使用者指南

AWS CodeCommit

API 版本 2015-04-13

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodeCommit: 使用者指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任從何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS CodeCommit？

AWS CodeCommit是一種由 Amazon Web Services 託管的版本控制服務，可讓您在雲端存放和管理私有資產 (例如文件、來源碼和二進位檔案)。如需 CodeCommit 定價的相關資訊，請參閱定價。

> **Note**
>
> CodeCommit 適用於許多合規計劃。如需 AWS 和合規性工作的詳細資訊，請參閱合規計劃的 AWS 服務範圍。
> 此為 HIPAA 合格服務。如需 AWS、1996 美國健康保險流通與責任法案 (HIPAA) 與使用 AWS 服務處理、存放和傳輸受保護健康資訊 (PHI) 的詳細資訊，請參閱 HIPAA 概觀。
> 如需有關此服務和 ISO 27001 (指定安全性管理最佳做法的安全性管理標準) 的詳細資訊，請參閱 ISO 27001 概觀。
> 如需有關此服務和支付卡產業資料安全標準 (PCI DSS) 的詳細資訊，請參閱 PCI DSS 概觀。
> 如需此服務的詳細資訊，以及具體說明可保護敏感資訊密碼模組之安全請求的聯邦資訊處理標準 (FIPS) 第 140-2 號公報美國政府標準，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀和Git 連接端點。

主題

CodeCommit 簡介

CodeCommit是一項安全、高度可擴展、可受管私有的 Git 儲存庫。CodeCommit您無需管理自己的原始檔控制系統，也不必擔心擴充其基礎架構。從程式碼到二進位檔，CodeCommit 都可以儲存。它支援 Git 的標準功能，可以完美搭配以 Git 為基礎的現有工具一起運作。

CodeCommit 可讓您：

- 受益於託管的完全託管服務AWS。CodeCommit提供高服務可用性和耐久性，並消除管理自己的硬體和軟體所產生的管理負擔。無需佈建和擴充硬體，也無需安裝、設定和更新伺服器軟體。
- 安全地儲存您的程式碼。CodeCommit儲存庫在靜態和傳輸過程中都會加密。
- 在程式碼上協同工作。CodeCommit儲存庫支援提取要求，使用者可以在將其合併到分支之前，先檢視彼此的程式碼變更並加上註解；自動傳送有關提取要求和註解的電子郵件的通知；以及更多功能。
- 輕鬆擴展您的版本控制項目。CodeCommit儲存庫可以擴充以滿足您的開發需求。此服務可處理具有大量檔案或分支、大型檔案及冗長修訂歷史記錄的儲存庫。
- 隨時存儲任何東西。CodeCommit對儲存庫的大小沒有任何限制，以及可存放的檔案類型沒有任何限制。
- 與其他AWS和第三方服務集成。CodeCommit讓您的儲存庫靠近AWS雲端中的其他生產資源，這有助於提高開發生命週期的速度和頻率。它與 IAM 集成，可以與其他AWS服務一起使用，並與其他存儲庫 parallel 使用。如需詳細資訊，請參閱產品與服務整合 AWS CodeCommit。
- 輕鬆從其他遠端儲存庫遷移檔案。您可以從以 Git 為基礎的任何儲存庫遷移至 CodeCommit。
- 使用您已經知道的 Git 工具。CodeCommit支持 Git 命令以及它自己的AWS CLI命令和 API。

CodeCommit，Git，並選擇適合您需求的AWS服務

做為 Git 型服務，CodeCommit 非常適合大部分版本控制的需求。沒有檔案大小、檔案類型和儲存庫大小方面的任意限制。不過，Git 存在固有的侷限性，可能對某些操作種類的效能造成負面影響，特別是在經過一段時間之後。有其他 AWS 服務更適合該任務時，避免將其用於這種使用案例，可以避免 CodeCommit 儲存庫的潛在效能下降。您也可以針對複雜儲存庫來最佳化 Git 的效能。以下提供一些使用案例，在其中 Git 以及 CodeCommit 可能不是最適合您的解決方案，或者您可能需要採取其他步驟來最佳化 Git。

使用案例	描述	可考慮的其他服務
經常變更的大型檔案	Git 使用 Delta 編碼來存放檔案版本之間的差異。例如，如果您變更文件中的幾個字詞，Git 只會存放這些變更過的字詞。如果您有超過 5 MB 的檔案或物件且具有許多變更，Git 可能需要重建大型的 Delta 差異鏈。當這些檔案隨著時間增長，這樣會消耗本機電腦和	若為大檔案版本，請考慮 Simple Storage Service (Amazon S3)。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用指南中的使用版本控。

使用案例	描述	可考慮的其他服務
	CodeCommit 中越來越多的運算資源。	
資料庫	Git 儲存庫會隨著時間增長。由於版本控制會追蹤所有變更，任何變更都會增加儲存庫的大小的。換言之，當您遞交資料，即使您刪除遞交中的資料，資料仍會新增到儲存庫。隨著時間需要處理和傳輸的資料越來越多，Git 就會變慢。這對資料庫的使用案例尤其不利。Git 並非專為資料庫而設計。	若要建立和使用具有一致效能的資料庫，不論大小為何，請考慮使用 Amazon DynamoDB。如需詳細資訊，請參閱《Amazon DynamoDB 資料庫入門指南》。
稽核線索	一般而言，稽核線索會保留很長一段時間，並由系統程序以非常頻繁的頻率連續產生。Git 的設計目的是為了安全地存放開發人員群組在開發週期產生的原始程式碼。不斷存放以編寫程式方式產生之系統變更的快速變化儲存庫，在一段時間後效能會下降。	若要存放稽核追蹤，請考慮 Amazon Storage Service (Amazon S3)。 若要稽核AWS活動，視您的使用案例而定，請考慮使用 AWS CloudTrailAWS Config, 或 AmazonCloudWatch。
備份	Git 的設計目的是為了對開發人員編寫的原始程式碼進行版本控制。您可以推送遞交到兩個遠端儲存庫 (包括 CodeCommit 儲存庫) 做為備份策略。不過，Git 不是專為處理電腦檔案系統、資料庫傾印或類似備份內容的備份而設計的。這樣做可能會降低系統速度，並提高複製和推送儲存庫所需的時間。	如需有關備份到 AWS 雲端的資訊，請參閱備份和還原。

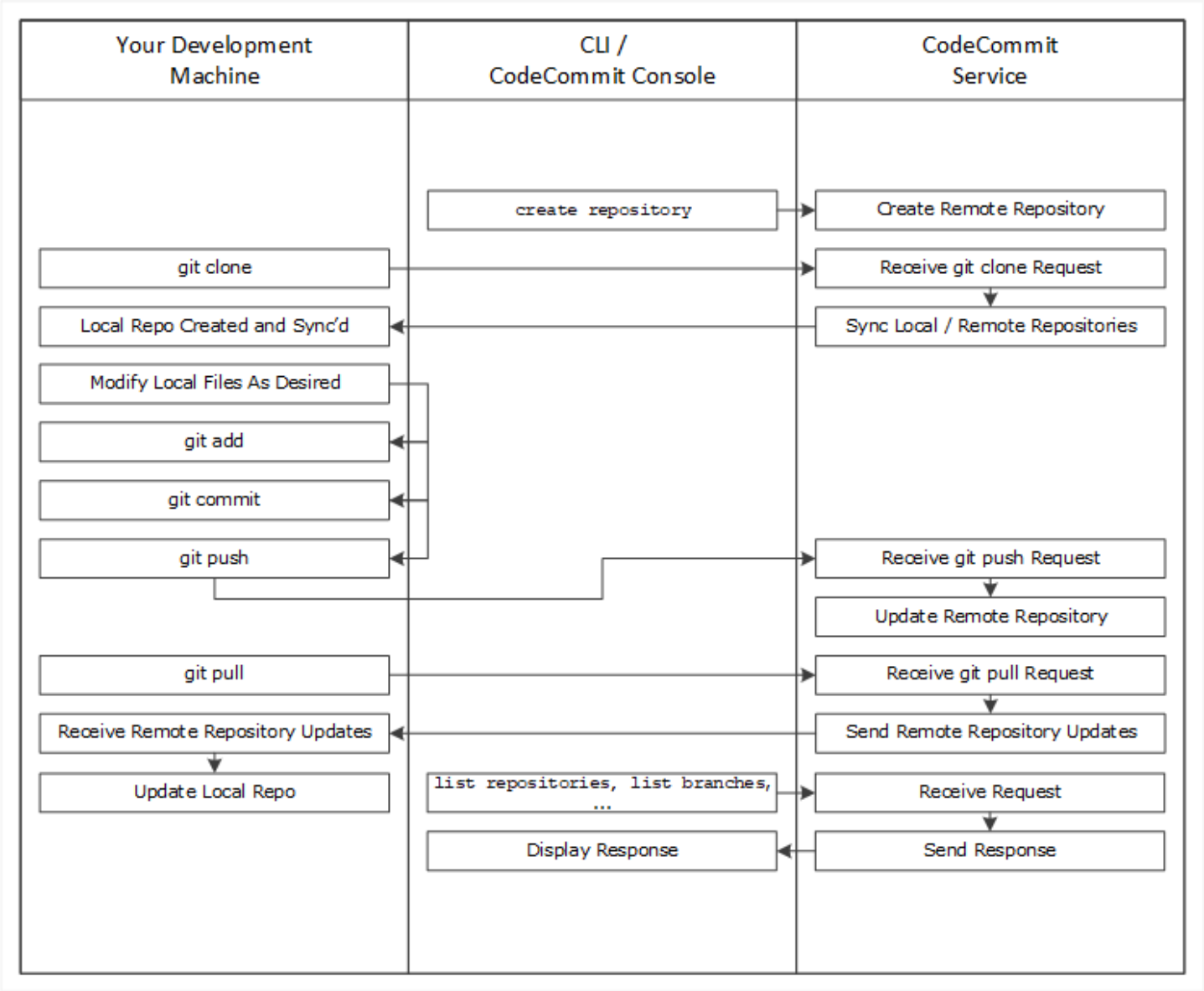
使用案例	描述	可考慮的其他服務

大量的分支或參考	當 Git 用戶端推送或提取儲存庫資料時，即使您只需要一個分支，遠端伺服器仍必須傳送所有分支和參考 (例如標籤)。如果您擁有數千個分支和參考，這可能需要一些時間來處理和傳送 (封包協議) 並造成明顯緩慢的儲存庫回應。您擁有越多分支和標籤，這個處理程序就需要越長的時間。我們建議您使用 CodeCommit，但刪除不再需要的分支和標籤。	若要分析 CodeCommit 儲存庫中的參考數量，以判斷不再需要哪些項目，您可以使用下列其中一個命令：
		• Linux、macOS 或 Unix：
		`git ls-remote | wc -l`
		• Powershell：
		`git ls-remote | Measure-Object -line`

CodeCommit 如何運作？

CodeCommit對於基於 GIT 的存儲庫的用戶來說很熟悉，但即使是那些不熟悉的用戶也應該找到CodeCommit相對簡單的過渡。CodeCommit提供了一個控制台，用於輕鬆創建存儲庫以及現有存儲庫和分支的列表。只需要幾個簡單的步驟，使用者就可以找到儲存庫的相關資訊並複製到電腦，建立本機儲存庫來進行變更，然後將變更推送到 CodeCommit 儲存庫。使用者可以在本機機器上從命令列工作，或使用以 GUI 為基礎的編輯器。

下圖顯示如何使用您的開發機器、AWS CLI 或 CodeCommit 主控台及 CodeCommit 服務，以建立和管理儲存庫：



1. 使用AWS CLI或主CodeCommit控台建立CodeCommit存放庫。
2. 從您的開發電腦上，使用 Git 執行 git clone，並指定 CodeCommit 儲存庫的名稱。這將創建一個連接到CodeCommit存儲庫的本地回購。
3. 使用開發機器上的本機存放庫修改 (新增、編輯和刪除) 檔案，然後執行以在本git add機暫存修改過的檔案。執行git commit以在本機提交檔案，然後執行git push以將檔案傳送至CodeCommit儲存庫。
4. 下載其他使用者所做的變更。執行git pull以將儲存庫中的檔案與您的本機CodeCommit存放庫同步。這可確保您使用的是最新版本的檔案。

您可以使用 AWS CLI 或 CodeCommit 主控台來追蹤和管理您的儲存庫。

與 Amazon S3 中的文件版本控制有何CodeCommit不同？

CodeCommit 針對團隊軟體開發而最佳化。它管理跨多個文件的批次更改，這些更改可以與其他開發人員所做的更改 parallel 發生。Amazon S3 版本控制支援復原舊版檔案，但並不著重於軟體開發團隊所需的協作檔案追蹤功能。

如何開始使用 CodeCommit？

開始使用 CodeCommit：

1. 遵循設定 中的步驟來準備您的開發機器。
2. 遵循入門中一或多個教學課程的步驟。
3. 在中建立版本控制專案，CodeCommit或將版本控制專案移轉至CodeCommit.

我可以在哪裡進一步了解 Git？

如果您沒聽過 Git，請了解如何使用 Git。以下是一些有用的資源：

- Pro Git，這是線上版本的 Pro Git 書籍。作者為 Scott Chacon。由 Apress 出版。
- Git 沉浸式，一個try-it-yourself導遊，引導你通過使用 Git 的基礎知識。由 Neo Innovation, Inc. 出版。
- Git 參考，這是線上快速參考，也可當做更深入的 Git 教學課程。由GitHub團隊出版。
- Git 速查表，含 Git 命令基本語法。由GitHub團隊出版。
- Git 隨身指南。作者為 Richard E. Silverman。由 O'Reilly Media, Inc. 出版。

設定 AWS CodeCommit

您可以登入到AWS Management Console和上傳、添加或編輯文件直接從AWS CodeCommit主控台。這是快速進行變更的方式。不過，如果您想要處理多個檔案、跨分支的檔案等等，請考慮設定本機電腦來使用儲存庫。設定 CodeCommit 的最簡單方法就是設定 HTTPS Git 登入資料來設定AWS CodeCommit。此 HTTPS 身份驗證方法：

- 使用靜態使用者名稱和密碼。
- 適用於 CodeCommit 支援的所有作業系統。
- 也相容於整合開發環境 (IDE) 和其他支援 Git 登入資料的開發工具。

如果您基於操作理由而不想要或無法使用 Git 登入資料，則可以使用其他方法。舉例來說，如果您使用聯合存取、暫時性登入資料或 Web 身分供應商來存取 CodeCommit 儲存庫，則無法使用 Git 登入資料。建議您使用 `git-remote-codecommit` 命令設定本機電腦。請仔細檢閱這些選項，決定最適合您的替代方法。

- 使用 Git 憑據進行設置
- 使用其他方法進行設置
- CodeCommit、Git 和其他元件的相容性

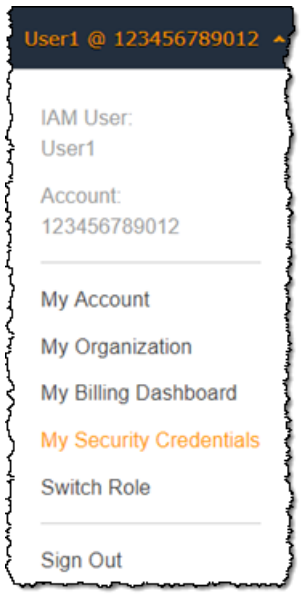
如需如何使用 CodeCommit 和 Amazon Virtual Private Cloud 的詳細信息，請參閱AWS CodeCommit與介面 VPC 端點搭配使用。

查看和管理您的憑據

您 CodeCommit 透過AWS控制台通過我的安全登入資料。

> **Note**
>
> 使用聯合存取、暫時性登入資料或 Web 身分供應商的使用者不可用。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 My Security Credentials (我的安全憑證)。



3. 選擇AWS CodeCommit證書索引標籤。

使用 Git 憑據進行設置

使用 HTTPS 連線和 Git 登入資料，您可以在 IAM 中產生靜態的使用者名稱和密碼。然後，您會將這些登入資料用於 Git，以及任何支援 Git 使用者名稱和密碼身份驗證的第三方工具。大部分 IDE 和開發工具都支援此方法。這是與 CodeCommit 一起使用的最簡單又最輕鬆的連線方法。

- 適用於使用 Git 認證的 HTTPS 使用者：依照這些指示，使用 Git 登入資料來設定本機電腦和 CodeCommit 儲存庫之間的連線。
- 用於來自開發工具的連接：依照這些準則，使用 Git 登入資料來設定 IDE 或其他開發工具與 CodeCommit 儲存庫之間的連線。支援 Git 登入資料的 IDE 包括 (但不限於) Visual Studio、Eclipse、Xcode 和 IntelliJ。

使用其他方法進行設置

您可以使用 SSH 協議而不是 HTTPS 連線到您的 CodeCommit 儲存庫。使用 SSH 連接，您可以在本機電腦上建立 Git 和 CodeCommit 用於 SSH 身分驗證的公有和私有金鑰檔案。您需要將公有金鑰與您的 IAM 使用者相關聯。您需要將私有金鑰存放在本機電腦。由於 SSH 需要手動建立和管理公有和私有金鑰文件，您可能會發現 Git 登入資料與 CodeCommit 一起使用時很簡單又輕鬆。

與 Git 登入資料不同，根據本機電腦的作業系統而定，SSH 連線設定會有所不同。

- 對於 SSH 用户不使用AWS CLI：如果您已具備公有私有金鑰對，且熟悉在本機電腦上使用 SSH 連線，請遵循這些簡短指示。
- Linux：依照這些指示，在 Linux、macOS 或 Unix 作業系統上逐步建立公有私有金 key pair 和設定連線。
- 適用於 Windows 上的 SSH 連線：依照這些指示，在 Windows 作業系統上逐步建立公有私有金鑰對和設定連線。

如果您正在連接到 CodeCommit 並AWS登入資料、身分供應商或暫時性登入資料，或如果您不想要設定 IAM 使用者或 IAM 使用者的 Git 登入資料，您可以透過下列兩種方式的其中一種來設定 CodeCommit 儲存庫連線：

- 安裝並使用 git-remote-codecommit (建議使用)。
- 安裝並使用 AWS CLI 隨附的登入資料協助程式。

這兩種方法都支援存取 CodeCommit 儲存庫，而不需要 IAM 使用者，這意味着您可以使用聯合存取和暫時性登入資料來連線到儲存庫。建議使用的方法是 git-remote-codecommit 公用程式。它延伸了 Git，而且可與各種 Git 版本和登入資料協助程式相容。不過，並非所有 IDE 均可支援 `git-remote-codecommit` 使用的複製 URL 格式。您可能必須先將儲存庫手動複製到本機電腦，才能在 IDE 中使用這些儲存庫。

- 請遵循HTTPS 連線的設定步驟AWS CodeCommit具有 git-remote-codecommit 的儲存庫安裝與設定git-remote-codecommit在 Windows、Linux、macOS 或 Unix。

隨附的登入資料協助程式可讓您使用AWS CLI每當 Git 需要向驗證時，您可以使用 HTTPS 和密碼編譯簽章版本的 IAM 使用者憑證或 Amazon EC2 執行個體角色。AWS來與 CodeCommit 存儲庫進行交互。有些作業系統和 Git 版本有自己的登入資料協助程式，但會與 AWS CLI 包含的登入資料協助程式發生衝突。結果導致 CodeCommit 的連線問題。

- 對於在 Linux、蘋果系統或 Unix 上使用 HTTPS 連接AWS CLI憑證助手：依照這些指示，在 Linux、macOS 或 Unix 系統上逐步安裝和設定登入資料協助程式。
- 對於在視窗上使用 HTTPS 連線AWS CLI憑證助手：依照這些指示，在 Windows 系統上逐步安裝和設定登入資料協助程式。

如果您使用另一個 Amazon Web Services 帳戶中託管的 CodeCommit 儲存庫，您可以使用角色、策略和隨附的登入資料協助程式來設定存取和設定連線。AWS CLI。

- 使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取：依照這些指示，在另一個 Amazon Web Services 賬戶中逐步設定 IAM 組中的使用者。

CodeCommit、Git 和其他元件的相容性

使用 CodeCommit 時，您會使用 Git。您也可以使用其他程式。下表提供關於版本相容性的最新指導。根據最佳實務，我們建議您使用 Git 和其他軟體的最新版本。

版本兼容性信息AWS CodeCommit

元件	版本
Git	CodeCommit 支援 Git 1.7.9 版和更新版本。Git 2.28 版支持為初始提交配置分支名稱。我們建議您使用最新版本的 Git。
Curl	CodeCommit 需要捲曲 7.33 及更高版本。不過，HTTPS 和 curl 更新 7.41.0 有一個已知問題。如需詳細資訊，請參閱 疑難排解。
Python (僅限 git-remote-codecommit)	git-remote-codecommit 需要第 3 版和更新版本。
Pip (僅限 git-remote-codecommit)	git-remote-codecommit 需要 9.0.3 版和更新版本。
AWS CLI (僅限 git-remote-codecommit)	我們建議您使用最新版本的AWS CLI版本 2 適用於所有 CodeCommit 用户。git-remote-codecommit需要AWS CLI版本 2 來支援 AWSSSO 和需要臨時證書的連接，例如聯合用户。

使用 Git 認證的 HTTPS 使用者進行設定

設定 AWS CodeCommit 儲存庫連線的最簡單方法是在 IAM 主控台 CodeCommit 中設定 Git 認證，然後將這些認證用於 HTTPS 連線。您也可以將這些相同的認證與任何協力廠商工具或整合式開發環境 (IDE) 搭配使用，使用靜態使用者名稱和密碼來支援 HTTPS 驗證。如需範例，請參閱 用於來自開發工具的連接。

> **Note**
>
> 如果您先前已將本機電腦設定為使用認證協助程式 CodeCommit，您必須先編輯 .gitconfig 檔案，以移除檔案中的認證協助程式資訊，才能使用 Git 認證。如果您的本機電腦執行 macOS，您可能需要從「鑰匙圈存取」清除快取的認證。

步驟 1：初始配置 CodeCommit

請依照下列步驟設定 Amazon Web Services 帳戶、建立 IAM 使用者，以及設定存取權限 CodeCommit。

若要建立和設定 IAM 使用者以存取 CodeCommit

1. 通過轉到 http://aws.amazon.com 並選擇註冊創建一個 Amazon Web Services 帳戶。
2. 在您的 Amazon Web Services 帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的 Amazon Web Services 帳戶中建立 IAM 使用者。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕 AWS KMS 執行所需的動作 CodeCommit。如需詳細資訊，請參閱 AWS KMS和加密。

3. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
4. 在 IAM 主控台的導覽窗格中，選擇 [使用者]，然後選擇要設定用於 CodeCommit 存取的 IAM 使用者。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。

6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從策略清單中，選取AWSCodeCommitPowerUser或另一個受管理的策略以進行 CodeCommit 存取。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的政策後，請選擇 [下一步：檢閱] 以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策以及與其他群組和使用者共用存放庫存取權的詳細資訊，請參閱共用儲存庫和AWS CodeCommit 的身分驗證與存取控制。

如果您想要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 建議您建立使用 AWS CLI 與的設定檔 CodeCommit。若要取得更多資訊，請參閱〈命令列參考使用具名紀要〉。

步驟 2：安裝 Git

若要處理 CodeCommit 儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

若要安裝 Git，我們建議使用 Git 下載等網站。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其運作方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱疑難排解.

第 3 步：為 HTTPS 連接創建 Git 憑據 CodeCommit

安裝 Git 之後，請在 IAM 中為您的 IAM 使用者建立 Git 登入資料。

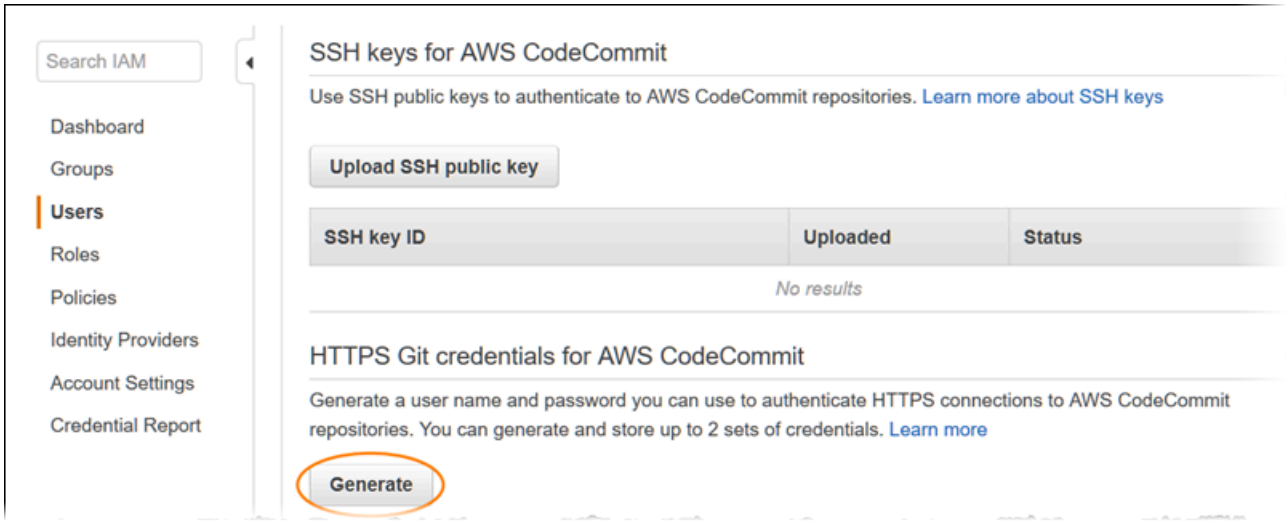
若要設定以下項目的認證 CodeCommit

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。

   請務必以 IAM 使用者身分登入，該使用者將建立並使用 Git 認證進行連線 CodeCommit。
2. 在 IAM 主控台的導覽窗格中，選擇 [使用者]，然後從使用者清單中選擇您的 IAM 使用者。

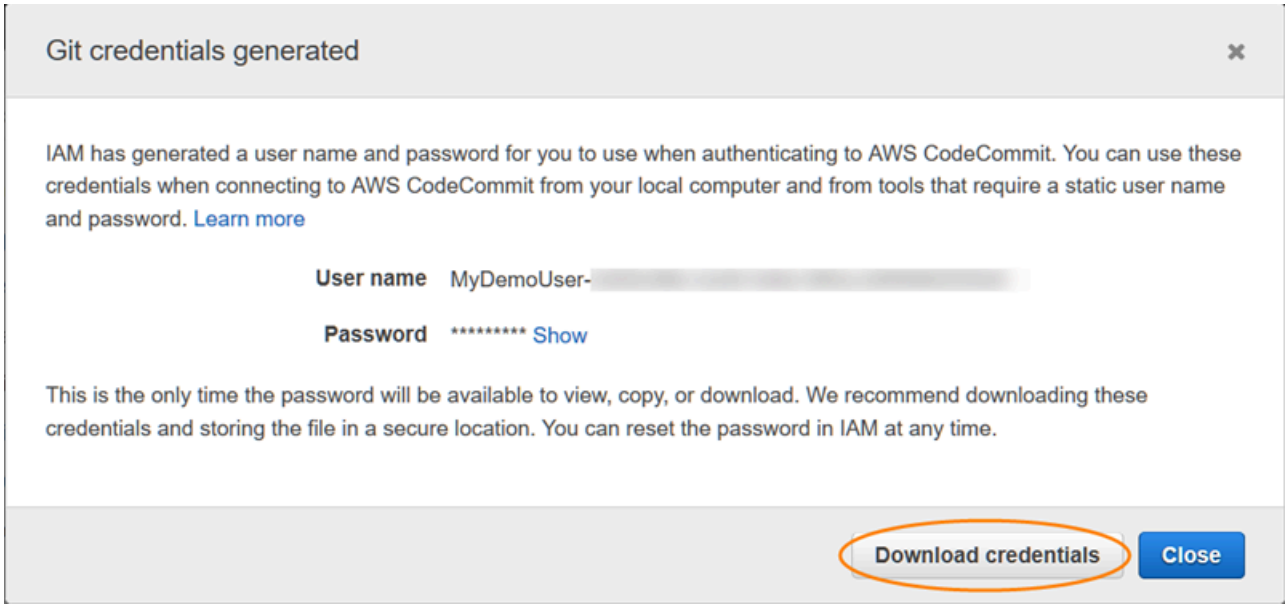
> **Note**
>
> 您可以在「我的安全 CodeCommit 登入資料」中直接檢視和管理您的認證。如需詳細資訊，請參閱 查看和管理您的憑據。

3. 在使用者詳細資料頁面上，選擇 [安全認證] 索引標籤，然後在 [HTTPS Git 認證] 中選擇 [產生]。AWS CodeCommit



> **Note**
>
> 您無法為 Git 憑證選擇自己的使用者名稱或密碼。如需詳細資訊，請參閱搭配使用 Git 認證和 HTTPS CodeCommit。

4. 複製 IAM 為您產生的使用者名稱和密碼，方法是顯示、複製這些資訊，然後貼到本機電腦上的安全檔案中，或選擇 [下載認證] 將此資訊下載為 .CSV 檔案。您需要此資訊才能連線到 CodeCommit。



儲存您的登入資料之後，選擇 Close (關閉)。

> **Important**
>
> 這是您儲存使用者名稱和密碼的唯一機會。如果未儲存它們，可以從 IAM 主控台複製使用者名稱，但無法查詢密碼。您必須重設密碼，然後儲存密碼。

步驟 4：Connect 到 CodeCommit 控制台並克隆存儲庫

如果管理員已經將 CodeCommit儲存區域的名稱和連線詳細資訊傳送給您，您可以略過此步驟並直接複製儲存區域。

連線至 CodeCommit 存放庫的步驟

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選取器中，選擇建立儲存庫的 AWS 區域 位置。儲存庫特定於 AWS 區域. 如需詳細資訊，請參閱 區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要搭配 IAM 使用者使用 Git 登入資料，或是 AWS CLI.
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。

- 如果您將 SSH 公開/私密 key pair 與 IAM 使用者搭配使用，請複製 SSH URL。

> **Note**
>
> 如果您看到「歡迎」頁面而非儲存庫清單，表示您登入的 AWS 區域 地方沒有與您的 AWS 帳戶相關聯的儲存庫。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 開啟終端機、命令列或 Git shell。使用您複製的 HTTPS 複製 URL，執行 git clone 命令來複製儲存庫。例如，若要複製名*MyDemoRepo*為美國東部 (俄亥俄) 區域名稱*my-demo-repo*的本機存放庫：

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

第一次連接時，系統會提示您提供儲存庫的使用者名稱和密碼。視您本機電腦的組態而定，此提示可能來自作業系統的認證管理系統、Git 版本的認證管理員公用程式 (例如 Windows 版 Git 中包含的 Git 認證管理員)、您的 IDE 或 Git 本身。輸入 IAM 中為 Git 憑據生成的用戶名和密碼（您在中創建的用戶名第 3 步：為 HTTPS 連接創建 Git 憑據 CodeCommit）。根據您的作業系統和其他軟體而定，此資訊可能儲存在登入資料存放區或登入資料管理公用程式中。如果是這樣，除非您變更密碼、停用 Git 認證或刪除 IAM 中的 Git 認證，否則不應再次收到提示。

如果本機電腦上沒有設定登入資料存放區或登入資料管理公用程式，您可以安裝此工具。如需 Git 和如何管理登入資料的詳細資訊，請參閱 Git 文件中的登入資料儲存。

如需詳細資訊，請參閱 透過複製存 CodeCommit 放庫連 Connect 至儲存庫 及 創建一個提交。

後續步驟

您已完成事前準備。請按照中的步驟開始使用 CodeCommit 開始使用 CodeCommit。

要了解如何創建和推送第一次提交，請參閱在中創建提交 AWS CodeCommit。如果您是初次使用 Git，您可能還需要檢閱我可以在哪裡進一步了解 Git？和開始使用 Git 和 AWS CodeCommit中的資訊。

HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit

如果您想要使用根帳戶、聯合存取或暫時性登入資料連線到 CodeCommit，則應使用 git-remote-codecommit 設定存取。此公用程式提供透過擴充 Git 從 CodeCommit 儲存庫推送和提取程式碼的簡單方法。這是支援使用聯合存取、身分識別提供者和臨時登入資料建立的連線的建議方法。若要將權限指派給同盟身分識別，您可以建立角色並定義角色的權限。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。

您也可以使用git-remote-codecommit與 IAM 使用者一起使用。git-remote-codecommit 與其他 HTTPS 連線方法不同，不需要為使用者設定 Git 登入資料。

> **Note**
>
> 某些 IDE 不支援 `git-remote-codecommit` 使用的複製 URL 格式。您可能必須先將儲存庫手動複製到本機電腦，才能在偏好使用的 IDE 中使用這些儲存庫。如需詳細資訊，請參閱針對 git-remote-codecommit 和 AWS CodeCommit 進行故障診斷。

這些程序的撰寫方式是假設您擁有 Amazon Web 服務帳戶，並在中建立至少一個儲存庫 CodeCommit，並在連線到時使用 IAM 使用者搭配受管政策CodeCommit儲存庫。如需如何為聯合身分使用者設定存取和其他輪換登入資料類型的相關資訊，請參閱使用旋轉認證連線至AWS CodeCommit儲存庫。

主題

步驟 0：安裝先決條件git-remote-codecommit

您必須先在本機電腦上安裝某些必要項目，才能使用 git-remote-codecommit。其中包含：

- Python (第 3 版或更新版本) 及其套件管理工具 PIP (若未安裝)。若要下載並安裝最新版 Python，請造訪 Python 網站。
- Git

> **Note**
>
> 當您在 Windows 上安裝 Python 時，請務必選擇此選項以將 Python 新增到路徑中。

git-remote-codecommit 需要 PIP 9.0.3 版或更新版本。若要檢查您的 PIP 版本，請開啟終端機或命令列，然後執行下列命令：

```
pip --version
```

您可以執行下列兩個命令，將 PIP 版本更新為最新版：

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

若要使用中的檔案、認可和其他資訊CodeCommit存儲庫中，您必須在本地計算機上安裝 Git。CodeCommit 支援 Git 1.7.9 版和更新版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

要安裝 Git，我們建議網站如Git 下載。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時，一項功能的改變，可能會影響此功能與 CodeCommit 運作的方式。如果您在特定版本的 Git 和 CodeCommit 方面遇到問題，請檢閱疑難排解中的資訊。

步驟 1：初始配置CodeCommit

依照下列步驟建立 IAM 使用者、使用適當的政策進行設定、取得存取金鑰和秘密金鑰，以及安裝和設定AWS CLI。

若要建立和設定 IAM 使用者以存取CodeCommit

1. 通過轉到創建一個亞馬遜網絡服務帳戶http://aws.amazon.com並選擇立即註冊。
2. 在您的亞馬遜網路服務帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的亞馬遜網絡服務帳戶中創建 IAM 用戶。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS所需的動作CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇使用者，然後選擇您要設定的 IAM 使用者CodeCommit訪問。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從政策清單中，選取 AWSCodeCommitPowerUser，或另一個用於存取 CodeCommit 的受管政策。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的策略之後，請選擇下一個:評論以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策及分享儲存庫存取權給其他群組和使用者的詳細資訊，請參閱共用儲存庫和 AWS CodeCommit 的身分驗證與存取控制。

安裝及設定 AWS CLI

1. 在本機電腦上，下載並安裝 AWS CLI。這是從命令列與 CodeCommit 互動的必要步驟。我們建議您安裝AWS CLI版本 2. 它是最新的主要版本AWS CLI並支持所有最新功能。這是唯一的版本AWS CLI支援使用 root 帳戶、同盟存取或臨時登入資料git-remote-codecommit。

如需詳細資訊，請參閱開始設定 AWS 命令列界面。

> **Note**
>
> CodeCommit僅適用於AWS CLI版本 1.7.38 及更高版本。根據最佳實務，安裝 AWS CLI 或將其升級到可用的最新版本。若要判斷您已安裝的 AWS CLI 版本，請執行 aws --version 命令。
> 若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 執行此命令以驗證CodeCommit的指令AWS CLI已安裝。

```
aws codecommit help
```

這個命令返回一個列表CodeCommit命令。

3. 配置AWS CLI使用設定檔configure命令，如下所示：。

```
aws configure
```

出現提示時，請指定AWS存取金鑰和AWS要搭配使用的 IAM 使用者的秘密存取金鑰CodeCommit。此外，請務必指定AWS 區域存放庫所在的位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

```
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
```

如需建立和設定與 AWS CLI 搭配使用之描述檔的詳細資訊，請參閱下列內容：

- 命名設定檔
- 在中使用 IAM 角色AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源AWS 區域，您必須重新設定AWS CLI使用默認的地區名稱。CodeCommit 支援的預設區域名稱包括：

- 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

如需 CodeCommit 和 AWS 區域 的詳細資訊，請參閱「區域和 Git 連線端點」。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得認證？和管理 IAM 使用者的存取金鑰。如需 AWS CLI 和描述檔的詳細資訊，請參閱具名描述檔。

步驟 2：安裝git-remote-codecommit

請按照下列步驟安裝 git-remote-codecommit。

安裝 git-remote-codecommit

1. 在終端機或命令列，執行下列命令：

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

> **Note**
>
> 視您的作業系統和組態而定，您可能需要以提升的權限 (例如 sudo) 執行此命令，或使用-user 參數安裝到不需要特殊權限的目錄，例如您目前的使用者帳戶。例如，在執行 Linux、macOS 或 Unix 的電腦上：
>
> ```
> sudo pip install git-remote-codecommit
> ```
>
> 在執行視窗的電腦上：
>
> ```
> pip install --user git-remote-codecommit
> ```

2. 監視安裝程序，直到您看到成功訊息為止。

步驟 3：連接到CodeCommit控制台並克隆存儲庫

如果管理員已將可搭配 git-remote-codecommit 用於 CodeCommit 儲存庫的複製 URL 傳送給您，您可以直接跳過連線到主控台及複製儲存庫的程序。

連接到 CodeCommit 儲存庫

1. 打開CodeCommit控制台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選擇器中，選擇AWS 區域存放庫的建立位置。存儲庫特定於AWS 區域。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要將 Git 認證與 IAM 使用者搭配使用，或是隨附的認證協助程式使用，請複製 HTTPS 網址AWS CLI。
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。
   - 如果您將 SSH 公開/私密金鑰配對與 IAM 使用者搭配使用，請複製 SSH URL。

   > **Note**
   >
   > 如果您看到歡迎頁面而不是存儲庫列表，沒有與您的相關聯的存儲庫AWS帳戶中的AWS區域您登入的位置。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 在終端機或命令提示視窗中，使用 git clone 命令複製儲存庫。使用 HTTPSgit-remote-codecommit您複製的網址和名稱AWS CLI設定檔，如果您建立了具名的設定檔。如果您沒有指定描述檔，此命令會假定預設描述檔。在您執行此命令的目錄中，將會在該目錄的子目錄中建立本機儲存庫。例如，若要將名為 *MyDemoRepo* 的儲存庫，複製到名為 *my-demo-repo* 的本機儲存庫：

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

   使用名為的設定檔複製相同的儲存庫*CodeCommitProfile*:

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

   若要複製不同的儲存庫AWS 區域比您的個人資料中配置的那個，包括AWS 區域名稱。例如：

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

後續步驟

您已完成事前準備。按照中的步驟操作開始使用 CodeCommit 開始使用CodeCommit。

要了解如何創建和推送第一次提交，請參閱在中創建提交 AWS CodeCommit。如果您是初次使用 Git，您可能還需要檢閱我可以在哪裡進一步了解 Git？和開始使用 Git 和 AWS CodeCommit中的資訊。

使用 Git 憑據從開發工具設置連接

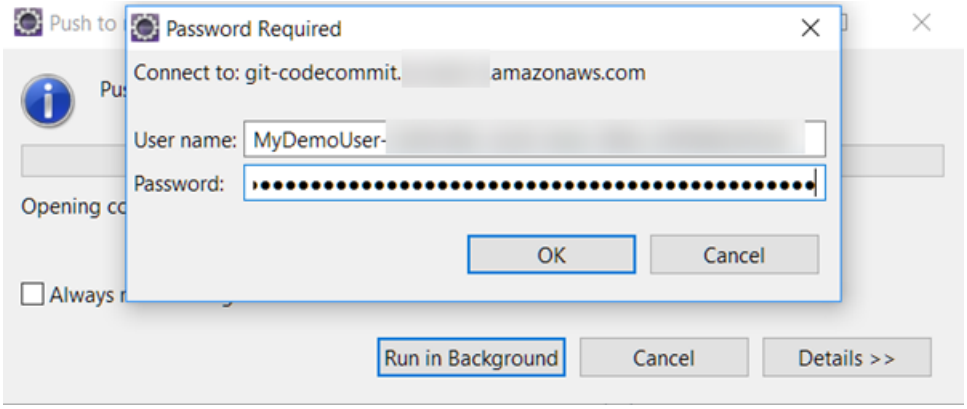
AWS CodeCommit 在 IAM 主控台中設定 Git 登入資料之後，您可以將這些認證與任何支援 Git 認證的開發工具搭配使用。例如，您可以在視覺工作室，日食，Xcode AWS Cloud9，IntelliJ 或任何集成 Git 憑據的集成開發環境（IDE）中配置對 CodeCommit存儲庫的訪問。在您設定存取權之後，您可以直接從 IDE 或其他開發工具中編輯程式碼、遞交變更及推送。

> **Note**
>
> 如果您使用聯合 CodeCommit 存取、臨時登入資料或 Web 相同性提供者存取儲存庫，則無法使用 Git 認證。建議您使用 `git-remote-codecommit` 命令設定本機電腦。不過，並非所有 IDE 皆可與 git-remote-codecommit 等 Git 遠端協助程式完全相容。如果您遇到問題，請參閱針對 git-remote-codecommit 和 AWS CodeCommit 進行故障診斷。

主題

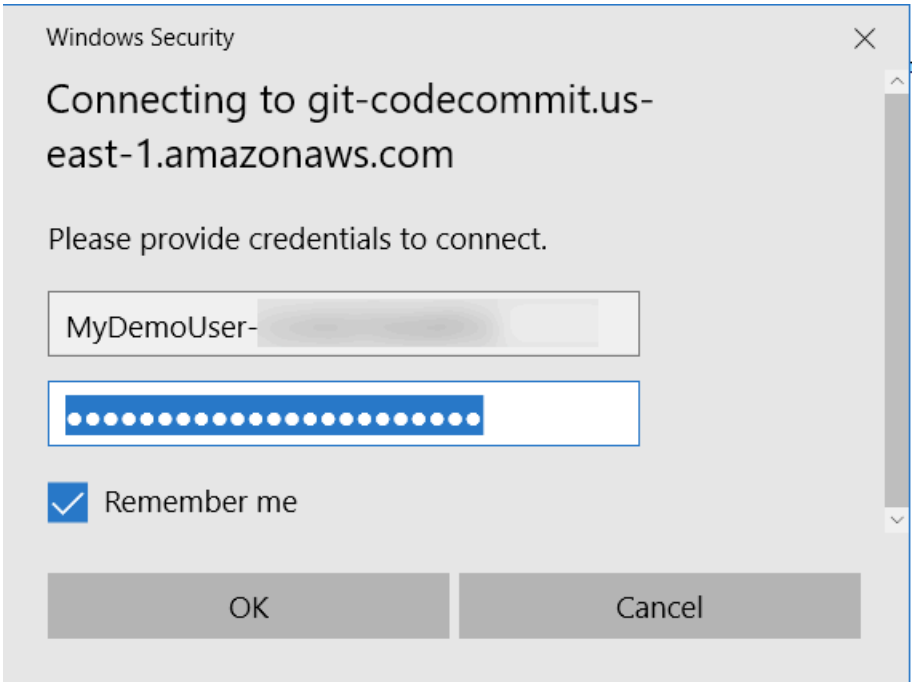
- 將 AWS Cloud9 與 AWS CodeCommit 整合
- 整合視覺工作室與AWS CodeCommit
- Eclipse 與 AWS CodeCommit 整合

當您的 IDE 或開發工具提示您輸入用於連線到 CodeCommit儲存庫的使用者名稱和密碼時，請為您在 IAM 中建立的使用者名稱和密碼提供 Git 認證。例如，如果 Eclipse 中提示您輸入使用者名稱和密碼，請如下所示提供 Git 登入資料：



如需有關 AWS 區域 和端點的詳細資訊 CodeCommit，請參閱區域和 Git 連線端點。

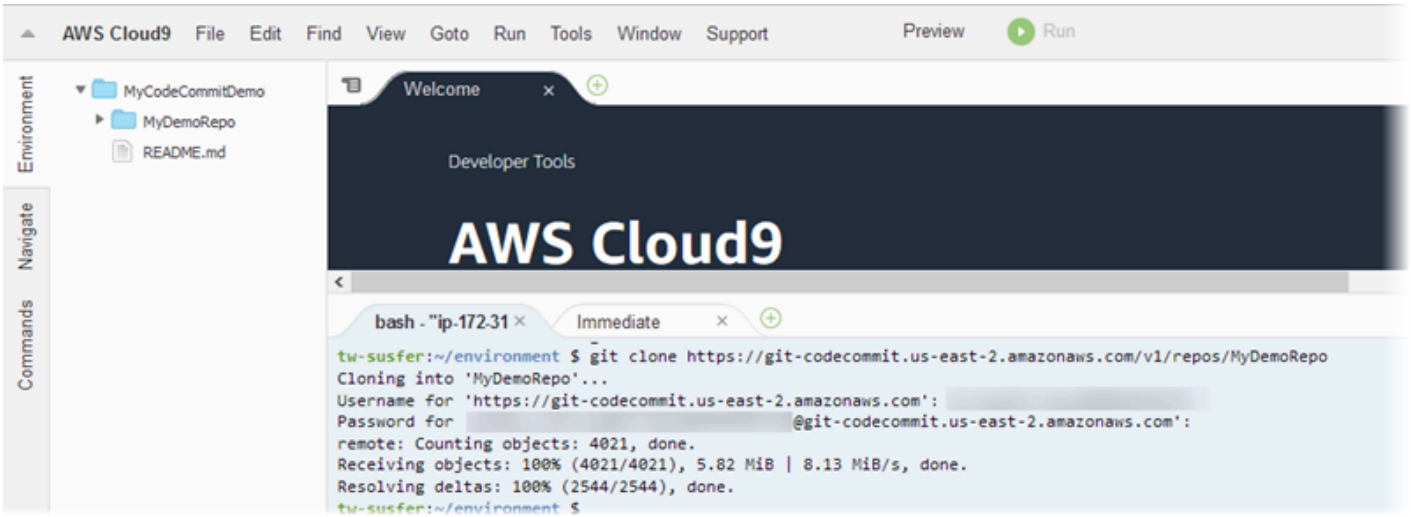
您的作業系統也可能提示您存放使用者名稱和密碼。例如，在 Windows 中，您需要如下所示提供 Git 登入資料：



