 Unix 上对 AWS CodeCommit 存储库进行 HTTPS 连接的设置步骤或使用 AWS CLI 凭证助手在 Windows 上设置到 AWS CodeCommit 存储库的 HTTPS 连接的步骤。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅管理 IAM 用户的访问密钥和如何获取凭证？。

macOS 版 Git：我成功配置了凭证助手，但在访问我的存储库时被系统拒绝 (403)

问题：在 macOS 上，凭证助手似乎无法正常访问或使用您的凭证。这可能是由以下两种原因导致的：

- 将 AWS CLI 配置为与存储库所在区域不同的 AWS 区域。
- Keychain Access 实用程序保存的凭证已过期。

可能的修复措施：要验证为 AWS CLI 配置的区域是否正确，请运行aws configure 命令，并查看显示的信息。如果 CodeCommit 存储库所在的 AWS 区域与 AWS CLI 显示的 AWS 区域不同，则必须运行 aws configure 命令，将这些值更改为该区域相应的值。有关更多信息，请参阅步骤 1：CodeCommit 的初始配置。

在 OS X 和 macOS 上发布的 Git 默认版本使用 Keychain Access 实用程序来保存生成的凭证。为安全起见，为访问您的 CodeCommit 存储库生成的密码是临时的，因此密钥链中存储的凭证将在约 15 分钟后失效。如果仅在使用 CodeCommit 时访问 Git，请尝试以下操作：

1. 在终端上，运行 git config 命令查找在其中定义 Keychain Access 实用程序的 Git 配置文件 (`gitconfig`)。根据本地系统和首选项的不同，您可能有多个 `gitconfig` 文件。

```
git config -l --show-origin | grep credential
```

在此命令的输出中，搜索类似于以下内容的结果：

```
file:/path/to/gitconfig  credential.helper=osxkeychain
```

该行开头列出的文件就是您必须编辑的 Git 配置文件。

2. 要编辑 Git 配置文件，请使用纯文本编辑器或运行以下命令：

```
nano /usr/local/git/etc/gitconfig
```

3. 使用以下任一策略修改配置：

- 注释掉或删除包含 `helper = osxkeychain` 的凭证部分。例如：

```
# helper = osxkeychain
```

- 更新 `aws credential helper` 和 `osxkeychain` 凭证助手部分以包含上下文。例如，如果 `osxkeychain` 用于向 GitHub 进行身份验证：

```
[credential "https://git-codecommit.us-east-1.amazonaws\.com"]
  helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
  UseHttpPath = true
[credential "https://github.com"]
  helper = osxkeychain
```

在此配置中，当远程主机匹配“`https://github.com`”时，Git 将使用 `osxkeychain` 助手，当远程主机匹配“`https://git-codecommit\.us-east-1\.amazonaws.com`”时，Git 将使用凭证助手。

- 在凭证助手之前添加一个空字符串助手。例如：

```
[credential]
  helper =
  helper = !aws --profile CodeCommitProfile codecommit credential-helper $@
  UseHttpPath = true
```

或者，如果您希望继续使用 Keychain Access 实用程序来缓存其他 Git 存储库的凭证，请修改标头，而不要注释掉该行。例如，要允许使用缓存的 GitHub 凭证，您可以按如下所示修改标头：

```
[credential "https://github.com"]
   helper = osxkeychain
```

如果您使用 Git 访问其他存储库，则可以将 Keychain Access 实用程序配置为不为您的 CodeCommit 存储库提供凭证。配置 Keychain Access 实用程序：

1. 打开 Keychain Access 实用程序。(您可以使用 Finder 查找它。)
2. 搜索 `git-codecommit.us-east-2.amazonaws.com` 并将 *us-east-2* 替换为存储库所在的 AWS 区域。突出显示该行，打开上下文菜单 (右键单击)，然后选择 Get Info。
3. 选择 Access Control 选项卡。
4. 在 Confirm before allowing access (运行访问前进行确认) 中，选择 `git-credential-osxkeychain`，然后选择减号将它从列表中删除。

> **Note**
>
> 从列表中删除 `git-credential-osxkeychain` 后，您在每次运行 Git 命令时都会看到一个对话框。选择拒绝以继续。如果不希望显示弹出对话框，可考虑下面几种替代选项：
>
> - 使用 SSH 或 Git 凭证而不是使用凭证助手和 HTTPS 连接到 CodeCommit。有关更多信息，请参阅 适用于 Linux、macOS 或 Unix 上的 SSH 连接 和 适用于使用 Git 凭证的 HTTPS 用户的设置：
> - 在 Keychain Access 实用程序中，在 `git-codecommit.us-east-2.amazonaws.com` 的访问控制选项卡上，选择允许所有应用程序访问此项目（对此项目的访问不受限制）选项。这将阻止弹出窗口，但凭证终究会到期（平均而言，大约需要 15 分钟），然后会出现 403 错误消息。如果发生这种情况，您必须删除密钥链项才能恢复功能。
> - 安装默认不使用密钥链的 Git 版本。
> - 考虑制定一个删除密钥链项的脚本解决方案。要查看社区生成的脚本解决方案示例，请参阅中的 定期删除 OS X Certificate Store 中缓存凭证的 Mac OS X 脚本产品和服务集成。

如果要完全阻止 Git 使用 Keychain Access 实用工具，可以配置 Git 以停止使用 osxkeychain 作为凭证辅助程序。例如，如果您打开一个终端并运行命令 `git config --system`

`credential.helper`，并且它返回 `osxkeychain`，则 Git 设置为使用 Keychain Access 实用程序。您可以使用以下命令更改此设置：

```
git config --system --unset credential.helper
```

请注意，通过使用 `--system` 选项运行此命令将在系统范围内更改所有用户的 Git 行为，这可能会对其他用户或其他存储库产生意外的后果（如果您使用的是 CodeCommit 之外的其他存储库服务）。还要注意，此方法可能需要使用 `sudo`，并且您的账户可能没有足够的系统权限来应用此更改。确保通过重新运行 `git config --system credential.helper` 命令来验证该命令是否已成功应用。有关更多信息，请参阅自定义 Git – Git 配置和 Stack Overflow 上的此文章。

Windows 版 Git：我安装了 Windows 版 Git，但在访问我的存储库时被系统拒绝 (403)

问题：在 Windows 上，凭证辅助程序似乎无法正常访问或使用您的凭证。这可能是由多种原因导致的：

- 将 AWS CLI 配置为与存储库所在区域不同的 AWS 区域。
- 默认情况下，Windows 版 Git 会安装与 CodeCommit 连接不兼容的 Git Credential Manager 实用程序，CodeCommit 连接使用 AWS 凭证助手。安装后，即使凭证助手已随 AWS CLI 安装并针对到 CodeCommit 的连接进行了配置，仍会导致与存储库的连接失败。
- 某些版本的 Windows 版 Git 不完全符合 RFC 2617 和 RFC 4559 标准，这可能会导致 Git 凭证和 AWS CLI 随附的凭证辅助程序出现问题。有关更多信息，请参阅 Version 2.11.0(3) does not ask for username/password。

可能的修复措施：

- 如果尝试使用 AWS CLI 随附的凭证辅助程序，请考虑经由 HTTPS 使用 Git 凭证进行连接，而不要使用凭证辅助程序。与 AWS CodeCommit 凭证助手不同，为 IAM 用户配置的 Git 凭证与 Windows 版 Git Credential Manager 兼容。有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户。

  如果需要使用凭证助手，请运行 aws configure 命令并查看显示的信息，验证为 AWS CLI 配置的 AWS 区域是否正确。如果 CodeCommit 存储库所在的 AWS 区域与 AWS CLI 显示的 AWS 区域不同，则必须运行 aws configure 命令，将这些值更改为该区域相应的值。有关更多信息，请参阅步骤 1：CodeCommit 的初始配置。
- 如果可能，请卸载并重新安装 Windows 版 Git。在安装 Windows 版 Git 时，清除安装 Git Credential Manager 实用程序选项的复选框。该凭证管理器与 AWS CodeCommit 的凭证辅助程序不兼

容。如果安装了 Git Credential Manager 或其他凭证管理实用程序，但不想卸载它，可以修改 `.gitconfig` 文件并为 CodeCommit 添加特定的凭证管理：

1. 打开控制面板，选择凭证管理器，并删除为 CodeCommit 存储的任何凭证。
2. 用任意纯文本编辑器（如“记事本”）打开您的 `.gitconfig` 文件。

   > **Note**
   >
   > 如果使用多个 Git 配置文件，则可能同时存在本地和全局 `.gitconfig` 文件。请务必编辑相应的文件。

3. 向您的 `.gitconfig` 文件添加以下部分：

   ```
   [credential "https://git-codecommit.*.amazonaws.com"]
       helper = !aws codecommit credential-helper $@
       UseHttpPath = true
   ```

4. 保存文件，打开一个新的命令行会话，再次尝试连接。

如果需要在连接到 CodeCommit 存储库时使用 AWS CodeCommit 凭证助手，而在连接到其他托管存储库（例如 GitHub 存储库）时使用其他凭证管理系统，也可以使用这种方法。

要重置默认使用的凭证辅助程序，可以在运行 git config 命令时使用 --system 选项取代 --global 或 --local。

- 在 Windows 计算机上使用 Git 凭证时，可以通过在连接字符串中包含 Git 凭证用户名来解决任何 RFC 不合规问题。例如，要解决这个问题并克隆美国东部（俄亥俄州）区域中名为 *MyDemoRepo* 的存储库：

  ```
  git clone https://Your-Git-Credential-Username@git-codecommit.us-
  east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
  ```

  > **Note**
  >
  > 如果您的 Git 凭证用户名中含有 @ 字符，则这种方法无效。您必须对该字符进行 URL 编码（也称作 URL 转义或百分号编码）。

Git 客户端和 AWS CodeCommit 问题排查

以下信息可帮助您排查对 AWS CodeCommit 存储库使用 Git 时的常见问题。有关排查在使用 HTTPS 或 SSH 时出现的与 Git 客户端相关的问题，另请参阅Git 凭证 (HTTPS) 问题排查、SSH 连接问题排查和凭证助手 (HTTPS) 问题排查。

主题

Git 错误：Error: RPC failed; result=56, HTTP code = 200 fatal: The remote end hung up unexpectedly

问题：在推送较大的更改、大量更改或大型存储库时，长时间运行的 HTTPS 连接通常会因为网络问题或防火墙设置而提前终止。

可能的修复措施：改用 SSH 推送，或在迁移大型存储库时按照以增量方式迁移存储库中的步骤操作。还要确保未超出单个文件的大小限制。有关更多信息，请参阅配额。

Git 错误：引用更新命令过多

问题：每次推送的引用更新数量最多为 4000 个。当推送包含超过 4000 个引用更新时，会出现这种错误。

可能的修复措施：尝试使用 `git push --all` 和 `git push --tags` 分别推送分支和标签。如果标签过多，将标签拆分成多个推送批次。有关更多信息，请参阅配额。

Git 错误：在某些版本的 Git 中，无法通过 HTTPS 执行推送

问题：更新到 7.41.0 的 curl 存在导致基于 SSPI 的摘要身份验证失败的问题。已知受影响的 Git 版本包括 1.9.5.msysgit.1。某些版本的 Windows 版 Git 可能不完全符合 RFC 2617 和 RFC 4559 标准，这可能会导致使用 Git 凭证或 AWS CLI 随附的凭证辅助程序的 HTTPS 连接出现问题。

可能的修复措施：检查 Git 版本是否存在已知问题，或使用更早/更高的版本。有关 mysysgit 的更多信息，请参阅 Github 论坛中的 Push to HTTPS Is Broken。有关 Windows 版 Git 的版本问题的更多信息，请参阅版本 2.11.0(3) 不需要用户名/密码。

Git 错误：“gnutls_handshake() failed”

问题：在 Linux 中，尝试使用 Git 与 CodeCommit 存储库通信时，出现一条错误消息，其中包含 `error: gnutls_handshake() failed` 字样。

可能的修复措施：基于 OpenSSL 编译 Git。Ask Ubuntu 论坛提供了一个解决方案，请参阅 "Error: gnutls_handshake() failed" When Connecting to HTTPS Servers。

或者，使用 SSH 而不是 HTTPS 与 CodeCommit 存储库通信。

Git 错误：Git 找不到 CodeCommit 存储库或无权访问该存储库

问题：连接字符串中的尾部斜杠可能导致连接尝试失败。

可能的修复措施：确保提供正确的存储库名称和连接字符串，并且没有尾随斜杠。有关更多信息，请参阅连接存储库。

Windows 上的 Git：没有支持的身份验证方法可用 (publickey)

问题：为 Windows 配置 SSH 访问后，在尝试使用 git pull、git push 或 git clone 命令时出现拒绝访问错误。

可能的修复措施：导致该错误的最常见原因是计算机上存在 GIT_SSH 环境变量，并且该变量配置为支持其他连接实用程序 (如 PuTTY)。要解决这个问题，请尝试以下操作之一：

- 打开 Bash 仿真器，在 Git 命令前添加 `GIT_SSH_COMMAND="ssh"` 参数。例如，在尝试克隆存储库时，不要运行 git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo，而是运行：

  ```
  GIT_SSH_COMMAND="ssh" git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/
  repos/MyDemoRepo my-demo-repo
  ```

- 重命名或删除不再使用的 `GIT_SSH` 环境变量。然后，打开新的命令提示符或 Bash 仿真器会话，再次尝试您的命令。

有关在 Windows 使用 SSH 时如何排查 Git 问题的更多信息，请参阅SSH 连接问题排查。

访问错误和 AWS CodeCommit 问题排查

以下信息可帮助您排查在连接 AWS CodeCommit 存储库时遇到的访问错误。

主题

访问错误：从 Windows 连接到 CodeCommit 存储库时，系统提示输入用户名和密码

问题：尝试使用 Git 与 CodeCommit 存储库通信时，出现一个对话框，提示您输入您的用户名和密码。

可能的修复措施：这可能是 Windows 内置的凭证管理系统造成的。根据您的配置，执行以下操作之一：

- 如果使用 Git 凭证进行 HTTPS 连接，则您的 Git 凭证尚未存储在系统中。请提供 Git 凭证并继续。应该不会再提示您。有关更多信息，请参阅适用于使用 Git 凭证的 HTTPS 用户。
- 如果使用 AWS CodeCommit 的凭证辅助程序进行 HTTPS 连接，它与 Windows 凭证管理系统不兼容。选择 Cancel (取消)。

  这也可能表明您在安装 Windows 版 Git 时安装了 Git Credential Manager。Git Credential Manager 与 AWS CLI 中包含的 CodeCommit 凭证助手不兼容。考虑卸载 Git Credential Manager。您还可以安装和配置 git-remote-codecommit 作为使用 CodeCommit 凭证助手的替代方法。

  有关更多信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤、适用于在 Windows 上使用 AWS CLI 凭证助手进行 HTTPS 连接和Windows 版 Git：我安装了 Windows 版 Git，但在访问我的存储库时被系统拒绝 (403)。

访问错误：连接到 CodeCommit 存储库时，公有密钥被拒绝

问题：尝试使用 SSH 端点与 CodeCommit 存储库通信时，出现一条错误消息，其中包含 `Error: public key denied` 字样。

可能的修复措施：导致出现此错误的最常见原因是您尚未完成 SSH 连接设置。请配置公有和私有 SSH 密钥对，然后将公有密钥与您的 IAM 用户相关联。有关配置 SSH 的更多信息，请参阅适用于 Linux、macOS 或 Unix 上的 SSH 连接 和适用于 Windows 上的 SSH 连接。

访问错误：连接到 CodeCommit 存储库时，出现“速率超出限制”或“429”消息

问题：尝试与 CodeCommit 存储库通信时，出现一条消息，指示“速率超出限制”或显示错误代码“429”。通信速度显著减慢或失败。

原因：对 CodeCommit 的所有调用（无论是来自应用程序、AWS CLI、Git 客户端还是 AWS Management Console）都受每秒最大请求数和活动请求总数的约束。您不得超过任何 AWS 区域的 Amazon Web Services 账户的最大允许请求速率。如果请求超过最大速率，您将收到一个错误，并且系统会对您的 Amazon Web Services 账户的进一步调用执行临时节流操作。在节流期间，您与 CodeCommit 的连接会变慢，甚至可能会失败。

可能的修复措施：采取措施以减少与 CodeCommit 的连接数或对 CodeCommit 的调用数或者分散请求。可考虑采用的一些方法：

- 在请求中实现抖动，尤其是在定期轮询请求中

  如果您有一个定期轮询 CodeCommit 的应用程序，并且该应用程序在多个 Amazon EC2 实例上运行，请引入抖动（随机延迟时长），以使不同的 Amazon EC2 实例不会在同一时间轮询。我们建议使用 0 到 59 秒的随机数值，以便在 1 分钟的时间范围内均匀分布轮询机制。
- 使用基于事件的架构而不是轮询

  使用基于事件的架构而不是轮询，以便仅在事件发生时进行调用。考虑将 CloudWatch Events 通知用于 AWS CodeCommit 事件以触发工作流。
- 实现 API 和自动 Git 操作的错误重试和指数回退

  错误重试和指数回退可帮助限制调用速率。每个 AWS 开发工具包均实施自动重试逻辑和指数回退算法。对于自动 Git 推送和 Git 拉取，您可能需要实施自己的重试逻辑。有关更多信息，请参阅 AWS 中的错误重试和指数回退。
- 在 AWS Support Center 请求提高 CodeCommit 服务限额

  要接收提高服务限制，您必须确认您已遵循此处提供的建议，包括实施错误重试或指数回退方法。在您的请求中，您还必须提供 AWS 区域、Amazon Web Services 账户以及受节流问题影响的时间范围。

配置错误和 AWS CodeCommit 问题排查

以下信息可帮助您排查在连接 AWS CodeCommit 存储库时可能遇到的配置错误。

主题

配置错误：无法在 macOS 上配置 AWS CLI 凭证

问题：在运行 `aws configure` 配置 AWS CLI 时，看到一条 `ConfigParseError` 消息。

可能的修复措施：导致该错误的最常见原因是凭证文件已存在。请浏览到 ~/.aws，查找名为 `credentials` 的文件。重命名或删除该文件，然后再次运行 aws configure。

控制台错误和 AWS CodeCommit 问题排查

以下信息可帮助您排查在使用 AWS CodeCommit 存储库时可能出现的控制台错误。

主题

访问错误：在控制台或 AWS CLI 中使用加密密钥访问 CodeCommit 存储库时被系统拒绝

问题：尝试在控制台或 AWS CLI 中访问 CodeCommit 时，出现一条错误消息，其中包含 `EncryptionKeyAccessDeniedException` 或 `User is not authorized for the KMS default key for CodeCommit 'aws/codecommit' in your account` 字样。

可能的修复措施：出现此错误的最常见原因是，您的 Amazon Web Services 账户未订阅 AWS Key Management Service（CodeCommit 要求必须订阅该服务）。打开 AWS KMS 控制台，选择 AWS 托管密钥，然后选择立即开始使用。如果看到一条消息，指示您当前没有订阅 AWS Key Management Service 服务，请按照页面上的说明订阅该服务。有关 CodeCommit 和 AWS Key Management Service 的更多信息，请参阅 AWS KMS 和加密。

加密错误：无法解密存储库

问题：当您尝试在控制台或 AWS CLI 中访问 CodeCommit 存储库时，出现一条错误消息，其中包含`Repository can't be decrypted`字样。