如需有關為特定軟體程式或開發工具來設定 Git 登入資料的資訊，請參閱產品文件。

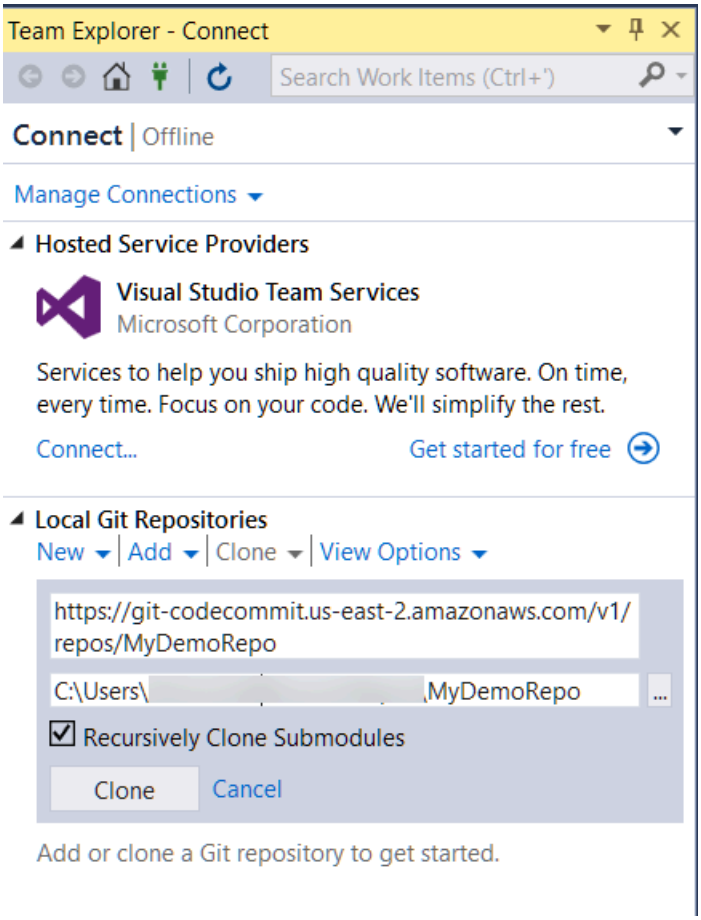
下面是不全面的 IDE 列表。這些鏈接僅用於幫助您了解有關這些工具的更多信息。 AWS 對任何這些主題的內容概不負責。

- AWS Cloud9



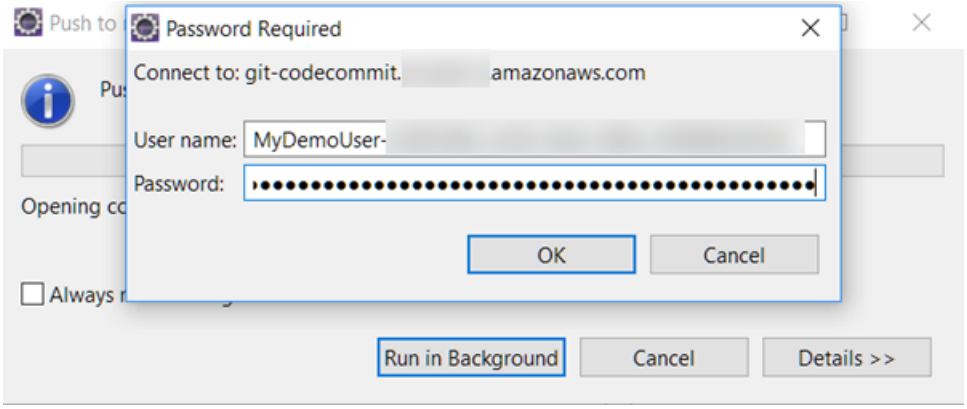
- Visual Studio

  或者，安裝 AWS Toolkit for Visual Studio. 如需詳細資訊，請參閱 整合視覺工作室與AWS CodeCommit。



- EGit with Eclipse

或者，安裝 AWS Toolkit for Eclipse. 如需詳細資訊，請參閱 Eclipse 與 AWS CodeCommit 整合。



- XCode

將 AWS Cloud9 與 AWS CodeCommit 整合

您可以使用AWS Cloud9以在 CodeCommit 儲存庫。AWS Cloud9包含用於編寫程式碼以及建置、執行、測試、偵錯以及發行軟體的工具集合。您可以複製現有的儲存庫、建立儲存庫、將程式碼變更遞交並推送至儲存庫等等，一切都在 AWS Cloud9 EC2 開發環境內完成。所以此AWS Cloud9EC2 開發環境通常已預先設定AWS CLI、Amazon EC2 角色以及 Git，所以在大多數情況下，您只要執行幾個簡單的命令，就能開始與儲存庫互動。

使用AWS Cloud9，您需要下列項目：

- Amazon Linux 上執行的 AWS Cloud9 EC2 開發環境。
- Web 瀏覽器中開啟的 AWS Cloud9 IDE 首頁。
- 具有其中一個 CodeCommit 託管策略和其中一個AWS Cloud9託管策略。

  如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策和了解和取得您的安全登入資料。

> **Note**
>
> 本主題介紹如何設置與 CodeCommit 和AWS Cloud9從網際網路普通存取。您可以設定 CodeCommit 和AWS Cloud9，但這需要額外的步驟。如需詳細資訊，請參閱：
>
> - AWS CodeCommit 與介面 VPC 端點搭配使用
> - 使用存取無輸入的 Amazon EC2 執行個體AWS Systems Manager
> - 使用共享環境
> - 與其他帳戶共享 VPC

- 部落格文章：隔離網路存取您的AWS Cloud9環境

主題

步驟 1：建立AWS Cloud9開發環境

AWS Cloud9會在 Amazon EC2 執行個體上託管您的開發環境。這是最簡單的集成方式，因為您可以使用AWS託管的實例臨時證書以連接到您的 CodeCommit 儲存庫。如果您想改用自己的伺服器，請參AWS Cloud9使用者指南。

建立 AWS Cloud9 環境

1. 登入AWS作為您已設定的 IAM 用户，然後打開AWS Cloud9主控台。
2. 在 AWS Cloud9 主控台，選擇 Create environment (建立環境)。
3. In步驟 1：名稱環境，輸入環境的名稱和選用描述，然後選擇下一個步驟。
4. In步驟 2：進行設定，請按如下方式設定環境：
   - 在 Environment type (環境類型) 中，選擇 Create a new instance for environment (EC2) (為環境建立新的執行個體 (EC2))。
   - 在 Instance type (執行個體類型) 中，為您的開發環境選擇適當的執行個體類型。例如，如果您只是探索服務，您可以選擇預設值 t2.micro。如果您想要將此環境用於開發工作，請選擇較大的執行個體類型。
   - 接受其他預設設定，除非您有理由需要另外選擇 (例如，您的組織使用特定的 VPC，或您的 Amazon Web Services 帳户沒有設定任何 VPC)，然後選擇下一個步驟。
5. In步驟 3：檢閱下，可檢您的設定。如需進行任何變更，請選擇 Previous step (上一步)。否則，請選擇 Create environment (建立環境)。

   建立環境後，首次連接此環境需要幾分鐘的時間。如果它似乎需要非常長的時間，請參閱故障診斷中的AWS Cloud9使用者指南。

6. 連接到環境後，請在終端機視窗執行 git --version 命令，以檢查 Git 是否已安裝，而且是支援的版本。

   如果 Git 未安裝或不是支援的版本，請安裝支援的版本。 CodeCommit 支援 Git 1.7.9 版和更新版本。Git 2.28 版支持為初始提交配置分支名稱。我們建議您使用最新版本的 Git。若要安裝 Git，建議前往Git 下載。

   > **Tip**
   >
   > 根據環境的作業系統而定，您或許可以使用 yum 命令搭配 sudo 選項來安裝更新，包括 Git。例如，管理命令序列可能類似於下列三個命令：
   >
   > ```
   > sudo yum -y update
   > sudo yum -y install git
   > git --version
   > ```

7. 執行 git config 命令，以設定要與 Git 遞交相關聯的使用者名稱和電子郵件。例如：

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

步驟 2：設定AWS CLI登入資料協助程式AWS Cloud9EC2 開發環境

在創建AWS Cloud9環境中，您可以配置AWS CLI登入資料協助程式來管理連線的登入資料 CodeCommit 儲存庫。所以此AWS Cloud9開發環境附帶AWS與您的 IAM 用户相關聯的託管臨時登入資料。您會透過 AWS CLI 登入資料協助程式來使用這些登入資料。

1. 開啟終端機視窗，並執行下列命令來驗證 AWS CLI 已安裝：

   ```
   aws --version
   ```

   如果成功，這個命令會傳回目前已安裝的 AWS CLI 版本。若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 在終端機，執行下列命令來設定 AWS CLI 登入資料協助程式，以使用 HTTPS 連線：

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

```
git config --global credential.UseHttpPath true
```

> **Tip**
>
> 登入資料協助程式會將預設的 Amazon EC2 執行個體角色用於您的開發環境。如果您想要使用開發環境來連接的儲存庫託管位置不是在 CodeCommit 中，請對這些儲存庫設定 SSH 連線，或將本機`.gitconfig`文件，以便在連接到這些其他存儲庫時使用備用憑據管理系統。如需詳細資訊，請參閱 Git 網站上的 Git Tools - Credential Storage。

步驟 3：複製 CodeCommit 儲存庫拖入AWS Cloud9EC2 開發環境

當您設定AWS CLI登入資料協助程式，您可以將 CodeCommit 儲存庫複製到其中。然後，您就可以開始使用程式碼。

1. 在終端機，執行 git clone 命令，並針對您想要複製的儲存庫，指定其 HTTPS 複製 URL。例如，如果您想要複製名為的儲存庫 MyDemoRepo 在美國東部 (俄亥俄) 區域，請輸入：

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

   > **Tip**
   >
   > 您可以在 CodeCommit 控制台，通過選擇複製 URL。

2. 當複製完成時，在側邊導覽中展開儲存庫的資料夾，然後選擇您要開啟來編輯的檔案。或者，選擇 File (檔案)，然後選擇 New File (新增檔案) 以建立檔案。

3. 完成編輯或建立檔案時，請在終端機視窗中，切換到已複製的儲存庫所在的目錄，然後遞交並推送您的變更。例如，如果您新增的檔案名為 *MyFile.py*：

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

後續步驟

如需詳細資訊，請參閱 。AWS Cloud9使用者指南和的 CodeCommit 示例AWS Cloud9。如需使用 Git 搭配 CodeCommit 的詳細資訊，請參開始使用 Git 和 AWS CodeCommit。

整合視覺工作室與AWS CodeCommit

您可以使用 Visual Studio 在 CodeCommit 儲存庫中進行程式碼變更。AWS Toolkit for Visual Studio現在包含的功能，可讓您在 Visual Studio 中工作時更 CodeCommit 輕鬆、更方便。Visual Studio 整合的工具組是專為與 Git 登入資料和 IAM 使用者搭配使用而設計的。您可以複製現有的儲存庫、建立儲存庫、將程式碼變更遞交並推送至儲存庫等等。

> **Important**
>
> 此工 Toolkit for Visual Studio 用於安裝在 Windows 作業系統上。如果您正在尋找有關使用 Visual Studio 程式碼的資訊，請參閱AWS Toolkit for Visual Studio Code。

如果您之前使用過 Toolkit for Visual Studio，您可能已經熟悉設定包含存取金鑰和秘密金鑰的AWS認證設定檔。認證設定檔用於 Visual Studio 的工具組中，以啟用對AWS服務 API 的呼叫 (例如，對 Amazon S3 列出儲存貯體或列出儲存庫)。 CodeCommit 要將代碼提取並推送到 CodeCommit 存儲庫，您還需要 Git 憑據。如果您沒有 Git 認證，Visual Studio 的工具組可以為您產生並套用這些認證。這可以節省許多時間。

若要搭配使用視覺工作室 CodeCommit，您需要下列項目：

- 具有為其設定的一組有效登入資料 (存取金鑰和秘密金鑰) 的 IAM 使用者。此 IAM 使用者也應具備：

  其中一個受 CodeCommit 管理的策略和套用的IAMSelfManageServiceSpecificCredentials受管理策略。

  或

  如果 IAM 使用者已設定 Git 登入資料，則其中一個受 CodeCommit管政策或同等權限。

  如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策和了解和取得您的安全登入資料。
- AWS Toolkit for Visual Studio安裝在您已安裝視覺工作室的電腦上。如需詳細資訊，請參閱設定 AWS Toolkit for Visual Studio。

如需AWS Toolkit for Visual Studio搭配使用的詳細資訊 CodeCommit，請參閱AWS CodeCommit搭配 Visual Studio 使用者指南的工具組中的搭配 Visual Studio 小組總管使用。

Eclipse 與 AWS CodeCommit 整合

您可以使用 Eclipse 在 CodeCommit 儲存庫中進行程式碼變更。適用於 Eclipse 整合的工具組旨在與 Git 登入資料和 IAM 使用者搭配使用。您可以複製現有的儲存庫、建立儲存庫、將程式碼變更遞交並推送至儲存庫等等。

要使用 Toolkit for Eclipse CodeCommit，您需要以下內容：

- 安裝在本機電腦的 Eclipse。
- 具有為其設定的一組有效登入資料 (存取金鑰和秘密金鑰) 的 IAM 使用者。此 IAM 使用者也應具備：

  其中一個受 CodeCommit 管理的策略和套用的IAMSelfManageServiceSpecificCredentials受管理策略。

  或

  如果 IAM 使用者已設定 Git 登入資料，則其中一個受 CodeCommit管政策或同等權限。

  如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策和了解和取得您的安全登入資料。
- 為 IAM 中的使用者設定的一組作用中 Git 認證。如需詳細資訊，請參閱第 3 步：為 HTTPS 連接創建 Git 憑據 CodeCommit。

主題

步驟 1：取得您 IAM 使用者的存取存取金鑰和秘密金鑰

如果您尚未在安裝 Eclipse 的電腦上設定認證設定檔，您可以使用AWS CLI和aws configure指令來設定一個認證設定檔。或者，您可以依照此程序中的步驟來建立和下載您的登入資料。出現提示時，將它們提供給 Eclipse 的工具包。

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	若要	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDK 或 AWS API 的程式請求。	請依照您要使用的介面遵循指示。 • 對於 AWS CLI，請參閱 AWS Command Line Interface 使用者指南中的設定 AWS CLI 以使用 AWS IAM Identity Center。 • 對於 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM Identity Center 身分驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDK 或 AWS API 的程式請求。	請遵循《IAM 使用者指南》中使用臨時憑證搭配 AWS 資源中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDK 或 AWS API 的程式請求。	請依照您要使用的介面遵循指示。 • 對於 AWS CLI，請參閱 AWS Command Line Interface 使用者指南中的使用 IAM 使用者憑證進行身分驗證。 • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑證進行身分驗證。

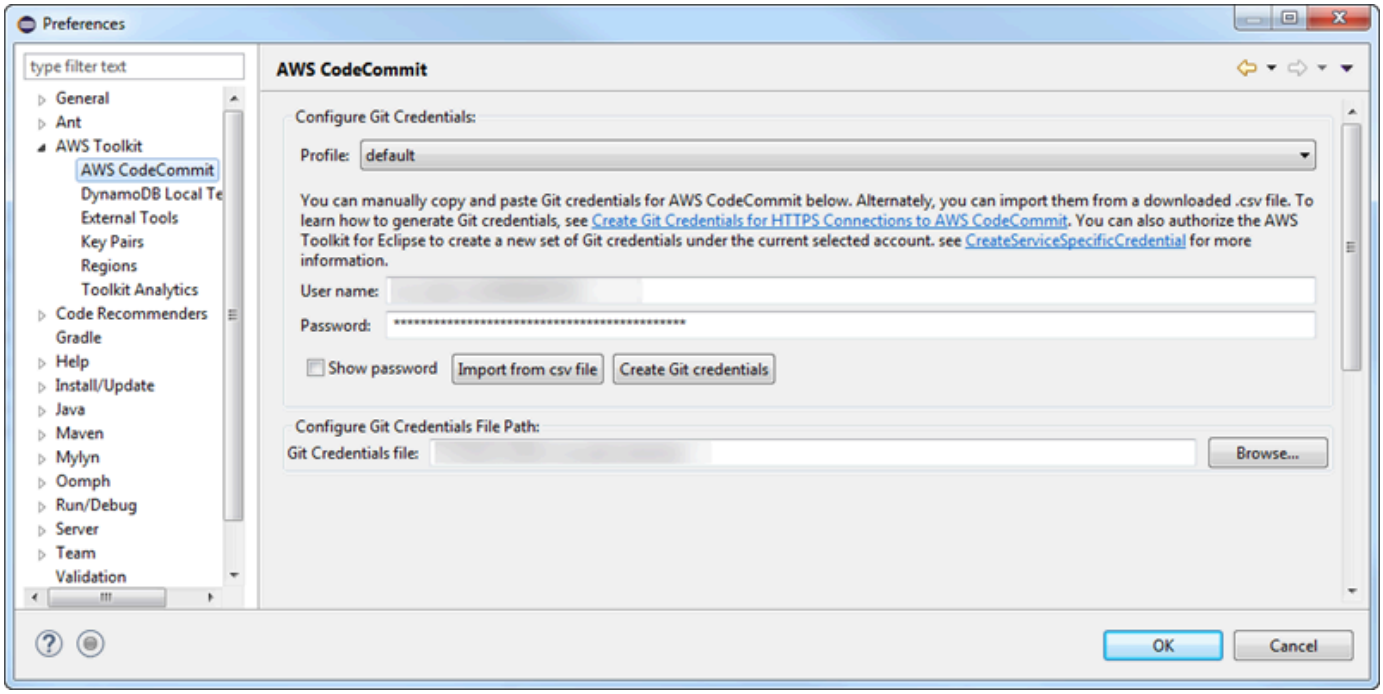
哪個使用者需要程式設計存取權？	若要	By
		• 對於 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

步驟 2：安裝AWS Toolkit for Eclipse並連接 CodeCommit

Toolkit for Eclipse 包是一個軟件包，你可以添加到 Eclipse。安裝並使用AWS憑據配置文件對其進行配置後，您可以 CodeCommit 從 Eclipse 中的AWS資源管理器連接到。

使用AWS CodeCommit模塊安裝 Eclipse 工具包並配置對項目存儲庫的訪問

1. 如果您沒有安裝受支持的版本，請在本地計算機上安裝 Toolkit for Eclipse。如果您需要更新 Eclipse 的工具包版本，請按照設置工具包中的說明進行操作。
2. 在 Eclipse 中，請遵循第一個體驗，或者從 Eclipse 菜單系統中打開首選項（位置取決於您的版本和操作系統），然後選擇AWS工具包。
3. 執行下列任意一項：
   - 如果您遵循 firstrun 體驗，請在系統提示您設定認證設定檔時提供您的AWS安全性認證。
   - 如果您在 Preferences (偏好設定)中設定，且電腦上已設定登入資料描述檔，請從 Default Profile (預設描述檔) 中選擇此描述檔。
   - 如果您在 Preferences (偏好設定) 中設定，但沒看到您想要使用的描述檔，或清單是空的，請選擇 Add profile (新增設定檔)。在「設定檔詳細資料」中，輸入設定檔的名稱和 IAM 使用者的登入資料 (存取金鑰和秘密金鑰)，或者輸入登入資料檔案的位置。
   - 如果您在 Preferences (偏好設定) 中設定，且您尚未設定描述檔，請使用連結來註冊帳戶或管理現有的 AWS 安全登入資料。
4. 在 Eclipse 中，展開「AWS工具包」菜單並選擇AWS CodeCommit。選擇您的登入資料描述檔，然後輸入 Git 登入資料的使用者名稱和密碼，或從 .csv 檔案匯入 Git 登入資料。選擇 Apply (套用)，然後選擇 OK (確定)。



以描述檔登入之後，AWS CodeCommit 連線面板會出現在 Team Explorer 中，並提供複製、建立或登出等選項。選擇「複製」會將現有 CodeCommit 存放庫複製到您的本機電腦，以便您可以開始處理程式碼。這是最常用的選項。

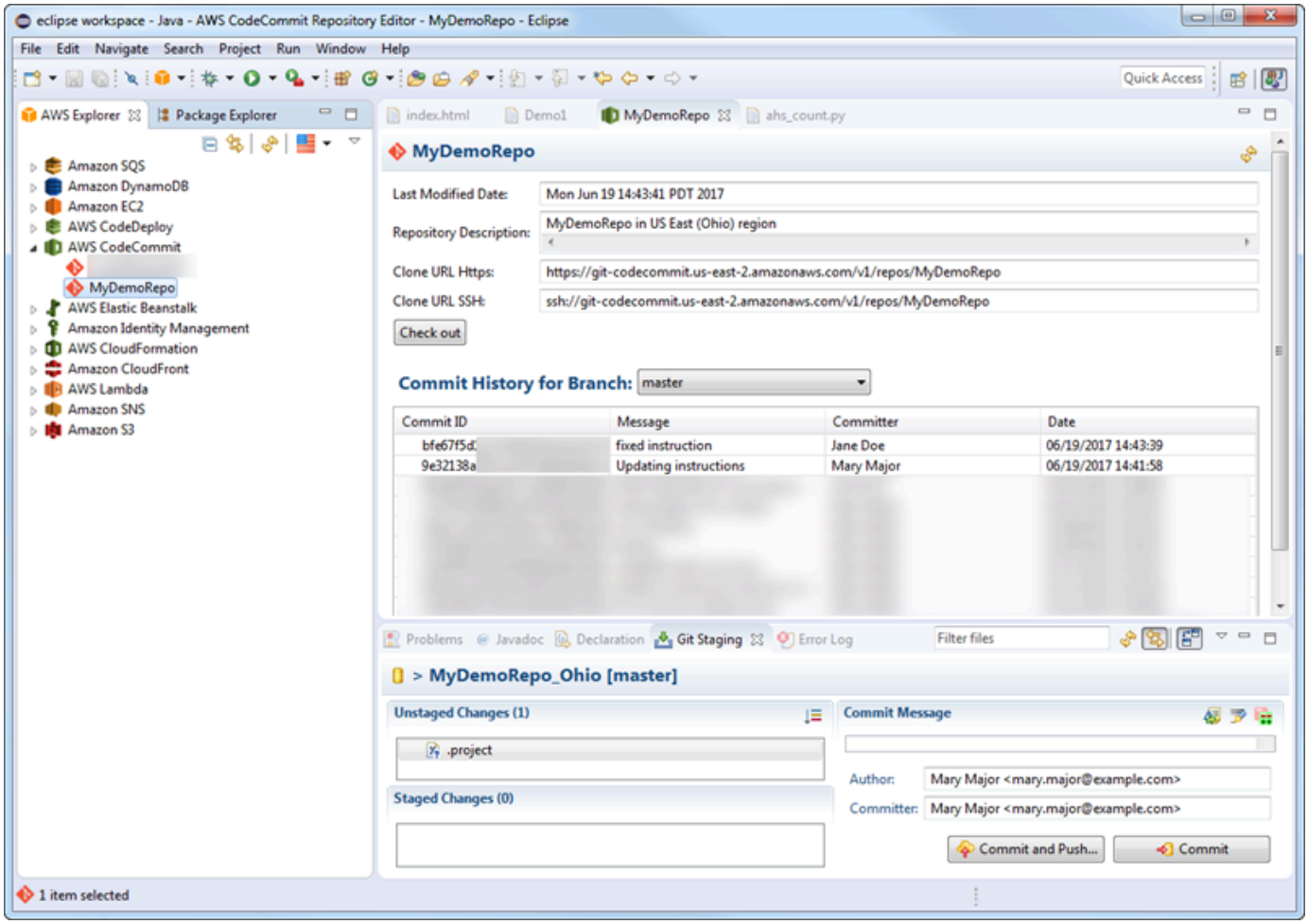
如果您沒有任何儲存庫，或是想要建立儲存庫，請選擇 Create (建立)。

從日食克隆一個 CodeCommit 存儲庫

設定您的登入資料之後，您可以在 Eclipse 中簽出儲存庫，將此儲存庫複製到電腦上當做本機儲存庫。然後，您就可以開始使用程式碼。

1. 在日食中，打開AWS資源管理器。如需其位於何處的相關資訊，請參閱如何存取 AWS Explorer。展開 AWS CodeCommit，然後選擇您想要使用的 CodeCommit 儲存庫。您可以檢視儲存庫的遞交歷史記錄和其他詳細資訊，以協助您判斷這是否為您要複製的儲存庫和分支。

> **Note**
>
> 如果您沒有看到存放庫，請選擇旗標圖示以開啟AWS 區域功能表，然後選擇建立存放庫的AWS 區域位置。



2. 選擇 Check out (簽出)，然後依照指示將儲存庫複製到本機電腦上。
3. 當您完成複製專案時，就可以開始在 Eclipse 中編輯程式碼，並將變更暫存、遞交和推送到 CodeCommit 中的專案儲存庫。

從日食創建一個 CodeCommit 存儲庫

您可以使用 Eclipse 的工具包從 Eclipse 創建 CodeCommit 存儲庫。建立儲存庫時，您也會將儲存庫複製到電腦上當做本機儲存庫，讓您立即開始使用儲存庫。

1. 在AWS檔案總管中，以滑鼠右鍵按一下 AWS CodeCommit，然後選擇 [建立

> **Note**
>
> 儲存庫專屬於特定區域。建立儲存庫之前，請先確定您已選取正確的儲存庫AWS 區域。您無法在啟動存放庫建立程序AWS 區域之後選擇。

2. 在 Repository Name (儲存庫名稱) 中，輸入此儲存庫的名稱。儲存庫名稱在 Amazon Web Services 帳戶中必須不重複。字元和長度有所限制。如需詳細資訊，請參閱配額。在 Repository Description (儲存庫描述) 中，輸入此儲存庫的選用描述。這有助於其他人了解此儲存庫的用途，並協助與區域中的其他儲存庫有所區別。選擇 OK (確定)。
3. 在AWS檔案總管中 AWS CodeCommit，展開，然後選擇您剛建立的 CodeCommit 存放庫。您會看到此儲存庫沒有遞交歷史記錄。選擇 Check out (簽出)，然後依照指示將儲存庫複製到本機電腦上。

使用 CodeCommit 儲存庫

連線到之後 CodeCommit，您可以在AWS檔案總管中看到與您帳戶相關聯的儲存庫清單。AWS 區域選擇旗標功能表來變更區域。

> **Note**
>
> CodeCommit 可能並非在 Eclipse 工具包AWS 區域支持的所有支持中都可用。

在 Toolkit for Eclipse 包中，您可以從導航和 Package 資源管理器視圖瀏覽這些存儲庫的內容。若要開啟檔案，請從清單中選擇檔案。

Toolkit for Eclipse 中的 Git 操作與其他任何基於 GIT 的存 CodeCommit 儲庫的工作方式完全相同。您可以變更程式碼、新增檔案，以及建立本機遞交。當您準備好共用時，您可以使用 Git 暫存選項將提交推送到 CodeCommit 儲存庫。如果您在 Git 描述檔中尚未設定作者和遞交者資訊，您可以在遞交和推送之前這樣做。由於 IAM 使用者的 Git 登入資料已儲存在本機，並與連線的AWS認證設定檔相關聯，因此在您推送至時，系統不會提示您再次提供這些登入資料 CodeCommit。

如需使用 Toolkit of Eclipse。AWS Toolkit for Eclipse

為 SSH 用户安裝不使用AWS CLI

如果您想要對儲存庫使用 SSH 連線，您可以連接到 AWS CodeCommit，而無需安裝 AWS CLI。所以此AWS CLI包含您使用和管理 CodeCommit 儲存庫時很有用的命令，但初始設定不需要使用此工具。

此主題假設：

- 您已設定 IAM 使用者且所需的政策或許 CodeCommit，以及IAMUserSSHKeys託管策略或上傳密鑰所需的等效權限。如需詳細資訊，請參閱 使用以身分為基礎的政策 (IAM 政策) CodeCommit。
- 您已經具有 (或知道如何建立) 公有-私有金鑰對。強烈建議您對 SSH 金鑰使用安全的複雜密碼。
- 您熟悉 SSH、Git 用戶端及其組態檔案。
- 如果您使用 Windows，則您已安裝可模擬 bash shell 的命令列公用程式，例如 Git Bash。

如果您需要更多指導，請遵循 Linux或 適用於 Windows 上的 SSH 連線中的指示。

主題

步驟 1：將您的公有金鑰與 IAM 使用者相關聯

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇使用者，然後從使用者清單中選擇 IAM 使用者。
3. 在 Security Credentials (安全登入資料) 標籤上，選擇 Upload SSH public key (上傳 SSH 公有金鑰)。
4. 將 SSH 公有金鑰的內容貼到欄位中，然後選擇 Upload SSH Key (上傳 SSH 金鑰)。

> **Tip**
>
> 公有-私有金鑰對必須為 SSH-2 RSA、OpenSSH 格式，且包含 2048 位元。金鑰看起來如下：

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

IAM 只接受 OpenSSH 格式的公有金鑰。如果您提供其他格式的公有金鑰，則會看到錯誤訊息，指出金鑰格式無效。

5. 複製 SSH 金鑰 ID (例如，*APKAEIBAERJR2EXAMPLE*) 並關閉主控台。

步驟 2：將 CodeCommit 添加到您的 SSH 配置

1. 在終端機 (Linux、macOS 或 Unix) 或 bash 模擬器 (Windows) 中，輸入cat>> ~/.ssh/config：

```
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**
>
> 如果您有多個 SSH 組態，請務必在內容前後包含空白行。同時按下 `Ctrl` 和 `d` 鍵來儲存檔案。

2. 執行下列命令以測試 SSH 組態：

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

出現提示時，為您的 SSH 金鑰檔案輸入複雜密碼。如果一切都設定正確，您應該會看到以下成功訊息：

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

下一步驟

您已完成事前準備。請遵循開始使用 CodeCommit 開始使用 CodeCommit。

若要連接到儲存庫，請遵循連接到儲存庫中的步驟。若要建立儲存庫，請遵循建立 儲存庫中的步驟。

Linux Linux Linux LinuxAWS CodeCommit Linux，Linux macOS Unix

第一次連接到CodeCommit，您必須先完成一些初始設定步驟。在您設定電腦和設定AWS檔之後，您可以連線到CodeCommit儲存庫，並將該儲存庫複製到您的電腦 (也稱為建立本機存放庫)。如果您是初次使用 Git，您可能還需要檢閱我可以在哪裡進一步了解 Git？中的資訊。

主題

- 步驟 1：初始配置CodeCommit
- 步驟 2：安裝 Git
- Linux 步驟 Linux 步驟 Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux
- 步驟 4：Connect 至主CodeCommit控台並複製存放庫
- 後續步驟

步驟 1：初始配置CodeCommit

請依照下列步驟設定 Amazon Web Services 帳戶、建立 IAM 使用者，以及設定存取權限 CodeCommit。

若要建立和設定 IAM 使用者以存取CodeCommit

1. 通過轉到 http://aws.amazon.com 並選擇註冊創建一個 Amazon Web Services 帳戶。

2. 在您的 Amazon Web Services 帳戶中建立 IAM 使用者，或使用現有的使用者。確認您有存取金鑰 ID，以及與該 IAM 使用者相關聯的私密存取金鑰。如需詳細資訊，請參閱在 Amazon Web Services 帳戶中建立 IAM 使用者。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS執行所需的動作CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中選擇使用者，然後選擇您要為CodeCommit存取的 IAM 使用者。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從政策清單中，選取 AWSCodeCommitPowerUser，或另一個用於存取 CodeCommit 的受管政策。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的政策之後，請選擇 [下一步：檢閱] 以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策及分享儲存庫存取權給其他群組和使用者的詳細資訊，請參閱共用儲存庫和 AWS CodeCommit 的身分驗證與存取控制。

> **Note**
>
> 如果您想要使用 AWS CLI 命令來搭配 CodeCommit，請安裝 AWS CLI。如需詳細資訊，請參閱命令列參考。

步驟 2：安裝 Git

若要處理CodeCommit儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 1.7.9 版和更新版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

若要安裝 Git，我們建議使用 Git 下載等網站。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時，一項功能的改變，可能會影響此功能與 CodeCommit 運作的方式。如果您在特定版本的 Git 和 CodeCommit 方面遇到問題，請檢閱疑難排解中的資訊。

Linux 步驟 Linux 步驟 Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux Linux

安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰CodeCommit

若要為 Git 和設定公開金鑰和私密金鑰CodeCommit

1. 從本機電腦上的終端機，執行 ssh-keygen 命令，並依照指示將檔案儲存到描述檔的 .ssh 目錄。

   > **Note**
   >
   > 請務必向系統管理員詢問應該金鑰檔的存放位置，以及應該採用的檔案命名模式。

   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 . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

這會產生：

- *codecommit_rsa* 檔案，這是私有金鑰檔案。
- *codecommit_rsa*.pub 檔案，這是公有金鑰檔案。

> **Tip**
>
> 依預設，ssh-keygen會產生 2048 位元金鑰。您可以使用-t 和-b 參數來指定鍵的類型和長度。如果您想要 rsa 格式的 4096 位元金鑰，您可以使用下列參數執行命令來指定：
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> 如需 SSH 金鑰所需格式和長度的詳細資訊，請參閱搭配使用 IAMCodeCommit。

2. 執行以下命令來顯示公有金鑰檔案 (*codecommit_rsa*.pub) 的值：

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

複製這個值。它看起來類似下列：

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

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

> **Note**
>
> 您可以在「我的安全CodeCommit登入資料」中直接檢視和管理您的認證。如需詳細資訊，請參閱查看和管理您的憑據。

4. 在 IAM 主控台的導覽窗格中，選擇 [使用者]，然後從使用者清單中選擇您的 IAM 使用者。
5. 在使用者詳細資訊頁面，選擇 Security Credentials (安全登入資料) 標籤，然後選擇 Upload SSH public key (上傳 SSH 公有金鑰)。
6. 將 SSH 公有金鑰的內容貼到欄位中，然後選擇 Upload SSH public key (上傳 SSH 公有金鑰)。
7. 複製或儲存 SSH Key ID (SSH 金鑰 ID) 中的資訊 (例如，*APKAEIBAERJR2EXAMPLE*)。

> **Note**
>
> 如果您已上傳一個以上的 SSH 金鑰 ID，則會依金鑰 ID 的字母順序列出金鑰，而非依上傳日期。請確定您已複製的金鑰 ID 與正確的上傳日期相關聯。

8. 在本機電腦上，使用文字編輯器在 ~/.ssh 目錄中建立組態檔，然後將下列幾行新增至檔案，其中 *User* 的值是您稍早複製的 SSH 金鑰 ID：

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

> **Note**
>
> 如果您將私有金鑰檔案命名為 *codecommit_rsa* 以外的名稱，請務必在此處使用該名稱。
>
> 您可以在多個 Amazon Web Services 帳戶中設定存放庫的 SSH 存取權，如需詳細資訊，請參閱疑難排解 SSH 連線至AWS CodeCommit。

儲存此檔案並命名為 config。

9. 從終端機，執行下列命令來變更組態檔的許可：

```
chmod 600 config
```

10. 執行下列命令以測試 SSH 組態：

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

系統會要求您確認連線，因git-codecommit.us-east-2.amazonaws.com為尚未包含在已知的 hosts 檔案中。CodeCommit伺服器指紋會顯示為驗證的一部分(a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e針3lBlW2g5xn/NA2Ck6dyeJIrQ0Wvn7n8UEs56fG6ZIzQ對 MD5 或 SHA256)。

> **Note**
>
> CodeCommit每個伺服器指紋都是唯一的AWS 區域。若要檢視的伺服器指紋AWS 區域，請參閱伺服器指紋 CodeCommit。

確認連線之後，您應該會看到確認訊息，指出您已將伺服器新增到已知主機檔案，還會出現成功連線的訊息。如果您沒有看到成功訊息，請檢查您是否將config檔案儲存在您設定要存取的 IAM 使用者的 ~/.ssh 目錄中CodeCommit，以及您指定了正確的私密金鑰檔案。

如需協助您對問題進行故障診斷，請使用-v參數執行ssh命令。例如：

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

如需資訊來協助您排除連線問題，請參閱疑難排解 SSH 連線至AWS CodeCommit。

步驟 4：Connect 至主CodeCommit控台並複製存放庫

如果管理員已將 CodeCommit 儲存庫的名稱和連線詳細資訊傳送給您，則您可以略過此步驟，並直接複製儲存庫。

連接到 CodeCommit 儲存庫

1. 開啟主CodeCommit控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選取器中，選擇建立儲存庫的AWS 區域位置。儲存庫特定於AWS 區域. 如需詳細資訊，請參閱區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要搭配 IAM 使用者使用 Git 登入資料，或是AWS CLI.
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。
   - 如果您將 SSH 公開/私密 key pair 與 IAM 使用者搭配使用，請複製 SSH URL。

   > **Note**
   >
   > 如果您看到「歡迎」頁面而不是儲存庫清單，表示您登入的AWS 區域地方沒有與您的AWS帳戶相關聯的儲存庫。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 開啟 終端機。從 /tmp 目錄，使用您複製的 SSH URL，執行 git clone 命令來複製儲存庫。例如，若要複製名*MyDemoRepo*為美國東部 (俄亥俄) 區域名稱*my-demo-repo*的本機存放庫：

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

> **Note**
>
> 如果您成功測試連線，但複製命令失敗，可能是您沒有組態檔的必要存取權，也可能是其他設定與您的組態檔發生衝突。嘗試重新連線，這次在命令中包含 SSH 金鑰 ID。例如：
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 如需詳細資訊，請參閱存取錯誤：公開金鑰已成功上傳至 IAM，但在 Linux、macOS 或 Unix 系統上連線失敗。

如需有關如何連接到儲存庫的詳細資訊，請參閱透過複製存 CodeCommit 放庫連 Connect 至儲存庫。

後續步驟

您已完成事前準備。請按照中的步驟開始使用 CodeCommit 開始使用CodeCommit。

SSH 連接到AWS CodeCommit存儲庫

在您可以連接到AWS CodeCommit第一次，您必須先完成一些初始設定步驟。在您設定您的計算機和AWS描述檔，則您可以連接到 CodeCommit 儲存庫，並將該儲存庫複製到您的電腦 (也稱為建立本機儲存庫)。如果您是初次使用 Git，您可能還需要檢閱我可以在哪裡進一步了解 Git？中的資訊。

主題

步驟 1：CodeCommit 的初始設定

請遵循以下步驟設定 Amazon Web Services 賬户、建立 IAM 使用者，以及設定 CodeCommit 的存取。

建立和設定 IAM 使用者以存取 CodeCommit

1. 前往http://aws.amazon.com並選擇註冊。
2. 在您的 Amazon Web Services 賬户中建立 IAM 使用者或使用現有 IAM 使用者。請確定您具有與 IAM 使用者關聯的存取金鑰 ID 和私密存取金鑰。如需詳細資訊，請參閱「」在您的 Amazon Web Services 賬户中建立 IAM 使用者。

> **Note**
>
> CodeCommit 需要AWS Key Management Service。如果您使用現有 IAM 使用者，請確定連接到使用者的政策沒有明確拒絕AWS KMSCodeCommit 所需的操作。如需詳細資訊，請參閱 AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇使用者，然後選擇您要設定用於 CodeCommit 存取的 IAM 使用者。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從政策清單中，選擇AWSCodeCommitPowerUser或另一個用於 CodeCommit 訪問的託管策略。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取您要連接的政策之後，選擇下一頁: 檢閱以檢要連接到 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需 CodeCommit 託管政策以及與其他組和使用者共享儲存庫存取權的詳細資訊，請參共用儲存庫和AWS CodeCommit 的身分驗證與存取控制。

> **Note**
>
> 如果您想要使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

步驟 2：安裝 Git

要處理 CodeCommit 存儲庫中的文件、提交和其他信息，必須在本地計算機上安裝 Git。CodeCommit 支援 Git 1.7.9 版和更新版本。Git 2.28 版支持為初始提交配置分支名稱。我們建議您使用最新版本的 Git。

若要安裝 Git，建議前往Git 下載。

> **Note**
>
> Git 是一種持續演進且定期更新的平台。有時，功能更改可能會影響它與 CodeCommit 搭配使用的方式。如果您在特定版本的 Git 和 CodeCommit 方面遇到問題，請檢疑難排解。

如果您安裝的 Git 版本不含 Bash 模擬器，例如 Git Bash，請安裝模擬器。當您設定 SSH 連線時，您將使用此模擬器，而不是 Windows 命令列。

步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰

設定 Git 和 CodeCommit 的公有和私有金鑰Windows 上的

1. 開啟 Bash 模擬器。

   > **Note**
   >
   > 您可能需要以管理許可來執行模擬器。

   從模擬器，執行 ssh-keygen 命令，並依照指示將檔案儲存到描述檔的 .ssh 目錄。

   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 . . . |
|          .      |
|                 |
|                 |
|                 |
+-----------------+
```

這會產生：

- *codecommit_rsa* 檔案，這是私有金鑰檔案。
- *codecommit_rsa*.pub 檔案，這是公有金鑰檔案。

> **Tip**
>
> 在預設情況下，ssh-keygen生成一個 2048 位密鑰。可以使用-t 和-b 參數指定密鑰的類型和長度。如果要使用 rsa 格式的 4096 位密鑰，則可以通過運行具有以下參數的命令來指定此項：
>
> ```
> ssh-keygen -t rsa -b 4096
> ```
>
> 如需 SSH 金鑰所需的格式和長度的詳細資訊，請參使用 IAM 與 CodeCommit。

2. 執行以下命令來顯示公有金鑰檔案 (*codecommit_rsa*.pub) 的值：

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

複製檔案的內容，然後關閉「記事本」但不儲存。檔案的內容類似如下：

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

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

> **Note**
>
> 您可以直接查看和管理您的 CodeCommit 憑據我的安全登入資料。如需詳細資訊，請參閱查看和管理您的憑據。

4. 在 IAM 主控台的導覽窗格中，選擇使用者，然後從使用者清單中選擇您的 IAM 使用者。
5. 在使用者詳細資訊頁面，選擇 Security Credentials (安全登入資料) 標籤，然後選擇 Upload SSH public key (上傳 SSH 公有金鑰)。
6. 將 SSH 公有金鑰的內容貼到欄位中，然後選擇 Upload SSH public key (上傳 SSH 公有金鑰)。
7. 複製或儲存 SSH Key ID (SSH 金鑰 ID) 中的資訊 (例如，*APKAEIBAERJR2EXAMPLE*)。

> **Note**
>
> 如果您已上傳一個以上的 SSH 金鑰 ID，則會依金鑰 ID 的字母順序列出金鑰，而非依上傳日期。請確定您已複製的金鑰 ID 與正確的上傳日期相關聯。

8. 在 Bash 模擬器中，執行下列命令在 ~/.ssh 目錄中建立組態檔，如果此檔案已存在，請編輯檔案：

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

9. 將下列幾行新增至檔案，其中 *User* 的值是您稍早複製的 SSH 金鑰 ID，*IdentityFile* 的值是私有金鑰檔案的路徑與名稱：

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

> **Note**
>
> 如果您將私有金鑰檔案命名為 *codecommit_rsa* 以外的名稱，請務必在此處使用該名稱。
> 您可以在多個 Amazon Web Services 帳戶中設置存儲庫的 SSH 訪問權限，有關更多信息，請參閱疑難排解 SSH 連線至AWS CodeCommit。

將檔案儲存為 config (而非 config.txt)，然後關閉「記事本」。

> **Important**
>
> 檔案名稱必須是 `config` 且沒有副檔名。否則，SSH 連線會失敗。

10. 執行下列命令以測試 SSH 組態：

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

系統會請您確認連接，因為`git-codecommit.us-east-2.amazonaws.com`尚未包含在已知主機文件中。CodeCommit 服務器指紋顯示為驗證的一部分 (`a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`MD5 或`3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`對於 SHA256 來說）。

> **Note**
>
> CodeCommit 服務器指紋對於每個AWS 區域。要查看服務器指紋AWS 區域，請參伺服器指紋 CodeCommit。

確認連線之後，您應該會看到確認訊息，指出您已將伺服器新增到已知主機檔案，還會出現成功連線的訊息。如果您看不到成功消息，請再次檢查您保存`config`文件位於您配置為 CodeCommit 訪問而配置的 IAM 用户的 ~/.ssh 目錄中，`config`文件沒有文件擴展名（例如，它不能命名為 config.txt），並且您指定了正確的私鑰文件（*#### _rsa*，而不是*#### _rsa*.pub)。

要解決問題，請運行`ssh`命令與`-v`參數。例如：

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

如需資訊來協助您排除連線問題，請參閱疑難排解 SSH 連線至AWS CodeCommit。

步驟 4：Connect 到 CodeCommit 控制台並克隆存儲庫

如果管理員已將 CodeCommit 儲存庫的名稱和連線詳細資訊傳送給您，則您可以略過此步驟，並複製儲存庫 直接執行。

連接到 CodeCommit 儲存庫

1. 開啟 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選擇器中，選擇AWS 區域儲存庫建立。儲存庫為AWS 區域。如需詳細資訊，請參閱 區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您是將 Git 登入資料與 IAM 使用者搭配使用，或是使用AWS CLI。
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。
   - 如果您是將 SSH 公有/私有金key pair 與 IAM 使用者搭配使用，請複製 SSH URL。

   > **Note**
   >
   > 如果您看見歡迎頁面而不是儲存庫清單，這表示沒有與AWS帳户AWS 區域在您登入的位置。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 在 Bash 模擬器，使用您複製的 SSH URL 執行 git clone 命令，以複製儲存庫。在您執行此命令的目錄中，此命令會在該目錄的子目錄中建立本機儲存庫。例如，若要複製名為的儲存庫*MyDemoRepo*添加到名為*my-demoy-repo*在美國東部 (俄亥俄) 區域：

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

   或者，開啟命令提示字元，並使用您上傳到 IAM 的公有金鑰的 URL 和 SSH 金鑰 ID，執行git clone命令。在您執行此命令的目錄中，將會在該目錄的子目錄中建立本機儲存庫。例如，若要複製名為的儲存庫*MyDemoRepo*添加到名為*my-demoy-repo*：

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

如需更多詳細資訊，請參閱 透過複製存 CodeCommit 放庫連 Connect 至儲存庫 及 創建一個提交。

下一步驟

您已完成事前準備。請遵循開始使用 CodeCommit 開始使用 CodeCommit。

HTTPS 連線的設定步驟AWS CodeCommitLinux、MacOS 或 Unix 上的儲存庫AWS CLI憑證助手

第一次連接到 AWS CodeCommit 之前，您必須先完成初始設定步驟。對於大多數使用者，最輕鬆的方式是遵循適用於使用 Git 認證的 HTTPS 使用者中的步驟。不過，如果您想要使用根帳戶、聯合存取或暫時性登入資料來連線到 CodeCommit，您可以使用 AWS CLI 隨附的登入資料協助程式。

> **Note**
>
> 雖然登入資料協助程式是可支援使用聯合存取、身分提供者或暫時性登入資料連線到 CodeCommit 的方法，但建議採用的方法是安裝並使用 git-remote-codecommit 公用程式。如需詳細資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

主題

步驟 1：初始配置CodeCommit

請依照下列步驟設定 Amazon 網路服務帳戶、建立和設定 IAM 使用者，以及安裝AWS CLI。

若要建立和設定 IAM 使用者以存取CodeCommit

1. 通過轉到創建一個亞馬遜網絡服務帳戶http://aws.amazon.com並選擇立即註冊。
2. 在您的亞馬遜網路服務帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的亞馬遜網絡服務帳戶中創建 IAM 用戶。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS所需的動作CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇使用者，然後選擇您要設定的 IAM 使用者CodeCommit訪問。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從政策清單中，選取 AWSCodeCommitPowerUser，或另一個用於存取 CodeCommit 的受管政策。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的策略之後，請選擇下一個:評論以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策及分享儲存庫存取權給其他群組和使用者的詳細資訊，請參閱共用儲存庫和 AWS CodeCommit 的身分驗證與存取控制。

安裝及設定 AWS CLI

1. 在本機電腦上，下載並安裝 AWS CLI。這是從命令列與 CodeCommit 互動的必要步驟。我們建議您安裝AWS CLI版本 2. 它是最新的主要版本AWS CLI並支持所有最新功能。這是唯一的版本AWS CLI支援使用 root 帳戶、同盟存取或臨時登入資料git-remote-codecommit。

   如需詳細資訊，請參閱開始設定 AWS 命令列界面。

> **Note**
>
> CodeCommit僅適用於AWS CLI版本 1.7.38 及更高版本。根據最佳實務，安裝 AWS CLI 或將其升級到可用的最新版本。若要判斷您已安裝的 AWS CLI 版本，請執行 aws --version 命令。
> 若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 執行此命令以驗證CodeCommit指令用於AWS CLI已安裝。

```
aws codecommit help
```

此命令返回一個列表CodeCommit命令。

3. 配置AWS CLI使用設定檔configure命令，如下所示：。

```
aws configure
```

出現提示時，請指定AWS存取金鑰和AWS要搭配使用的 IAM 使用者的秘密存取金鑰 CodeCommit。此外，請務必指定AWS 區域存放庫所在的位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

```
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
```

如需建立和設定與 AWS CLI 搭配使用之描述檔的詳細資訊，請參閱下列內容：

- 命名設定檔
- 在中使用 IAM 角色AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源AWS 區域，您必須重新設定AWS CLI使用默認的地區名稱。CodeCommit 支援的預設區域名稱包括：

- 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

如需 CodeCommit 和 AWS 區域 的詳細資訊，請參閱「區域和 Git 連線端點」。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得認證？和管理 IAM 使用者的存取金鑰。如需 AWS CLI 和描述檔的詳細資訊，請參閱具名描述檔。

步驟 2：安裝 Git

若要使用中的檔案、認可和其他資訊CodeCommit存儲庫中，您必須在本地計算機上安裝 Git。CodeCommit 支援 Git 1.7.9 版和更新版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

要安裝 Git，我們建議網站如Git 下載。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時，一項功能的改變，可能會影響此功能與 CodeCommit 運作的方式。如果您在特定版本的 Git 和 CodeCommit 方面遇到問題，請檢閱疑難排解中的資訊。

步驟 3：設置憑證助手

1. 從終端機，使用 Git 執行 git config，並指定使用 Git 登入資料協助程式搭配 AWS 登入資料描述檔，同時讓 Git 登入資料協助程式將路徑傳送至儲存庫：

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

> **Tip**
>
> 憑證助手使用默認值AWS登入資料設定檔或 Amazon EC2 執行個體角色。如果您已建立 AWS 登入資料描述檔來用於 CodeCommit，則可以指定要使用的描述檔，例如 `CodeCommitProfile`：
>
> ```
> git config --global credential.helper '!aws --profile CodeCommitProfile
>  codecommit credential-helper $@'
> ```

如果描述檔名稱包含空格，請務必以引號 (") 括住名稱。
您可以使用 `--local` 設定每個儲存庫的描述檔，而非使用 `--global` 來全體設定。

Git 登入資料協助程式會將下列值寫入 `~/.gitconfig`：

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

> **Important**
>
> 如果您想要在相同的本機電腦上使用不同的 IAM 使用者CodeCommit，您必須執行git config再次命令並指定不同的AWS認證設定檔。

2. 執行 git config --global --edit 以確認上述值已寫入 `~/.gitconfig`。如果成功，您應該會看到上述值 (除了可能已存在於 Git 全域組態檔案中的值之外)。若要結束，您通常會輸入 **:q**，然後按 Enter 鍵。

   如果您在設定登入資料協助程式之後遇到問題，請參閱疑難排解。

> **Important**
>
> 如果您使用的是 macOS，請按照下列步驟確保認證協助程式設定正確。

3. 如果您使用的是 macOS，請使用 HTTPS連接到CodeCommit儲存庫。第一次透過 HTTPS 連線到 CodeCommit 儲存庫之後，大約 15 分鐘之後，後續存取會失敗。macOS 上的預設 Git 版本會使用「鑰匙圈存取」公用程式來儲存認證。基於安全考量，為存取 CodeCommit 儲存庫而產生的密碼是臨時性，因此，存放在金鑰鏈中的登入資料大約 15 分鐘之後會失效。為避免使用這些過期的登入資料，您必須：

   - 安裝一個依預設不使用金鑰鏈的 Git 版本。
   - 將 Keychain Access 公用程式設定為不提供登入資料給 CodeCommit 儲存庫。

   1. 開啟 Keychain Access 公用程式。(您可以使用 Finder 找到它。)
   2. 搜尋 `git-codecommit.us-east-2.amazonaws.com`。反白此列，開啟操作功能表或以滑鼠右鍵按一下此列，然後選擇 Get Info (取得資訊)。

3. 選擇 Access Control (存取控制) 標籤。
4. 在 Confirm before allowing access (允許存取之前確認) 中，選擇 `git-credential-osxkeychain`，然後選擇減號從清單移除它。

> **Note**
>
> 從清單移除 `git-credential-osxkeychain` 之後，每當您執行 Git 命令時就會看到快顯訊息。選擇 Deny (拒絕) 以繼續。如果您覺得快顯帶來太多干擾，以下是一些其他選項：
>
> - 使用 SSH (而不是 HTTPS) 連接到 CodeCommit。如需詳細資訊，請參閱Linux Linux。
> - 在「鑰匙圈存取」公用程式中，於存取控制索引標籤`git-codecommit.us-east-2.amazonaws.com`，選擇允許所有應用程式存取此項目 (存取此項目不受限制)選項。這可防止快顯，但登入資料最終仍會過期 (平均大約 15 分鐘)，您將會看到 403 錯誤訊息。發生這種情況時，您必須刪除金鑰鏈項目才能還原功能。
> - 如需詳細資訊，請參閱適用於 macOS 的 Git：我成功設定登入資料協助程式，但現在拒絕我存取儲存庫 (403)。

步驟 4：連接到CodeCommit控制台並克隆存儲庫

如果管理員已將 CodeCommit 儲存庫的名稱和連線詳細資訊傳送給您，則您可以略過此步驟，並直接複製儲存庫。

連接到 CodeCommit 儲存庫

1. 打開CodeCommit控制台在https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選擇器中，選擇AWS 區域存放庫的建立位置。存儲庫特定於AWS 區域。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要將 Git 認證與 IAM 使用者搭配使用，或是隨附的認證協助程式使用，請複製 HTTPS 網址AWS CLI。
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。

- 如果您將 SSH 公開/私密金鑰配對與 IAM 使用者搭配使用，請複製 SSH URL。

> **Note**
>
> 如果您看到歡迎頁面而不是存儲庫列表，沒有與您的相關聯的存儲庫AWS帳戶中的AWS區域您登入的位置。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 開啟終端機並使用您複製的 HTTPS URL 來執行 git clone 命令。例如，若要複製名為的儲存庫*MyDemoRepo*到一個名為的本地回購*my-demo-repo*在美國東部 (俄亥俄) 地區：

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

後續步驟

您已完成事前準備。按照中的步驟操作開始使用 CodeCommit 開始使用CodeCommit。

HTTPS 連線的設定步驟AWS CodeCommit使用視窗上的儲存庫AWS CLI憑證助手

第一次連接到 AWS CodeCommit 之前，您必須先完成初始設定步驟。對於大多數使用者，最輕鬆的方式是遵循適用於使用 Git 認證的 HTTPS 使用者中的步驟。不過，如果您想要使用根帳戶、聯合存取或暫時性登入資料來連線到 CodeCommit，您可以使用 AWS CLI 隨附的登入資料協助程式。

> **Note**
>
> 雖然登入資料協助程式是可支援使用聯合存取、身分提供者或暫時性登入資料連線到 CodeCommit 的方法，但建議採用的方法是安裝並使用 git-remote-codecommit 公用程式。如需詳細資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

本主題將引導您完成安裝AWS CLI，設置您的計算機和AWS設定檔, 連線到CodeCommit存儲庫，並將該存儲庫克隆到您的計算機上，也稱為創建本地存儲庫。如果您是初次使用 Git，您可能還需要檢閱我可以在哪裡進一步了解 Git？中的資訊。

主題

步驟 1：初始配置CodeCommit

請依照下列步驟設定 Amazon 網路服務帳戶、建立和設定 IAM 使用者，以及安裝AWS CLI。AWS CLI 包含登入資料協助程式，供您設定以透過 HTTPS 連線到 CodeCommit 儲存庫。

若要建立和設定 IAM 使用者以存取CodeCommit

1. 通過轉到創建一個亞馬遜網絡服務帳戶http://aws.amazon.com並選擇立即註冊。
2. 在您的亞馬遜網路服務帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的亞馬遜網絡服務帳戶中創建 IAM 用戶。

   > **Note**
   >
   > CodeCommit 需要 AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS所需的動作CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇使用者，然後選擇您要設定的 IAM 使用者CodeCommit訪問。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從政策清單中，選取 AWSCodeCommitPowerUser，或另一個用於存取 CodeCommit 的受管政策。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的策略之後，請選擇下一個:評論以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

如需有關 CodeCommit 受管政策及分享儲存庫存取權給其他群組和使用者的詳細資訊，請參閱共用儲存庫和 AWS CodeCommit 的身分驗證與存取控制。

安裝及設定 AWS CLI

1. 在本機電腦上，下載並安裝 AWS CLI。這是從命令列與 CodeCommit 互動的必要步驟。我們建議您安裝AWS CLI版本 2. 它是最新的主要版本AWS CLI並支持所有最新功能。這是唯一的版本AWS CLI支援使用 root 帳戶、同盟存取或臨時登入資料git-remote-codecommit。

   如需詳細資訊，請參閱開始設定 AWS 命令列界面。

   > **Note**
   >
   > CodeCommit僅適用於AWS CLI版本 1.7.38 及更高版本。根據最佳實務，安裝 AWS CLI 或將其升級到可用的最新版本。若要判斷您已安裝的 AWS CLI 版本，請執行 aws --version 命令。
   > 若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 執行此命令以驗證CodeCommit的指令AWS CLI已安裝。

   ```
   aws codecommit help
   ```

   此命令返回一個列表CodeCommit命令。

3. 配置AWS CLI使用設定檔configure命令，如下所示：。

   ```
   aws configure
   ```

   出現提示時，請指定AWS存取金鑰和AWS要搭配使用的 IAM 使用者的秘密存取金鑰CodeCommit。此外，請務必指定AWS 區域存放庫所在的位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

   ```
   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
```

如需建立和設定與 AWS CLI 搭配使用之描述檔的詳細資訊，請參閱下列內容：

- 命名設定檔
- 在中使用 IAM 角色AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源AWS 區域，您必須重新設定AWS CLI使用默認的地區名稱。CodeCommit 支援的預設區域名稱包括：

- 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

如需 CodeCommit 和 AWS 區域 的詳細資訊，請參閱「區域和 Git 連線端點」。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得認證？和管理 IAM 使用者的存取金鑰。如需 AWS CLI 和描述檔的詳細資訊，請參閱具名描述檔。

步驟 2：安裝 Git

若要使用中的檔案、認可和其他資訊CodeCommit存儲庫中，您必須在本地計算機上安裝 Git。CodeCommit 支援 Git 1.7.9 版和更新版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

要安裝 Git，我們建議網站，如Git 的視窗。如果您使用此連結來安裝 Git，您可以接受所有安裝預設設定，但下列項目除外：

- 在期間出現提示時調整您的路徑環境步驟，選擇從命令行中使用 Git 的選項。
- (選擇性) 如果您打算搭配使用 HTTPS 搭配中包含的認證協助程式AWS CLI而不是配置 Git 憑據 CodeCommit，在「」設定額外選項頁面上，請確定啟用 Git 憑證管理器選項已清除。Git 憑據管理器僅兼容CodeCommit如果 IAM 用戶配置 Git 憑據。如需詳細資訊，請參閱 適用於使用 Git 認證的 HTTPS 使用者 及 適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時，一項功能的改變，可能會影響此功能與 CodeCommit 運作的方式。如果您在特定版本的 Git 和 CodeCommit 方面遇到問題，請檢閱疑難排解中的資訊。

步驟 3：設置憑證助手

AWS CLI 包含可與 CodeCommit 搭配使用的 Git 登入資料協助程式。Git 憑證協助程式需要AWS 認證設定檔，其中儲存 IAM 使用者的副本AWS存取金鑰 ID 和AWS秘密訪問密鑰（以及默認值AWS 區域名稱和默認輸出格式）。Git 憑證助手使用此信息自動進行身份驗證CodeCommit所以你不需要每次使用 Git 進行交互時輸入這些信息CodeCommit。

1. 打開命令提示符並使用 Git 運行git config中，指定 Git 認證協助程式的使用方式AWS憑據配置文件，它使 Git 憑據幫助程序將路徑發送到存儲庫：

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

Git 登入資料協助程式會將下列資料寫入 .gitconfig 檔案：

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

> **Important**
>
> - 如果您使用的是 Bash 模擬器，而不是 Windows 命令列，您必須使用單引號而不是雙引號。
> - 憑證協助程式使用預設值AWS設定檔或亞馬遜 EC2 執行個體角色。如果您已建立要使用的 AWS 登入資料描述檔，例如 *CodeCommitProfile*，則可以如下修改命令來改用此描述檔：
>
> ```
> git config --global credential.helper "!aws codecommit credential-helper
>  --profile CodeCommitProfile $@"
> ```
>
> 這會將以下資料寫入 .gitconfig 檔案：
>
> ```
> [credential]
>     helper = !aws codecommit credential-helper --profile=CodeCommitProfile
>  $@
>     UseHttpPath = true
> ```

- 如果描述檔名稱包含空格，則執行此命令後，您必須編輯 .gitconfig 檔案，以單引號 (') 括住它。否則登入資料協助程式無法運作。
- 如果您安裝的 Git for Windows 包含 Git Credential Manager 公用程式，則在最先幾次嘗試連線之後，您會看到 403 錯誤，或提示您將登入資料提供給 Credential Manager 公用程式。解決此問題最可靠的方式是解除安裝 Git for Windows，再重新安裝，但不要選擇 Git Credential Manager 公用程式這個選項，因為此選項與 CodeCommit 不相容。如果您想要保留 Git Credential Manager 公用程式，則必須執行額外的設定步驟來一併使用 CodeCommit，包括手動修改 .gitconfig 檔案，以指定在連接到 CodeCommit 時，對 AWS CodeCommit 使用登入資料協助程式。從 Credential Manager 公用程式移除任何已儲存的登入資料 (您可以在控制台找到此公用程式)。在移除任何已儲存的登入資料後，請將以下資料新增到 .gitconfig 檔案、儲存檔案，然後從新的命令提示字元視窗中重試連接：

  ```
  [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
  ```

  您也可能需要指定 --system (而不是 --global 或 --local) 以重新設定 git config 的設定，所有連線才能正常運作。
- 如果您想要在相同的本機電腦上使用不同的 IAM 使用者CodeCommit，您應該指定 git config --local而不是git config --global，並為每個運行配置AWS認證設定檔。

2. 執行 git config --global --edit，以確認上述值已寫入使用者描述檔的 .gitconfig 檔案 (預設為 `%HOME%\.gitconfig` 或 *`drive`*`:\Users\`*`UserName`*`\.gitconfig`)。如果成功，您應該會看到上述值 (除了可能已存在於 Git 全域組態檔案中的值之外)。若要結束，您通常會輸入 **`:q`**，然後按 Enter 鍵。

步驟 4：連接到CodeCommit控制台並克隆存儲庫

如果管理員已將 CodeCommit 儲存庫的名稱和連線詳細資訊傳送給您，則您可以略過此步驟，並直接複製儲存庫。

連接到 CodeCommit 儲存庫

1. 打開CodeCommit控制台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選擇器中，選擇AWS 區域存放庫的建立位置。存儲庫特定於AWS 區域。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要將 Git 認證與 IAM 使用者搭配使用，或是隨附的認證協助程式使用，請複製 HTTPS URL。AWS CLI。
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。
   - 如果您將 SSH 公開/私密金鑰配對與 IAM 使用者搭配使用，請複製 SSH URL。

   > **Note**
   >
   > 如果您看到歡迎頁面而不是存儲庫列表，沒有與您的相關聯的存儲庫AWS帳戶中的AWS區域您登入的位置。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. 開啟命令提示字元並執行git clone使用您複製的 HTTPS 網址的命令。在您執行此命令的目錄中，將會在該目錄的子目錄中建立本機儲存庫。例如，若要複製名為的儲存庫*MyDemoRepo*到一個名為的本地回購*my-demo-repo*在美國東部 (俄亥俄) 地區：

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

   在某些版本的 Windows 上，您可能會看到快顯訊息，要求您輸入使用者名稱和密碼。這是 Windows 內建的登入資料管理系統，但與 AWS CodeCommit 的登入資料協助程式不相容。選擇 Cancel (取消)。

後續步驟

您已完成事前準備。按照中的步驟操作開始使用 CodeCommit 開始使用CodeCommit。

入門

開始使用 CodeCommit 最簡單的方法是遵循開始使用 CodeCommit 。如果您是初次使用 Git 和 CodeCommit，您也應該考慮遵循開始使用 Git 和 CodeCommit。這有助於您熟悉，以及使用 Git 來與 CodeCommit 儲存庫互動時的基本知識。

您也可以按照和的簡易管道演練，了解如何在持續交付管道中使用 CodeCommit 儲存庫。

本節中的教學課程假設您已完成先決條件和設定，包括：

- 將許可指派給使用者。
- 在您用於本教學課程的本機機器上，設定 HTTPS 或 SSH 連線的登入資料管理。
- 如果您想要使用命令列或終端機執行所有操作 (包括建立儲存庫)，請設定 AWS CLI。

主題

- 開始使用 AWS CodeCommit
- 開始使用 Git 和 AWS CodeCommit

開始使用 AWS CodeCommit

本教程向您展示如何使用一些關鍵 CodeCommit 功能。首先，您建立儲存庫並遞交一些變改給它。然後，您瀏覽檔案，並檢視變更。您也可以建立提取請求，讓其他使用者可以檢閱程式碼的變更並作評論。

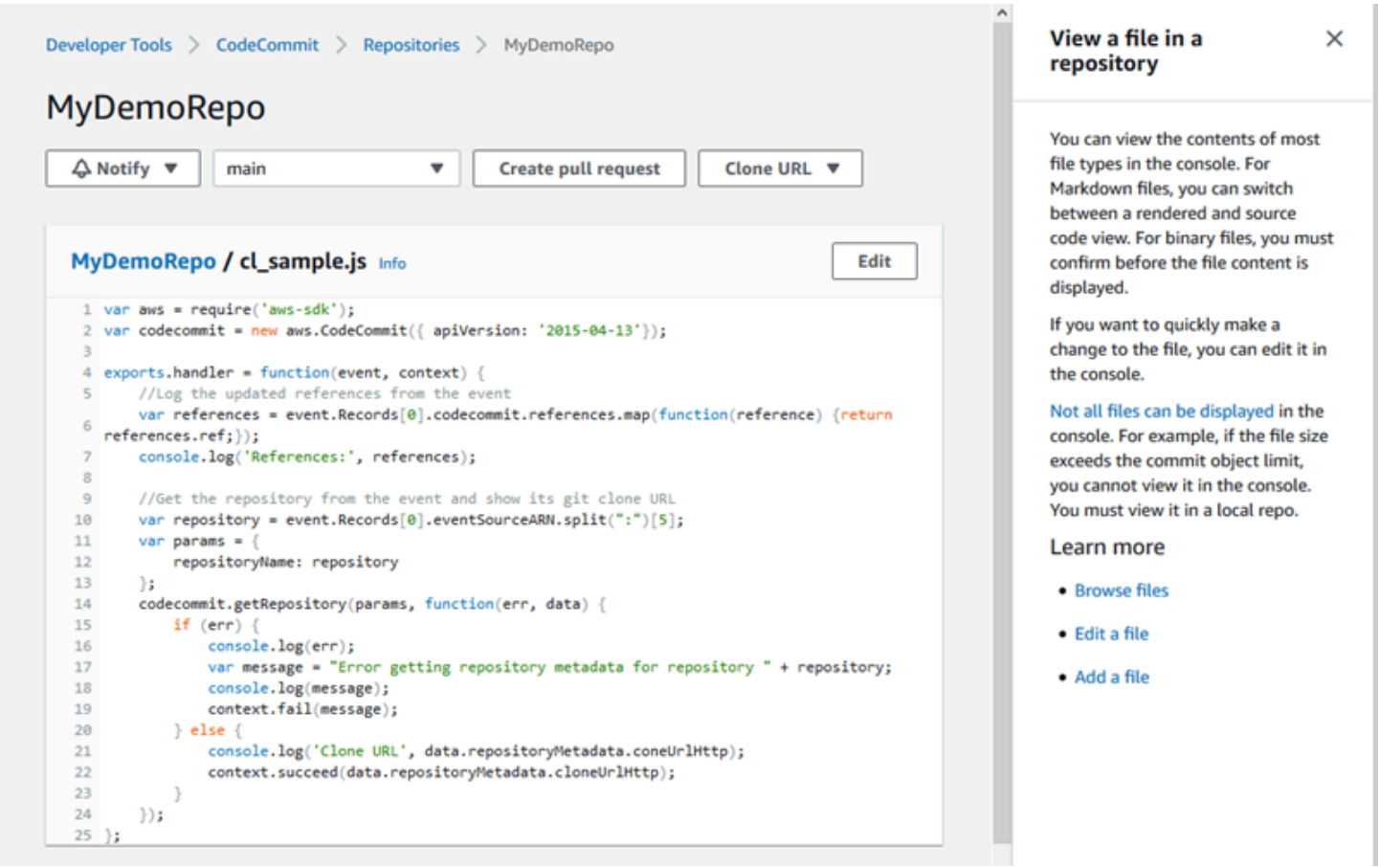
如果您要將現有程式碼移轉至 CodeCommit，請參閱遷移至 AWS CodeCommit。

如果你不熟悉 Git，也可以考慮完成開始使用 Git 和 CodeCommit。完成這些自學課程之後，您應該有足夠的練習開始用 CodeCommit 於您自己的專案和團隊環境中。

CodeCommit 主控台在可摺疊面板中包含實用資訊，您可以從資訊圖示

()

或頁面上的任何「資訊」連結開啟這些資訊。您隨時皆可關閉此面板。



主 CodeCommit 控台也提供一種快速搜尋資源的方法，例如儲存庫、建置專案、部署應用程式和管道。選擇 Go to resource (移至資源)，或按 / 鍵，然後輸入資源名稱。任何相符項目都會出現在清單中。搜尋不區分大小寫。您只會看到您有權檢視的資源。如需詳細資訊，請參閱 在主控台檢視資源。

必要條件

開始之前，您必須完成先決條件和設定程序，包括：

- 將許可指派給 IAM 使用者。
- 在本教學課程所使用的本機電腦上，設定 HTTPS 或 SSH 連線的登入資料管理。
- 配置是 AWS CLI 否要使用命令行或終端機進行所有操作，包括創建存儲庫。

主題

- 步驟 1：建立 CodeCommit 儲存庫
- 步驟 2：將文件添加到存儲庫
- 步驟 3：瀏覽存放庫的內容

步驟 1：建立 CodeCommit 儲存庫

您可以使用 CodeCommit 控制台來創建一個 CodeCommit 存儲庫。如果您已經有要用於此教學的儲存庫，則可略過此步驟。

> **Note**
>
> 根據您的使用情況，您可能需要支付建立或存取存放庫的費用。如需詳細資訊，請參閱 CodeCommit 產品資訊頁面上的定價。

若要建立存 CodeCommit 放庫

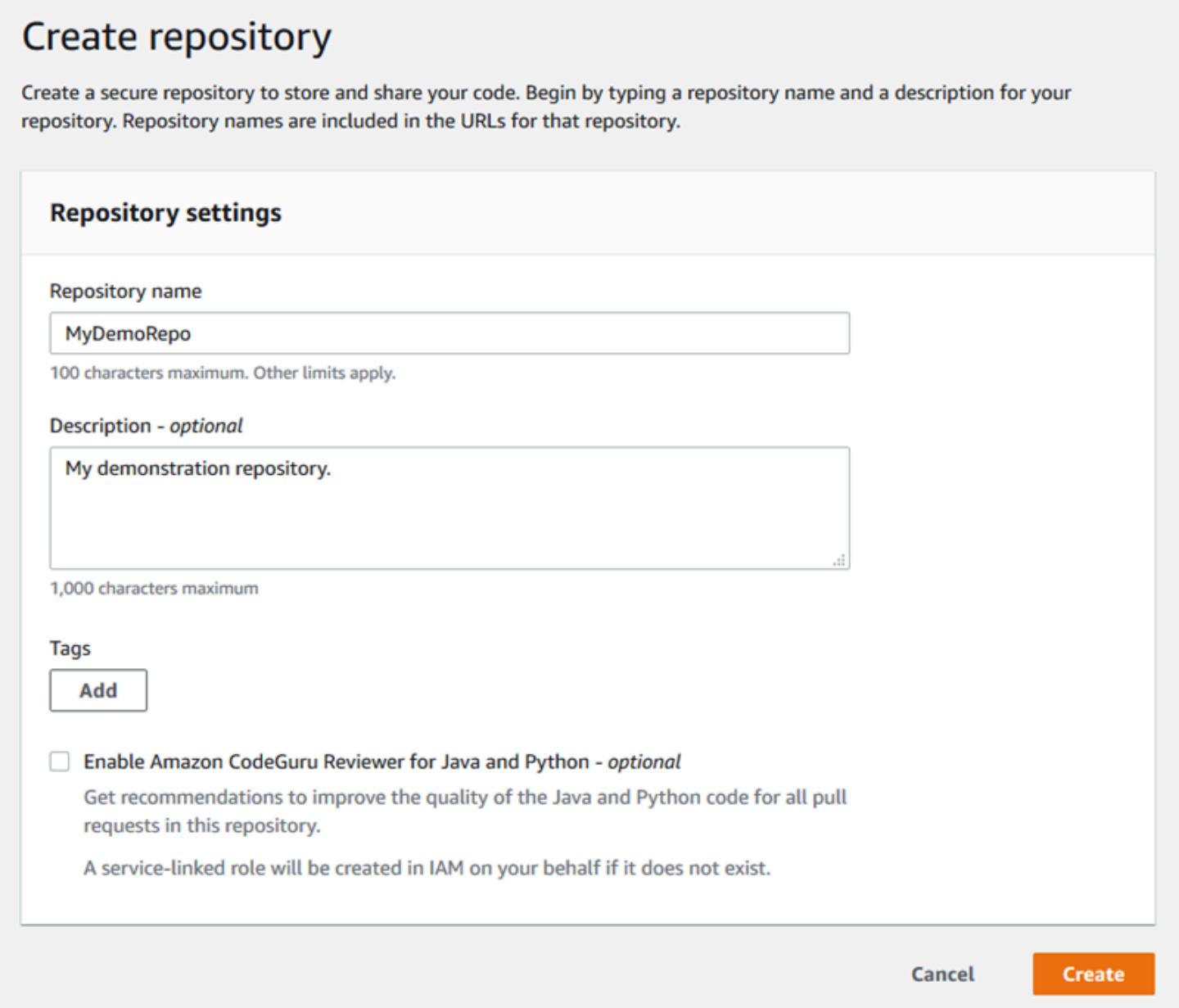
1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 使用區域選擇器來選擇您 AWS 區域 要建立儲存庫的位置。如需詳細資訊，請參閱 區域和 Git 連線端點。
3. 請在 Repositories (儲存庫) 頁面上，選擇 Create repository (建立儲存庫)。
4. 在 Create repository (建立儲存庫) 頁面的 Repository name (儲存庫名稱) 中，輸入儲存庫的名稱 (例如 **`MyDemoRepo`**)。

   > **Note**
   >
   > 儲存庫名稱需區分大小寫且不能超過 100 個字元。如需詳細資訊，請參閱限制。

5. (選用) 在 Description (描述) 中，輸入描述 (例如，**`My demonstration repository`**)。這可協助您和其他使用者識別儲存庫的用途。
6. (選擇性) 選擇「新增標籤」，將一或多個儲存庫標籤 (可協助您組織及管理 AWS 資源的自訂屬性標籤) 新增至儲存庫。如需詳細資訊，請參閱 標記儲存庫 AWS CodeCommit。
7. (選擇性) 展開其他組態以指定是使用預設金鑰 AWS 受管金鑰 還是您自己的客戶管理金鑰來加密和解密此儲存庫中的資料。如果您選擇使用自己的客戶管理金鑰，則必須確定該金鑰可在您

建立儲存庫的 AWS 區域 位置使用，且金鑰處於作用中狀態。如需詳細資訊，請參閱 AWS Key Management Service和AWS CodeCommit存儲庫的加密。

8. (選擇性) 如果此儲存庫將包含 Java 或 Python 程式碼，且您想要讓 CodeGuru 審核者分析該程式碼，請選取啟用 Java 和 Python 的 Amazon CodeGuru 審核者。 CodeGuru Reviewer 使用多個機器學習模型來尋找程式碼瑕疵，並在提取要求中自動建議改進和修正。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南。
9. 選擇建立。



> **Note**
>
> 如果您的儲存庫使用 MyDemoRepo 以外的名稱，請務必在剩餘的步驟中使用該名稱。

存放庫開啟時，您會看到如何直接從 CodeCommit 主控台新增檔案的相關資訊。

步驟 2：將文件添加到存儲庫

您可以透過以下方式，將檔案新增至儲存庫：

- 在 CodeCommit 控制台中創建文件。如果您在控制台中為存儲庫創建第一個文件，則會為您創建一個名為 main 的分支。此分支是儲存庫的預設分支。
- 使用 CodeCommit 主控台從本機電腦上傳檔案。如果您從控制台上傳存儲庫的第一個文件，則會為您創建一個名為 main 的分支。此分支是儲存庫的預設分支。
- 使用 Git 客戶端將存儲庫克隆到本地計算機，然後將文件添加，提交和推送到 CodeCommit 存儲庫。系統會為您建立一個分支，作為 Git 第一次提交的一部分，並將其設定為儲存庫的預設分支。分支的名稱是 Git 客戶端的默認選擇。考慮將 Git 客戶端配置為使用 main 作為初始分支的名稱。

> **Note**
>
> 您可以隨時建立分支並變更儲存庫的預設分支。如需詳細資訊，請參閱 使用 AWS CodeCommit 儲存庫中的分支。

開始使用最簡單的方法是開啟主 CodeCommit 控台並新增檔案。這樣，您還可以為名為 main 的存儲庫創建一個默認分支。如需有關如何使用新增檔案並建立第一次提交至儲存庫的指示 AWS CLI，請參閱使用 AWS CLI.

將檔案新增至儲存庫

1. 在儲存庫的導覽列中，選擇 Code (程式碼)。
2. 選擇 Add file (新增檔案)，然後選擇是否建立檔案或從電腦上傳檔案。本教學課程將告訴您如何執行這兩項操作。
3. 若要新增檔案，請執行下列操作：

   a. 在分支的下拉式清單中，選擇您要新增檔案的分支。系統會自動為您選取預設分支。在此處顯示的範例中，預設分支名為 *main*。如果要將檔案新增至不同的分支，請選擇不同的分支。

b. 在 File name (檔案名稱) 中，輸入檔案的名稱。在程式碼編輯器中，輸入檔案的程式碼。

c. 在 Author name (作者名稱) 中，輸入您要向其他儲存庫使用者顯示的名稱。

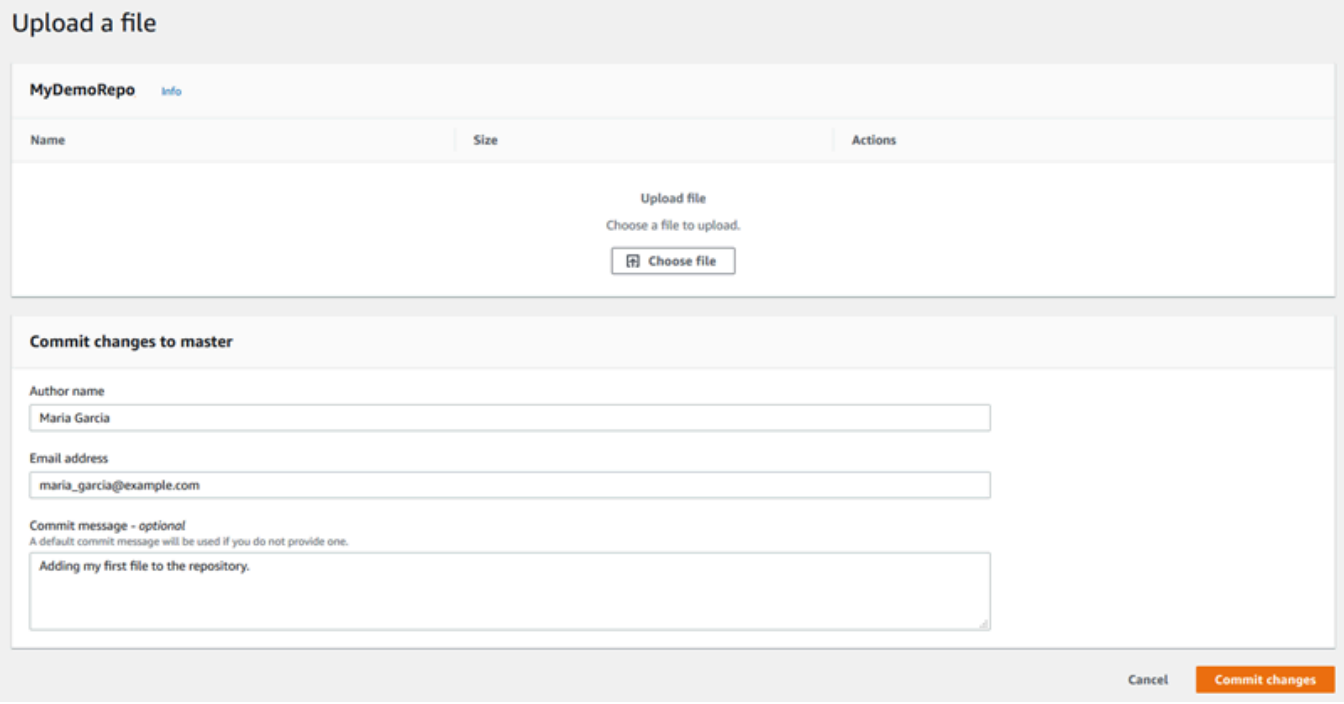
d. 在 Email address (電子郵件地址) 中，輸入電子郵件地址。

e. (選用) 在 Commit message (遞交訊息) 中，輸入簡短訊息。雖然這是選用的，但建議您新增遞交訊息，以幫助團隊成員了解您新增這個檔案的原因。如果未輸入遞交訊息，則會使用預設訊息。

f. 選擇 Commit changes (遞交變更)。

若要上傳檔案，請執行下列操作：

- 如果是上傳檔案，請選擇您要上傳的檔案。



- 在 Author name (作者名稱) 中，輸入您要向其他儲存庫使用者顯示的名稱。
- 在 Email address (電子郵件地址) 中，輸入電子郵件地址。
- (選用) 在 Commit message (遞交訊息) 中，輸入簡短訊息。雖然這是選用的，但建議您新增遞交訊息，以幫助團隊成員了解您新增這個檔案的原因。如果未輸入遞交訊息，則會使用預設訊息。
- 選擇 Commit changes (遞交變更)。

如需詳細資訊，請參閱 在中使用檔案AWS CodeCommit儲存庫。

若要使用 Git 用戶端複製儲存庫，請在本機電腦上安裝 Git，然後再複製 CodeCommit 儲存庫。將一些文件添加到本地回購並將其推送到 CodeCommit 存儲庫中。如需深入介紹，請參閱開始使用 Git 和 CodeCommit。如果您熟悉 Git，但不確定如何使用 CodeCommit 儲存庫來執行此操作，您可以在、或中檢視創建一個提交步驟 2：建立本機存放庫、或中的範例和指示連接到儲存庫。

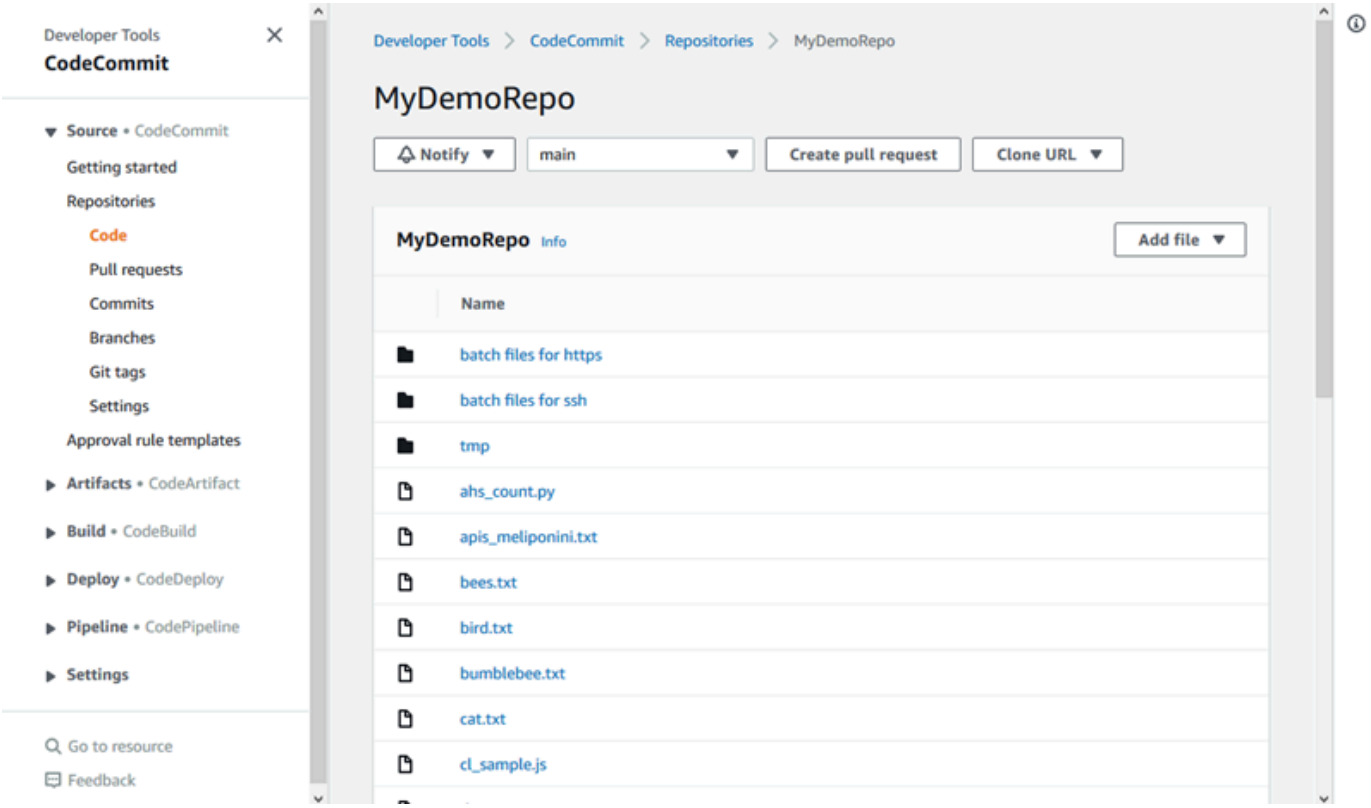
將一些檔案新增至 CodeCommit 儲存庫後，您可以在主控台中檢視這些檔案。

步驟 3：瀏覽存放庫的內容

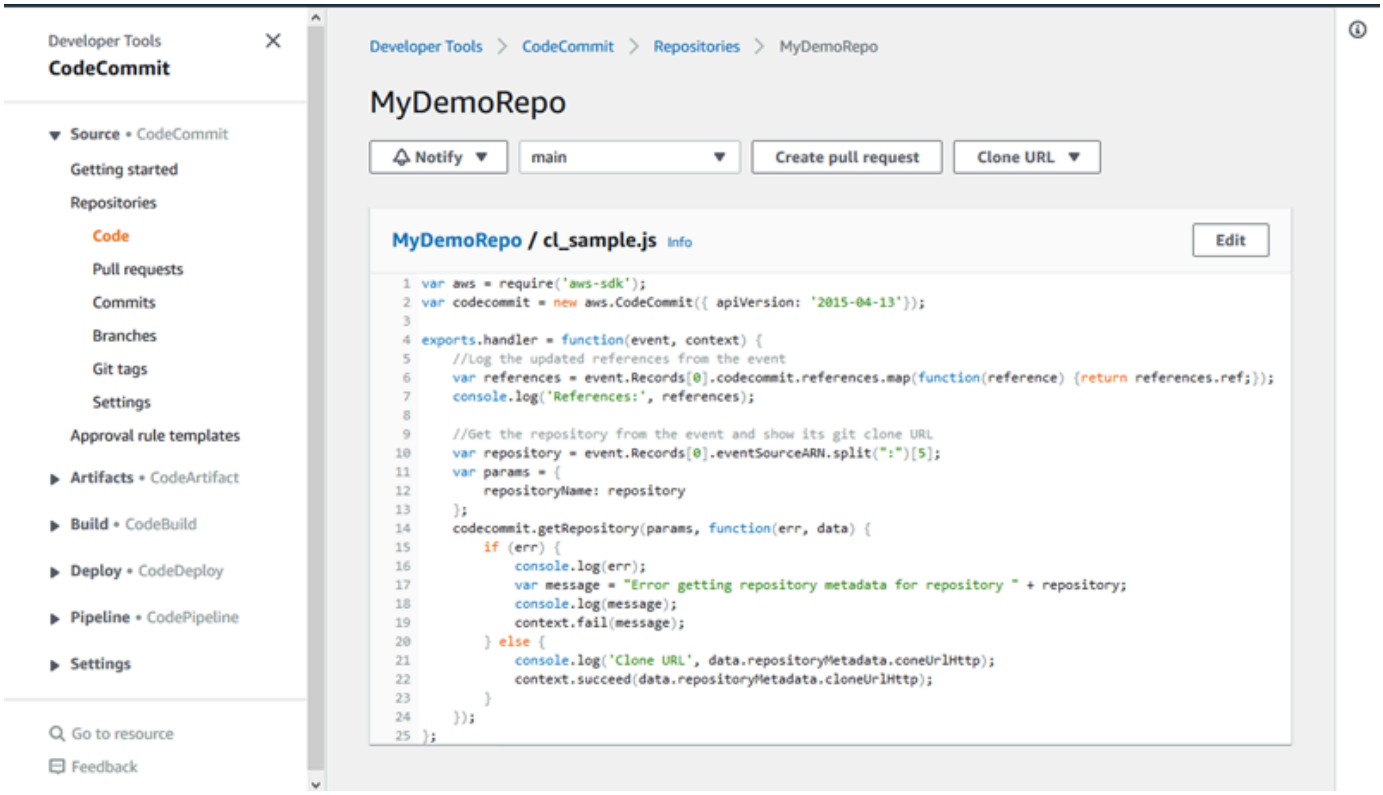
您可以使用 CodeCommit 主控台來檢閱儲存庫中的檔案，或快速讀取檔案的內容。這有助於您決定要檢查的分支，或是否建立儲存庫的本機副本。

瀏覽儲存庫

1. 從儲存庫中，選擇 MyDemoRepo。
2. 頁面會顯示儲存庫的預設分支中的內容。若要檢視其他分支或檢視特定標籤上的程式碼，請從清單中選擇您要檢視的分支或標籤。在下面的屏幕截圖中，視圖被設置為主分支。