可能的修复措施：出现此错误的最常见原因是，用于加密和解密此存储库中数据的 AWS KMS 密钥处于非活跃状态或待删除。CodeCommit 需要 AWS Key Management Service 中的活跃 AWS 托管式密钥或客户托管密钥。打开 AWS KMS 控制台，选择 AWS 托管式密钥或客户托管密钥，确保用于存储库的密钥在存储库所在的 AWS 区域中可用，并且其状态为活跃。有关 CodeCommit 和 AWS Key Management Service 的更多信息，请参阅 AWS KMS 和加密。

> **Important**
>
> 如果用于加密和解密存储库中数据的密钥已被永久删除或因其他原因无法访问，则使用该密钥加密的存储库中的数据无法访问。

控制台错误：无法在控制台中浏览 CodeCommit 存储库中的代码

问题：尝试在控制台中浏览存储库中的内容时，出现一条拒绝访问的错误消息。

可能的修复措施：出现此错误的最常见原因是，应用于您的 Amazon Web Services 账户的某个 IAM 策略拒绝了在 CodeCommit 控制台中浏览代码所需的一个或多个权限。有关 CodeCommit 访问权限和浏览的更多信息，请参阅AWS CodeCommit 的身份验证和访问控制。

显示错误：无法查看文件或文件之间的对比

问题：尝试在 CodeCommit 控制台中查看文件或文件的两个版本之间的对比时，出现一个错误，指示文件或差异太大而无法显示。

可能的修复措施：出现此错误的最常见原因是，文件太大而无法显示，文件包含的一行或多行超过文件中单行的字符限制，或者文件的两个版本之间的差异超过了行限制。有关更多信息，请参阅配额。要查看文件或文件版本之间的差异，可以在首选 IDE 中在本地打开该文件，使用 Git diff 工具或运行 git diff 命令。

触发器和 AWS CodeCommit 问题排查

以下信息可帮助您排查 AWS CodeCommit 中的触发器问题。

主题

触发器错误：存储库触发器未按预期运行

问题：为存储库配置的一个或多个触发器无法正常工作。

可能的修复措施：如果触发器的目标是 AWS Lambda 函数，请确保为该函数配置了允许 CodeCommit 访问的资源策略。有关更多信息，请参阅示例 3：为 AWS Lambda 与 CodeCommit 触发器的集成创建策略。

或者，编辑触发器并确保选中要触发操作的事件，并且触发器的分支包含要对操作做出响应的分支。尝试将触发器设置更改为 All repository events 和 All branches，然后测试触发器。有关更多信息，请参阅编辑存储库的触发器。

启用调试

问题：我需要启用调试来获取有关我的存储库及 Git 如何执行命令的更多信息。

可能的修复措施：请尝试以下操作：

1. 在终端或命令提示符下，在本地计算机上运行以下命令，然后运行 Git 命令：

   在 Linux、macOS 或 Unix 上：

   ```
   export GIT_TRACE_PACKET=1
   export GIT_TRACE=1
   export GIT_CURL_VERBOSE=1
   ```

   在 Windows 上：

   ```
   set GIT_TRACE_PACKET=1
   set GIT_TRACE=1
   set GIT_CURL_VERBOSE=1
   ```

   > **Note**
   >
   > 设置 `GIT_CURL_VERBOSE` 只对 HTTPS 连接有用。SSH 不使用 `libcurl` 库。

2. 要获取有关您的 Git 存储库的更多信息，我们建议您安装最新版本的 git- sizer。按照说明安装适合您的操作系统和环境的实用程序。安装完成后，在命令行或终端上，将目录更改为本地存储库，然后运行以下命令：

```
git-sizer --verbose
```

> **Tip**
>
> 考虑将命令的输出保存到文件中，以便在排查问题时可以轻松地与他人共享，尤其是随着时间的推移。

AWS CodeCommit 参考

以下参考主题可以帮助您更好地理解 Git CodeCommit AWS 区域、、服务限制等。

主题

- 的地区和 Git 连接终端节点 AWS CodeCommit
- AWS CodeCommit 与接口 VPC 终端节点一起使用
- 中的配额 AWS CodeCommit
- AWS CodeCommit 命令行参考
- 基本 Git 命令

的地区和 Git 连接终端节点 AWS CodeCommit

每个 CodeCommit 存储库都与一个相关联 AWS 区域。 CodeCommit 提供区域终端节点，供您向服务提出请求。此外，还为每个可用的区域 CodeCommit 提供 SSH 和 HTTPS 协议 CodeCommit 的 Git 连接端点。

本指南中的所有示例都使用美国东部（俄亥俄州）中用于 Git 的同一端点 URL：`git-codecommit.us-east-2.amazonaws.com`但是，在使用 Git 并配置连接时，请确保选择与托管 CodeCommit 仓库的 Git 连接端点相匹配 AWS 区域 的 Git 连接端点。例如，如果您要连接美国东部（弗吉尼亚州北部）中的存储库，请使用端点 URL `git-codecommit.us-east-1.amazonaws.com`。这也适用于 API 调用。使用 AWS CLI 或软件开发工具包连接到 CodeCommit 存储库时，请确保为存储库使用正确的区域终端节点。

主题

- AWS 区域 支持 CodeCommit
- Git 连接端点
- 的服务器指纹 CodeCommit

AWS 区域 支持 CodeCommit

您可以在以下位置创建和使用 CodeCommit 存储库 AWS 区域：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚州北部）

- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- Europe (Paris)
- 欧洲地区（法兰克福）
- 欧洲地区（斯德哥尔摩）
- 欧洲地区（米兰）
- 非洲（开普敦）
- 以色列（特拉维夫）
- 亚太地区（东京)
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（雅加达）
- 中东（阿联酋）
- 亚太地区（首尔）
- 亚太地区（大阪）
- 亚太地区（孟买）
- 亚太地区（海得拉巴）
- 亚太地区（香港）
- 南美洲（圣保罗）
- 中东（巴林）
- 加拿大（中部）
- 中国（北京）
- 中国（宁夏）
- AWS GovCloud （美国西部）
- AWS GovCloud （美国东部）

CodeCommit 在某些地区增加了对联邦信息处理标准 (FIPS) 出版物 140-2 政府标准的支持。有关 FIPS 和 FIPS 终端节点的更多信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版概览。有关支持 FIPS 的 Git 连接终端节点的更多信息，请参阅 Git 连接端点。

有关区域终端节点 AWS CLI、服务和 API 调用的更多信息 CodeCommit，请参阅AWS CodeCommit 终端节点和配额。

Git 连接端点

在配置与 CodeCommit仓库的 Git 连接时，请使用以下 URL：

的 Git 连接端点 AWS CodeCommit

区域名称	区域	端点 URL	协议
美国东部（俄亥俄州）	us-east-2	https://git-codecommit.us-east-2.amazonaws.com	HTTPS
美国东部（俄亥俄州）	us-east-2	ssh://git-codecommit.us-east-2.amazonaws.com	SSH
美国东部（俄亥俄州）	us-east-2	https://git-codecommit-fips.us-east-2.amazonaws	HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	https://git-codecommit.us-east-1.amazonaws.com	HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	ssh://git-codecommit.us-east-1.amazonaws.com	SSH
美国东部（弗吉尼亚州北部）	us-east-1	https://git-codecommit-fips.us-east-1.amazonaws	HTTPS
美国西部（俄勒冈州）	us-west-2	https://git-codecommit.us-west-2.amazonaws.com	HTTPS

区域名称	区域	端点 URL	协议
美国西部（俄勒冈州）	us-west-2	ssh://git-codecommit.us-west-2.amazonaws.com	SSH
美国西部（俄勒冈州）	us-west-2	https://git-codecommit-fips.us-west-2.amazonaws	HTTPS
美国西部（北加利福尼亚）	us-west-1	https://git-codecommit.us-west-1.amazonaws.com	HTTPS
美国西部（北加利福尼亚）	us-west-1	ssh://git-codecommit.us-west-1.amazonaws.com	SSH
美国西部（北加利福尼亚）	us-west-1	https://git-codecommit-fips.us-west-1.amazonaws	HTTPS
欧洲地区（爱尔兰）	eu-west-1	https://git-codecommit.eu-west-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	ssh://git-codecommit.eu-west-1.amazonaws.com	SSH
亚太地区（东京)	ap-northeast-1	https://git-codecommit.ap-northeast-1.amazonaws.com	HTTPS
亚太地区（东京)	ap-northeast-1	ssh://git-codecommit.ap-northeast-1.amazonaws.com	SSH

区域名称	区域	端点 URL	协议
亚太地区（新加坡）	ap-southeast-1	https://git-codecommit.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（新加坡）	ap-southeast-1	ssh://git-codecommit.ap-southeast-1.amazonaws.com	SSH
亚太地区（悉尼）	ap-southeast-2	https://git-codecommit.ap-southeast-2.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	ssh://git-codecommit.ap-southeast-2.amazonaws.com	SSH
亚太地区（雅加达）	ap-southeast-3	https://git-codecommit.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（雅加达）	ap-southeast-3	ssh://git-codecommit.ap-southeast-3.amazonaws.com	SSH
中东（阿联酋）	me-central-1	https://git-codecommit.me-central-1.amazonaws.com	HTTPS
中东（阿联酋）	me-central-1	ssh://git-codecommit.me-central-1.amazonaws.com	SSH
欧洲（法兰克福）	eu-central-1	https://git-codecommit.eu-central-1.amazonaws.com	HTTPS