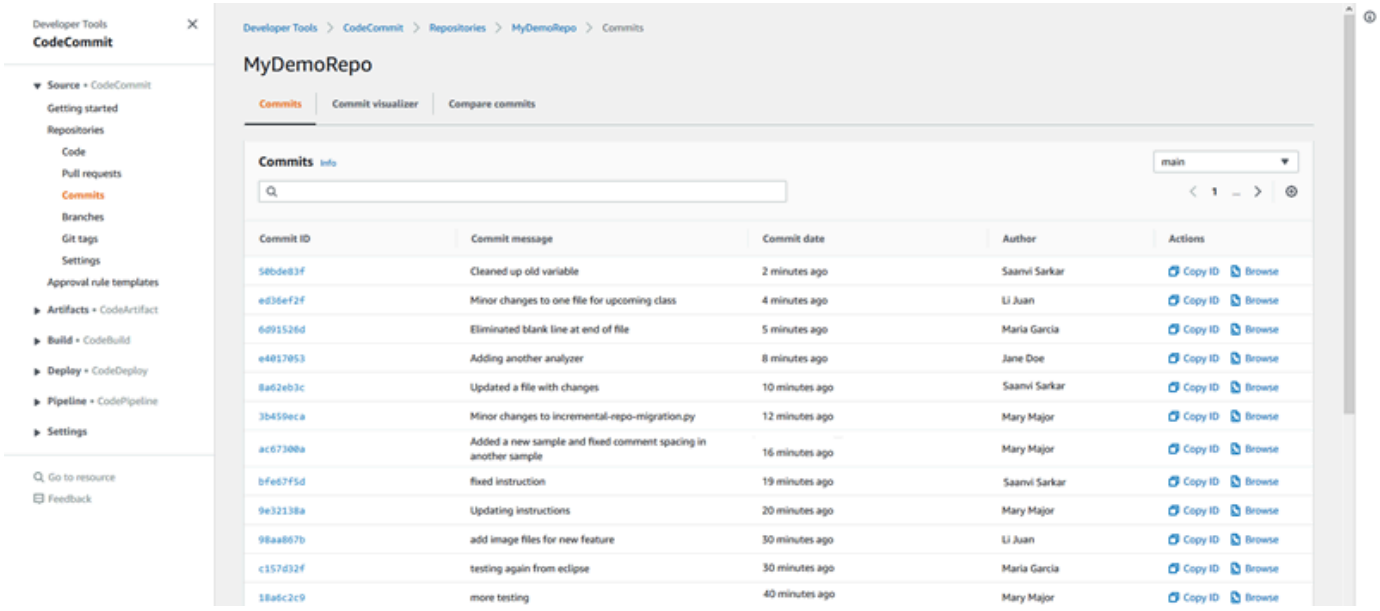


3. 若要檢視儲存庫中某個檔案的內容，請從清單中選擇檔案。若要變更顯示的程式碼顏色，請選擇設定圖示。



如需詳細資訊，請參閱 瀏覽存儲庫中的文件。

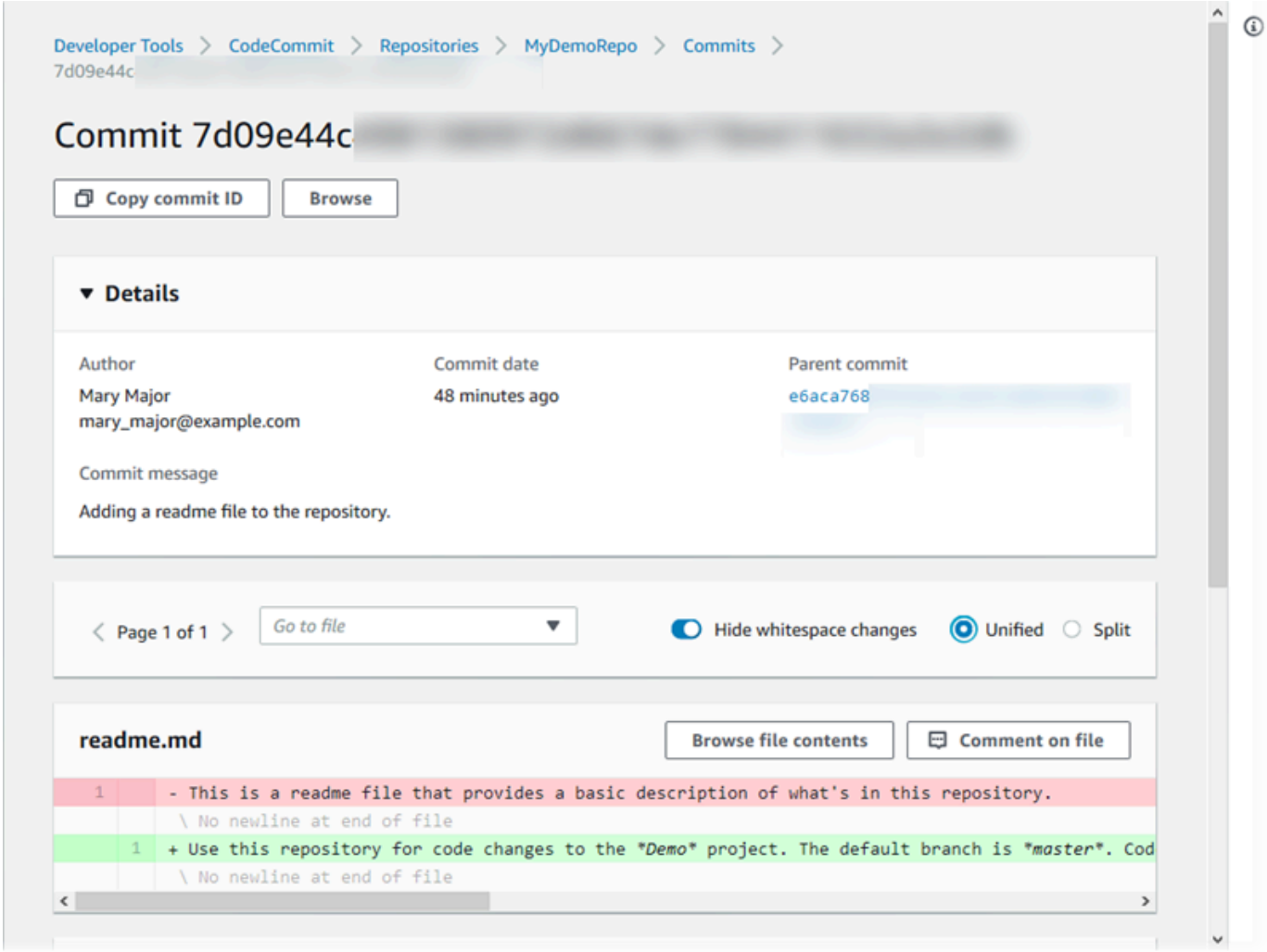
4. 若要瀏覽儲存庫的遞交歷史記錄，請選擇 Commits (遞交)。主控台會以相反的時間順序顯示預設分支的遞交歷史記錄。依作者、日期等檢閱遞交詳細資料。



5. 若要依分支或 Git 標籤檢視遞交歷史記錄，請從清單中選擇您要檢視的分支或標籤。

6. 若要檢視遞交與其父遞交之間的差異，請選擇縮寫的遞交 ID。您可以選擇如何顯示變更，包括顯示或隱藏空格變更，以及是否以內嵌 (Unified (統一)檢視) 或並排 (Split (分割)檢視) 方式來檢視變更。

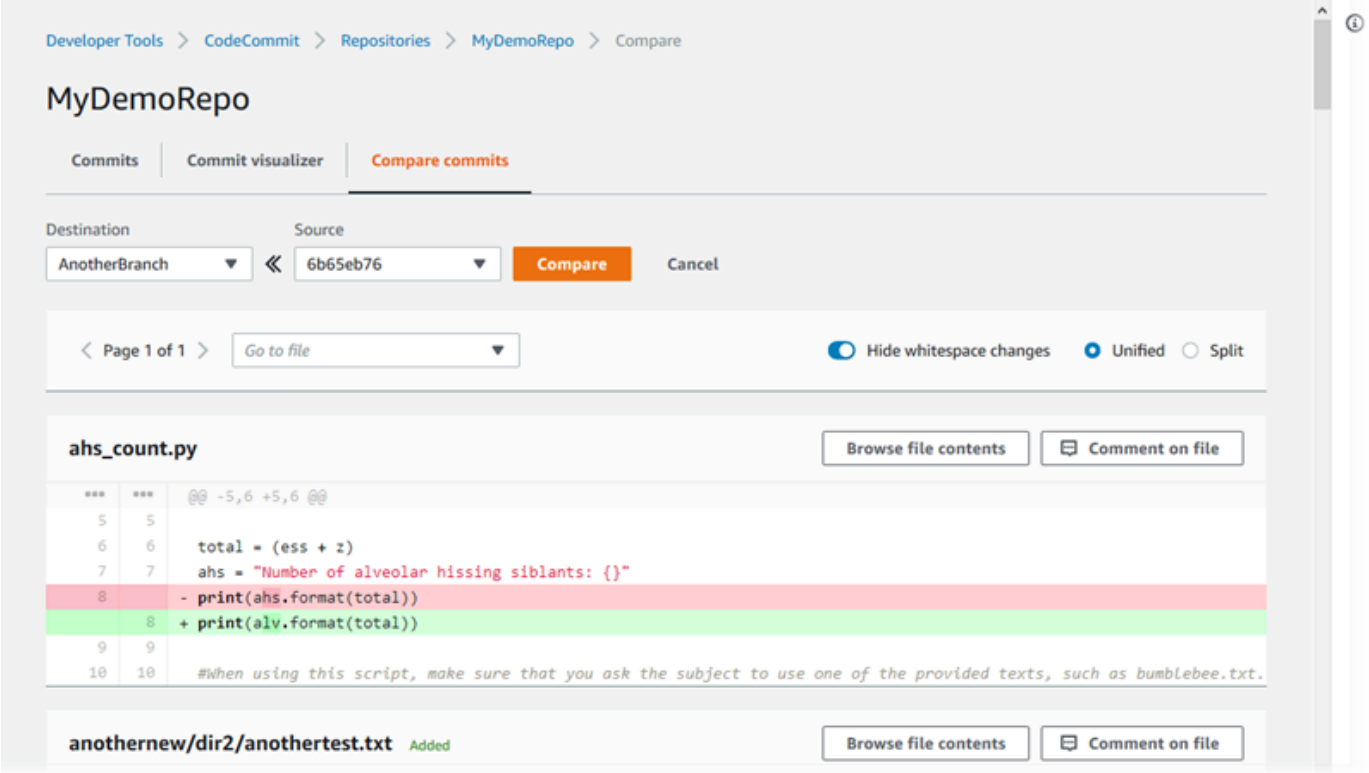
> **Note**
>
> 用於檢視程式碼的偏好設定及其他主控台設定只要有變動，就會儲存為瀏覽器 Cookie。如需詳細資訊，請參閱 使用者偏好設定。



7. 若要檢視遞交的所有註解，請選擇遞交，然後捲動變更以內嵌方式檢視。您也可以新增自己的註解，並回覆其他人所做的註解。

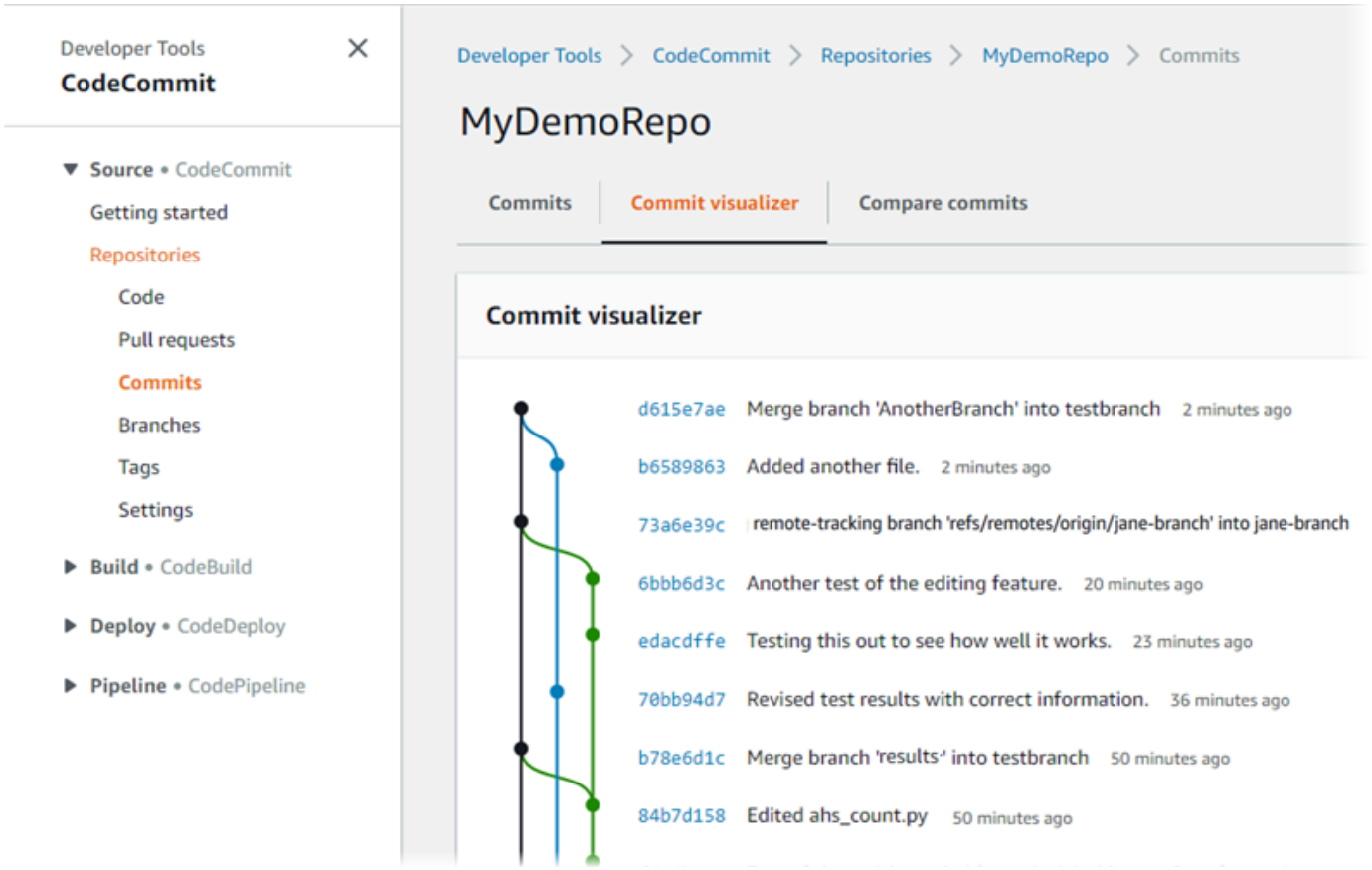
   如需詳細資訊，請參閱 對遞交做註解。

8. 若要檢視任何兩個遞交指標之間的差異，包括標籤、分支和遞交 ID，請在導覽窗格中選擇 Commits (遞交)，然後選擇 Compare commits (比較遞交)。



如需詳細資訊，請參閱 瀏覽存儲庫的提交歷史記錄 及 比較遞交。

9. 在 Commits (遞交) 中，選擇 Commit visualizer (遞交視覺化工具) 標籤。



隨即顯示遞交圖表，在圖表中，每個遞交會在該點旁邊顯示主旨列。主旨列最多顯示 80 個字元。

10. 若要查看遞交的詳細資訊，請選擇其縮寫遞交 ID。若要呈現從特定遞交開始的圖表，請在圖表中選擇該點。如需詳細資訊，請參閱 查看存儲庫提交歷史記錄的圖表 。

步驟 4：在提取請求上建立和共同作業

當您與其他使用者在儲存庫中工作時，你們可以協同處理程式碼並檢閱變更。您可以建立提取請求，讓其他使用者可以檢閱您在分支中的程式碼變更並作評論。您也可以為提取請求建立一或多個核准規則。例如，您可以建立核准規則，要求至少兩位其他使用者核准提取請求後，才能合併提取請求。核准提取請求之後，您可以將這些變更合併到目的地分支中。如果您為儲存庫設定通知，則儲存庫使用者可以收到與儲存庫事件相關的電子郵件 (例如提取請求或當有人對程式碼做註解時)。如需詳細資訊，請參閱 設定 AWS CodeCommit 存放庫中事件的通知。

> **Important**
>
> 您必須建立分支，其中包含您希望檢閱的程式碼變更，才能建立提取請求。如需詳細資訊，請參閱 建立分支。

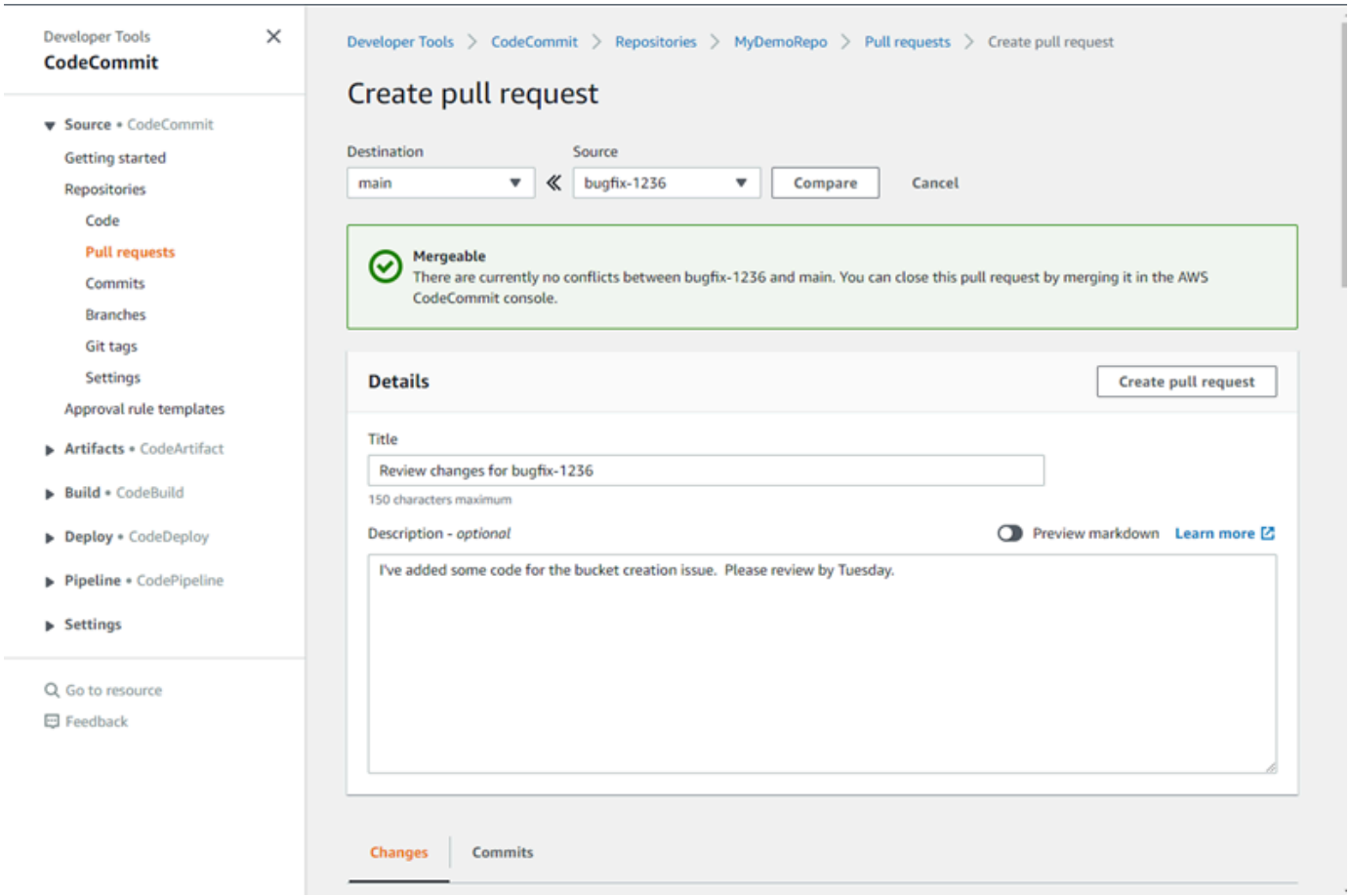
建立和協同處理提取請求

1. 在導覽窗格中，選擇 Pull requests (提取請求)。
2. 在 Pull request (提取請求) 中，選擇 Create pull request (建立提取請求)。

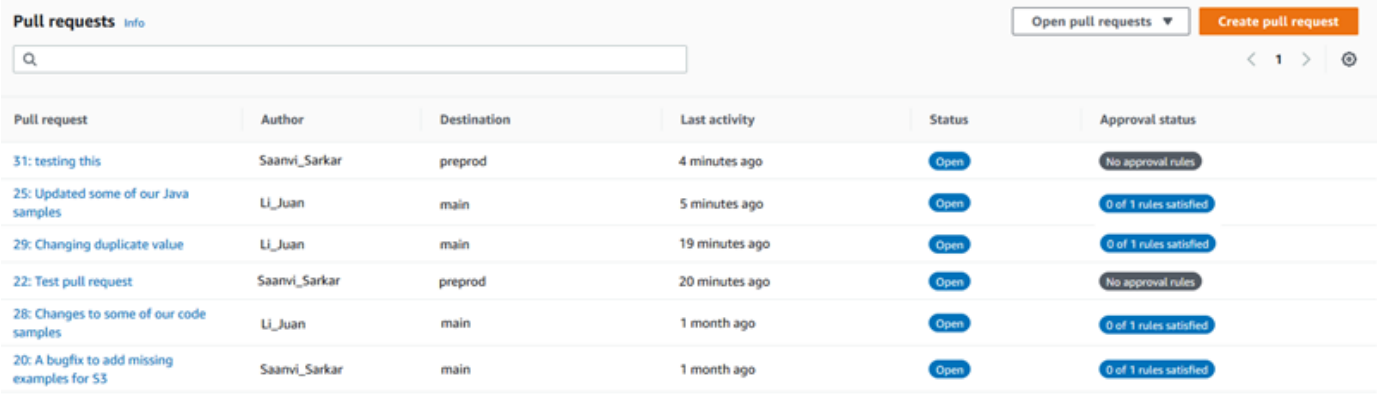
> **Tip**
>
> 您也可以從 Branches (分支) 和 Code (程式碼) 建立提取請求。

在 Create pull request (建立提取請求) 的 Source (來源) 中，選擇分支，其中包含您希望檢閱的變更。在 Destination (目的地) 中，選擇分支，表示您希望提取請求關閉時，將已檢閱的程式碼合併到此分支。選擇 Compare (比較)。

3. 檢閱合併詳細資訊和變更，以確認提取請求包含您希望檢閱的變更和遞交。如果沒問題，請在 Title (標題) 中輸入此檢閱的標題。這是在儲存庫的提取請求清單中顯示的標題。在 Description (描述) 中，輸入有助於了解此檢閱的詳細資訊，以及對檢閱者有用的任何其他資訊。選擇建立。

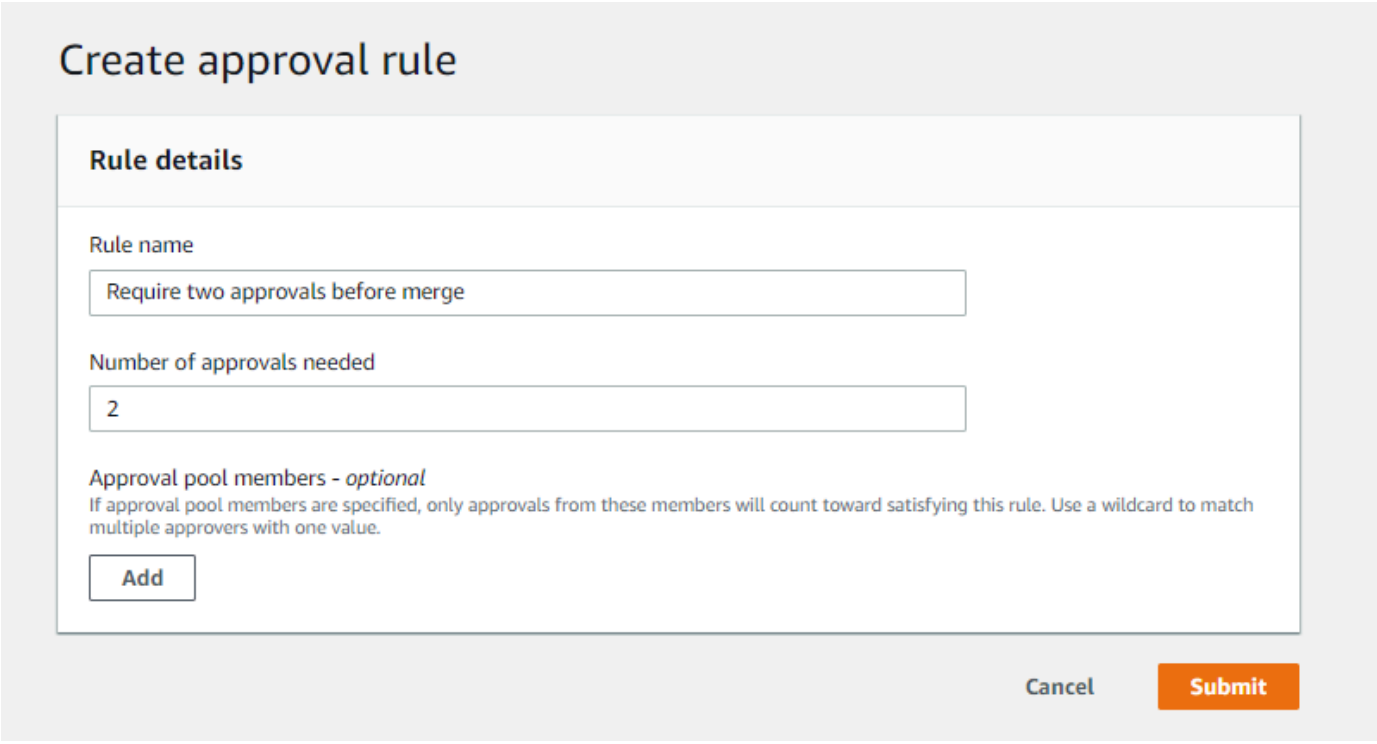


4. 您的提取請求會出現在儲存庫的提取請求清單中。您可以篩選檢視，只顯示開啟的請求、關閉的請求、您建立的請求等等。

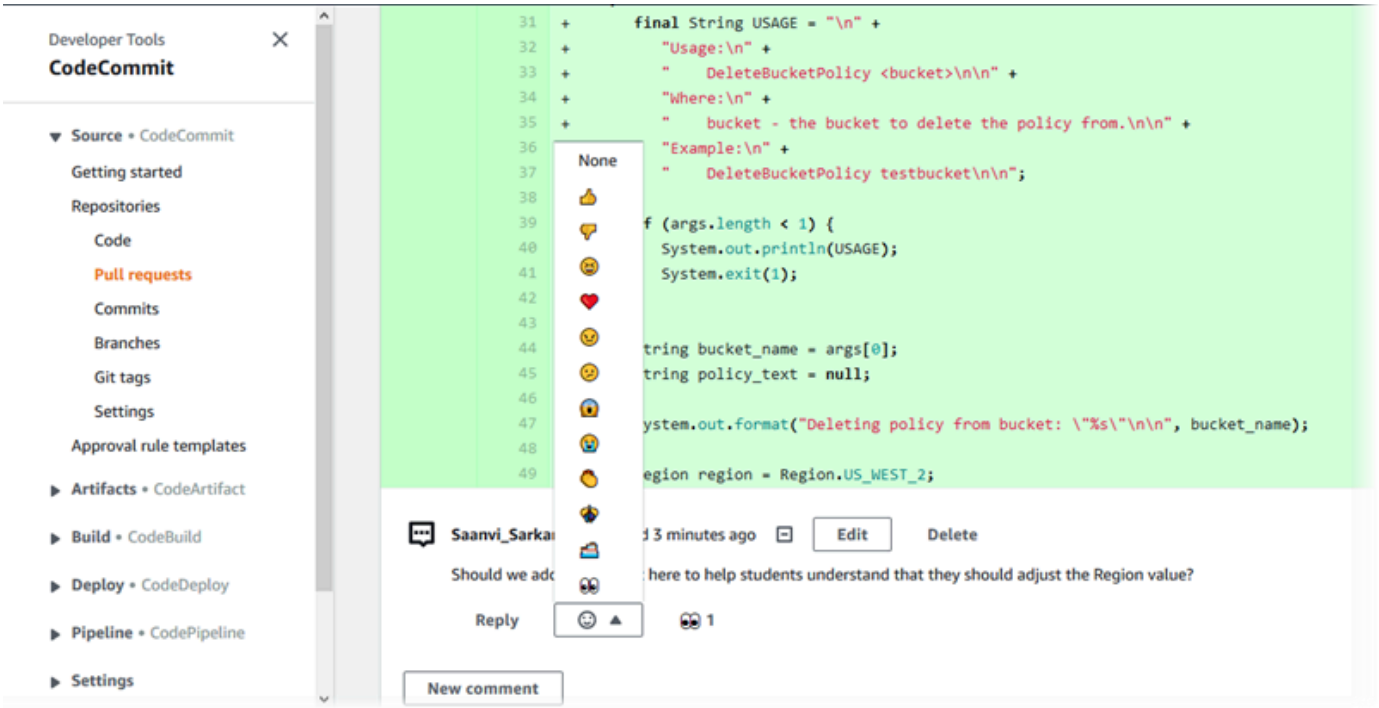


5. 您可以在提取請求中新增核准規則，以確保在合併之前符合特定條件。若要將核准規則新增至提取請求，請從清單中選擇提取請求。在 Approvals (核准) 索引標籤上，選擇 Create approval rule (建立核准規則)。

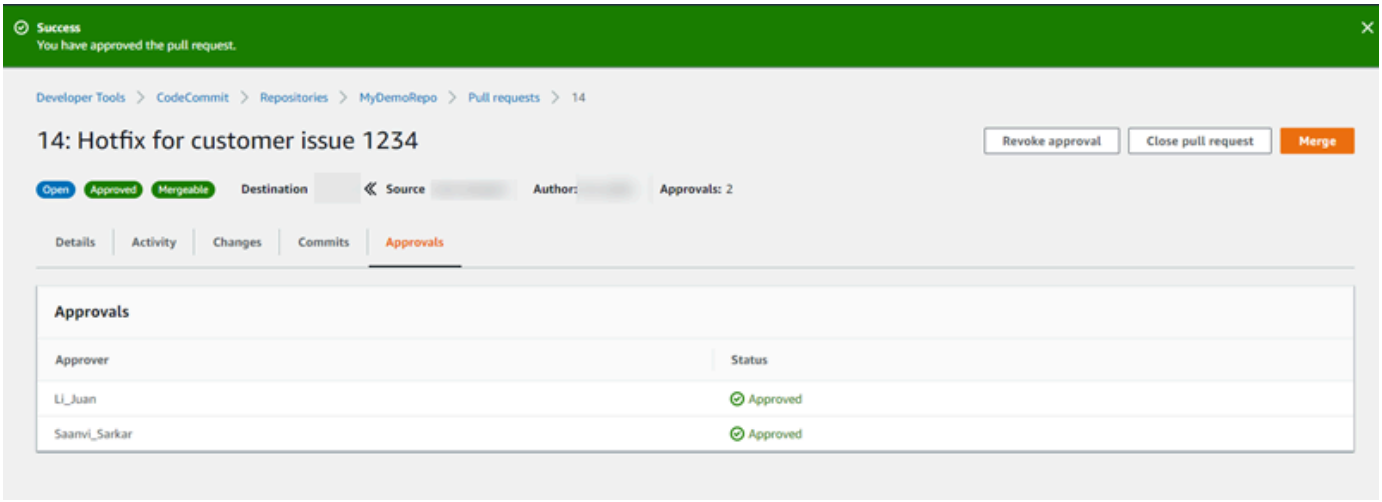
6. 在 Rule name (規則名稱) 中，為規則指定描述性名稱。例如，如果需要兩個人先核准提取請求後，才能合併提取請求，您可以將規則命名為 **`Require two approvals before merge`**。在 Number of approvals needed (需要的核准數目) 中，輸入 **2**，此為您要的數目。預設為 1。選擇提交。若要深入了解核准規則和核准集區成員，請參閱建立提取請求的核准規則。



7. 如果您為儲存庫設定通知，並選擇將提取請求事件通知使用者，則使用者會收到新提取請求的相關電子郵件。使用者可以檢視特定程式碼行、檔案和提取請求本身的變更並做註解。他們也可以使用文字和表情符號回覆留言。如有需要，您可以將變更推送到提取請求分支，以更新提取請求。



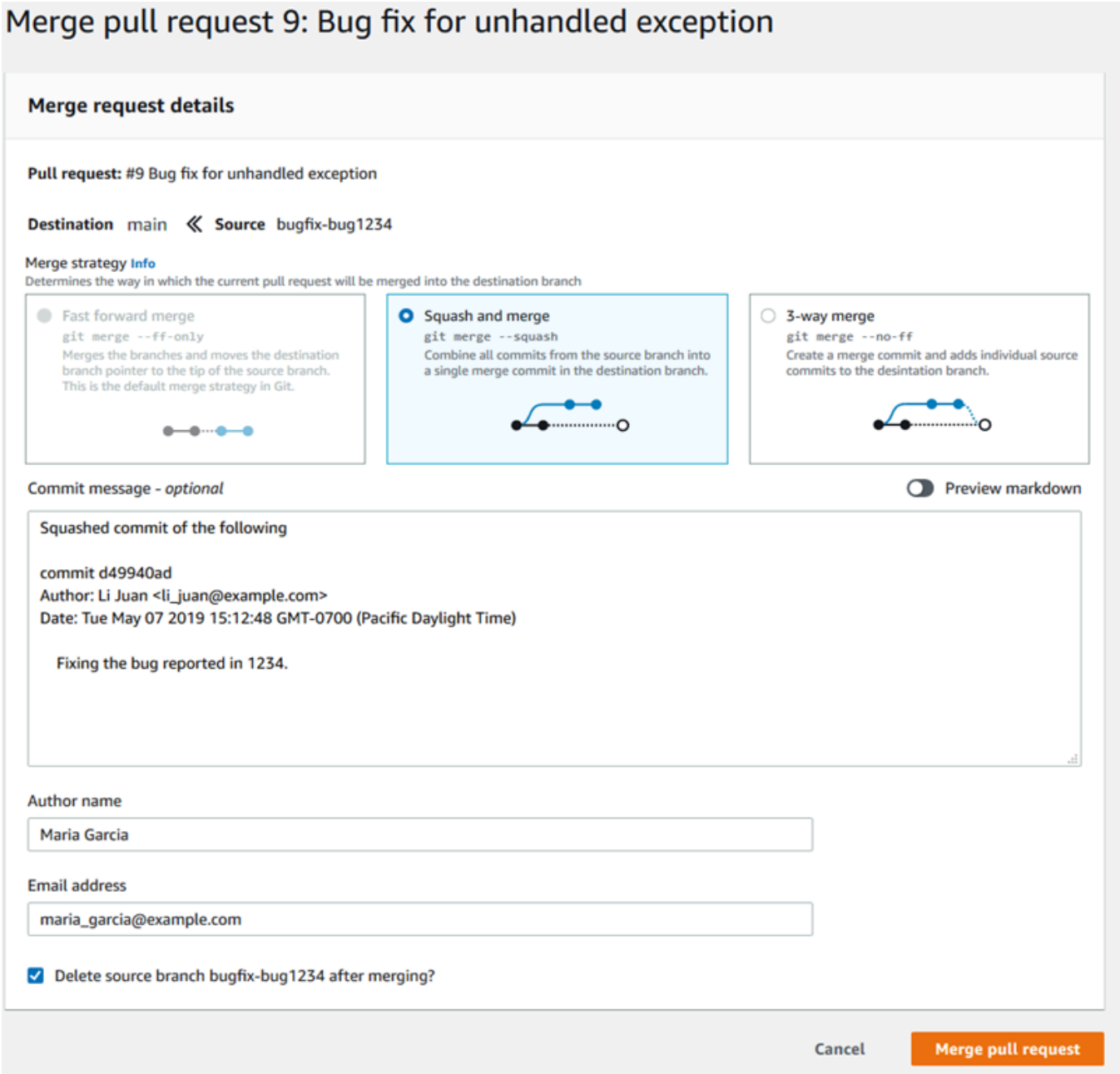
8. 如果您對請求中所做的變更感到滿意，請選擇 Approve (核准)。即使未設定提取請求的核准規則，您也可以選擇核准該提取請求。這會清楚記錄您已檢閱提取請求及核准變更。如果您改變主意，您也可以選擇撤銷核准。



> **Note**
>
> 您無法核准您自己建立的提取請求。

9. 當您滿意所有程式碼變更都經過檢閱並獲得同意時，請從提取請求執行下列其中一項動作：

   - 如果您想關閉提取請求但不合併分支，請選擇 Close pull request (關閉提取請求)。
   - 如果您要合併分支並關閉提取請求，請選擇 Merge (合併)。您可以在適用於您程式碼的合併策略之間選擇 (這取決於來源和目的地分支之間的差異)，以及是否在完成合併之後自動刪除來源分支。做出選擇之後，請選擇 Merge pull request (合併提取請求) 完成合併。



   - 如果分支中存在無法自動解決的合併衝突，則可以在 CodeCommit 控制台中解決它們，也可以使用本地 Git 客戶端合併分支，然後推送合併。如需詳細資訊，請參閱 解決中提取請求的衝突 AWS CodeCommit儲存庫。

> **Note**
>
> 您一律可以在本機儲存庫中使用 git merge 命令並推送變更，以手動合併分支，包括提取請求分支。

如需詳細資訊，請參閱 使用提取請求 及 使用核准規則範本。

步驟 5：清除

如果您不再需要存 CodeCommit 放庫，您應該刪除存 CodeCommit 放庫和您在本練習中使用的其他資源，這樣就不會繼續支付儲存空間的費用。

> **Important**
>
> 這個操作無法復原。刪除此存儲庫後，您無法再將其克隆到任何本地存儲庫或共享存儲庫。您也不能再從任何本地存儲庫或共享存儲庫中提取數據或將數據推送到它，或執行任何 Git 操作。
>
> 如果您為儲存庫設定了通知，刪除儲存庫也會刪除為儲存庫建立的 Amazon E CloudWatch vents 規則。它不會刪除做為該規則目標使用的 Amazon SNS 主題。
>
> 如果您為儲存庫設定了觸發器，刪除存放庫並不會刪除您設定為這些觸發程序目標的 Amazon SNS 主題或 Lambda 函數。如果您已不需要這些資源，務必刪除。如需詳細資訊，請參閱 從儲存庫刪除觸發器。

若要刪除存 CodeCommit 放庫

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要刪除的儲存庫。如果您遵循本主題中的命名慣例，則會將其命名為MyDemoRepo。
3. 在導覽窗格中，選擇設定。
4. 在 Settings (設定) 頁面的 Delete repository (刪除儲存庫) 中，選擇 Delete repository (刪除儲存庫)。
5. 輸入 **`delete`**，然後選擇 Delete (刪除)。就會永久刪除這個儲存庫。

步驟 6：後續步驟

現在您已經熟悉了自己 CodeCommit 及其某些功能，請考慮執行以下操作：

- 如果您是 Git 的新手， CodeCommit 或想要檢閱使用 Git 的範例 CodeCommit，請繼續進行開始使用 Git 和 CodeCommit教學課程。
- 如果您要與 CodeCommit 存放庫中的其他人一起使用，請參閱共用儲存庫。如果您想要與其他 Amazon Web Services 帳戶中的使用者共用儲存庫，請參閱使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取。)
- 如果您要將存放庫移轉至 CodeCommit，請遵循中的步驟遷移到 CodeCommit。
- 如果您想要將儲存庫新增到持續交付管道，請遵循簡易管道逐步解說中的步驟。
- 如果您想進一步了解與之整合的產品和服務 CodeCommit，包括來自社群的範例，請參閱產品和服務整合。

開始使用 Git 和 AWS CodeCommit

如果您是 Git 的新手 CodeCommit，並且，本教程可幫助您學習一些簡單的命令來幫助您開始使用。如果您已熟悉 Git，則可以略過此教學課程而直接進入 開始使用 CodeCommit 。

在本教學課程中，您會建立代表儲存庫的本機副本的 CodeCommit 存放庫，我們稱之為本機存放庫。

創建本地存儲庫後，您對其進行了一些更改。然後，您將更改發送（推送）到 CodeCommit 存儲庫。

您還可以模擬一個團隊環境，其中兩個用戶將更改獨立提交到其本地存儲庫，並將這些更改推送到 CodeCommit 存儲庫。然後，用戶將更改從 CodeCommit 存儲庫中提取到自己的本地存儲庫，以查看其他用戶所做的更改。

您還可以創建分支和標籤，並管理 CodeCommit 存儲庫中的某些訪問權限。

完成此教學課程後，您對核心 Git 和 CodeCommit 概念應該就具備足夠的實務，即可將這些實務運用到您自己的專案中。

完成先決條件和設定，包括：

- 將許可指派給 IAM 使用者。
- 設定 CodeCommit 為使用 HTTPS、SSH 或連線到存放庫git-remote-codecommit。如需這些選擇的詳細資訊，請參閱設定 AWS CodeCommit。

- 如果您想要使用命令列或終端機執行所有操作 (包括建立儲存庫)，請設定 AWS CLI。

主題

步驟 1：建立 CodeCommit 儲存庫

在此步驟中，您可以使用 CodeCommit 主控台來建立存放庫。

如果您已經有要使用的 CodeCommit 儲存庫，則可以略過此步驟。

> **Note**
>
> 根據您的使用情況，您可能需要支付建立或存取存放庫的費用。如需詳細資訊，請參閱 CodeCommit 產品資訊頁面上的定價。

若要建立存 CodeCommit 放庫

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 使用區域選擇器來選擇您AWS 區域要建立儲存庫的位置。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 請在 Repositories (儲存庫) 頁面上，選擇 Create repository (建立儲存庫)。
4. 在 Create repository (建立儲存庫) 頁面的 Repository name (儲存庫名稱) 中，輸入儲存庫的名稱 (例如 **MyDemoRepo**)。

> **Note**
>
> 儲存庫名稱需區分大小寫且不能超過 100 個字元。如需詳細資訊，請參閱限制。

5. (選用) 在 Description (描述) 中，輸入描述 (例如，**`My demonstration repository`**)。這可協助您和其他使用者識別儲存庫的用途。
6. (選用) 選擇 Add tag (新增標籤)，以新增一或多個儲存庫標籤 (自訂屬性標籤，可協助您整理和管理您的 AWS 資源) 到您的儲存庫。如需詳細資訊，請參閱標記儲存庫 AWS CodeCommit。
7. (選擇性) 展開其他組態以指定是使用預設金鑰AWS 受管金鑰還是您自己的客戶管理金鑰來加密和解密此儲存庫中的資料。如果您選擇使用自己的客戶管理金鑰，則必須確定該金鑰可在您建立儲存庫的AWS 區域位置使用，且金鑰處於作用中狀態。如需詳細資訊，請參閱AWS Key Management Service和AWS CodeCommit存儲庫的加密。
8. (選擇性) 如果此儲存庫將包含 Java 或 Python 程式碼，且您想要讓 CodeGuru 審核者分析該程式碼，請選取啟用 Java 和 Python 的 Amazon CodeGuru 審核者。 CodeGuru Reviewer 使用多個機器學習模型來尋找程式碼瑕疵，並在提取要求中自動建議改進和修正。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南。
9. 選擇建立。

> **Note**
>
> 本教學課程中的其餘步驟用`MyDemoRepo`於 CodeCommit 存放庫的名稱。如果您選擇不同名稱，請在此教學課程中都使用此名稱。

如需有關建立儲存庫的詳細資訊，包括如何從終端機或命令列建立儲存庫，請參閱建立 儲存庫。

步驟 2：建立本機存放庫

在此步驟中，您要在本機電腦上設定本機儲存庫，以連接至您的儲存庫。若要這樣做，請在本機電腦上選取可代表本機儲存庫的目錄。您可以使用 Git 克隆和初始化該目錄內空 CodeCommit 存儲庫的副本。然後，您可以指定用於註釋提交的 Git 用戶名和電子郵件地址。

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選取器中，選擇建立儲存庫的AWS 區域位置。儲存庫特定於AWS 區域. 如需詳細資訊，請參閱區域和 Git 連線端點。

3. 尋找您要從清單連接的儲存庫並加以選擇。選擇 Clone URL (複製 URL)，然後選擇複製和連線至儲存庫時要使用的通訊協定。這會將複製 URL 複製。

   - 如果您要搭配 IAM 使用者使用 Git 登入資料，或是AWS CLI.
   - 如果您是在本機電腦上使用 git-remote-codecommit 命令，請複製 HTTPS (GRC) URL。
   - 如果您將 SSH 公開/私密 key pair 與 IAM 使用者搭配使用，請複製 SSH URL。

   > **Note**
   >
   > 如果您看到「歡迎」頁面而非儲存庫清單，表示您登入的AWS 區域地方沒有與您的AWS 帳戶相關聯的儲存庫。要建立儲存庫，請參閱 the section called “建立 儲存庫” 或依照 開始使用 Git 和 CodeCommit 教學課程中的步驟。

4. (選擇性) 我們建議您將本機 Git 用戶端設定**`main`**為使用做為儲存庫預設分支的名稱。這是本指南中所有範例中用於預設分支的名稱。如果您在控制台中進行第一次提交，它也與默認分支名稱 CodeCommit 相同。執行下列命令，為您的系統全域設定預設分支名稱：

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

   如果您希望為所有儲存庫使用不同的預設分支名稱，請以您偏好**`main`**的名稱取代。本教學課程假設您的預設分支名為 main。

   如果您想為不同的存儲庫使用不同的默認分支名稱，則可以在本地設置此屬性（--local）而不是全局（--global）。

5. 在終端機或命令提示字元中，使用git clone指令複製儲存庫，並提供您在步驟 3 中複製的複製 URL。您的複製 URL 取決於您使用的通訊協定和組態。例如，如果您使用 HTTPS 搭配 Git 認證來複製位於美國東部 (俄亥俄) 區域的儲存庫：*MyDemoRepo*

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

   如果您是使用 HTTPS 搭配 git-remote-codecommit：

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

   如果您是使用 SSH：

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

> **Note**
>
> 如果您在嘗試複製儲存庫時出現錯誤，您可能尚未完成本機電腦所需的設定作業。如需詳細資訊，請參閱設定 AWS CodeCommit。

步驟 3：創建您的第一個提交

在此步驟中，您將在本地存儲庫中創建第一次提交。為此，您可以在本地存儲庫中創建兩個示例文件。您可以使用 Git 將更改暫存到，然後將更改提交到本地存儲庫。

1. 使用文字編輯器，在您目錄中建立以下兩個範例文字檔案。將檔案命名為 `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. 運行git config以將佔*your-user-name*位符表示的用戶名和電子郵件地址添加*your-email-address*到您的本地存儲庫中。這可讓您更輕鬆地識別您所做的遞交：

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

3. 如果您在建立本機存放庫時並未全域設定預設分支名稱，請執行下列指令，將預設分支名稱設定為**main**：

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

4. 執行 git add 來暫存變更：

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

5. 執行 git commit 來遞交變更：

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

> **Tip**
>
> 若要查看您剛完成之遞交的詳細資訊，請執行 git log。

步驟 4：推送您的第一次提交

在此步驟中，您將提交從本地存儲庫推送到 CodeCommit 存儲庫中。

運行git push以通過 Git 用於 CodeCommit 存儲庫的默認遠程名稱（`origin`）從本地 repo（`main`）中的默認分支推送提交：

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

> **Tip**
>
> 將檔案推送到 CodeCommit 儲存庫之後，您可以使用主 CodeCommit 控台來檢視內容。如需詳細資訊，請參閱瀏覽存儲庫中的文件。

第 5 步：共享 CodeCommit 存儲庫並推送並提取另一個提交

在此步驟中，您會與其他團隊成員共用有關 CodeCommit 存放庫的資訊。專案團隊成員會使用此資訊取得本機副本、對其進行一些變更，然後將修改的本機副本推送至您的 CodeCommit 存放庫。然後，您將更改從 CodeCommit 存儲庫中提取到本地回購。

在此教學課程中，您模擬該同事使用者，在您於步驟 2 建立的目錄以外，讓 Git 再建立另一個目錄。(通常，此目錄位於不同的機器上)。這個新目錄是 CodeCommit 存放庫的副本。您對現有目錄或這個新目錄所做的任何變更彼此無關。識別這些目錄更改的唯一方法是從 CodeCommit 存儲庫中提取。

即使這些目錄在相同的本機機器上，我們仍分別將現有目錄稱為本機儲存庫，而將新目錄稱為共用儲存庫。

從新目錄中，您會取得 CodeCommit 儲存庫的單獨副本。然後添加一個新的示例文件，將更改提交到共享存儲庫，然後將提交從共享存儲庫推送到您的 CodeCommit 存儲庫。

最後，您將更改從存儲庫拉到本地存儲庫，然後瀏覽它以查看其他用戶提交的更改。

1. 切換到 `/tmp` 目錄或 `c:\temp` 目錄。
2. 執行 git clone，將儲存庫的副本提取到共用儲存庫：

   針對 HTTPS：

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

   針對搭配 git-remote-codecommit 的 HTTPS：

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

   針對 SSH：

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

   > **Note**
   >
   > 當您在 Windows 作業系統上使用 SSH 複製儲存庫時，您可能需要將 SSH 金鑰 ID 新增到連線字串，如下所示：
   >
   > ```
   > git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/
   > repos/MyDemoRepo my-demo-repo
   > ```
   >
   > 如需詳細資訊，請參閱適用於 Windows 上的 SSH 連線。

   在此命令中，`MyDemoRepo`是存 CodeCommit 放庫的名稱。 `shared-demo-repo`是 Git 在目錄或目錄中建立的`/tmp`目錄名稱。`c:\temp`在 Git 建立目錄之後，Git 將儲存庫的副本提取到 `shared-demo-repo` 目錄。

3. 切換到 shared-demo-repo 目錄：

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

4. 執行git config以新增由預留位置和表示的其他使用者名稱*other-user-name*和*other-email-address*電子郵件地址。這可讓您更輕鬆地識別其他使用者所做的遞交：

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

5. 使用文字編輯器，在 shared-demo-repo 目錄中建立以下範例文字檔案。將檔案命名為 horse.txt：

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

6. 執行 git add，將變更暫存到共用儲存庫：

```
git add horse.txt
```

7. 執行 git commit，將變更遞交到共用儲存庫：

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

8. 運行git push以通過 Git 用於存儲庫的默認遠程名稱（ ）從本地 CodeCommit repo（originmain）中的默認分支推送初始提交：

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

9. 切換到您的本地回購並運行git pull以拉入您的本地回購，將共享回購提交到 CodeCommit 存儲庫。然後執行 git log，以查看從共用儲存庫起始的遞交。

步驟 6：創建並共享分支

在此步驟中，您可以在本地存儲庫中創建一個分支，進行一些更改，然後將分支推送到您的 CodeCommit 存儲庫。然後，您將分支從 CodeCommit 存儲庫中拉到共享回購。

分支可讓您單獨開發不同版本的儲存庫內容 (例如，試用新的軟體功能，而不會影響團隊成員的工作)。當該功能穩定之後，便將分支合併到軟體中更穩定的分支。

您會使用 Git 建立分支，然後將分支指向您所做的第一個遞交。您可以使用 Git 將分支推送到 CodeCommit 存儲庫。然後切換到您的共享倉庫，並使用 Git 將新分支拉入共享的本地存儲庫並探索該分支。

1. 從您的本地回購中運行git checkout，指定分支的名稱（例如，MyNewBranch）和您在本地回購中進行的第一次提交的 ID。

   如果您不知道遞交 ID，請執行 git log 以取得 ID。請確定遞交具有您的使用者名稱和電子郵件地址，而不是其他使用者的使用者名稱和電子郵件地址。這是為了模擬這main是一個穩定版本的 CodeCommit 存儲庫，並且該MyNewBranch分支適用於一些新的，相對不穩定的功能：

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

2. 運行git push以將新分支從本地回購發送到 CodeCommit 存儲庫：

   ```
   git push origin MyNewBranch
   ```

3. 現在，將分支提取到共用儲存庫並檢查結果：

   1. 切換到共用儲存庫目錄 (shared-demo-repo)。
   2. 提取新的分支 (git fetch origin)。
   3. 確認已提取分支 (git branch --all 會顯示儲存庫所有分支的清單)。
   4. 切換到新的分支 (git checkout MyNewBranch)。
   5. 確認您已執行 git status 或 git branch 以切換到 MyNewBranch 分支。輸出顯示您所在的分支。在此案例中應該是 MyNewBranch。
   6. 檢視分支中的遞交清單 (git log)。

   以下是需要呼叫的 Git 命令清單：

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

4. 切換回 main 分支並檢視其遞交清單。Git 命令應如下所示：

```
git checkout main
git log
```

5. 切換到本地存儲庫中的main分支。您可以執行 git status 或 git branch。輸出顯示您所在的分支。在此案例中應該是 main。Git 命令應如下所示：

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

步驟 7：建立並分享標籤

在此步驟中，您可以在本地回購中創建兩個標籤，將標籤與提交相關聯，然後將標籤推送到存 CodeCommit 儲庫中。然後，您將更改從 CodeCommit 存儲庫中提取到共享回購。

標籤用於讓遞交 (或分支，甚至是另一個標籤) 有一個人類可讀的名稱。例如，如果想要將遞交加上標籤 v2.1，就可以這樣做。遞交、分支或標籤可以擁有任何數量相關聯的標籤數目，但個別標籤只能與一個遞交、分支或標籤相關聯。在此教學課程中，您會將一個遞交加上 release 標籤，而將另一個遞交加上 beta 標籤。

您會使用 Git 建立標籤，並將 release 標籤指向您所做的第一個遞交，而將 beta 標籤指向另一個使用者所做的遞交。然後，您可以使用 Git 將標籤推送到 CodeCommit 儲存庫。然後你切換到你的共享倉庫，並使用 Git 將標籤拉到你的共享本地倉庫中，並探索這些標籤。

1. 從您的本地回購中運行git tag，指定 new tag (release) 的名稱和您在本地回購中進行的第一次提交的 ID。

   如果您不知道遞交 ID，請執行 git log 以取得 ID。請確定遞交具有您的使用者名稱和電子郵件地址，而不是其他使用者的使用者名稱和電子郵件地址。這是為了模擬您的提交是 CodeCommit 存儲庫的穩定版本：

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

   執行 git tag，將另一個使用者所做的遞交加上 beta 標籤。這是為了模擬遞交是一些較不穩定的新功能：

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

2. 執行git push --tags以將標籤傳送至 CodeCommit 儲存庫。
3. 現在，將標籤提取到共用儲存庫並檢查結果：

   1. 切換到共用儲存庫目錄 (shared-demo-repo)。
   2. 提取新的標籤 (git fetch origin)。
   3. 確認已提取標籤 (git tag 會顯示儲存庫的標籤清單)。
   4. 檢視每個標籤 (git log release 和 git log beta) 的相關資訊。

   以下是需要呼叫的 Git 命令清單：

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

4. 也可以在本地回購中嘗試這個：

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

步驟 8：設定存取權限

在此步驟中，您授予使用者同步處理共用存放庫與 CodeCommit 存放庫的權限。此為選用步驟。建議對於有興趣了解如何在使用者使用 Git 認證或 SSH 金鑰配對與 IAM 使用者存取 CodeCommit 儲存庫時控制 CodeCommit 存放庫存取權的使用者。

> **Note**
>
> 如果您使用聯合存取、臨時登入資料或網路身分供應商 (例如 IAM Identity Center)，請為您的身分提供者設定使用者、存取權和許可，然後使用git-remote-codecommit。如需詳細資訊，請參閱 HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit 及 使用旋轉認證連線至AWS CodeCommit儲存庫。

若要這麼做，您可以使用 IAM 主控台建立使用者，使用者預設沒有將共用存放庫與 CodeCommit 存放庫同步處理的權限。您可以執行 git pull 來驗證是否如此。如果新使用者沒有同步的許可，則命令不會有作用。然後返回 IAM 主控台並套用允許使用者使用的政策git pull。同樣地，您可以執行 git pull 來驗證是否如此。

此步驟的撰寫方式是假設您有權在 Amazon Web Services 帳戶中建立 IAM 使用者的權限。如果您沒有這些許可，則無法執行此步驟中的程序。直接跳到步驟 9：清除，以清除您用於此教學課程的資源。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

   務必使用您在 設定 中使用的相同使用者名稱和密碼。

2. 在導覽窗格中，選擇 Users (使用者)，然後選擇 Create New Users (建立新使用者)。
3. 在第一個 Enter User Names (輸入使用者名稱) 方塊中，輸入範例使用者名稱 (例如，**JaneDoe-CodeCommit**)。選取 Generate an access key for each user (為每位使用者產生存取金鑰) 方塊，然後選擇 Create (建立)。
4. 選擇 Show User Security Credentials (顯示使用者安全登入資料)。記下存取金鑰 ID 和私密存取金鑰，或選擇 Download Credentials (下載登入資料)。
5. 按照中適用於使用 Git 認證的 HTTPS 使用者的說明產生並提供 IAM 使用者的登入資料。

   如果您想要使用 SSH，請按照 安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰 CodeCommit或 步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰中的指示，以公有和私有金鑰設定使用者。

6. 執行 git pull。應該會出現以下錯誤：

   針對 HTTPS：

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

   針對 SSH：

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

   出現錯誤是因為新用戶沒有將共享存儲庫與 CodeCommit 存儲庫同步的權限。

7. 返回 IAM 主控台。在導覽窗格中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。(出現 Get Started (開始使用) 按鈕時先選擇它，然後選擇 Create Policy (建立政策)。)
8. 在建立您自己的政策旁邊，選擇選取。
9. 在 Policy Name (政策名稱) 方塊中，輸入名稱 (例如，**CodeCommitAccess-GettingStarted**)。
10. 在「政策文件」方塊中輸入以下內容，讓 IAM 使用者從與 IAM 使用者關聯的任何儲存庫中提取：

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

> **Tip**
>
> 如果您希望 IAM 使用者能夠將提交推送到與 IAM 使用者相關聯的任何儲存庫，請改為輸入：
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Action": [
>         "codecommit:GitPull",
>         "codecommit:GitPush"
>       ],
>       "Resource": "*"
>     }
>   ]
> }
> ```

如需可授與使用者的其他 CodeCommit 動作和資源權限的相關資訊，請參閱AWS CodeCommit 的身分驗證與存取控制。

11. 在導覽窗格中，選擇使用者 。
12. 選擇您要連接政策的範例使用者名稱 (例如，**`JaneDoe-CodeCommit`**)。
13. 選擇 Permissions (許可) 索引標籤標籤。
14. 在 Managed Policies (受管政策) 中，選擇 Attach Policy (連接政策)。
15. 選取您剛建立的 **`CodeCommitAccess-GettingStarted`** 政策，然後選擇 Attach Policy (連接政策)。
16. 執行 git pull。這一次，命令應該有作用，也應該會出現 `Already up-to-date` 訊息。
17. 如果您是使用 HTTPS，請切換到您的原始 Git 登入資料，或者如果您是使用 git-remote-codecommit，則請使用您平常使用的描述檔。如需詳細資訊，請參閱使用 Git 認證的 HTTPS 使用者進行設定或HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit中的指示。

    如果是使用 SSH，請切換到您的原始金鑰。如需詳細資訊，請參閱 安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰CodeCommit 或 步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰 。

您已來到此教學的最後部分。

步驟 9：清除

在此步驟中，您會刪除您在本教學課程中使用的存 CodeCommit 放庫，這樣就不會繼續支付儲存空間的費用。

您還可以刪除本地計算機上的本地存儲庫和共享回購，因為刪除 CodeCommit 存儲庫後不需要它們。

> **Important**
>
> 刪除此存儲庫後，您將無法將其克隆到任何本地回購或共享存儲庫。您也將無法從任何本地回購或共享回購中提取數據或將數據推送到它。這個操作無法復原。

若要刪除存 CodeCommit 放庫 (主控台)

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home

2. 在 Dashboard (儀表板) 頁面的儲存庫清單中，選擇 MyDemoRepo。
3. 在導覽窗格中，選擇設定。
4. 在 Settings (設定) 頁面的 Delete repository (刪除儲存庫) 中，選擇 Delete repository (刪除儲存庫)。
5. 在 Type the name of the repository to confirm deletion (輸入儲存庫名稱以確認刪除) 旁的方塊中，輸入 **MyDemoRepo**，然後選擇 Delete (刪除)。

若要刪除 CodeCommit 存放庫 (AWS CLI)

運行刪除存儲庫命令：

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

刪除本地回購和共享回購

若為 Linux、macOS 或 Unix：

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

針對 Windows：

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

產品與服務整合 AWS CodeCommit

默認情況下 CodeCommit ，與許多 AWS 服務集成。您也可以在以外 CodeCommit 的產品和服務一起使用 AWS。下列資訊可協助您設定 CodeCommit為與您使用的產品和服務整合。

> **Note**
>
> 您可以通過集成自動構建和部署提交到 CodeCommit 存儲庫 CodePipeline。若要進一步了解，請遵循《入 DevOps 門指南》中的AWS 步驟。

主題

與其他 AWS 服務整合

CodeCommit 與以下 AWS 服務集成：

AWS Amplify	AWS Amplify可讓您輕鬆建立、設定和實作搭載的可擴充行動應用程式 AWS。Amplify 可無縫佈建和管理您的行動後端，並提供簡單的架構來將您的後端與 iOS、Android、Web 和 React Native 前端輕鬆整合。Amplify 也可自動化前端與後端應用程式發行程序，讓您更快速地交付功能。 您可以在 Amplify 控制台中連接 CodeCommit 存儲庫。在您授權 Amplify 主控台之後，Amplify 會從存放庫提供者擷取存取權杖，但不會將權杖儲存在伺服器上。 AWS Amplify 只會使用安裝在特定儲存庫中的部署金鑰來存取您的儲存庫。 進一步了解： • AWS Amplify 使用者指南

	• 入門
AWS Cloud9	AWS Cloud9 包含用於在雲端編寫程式碼、建置、執行、測試、偵錯以及發行軟體的工具集合。這個工具集合稱為 AWS Cloud9 整合式開發環境或 IDE。 您可以透過網頁瀏覽器存取 AWS Cloud9 IDE。IDE 提供豐富的程式碼編輯體驗，可支援多種程式設計語言和執行時間除錯器，以及內建終端機。 進一步了解： • AWS Cloud9 使用者指南 • AWS CodeCommit 樣品 AWS Cloud9 • 將 AWS Cloud9 與 AWS CodeCommit 整合
AWS CloudFormation	AWS CloudFormation這項服務可協助您建立資源模型並設定 AWS 資源，以減少管理這些資源的時間，將更多時間用於應用程式。您可以建立描述資源 (包括 CodeCommit 存放庫) 的範本，AWS CloudFormation 並為您處理佈建和設定這些資源。 進一步了解： • AWS CodeCommit 儲存庫資源頁面

AWS CloudTrail	CloudTrail擷取 Amazon Web 服務帳戶或代表 Amazon Web Services 帳戶發出的 AWS API 呼叫和相關事件，並將日誌檔交付到您指定的 Amazon S3 儲存貯體。您可以設定 CloudTrail 為從 AWS CodeCommit 主控台擷取 API 呼叫、來自本機 Git 用戶端的 CodeCommit 命令，以及從 API 擷取 CodeCommit API。 AWS CLI 進一步了解： • 使用 AWS CloudTrail 記錄 AWS CodeCommit API 呼叫
Amazon CloudWatch 活動	CloudWatch E@@ vents 提供近乎即時的系統事件串流，用來描述 AWS 資源變更。使用可快速設置的簡單規則，您可以匹配事件並將其路由到一個或多個目標函數或流。 CloudWatch 事件在發生時會意識到操作變化。 CloudWatch Events 會回應這些作業變更，並視需要採取行動，方法是傳送訊息以回應環境、啟動功能、進行變更，以及擷取狀態資訊。 您可以設定 CloudWatch 事件以監控 CodeCommit 儲存庫並回應儲存庫事件，方法是將串流、函數、任務或其他 AWS 服務 (例如 Amazon 簡單佇列服務、Amazon Kinesis 等) 中的其他程序鎖定目標。 AWS Lambda 進一步了解： • CloudWatch 活動用戶指南 • AWS CodeCommit 事件 • 部落格文章：使用 Amazon CloudWatch 事件和 JGit 建置無伺服器 AWS CodeCommit 工作流程

AWS CodeBuild	CodeBuild是雲端中完全受控的建置服務，可編譯原始程式碼、執行單元測試，以及產生準備好部署的成品。您可以將要構建的源代碼和構建規範存儲在 CodeCommit存儲庫中。您可以 CodeBuild 直接搭配使用 CodeCommit，也可以將 CodeBuild和合併到持續交付管道 CodeCommit 中 CodePipeline。 進一步了解： • 規劃組建 • 建立組建專案 • CodePipeline 搭配使用 AWS CodeBuild 來執行組建
Amazon 評論 CodeGuru 家	Amazon CodeGuru Reviewer 是一種自動化程式碼檢閱服務，它使用程式分析和機器學習來偵測常見問題，並建議 Java 或 Python 程式碼中的修正程式。您可以將 Amazon Web Services 帳戶中的儲存庫與 CodeGuru 審核者建立關聯。當您這麼做時， CodeGuru Reviewer 會建立服務連結角色，讓 CodeGuru Reviewer 分析建立關聯後所建立之所有提取要求中的程式碼。 進一步了解： • 將 AWS CodeCommit 儲存庫與 Amazon CodeGuru 審核者建立關聯或取消關聯 • Amazon 評論 CodeGuru 者用戶指南

AWS CodePipeline	CodePipeline是一種持續交付服務，可用於建模、視覺化和自動化發行軟體所需的步驟。您可以設 CodePipeline 定使用 CodeCommit 存放庫做為管線中的來源動作，並自動建置、測試和部署變更。 進一步了解： • 與 CodePipeline 和的簡單管道逐步解說 AWS CodeCommit • 使用 CodeCommit儲存庫遷移到 Amazon 管道 CloudWatch 事件變更偵測 • 用來自動啟動管道的變更偵測方法
AWS CodeStar	AWS CodeStar是一種基於雲的服務，用於在上創建，管理和處理軟件開發項目 AWS。您可以 AWS 透過 AWS CodeStar 專案快速開發、建置和部署應用程式。 AWS CodeStar 專案會為您的專案開發工具鏈建立並整合 AWS 服務，包括專案的 CodeCommit 存放庫。 AWS CodeStar 也會將權限指派給該專案的專案團隊成員。這些權限會自動套用，包括存取 CodeCommit、建立和管理 Git 認證的權限等。 您可以使用 AWS CodeCommit 主控台、本機 Git 用戶端和 CodeCommit API 中的 CodeCommit 指令，來設定為 AWS CodeStar 專案建立的儲 CodeCommit 存庫 AWS CLI，就像對其他任何儲存庫一樣。 進一步了解： • 使用儲存庫 • 使用 AWS CodeStar 專案 • 使用 AWS CodeStar 團隊

AWS Elastic Beanstalk	Elastic Beanstalk 是一種託管服務，可讓您輕鬆部署和管理 AWS 雲端中的應用程式，而不必擔心執行這些應用程式的基礎架構。您可以使用 Elastic Beanstalk 命令列介面 (EB CLI) 直接從新的或現有的儲存庫部署應用程式。CodeCommit 進一步了解： • 使用 EB CLI 搭配 AWS CodeCommit • 使用現有的存 AWS CodeCommit 放庫 • eb codesource (EB CLI 命令)
AWS Key Management Service	AWS KMS 是一種受管服務，可讓您輕鬆地建立和控制用來加密資料的加密金鑰。依預設，CodeCommit 會用 AWS KMS 來加密儲存庫。 進一步了解： • AWS KMS和加密
AWS Lambda	Lambda 可讓您執行程式碼，無須佈建或管理伺服器。您可以為呼叫 Lambda 函數以回應 CodeCommit 儲存庫事件的儲存庫設定觸發器。 進一步了解： • 為 Lambda 函數建立觸發器 • AWS Lambda 開發人員指南

Amazon Simple Notification Service	Amazon SNS 是一種 Web 服務，可讓應用程式、最終使用者和裝置立即從雲端傳送和接收通知。您可以為存放 CodeCommit 庫設定觸發器，這些儲存庫會傳送 Amazon SNS 通知以回應儲存庫事件。您也可以使用 Amazon SNS 通知與其他 AWS 服務整合。例如，您可以使用 Amazon SNS 通知將訊息傳送到 Amazon 簡單佇列服務佇列。 進一步了解： • 為 Amazon SNS 主題創建觸發器 • 《Amazon Simple Notification Service 開發人員指南》https://docs.aws.amazon.com/sns/latest/dg/welcome.html

來自社群的整合範例

下列各節提供部落格文章、文章和社群所提供範例的連結。

> **Note**
>
> 這些連結僅供參考用途，不應視為完整清單或對範例內容的背書。 AWS 對外部內容的內容或準確性概不負責。

主題

- 部落格文章
- 程式碼範例

部落格文章

- 整合 SonarQube 為提取請求核准者 AWS CodeCommit

  瞭解如何建立需要成功進行 SonarQube 品質分析的 CodeCommit 儲存庫，然後才能合併提取要求。

發佈日期：2019 年 12 月 12 日

- 移轉至 AWS CodeCommitAWS CodePipeline、以及 AWS CodeBuild 從 GitLab

  了解如何使 AWS CodePipeline 用 GitLab和將多個儲存庫移轉 AWS CodeCommit 至 CI/CD 管道，並設定。 AWS CodeBuild

  發佈日期：2019 年 11 月 22 日

- 實作 GitFlow 使用 AWS CodePipelineAWS CodeCommit、 AWS CodeBuild、和 AWS CodeDeploy

  了解如何實作 GitFlow 使用 AWS CodePipeline AWS CodeCommit、 AWS CodeBuild、和 AWS CodeDeploy。

  發佈日期：2019 年 2 月 22 日

- AWS CodeCommit 跨多個 AWS 帳戶使用 Git

  了解如何跨多個 Amazon Web Services 帳戶管理 Git 組態。

  發佈日期：2019 年 2 月 12 日

- 使 AWS CodeBuild用和驗證 AWS CodeCommit 提取請求 AWS Lambda

  了解如何使用 AWS CodeCommit、 AWS CodeBuild和驗證提取請求 AWS Lambda。在將建議的變更合併到預設分支之前，針對建議的變更執行測試，您可以協助確保提取要求的高品質、catch 測任何潛在問題，並提升開發人員對其變更的信心。

  發佈日期：2019 年 2 月 11 日

- 使用同盟身分 AWS CodeCommit

  了解如何使用企業中 AWS CodeCommit 使用的身分來存取儲存庫。

  發佈日期：2018 年 10 月 5 日

- 細化對於中分支的存取 AWS CodeCommit

  了解如何透過建立和套用使用內容金鑰的 IAM 政策來限制存放庫分支的提交。

  發佈日期：2018 年 5 月 16 日

- 使用 AWS Far AWS CodeCommit gate 在區域之間複製儲存庫

  了解如何使用無伺服器架構，設定 CodeCommit 儲存庫從一個 AWS 區域連續複寫到另一個區域。

發佈日期：2018 年 4 月 11 日

- 分發您的 AWS OpsWorks for Chef Automate 基礎結

  了解如何使用 CodePipeline CodeCommit CodeBuild、和確保食譜和 AWS Lambda 其他配置一致地部署在一個或多個位於一個或多個 Chef Server 的兩個或多個 Chef Server 上。 AWS 區域

  發佈日期：2018 年 3 月 9 日

- Peanut Butter 和 Chocolate：使用 AWS CodeCommit的 Azure 函數 CI/CD 管道

  了解如何建立 PowerShell以 Azure 函式為基礎的 CI/CD 管線，其中程式碼會儲存在存放庫中 CodeCommit 。

  發佈日期：2018 年 2 月 19 日

- 使用 AWS CodePipeline、 AWS CodeCommit、Amazon ECR 和持續部署至庫伯內特 AWS CodeBuildAWS Lambda

  了解如何使用 Kubernetes 以及 AWS 一起為容器型應用程式建立完全受控的持續部署管道。

  發佈日期：2018 年 1 月 11 日

- 使用提 AWS CodeCommit 取要求來要求程式碼檢閱並討論程式碼

  了解如何使用提取要求來檢閱、加上註解，並以互動方式重複 CodeCommit 存放庫中的程式碼變更。

  發佈日期：2017 年 11 月 20 日

- 使用 Amazon CloudWatch 事件和 JGit 建置無伺服器 AWS CodeCommit 工作流程

  瞭解如何使用存放庫 CloudWatch 事件和其他 AWS 服務中的目標動作來建立事件規則，以處理 CodeCommit存放庫中的變更。範例包括在提交上強制執行 Git 提交訊息政策、複寫 CodeCommit儲存庫以及將 CodeCommit 存放庫備份到 Amazon S3 的 AWS Lambda 功能。

  發佈日期：2017 年 8 月 3 日

- 移轉至 AWS CodeCommit

  了解如何將代碼推送到兩個存儲庫，作為從使用另一個 Git 存儲庫遷移到使用 CodeCommit 時的一部分 SourceTree。

  發佈日期：2016 年 9 月 6 日

- 建立連續測試與 Appium，詹金斯 AWS CodeCommit，和 AWS Device Farm

  了解如何使用 Appium， CodeCommitJenkins 和設備農場為移動設備創建持續測試過程。

  發佈日期：2016 年 2 月 2 日

- 在多個 Amazon Web Services 帳戶中使 AWS CodeCommit 用 Git 存儲庫

  了解如何複製存 CodeCommit 放庫，並在一個命令中設定認證協助程式，使其使用特定 IAM 角色連線至該儲存庫。

  發佈日期：2015 年 11 月

- 整合 AWS OpsWorks 與 AWS CodeCommit

  瞭解如 AWS OpsWorks 何從 CodeCommit中自動擷取應用程式和 Chef 食譜。

  發佈日期：2015 年 8 月 25 日

- 使用 AWS CodeCommit 和 GitHub 憑證助手

  了解如何配置 gitconfig 文件以與 GitHub憑證助手一起 CodeCommit 使用。

  發佈日期：2015 年 9 月

- AWS CodeCommit 從日蝕中使用

  了解如何使用 Eclipse 中的 eGit 工具來使用 CodeCommit。

  發佈日期：2015 年 8 月

- AWS CodeCommit 與 Amazon EC2 角色登入資料

  了解如何在設定存放庫的自動代理程式 CodeCommit 存取權限時，使用 Amazon EC2 的執行個體設定檔。

  發佈日期：2015 年 7 月

- AWS CodeCommit 與詹金斯集成

  瞭解如何使用 CodeCommit 和 Jenkins 來支援兩個簡單的持續整合 (CI) 案例。

  發佈日期：2015 年 7 月

- AWS CodeCommit 與檢閱委員會整合

  瞭解如何使用審核委員會程式碼檢閱系統整合 CodeCommit 至開發工作流程。

發佈日期：2015 年 7 月

程式碼範例

以下是 CodeCommit 使用者可能感興趣的程式碼範例。

- 可定期刪除 OS X 憑證存放區中快取登入資料的 Mac OS X 的指令碼

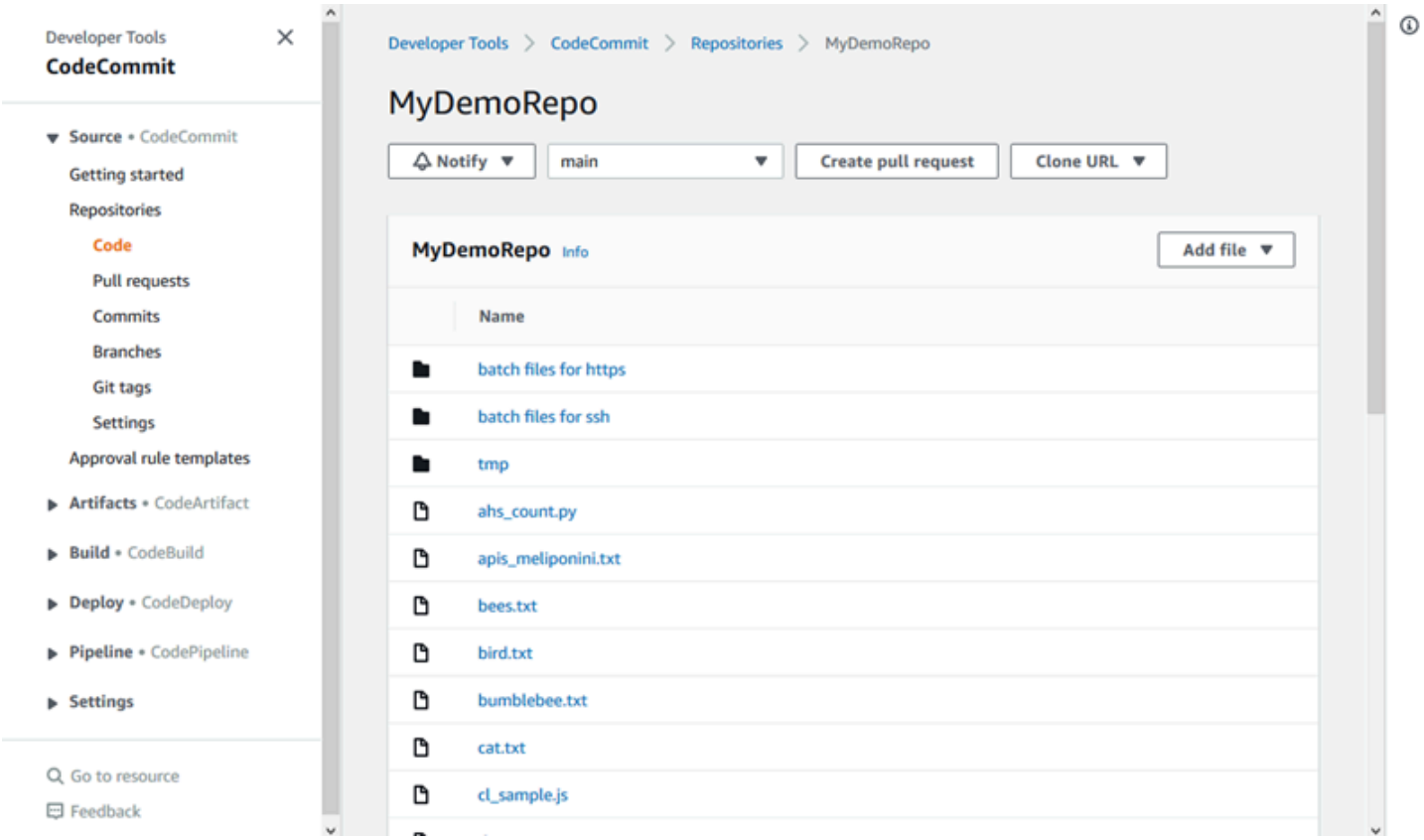
  如果您 CodeCommit 在 Mac OS X 上使用的認證協助程式，您可能已熟悉快取認證的問題。此指令碼示範一個解決方案。

  作者：Nico Coetzee

  發布日期：2016 年 2 月

使用儲存庫 AWS CodeCommit

存放庫是中的基本版本控制物件 CodeCommit。專案的程式碼和檔案安全地存放於此處。其中還存放您的專案歷史記錄，包括從第一個遞交到最新的變更。您可以將儲存庫共用給其他使用者，讓你們共同處理專案。如果您將 AWS 標籤新增至儲存庫，您可以設定通知，讓儲存庫使用者接收有關事件的電子郵件 (例如，其他使用者對程式碼發表評論)。您也可以變更儲存庫的預設設定、瀏覽其內容等等。您可以為儲存庫建立觸發，讓程式碼推送或其他事件觸發動作，例如電子郵件或程式碼函數。您甚至可以在本機電腦上設定儲存庫 (本機儲存庫)，以將您的變更推送到多個儲存庫。



您必須先在 Amazon Web Services 帳戶中設定 IAM 使用者，或設定聯合 CodeCommit 存取權或臨時登入資料的存取權，才能將變更推送到儲存庫。如需詳細資訊，請參閱 步驟 1：初始配置 CodeCommit 及 HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

如需有關在中使用儲存庫其他方面的資訊 CodeCommit 使用檔案使用提取請求，請參閱使用提交使用分支、、和使用者偏好設定。若要取得有關移轉至的資訊 CodeCommit，請參閱遷移到 CodeCommit。

主題

- 建立 AWS CodeCommit 儲存庫

- Connect 到 AWS CodeCommit 儲存庫
- 共用儲 AWS CodeCommit 存庫
- 設定 AWS CodeCommit 存放庫中事件的通知
- 標記儲存庫 AWS CodeCommit
- 管理 AWS CodeCommit 儲存庫的觸發器
- 將 AWS CodeCommit 儲存庫與 Amazon CodeGuru 審核者建立關聯或取消關聯
- 檢視 CodeCommit 儲存庫詳細
- 變更 AWS CodeCommit 儲存庫設定
- 同步本地存儲庫和 AWS CodeCommit 存儲庫之間的更改
- 將提交推送到額外的 Git 存儲庫
- 使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取
- 刪除 AWS CodeCommit 儲存庫

建立 AWS CodeCommit 儲存庫

使用 AWS CodeCommit 控制台或 AWS Command Line Interface （AWS CLI）來創建一個空的 CodeCommit 存儲庫。若要在建立之後將標籤新增到存放庫，請參閱新增標籤至儲存庫。

這些說明假設您已完成設定 中的步驟。

> **Note**
>
> 根據您的使用情況，您可能需要支付建立或存取存放庫的費用。如需詳細資訊，請參閱 CodeCommit 產品資訊頁面上的定價。

主題

- 創建一個存儲庫（控制台）
- 建立儲存庫 (AWS CLI)

創建一個存儲庫（控制台）

若要建立存 CodeCommit 放庫

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在區域選取器中，選擇您 AWS 區域 要建立儲存庫的位置。如需詳細資訊，請參閱 區域和 Git 連線端點。
3. 請在 Repositories (儲存庫) 頁面上，選擇 Create repository (建立儲存庫)。
4. 在 Create repository (建立儲存庫) 頁面的 Repository name (儲存庫名稱) 中，輸入儲存庫的名稱。

   > **Note**
   >
   > 儲存庫名稱需區分大小寫。該名稱在您的 Amazon Web Services 帳戶中必須是唯一的。AWS 區域

5. (選用) 在 Description (描述) 中，輸入儲存庫的描述。這可協助您和其他使用者識別儲存庫的用途。

   > **Note**
   >
   > 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的應用程式開發人員，且計劃在網頁瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

6. (選擇性) 選擇「新增標籤」，將一或多個儲存庫標籤 (可協助您組織及管理 AWS 資源的自訂屬性標籤) 新增至儲存庫。如需詳細資訊，請參閱 標記儲存庫 AWS CodeCommit。
7. (選擇性) 展開其他組態以指定是使用預設金鑰 AWS 受管金鑰 還是您自己的客戶管理金鑰來加密和解密此儲存庫中的資料。如果您選擇使用自己的客戶管理金鑰，則必須確定該金鑰可在您建立儲存庫的 AWS 區域 位置使用，且金鑰處於作用中狀態。如需詳細資訊，請參閱 AWS Key Management Service和AWS CodeCommit存儲庫的加密。
8. (選擇性) 如果此儲存庫包含 Java 或 Python 程式碼，且您希望 CodeGuru 審核者對其進行分析，請選取啟用 Java 和 Python 的 Amazon CodeGuru 審核者。 CodeGuru Reviewer 使用多種機器學習模型來尋找程式碼瑕疵，並在提取要求中提供改善和修正建議。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南。
9. 選擇建立。

建立儲存庫之後，您可以透過 CodeCommit 主控台或本機 Git 用戶端連線至該儲存庫並開始新增程式碼，或是將 CodeCommit 儲存庫與您最愛的 IDE 整合。如需詳細資訊，請參閱 設定 AWS CodeCommit。您也可以將您的儲存庫新增到持續交付管道。如需詳細資訊，請參閱簡易管道演練。

若要取得有關新 CodeCommit 儲存區域的資訊 (例如複製儲存區域時要使用的 URL)，請從清單中選擇儲存區域的名稱，或直接在儲存區域名稱旁選擇要使用的連線協定。

若要與他人共用這個儲存庫，您必須將 HTTPS 或 SSH 連結傳送給他們，以用來複製儲存庫。確定他們擁有存取儲存庫所需的許可。如需詳細資訊，請參閱 共用儲存庫 及 AWS CodeCommit 的身分驗證與存取控制。

建立儲存庫 (AWS CLI)

您可以使用 AWS CLI 建立 CodeCommit 存放庫。與主控台不同的是，如果您使用 AWS CLI來建立，您可以將標籤新增到儲存庫。

1. 請確定您已將存放庫所在 AWS CLI 的 AWS 區域 位置設定為。若要驗證區域，請在命令列或終端機執行下列命令，並檢閱預設區域名稱的資訊：

   ```
   aws configure
   ```

   預設區域名稱必須與中 AWS 區域 的儲存庫相符 CodeCommit。如需詳細資訊，請參閱 區域和 Git 連線端點。
2. 執行 create-repository 命令，並指定：

   - 唯一識別 CodeCommit 存放庫的名稱 (使用`--repository-name`選項)。

     > **Note**
     >
     > 此名稱在 Amazon Web Services 帳戶中必須是唯一的。

   - 關於 CodeCommit 存儲庫的可選註釋（帶有選`--repository-description`項）。
   - 一或多個選擇性的索引鍵值配對，用作 CodeCommit 儲存庫的標籤 (使用選`--tags`項)。
   - 加密和解密此儲存庫時要使用的選用客戶管理金鑰。所有存儲庫都在傳輸過程中和靜態使用密鑰進行加密 AWS KMS。如果未指定任何金鑰，則會使用預設的 AWS Managed 金鑰`aws/codecommit`。

例如，若要建立一個使用說明命名MyDemoRepo的 CodeCommit 儲存庫，"My demonstration repository"並使用名為 *Team* 的索引鍵 (具有 *Sanvi*) 的標籤，請使用此指令。

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

> **Note**
>
> 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用GetRepository或 BatchGetRepositories API 的應用程式開發人員，且計劃在網頁瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

3. 如果成功，此命令會輸出 repositoryMetadata 物件，以及下列資訊：

   - 描述 (repositoryDescription)。
   - 唯一、系統產生的 ID (repositoryId)。
   - 名稱 (repositoryName)。
   - 與 CodeCommit 儲存庫相關聯之 Amazon Web Services 帳戶的識別碼 (accountId)。

   以下是基於上述範例命令的範例輸出。

```
{
    "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**
>
> 在建立儲存庫時新增的標籤，不會在輸出中傳回。若要檢視與儲存庫關聯的標籤清單，請執行 list-tags-for-resource 命令。

4. 記下 CodeCommit 存放庫的名稱和 ID。您需要它們來監視和更改有關 CodeCommit 存儲庫的信息，尤其是在使用時 AWS CLI。

   如果您忘記名稱或 ID，請遵循檢視 CodeCommit 儲存庫詳細資訊 (AWS CLI)中的指示。

建立儲存庫之後，您可以連接到該儲存庫，並開始新增程式碼。如需詳細資訊，請參閱 連接到儲存庫。您也可以將您的儲存庫新增到持續交付管道。如需詳細資訊，請參閱簡易管道演練。

Connect 到 AWS CodeCommit 儲存庫

當您第一次連線到 CodeCommit 存放庫時，通常會將其內容複製到本機電腦。您也可以直接從 CodeCommit 主控台將檔案新增至儲存庫中並編輯檔案。或者，如果您已經擁有本地存儲庫，則可以將 CodeCommit存儲庫添加為遠程庫。本主題提供連線至 CodeCommit 存放庫的指示。如果您要將現有存放庫移轉至 CodeCommit，請參閱遷移到 CodeCommit。

> **Note**
>
> 根據您的使用情況，您可能需要支付建立或存取存放庫的費用。如需詳細資訊，請參閱 CodeCommit 產品資訊頁面上的定價。

主題

- 連線至 CodeCommit儲存庫的先決條件
- 透過複製存 CodeCommit 放庫連 Connect 至儲存庫
- 將本地回購 Connect 到存 CodeCommit 儲庫

連線至 CodeCommit儲存庫的先決條件

在您可以克隆 CodeCommit 存儲庫或將本地回購連接到 CodeCommit 存儲庫之前：

- 您必須使用連線所需的軟體和設定來設定本機電腦 CodeCommit。這包括安裝和設定 Git。如需詳細資訊，請參閱 設定 及 開始使用 Git 和 AWS CodeCommit。
- 您必須具有要連線之 CodeCommit 儲存庫的複製 URL。如需詳細資訊，請參閱 檢視儲存庫詳細。

  如果您尚未建立 CodeCommit 儲存區域，請遵循中的指示建立 儲存庫，複製 CodeCommit 儲存區域的複製 URL，然後返回此頁面。

  如果您有 CodeCommit 儲存庫，但不知道其名稱，請遵循中的指示檢視儲存庫詳細。
- 您必須在本機電腦上有一個位置，才能 CodeCommit 儲存您連線到的存放庫的本機副本。（這個 CodeCommit 存儲庫的本地副本稱為本地回購。）然後從該位置切換到和執行 Git 命令。例如，您可以使用 `/tmp` (適用於 Linux、macOS 或 Unix) 或 `c:\temp` (適用於 Windows)，如果您要為測試目的而進行暫時翻製。這是用於這些範例中的目錄路徑。

> **Note**
>
> 您可以使用您要的任何目錄。如果您要複製長期使用的儲存庫，請考慮從工作目錄建立複製，而非用於暫時檔案的複製。如果是使用 `/tmp` 或 `c:\temp` 以外的目錄，當您依照這些指示時，請務必將該目錄以我們的目錄取代。

透過複製存 CodeCommit 放庫連 Connect 至儲存庫

如果您還沒有本機存放庫，請依照此程序中的步驟將 CodeCommit 存放庫複製到您的本機電腦。

1. 完成必要條件，包括設定 。

   > **Important**
   >
   > 如果尚未完成設定，您無法連接到或複製儲存庫。

2. 從 `/tmp` 目錄或 `c:\temp` 目錄，使用 Git 來執行 clone 命令。下列範例顯示如何複製美國東部 (俄亥俄) 區域名為*MyDemoRepo*的儲存庫。

   針對使用 Git 登入資料 或 AWS CLI隨附的登入資料協助程式的 HTTPS：

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

對於 HTTPS 使用git-remote-codecommit，假設默認配置文件並在以下內容中 AWS 區域 配置 AWS CLI：

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

針對 SSH：

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

在此範例中，`git-codecommit.us-east-2.amazonaws.com`是存放庫所在的美國東部 (俄亥俄) 區域的 Git 連線點，`MyDemoRepo`代表儲存 CodeCommit 庫的名稱，並`my-demo-repo`代表 Git 在目錄或目錄中建立的`/tmp`目`c:\temp`錄名稱。如需有關 AWS 區域 該支援 CodeCommit 及其 Git 連線的詳細資訊 AWS 區域，請參閱區域和 Git 連線端點。

> **Note**
>
> 當您在 Windows 作業系統上使用 SSH 以複製儲存庫時，您必須將 SSH 金鑰 ID 新增到連線字串，如下所示：
>
> ```
> git clone ssh://Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo
> ```
>
> 如需詳細資訊，請參閱 適用於 Windows 上的 SSH 連線 及 疑難排解。

Git 創建目錄後，它將 CodeCommit 存儲庫的副本拉下到新創建的目錄中。

如果 CodeCommit 存放庫是新的或空的，您會看到一則訊息，指出您正在複製空的存放庫。這是預期的行為。

> **Note**
>
> 如果您收到 Git 找不到 CodeCommit 儲存庫的錯誤訊息，或者您沒有連線到 CodeCommit 儲存庫的權限，請確定您已完成先決條件，包括將權限指派給 IAM 使用者，以及在 Git 和本機電腦 CodeCommit 上設定 IAM 使用者認證。同時，請確定您已指定正確的儲存庫名稱。

在您成功地將本機存放庫連接到 CodeCommit 儲存庫之後，您現在就可以開始從本機存放庫執行 Git 命令，以建立提交、分支和標記，並推送至儲存庫和從 CodeCommit 儲存庫中提取。

將本地回購 Connect 到存 CodeCommit 儲庫

如果您已經有本機存放庫，並且想要將儲存庫新增為遠端 CodeCommit儲存庫，請完成下列步驟。如果您已經擁有遠端儲存庫，而且想要將提交推送至 CodeCommit 該遠端儲存庫，請依照中的步驟執行推送提交到兩個存儲庫。

1. 完成先決條件。
2. 從命令提示符或終端，切換到本地 repo 目錄並運行該git remote add命令以將 CodeCommit 存儲庫添加為本地存儲庫的遠程存儲庫。

   例如，下列命令會將暱稱的遠端新增**origin**至 https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo：

   針對 HTTPS：

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

   針對 SSH：

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

   此命令不會傳回任何結果。
3. 要驗證您是否已將 CodeCommit 存儲庫添加為本地存儲庫的遠程庫，請運行該git remote -v命令，該命令應該創建類似於以下內容的輸出：

   針對 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)
   ```