区域名称	区域	端点 URL	协议
欧洲地区（法兰克福）	eu-central-1	ssh://git-codecommit.eu-central-1.amazonaws.com	SSH
Asia Pacific (Seoul)	ap-northeast-2	https://git-codecommit.ap-northeast-2.amazonaws.com	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	ssh://git-codecommit.ap-northeast-2.amazonaws.com	SSH
南美洲（圣保罗）	sa-east-1	https://git-codecommit.sa-east-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	ssh://git-codecommit.sa-east-1.amazonaws.com	SSH
欧洲（伦敦）	eu-west-2	https://git-codecommit.eu-west-2.amazonaws.com	HTTPS
欧洲地区（伦敦）	eu-west-2	ssh://git-codecommit.eu-west-2.amazonaws.com	SSH
亚太地区（孟买）	ap-south-1	https://git-codecommit.ap-south-1.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	ssh://git-codecommit.ap-south-1.amazonaws.com	SSH

区域名称	区域	端点 URL	协议
亚太地区（海得拉巴）	ap-south-2	https://git-codecommit.ap-south-2.amazonaws.com	HTTPS
亚太地区（海得拉巴）	ap-south-2	ssh://git-codecommit.ap-south-2.amazonaws.com	SSH
加拿大（中部）	ca-central-1	https://git-codecommit.ca-central-1.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	ssh://git-codecommit.ca-central-1.amazonaws.com	SSH
加拿大（中部）	ca-central-1	https://git-codecommit-fips.ca-central-1.amazonaw	HTTPS
欧洲地区（巴黎）	eu-west-3	https://git-codecommit.eu-west-3.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	ssh://git-codecommit.eu-west-3.amazonaws.com	SSH
AWS GovCloud（美国西部）	us-gov-west-1	https://git-codecommit。 us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	SSH: //git-codecommit。 us-gov-west-1.amazonaws.com	SSH

区域名称	区域	端点 URL	协议
AWS GovCloud（美国西部）	us-gov-west-1	https://git-codecommit-fips。 us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	https://git-codecommit。 us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	SSH: //git-codecommit。 us-gov-east-1.amazonaws.com	SSH
AWS GovCloud（美国东部）	us-gov-east-1	https://git-codecommit-fips。 us-gov-east-1.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	https://git-codecommit.eu-north-1.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	ssh://git-codecommit.eu-north-1.amazonaws.com	SSH
中东（巴林）	me-south-1	https://git-codecommit.me-south-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	ssh://git-codecommit.me-south-1.amazonaws.com	SSH
亚太地区（香港）	ap-east-1	https://git-codecommit.ap-east-1.amazonaws.com	HTTPS

区域名称	区域	端点 URL	协议
亚太地区（香港）	ap-east-1	ssh://git-codecommit.ap-east-1.amazonaws.com	SSH
中国（北京）	cn-north-1	https://git-codecommit.cn-north-1.amazonaws.com.cn	HTTPS
中国（北京）	cn-north-1	ssh://git-codecommit.cn-north-1.amazonaws.com.cn	SSH
中国（宁夏）	cn-northwest-1	https://git-codecommit.cn-northwest-1.amazonaws.com.cn	HTTPS
中国（宁夏）	cn-northwest-1	ssh://git-codecommit.cn-northwest-1.amazonaws.com.cn	SSH
欧洲（米兰）	eu-south-1	https://git-codecommit.eu-south-1.amazonaws.com	HTTPS
欧洲（米兰）	eu-south-1	ssh://git-codecommit.eu-south-1.amazonaws.com	SSH
Asia Pacific (Osaka)	ap-northeast-3	https://git-codecommit.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacific (Osaka)	ap-northeast-3	ssh://git-codecommit.ap-northeast-3.amazonaws.com	SSH

区域名称	区域	端点 URL	协议
非洲（开普敦）	af-south-1	https://git-codecommit.af-south-1.amazonaws.com	HTTPS
非洲（开普敦）	af-south-1	ssh://git-codecommit.af-south-1.amazonaws.com	SSH
以色列（特拉维夫）	il-central-1	https://git-codecommit.il-central-1.amazonaws.com	HTTPS
以色列（特拉维夫）	il-central-1	ssh://git-codecommit.il-central-1.amazonaws.com	SSH

的服务器指纹 CodeCommit

下表列出了中 Git 连接端点的公共指纹。 CodeCommit验证过程中会显示这些服务器指纹，以用于向已知主机文件中添加终端节点。

的公共指纹 CodeCommit

Server	加密哈希类型	指纹
git-codecommit.us-east-2.amazonaws.com	MD5	`a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`
git-codecommit.us-east-2.amazonaws.com	SHA256	`3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`
git-codecommit.us-east-1.amazonaws.com	MD5	`a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84`

Server	加密哈希类型	指纹
git-codecommit.us-east-1.amazonaws.com	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git-codecommit.us-west-2.amazonaws.com	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git-codecommit.us-west-2.amazonaws.com	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git-codecommit.eu-west-1.amazonaws.com	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git-codecommit.eu-west-1.amazonaws.com	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git-codecommit.ap-northeast-1.amazonaws.com	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git-codecommit.ap-northeast-1.amazonaws.com	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git-codecommit.ap-southeast-1.amazonaws.com	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git-codecommit.ap-southeast-1.amazonaws.com	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno

Server	加密哈希类型	指纹
git-codecommit.ap-southeast-2.amazonaws.com	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0
git-codecommit.ap-southeast-2.amazonaws.com	SHA256	nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec
git-codecommit.ap-southeast-3.amazonaws.com	MD5	64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6
git-codecommit.ap-southeast-3.amazonaws.com	SHA256	ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g
git-codecommit.me-central-1.amazonaws.com	MD5	bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76
git-codecommit.me-central-1.amazonaws.com	SHA256	grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA
git-codecommit.eu-central-1.amazonaws.com	MD5	74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d
git-codecommit.eu-central-1.amazonaws.com	SHA256	MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY
git-codecommit.ap-northeast-2.amazonaws.com	MD5	9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed

Server	加密哈希类型	指纹
git-codecommit.ap-northeast-2.amazonaws.com	SHA256	eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko
git-codecommit.sa-east-1.amazonaws.com	MD5	74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51
git-codecommit.sa-east-1.amazonaws.com	SHA256	kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU
git-codecommit.us-west-1.amazonaws.com	MD5	3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1
git-codecommit.us-west-1.amazonaws.com	SHA256	gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg
git-codecommit.eu-west-2.amazonaws.com	MD5	a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3
git-codecommit.eu-west-2.amazonaws.com	SHA256	r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs
git-codecommit.ap-south-1.amazonaws.com	MD5	da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21
git-codecommit.ap-south-1.amazonaws.com	SHA256	hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE

Server	加密哈希类型	指纹
git-codecommit.ap-south-2.amazonaws.com	MD5	bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce
git-codecommit.ap-south-2.amazonaws.com	SHA256	Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig
git-codecommit.ca-central-1.amazonaws.com	MD5	9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23
git-codecommit.ca-central-1.amazonaws.com	SHA256	Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w
git-codecommit.eu-west-3.amazonaws.com	MD5	1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76
git-codecommit.eu-west-3.amazonaws.com	SHA256	uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI
git-codecommit us-gov-west-1.amazonaws.com	MD5	9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69
git-codecommit us-gov-west-1.amazonaws.com	SHA256	djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4
git-codecommit us-gov-east-1.amazonaws.com	MD5	00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2

Server	加密哈希类型	指纹
git-codecommit us-gov-east-1.amazonaws.com	SHA256	fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98
git-codecommit.eu-north-1.amazonaws.com	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
git-codecommit.eu-north-1.amazonaws.com	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git-codecommit.me-south-1.amazonaws.com	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git-codecommit.me-south-1.amazonaws.com	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git-codecommit.ap-east-1.amazonaws.com	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git-codecommit.ap-east-1.amazonaws.com	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git-codecommit.cn-north-1.amazonaws.com.cn	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git-codecommit.cn-north-1.amazonaws.com.cn	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8

Server	加密哈希类型	指纹
git-codecommit.cn-northwest-1.amazonaws.com.cn	MD5	2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6
git-codecommit.cn-northwest-1.amazonaws.com.cn	SHA256	wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ
git-codecommit.eu-south-1.amazonaws.com	MD5	b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11
git-codecommit.eu-south-1.amazonaws.com	SHA256	lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI
git-codecommit.ap-northeast-3.amazonaws.com	MD5	25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29
git-codecommit.ap-northeast-3.amazonaws.com	SHA256	2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA
git-codecommit.af-south-1.amazonaws.com	MD5	21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29
git-codecommit.af-south-1.amazonaws.com	SHA256	C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM
git-codecommit.il-central-1.amazonaws.com	MD5	04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9

Server	加密哈希类型	指纹
git-codecommit.il-central-1.amazonaws.com	SHA256	uFxhp51kUWhleTLeYbxQVYm4RnNLNZ5Dbdm1cgdSl/8

AWS CodeCommit 与接口 VPC 终端节点一起使用

如果您使用亚马逊虚拟私有云（Amazon VPC）托管 AWS 资源，则可以在您的 VPC 和之间建立私有连接 CodeCommit。您可以使用此连接 CodeCommit 来实现与您的 VPC 上的资源进行通信，而无需通过公共互联网。

Amazon VPC 是一项 AWS 服务，可用于在您定义的虚拟网络中启动 AWS 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。使用 VPC 终端节点，VPC 和 AWS 服务之间的路由由 AWS 网络处理，您可以使用 IAM 策略来控制对服务资源的访问。

要将您的 VPC 连接到 CodeCommit，您需要为定义接口 VPC 终端节点 CodeCommit。接口终端节点是一个带有私有 IP 地址的 elastic network 接口，该地址用作发往受支持 AWS 服务的流量的入口点。该端点 CodeCommit 无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接即可提供可靠、可扩展的连接。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC。

> **Note**
>
> 其他提供 VPC 支持并与 CodeCommit之集成的 AWS 服务（例如 AWS CodePipeline）可能不支持使用 Amazon VPC 终端节点进行集成。例如， CodePipeline 和之间的流量 CodeCommit 不能限制在 VPC 子网范围内。支持集成的服务（如 AWS Cloud9）可能需要其他服务（如 AWS Systems Manager）。