   針對 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)
```

在您成功地將本機存放庫連接到 CodeCommit 儲存庫之後，您就可以開始從本機存放庫執行 Git 命令來建立提交、分支和標記，以及從 CodeCommit 儲存庫推送和提取。

共用儲 AWS CodeCommit 存庫

建立 CodeCommit 儲存庫之後，您可以與其他使用者共用。首先，請確定在存取時是否要使用聯合存取權、臨時登入資料或網路身分提供者 (例如 IAM 身分中心) CodeCommit，或是要與 IAM 使用者使用 Git 登入資料或 SSH 金鑰配對。如果您使用前者，則需要為身分提供者設定使用者、存取權和權限，然後提供使用者使用的指示git-remote-codecommit。如需詳細資訊，請參閱 HTTPS 連線的設定步驟 AWS CodeCommit與git-remote-codecommit 及 使用旋轉認證連線至AWS CodeCommit儲存庫。

您無法將 Git 認證或 SSH 金鑰配對與聯合存取或身分識別提供者搭配使用，但是許多 IDE 最適合使用這些認證。在這種情況下，請決定在複製和使用 Git 用戶端或 IDE 連接到存放庫時向使用者推薦哪種通訊協定 (HTTPS 或 SSH)。然後將 URL 和連線資訊傳送到您想要與其共用儲存庫的使用者。根據您的安全要求，共用儲存庫可能還需要建立 IAM 群組、將受管政策套用至該群組，以及編輯 IAM 政策以改善存取權限，或建立和使用 IAM 角色。

> **Note**
>
> 授予使用者對儲存庫的主控台存取之後，這些使用者即可直接在主控台中新增或編輯檔案，而不需設定 Git 用戶端或其他連線。如需詳細資訊，請參閱 建立檔案或將檔案新增至AWS CodeCommit儲存庫 及 編輯AWS CodeCommit儲存庫中檔案的內容。

這些指示的撰寫是假設您已完成設定 和建立 儲存庫中的步驟。

> **Note**
>
> 根據您的使用情況，您可能需要支付建立或存取存放庫的費用。如需詳細資訊，請參閱 CodeCommit 產品資訊頁面上的定價。

主題

- 選擇要與使用者共用的連線通訊協定
- 為儲存庫建立 IAM 政策
- 為儲存庫使用者建立 IAM 群組
- 與您的使用者分享連線資訊

選擇要與使用者共用的連線通訊協定

當您在中建立存放庫時 CodeCommit，會產生兩個端點：一個用於 HTTPS 連線，另一個用於 SSH 連線。兩者都提供網路上的安全連線。您的使用者可以使用任一通訊協定。無論您建議使用者採用哪個通訊協定，這兩個端點都保持在作用中。

HTTPS 連線需要以下兩者中的一個：

- Git 認證，IAM 使用者可以在 IAM 中為自己產生這些憑證。Git 登入資料是儲存庫的使用者設定並使用的最簡單方法。
- 儲存庫使用者必須在其認證設定檔中配置的存 AWS 取金鑰或角色。您可以設定 git-remote-codecommit (建議使用) 或 AWS CLI隨附的登入資料協助程式。根帳戶或聯合身分使用者只能使用這些方法。

SSH 連接需要使用者進行以下動作：

- 產生公私金鑰對。
- 存放公有金鑰。
- 將公開金鑰與其 IAM 使用者建立關聯。
- 在使用者自己的本機電腦上設定已知的主機檔案。
- 在使用者自己的本機電腦上建立和維護設定檔。

因為這是較複雜的設定程序，因此建議您選擇 HTTPS 和 Git 認證來連線到 CodeCommit.

如需 HTTPS、SSH、Git、git-remote-codecommit 和遠端儲存庫的詳細資訊，請參閱設定 、使用旋轉認證連線至AWS CodeCommit儲存庫或查詢您的 Git 文件。如需通訊協定的一般概觀，以及每個通訊協定如何與遠端儲存庫通訊的詳細資訊，請參閱概觀伺服器上的 Git - 通訊協定。

> **Note**
>
> 雖然 Git 支援各種連線通訊協定，但 CodeCommit 不支援使用不安全通訊協定的連線，例如本機通訊協定或一般 HTTP。

為儲存庫建立 IAM 政策

AWS 在 IAM 中提供三種受管政策，適用於 CodeCommit. 這些政策無法編輯，也無法套用至與您的 Amazon Web Services 帳戶相關聯的所有儲存庫。不過，您可以使用這些政策做為範本，以建立只套用至您想要共用的儲存庫的自訂受管政策。您的客戶受管政策可以特別套用至您想要共用的儲存庫。如需詳細資訊，請參閱受管政策和 IAM 使用者和群組。

> **Tip**
>
> 若要更精細地控制存放庫的存取權，您可以建立多個客戶受管政策，並將這些政策套用至不同的 IAM 使用者和群組。

如需有關檢閱受管政策的內容，以及使用政策來建立和套用許可的資訊，請參閱AWS CodeCommit 的身分驗證與存取控制。

為您的儲存庫建立客戶受管政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。
3. 在 [建立原則] 頁面上，選擇 [匯入受管理的原則]。
4. 在 [匯入受管原則] 頁面的 [篩選器策略] 中，輸入**AWSCodeCommitPowerUser**。選擇原則名稱旁邊的按鈕，然後選擇 [匯入]。
5. 在 建立政策頁面上，選擇 JSON。將 CodeCommit 動作`Resource`行的「*」部分取代為 CodeCommit 儲存庫的 Amazon 資源名稱 (ARN)，如下所示：

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

> **Tip**
>
> 若要尋找 CodeCommit 存放庫的 ARN，請移至 CodeCommit 主控台，從清單中選擇存放庫名稱，然後選擇 [設定]。如需詳細資訊，請參閱 檢視儲存庫詳細。

如果您希望此政策套用到多個儲存庫，請指定儲存庫的 ARN，將每個儲存庫新增為資源。在每個資源陳述式之間包含逗號，如下所示：

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

完成編輯後，請選擇 [檢閱原則]。

6. 在 [檢閱原則] 頁面的 [名稱] 中，輸入原則的新名稱 (例如 *AWSCodeCommitPowerUser-MyDemoRepo*)。選擇性地提供此原則的說明。
7. 選擇建立政策。

為儲存庫使用者建立 IAM 群組

若要管理存放庫的存取權，請為其使用者建立 IAM 群組，將 IAM 使用者新增至該群組，然後附加您在上一個步驟中建立的客戶受管政策。或者，您可以建立具有附加客戶管理策略的角色，並讓使用者擔任該角色。

如果您使用 SSH，則必須將另一個受管政策附加到 IAMUserSSHKeys 群組，這是允許使用者上傳其安全殼層公開金鑰，並將其與用來連線的 IAM 使用者建立關聯的 IAM 受管政策。 CodeCommit

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Groups (群組)，然後選擇 Create New Group (建立新的群組)。
3. 在 [設定群組名稱] 頁面的 [群組名稱] 中，輸入群組的名稱 (例如，*MyDemoRepoGroup*)，然後選擇 [下一步]。請考慮將儲存庫名稱包含於群組名稱中。

> **Note**
>
> 此名稱在 Amazon Web Services 帳戶中必須是唯一的。

4. 選取您在上一節建立的客戶管理策略旁邊的核取方塊 (例如，AWSCodeCommitPowerUser-MyDemoRepo)。
5. 在 Review (檢閱) 頁面上，選擇 Create Group (建立群組)。IAM 會使用已附加的指定政策建立此群組。該群組會出現在與您的 Amazon Web Services 帳戶關聯的群組清單中。
6. 從清單中選擇您的群組。
7. 在群組摘要頁面上，選擇 Users (使用者) 標籤，然後選擇 Add Users to Group (新增使用者到群組)。在顯示與 Amazon Web Services 帳戶關聯的所有使用者的清單中，選取您要允許存取 CodeCommit 存放庫的使用者旁邊的核取方塊，然後選擇 [新增使用者]。

> **Tip**
>
> 您可以使用 [Search (搜尋)] 方塊，依名稱快速尋找使用者。

8. 新增使用者後，請關閉 IAM 主控台。

與您的使用者分享連線資訊

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在區域選取器中，選擇建立儲存庫的 AWS 區域 位置。儲存庫特定於 AWS 區域. 如需詳細資訊，請參閱 區域和 Git 連線端點。
3. 在 Repositories (儲存庫) 頁面上，選擇您要共用的儲存庫。
4. 在 Clone URL (複製 URL) 中，選擇您要讓使用者使用的通訊協定。這樣會複製該連線通訊協定的複製 URL。
5. 向您的使用者傳送複製 URL 以及任何其他指示，例如安裝 AWS CLI、設定設定檔或安裝 Git。請務必包含連線通訊協定的組態資訊 (例如 HTTPS)。

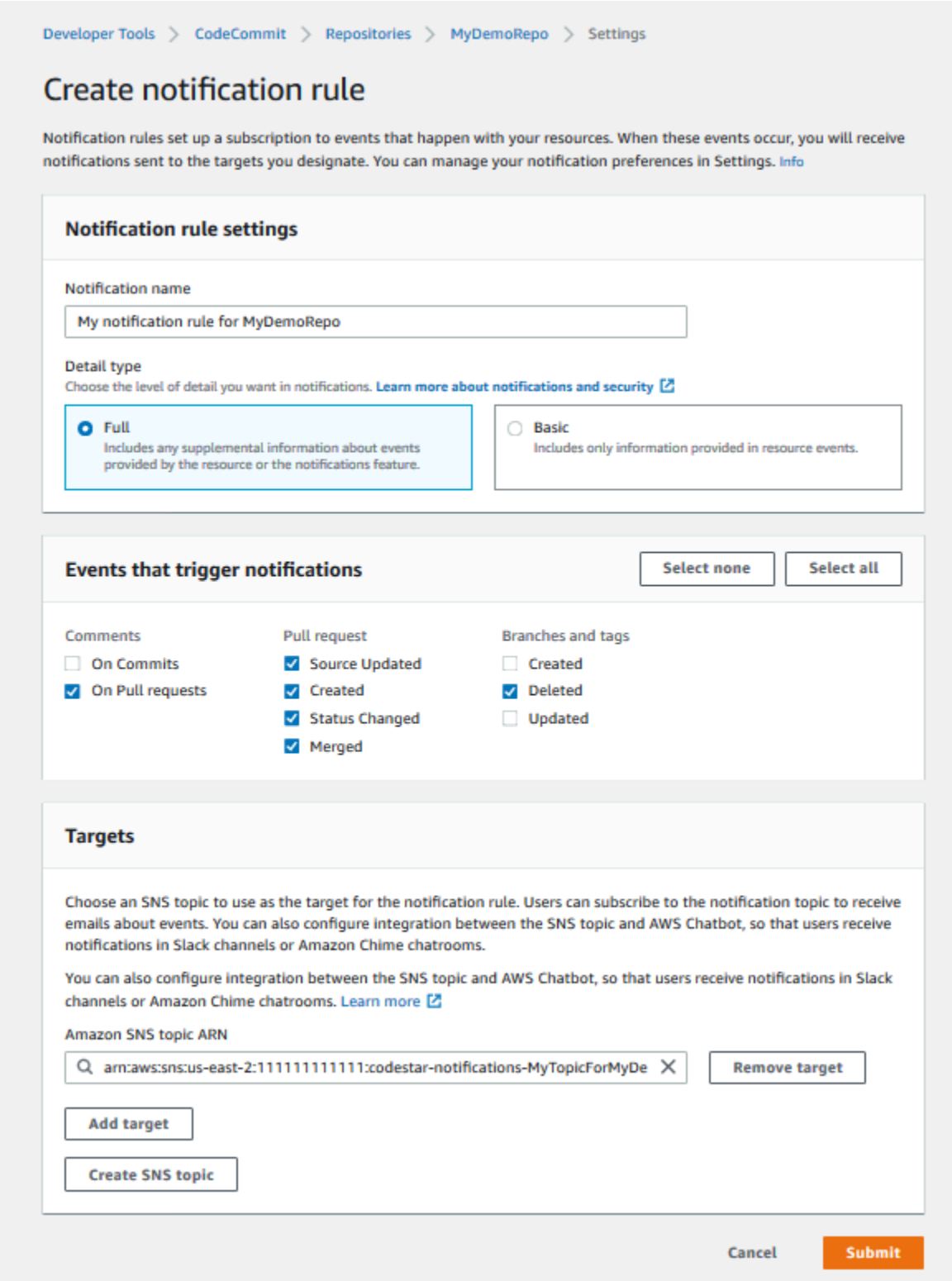
下列範例電子郵件為使用美國東部 (俄亥俄) (us-east-2) MyDemoRepo 區域的 HTTPS 連線通訊協定和 Git 認證連線至儲存庫的使用者提供相關資訊。此電子郵件的撰寫是假設使用者已安裝 Git 且熟悉其使用方式。

```
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.
```

您可以在中找到完整的設定說明設定 。

設定 AWS CodeCommit 存放庫中事件的通知

您可以設定儲存庫的通知規則，使得儲存庫使用者會收到有關您指定的儲存庫事件類型的電子郵件。當事件符合通知規則設定時會傳送通知。您可以建立用於通知的 Amazon SNS 主題，或使用 Amazon Web Services 帳戶中現有的主題。您可以使用主 CodeCommit 控台和 AWS CLI 來設定通知規則。



主題

- 使用儲存庫通知規則
- 建立通知規則

- 變更或停用通知
- 刪除通知

使用儲存庫通知規則

設定通知規則可在在有人採取的動作會影響其他使用者時傳送電子郵件，藉以協助您的儲存庫使用者。例如，您可以設定一個通知規則，在對遞交進行評論時傳送通知。在此組態中，當儲存庫使用者對遞交中的某個程式碼行進行評論時，其他儲存庫中使用者會收到一封電子郵件。他們可以登入和檢視評論。對評論的回應也會產生電子郵件，使得儲存庫使用者可掌握資訊。

通知規則與存放庫觸發器不同，而且也與您在 2019 年 11 月 5 日之前在 CodeCommit 主控台中設定的通知不同。

- 雖然您可以將觸發器設定為使用 Amazon SNS 傳送有關某些儲存庫事件的電子郵件，但這些事件僅限於操作事件，例如建立分支和將程式碼推送至分支。觸發程序不會使用 CloudWatch 事件規則來評估存放庫事件。它們的範圍更有限。如需如何使用觸發的詳細資訊，請參閱管理儲存庫的觸發。
- 在 2019 年 11 月 5 日之前設定的通知可用的事件類型較少，且無法設定為與 Amazon Chime 聊天室或 Slack 頻道整合。您可以繼續使用在 2019 年 11 月 5 日之前設定的通知，但無法建立此類型的通知。請改為建立並使用通知規則。建議您使用通知規則，並停用或刪除 2019 年 11 月 5 日之前建立的通知。如需詳細資訊，請參閱 建立通知規則 及 刪除通知。

建立通知規則

您可以使用通知規則來通知使用者重要的變更，例如：在儲存庫中建立提取請求時。通知規則指定用於傳送通知的事件和 Amazon SNS 主題。如需詳細資訊，請參閱什麼是通知？

> **Note**
>
> 歐洲 (米蘭) 地區不提供此功能。若要瞭解如何在該區域的可用體驗中設定通知，請參閱設定儲存庫通知。

您可以使用主控台或 AWS CLI 建立的通知規則 AWS CodeCommit。

建立通知規則 (主控台)

1. 請登入 AWS Management Console 並開啟 CodeCommit 主控台，網址為 https://console.aws.amazon.com/codecommit/。
2. 選擇 Repositories (儲存庫)，然後選擇您要新增通知規則的儲存庫。
3. 在儲存庫頁面中，選擇 Notify (通知)，然後選擇 Create notification rule (建立通知規則)。您也可以移至儲存庫的 Settings (設定) 頁面，然後選擇 Create notification rule (建立通知規則)。
4. 在 Notification name (通知名稱) 中，輸入規則的名稱。
5. 如果您只想要提供給 Amazon 的資訊 EventBridge 包含在通知中，請在 [詳細資料類型] 中選擇 [基本]。如果您想要包含提供給 Amazon 的資訊以 EventBridge 及可能由 CodeCommit 或通知管理員提供的資訊，請選擇「完整」。

   如需詳細資訊，請參閱了解通知內容與安全性。
6. 在 Events that trigger notifications (觸發通知的事件) 中，選取您要傳送通知的事件。如需詳細資訊，請參閱儲存庫上通知規則的事件。
7. 在 Targets (目標) 中，執行下列其中一個動作：

   - 如果您已設定要與通知搭配使用的資源，請在 Choose target type (選擇目標類型) 中，選擇 AWS Chatbot (Slack) 或 SNS topic (SNS 主題)。在選擇目標中，選擇 Amazon SNS 主題的用戶端名稱 (針對在中設定的 Slack 用戶端 AWS Chatbot) 或 Amazon 資源名稱 (ARN) (適用於已設定通知所需政策的 Amazon SNS 主題)。
   - 如果您尚未設定要與通知搭配使用的資源，請選擇 Create target (建立目標)，然後選擇 SNS topic (SNS 主題)。在 codestar-notifications- 之後，提供主題名稱，然後選擇 Create (建立)。

   > **Note**
   >
   > - 如果您在建立通知規則的過程中建立 Amazon SNS 主題，將會為您套用允許通知功能將事件發佈至主題的政策。使用針對通知規則建立的主題，有助於確保您只訂閱需要接收此資源相關通知的使用者。
   > - 您無法建立 AWS Chatbot 用戶端做為建立通知規則的一部分。如果您選擇 AWS Chatbot （ Slack ），您將看到一個按鈕，指示您在中 AWS Chatbot配置客戶端。選擇該選項會開啟主 AWS Chatbot 控台。如需詳細資訊，請參閱設定通知與之間的整合 AWS Chatbot。

- 如果您想要使用現有的 Amazon SNS 主題作為目標，除了該主題可能存在的任何其他政策之外，還必須新增AWS CodeStar通知的必要政策。如需詳細資訊，請參閱為通知設定 Amazon SNS 主題和了解通知內容與安全性。

8. 若要完成建立規則，請選擇 Submit (提交)。
9. 您必須先訂閱使用者訂閱規則的 Amazon SNS 主題，才能收到通知。如需詳細資訊，請參閱為目標的 Amazon SNS 主題訂閱使用者。您也可以設定通知之間的整合，並 AWS Chatbot 將通知傳送到 Amazon Chime 聊天室。如需詳細資訊，請參閱設定通知與之間的整合 AWS Chatbot。

建立通知規則 (AWS CLI)

1. 在終端機或命令提示字元中，執行 create-notification rule 命令以產生 JSON 架構：

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

您可以將檔案命名為任何您想要的名稱。在此範例中，檔案命名為 *rule.json*。

2. 在純文字編輯器中開啟 JSON 檔案，並編輯成包含您想要用於規則的資源、事件類型和目標。下列範例顯示的通知規則是**MyNotificationRule** AWS 針對在識別碼為 *123* 456789012 的帳戶*MyDemoRepo*中命名的儲存庫。具有完整詳細資料類型的通知會傳送至建立分支和標籤*MyNotificationTopic*時命名的 Amazon SNS 主題：

```
{
    "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"
}
```

儲存檔案。

3. 在終端機或命令列中，再次執行 create-notification-rule 命令，使用您剛編輯的檔案建立通知規則：

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

4. 如果成功，此命令會傳回通知規則的 ARN，如下所示：

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

變更或停用通知

您可以使用主 AWS CodeCommit 控台變更 2019 年 11 月 5 日之前建立之通知的設定方式，包括傳送電子郵件給使用者的事件類型，以及用來傳送有關儲存庫之電子郵件的 Amazon SNS 主題。您也可以使用主 CodeCommit控台來管理訂閱此主題的電子郵件地址和端點清單，或停用通知。

變更通知設定

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要設定 2019 年 11 月 5 日前建立之通知所在的儲存庫名稱。
3. 在導覽窗格中，選擇 Settings (設定)，然後選擇 Notifications (通知)。如果有橫幅通知您有通知而非通知規則，請選擇 Manage existing notifications (管理現有通知)。
4. 選擇編輯。
5. 進行變更，然後選擇 Save (儲存變更)。

停用通知是暫時防止使用者接收有關儲存庫事件之電子郵件的一個簡單方法。

若要永久刪除 2019 年 11 月 5 日之前建立的通知，請依照 刪除通知 中的步驟進行操作。

若要關閉通知

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要停用通知所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Settings (設定)，然後選擇 Notifications (通知)。選擇 Manage existing notifications (管理現有通知)。
4. 選擇 Edit (編輯), 然後在 Event status (事件狀態) 中使用滑桿來關閉 Enable notifications (啟用通知)。選擇儲存。
5. 事件狀態會變更為 Disabled (已停用)。不會傳送關於事件的電子郵件。停用通知時，會自動停用存放庫的 CloudWatch 事件規則。請勿在「 CloudWatch 事件」主控台中手動變更其狀態。

刪除通知

如果您不想再使用在 2019 年 11 月 5 日之前為儲存庫建立的通知，您可以刪除與該通知相關聯的 Amazon CloudWatch 活動規則。這將自動刪除通知。它不會刪除任何用於通知的訂閱或 Amazon SNS 主題。

> **Note**
>
> 如果您從主控台變更儲存庫的名稱，不需修改，2019 年 11 月 5 日前建立的通知即可繼續運作。不過，如果您從命令列或使用 API 變更儲存庫的名稱，通知將不再運作。還原通知最簡單的方法是刪除通知設定，然後再次對其進行設定。

刪除通知設定

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要移除 2019 年 11 月 5 日前建立之通知的所在儲存庫名稱。
3. 在導覽窗格中，選擇 Settings (設定)，然後選擇 Notifications (通知)。如果有橫幅通知您有通知而非通知規則，請選擇 Manage existing notifications (管理現有通知)。
4. 在CloudWatch 事件規則中，複製為通知建立的規則名稱。
5. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。

6. 在 Events (事件) 中，選擇 Rules (規則)。在 Name (名稱) 中，貼上為通知建立的規則名稱。選擇規則，然後在 Actions (動作) 中選擇 Delete (刪除)。
7. (選擇性) 若要在刪除通知設定後變更或刪除用於通知的 Amazon SNS 主題，請前往 Amazon SNS 主控台 https://console.aws.amazon.com/sns/v3/home。如需詳細資訊，請參閱 Amazon 簡易通知服務開發人員指南中的清理。

標記儲存庫 AWS CodeCommit

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。 AWS 標籤與 Git 標籤不同，這些標籤可以應用於提交。每個 AWS 標籤都有兩個部分：

- 標籤鍵 (例如，`CostCenter`、`Environment`、`Project` 或 `Secret`)。標籤鍵會區分大小寫。
- 一個名為標籤值 (例如，`111122223333`、`Production` 或團隊名稱) 的選用欄位。忽略標籤值基本上等同於使用空字串。與標籤鍵相同，標籤值會區分大小寫。

這些合稱為鍵值組。對於儲存庫上標籤數量的限制以及標籤金鑰和值的限制，請參閱 限制。

標籤可協助您識別和整理資 AWS 源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。例如，您可以將相同的標籤指派給指派給 Amazon S3 CodeCommit 儲存貯體的儲存庫。有關標記策略的詳細資訊，請參閱標記 AWS 資源。

在中 CodeCommit，主要資源是存放庫。您可以使用 CodeCommit 控制台 AWS CLI、 CodeCommit API 或 AWS SDK 來新增、管理和移除存放庫的標籤。除了使用標籤識別、組織和追蹤儲存庫之外，您還可以在 IAM 政策中使用標籤來協助控制誰可以檢視您的儲存庫並與之互動。如需以標籤為基礎的存取政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

主題

- 新增標籤至儲存庫
- 檢視儲存庫的標籤
- 檢視儲存庫的標籤
- 從儲存庫中移除標籤

新增標籤至儲存庫

將標籤新增至儲存庫可協助您識別和組織資 AWS 源，並管理資源的存取權限。首先，您將一或多個標籤 (金鑰值組) 到儲存庫。請記住儲存庫的標籤數有所限制。金鑰和值欄位可使用的字數有所限制。如

需詳細資訊，請參閱限制。擁有標籤之後，您可以建立 IAM 政策，以根據這些標記管理存放庫的存取權限。您可以使用主 CodeCommit 控台或將標籤新增 AWS CLI 至存放庫。

> **Important**
>
> 新增標籤到儲存庫可能會影響存取該儲存庫。在將標籤新增至儲存庫之前，請務必檢閱任何可能使用標籤來控制對資源 (例如儲存庫) 的存取權限的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

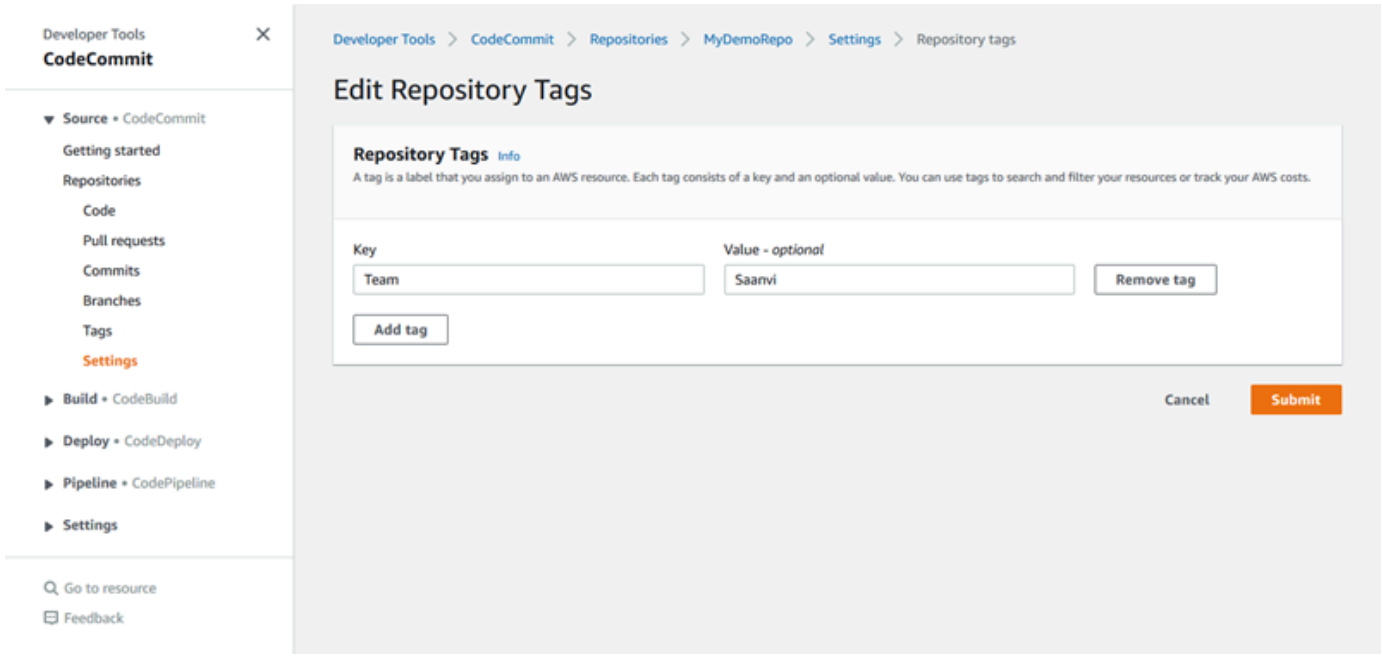
當您建立時，如需有關將標籤新增到儲存庫的詳細資訊，請參閱 創建一個存儲庫（控制台）。

主題

將標籤新增至儲存庫 (主控台)

您可以使用 CodeCommit 控制台將一個或多個標籤添加到 CodeCommit 存儲庫中。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要新增標籤所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。選擇 Repository tags (儲存庫標籤)。
4. 如果沒有已經新增到儲存庫的標籤，請選擇 Add tag (新增標籤)。否則，選擇 Edit (編輯)，然後選擇 Add tag (新增標籤)。
5. 在 Key (金鑰) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。



6. (選用) 若要新增另一個標籤，再選擇 Add tag (新增標籤) 一次。
7. 當您完成新增標籤的作業時，請選擇 Submit (提交)。

將標籤新增至儲存庫 (AWS CLI)

請依照下列步驟使 AWS CLI 用將標籤新增至 CodeCommit 存放庫。若要在建立儲存庫將標籤新增到儲存庫，請參閱建立儲存庫 (AWS CLI)。

在這些步驟中，我們假設您已經安裝新版 AWS CLI 或更新到最新版本。如需詳細資訊，請參閱安裝 AWS Command Line Interface。

在終端機或命令列，執行 tag-resource 命令，指定您要新增標籤之儲存庫的 Amazon Resource Name (ARN)，和您想新增之標籤的索引鍵和值。您可以新增多個標籤到儲存庫。例如，若要標記以兩個標籤命名*MyDemoRepo*的儲存庫，一個名為 S *tatus* 的標籤金鑰 (標籤值為 *Secret*)，以及一個名為 *Team* 的標籤鍵加上 *San* vi 的標籤鍵：

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

若成功，此命令不會傳回任何內容。

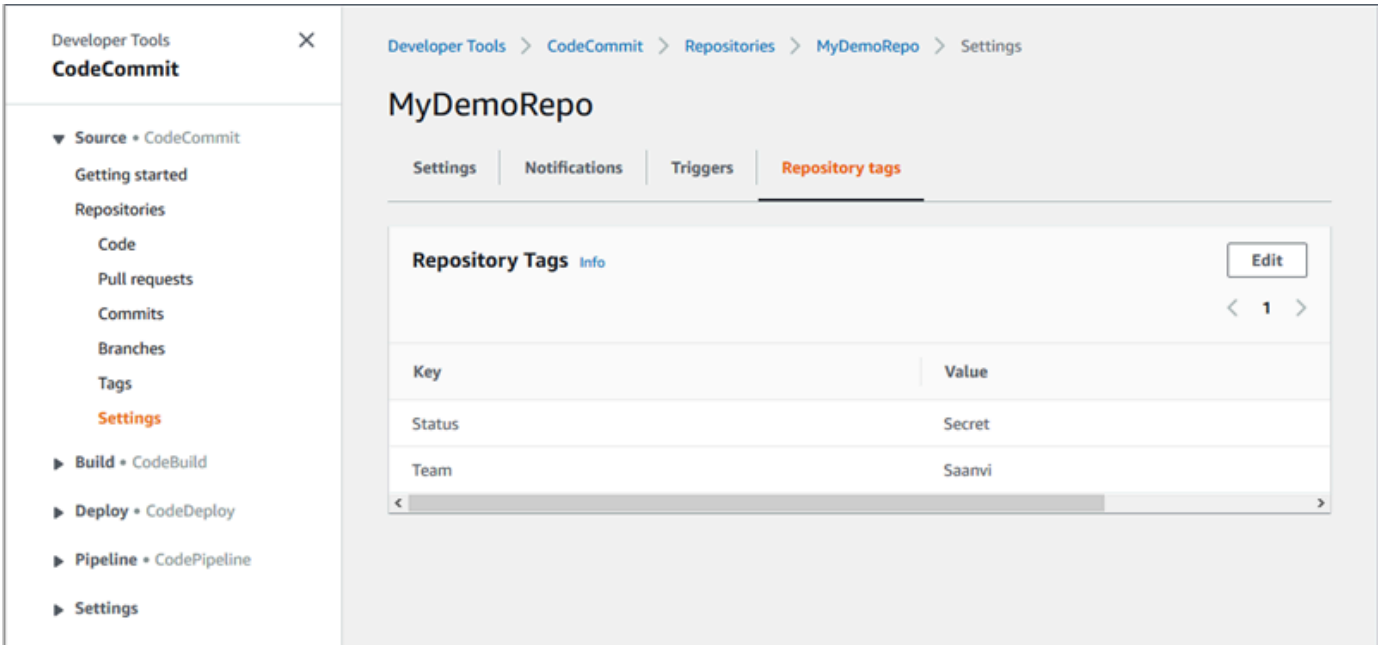
檢視儲存庫的標籤

標籤可協助您識別和組織資 AWS 源，並管理資源的存取權限。有關標記策略的詳細資訊，請參閱標記 AWS 資源。如需以標籤為基礎的存取政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

檢視儲存庫的標籤 (主控台)

您可以使用主 CodeCommit 控台來檢視與 CodeCommit 儲存庫相關聯的標籤。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢視標籤所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。選擇 Repository tags (儲存庫標籤)。



檢視儲存庫的標籤 (AWS CLI)

請遵循下列步驟 AWS CLI 來使用檢視 CodeCommit 存放庫的 AWS 標籤。若未新增標籤，傳回的清單空白。

在終端機或命令列上執行 list-tags-for-resource 命令。例如，若要檢視以 ARN ARN*: awn: ####*
#:US-## 2:111111111111 ######MyDemoRepo############: MyDemoRepo

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

若成功，此命令會傳回類似如下的資訊：

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

檢視儲存庫的標籤

您可以變更與儲存庫相關聯標籤的值。您也可以變更金鑰名稱，等於移除目前的標籤，並新增一個不同的新的名稱和相同的值作為其他金鑰。請記住金鑰和值欄位可使用的字數有所限制。如需詳細資訊，請參閱限制。

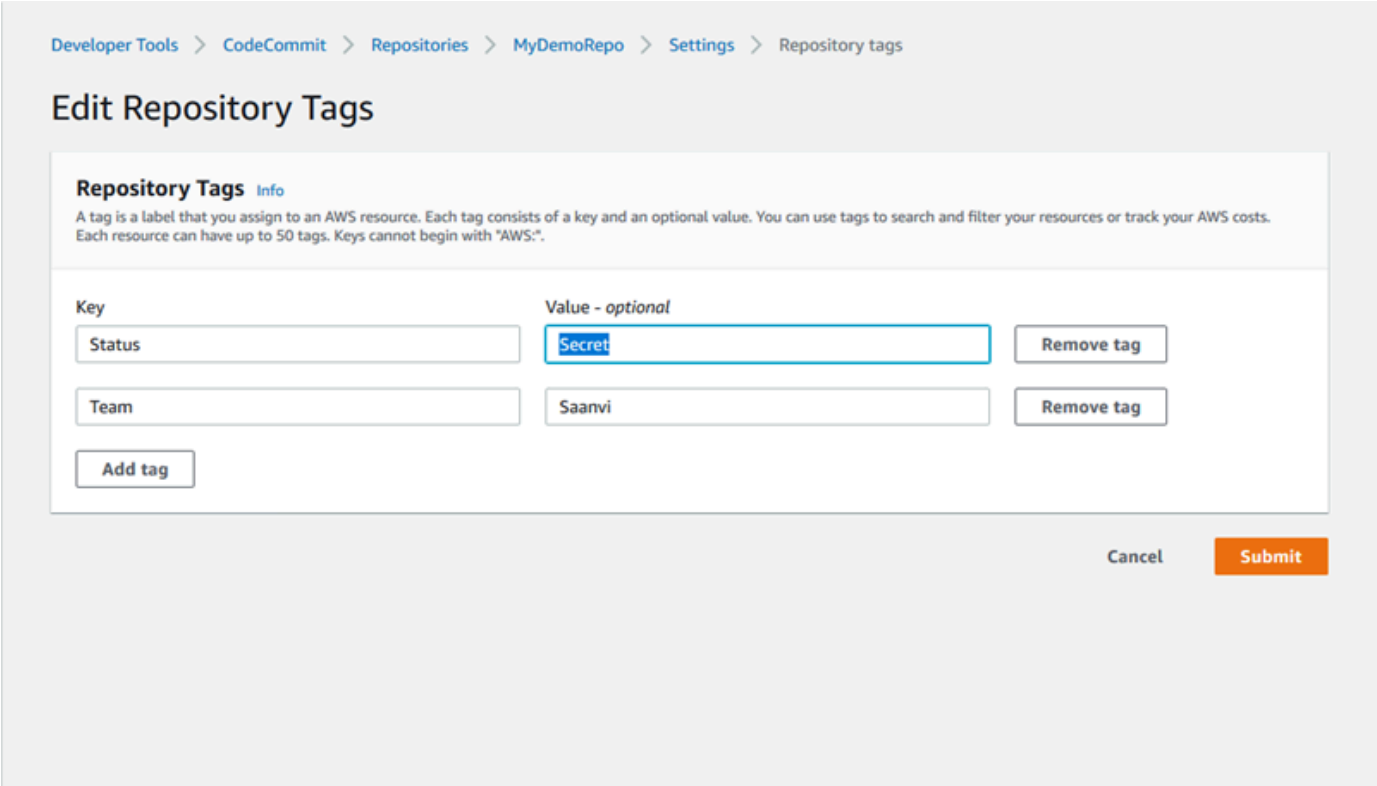
> **Important**
>
> 編輯儲存庫標籤可能會影響存取該儲存庫。在編輯儲存庫標籤的名稱 (金鑰) 或值之前，請務必檢閱任何可能使用標籤金鑰或值的 IAM 政策，以控制對資源 (例如儲存庫) 的存取。如需以標籤為基礎的存取政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

編輯儲存庫的標籤 (主控台)

您可以使用主 CodeCommit 控台編輯與 CodeCommit 存放庫相關聯的標籤。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要編輯標籤所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。選擇 Repository tags (儲存庫標籤)。
4. 選擇編輯。

5.



執行以下任意一項：

- 若要變更標籤，請在 Key (金鑰) 輸入新的名稱。變更標籤名稱等於移除標籤並使用新的金鑰名稱新增一個新的標籤。
- 若要變更標籤的值，請輸入新的值。如果您想要變更值為沒有，請刪除目前的值並保留欄位空白。

6. 當您完成編輯標籤，選擇 Submit (提交)。

編輯儲存庫的標籤 (AWS CLI)

請依照下列步驟使用 AWS CLI 來更新 CodeCommit 存放庫的標籤。您可以變更現有索引鍵的值或新增其他索引鍵。

在終端機或命令列，執行 tag-resource 命令，指定您要更新標籤之儲存庫的 Amazon Resource Name (ARN)，並指定標籤索引鍵和標籤值：

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

從儲存庫中移除標籤

您可以移除一或多個儲存庫關聯的標籤。移除標籤並不會從與該標籤相關聯的其他 AWS 資源中刪除該標籤。

> **Important**
>
> 移除儲存庫標籤可能會影響存取該儲存庫。從儲存庫移除標籤之前，請務必檢閱任何可能使用標籤的金鑰或值來控制存取資源 (例如儲存庫) 的 IAM 政策。如需以標籤為基礎的存取政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

從儲存庫中移除標籤 (主控台)

您可以使用 CodeCommit 控制台移除標籤與 CodeCommit 存放庫之間的關聯。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要移除標籤所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。選擇 Repository tags (儲存庫標籤)。
4. 選擇編輯。
5. 尋找您想要移除的標籤，然後選擇 Remove tag (移除標籤)。
6. 當您完成移除標籤，請選擇 Submit (提交)。

從儲存庫中移除標籤 (AWS CLI)

請依照下列步驟使 AWS CLI 用從 CodeCommit 存放庫移除標籤。移除標籤並不會將其刪除，只會移除標籤和儲存庫之間的關聯性。

> **Note**
>
> 如果您刪除 CodeCommit 存放庫，所有標籤關聯都會從已刪除的存放庫中移除。您不需要在刪除儲存庫之前移除標籤。

在終端機或命令列，執行 untag-resource 命令，指定您要移除標籤之儲存庫的 Amazon Resource Name (ARN)，和您想移除之標籤的標籤索引鍵。例如，若要移除使用標籤鍵 S *tatus* 命名*MyDemoRepo*的儲存庫上的標籤：

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

若成功，此命令不會傳回任何內容。若要驗證與儲存庫關聯的標籤，請執行 list-tags-for-resource 命令。

管理 AWS CodeCommit 儲存庫的觸發器

您可以設定 CodeCommit 儲存庫，讓程式碼推送或其他事件觸發動作，例如從 Amazon 簡單通知服務 (Amazon SNS) 傳送通知，或叫用中的函數。 AWS Lambda您最多可以為每個 CodeCommit 儲存庫建立 10 個觸發程序。

設定觸發通常基於下列原因：

- 每當有資料推送到儲存庫時，即傳送電子郵件給訂閱使用者。
- 當有資料推送到儲存庫的主要分支後，便通知外部建置系統以開始建置。

通知外部建置系統等案例需要撰寫 Lambda 函數，才能與其他應用程式互動。電子郵件案例只需要建立 Amazon SNS 主題即可。

本主題說明如何設定允許 CodeCommit 在 Amazon SNS 和 Lambda 中觸發動作的許可。其中也包含用於建立、編輯、測試和刪除觸發的範例連結。

主題

建立資源並新增權限 CodeCommit

您可以將 Amazon SNS 主題和 Lambda 函數與觸發器整合在中 CodeCommit，但您必須先建立資源，然後使用政策授予 CodeCommit 與這些資源互動的權限來設定資源。您必須在與 CodeCommit 存放庫相同 AWS 區域 的資源中建立資源。例如，如果存放庫位於美國東部 (俄亥俄州) (us-east-2)，則 Amazon SNS 主題或 Lambda 函數必須位於美國東部 (俄亥俄)。

- 對於 Amazon SNS 主題，如果 Amazon SNS 主題是使用與 CodeCommit 儲存庫相同的帳戶建立，則不需要設定其他 IAM 政策或許可。您可以在建立並訂閱 Amazon SNS 主題後立即建立 CodeCommit 觸發器。
  - 如需在 Amazon SNS 中建立主題的詳細資訊，請參閱 Amazon SNS 文件。
  - 如需使用 Amazon SNS 將訊息傳送至 Amazon SQS 佇列的相關資訊，請參閱 Amazon SNS 開發人員指南中的將訊息傳送至 Amazon SQS 佇列。
  - 如需使用 Amazon SNS 呼叫 Lambda 函數的相關資訊，請參閱 Amazon SNS 開發人員指南中的叫用 Lambda 函數。
- 如果您想要將觸發器設定為在其他 AWS 帳戶中使用 Amazon SNS 主題，則必須先使用允許發佈 CodeCommit 到該主題的政策來設定該主題。如需詳細資訊，請參閱 範例 1：建立允許跨帳戶存取 Amazon SNS 主題的政策。
- 您可以在 Lambda 主控台中建立觸發器來設定 Lambda 函數，做為函數的一部分。這是最簡單的方法，因為在 Lambda 主控台中建立的觸發程序會自動包含叫用 Lambda 函數所需的權限。CodeCommit 如果您在中建立觸發器 CodeCommit，則必須包含允許 CodeCommit 呼叫函數的原則。如需詳細資訊，請參閱 為現有 Lambda 函數建立觸發器 及 範例 3：建立與 CodeCommit 觸發器AWS Lambda整合的策略。

範例：建立 Amazon SNS 主題的 AWS CodeCommit 觸發器

您可以為 CodeCommit 儲存庫建立觸發器，以便該儲存庫中的事件觸發來自 Amazon Simple Notification Service (Amazon SNS) 主題的通知。您可能想要建立 Amazon SNS 主題的觸發器，讓使用者能夠訂閱有關儲存庫事件的通知，例如刪除分支。您也可以充分利用 Amazon SNS 主題與其他服務的整合，例如 Amazon Simple Queue Service (Amazon SQS) 和 AWS Lambda.

> **Note**
>
> - 您必須將觸發器指向現有的 Amazon SNS 主題，該主題為回應儲存庫事件所採取的動作。如需有關建立和訂閱 Amazon SNS 主題的詳細資訊，請參閱 Amazon 簡單通知服務入門。

- CodeCommit 觸發程序不支援 Amazon SNS FIFO (先進先出) 主題。

主題

針對 CodeCommit儲存庫 (主控台) 建立 Amazon SNS 主題的觸發器

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以針對儲存庫事件建立觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)，然後選擇 Triggers (觸發)。
4. 選擇 Create trigger (建立觸發)，然後執行下列動作：
   - 在觸發器名稱中，輸入觸發程序的名稱 (例如，*MyFirstTrigger*)。
   - 在事件中，選擇觸發 Amazon SNS 主題的存放庫事件以傳送通知。

     如果您選擇 All repository events (所有儲存庫事件)，則無法選擇其他任何事件。若要選擇事件子集，請移除 All repository events (所有儲存庫事件)，然後從清單中選擇一或多個事件。例如，如果您希望觸發程序只在使用者在 CodeCommit存放庫中建立分支或標籤時執行，請移除 [所有存放庫事件]，然後選擇 [建立分支或標記]。
   - 如果您希望將觸發套用到儲存庫的所有分支，請在 Branches (分支) 中保持空白選擇，因為這個預設選項會自動將觸發套用到所有分支。如果您希望此觸發只套用到特定分支，請從儲存庫分支清單中選擇最多 10 個分支名稱。
   - 在選擇要使用的服務中，選擇 Amazon SNS。
   - 在 Amazon SNS 中，從清單中選擇主題名稱或輸入主題的 ARN。

   > **Note**
   >
   > CodeCommit 觸發程序不支援 Amazon SNS FIFO (先進先出) 主題。您必須選擇類型設定為標準的 Amazon SNS 主題。如果您想要使用 Amazon SNS FIFO 主題，則必須針對將 SNS FIFO 主題設定為其目標的 CodeCommit 事件設定 Amazon Eventbridge 接規則。

- 在自訂資料中，提供您希望在 Amazon SNS 主題傳送的通知中包含的任何選擇性資訊 (例如，開發人員在討論此儲存庫中的開發時使用的 IRC 頻道名稱)。此欄位是字串。無法用於傳遞任何動態參數。

5. (選用) 選擇 Test trigger (測試觸發)。此步驟可協助您確認是否已正確設定 CodeCommit 和 Amazon SNS 主題之間的存取權限。它會使用 Amazon SNS 主題，使用儲存庫中的資料 (如果有的話) 傳送測試通知。如果沒有實際的資料可用，測試通知會包含範例資料。
6. 選擇 Create trigger (建立觸發)，以完成建立觸發。

為 CodeCommit 儲存庫建立 Amazon SNS 主題的觸發器 (AWS CLI)

您也可以使用命令列為 Amazon SNS 主題建立觸發器，以回應 CodeCommit 儲存庫事件，例如有人將提交推送至您的儲存庫時。

建立 Amazon SNS 主題的觸發器

1. 開啟純文字編輯器，並建立 JSON 檔案，在其中指定：

   - Amazon SNS 主題名稱。

     > **Note**
     >
     > CodeCommit 觸發程序不支援 Amazon SNS FIFO (先進先出) 主題。您必須選擇類型設定為標準的 Amazon SNS 主題。如果您想要使用 Amazon SNS FIFO 主題，則必須針對將 SNS FIFO 主題設定為其目標的 CodeCommit 事件設定 Amazon Eventbridge 接規則。

   - 您要使用此觸發來監控的儲存庫和分支。(如果您不指定任何分支，則觸發會套用到儲存庫中的所有分支。)
   - 啟動此觸發的事件。

   儲存檔案。

   ```
   ##################MyDemoRepo################## MySNSTopic # Amazon SNS # ##############
   ```

   ```
   {
   ```

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

儲存庫的每個觸發在 JSON 中都必須有一個觸發區塊。若要為儲存庫建立多個觸發，請在 JSON 中包含多個觸發區塊。請記住，此檔案中建立的所有觸發是針對指定的儲存庫。您無法在單一 JSON 檔案中為多個儲存庫建立觸發。例如，如果您想要為儲存庫建立兩個觸發，則可以建立含有兩個觸發區塊的 JSON 檔案。在下列範例中，第二個觸發未指定分支，因此該觸發會套用到所有分支：

```
{
    "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"
            ]
        }
    ]
}
```

您可以為您指定的事件建立觸發，例如將遞交推送至儲存庫時。事件類型包括：

- `all` 代表指定的儲存庫和分支中的所有事件。
- `updateReference` 表示遞交推送到指定的儲存庫和分支。
- `createReference` 表示指定的儲存庫中建立新的分支或標籤。
- `deleteReference` 表示指定的儲存庫中刪除分支或標籤。

> **Note**
>
> 您可以在一個觸發中使用多個事件類型。不過，如果您指定 `all`，則無法指定其他事件。

若要查看有效事件類型的完整清單，請在終端機或命令提示字元中輸入 aws codecommit put-repository-triggers help。

此外，您可以在 `customData` 中包含字串 (例如，開發人員在此儲存庫中討論開發時所使用的 IRC 管道名稱)。此欄位是字串。無法用於傳遞任何動態參數。此字串會附加為回應觸發程序而傳回的 CodeCommit JSON 屬性。

2. (選用) 在終端機或命令提示字元中，執行 test-repository-triggers 命令。此測試使用儲存庫中的範例資料 (如果沒有資料可用，則產生範例資料) 傳送通知給 Amazon SNS 主題的訂閱者。例如，以下內容用於測試名為 trigger *er.json ####### JSON* 是否有效，並且 CodeCommit 可以發佈到 Amazon SNS 主題：

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

若成功，此命令會傳回類似如下的資訊：

```
{
    "successfulExecutions": [
        "MyFirstTrigger"
```

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

3. 在終端機或命令提示字元中，執行put-repository-triggers指令以在中建立觸發器 CodeCommit。例如，若要使用名為 *trigger.json* 的 JSON 檔案來建立觸發：

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

   此命令會傳回類似以下的組態 ID：

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

4. 若要檢視觸發的組態，請執行 get-repository-triggers 命令，並指定儲存庫的名稱：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   此命令會傳回為儲存庫設定之所有觸發的結構，類似如下：

   ```
   {
       "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. 為了測試觸發本身的功能，請進行遞交並推送至您已設定觸發的儲存庫。您應該會看到來自 Amazon SNS 主題的回應。例如，如果您將 Amazon SNS 主題設定為傳送電子郵件，您應該會在訂閱該主題的電子郵件帳戶中看到來自 Amazon SNS 的電子郵件。

以下是從 Amazon SNS 傳送以回應推送至 CodeCommit儲存庫的電子郵件的範例輸出：

```
{
  "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",
     }
  ]
}
```

範例：建立 AWS Lambda 函數的 AWS CodeCommit 觸發器

您可以為存放庫建立觸發器，以便 CodeCommit 儲存庫中的事件叫用 Lambda 函數。在此範例中，您會建立 Lambda 函數，該函數會傳回用來將儲存庫複製到 Amazon CloudWatch 日誌的 URL。