Interface VPC 终端节点由 AWS PrivateLink一种 AWS 技术提供支持，该技术使用带有私有 IP 地址的弹性网络接口实现 AWS 服务之间的私密通信。有关更多信息，请参阅AWS PrivateLink。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的入门。

可用性

CodeCommit 目前支持以下 VPC 终端节点 AWS 区域：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚州北部）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- Europe (Paris)
- 欧洲地区（法兰克福）
- 欧洲地区（斯德哥尔摩）
- 欧洲地区（米兰）
- 非洲（开普敦）
- 以色列（特拉维夫）
- 亚太地区（东京)
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（雅加达）
- 中东（阿联酋）
- 亚太地区（首尔）
- 亚太地区（大阪）
- 亚太地区（孟买）
- 亚太地区（海得拉巴）
- 亚太地区（香港）
- 南美洲（圣保罗）
- 中东（巴林）
- 加拿大（中部）
- 中国（北京）
- 中国（宁夏）
- AWS GovCloud （美国西部）
- AWS GovCloud （美国东部）

为创建 VPC 终端节点 CodeCommit

要开始在您的 VPC 中使用 CodeCommit ，请为创建一个接口 VPC 终端节点 CodeCommit。CodeCommitGit 操作和 CodeCommit API 操作需要单独的端点。根据您的业务需求，您可能需要创建多个 VPC 终端节点。为创建 VPC 终端节点时 CodeCommit，选择AWS 服务，然后在服务名称中，从以下选项中进行选择：

- com.amazonaws。 regi@@ ***on*** .git-codecomm it：如果您想使用存储库为 Git 操作创建 VPC 终端节点，请选择此选项。 CodeCommit 例如，如果您的用户使用 Git 客户端以及诸如、和之类的命令`git pull`，则`git push`当他们与 CodeCommit 仓库交互时`git commit`，请选择此选项。
- com.amazonaws。 ***##***。 git-codecommit-fips：如果您想使用符合联邦信息处理标准 (FIPS) 出版物 140-2 美国政府标准的 CodeCommit 存储库为 Git 操作创建 VPC 终端节点，请选择此选项。

> **Note**
>
> Git 的 FIPS 终端节点并非在所有 AWS 地区都可用。有关更多信息，请参阅 Git 连接端点。

- com.amazonaws。 regi@@ ***on*** .codecommit：如果您想为 CodeCommit API 操作创建 VPC 终端节点，请选择此选项。例如，如果您的用户使用、 CodeCommit API 或 AWS 软件开发工具包进行交互以执行诸如、和`PutFile`之类 CodeCommit 的操作`CreateRepository`、`ListRepositories`，请选择此选项。 AWS CLI
- com.amazonaws。 regi@@ ***on*** .codecommit-fips：如果您想为 CodeCommit API 操作创建符合联邦信息处理标准 (FIPS) 出版物 140-2 美国政府标准的 VPC 终端节点，请选择此选项。

> **Note**
>
> FIPS 终端节点并非在所有 AWS 地区都可用。有关更多信息，请参阅联邦信息处理标准 (FIPS) 140-2 概述 AWS CodeCommit 中的条目。

为创建 VPC 终端节点策略 CodeCommit

您可以为 Amazon VPC 终端节点创建策略，您可以在其中指定： CodeCommit

- 可执行操作的主体。
- 可执行的操作。
- 可用于执行操作的资源。

例如，公司可能希望将对存储库的访问限制为 VPC 的网络地址范围。您可以在此处查看此类策略的示例：示例 3：允许从指定 IP 地址范围连接的用户访问存储库 。公司为美国东部（俄亥俄州）区域配置了两个 Git VPC 端点：com.amazonaws.us-east-2.codecommit 和 com-amazonaws.us-east-2.git-codecommit-fips。他们只想允许将代码推送到*MyDemoRepo*仅在符合 FIPS 的端点上命名的 CodeCommit 存储库。为了强制执行此操作，他们将在 com.amazonaws.us-east-2.codecommit 终端节点上配置一个与以下策略类似的策略，该策略专用于拒绝 Git 推送操作：

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "codecommit:GitPush",
            "Effect": "Deny",
            "Resource": "arn:aws:codecommit:us-west-2:123456789012:MyDemoRepo",
            "Principal": "*"
        }
    ]
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的创建接口终端节点。

中的配额 AWS CodeCommit

下表描述了中的配额 CodeCommit。有关可以更改的配额的信息，请参阅 AWS CodeCommit 终端节点和配额。有关请求增加服务限额的更多信息，请参阅 AWS 服务限额。有关 Git 和其他软件所需版本的信息，请参阅CodeCommit、Git 及其他组件的兼容性。

审批规则和审批规则模板名称	任意字母、数字、点号、空格、下划线和短划线的组合，长度在 1 到 100 个字符之间。名称区分大小写。名称不能以 .git 结尾，并且不能包含以下任意字符：! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :

审批规则内容长度	3000 个字符
审批规则模板描述长度	1000 个字符
审批规则模板目标引用	100
审批规则模板	1000 一进一步 AWS 区域
拉取请求的审批规则	最多 30 个。其中最多 25 个可以来自审批规则模板。
从审批规则模板创建的拉取请求的审批规则	25
对拉取请求的审批	200
审批池中的审批人	50
分支名称	允许长度介于 1 到 256 个字符的任意字符组合，但不允许正好包含 40 个十六进制字符的分支名称。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。
评论长度	最多可使用 10240 个字符。
触发器的自定义数据	这是一个限制为 1000 个字符的字符串字段。它不能用于传递任何动态参数。

在控制台中显示	在以下情况下，可能无法在控制台中查看文件或文件之间的比较： • 文件大于 2 MB • 文件在一行中包含超过 25000 个字符 • 比较总共包含超过 6500 行差异
在控制台中提交的电子邮件地址	允许的字符的任意组合，长度在 1 到 256 个字符之间。不会验证电子邮件地址。

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 CodeCommit 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 CodeCommit 控制台、API 或时，任何单个文件的最大容量为 6 MB AWS CLI。

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。
提交可视化工具中的分支图形显示	每页 35 个。如果单个页面上有超过 35 个分支，图形将不显示。
提交的元数据	使用 CodeCommit 控制台、API 或时，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 20 MB AWS CLI。 **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，单个文件不超过 6 MB，并且单个 blob 不超过 2 GB。
一个提交中的文件数	最多 100 个。
未处理的拉取请求数	最多 1000 个。
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。
存储库的数目	每个 Amazon Web Services 账户最多 5000 个。此限制可以更改。有关更多信息，请参阅 AWS CodeCommit 端点和配额和 AWS 服务限额。
一个存储库中的触发器数	最大 10。

区域	CodeCommit 有以下几种版本 AWS 区域：
	• 美国东部（俄亥俄） • 美国东部（弗吉尼亚州北部） • 美国西部（加利福尼亚北部） • 美国西部（俄勒冈） • 欧洲地区（爱尔兰） • 欧洲地区（伦敦） • Europe (Paris) • 欧洲地区（法兰克福） • 欧洲地区（斯德哥尔摩） • 欧洲地区（米兰） • 非洲（开普敦） • 以色列（特拉维夫） • 亚太地区（东京) • 亚太地区（新加坡） • 亚太地区（悉尼） • 亚太地区（雅加达） • 中东（阿联酋） • 亚太地区（首尔） • 亚太地区（大阪） • 亚太地区（孟买） • 亚太地区（海得拉巴） • 亚太地区（香港） • 南美洲（圣保罗） • 中东（巴林） • 加拿大（中部） • 中国（北京） • 中国（宁夏） • AWS GovCloud （美国西部） • AWS GovCloud （美国东部）

	有关更多信息，请参阅 区域和 Git 连接端点。
存储库描述	任意字符组合，长度在 0 到 1000 个字符之间。存储库描述是可选的。
存储库名称	任意字母、数字、点号、下划线和短划线的组合，长度在 1 到 100 个字符之间。名称区分大小写。存储库名称不能以 .git 结尾，并且不能包含以下任意字符：! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
存储库标签键名称	UTF-8 格式的 Unicode 字母、数字、空格和允许使用的字符的任意组合，长度为 1 到 128 个字符。允许使用的字符为 + - = . _ : / @ 标签键名称必须是唯一的，而且每个键只能有一个值。标签不能： • 以 aws: 开头 • 只包含空格 • 以空格结尾 • 包含表情符号或以下任意字符：? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;
存储库标签值	UTF-8 格式的 Unicode 字母、数字、空格和允许使用的字符的任意组合，长度为 1 到 256 个字符。允许使用的字符为 + - = . _ : / @ 一个键只能有一个值，但许多键可以具有相同的值。标签不能： • 只包含空格 • 以空格结尾 • 包含表情符号或以下任意字符： ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

存储库标签	标签区分大小写。每个资源最多 50 个。不允许使用正好包含 40 个十六进制字符的标签名称。
触发器名称	任意字母、数字、点号、下划线和短划线的组合，长度在 1 到 100 个字符之间。触发器名称不能包含空格或逗号。
在控制台中提交的用户名	允许的字符的任意组合，长度在 1 到 1024 个字符之间。

AWS CodeCommit 命令行参考

此参考资料可帮助您了解如何使用 AWS CLI。

要安装和配置 AWS CLI

1. 在您的本地计算机上，下载并安装 AWS CLI。这是通过命令行与交互 CodeCommit 的先决条件。我们建议您安装 AWS CLI 版本 2。它是的最新主要版本 AWS CLI ，支持所有最新功能。它是唯一支持使用根账户、联合访问权限或临时证书的版本git-remote-codecommit。 AWS CLI

   有关更多信息，请参阅使用 AWS 命令行界面进行设置。

   > **Note**
   >
   > CodeCommit 仅适用于 1.7.38 及更高 AWS CLI 版本。作为最佳实践，请安装或升级 AWS CLI 到可用的最新版本。要确定 AWS CLI 您安装了哪个版本，请运行aws --version命令。
   > 要将旧版本的升级 AWS CLI 到最新版本，请参阅安装 AWS Command Line Interface。

2. 运行此命令以验证的 CodeCommit 命令 AWS CLI 是否已安装。

   ```
   aws codecommit help
   ```

   此命令返回 CodeCommit 命令列表。

3. AWS CLI 使用configure命令配置配置文件，如下所示：。

   ```
   aws configure
   ```

出现提示时，指定要与之 AWS 配合使用的 IAM 用户的访问 AWS 密钥和私有访问密钥 CodeCommit。另外，请务必指定存储库的 AWS 区域 存在位置，例如us-east-2。系统提示指定默认输出格式时，指定 json。例如，如果您正在为 IAM 用户配置相关配置文件：

```
AWS Access Key ID [None]: Type your IAM user AWS access key ID here, and then press
 Enter
AWS Secret Access Key [None]: Type your IAM user AWS secret access key here, and
 then press Enter
Default region name [None]: Type a supported region for CodeCommit here, and then
 press Enter
Default output format [None]: Type json here, and then press Enter
```

有关创建和配置用于的配置文件的更多信息 AWS CLI，请参阅以下内容：

- 命名配置文件
- 在中使用 IAM 角色 AWS CLI
- 设置命令
- 使用轮换凭证连接到 AWS CodeCommit 存储库

要连接到存储库或其他存储库中的资源 AWS 区域，必须 AWS CLI 使用默认的区域名称重新配置。支持的默认区域名称 CodeCommit 包括：

- us-east-2
- us-east-1
- eu-west-1
- us-west-2
- ap-northeast-1
- ap-southeast-1
- ap-southeast-2
- ap-southeast-3
- me-central-1
- eu-central-1
- ap-northeast-2
- sa-east-1

- us-west-1
- eu-west-2
- ap-south-1
- ap-south-1
- ca-central-1
- us-gov-west-1
- us-gov-east-1
- eu-north-1
- ap-east-1
- me-south-1
- cn-north-1
- cn-northwest-1
- eu-south-1
- ap-northeast-3
- af-south-1
- il-central-1

有关 CodeCommit 和的更多信息 AWS 区域，请参阅区域和 Git 连接端点。有关 IAM、访问密钥和秘密密钥的更多信息，请参阅如何获取凭证？和管理 IAM 用户的访问密钥。有关 AWS CLI 和配置文件的更多信息，请参阅命名配置文件。

要查看所有可用 CodeCommit 命令的列表，请运行以下命令：

```
aws codecommit help
```

要查看有关 CodeCommit 命令的信息，请运行以下命令，其中 co *mmand-nam* e 是命令的名称（例如，create-repository）：

```
aws codecommit command-name help
```

请参阅以下内容，查看 AWS CLI中命令的描述和示例用法：

- associate-approval-rule-template-with 存储库

- batch-associate-approval-rule-template-with-repositories
- batch-disassociate-approval-rule-template-from-repositories
- batch-describe-merge-conflicts
- batch-get-commits
- batch-get-repositories
- create-approval-rule-template
- create-branch
- create-commit
- create-pull-request
- create-pull-request-approval-规则
- create-repository
- create-unreferenced-merge-commit
- delete-approval-rule-template
- delete-branch
- delete-comment-content
- delete-file
- delete-repository
- describe-merge-conflicts
- delete-pull-request-approval-规则
- describe-pull-request-events
- disassociate-pull-request-approval-rule-template-from-repository
- evaluate-pull-request-approval-规则
- get-approval-rule-template
- get-blob
- get-branch
- get-comment
- get-comment-reactions
- get-comments-for-compared-提交

- get-comments-for-pull-请求
- get-commit
- get-differences
- get-merge-commit
- get-merge-conflicts
- get-merge-options
- get-pull-request
- get-pull-request-approval-州
- get-pull-request-override-州
- get-repository
- get-repository-triggers
- list-approval-rule-templates
- list-associated-approval-rule-templates-for-repository
- list-branches
- list-pull-requests
- list-repositories
- list-repositories-for-approval-规则模板
- list-tags-for-resource
- merge-branches-by-fast-向前
- merge-branches-by-squash
- merge-branches-by-three-way
- merge-pull-request-by-快进
- merge-pull-request-by-南瓜
- merge-pull-request-by-三向
- override-pull-request-approval-规则
- post-comment-for-compared-提交
- post-comment-for-pull-请求
- post-comment-reply
- put-comment-reaction

- put-file
- put-repository-triggers
- tag-resource
- test-repository-triggers
- untag-resource
- update-approval-rule-template-内容
- update-approval-rule-template-描述
- update-approval-rule-template-名字
- update-comment
- update-default-branch
- update-pull-request-approval-规则内容
- update-pull-request-approval-州
- update-pull-request-description
- update-pull-request-status
- update-pull-request-title
- update-repository-description
- update-repository-name

基本 Git 命令

您可以使用 Git 来处理本地 CodeCommit 存储库和已连接本地存储库的存储库。

以下是常用 Git 命令的一些基本示例。

有关更多选项，请参阅 Git 文档。

主题

- 配置变量
- 远程存储库
- 提交
- Branches

- 标签

配置变量

列出所有配置变量。	`git config --list`
只列出本地配置变量。	`git config --local -l`
只列出系统配置变量。	`git config --system -l`
只列出全局配置变量。	`git config --global -l`
在指定的配置文件中设置配置变量。	`git config [--local | --global | --system] variable-name variable-value`
在对还没有默认分支的存储库进行初始提交时，将所有本地存储库的默认分支名称设置为 main	git config --global init.defaultBranch main
直接编辑配置文件。也可用来发现特定配置文件的位置。要退出编辑模式，通常可键入 :q (不保存更改并退出) 或 :wq (保存更改并退出)，然后按 Enter。	`git config [--local | --global | --system] --edit`

远程存储库

初始化本地存储库，为将其连接到存储库做准备。 CodeCommit	`git init`
可用于在本地存储库和远程存储库（例如存储库）之间建立连接，使用本地 CodeCommit 存储库为存储库指定的昵称和指向 CodeCommit 存储库的指定 URL。 CodeCommit	`git remote add remote-name remote-url`
通过在本地计算机上当前文件夹的指定子文件夹中以指定 URL 复制 CodeCommit 存储库来创建本地存储库。此命令还会为克隆 CodeCommit	`git clone remote-url local-sub folder-name`

存储库中的每个分支创建一个远程跟踪分支，并创建和检出从克隆存储库中当前默认分支派生的初始分支。 CodeCommit	
显示本地存储库为 CodeCommit 存储库使用的昵称。	`git remote`
显示本地存储库用于提取和推送到存储库的昵称和网址。 CodeCommit	`git remote -v`
使用本地存储库为 CodeCommit 存储库和指定分支设置的指定昵称，将已完成的提交从本地存储库推送到 CodeCommit 存储库。推送期间还会为本地存储库设置上游跟踪信息。	`git push -u` *`remote-name`* *`branch-name`*
设置上游跟踪信息后，将已完成的提交从本地 CodeCommit 存储库推送到存储库。	`git push`
使用本地存储库为存储库和指定分支设置的指定昵称，从 CodeCommit 存储库中提取到本地 CodeCommit 存储库的最终提交	`git pull` *`remote-name`* *`branch-name`*
设置上游跟踪信息后，将已完成的提交从 CodeCommit 存储库中提取到本地存储库。	`git pull`
使用本地存储库为 CodeCommit 存储库指定的昵称，断开本地存储库与存储库的连接。 CodeCommit	`git remote rm` *`remote-name`*

提交

显示已添加或尚未添加到本地存储库中待处理提交的内容。	`git status`
以简明格式显示已添加或尚未添加到本地存储库中的待处理提交的内容。 (M = 已修改，A = 已添加，D = 已删除，等)	`git status -sb`

显示本地存储库中待处理提交和最新提交之间的更改。	`git diff HEAD`
将特定文件添加到本地存储库中的待处理提交。	`git add [file-name-1 file-name-2 file-name-N | file-pattern ]`
将所有新建的、修改的和删除的文件添加到本地存储库中的待处理提交。	`git add`
开始最终确定本地存储库中的待处理提交，这会显示一个编辑器供您输入提交消息。输入消息后，待处理提交变成最终确定状态。	`git commit`
最终确定本地存储库中的待处理提交，包括指定提交消息。	`git commit -m "Some meaningful commit comment"`
列出本地存储库中的最新提交。	`git log`
以图形格式列出本地存储库中的最新提交。	`git log --graph`
以预定义的紧缩格式列出本地存储库中的最新提交。	`git log --pretty=oneline`
以预定义的紧缩格式列出本地存储库中的最新提交，同时附上图形。	`git log --graph --pretty=oneline`
以自定义格式列出本地存储库中的最新提交，同时附上图形。 (有关更多选项，请参阅 Git 基础 - 查看提交历史)	`git log --graph --pretty=format:"%H (%h) : %cn : %ar : %s"`

Branches

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`

将有关存储库中所有现有分支的信息提取到本地 CodeCommit 存储库。	`git fetch`
列出本地存储库中的所有分支和本地存储库中的远程跟踪分支。	`git branch -a`
只列出本地存储库中的远程跟踪分支。	`git branch -r`
使用指定的分支名称在本地存储库中创建一个新的分支。	`git branch new-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`
使用指定的分支名称在本地存储库中创建一个新的分支，然后切换到该分支。	`git checkout -b new-branch-name`
使用本地存储库为 CodeCommit 仓库设置的指定昵称和指定的分支名称，将新分支从本地 CodeCommit 存储库推送到存储库。推送期间会为本地存储库中的分支设置上游跟踪信息。	`git push -u remote-name new-branch-name`
使用指定的分支名称在本地存储库中创建一个新的分支。然后，使用本地存储库为存储库设置的指定昵称和指定的分支名称，将本地 CodeCommit 存储库中的新分支连接到 CodeCommit 存储库中的现有分支。	`git branch --track new-branch-name remote-name /remote-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 CodeCommit 存储库为仓库设置的指定昵称和指定的分支名称删除 CodeCommit 存储库中的分支。(注意冒号 (:) 的用法。)	`git push remote-name :branch-name`