主題

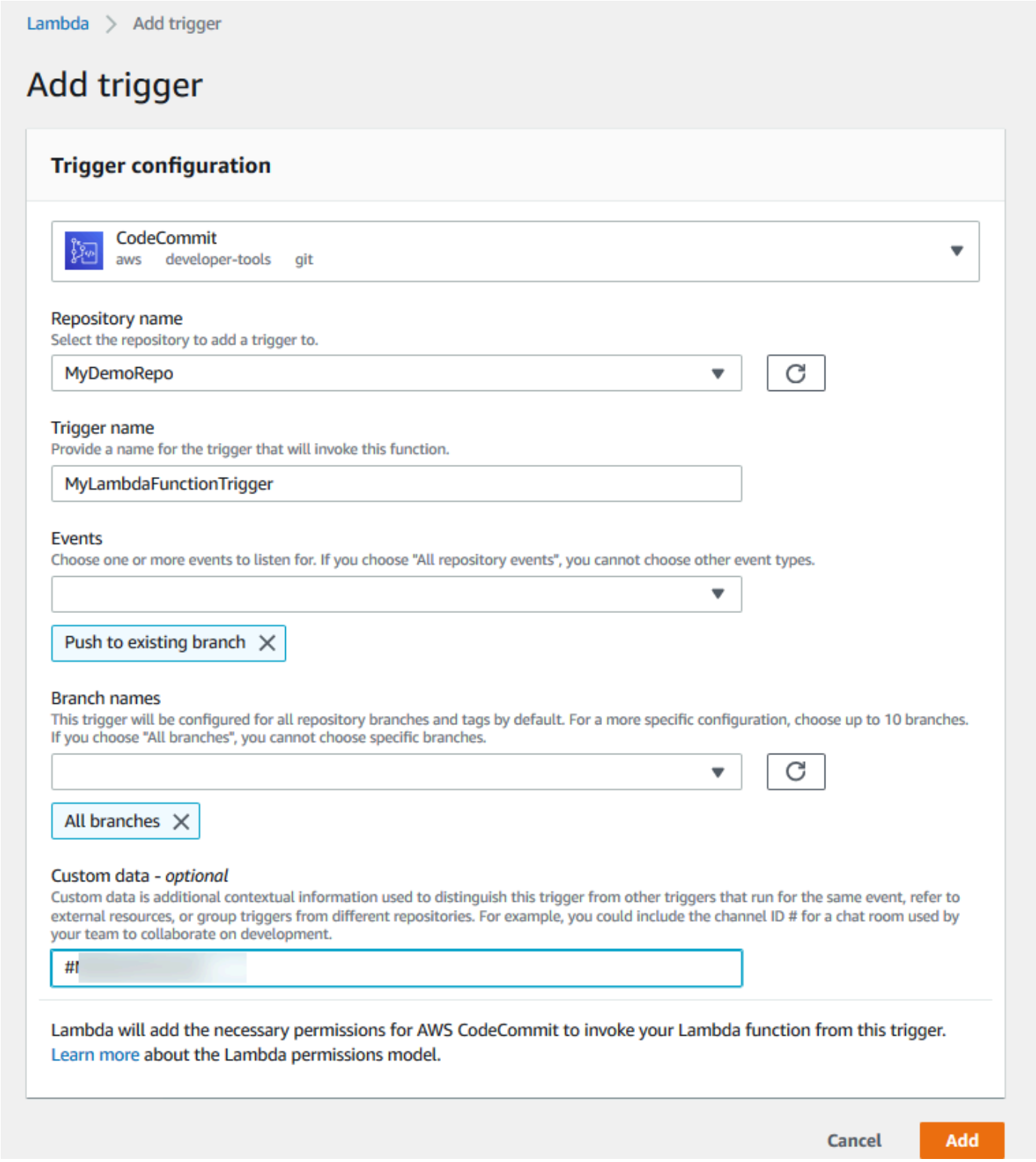
- 建立 Lambda 函式
- 檢視 AWS CodeCommit 儲存庫中 Lambda 函數的觸發程序

建立 Lambda 函式

當您使用 Lambda 主控台建立函數時，也可以為 Lambda 函數建立 CodeCommit觸發器。下列步驟包含範例 Lambda 函數。該示例有兩種語言版本： JavaScript 和 Python。該函數返回用於將存儲庫克隆到 CloudWatch 日誌的 URL。

若要使用 Lambda 藍圖建立 Lambda 函數

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 在 Lambda 函數頁面上，選擇建立函數。如果您之前尚未使用過 Lambda，請選擇立即開始使用。)
3. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。例如，在函數名稱中，提供函數的名稱*MyLambdaFunctionforCodeCommit*。在 Runtime (執行時間) 中，選擇您想要用來寫入函數的語言，然後選擇 Create function (建立函數)。
4. 在 Configuration (組態) 索引標籤上，選擇 Add trigger (新增觸發條件)。
5. 在 [觸發器組態] 中，CodeCommit從 [服務] 下拉式清單選擇。



- 在存放庫名稱中，選擇要在其中設定觸發器的存放庫名稱，該觸發器使用 Lambda 函數回應存放庫事件。

- 在觸發器名稱中，輸入觸發器的名稱 (例如，*MyLambdaFunctionTrigger*)。
- 在事件中，選擇觸發 Lambda 函數的儲存庫事件。如果您選擇 All repository events (所有儲存庫事件)，則無法選擇其他任何事件。如果您想要選擇事件子集，請清除 All repository events (所有儲存庫事件)，然後從清單中選擇您要的事件。例如，如果您希望觸發程序只在使用者在 AWS CodeCommit 存放庫中建立標籤或分支時執行，請移除 [所有存放庫事件]，然後選擇 [建立分支或標籤]。
- 如果您希望將觸發套用至儲存庫的所有分支，請在 Branches (分支) 中選擇 All branches (所有分支)。否則，選擇 Specific branches (特定分支)。依預設會新增儲存庫的預設分支。您可以保留此分支，或從清單中刪除此分支。從儲存庫分支清單中，最多選擇十個分支名稱。
- (選擇性) 在自訂資料中，輸入您想要包含在 Lambda 函數中的資訊 (例如，開發人員用來討論存放庫中開發的 IRC 頻道名稱)。此欄位是字串。無法用於傳遞任何動態參數。

選擇新增。

6. 在 Configuration (組態) 頁面的 Function Code (函數程式碼) 上，在「程式碼項目類型」中選擇「以內嵌方式編輯程式碼」。在 Runtime (執行時間) 中選擇 Node.js。如果您想要建立範例 Python 函數，請選擇 Python。
7. 在 Code entry type (程式碼輸入類型) 中，選擇 Edit code inline (以內嵌方式編輯程式碼)，然後將 hello world 程式碼換成以下兩個範例其中一個。

適用於 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}`,
    );
  }
};
```

適用於 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「權限」索引標籤的「執行角色」中，選擇要在 IAM 主控台中開啟的角色。編輯附加的政策，針對您想要使用觸發條件的存放庫新增 `GetRepository` 許可。

檢視 AWS CodeCommit 儲存庫中 Lambda 函數的觸發程序

建立 Lambda 函數之後，您可以在中檢視和測試觸發器 AWS CodeCommit。測試觸發會執行函數以回應您指定的儲存庫事件。

若要檢視和測試 Lambda 函數的觸發器

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以檢視其中的觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)，然後選擇 Triggers (觸發)。
4. 檢閱儲存庫的觸發清單。您應該會看到您在 Lambda 主控台中建立的觸發器。從清單中選擇它，然後選擇 Test trigger (測試觸發)。此選項會嘗試以有關儲存庫的範例資料來叫用函數，包括儲存庫最新的遞交 ID。(如果不存在遞交歷史記錄，則會產生由零組成的範例值。) 這可協助您確認您已正確設定 AWS CodeCommit 和 Lambda 函數之間的存取。
5. 為了進一步驗證觸發的功能，請進行遞交並推送至您已設定觸發的儲存庫。您應該會在 Lambda 主控台的 [監控] 索引標籤上看到來自 Lambda 函數的回應。從監控索引標籤中，選擇檢視記錄 CloudWatch。主 CloudWatch 控台會在新索引標籤中開啟，並顯示函數的事件。從清單中，根據您推送遞交的時間，選取對應於此時間的日誌串流。您應該會看到類似下列的事件資料：

```
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
```

範例：為現有 AWS Lambda 函數建立中 AWS CodeCommit 的觸發程序

若要建立叫用 Lambda 函數的觸發器，最簡單的方法就是在 Lambda 主控台中建立該觸發器。此內建整合可確保 CodeCommit 具有執行函數所需的權限。若要為現有 Lambda 函數新增觸發器，請前往 Lambda 主控台，然後選擇函數。在函數的 Triggers (觸發) 標籤上，依照 Add trigger (新增觸發) 中的步驟。這些步驟類似於建立 Lambda 函式中的步驟。

您也可以為 CodeCommit 儲存庫中的 Lambda 函數建立觸發器。如此一來，您必須選擇要叫用的現有 Lambda 函數。它還要求您手動配置運行該函數所需 CodeCommit 的權限。

主題

- 手動設定許可以允 CodeCommit 許執行 Lambda 函數
- 在 CodeCommit儲存庫 (主控台) 中為 Lambda 函數建立觸發器
- 為 CodeCommit儲存庫 (AWS CLI) 建立 Lambda 函數的觸發程序

手動設定許可以允 CodeCommit 許執行 Lambda 函數

如果您在中建立一個叫 CodeCommit 用 Lambda 函數的觸發器，則必須手動設定允 CodeCommit 許執行 Lambda 函數的許可。若要避免此手動設定，請考慮改為在 Lambda 主控台中建立函數的觸發器。

若要允 CodeCommit 許執行 Lambda 函數

1. 開啟純文字編輯器並建立 JSON 檔案，以指定 Lambda 函數名稱、存放 CodeCommit 庫的詳細資訊，以及您要在 Lambda 中允許的動作，類似下列內容：

```
{
    "FunctionName": "MyCodeCommitFunction",
    "StatementId": "1",
    "Action": "lambda:InvokeFunction",
    "Principal": "codecommit.amazonaws.com",
    "SourceArn": "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo",
    "SourceAccount": "111122223333"
}
```

2. 將檔案儲存為 JSON 檔案，其名稱很容易記住 (例如 *AllowAccessfromMyDemoRepo*.json)。

3. 使用您剛剛建立的 JSON 檔案，在終端機 (Linux、macOS 或 Unix) 或命令列 (Windows) 執行aws lambda add-permissions命令，將權限新增至與 Lambda 函數相關聯的資源政策：

```
aws lambda add-permission --cli-input-json file://AllowAccessfromMyDemoRepo.json
```

此命令會傳回您剛新增的政策陳述式的 JSON，類似如下：

```
{
    "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\"}"
}
```

如需 Lambda 函數之資源政策的詳細資訊，請參閱AWS Lambda 使用指南中的「提取/推送事件模型」AddPermission和「提取/推送事件

4. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
5. 在 Dashboard (儀表板) 導覽窗格中，選擇 Roles (角色)，然後在角色清單中，選取 *lambda_basic_execution*。
6. 在角色的摘要頁面上，選擇 Permissions (許可) 標籤，然後在 Inline Policies (內嵌政策) 中，選擇 Create Role Policy (建立角色政策)。
7. 在 Set Permissions (設定許可) 頁面上，選擇 Policy Generator (政策產生器)，然後選擇 Select (選取)。
8. 在 Edit Permissions (編輯許可) 頁面上，執行下列動作：

   - 在 Effect (效果) 中，選擇 Allow (允許)。
   - 在 [AWS 服務] 中，選擇AWS CodeCommit。
   - 在動作中，選取GetRepository。
   - 在 Amazon Resource Name (ARN) 中，輸入儲存庫的 ARN (例如，`arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo`)。

   選擇 Add Statement (新增陳述式)，然後選擇 Next Step (下一步)。
9. 在 Review Policy (檢閱政策) 頁面上，選擇 Apply Policy (套用政策)。

您的政策陳述式應該類似於下列範例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt11111111",
            "Effect": "Allow",
            "Action": [
                "codecommit:GetRepository"
            ],
            "Resource": [
                "arn:aws:codecommit:us-east-1:111122223333:MyDemoRepo"
            ]
        }
    ]
}
```

在 CodeCommit儲存庫 (主控台) 中為 Lambda 函數建立觸發器

建立 Lambda 函數之後，您可以在中建立觸發器 CodeCommit ，執行函數以回應您指定的儲存庫事件。

> **Note**
>
> 在您可以成功測試或執行範例的觸發器之前，您必須先設定允許 CodeCommit 叫用函數的政策和 Lambda 函數以取得有關存放庫的資訊。如需詳細資訊，請參閱 若要允 CodeCommit 許執行 Lambda 函數。

若要建立 Lambda 函數的觸發器

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以針對儲存庫事件建立觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)，然後選擇 Triggers (觸發)。
4. 選擇 Create trigger (建立觸發)。
5. 在 Create trigger (建立觸發) 中，執行下列操作：

- 在觸發器名稱中，輸入觸發器的名稱 (例如，*MyLambdaFunctionTrigger*)。
- 在事件中，選擇觸發 Lambda 函數的儲存庫事件。

  如果您選擇 All repository events (所有儲存庫事件)，則無法選擇其他任何事件。如果您想要選擇事件子集，請清除 All repository events (所有儲存庫事件)，然後從清單中選擇您要的事件。例如，如果您希望觸發程序只在使用者在 CodeCommit 存放庫中建立標籤或分支時執行，請移除 [所有存放庫事件]，然後選擇 [建立分支或標籤]。
- 如果您希望將觸發套用到儲存庫的所有分支，請在 Branches (分支) 中保持空白選擇，因為這個預設選項會自動將觸發套用到所有分支。如果您希望此觸發只套用到特定分支，請從儲存庫分支清單中選擇最多 10 個分支名稱。
- 在 Choose the service to use (選擇要使用的服務) 中，選擇 AWS Lambda。
- 在 Lambda 函數中，從清單中選擇函數名稱，或輸入函數的 ARN。
- (選擇性) 在自訂資料中，輸入您想要包含在 Lambda 函數中的資訊 (例如，開發人員用來討論存放庫中開發的 IRC 頻道名稱)。此欄位是字串。無法用於傳遞任何動態參數。

6. (選用) 選擇 Test trigger (測試觸發)。此選項會嘗試以有關儲存庫的範例資料來叫用函數，包括儲存庫最新的遞交 ID。(如果不存在遞交歷史記錄，則會產生由零組成的範例值。) 這可協助您確認您已正確設定 CodeCommit 和 Lambda 函數之間的存取。
7. 選擇 Create trigger (建立觸發)，以完成建立觸發。
8. 為了驗證觸發的功能，請進行遞交並推送至您已設定觸發的儲存庫。您應該會在 Lambda 主控台的 [監控] 索引標籤上看到來自 Lambda 函數的回應。

為 CodeCommit儲存庫 (AWS CLI) 建立 Lambda 函數的觸發程序

您也可以使用命令列建立 Lambda 函數的觸發程序，以回應 CodeCommit 儲存庫事件，例如當有人將提交推送至您的儲存庫時。

若要建立 Lambda 函數的觸發器

1. 開啟純文字編輯器，並建立 JSON 檔案，在其中指定：

   - Lambda 函數名稱。
   - 您要使用此觸發來監控的儲存庫和分支。(如果您不指定任何分支，則觸發會套用到儲存庫中的所有分支。)
   - 啟動此觸發的事件。

儲存檔案。

例如，如果您想要為名為的存放庫建立觸發器，*MyDemoRepo*該存放庫事件將所有儲存庫事件發佈到名*MyCodeCommitFunction*為兩個分支的 Lambda 函數，*main* 和 *prepro* d：

```
{
    "repositoryName": "MyDemoRepo",
    "triggers": [
        {
            "name": "MyLambdaFunctionTrigger",
            "destinationArn": "arn:aws:lambda:us-
east-1:111122223333:function:MyCodeCommitFunction",
            "customData": "",
            "branches": [
                "main", "preprod"
            ],
            "events": [
                "all"
            ]
        }
    ]
}
```

儲存庫的每個觸發在 JSON 中都必須有一個觸發區塊。若要為儲存庫建立多個觸發，請在 JSON 中包含更多區塊。請記住，此檔案中建立的所有觸發是針對指定的儲存庫。您無法在單一 JSON 檔案中為多個儲存庫建立觸發。例如，如果您想要為儲存庫建立兩個觸發，則可以建立含有兩個觸發區塊的 JSON 檔案。在下列範例中，第二個觸發區塊中未指定分支，因此觸發會套用到所有分支：

```
{
    "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"
            ]
        }
    ]
}
```

您可以為您指定的事件建立觸發，例如將遞交推送至儲存庫時。事件類型包括：

- `all` 代表指定的儲存庫和分支中的所有事件。
- `updateReference` 表示遞交推送到指定的儲存庫和分支。
- `createReference` 表示指定的儲存庫中建立新的分支或標籤。
- `deleteReference` 表示指定的儲存庫中刪除分支或標籤。

> **Note**
>
> 您可以在一個觸發中使用多個事件類型。不過，如果您指定 `all`，則無法指定其他事件。

若要查看有效事件類型的完整清單，請在終端機或命令提示字元中輸入 aws codecommit put-repository-triggers help。

此外，您可以在 `customData` 中包含字串 (例如，開發人員在此儲存庫中討論開發時所使用的 IRC 管道名稱)。此欄位是字串。無法用於傳遞任何動態參數。此字串會附加為回應觸發程序而傳回的 CodeCommit JSON 屬性。

2. (選用) 在終端機或命令提示字元中，執行 test-repository-triggers 命令。例如，以下內容用於測試名為觸發器 *.json # JSON ######### CodeCommit #### Lambda ##*。如果沒有實際資料可用，此測試會使用範例資料來觸發該函數。

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

若成功，此命令會傳回類似如下的資訊：

```
{
    "successfulExecutions": [
        "MyLambdaFunctionTrigger"
    ],
    "failedExecutions": []
}
```

3. 在終端機或命令提示字元中，執行put-repository-triggers指令以在中建立觸發器 CodeCommit。例如，若要使用名為 *trigger.json* 的 JSON 檔案來建立觸發：

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

   此命令會傳回類似以下的組態 ID：

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

4. 若要檢視觸發的組態，請執行 get-repository-triggers 命令，並指定儲存庫的名稱：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
   ```

   此命令會傳回為儲存庫設定之所有觸發的結構，類似如下：

   ```
   {
       "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. 為了測試觸發的功能，請進行遞交並推送至您已設定觸發的儲存庫。您應該會在 Lambda 主控台的 [監控] 索引標籤上看到來自 Lambda 函數的回應。

編輯 AWS CodeCommit 儲存庫的觸發器

您可以編輯已針對 CodeCommit 存放庫建立的觸發器。您可以變更觸發的事件和分支、為了回應事件而採取的動作，以及其他設定。

主題

- 編輯存放庫（控制台）的觸發器
- 編輯存放庫的觸發器 (AWS CLI)

編輯存放庫（控制台）的觸發器

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以編輯其中儲存庫事件的觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)，然後選擇 Triggers (觸發)。
4. 從儲存庫的觸發清單中，選擇您要編輯的觸發，然後選擇 Edit (編輯)。
5. 視需要變更觸發，然後選擇 Save (儲存)。

編輯存放庫的觸發器 (AWS CLI)

1. 在終端機 (Linux、macOS 或 Unix) 或命令提示字元 (Windows) 上，執行get-repository-triggers指令以建立 JSON 檔案，其結構為您的存放庫設定的所有觸發程序。例如，若要建立名為 *MyTriggers.* json 的 JSON 檔案，其結構為名*MyDemoRepo*為的儲存庫設定的所有觸發程序：

```
aws codecommit get-repository-triggers --repository-name MyDemoRepo
 >MyTriggers.json
```

此命令不會傳回任何內容，但會在您執行命令的目錄中建立名為 *MyTriggers.json* 的檔案。

2. 在純文字編輯器中編輯 JSON 檔案，並針對您要編輯的觸發變更觸發區塊。將 configurationId 配對換成 repositoryName 配對。儲存檔案。

   例如，如果您想要編輯名為的存放庫*MyFirstTrigger*中名為的觸發程序，*MyDemoRepo*以套用至所有分支，請取代configurationId為repositoryName，並以******移除指定的main和preprod分支。在預設情況下，如果沒有指定分支，觸發將套用到儲存庫中的所有分支：

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

3. 在終端機或命令列上執行 put-repository-triggers 命令。這會更新存放庫的所有觸發程序，包括您對*MyFirstTrigger*觸發器所做的變更：

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

   此命令會傳回類似以下的組態 ID：

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

測試 AWS CodeCommit 存放庫的觸發器

您可以測試已針對 CodeCommit 存放庫建立的觸發程序。測試涉及以儲存庫的範例資料來執行觸發，包括最新的遞交 ID。如果儲存庫沒有遞交歷史記錄，則會產生由零組成的範例值。測試觸發器可協助您確認您是否已正確設定觸發器目標 CodeCommit 與觸發器目標之間的存取，無論是 AWS Lambda 功能還是 Amazon 簡單通知服務通知。

主題

測試存放庫的觸發器（控制台）

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以測試其中儲存庫事件的觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)，然後選擇 Triggers (觸發)。
4. 選擇您要測試的觸發，然後選擇 Test trigger (測試觸發)。您應該會看到成功或失敗訊息。如果成功，您也應該會看到來自 Lambda 函數或 Amazon SNS 主題的對應動作回應。

測試存放庫的觸發器 (AWS CLI)

1. 在終端機 (Linux、macOS 或 Unix) 或命令提示字元 (Windows) 上，執行get-repository-triggers指令以建立 JSON 檔案，其結構為您的存放庫設定的所有觸發程序。例如，若要建立名為 *TestTrigger.* json 的 JSON 檔案，其結構為名 MyDemoRepo為的儲存庫設定的所有觸發程序：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
    >TestTrigger.json
   ```

   這個命令會在您執行命令的目錄中建立名為 *TestTriggers.json* 的檔案。
2. 在純文字編輯器中編輯 JSON 檔案，並變更觸發陳述式。將 `configurationId` 配對換成 `repositoryName` 配對。儲存檔案。

例如，如果您想要測試名為的存放庫*MyFirstTrigger*中名為的觸發程序，configurationId使其套用至所有分支，請取代為，repositoryName然後將類似下列內容的檔案儲存為 *TestTrigger.json*：*MyDemoRepo*

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

3. 在終端機或命令列上執行 test-repository-triggers 命令。這會更新存放庫的所有觸發程序，包括您對*MyFirstTrigger*觸發器所做的變更：

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

此命令會傳回類似以下的回應：

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

從 AWS CodeCommit 儲存庫刪除觸發器

您可能需要刪除已不再使用的觸發。刪除觸發之後就無法復原，但您可以再次建立觸發。

> **Note**
>
> 如果您為儲存庫設定了一或多個觸發器，刪除存放庫並不會刪除您設定為這些觸發程序目標的 Amazon SNS 主題或 Lambda 函數。如果不再需要這些資源，請務必一併刪除。

主題

從儲存庫刪除觸發器 (主控台)

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫以從中刪除儲存庫事件的觸發。
3. 在儲存庫的導覽窗格中，選擇 Settings (設定)。在 Settings (設定) 中，選擇 Triggers (觸發)。
4. 從觸發清單中選擇您要刪除的觸發，然後選擇 Delete (刪除)。
5. 在對話方塊中，輸入 delete (刪除) 以確認。

從儲存庫中刪除觸發器 (AWS CLI)

1. 在終端機 (Linux、macOS 或 Unix) 或命令提示字元 (Windows) 上，執行get-repository-triggers指令以建立 JSON 檔案，其結構為您的存放庫設定的所有觸發程序。例如，若要建立名為 *MyTriggers.* json 的 JSON 檔案，其結構為名 MyDemoRepo為的儲存庫設定的所有觸發程序：

   ```
   aws codecommit get-repository-triggers --repository-name MyDemoRepo
    >MyTriggers.json
   ```

   這個命令會在您執行命令的目錄中建立名為 *MyTriggers.json* 的檔案。
2. 在純文字編輯器中編輯 JSON 檔案，並針對您要刪除的觸發移除觸發區塊。將 `configurationId` 配對換成 `repositoryName` 配對。儲存檔案。

例如，如果您想要*MyFirstTrigger*從名為的儲存庫中移除名為的觸發程式 *MyDemoRepo*，您可以取代configurationId為repositoryName，然後移除######的陳述式：

```
{
    "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. 在終端機或命令列上執行 put-repository-triggers 命令。這會更新存放庫的觸發程序並刪除*MyFirstTrigger*觸發程序：

```
aws codecommit put-repository-triggers --repository-name MyDemoRepo
 file://MyTriggers.json
```

此命令會傳回類似以下的組態 ID：

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

```
}
```

> **Note**
>
> 要刪除名為的存儲庫的所有觸發器 *MyDemoRepo*，您的 JSON 文件看起來像這樣：
>
> ```
> {
>     "repositoryName": "MyDemoRepo",
>     "triggers": []
> }
> ```

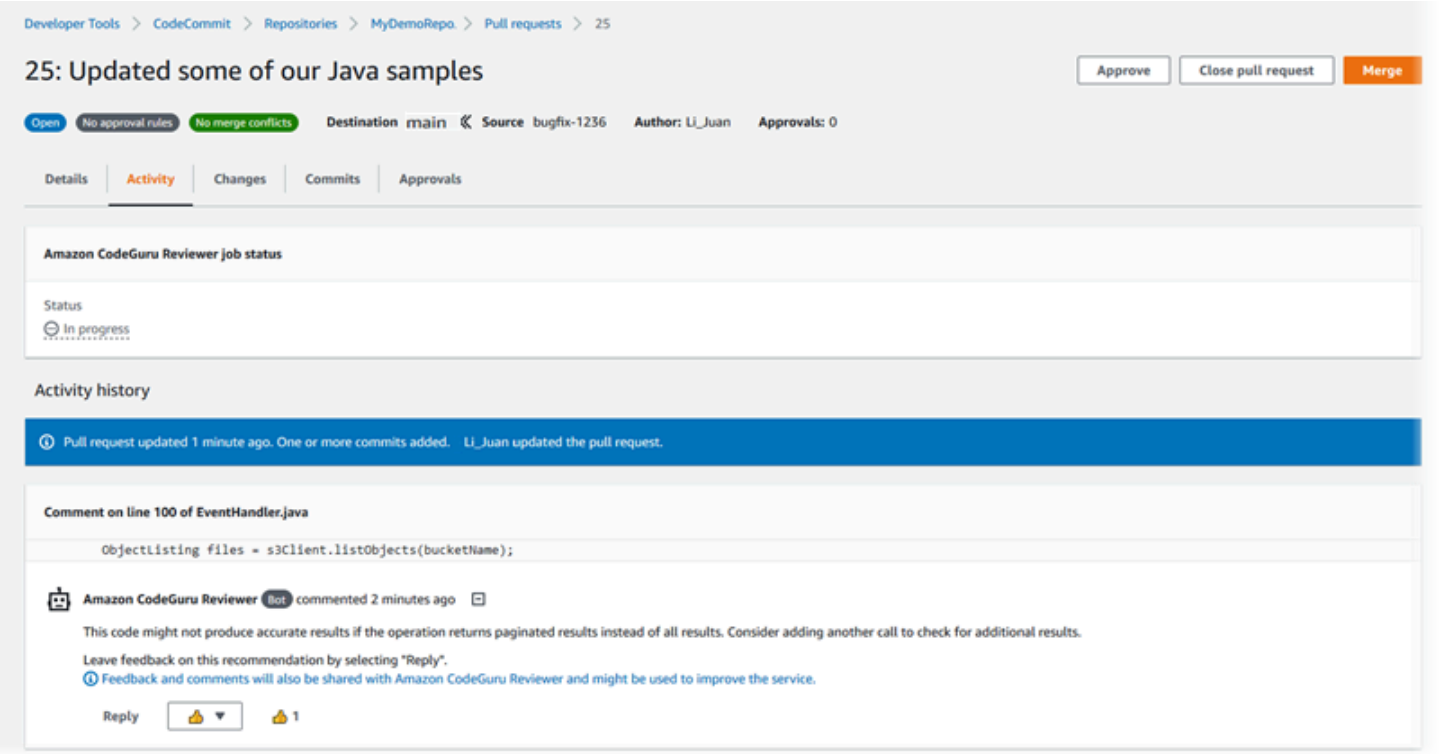
將 AWS CodeCommit 儲存庫與 Amazon CodeGuru 審核者建立關聯或取消關聯

Amazon CodeGuru Reviewer 是一種自動化程式碼檢閱服務，它使用程式分析和機器學習來偵測常見問題，並建議 Java 或 Python 程式碼中的修正程式。您可以將 Amazon Web Services 帳戶中的儲存庫與 CodeGuru 審核者建立關聯。當您這麼做時， CodeGuru Reviewer 會建立服務連結角色，讓 CodeGuru Reviewer 分析建立關聯後所建立之所有提取要求中的程式碼。

建立儲存庫的關聯後， CodeGuru 審核者會分析並針對您建立提取請求時發現的任何問題進行註解。每條評論都清楚地標記為來自評論 CodeGuru 者，並指定 Amazon CodeGuru 審閱者。您可以回覆這些評論，就像回覆提取請求中的任何其他評論一樣，也可以對建議品質提出意見回饋。此意見反應會與 CodeGuru 審核者分享，有助於改善服務及其建議。

> **Note**
>
> 在與儲存庫建立關聯之前所建立的提取要求中，您將不會看到 CodeGuru Reviewer 的註解。由於下列原因，您可能無法在關聯之後建立的提取請求中看到評論：
>
> - 提取請求不包含 Java 或 Python 代碼。
> - CodeGuru 審核者沒有足夠的時間執行和檢閱提取要求中的程式碼。此程序最多需要 30 分鐘的時間。註解可以隨著檢閱進度顯示，但在工作狀態顯示為「已完成」之前，註解才會完成。
> - CodeGuru 審閱者在提取請求中沒有發現 Java 或 Python 代碼中的任何問題。
> - 無法執行程式碼檢閱任務。若要檢閱提取請求的審核狀態，請參閱提取請求的「活動」標籤。

- 您正在「變更」索引標籤中檢視提取請求的變更、提取請求已更新，且 Amazon CodeGuru Reviewer 在變更中未發現任何問題。只有在提取請求的最 CodeGuru 新修訂版本中提出註解時，Amazon 審核者註解才會顯示在「變更」索引標籤中。它們始終顯示在「活動」標簽中。



如需詳細資訊，請參閱在中使用提取請求AWS CodeCommit儲存庫檢閱提取請求、和 Amazon CodeGuru 審核者使用者指南。

> **Note**
>
> 您必須使用具有足夠權限的 IAM 使用者或角色登入，才能將儲存庫與 CodeGuru 審核者建立關聯或取消關聯。如需包含這些權限之受管理原則 CodeCommit 的相關資訊，請參閱 CodeCommit 的 AWS 受管政策和AWS CodeCommit受管政策和 Amazon CodeGuru 審閱者。如需 CodeGuru 審核者許可和安全性的相關資訊，請參閱 Amazon CodeGuru 審核者使用者指南。

主題

- 建立儲存庫與 CodeGuru 複查者的關聯

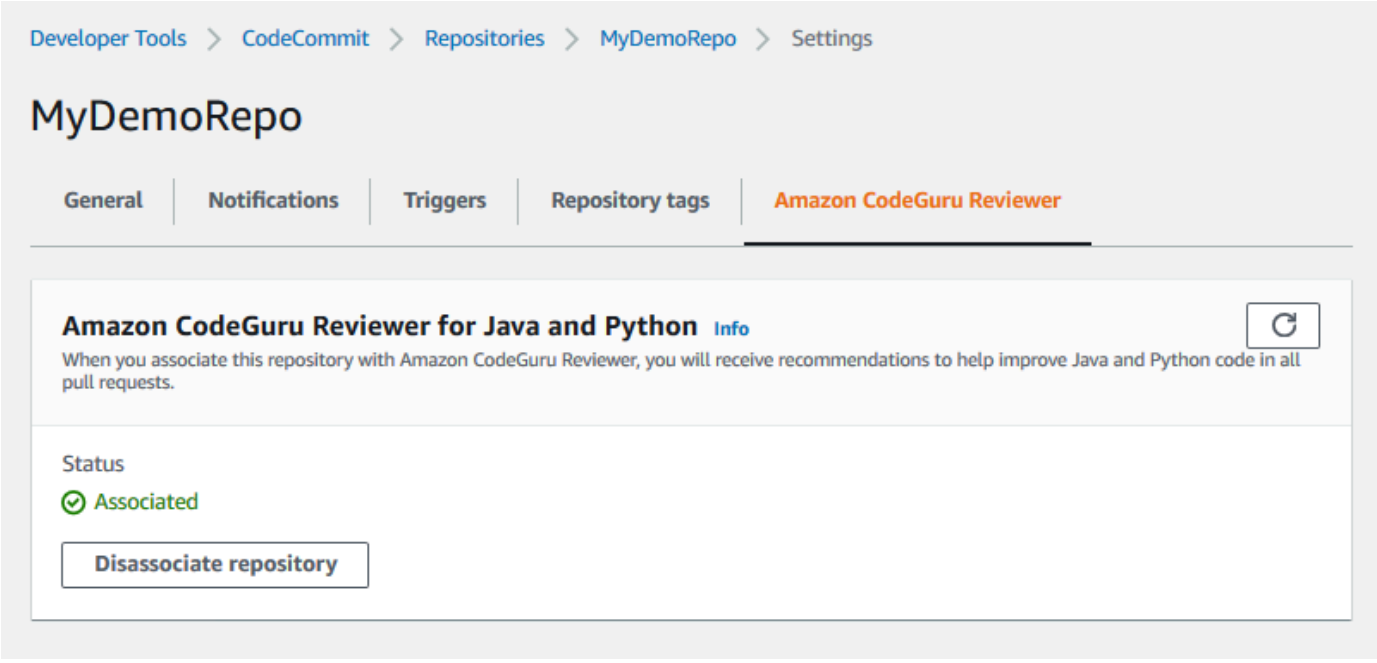
- 取消儲存庫與審核者的關聯 CodeGuru

建立儲存庫與 CodeGuru 複查者的關聯

使用主 AWS CodeCommit 控台快速將儲存庫與 CodeGuru 複查者建立關聯。如需其他方法，請參閱 Amazon CodeGuru 審核者使用者指南。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在儲存區域中，選擇要與 CodeGuru 複查者產生關聯的儲存庫名稱。
3. 選擇 [設定]，然後選擇 [Amazon CodeGuru 審核者]。
4. 選擇 Associate repository (建立儲存庫的關聯)。

> **Note**
>
> 將儲存庫與 CodeGuru 複查者完全關聯可能需要 10 分鐘的時間。不會自動更新狀態。若要檢視目前狀態，請選擇重新整理按鈕。



取消儲存庫與審核者的關聯 CodeGuru

使用主 AWS CodeCommit 控台可快速取消儲存庫與 CodeGuru 審核者的關聯。如需其他方法，請參閱 Amazon CodeGuru 審核者使用者指南。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在儲存庫中，選擇您要取消與 CodeGuru 審核者關聯的儲存庫名稱。
3. 選擇 [設定]，然後選擇 [Amazon CodeGuru 審核者]。
4. 選擇 Disassociate repository (取消儲存庫的關聯)。

檢視 CodeCommit 儲存庫詳細

您可以使用連線至儲存庫之本機存放庫的 AWS CodeCommit 主控台 AWS CLI、或 CodeCommit Git 來檢視有關可用儲存庫的資訊。

依照以下指示之前，請完成設定 中的步驟。

主題

- 檢視儲存庫詳細資訊 (主控台
- 檢視 CodeCommit 儲存庫詳細資料 (Git)
- 檢視 CodeCommit 儲存庫詳細資訊 (AWS CLI)

檢視儲存庫詳細資訊 (主控台

使用主 AWS CodeCommit 控台快速檢視使用 Amazon Web Services 帳戶建立的所有儲存庫。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在存放庫中，檢視您登入之 AWS 區域 儲存庫的詳細資訊。使用「地區」選擇器來選擇不同的 AWS 區域 t 來檢視該區域中的儲存庫。
3. 選擇您要檢視其更多詳細資訊的重新故事名稱，然後執行下列其中一個動作：

   - 若要檢視用於複製儲存庫的 URL，請選擇 Clone URL (複製 URL)，然後選擇複製儲存庫時想要使用的通訊協定。這會將複製 URL 複製。若要檢閱它，請將它貼到純文字編輯器中。

- 若要檢視存放庫的可配置選項以及詳細資訊 (例如存放庫 ARN 和存放庫 ID)，請在導覽窗格中選擇 [設定]。

> **Note**
>
> 如果您以 IAM 使用者身分登入，則可以設定並儲存喜好設定，以便檢視程式碼和其他主控台設定。如需詳細資訊，請參閱 使用者偏好設定。

檢視 CodeCommit 儲存庫詳細資料 (Git)

若要使用本機存放庫的 Git 來檢視有關 CodeCommit 儲存庫的詳細資料，請執行git remote show指令。

在執行這些步驟之前，請先將本機存放庫連線到 CodeCommit 儲存庫。如需說明，請參閱連接到儲存庫。

1. 執行指git remote show ***remote-name***令，其中####是 CodeCommit 儲存庫的別名 (預設為)。`origin`

   > **Tip**
   >
   > 若要取得 CodeCommit 儲存庫名稱及其 URL 的清單，請執行git remote -v指令。

   例如，若要使用別名檢視有關 CodeCommit 存放庫的詳細資訊`origin`：

   ```
   git remote show origin
   ```

2. 針對 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)
```

針對 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**
>
> 若要查詢 IAM 使用者的 SSH 金鑰 ID，請開啟 IAM 主控台，然後展開 IAM 使用者詳細資料頁面上的安全登入資料。您可以在的 SSH 金鑰中找到安全殼層金鑰識別碼 AWS CodeCommit。

如需更多選項，請參閱 Git 文件。

檢視 CodeCommit 儲存庫詳細資訊 (AWS CLI)

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用檢 AWS CLI 視儲存庫詳細資訊，請執行下列命令：

- 若要檢視 CodeCommit 儲存庫名稱及其對應 ID 的清單，請執行清單儲存庫。
- 若要檢視單一 CodeCommit 存放庫的相關資訊，請執行取得存放庫。
- 若要檢視中多個儲存庫的相關資訊 CodeCommit，請執行batch-get-repositories。

檢視 CodeCommit 儲存庫的清單

1. 執行 list-repositories 命令：

```
aws codecommit list-repositories
```

   您可以使用選用的 `--sort-by` 或 `--order` 選項來變更傳回資訊的順序。

2. 如果成功，此命令將輸出一個`repositories`物件，其中包含與 Amazon Web Services 帳戶 CodeCommit 相關聯的所有儲存庫的名稱和 ID。

   以下是基於上述命令的一些範例輸出：

```
{
    "repositories": [
        {
           "repositoryName": "MyDemoRepo",
           "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
        },
        {
           "repositoryName": "MyOtherDemoRepo",
           "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
        }
    ]
}
```

若要檢視單一 CodeCommit 儲存庫的詳細資訊

1. 運行get-repository命令，使用`--repository-name`選項指定 CodeCommit 存儲庫的名稱。

> **Tip**
>
> 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

   例如，若要檢視名為的 CodeCommit 儲存庫的詳細資訊MyDemoRepo：

```
aws codecommit get-repository --repository-name MyDemoRepo
```

2. 如果成功，此命令會輸出 `repositoryMetadata` 物件，以及下列資訊：

- 儲存庫的名稱 (repositoryName)。
- 儲存庫的描述 (repositoryDescription)。
- 儲存庫的唯一、系統產生的 ID (repositoryId)。
- 與儲存庫相關聯之 Amazon Web Services 帳戶的識別碼 (accountId)。

以下是基於上述範例命令的一些範例輸出：

```
{
        "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"
        }
}
```

若要檢視多個 CodeCommit 儲存庫的詳細資訊。

1. 使用 batch-get-repositories 選項執行 --repository-names 命令。在每個 CodeCommit 儲存庫名稱之間加上空格。

> **Tip**
>
> 要獲取存儲庫的名稱 CodeCommit，請運行列表庫命令。

例如，若要檢視名MyDemoRepo為 and 的兩個 CodeCommit 儲存庫的詳細資料MyOtherDemoRepo：

```
aws codecommit batch-get-repositories --repository-names MyDemoRepo MyOtherDemoRepo
```

2. 如果成功，此命令會輸出物件，以及下列資訊：

   - 找不到的任何 CodeCommit 儲存庫清單 (repositoriesNotFound)。
   - CodeCommit 儲存庫清單 (repositories)。每個 CodeCommit 存儲庫名稱後跟：
     - 儲存庫的描述 (repositoryDescription)。
     - 儲存庫的唯一、系統產生的 ID (repositoryId)。
     - 與儲存庫相關聯之 Amazon Web Services 帳戶的識別碼 (accountId)。

   以下是基於上述範例命令的一些範例輸出：

```
{
        "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": []
    }
```

變更 AWS CodeCommit 儲存庫設定

您可以使用 AWS CLI 和 AWS CodeCommit 控制台來變更 CodeCommit 存放庫的設定，例如其描述或名稱。

> **Important**
>
> 變更儲存庫的名稱可能會破壞在其遠端 URL 中使用舊名稱的任何本機儲存庫。執行 git remote set-url 命令來將遠端 URL 更新為使用新儲存庫的名稱。

主題

變更儲存庫設定 (主控台)

若要使用主 AWS CodeCommit 控台變更中的 CodeCommit 存放庫設定 AWS CodeCommit，請依照下列步驟執行。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要變更設定所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。
4. 若要變更儲存庫的名稱，在 Repository name (儲存庫名稱) 中，於 Name (名稱) 文字方塊輸入新的名稱，然後選擇 Save (儲存)。系統提示時，請確認您的選擇。

> **Important**
>
> 變更 AWS CodeCommit 存放庫的名稱會變更使用者需要連線到存放庫的 SSH 和 HTTPS URL。使用者將無法連接到此儲存庫，直到他們更新連線設定為止。此外，由於存放庫的 ARN 會變更，因此變更儲存庫名稱會使任何依賴此儲存庫 ARN 的 IAM 使用者政策失效。若要在變更名稱之後連接到儲存庫，每位使用者必須使用 git remote set-url 命令，並指定要使用的新 URL。例如，如果您將儲存庫的名稱從變更 MyDemoRepo 為 MyRenamedDemoRepo，則使用 HTTPS 連線至儲存庫的使用者會執行下列 Git 命令：
>
> ```
> git remote set-url origin https://git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyRenamedDemoRepo
> ```
>
> 使用 SSH 連接到儲存庫的使用者會執行以下 Git 命令：
>
> ```
> git remote set-url origin ssh://git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyRenamedDemoRepo
> ```
>
> 如需更多選項，請參閱 Git 文件。

5. 若要變更儲存庫的描述，請修改 Description (描述) 文字方塊中的文字，然後選擇 Save (儲存)。

> **Note**
>
> 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的應用程式開發人員，且計劃在網頁瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

6. 若要變更預設分支，請在 Default branch (預設分支) 中選擇分支下拉式清單，然後選擇不同的分支。選擇儲存。
7. 若要變更用於 AWS KMS 加密和解密儲存庫中資料的加密金鑰，請在儲存庫加密金鑰中選擇AWS 受管金鑰或 Custo mer Managed 金鑰來指定要使用的金鑰類型。如果選擇客戶管理的金鑰，請輸入金鑰的 ARN。選擇儲存。
8. 若要刪除儲存庫，請選擇 Delete repository (刪除儲存庫)。在 Type the name of the repository to confirm deletion (輸入儲存庫名稱以確認刪除) 旁的方塊中，輸入 **`delete`**，然後選擇 Delete (刪除)。

> **Important**
>
> 在中刪除此儲存庫之後 AWS CodeCommit，您將無法再將其複製到任何本機存放庫或共用存放庫。您也將不再能夠從任何本地回購或共享回購中提取數據或將數據推送到它。這個操作無法復原。

變更 AWS CodeCommit 儲存庫設定 (AWS CLI)

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若 AWS CLI 要使用在中變更 CodeCommit 存放庫的設定 AWS CodeCommit，請執行下列一或多個指令：

- update-repository-description以變更儲 CodeCommit 存庫的描述。
- update-repository-name以變更儲 CodeCommit 存庫的名稱。

若要變更儲 CodeCommit 存庫的描述

1. 執行 update-repository-description 命令，並指定：
   - CodeCommit 存放庫的名稱 (含選`--repository-name`項)。

     > **Tip**
     >
     > 若要取得 CodeCommit 儲存庫的名稱，請執行list-repositories指令。

   - 新儲存庫的描述 (使用 `--repository-description` 選項)。

     > **Note**
     >
     > 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的應用程式開發人員，且計劃在網頁瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

例如，若要將名為的 CodeCommit 存放庫的描述變更MyDemoRepo為This description was changed：

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

只有在發生錯誤時，此命令才會產生輸出。

2. 若要驗證變更的描述，請執行指get-repository令，指定您使用--repository-name選項變更其描述的 CodeCommit 儲存庫名稱。

   命令的輸出會在 repositoryDescription 中顯示變更的文字。

若要變更儲 CodeCommit 存庫的名稱

1. 執行 update-repository-name 命令，並指定：

   - CodeCommit 儲存庫的目前名稱 (含選--old-name項)。

     > **Tip**
     >
     > 要獲取 CodeCommit 存儲庫的名稱，請運行列表庫命令。

   - CodeCommit 存放庫的新名稱 (使用選--new-name項)。

   例如，若要將名為 MyDemoRepo 儲存庫的名稱變更為 MyRenamedDemoRepo：

   ```
   aws codecommit update-repository-name --old-name MyDemoRepo --new-name
    MyRenamedDemoRepo
   ```

   只有在發生錯誤時，此命令才會產生輸出。

   > **Important**
   >
   > 變更 AWS CodeCommit 存放庫的名稱會變更使用者需要連線到存放庫的 SSH 和 HTTPS URL。使用者需要更新其連線設定，才能連接到此儲存庫。此外，由於存放庫的 ARN 發生變更，因此變更存放庫名稱會使任何依賴此儲存庫 ARN 的 IAM 使用者政策無效。

2. 若要驗證變更的名稱，請執行 list-repositories 命令並檢閱儲存庫名稱的清單。

同步本地存儲庫和 AWS CodeCommit 存儲庫之間的更改

您可以使用 Git 同步本地存儲庫和連接到本地 CodeCommit 存儲庫的存儲庫之間的更改。

要將更改從本地存儲庫推送到 CodeCommit 存儲庫，請運行git push ***remote-name branch-name***。

要從存儲庫中提取對本地 CodeCommit 存儲庫的更改，請運行git pull ***remote-name branch-name***。

對於推送和拉動，*####*是本地回購用於存儲庫的 CodeCommit 暱稱。 分*###*是 CodeCommit 存儲庫上要推送或從中提取的分支的名稱。

> **Tip**
>
> 若要取得本機軟體庫用於 CodeCommit 儲存庫的暱稱，請執行git remote。若要取得分支名稱的清單，請執行 git branch。星號 (*) 會顯示在目前分支的名稱旁。(您也可以執行 git status 以顯示目前的分支名稱。)

> **Note**
>
> 如果您從本地存儲庫的角度克隆了存儲庫，則*####*不是存儲庫的名稱。 CodeCommit 複製儲存庫時，*remote-name* 會自動設定為 `origin`。

例如，要將更改從本地回購推送到帶有暱稱的 CodeCommit 存儲庫中的`main`分支`origin`：

```
git push origin main
```

同樣，要使用暱稱從 CodeCommit 存儲庫中的`main`分支中提取本地回購的更改`origin`：