标签

列出本地存储库中的所有标签。	`git tag`
将所有标签从 CodeCommit 存储库提取到本地存储库。	`git fetch --tags`
显示本地存储库中有关特定标签的信息。	`git show tag-name`
在本地存储库中创建“轻型”标签。	`git tag tag-name commit-id-to-point-tag-at`
使用本地存储库为 CodeCommit 存储库设置的指定昵称和指定的标签名称，将特定标签从本地存储库推送到 CodeCommit 存储库。	`git push remote-name tag-name`
使用本地存储库为 CodeCommit 仓库设置的指定昵称，将所有标签从本地存储库推送到存储库。 CodeCommit	`git push remote-name --tags`
删除本地存储库中的某个标签。	`git tag -d tag-name`
使用本地 CodeCommit 存储库为存储库设置的指定昵称和指定的标签名称，删除 CodeCommit 存储库中的标签。(注意冒号 (:) 的用法。)	`git push remote-name :tag-name`

AWS CodeCommit 用户指南文档历史记录

下表介绍了对 CodeCommit 文档的一些重要更改。要获得本文档的更新通知，您可以订阅 RSS 源。

- API 版本：2015-04-13

变更	说明	日期
CodeCommit 现在支持使用客户托管密钥	现在，您可以使用客户托管密钥或 AWS 托管式密钥来加密和解密存储库中的数据。有关更多信息，请参阅AWS KMS和加密、创建存储库和更改存储库设置。	2023 年 12 月 21 日
CodeCommit 现已在以色列（特拉维夫）推出；	CodeCommit 现已在以色列（特拉维夫）推出。有关更多信息，请参阅区域和 Git 连接终端节点。	2023 年 8 月 28 日
CodeCommit 托管策略的更改	AWSCodeCommitPowerUser 和 AWSCodeCommitFullAccess 策略已更新，增加了一项权限。有关更多信息，请参阅 CodeCommit 对 AWS 托管策略的更新。	2023 年 5 月 16 日
CodeCommit 现已在其他三个 AWS 区域推出	CodeCommit 现已在其他三个 AWS 区域推出：亚太地区（雅加达）、中东（阿联酋）和亚太地区（海得拉巴）。有关更多信息，请参阅区域和 Git 连接终端节点。	2023 年 2 月 28 日
CodeCommit 现已在非洲（开普敦）推出	CodeCommit 现已在其他 AWS 区域推出：非洲（开普敦）。	2021 年 9 月 15 日

	有关更多信息，请参阅区域和 Git 连接终端节点。	
CodeCommit 托管策略的更改	现已公布有关 CodeCommit 的 AWS 托管策略更新的详细信息。有关更多信息，请参阅 CodeCommit 对 AWS 托管策略的更新。	2021 年 8 月 18 日
CodeCommit 现已在亚太地区（大阪）推出	CodeCommit 现已在其他 AWS 区域推出：亚太地区（大阪）。有关更多信息，请参阅区域和 Git 连接终端节点。	2021 年 4 月 14 日
AWS CloudFormation 和 AWS Cloud Development Kit (AWS CDK) 更改了 CodeCommit 中默认分支的命名行为	使用 AWS CloudFormation 创建的存储库或包含初始代码提交的 AWS CDK 现在使用默认分支名称 main。这一变更不会影响现有的存储库或分支。使用本地 Git 客户端创建初始提交的客户有一个默认分支名称，该名称遵循这些 Git 客户端的配置。有关更多信息，请参阅使用 AWS CloudFormation 创建 CodeCommit 资源。	2021 年 3 月 4 日
CodeCommit 更改了默认分支的命名行为	自 2021 年 1 月 19 日起，初始提交向 CodeCommit 存储库创建的默认分支名称为 main。这一变更不会影响现有的存储库或分支。使用本地 Git 客户端创建初始提交的客户有一个默认分支名称，该名称遵循这些 Git 客户端的配置。有关更多信息，请参阅使用分支、创建提交和更改分支设置。	2021 年 1 月 19 日

CodeCommit 现已在欧洲地区（米兰）推出	CodeCommit 现已在其他 AWS 区域推出：欧洲地区（米兰）。有关更多信息，请参阅区域和 Git 连接终端节点。	2020 年 9 月 16 日
CodeCommit 增加了对使用表情符号回复评论的支持	CodeCommit 现在支持使用表情符号来回复其他用户的评论。有关更多信息，请参阅评论提交和审核拉取请求。	2020 年 6 月 24 日
CodeCommit 现已在中国（北京）和中国（宁夏）推出	CodeCommit 现已在两个其他 AWS 区域推出：中国（北京）和中国（宁夏）。有关更多信息，请参阅区域和 Git 连接终端节点。	2020 年 4 月 23 日
CodeCommit 增加了对 git-remote-codecommit 的支持	CodeCommit 支持使用 git-remote-codecommit（用于修改 Git 的实用程序）通过 HTTPS 连接到 CodeCommit 存储库。这是对 CodeCommit 存储库进行联合或临时访问连接的推荐方法。您也可以将 git-remote-codecommit 与 IAM 用户一起使用。git-remote-codecommit 不要求您为用户设置 Git 凭证。有关详细信息，请参阅使用 git-remote-codecommit 建立到 AWS CodeCommit 的 HTTPS 连接的设置步骤。	2020 年 3 月 4 日

CodeCommit 支持会话标签	CodeCommit 支持使用会话标签，这些标签是您在代入 IAM 角色、使用临时凭证或在 AWS Security Token Service (AWS STS) 中对用户进行联合身份验证时传递的键值对属性。您可以使用这些标签中提供的信息，以便更轻松地识别谁进行了更改或导致了事件。有关更多信息，请参阅监控 CodeCommit 和在 CodeCommit 中使用标签提供标识信息。	2019 年 12 月 19 日
CodeCommit 现已在亚太地区（香港）推出	现在，您可以在亚太地区（香港）使用 CodeCommit。有关更多信息（包括 Git 连接终端节点），请参阅区域。	2019 年 12 月 11 日
CodeCommit 支持 Amazon CodeGuru Reviewer	CodeCommit 支持 Amazon CodeGuru Reviewer，后者是一项自动代码审查服务，它使用程序分析和机器学习来检测 Java 或 Python 代码中的常见问题并提供修复建议。有关更多信息，请参阅将存储库与 Amazon CodeGuru Reviewer 关联或取消关联和使用拉取请求。	2019 年 12 月 3 日
CodeCommit 支持审批规则	现在，您可以使用审批规则模板帮助自定义跨存储库的开发工作流程，以便不同分支拥有适当的拉取请求审批和控制级别。有关更多信息，请参阅使用审批规则模板和使用拉取请求。	2019 年 11 月 20 日

CodeCommit 支持通知规则	现在，您可以使用通知规则向用户通知存储库中的重要更改。此功能取代了 2019 年 11 月 5 日之前创建的通知。有关更多信息，请参阅创建通知规则。	2019 年 11 月 5 日
CodeCommit 已在中东（巴林）推出	现在，您可以在中东（巴林）使用 CodeCommit。有关更多信息（包括 Git 连接终端节点），请参阅区域。	2019 年 10 月 30 日
CodeCommit 增加了对检索有关多个提交的信息的支持	您可以使用 AWS CLI 中的 batch-get-commits 命令获取有关多个提交的信息。有关更多信息，请参阅查看提交详细信息。	2019 年 8 月 15 日
CodeCommit 已在欧洲地区（斯德哥尔摩）推出	现在，您可以在欧洲地区（斯德哥尔摩）使用 CodeCommit。有关更多信息（包括 Git 连接终端节点），请参阅区域。	2019 年 7 月 31 日
CodeCommit 在 CodeCommit 控制台中增加了对为存储库添加标签的支持	现在，您可以为存储库添加、管理和删除标签以帮助您在 CodeCommit 控制台中管理 AWS 资源。有关更多信息，请参阅标记存储库。	2019 年 7 月 2 日
CodeCommit 增加了对更多 Git 合并策略的支持	现在，在 CodeCommit 中合并拉取请求时，您可以在 Git 合并策略之间进行选择。您也可以在 CodeCommit 控制台中解决合并冲突。有关更多信息，请参阅使用拉取请求。	2019 年 6 月 10 日

CodeCommit 已在 AWS GovCloud（美国东部）推出	现在，您可以在 AWS GovCloud（美国东部）使用 CodeCommit。有关更多信息（包括 Git 连接终端节点），请参阅区域。	2019 年 5 月 31 日
CodeCommit 增加了对为存储库添加标签的支持	现在，您可以为存储库添加、管理和移除标签以帮助您管理 AWS 资源。有关更多信息，请参阅标记存储库。	2019 年 5 月 30 日
在控制台中查找资源	现在，您可以快速搜索您的资源，如存储库、构建项目、部署应用程序和管道。选择转到资源或按下 / 键，然后键入资源的名称。有关更多信息，请参阅 CodeCommit 教程。	2019 年 5 月 14 日
CodeCommit 已在 AWS GovCloud（美国西部）推出	现在，您可以在 AWS GovCloud（美国西部）使用 CodeCommit。有关更多信息（包括 Git 连接终端节点），请参阅区域。	2019 年 4 月 18 日
CodeCommit 增加了对 Amazon VPC 端点的支持	现在，您可以在 VPC 和 CodeCommit 之间建立私有连接。有关更多信息，请参阅将 CodeCommit 与接口 VPC 端点一起使用。	2019 年 3 月 7 日
CodeCommit 增加了一个新 API	CodeCommit 增加了一个用于创建提交的 API。有关更多信息，请参阅创建提交。	2019 年 2 月 20 日
内容更新	本指南中的内容已更新，包含一些小的修补程序和其他故障排除指南。	2019 年 1 月 2 日

内容更新	为了支持全新的 CodeCommit 控制台体验，我们更新了本指南中的内容。	2018 年 10 月 30 日
CodeCommit 和美国联邦信息处理标准 (FIPS)	CodeCommit 在某些区域增加了对美国联邦信息处理标准 (FIPS) 第 140-2 版政府标准的支持。有关 FIPS 和 FIPS 终端节点的更多信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版概览。有关 Git 连接终端节点的更多信息，请参阅区域。	2018 年 10 月 25 日
CodeCommit 增加了三个 API	CodeCommit 增加了三个 API 以支持处理文件。有关 Git 连接端点的更多信息，请参阅对单个文件执行操作所需的权限和 AWS CodeCommit API 参考。	2018 年 9 月 27 日
可通过 RSS 源接收 CodeCommit 文档历史记录通知	现在，您可以通过订阅 RSS 源接收 CodeCommit 文档更新的通知。	2018 年 6 月 29 日

早期更新

下表介绍对 2018 年 6 月 29 日之前的文档的一些重要更改。

更改	描述	更改日期
新主题	增加了限制针对分支的推送和合并主题。更新了CodeCommit 权限参考主题。	2018 年 5 月 16 日
新的章节	增加了使用 AWS CodeCommit 存储库中的文件部分。更新了CodeCommit 权限参考和入门 AWS CodeCommit主题。	2018 年 2 月 21 日

更改	描述	更改日期
新主题	增加了使用角色配置对 AWS CodeCommit 仓库的跨账户访问权限主题。	2018 年 2 月 21 日
新主题	增加了将 AWS Cloud9 与 AWS CodeCommit 集成主题。产品和服务集成主题更新了有关 AWS Cloud9 的信息。	2017 年 12 月 1 日
新的章节	增加了使用 AWS CodeCommit 存储库中的拉取请求部分。已使用有关拉取请求和评论的权限的信息更新AWS CodeCommit 的身份验证和访问控制部分。该部分还包括更新后的托管策略语句。	2017 年 11 月 20 日
更新的主题	产品和服务集成主题已更新为包含面向以下客户的链接：需要更新其现有管道以使用 Amazon CloudWatch Events 启动管道来响应 CodeCommit 存储库中的更改。	2017 年 10 月 11 日
新主题	增加了AWS CodeCommit 的身份验证和访问控制部分。它取代访问权限参考主题。	2017 年 9 月 11 日
更新的主题	更新了管理存储库触发器部分以反映触发器配置中的更改。更新了整个指南中的主题和图像以反映导航栏中的更改。	2017 年 29 月 8 日
新主题	增加了使用用户首选项主题。更新了查看标签详细信息主题。产品和服务集成主题更新了有关与 Amazon CloudWatch Events 集成的信息。	2017 年 8 月 3 日
新主题	增加了将 Eclipse 与 AWS CodeCommit 集成 和将 Visual Studio 与 AWS CodeCommit 集成 主题。	2017 年 6 月 29 日
更新的主题	CodeCommit 现已在其他两个区域推出：亚太地区（孟买）和加拿大（中部）。更新了区域和 Git 连接端点主题。	2017 年 6 月 29 日
更新的主题	CodeCommit 现已在其他四个区域推出：亚太地区（首尔）、南美洲（圣保罗）、美国西部（北加利福尼亚）和欧洲地区（伦敦）。更新了区域和 Git 连接端点主题。	2017 年 6 月 6 日

更改	描述	更改日期
更新的主题	CodeCommit 现已在其他四个区域推出：亚太地区（东京)、亚太地区（新加坡）、亚太地区（悉尼）和欧洲地区（法兰克福）。更新了区域和 Git 连接端点主题，提供有关 CodeCommit 的 Git 连接端点和支持的区域的信息。	2017 年 5 月 25 日
新主题	增加了比较和合并分支主题。使用分支部分更新了有关通过 CodeCommit 控制台使用存储库中的分支的信息的内容。	2017 年 5 月 18 日
新主题	增加了比较提交主题，提供有关比较提交的信息。更新了用户指南结构，以调整存储库、提交和分支的相关内容。	2017 年 3 月 28 日
更新的主题	更新了查看提交详细信息主题，提供有关在控制台中查看提交与其父级之间的差异以及通过 AWS CLI 使用 get-differences 命令查看提交间的差异的信息。	2017 年 1 月 24 日
新主题	增加了使用 AWS CodeCommit 记录 AWS CloudTrail API 调用主题，提供有关使用 AWS CloudFormation 记录与 CodeCommit 的连接的信息。	2017 年 1 月 11 日
新主题	增加了适用于使用 Git 凭证的 HTTPS 用户主题，提供有关通过 HTTPS 使用 Git 凭证设置与 CodeCommit 的连接的信息。	2016 年 12 月 22 日
更新的主题	更新了产品和服务集成主题，纳入有关与 AWS CodeBuild 的集成的信息。	2016 年 12 月 5 日
更新的主题	CodeCommit 现已在其他区域推出：欧洲地区（爱尔兰）。更新了区域和 Git 连接端点主题，提供有关 CodeCommit 的 Git 连接端点和支持的区域的信息。	2016 年 11 月 16 日
更新的主题	CodeCommit 现已在其他区域推出：美国西部（俄勒冈州）。更新了区域和 Git 连接端点主题，提供有关 CodeCommit 的 Git 连接端点和支持的区域的信息。	2016 年 11 月 14 日

更改	描述	更改日期
新主题	更新了为 Lambda 函数创建触发器主题，以反映在创建 Lambda 函数的过程中创建 CodeCommit 触发器的功能。此简化流程简化了触发器的创建，并为触发器自动配置 CodeCommit 调用 Lambda 函数所需的权限。增加了为现有的 Lambda 函数创建触发器主题，纳入有关在 CodeCommit 控制台中为现有 Lambda 函数创建触发器的信息。	2016 年 10 月 19 日
新主题	CodeCommit 现已在其他区域推出：美国东部（俄亥俄州）。增加了区域和 Git 连接端点主题，提供有关 CodeCommit 的 Git 连接端点和支持的区域的信息。	2016 年 10 月 17 日
主题更新	更新了产品和服务集成主题，纳入有关与 AWS Elastic Beanstalk 的集成的信息。	2016 年 10 月 13 日
主题更新	更新了产品和服务集成主题，纳入有关与 AWS CloudFormation 的集成的信息。	2016 年 10 月 6 日
主题更新	修订了适用于 Windows 上的 SSH 连接主题，为在 Windows 上对 SSH 连接使用 Bash 仿真器而不是 PuTTY 工具套件提供指导。	2016 年 9 月 29 日
主题更新	更新了查看提交详细信息和 入门 CodeCommit 主题，纳入有关 CodeCommit 控制台中的提交可视化工具的信息。更新了配额主题，增加了单个推送中允许的引用数。	2016 年 9 月 14 日
主题更新	更新了查看提交详细信息和 入门 CodeCommit 主题，纳入有关在 CodeCommit 控制台中查看提交历史记录的信息。	2016 年 7 月 28 日
新主题	增加了将 Git 存储库迁移到 AWS CodeCommit 和将本地或非版本控制内容迁移到 AWS CodeCommit 主题。	2016 年 6 月 29 日
主题更新	对故障排除和适用于在 Windows 上使用 AWS CLI 凭证助手进行 HTTPS 连接主题进行了少量更新。	2016 年 6 月 22 日
主题更新	更新了产品和服务集成以及“访问权限参考”主题，纳入有关与 CodePipeline 集成的信息。	2016 年 4 月 18 日

更改	描述	更改日期
新主题	增加了管理存储库触发器部分。新主题包括有关如何创建、编辑和删除触发器的示例，包括策略和代码示例。	2016 年 3 月 7 日
新主题	增加了产品和服务集成主题。对故障排除进行了少量更新。	2016 年 3 月 7 日
主题更新	除 MD5 服务器指纹外，适用于 Linux、macOS 或 Unix 上的 SSH 连接和适用于 Windows 上的 SSH 连接中增加了适用于 CodeCommit 的 SHA256 服务器指纹。	2015 年 12 月 9 日
新主题	增加了浏览存储库中的文件主题。故障排除中增加了新问题。对整个用户指南进行了少量改进和修复。	2015 年 10 月 5 日
新主题	增加了适用于不使用 AWS CLI 的 SSH 用户主题。简化了设置 部分中的主题。提供了可帮助用户确定对其操作系统采取的步骤的指导，并提供了首选协议。	2015 年 8 月 5 日
主题更新	在 SSH 和 Linux、macOS 或 Unix：为 Git 和 CodeCommit 设置公有密钥和私有密钥和 步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥中的 SSH 密钥 ID 步骤中增加了说明和示例。	2015 年 7 月 24 日
主题更新	更新了步骤 3：为 Git 和 CodeCommit 设置公有密钥和私有密钥中的步骤，以解决 IAM 和保存公有密钥文件的问题。	2015 年 7 月 22 日
主题更新	更新了故障排除，提供了导航帮助。增加了更多凭证密钥链问题的故障排除信息。	2015 年 7 月 20 日
主题更新	AWS KMS 和加密和访问权限参考主题中增加了有关 AWS Key Management Service 权限的更多信息。	2015 年 7 月 17 日
主题更新	故障排除中增加了一个部分，提供有关 AWS Key Management Service 问题的故障排除信息。	2015 年 7 月 10 日
初始版本	这是 CodeCommit 用户指南的初始版本。	2015 年 7 月 9 日

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》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。