```
git pull origin main
```

> **Tip**
>
> 如果您將 `-u` 選項新增至 git push，您將設定上游追蹤資訊。例如，如果您執行 git push -u origin main，您可以在未來執行 git push 和 git pull 而不帶 *remote-name branch-name*。

若要取得上游追蹤資訊，請執行 git remote show ***remote-name*** (例如，git remote show origin)。

如需更多選項，請參閱 Git 文件。

將提交推送到額外的 Git 存儲庫

您可以設定本機儲存庫將變更推送到兩個遠端儲存庫。例如，您在試用 AWS CodeCommit時可能需要繼續使用現有的 Git 儲存庫解決方案。請依照下列基本步驟，將本機存放庫中的變更推送至 CodeCommit 個別的 Git 儲存庫。

> **Tip**
>
> 如果您沒有 Git 儲存庫，您可以在其他服務上建立一個空的儲存庫， CodeCommit然後再將 CodeCommit 儲存庫遷移到該儲存庫。您應該遵循類似於遷移到 CodeCommit中的步驟。

1. 從命令提示字元或終端機，切換到您的本機儲存庫目錄並執行 git remote -v 命令。您應該會看到類似下列的輸出：

   針對 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)
   ```

   針對 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. 運行git remote set-url --add --push origin ***git-repository-name***命令，其中*git-repository-name*是您要託管代碼的 Git 存儲庫的 URL 和名稱。這會將 origin 的推送目的地變更為該 Git 儲存庫。

> **Note**
>
> git remote set-url --add --push 會覆寫推送的預設 URL，所以您必須執行此命令兩次，如後續步驟所示。

例如，以下命令將原點推送更改為*## URL/*：MyDestinationRepo

```
git remote set-url --add --push origin some-URL/MyDestinationRepo
```

此命令不會傳回任何結果。

> **Tip**
>
> 如果您推送到需要登入資料的 Git 儲存庫，請務必在登入資料協助程式或 *some-URL* 字串的組態中設定這些登入資料。否則，對該儲存庫的推送會失敗。

3. 再次執行 git remote -v 命令，應該會建立類似如下的輸出：

   針對 HTTPS：

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

   針對 SSH：

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

4. 現在添加存 CodeCommit 儲庫。git remote set-url --add --push origin再次運行，這次使用存儲庫的 URL 和存儲 CodeCommit庫名稱。

   例如，下列指令會將原點推送新增至 https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo：

   針對 HTTPS：

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

針對 SSH：

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

此命令不會傳回任何結果。

5. 再次執行 git remote -v 命令，應該會建立類似如下的輸出：

針對 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)
```

針對 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)
```

您現在有兩個 Git 存儲庫作為推送的目標，但是您的推送首先轉到*## URL/*。MyDestinationRepo 如果推送至該儲存庫失敗，則您的遞交不會推送至任一儲存庫。

> **Tip**
>
> 如果其他儲存區域需要您想要手動輸入的認證，請考慮變更推送的順序，以便您 CodeCommit 先推送至。執行 git remote set-url --delete 來刪除先推送到的儲存庫，然後執行 git remote set-url --add 以再次新增此儲存庫，使之成為清單中的第二個推送目的地。
> 如需更多選項，請參閱 Git 文件。

6. 為了驗證您現在同時推送到這兩個遠端儲存庫，請使用文字編輯器，在本機儲存庫中建立以下文字檔案：

```
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. 執行 git add 將變更暫存在本機儲存庫：

```
git add bees.txt
```

8. 執行 git commit 來遞交本機儲存庫中的變更：

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

9. 若要將遞交從本機儲存庫推送到遠端儲存庫，請執行 git push -u ***remote-name branch-name***，其中 *remote-name* 是本機儲存庫用於遠端儲存庫的別名，*branch-name* 是要推送至儲存庫的分支名稱。

> **Tip**
>
> 只有在第一次推送時才需要使用 -u 選項。即會設定上游追蹤資訊。

例如，執行 git push -u origin main 會顯示推送同時前往兩個遠端儲存庫的預期分支，且輸出類似如下：

針對 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
```

針對 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
```

如需更多選項，請參閱 Git 文件。

使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取

您可以在另一個帳戶中為 IAM 使用者和群組設定 CodeCommit 存放庫的存取 AWS 權。這通常稱為跨帳戶存取。本節提供範例和 step-by-step 指示，將帳戶 (稱為 AccountA) *MySharedDemoRepo*中美國東部 (俄亥俄) 區域命名的儲存庫設定跨 AWS 帳戶存取權限給屬於另一個 AWS 帳戶指定之 IAM 群組的 IAM 使用者 (稱為 AccountB)。*DevelopersWithCrossAccountRepositoryAccess*

本節分成三個部分：

- AccountA 的管理員所執行的動作。
- AccountB 的管理員所執行的動作。
- AccountB 的儲存庫使用者所執行的動作。

設定跨帳戶存取：

- AccountA 中的管理員以 IAM 使用者身分登入，並具有在 IAM 中建立和管理儲存庫 CodeCommit 和建立角色所需的許可。如果您使用受管政策，請將 IAM FullAccess 套用 AWSCodeCommitFullAccess 至此 IAM 使用者。

  AccountA A 的帳戶識別碼範例為 *111122223333*。
- AccountB 中的管理員以 IAM 使用者身分登入，具有建立和管理 IAM 使用者和群組所需的許可，以及為使用者和群組設定政策。如果您使用受管政策，請FullAccess 將 IAM 套用至此 IAM 使用者。

  帳戶 AccountB 戶識別碼範例為 *8888888888*。
- AccountB 中的儲存庫使用者若要模擬開發人員的活動，請以 IAM 使用者身分登入，該使用者是為了允許存取 AccountA 中的 CodeCommit 存放庫而建立的 IAM 群組成員。這個帳戶必須設定：
  - AWS 管理主控台存取。
  - 連線至 AWS 資源時所使用的存取金鑰和秘密金鑰，以及存取 AccountA 中存取儲存庫時要承擔的角色 ARN。
  - 複製儲存庫所在位置的本機電腦上的 git-remote-codecommit 公用程式。此公用程式需要 Python 及其安裝程式 pip。您可從 Python Package Index 網站上的 git-remote-codecommit 下載此公用程式。

  如需詳細資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit和 IAM 使用者。

主題

跨帳戶儲存庫存取權：AccountA A 中管理員的動作

若要允許 AccountB 的使用者或群組存取 AccountA 中的儲存庫，AccountA 管理員必須：

- 在 AccountA 建立政策來授予對儲存庫的存取權。
- 在帳戶 TA 中建立可由帳戶 TB 中的 IAM 使用者和群組擔任的角色。
- 將政策連接到角色。

以下章節提供步驟和範例。

主題

步驟 1：在 AccountA A 中建立儲存庫存取權的原則

您可以在 AccountA 建立政策，來授予對 AccountB 中儲存庫的存取權。根據您想允許的存取層級而定，請執行以下其中一項：

- 設定政策，以允許 AccountB 使用者存取特定的儲存庫，但不允許他們檢視 AccountA 中所有儲存庫的清單。
- 設定額外的存取權限，以允許 AccountB 使用者從 AccountA 中所有儲存庫的清單中選擇儲存庫。

建立儲存庫存取政策

1. 以具有在 AccountA 中建立政策的權限的 IAM 使用者身分登入 AWS 管理主控台。
2. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
3. 在導覽窗格中，選擇政策。
4. 選擇 Create policy (建立政策)。
5. 選擇 JSON 標籤，將下列 JSON 政策文件貼到 JSON 文字方塊中。將 *us-east-2* 取代為儲存庫，*111122223333 AWS ##* 為 AccountA A 的 AccountA 識別碼，並*MySharedDemoRepo*使用帳號 A 中儲存庫的名稱取代： 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"
        ]
    }
]
}
```

如果您希望擔任此角色的使用者能夠檢視 CodeCommit 主控台首頁上的儲存區域清單，請將其他陳述式新增至原則，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": "*"
```

```
        }
    ]
}
```

擔任此角色的使用者可以使用這個存取權，更輕鬆地運用此政策來找出其可存取的儲存庫。他們可以從清單中選擇儲存庫名稱，就會自動前往共用儲存庫的首頁 (Code)。使用者無法存取他們在清單中看到的其他任何儲存庫，但可以在 Dashboard (儀表板) 頁面上檢視 AccountA 中的儲存庫。

如果您不想讓擔任該角色的使用者能夠檢視 AccountA 中所有儲存庫的清單，請使用第一個策略範例，但請務必將直接連結傳送至 CodeCommit 主控台中共用存放庫首頁的使用者。

6. 選擇檢閱政策。政策驗證器會報告語法錯誤 (例如，如果您忘記將範例 Amazon Web Services 帳戶 ID 和儲存庫名稱取代為您的 Amazon Web Services 帳戶 ID 和儲存庫名稱)。
7. 在 [檢閱策略] 頁面上，輸入策略的名稱 (例如，*CrossAccountAccessForMySharedDemoRepo*)。您也可以提供此政策的選用描述。選擇建立政策。

步驟 2：在 AccountA ount 中建立存放庫存取權的角色

設定政策後，請建立 AccountB 中的 IAM 使用者和群組可以承擔的角色，並將該政策附加到該角色。

建立儲存庫存取角色

1. 在 IAM 主控台，選擇 Roles (角色)。
2. 選擇建立角色。
3. 選擇另一個 Amazon Web Services 帳戶。
4. 在帳戶識別碼中，輸入帳戶 AccountB 的 Amazon Web Services 帳戶識別碼 (例如，*8888888888*)。選擇下一步：許可。
5. 在 [附加權限原則] 中，選取您在上一個程序 (*CrossAccountAccessForMySharedDemoRepo*) 中建立的原則。選擇下一步：檢閱。
6. 在角色名稱中，輸入角色的名稱 (例如，*MyCrossAccountRepositoryContributorRole*)。您也可以輸入選用描述，協助其他人了解該角色的用途。
7. 選擇建立角色。
8. 開啟您剛建立的角色，並複製角色 ARN (例如，arn:aws:iam::*111122223333*:role/*MyCrossAccountRepositoryContributorRole*)。您需要將此 ARN 提供給 AccountB 管理員。

跨帳戶存放庫存取：帳號中管理員的動作 TB

若要允許 AccountB 的使用者或群組存取 AccountA 中的儲存庫，AccountB 管理員必須在 AccountB 中建立群組。此群組必須設定政策，以允許群組成員擔任由 AccountA 管理員所建立的角色。

以下章節提供步驟和範例。

主題

步驟 1：為 AccountB 使用者建立存放庫存取權的 IAM 群組

管理 AccountB 中哪些 IAM 使用者可以存取帳戶 TA 儲存庫的最簡單方法是在 Account AccountB 中建立一個 IAM 群組，該群組具有在 AccountA 中擔任該角色的權限，然後將 IAM 使用者新增至該群組。

建立儲存庫跨帳戶存取的群組

1. 以 IAM 使用者身分登入 AWS 管理主控台，並具備在 AccountB 中建立 IAM 群組和政策所需的許可，以及管理 IAM 使用者。
2. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
3. 在 IAM 主控台中，選擇 [群組]。
4. 選擇 Create New Group (建立新群組)。
5. 在群組名稱中，輸入群組的名稱 (例如，*DevelopersWithCrossAccountRepositoryAccess*)。選擇 Next Step (後續步驟)。
6. 在 Attach Policy (連接政策) 中選擇 Next Step (下一步)。在下一個程序中，您將建立跨帳戶政策。完成群組的建立。

步驟 2：建立政策並將使用者新增至 IAM 群組

現在您有一個群組，請建立政策，以允許此群組的成員擔任可讓他們在 AccountA 中存取儲存庫的角色。然後，將您要允許在帳戶 AccountA 中存取的 AccountB 中的 IAM 使用者新增至群組。

建立群組的政策並將使用者新增到群組

1. 在 IAM 主控台中，選擇 [群組]，然後選擇剛建立的群組名稱 (例如 *DevelopersWithCrossAccountRepositoryAccess*)。

2. 選擇許可索引標籤標籤。展開 Inline Policies (內嵌政策)，然後選擇連結來建立內嵌政策。(如果您要設定的群組已有內嵌政策，請選擇 Create Group Policy (建立群組政策)。)
3. 選擇 Custom Policy (自訂政策)，然後選擇 Select (選取)。
4. 在策略名稱中，輸入策略的名稱 (例如，*AccessPolicyForSharedRepository*)。
5. 在 Policy Document (政策文件) 中，貼上以下政策。在中**Resource**，將 ARN 取代為管理員在 AccountA 中建立的原則的 ARN (例如，arn: aw: iam:: ***1111223*** 33: 角色/)，然後選擇 [套用原則]。***MyCrossAccountRepositoryContributorRole***如需 AccountA 管理員所建立政策的詳細資訊，請參閱步驟 1：在 AccountA A 中建立儲存庫存取權的原則。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":
 "arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole"
  }
}
```

6. 選擇 Users (使用者) 索引標籤。選擇 [將使用者新增至群組]，然後新增 AccountB IAM 使用者。例如，您可以將具有使用者名稱 *Saanvi_Sar* kar 的 IAM 使用者新增至群組。

> **Note**
>
> AccountB 中的使用者必須具有程式設計方式存取權 (包括存取金鑰和秘密金鑰)，才能設定其本機電腦以存取共 CodeCommit 用存放庫。如果您要建立 IAM 使用者，請務必儲存存取金鑰和秘密金鑰。為確保 AWS 帳戶的安全性，只有在建立金鑰時才能存取私密存取金鑰。

跨帳戶存放庫存取：帳號中儲存庫使用者的動作 TB

為了存取 AccountA 中的儲存庫，AccountB 群組的使用者必須設定其本機電腦來存取儲存庫。以下章節提供步驟和範例。

主題

- 步驟 1：為 AccountB 使用者設定 AWS CLI 和 Git，以存取 AccountA A 中的儲存庫
- 步驟 2：克隆並訪問 AccountA 中的 CodeCommit 存儲庫

步驟 1：為 AccountB 使用者設定 AWS CLI 和 Git，以存取 AccountA A 中的儲存庫

您無法使用 SSH 金鑰或 Git 登入資料存取其他 Amazon Web Services 帳戶中的儲存庫。AccountB 使用者必須將其電腦設定為使用 git-remote-codecommit (建議) 或認證協助程式來存取 AccountA 中的共用 CodeCommit 存放庫。不過，在存取 AccountB 中的儲存庫時，您可以繼續使用 SSH 金鑰或 Git 登入資料。

請按照下列步驟使用 git-remote-codecommit 設定存取權。如果您尚未安裝git-remote-codecommit，請從 Python Package 索引網站git-remote-codecommit上下載它。

若要設定 AWS CLI 和 Git 以進行跨帳戶存取

1. 在本地計算機 AWS CLI 上安裝。請參閱安裝 AWS CLI 中適用於您作業系統的指示。
2. 在本機電腦上安裝 Git。若要安裝 Git，建議可使用 Git Downloads 或 Git for Windows 等網站。

> **Note**
>
> CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其使用的方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱中的資訊疑難排解。

3. 從終端機或命令列，在您要複製儲存庫的目錄位置執行 git config --local user.name 和 git config --local user.email 命令，以針對您要對儲存庫所做的遞交，設定使用者名稱和電子郵件。例如：

```
git config --local user.name "Saanvi Sarkar"
git config --local user.email saanvi_sarkar@example.com
```

這些命令不會傳回任何訊息，但您指定的電子郵件和使用者名稱，將會與您對 AccountA 中的儲存庫所做的遞交相關聯。

4. 執行 aws configure --profile 命令，設定連線至 AccountB 中的資源時要使用的預設描述檔。出現提示時，請提供 IAM 使用者的存取金鑰和秘密金鑰。

> **Note**
>
> 如果您已經安裝 AWS CLI 並設定了設定檔，則可以略過此步驟。

例如，執行下列命令來建立預設設 AWS CLI 定檔，您可用來存取美國東部 (俄亥俄) 的 AccountB (us-east-2) 中的 AWS 資源：

```
aws configure
```

當出現提示時，請提供下列資訊：

```
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. 再次執行 aws configure --profile 命令，設定連線至 AccountA 中的儲存庫時要使用的指定描述檔。出現提示時，請提供 IAM 使用者的存取金鑰和秘密金鑰。例如，執行下列命令來建立名為的 AWS CLI 設定檔，*MyCrossAccountAccessProfile*該設定檔可用來存取美國東部 (俄亥俄) 的 AccountA (us-east-2) 中的儲存庫：

```
aws configure --profile MyCrossAccountAccessProfile
```

當出現提示時，請提供下列資訊：

```
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. 在純文字編輯器中，開啟 config 檔案，也稱為 AWS CLI 組態檔案。根據您的操作系統，此文件可能位於 Linux，macOS 或 Unix ~/.aws/config 上，或在###上：\ 用戶\ 用戶#\ .aws\ 配置在 Windows 上。
7. 在檔案中，找出您已針對存取 AccountB 中的儲存庫所設定之對應預設描述檔的項目。其看起來與下列類似：

```
[default]
region = us-east-2
output = json
```

將 account 新增到描述檔組態。提供 AccountB 的 AWS 帳號 ID。例如：

```
[default]
account = 888888888888
region = us-east-2
output = json
```

8. 在檔案中，尋找與您剛建立的*MyCrossAccountAccessProfile*設定檔相對應的項目。其看起來與下列類似：

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
output = json
```

將 account、role_arn 和 source_profile 新增到描述檔組態。提供帳戶 AccountA 的 Amazon Web Services 帳戶 ID、您假設存取其他帳戶中存取儲存庫的帳戶 A 中角色的 ARN，以及帳戶 AccountB 中預設設 AWS CLI 定檔的名稱。例如：

```
[profile MyCrossAccountAccessProfile]
region = us-east-2
account = 111122223333
role_arn = arn:aws:iam::111122223333:role/MyCrossAccountRepositoryContributorRole
source_profile = default
output = json
```

儲存您的變更並關閉純文字編輯器。

步驟 2：克隆並訪問 AccountA 中的 CodeCommit 存儲庫

執行git clonegit push、git pull以及複製、推送至跨帳戶 CodeCommit儲存庫以及從中提取。您也可以登入 AWS 管理主控台、切換角色，以及使用主 CodeCommit 控台與其他帳戶中的儲存庫互動。

> **Note**
>
> 視 IAM 角色的設定方式而定，您可能可以在的預設頁面上檢視存放庫 CodeCommit。如果您無法檢視儲存庫，請要求儲存庫管理員以電子郵件傳送連結至 CodeCommit 主控台中共用存放庫「代碼」頁的 URL 連結。URL 類似如下：

```
https://console.aws.amazon.com/codecommit/home?region=us-east-2#/
repository/MySharedDemoRepo/browse/HEAD/--/
```

將跨帳戶儲存庫複製到本機電腦

1. 在命令列或終端機，在您要複製儲存庫的目錄中執行 git clone 命令，並指定 HTTPS (GRC) 複製 URL。例如：

   ```
   git clone codecommit://MyCrossAccountAccessProfile@MySharedDemoRepo
   ```

   除非您另外指定，否則儲存庫會複製到與儲存庫同名的子目錄。
2. 切換到複製的儲存庫所在的目錄，然後新增檔案或變更檔案。例如，您可以新增名為 *NewFile.txt* 的檔案。
3. 將文件添加到本地存儲庫的跟踪更改中，提交更改並將文件推送到 CodeCommit 存儲庫中。例如：

   ```
   git add NewFile.txt
   git commit -m "Added a file to test cross-account access to this repository"
   git push
   ```

   如需詳細資訊，請參閱 開始使用 Git 和 AWS CodeCommit。

現在您已經添加了一個文件，請前往 CodeCommit 控制台查看您的提交，查看其他用戶對存儲庫的更改，參與提取請求等。

在主控台中存取跨帳戶儲存庫 CodeCommit

1. 以 IAM 使用者身分登 AWS Management Console 入 AccountB (*8888888888*)，該使用者已獲得對帳戶 AccountA 中存放庫的跨帳戶存取權限。
2. 在導覽列選擇您的使用者名稱，然後從下拉式清單中選擇 Switch Role (切換角色)。

   > **Note**
   >
   > 如果您是第一次選取此選項，請檢閱頁面上的資訊，然後再次選擇 Switch Role (切換角色)。

3. 在 Switch Role (切換角色) 頁面上，執行下列動作：

   - 在帳戶中，輸入 AccountA A 的帳戶識別碼 (例如，*11112* 2223333)。
   - 在角色中，輸入您想要假設存取 AccountA 中存放庫的角色名稱 (例如，*MyCrossAccountRepositoryContributorRole*)。
   - 在 Display Name (顯示名稱) 中，輸入此角色的易記名稱。當您擔任此角色時，此名稱會出現在主控台。您下次在主控台想要切換角色時，此名稱也會出現在擔任的角色清單中。
   - (選用) 在 Color (顏色) 中，選擇顯示名稱的標籤。
   - 選擇 Switch Role (切換角色)。

   如需詳細資訊，請參閱切換到角色 (AWS Management Console)。

4. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home

   如果擔任的角色有許可檢視 AccountA 中儲存庫的名稱，您會看到儲存庫清單，也會出現錯誤訊息，指出您沒有許可檢視其狀態。這是預期的行為。從清單中選擇共享儲存庫的名稱。

   如果擔任的角色沒有許可檢視 AccountA 中儲存庫的名稱，您會看到錯誤訊息和沒有儲存庫的空白清單。貼上儲存庫的 URL 連結，或修改主控台連結並將 `/list` 變更為共享儲存庫的名稱 (例如，*/MySharedDemoRepo*)。

5. 在 Code (程式碼) 中，找出您從本機電腦新增的檔案名稱。選擇此檔案以瀏覽檔案中的程式碼，然後瀏覽儲存庫的其餘部分，並開始使用其功能。

   如需詳細資訊，請參閱 開始使用 AWS CodeCommit。

刪除 AWS CodeCommit 儲存庫

您可以使用 CodeCommit 主控台或 AWS CLI 刪除 CodeCommit 存放庫。

> **Note**
>
> 刪除儲存庫不會刪除該儲存庫的任何本機副本 (本機儲存庫)。若要刪除本機存放庫，請使用本機電腦的目錄和檔案管理工具。

主題

刪除 CodeCommit 存放庫（控制台）

請依照下列步驟使用 CodeCommit 主控台刪除存 CodeCommit 放庫。

> **Important**
>
> 刪除 CodeCommit 儲存庫後，您將無法再將其複製到任何本機存放庫或共用存放庫。您也不再能夠從任何本地回購或共享回購中提取數據或將數據推送到它。這個操作無法復原。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要刪除的儲存庫名稱。
3. 在導覽窗格中，選擇設定。
4. 在 General (一般) 索引標籤的 Delete repository (刪除儲存庫) 中，選擇 Delete repository (刪除儲存庫)。輸入 **`delete`**，然後選擇 Delete (刪除)。就會永久刪除這個儲存庫。

> **Note**
>
> 刪除中的存放庫 CodeCommit 並不會刪除任何本機存放庫。

刪除本機存放庫

使用本機電腦的目錄和檔案管理工具刪除包含本機存放庫的目錄。

刪除本機 CodeCommit 存放庫並不會刪除任何可能連線的儲存庫。

刪除存 CodeCommit 放庫 (AWS CLI)

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用 AWS CLI 刪除 CodeCommit 存放庫，請執行指delete-repository令，指定要刪除的 CodeCommit 存放庫名稱 (使用`--repository-name`選項)。

> **Important**
>
> 刪除 CodeCommit 儲存庫後，您將無法再將其複製到任何本機存放庫或共用存放庫。您也不再能夠從任何本地回購或共享回購中提取數據或將數據推送到它。這個操作無法復原。

> **Tip**
>
> 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

例如，若要刪除名為 MyDemoRepo 的儲存庫：

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

如果成功，永久刪除的 CodeCommit 存放庫 ID 會顯示在輸出中：

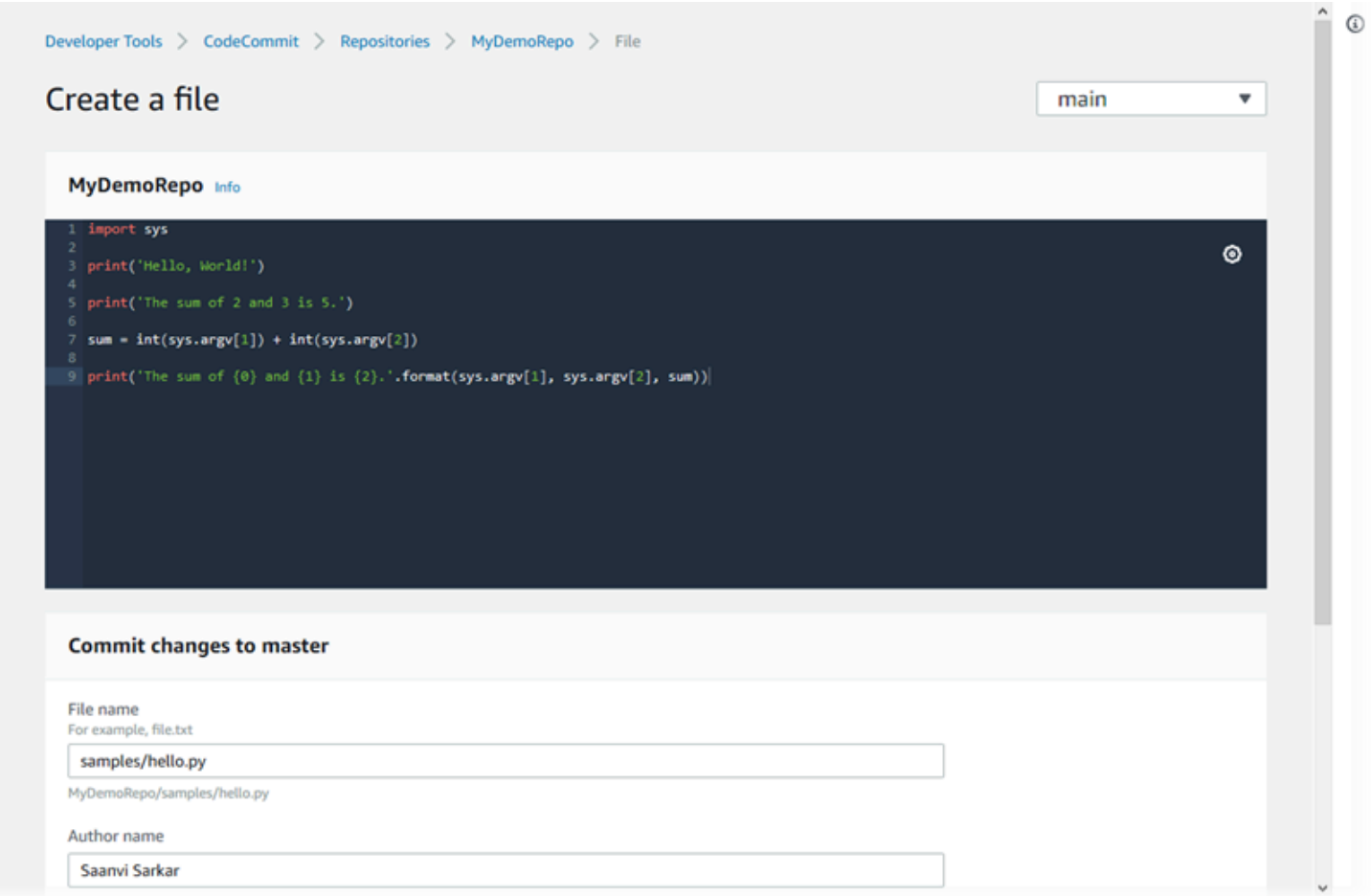
```
{
    "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

刪除 CodeCommit 存放庫並不會刪除任何可能連線到該儲存庫的本機存放庫。

在中使用檔案AWS CodeCommit儲存庫

在 CodeCommit 中，檔案是一種受版本控制、獨立自足的資訊片段，可供您及存放檔案的儲存庫和分支的其他使用者使用。就像在電腦上一樣，您可以用目錄結構來組織儲存庫檔案。與電腦不同，CodeCommit 會自動追蹤檔案的所有變更。您可以比較檔案的各個版本，並將檔案的不同版本存放在不同的儲存庫分支中。

若要在儲存庫中新增或編輯檔案，您可以使用 Git 用戶端。您也可以使用 CodeCommit 主控台，AWS CLI或 CodeCommit API。



如需有關在 CodeCommit 中使用儲存庫其他部分的資訊，請參使用儲存庫、使用提取請求、使用分支、使用提交，和使用者偏好設定。

主題

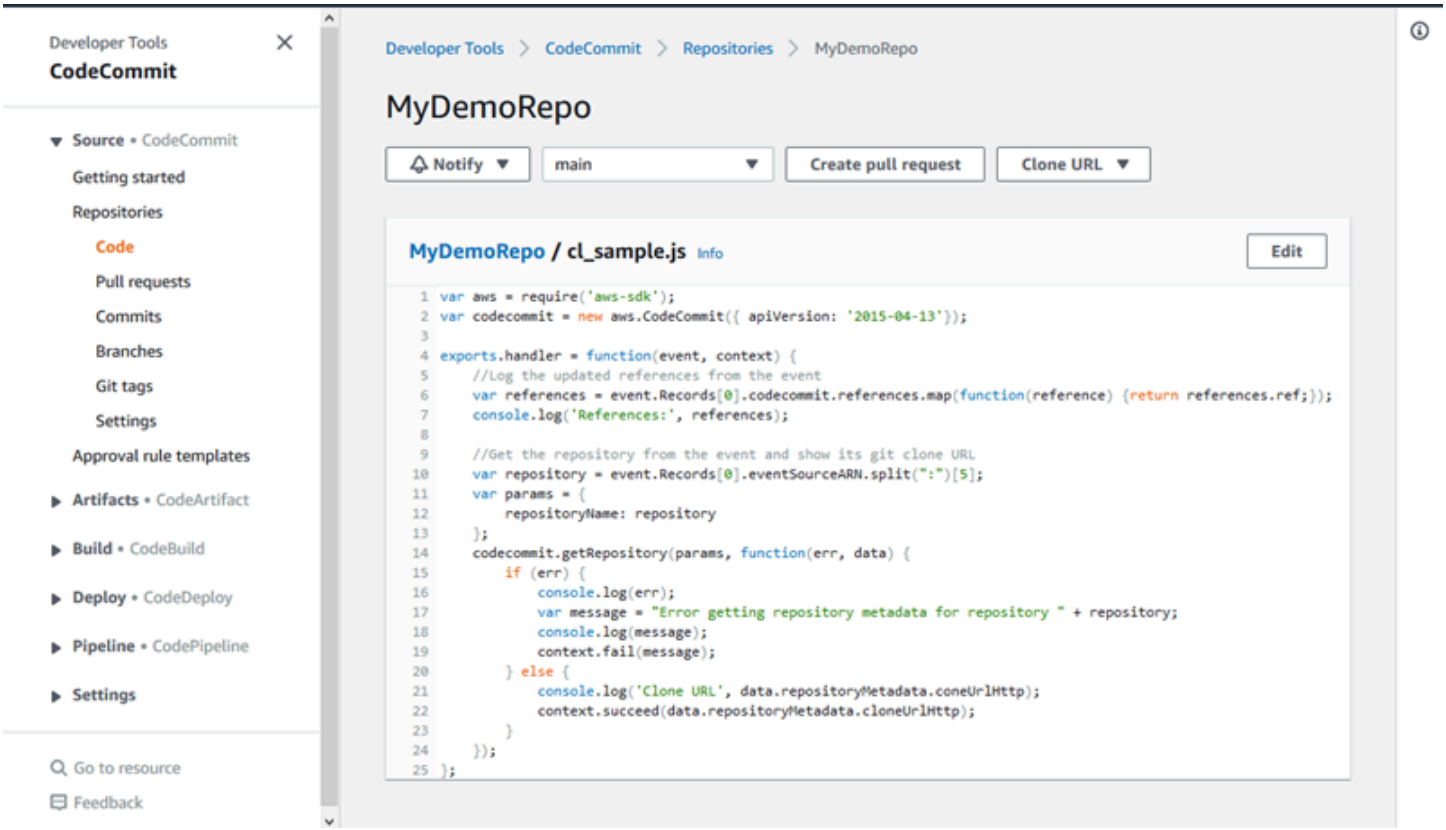
- 瀏覽中的檔案AWS CodeCommit儲存庫
- 建立檔案或將檔案新增至AWS CodeCommit儲存庫

- 編輯AWS CodeCommit儲存庫中檔案的內容

瀏覽中的檔案AWS CodeCommit儲存庫

連接到 CodeCommit 儲存庫時，您可以將它複製到本機儲存庫或使用 CodeCommit 主控台來瀏覽其內容。此主題說明如何使用CodeCommit來瀏覽CodeCommit儲存庫。

> **Note**
>
> 對於作用中的 CodeCommit 使用者，從 CodeCommit 主控台瀏覽程式碼不需任何費用。如需何時可能收費的詳細資訊，請參閱定價。



瀏覽CodeCommit儲存庫

您可以使用CodeCommit來檢儲存庫中包含的檔案，或快速讀檔案的內容。

瀏覽儲存庫的內容

1. 開啟CodeCommit主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 頁面上，從儲存庫清單選擇您想要瀏覽的儲存庫。
3. 在 Code (程式碼) 檢視中，瀏覽儲存庫預設分支的內容。

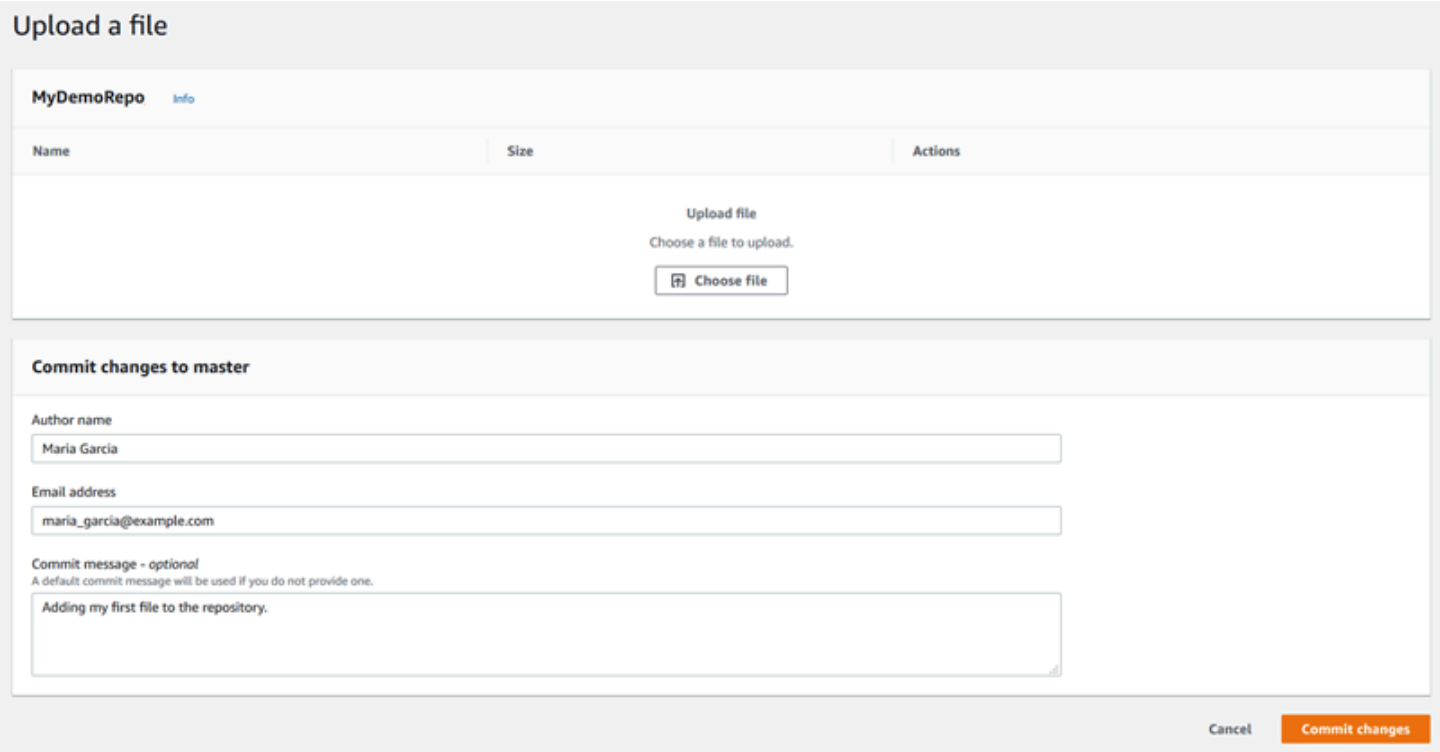
   若要將檢視變更為不同的分支或標籤，請選擇檢視選取器按鈕。從下拉式清單選擇分支或標籤名稱，或是在篩選方塊中輸入分支或標籤的名稱，然後從清單中選擇它。
4. 執行下列任一步驟：

   - 若要檢視目錄的內容，從清單中選擇目錄。您可以在導覽清單中選擇任何目錄以回到該目錄檢視。您也可以使用目錄清單上方的向上箭頭。
   - 若要檢視檔案的內容，從清單中選擇檔案。如果檔案大於遞交物件限制，就無法再顯示在主控台，必須改為在本機儲存庫中檢視。如需詳細資訊，請參閱 配額。若要結束檔案檢視，請從程式碼導覽列中，選擇您要檢視的目錄。

> **Note**
>
> 在主控台，並非所有二進位檔案都會顯示。如果您選擇一個二進制文件並且它可能是可以查看的，則會出現警告訊息，請求您確認是否要顯示內容。若要檢視檔案，請選擇 Show file contents (顯示檔案內容)。若不要檢視檔案，請從程式碼導覽列中，選擇您要檢視的目錄。
>
> 如果您選擇 Markdown 檔案 (.md)，請使用轉譯的 Markdown和Markdown 來源按鈕在渲染視圖和語法視圖之間切換。如需詳細資訊，請參閱「」在主控台中使用 Markdown。

建立檔案或將檔案新增至AWS CodeCommit儲存庫

您可以使用 CodeCommit 主控台、AWS CLI或 Git 用户端來將檔案新增至儲存庫。您可以從您的本機電腦上傳檔案至儲存庫，或者您可以使用主控台中的程式碼編輯器來建立檔案。編輯器是新增簡單檔案 (例如 readme.md 檔案) 至儲存庫中分支的一個快速且簡單的方式。



主題

- 創建或上傳文件（控制台）
- 新增檔案 (AWS CLI)
- 新增檔案 (Git)

創建或上傳文件（控制台）

您可以使用 CodeCommit 主控台來建立檔案，並將它新增至 CodeCommit 儲存庫中的分支。在建立檔案時，您可以提供使用者名稱和電子郵件地址。您也可以新增遞交訊息，讓其他使用者了解是誰新增了該檔案，以及原因。您也可以直接從本機電腦將檔案上傳至儲存庫中的分支。

將檔案新增到儲存庫

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要新增檔案所在的儲存庫。
3. 在 Code (程式碼) 檢視中，選擇您要新增檔案所在的分支。依預設，當您開啟 Code (程式碼) 檢視時，會顯示預設分支的內容。

若要將檢視變更至不同的分支，請選擇檢視選取器按鈕。從下拉式清單選擇分支名稱，或是在篩選方塊中輸入分支的名稱，然後從清單中選擇它。

4. 選擇 Add file (新增檔案)，然後選擇以下其中一個選項：
   - 若要使用程式碼編輯器來建立檔案的內容，並將它新增至儲存庫，請選擇 Create file (建立檔案)。
   - 若要從您的本機電腦將檔案上傳至儲存庫，請選擇 Upload file (上傳檔案)。
5. 把將此檔案新增至儲存庫的人員和原因等相關資訊提供給其他使用者。
   - 在 Author Name (作者名稱) 中，輸入名稱。此名稱會同時做為作者名稱和遞交資訊中的遞交名稱。CodeCommit 會預設為使用您的 IAM 使用者名稱或主控台登入的衍生做為作者名稱。
   - In電子郵件地址中，輸入電子郵件地址，讓其他儲存庫使用者可以與您聯絡變更的相關事項。
   - 在 Commit message (遞交訊息) 中，輸入簡短描述。此為選用步驟，但非常建議您加以執行。否則，會使用預設的遞交訊息。
6. 執行下列任一步驟：
   - 如果您要上傳檔案，請從您的本機電腦選擇檔案。
   - 如果要建立檔案，請在程式碼編輯器中輸入您要新增的內容，並為檔案提供名稱。
7. 選擇 Commit changes (遞交變更)。

新增檔案 (AWS CLI)

您可以使用AWS CLI與put-file命令，在 CodeCommit 儲存庫中新增檔案。您也可以使用 put-file 命令來為檔案新增目錄或路徑結構。

> **Note**
>
> 使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

將檔案新增到儲存庫

1. 在您的本機電腦上，建立您要新增至 CodeCommit 儲存庫的檔案。
2. 在終端機或命令列，執行 put-file 命令，並指定：
   - 您要新增檔案所在的儲存庫。

- 您要新增檔案所在的分支。
- 對該分支所進行最新遞交的完整遞交 ID，也稱為頂端或標頭遞交。
- 檔案的本機位置。用於此位置的語法因您的本機作業系統而有所不同。
- 您要新增的檔案名稱，包括已更新檔案在儲存庫中的存放路徑 (如果有)。
- 要與此檔案建立關聯的使用者名稱和電子郵件。
- 說明為何新增此檔案的遞交訊息。

使用者名稱、電子郵件地址和遞交訊息是選用的，但可協助其他使用者知道誰進行了變更以及原因。如果您不提供使用者名稱，會預 CodeCommit 使用您的 IAM 使用者名稱或主控台登入的衍生做為作者名稱。

例如，若要將名為 *ExampleSolution.py* 的檔案新增至名為 *MyDemoRepo* 儲存庫中名為 *feature-randomizationfeature* 的分支，且其最近遞交的 ID 為 *4c925148EXAMPLE*，請執行下列動作：

```
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**
>
> 新增二進位檔案時，請確定您使用 `fileb://` 來指定本機檔案的位置。

如果此命令成功執行，您會看到類似如下的輸出傳回：

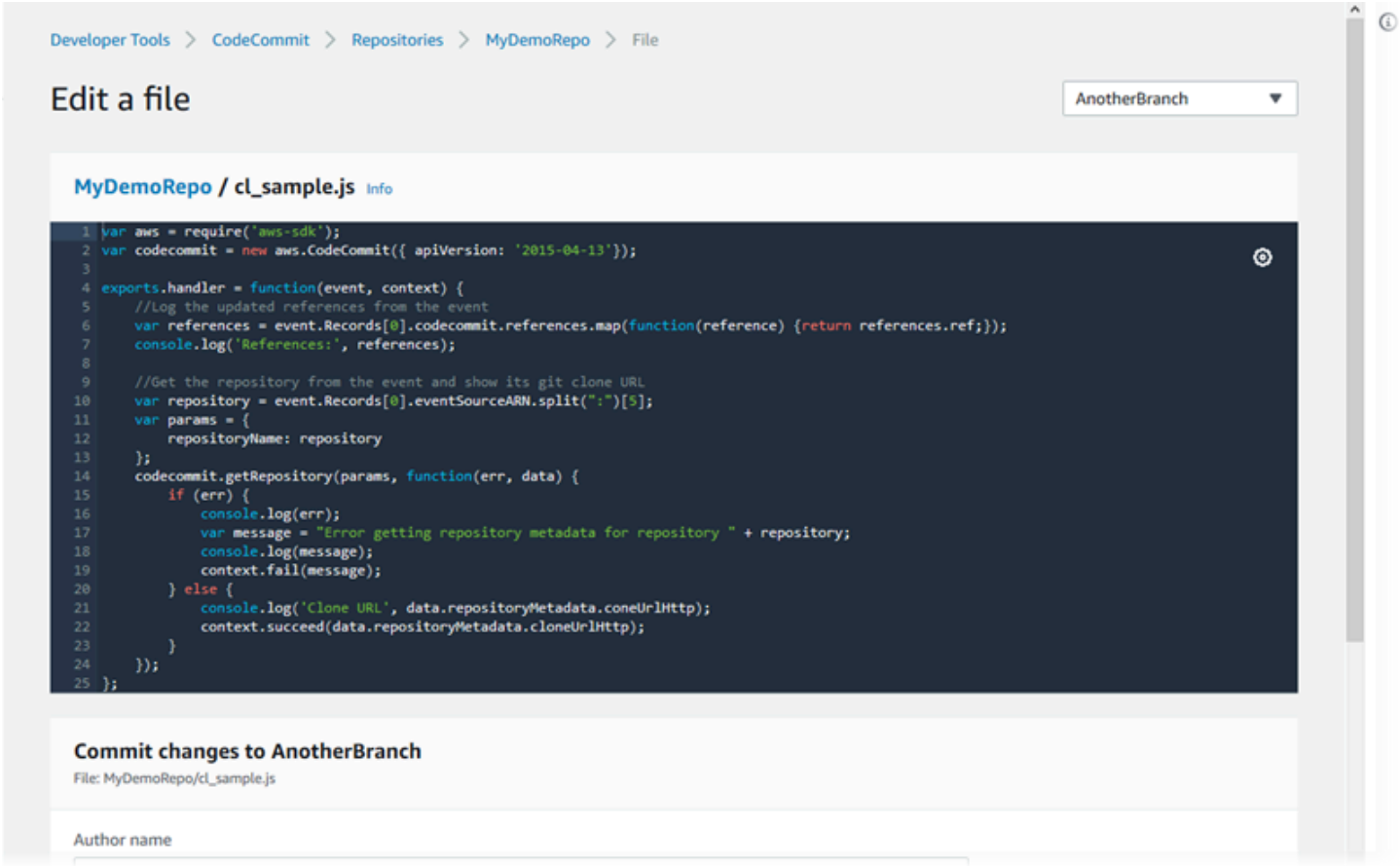
```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

新增檔案 (Git)

您可以在本機儲存庫中新增檔案，然後將變更推送至 CodeCommit 儲存庫。如需詳細資訊，請參閱開始使用 Git 和 AWS CodeCommit。

編輯AWS CodeCommit儲存庫中檔案的內容

您可以使用 CodeCommit 主控AWS CLI台或 Git 用戶端來編輯 CodeCommit 儲存庫中檔案的內容。



主題

- 編輯檔案 (主控台)
- 編輯或刪除檔案 (AWS CLI)
- 編輯檔案 (Git)

編輯檔案 (主控台)

您可以使用 CodeCommit 控制台編輯已添加到 CodeCommit存儲庫中分支的文件。在編輯檔案時，您可以提供您的使用者名稱和電子郵件地址。您也可以新增遞交訊息，讓其他使用者了解是誰做此變更，以及變更原因。

編輯儲存庫中的檔案

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要編輯的檔案所在的儲存庫。
3. 在 Code (程式碼) 檢視中，選擇您要編輯的檔案所在的分支。依預設，當您開啟 Code (程式碼) 檢視時，會顯示預設分支的內容。

   若要將檢視變更至不同的分支，請選擇檢視選取器按鈕。從下拉式清單選擇分支名稱，或是在篩選方塊中輸入分支的名稱，然後從清單中選擇它。
4. 導覽分支的內容，選擇您要編輯的檔案。在檔案檢視中，選擇 Edit (編輯)。

   > **Note**
   >
   > 如果您選擇二進位檔案，則會出現警告訊息，請求您確認是否要顯示內容。您不應使用 CodeCommit 控制台編輯二進製文件。
5. 編輯檔案，並將做此變更的使用者及變更原因等資訊提供給其他使用者。
   - 在 Author Name (作者名稱) 中，輸入名稱。這個名稱在提交信息中同時用作作者姓名和提交者名稱。 CodeCommit 預設使用您的 IAM 使用者名稱或衍生主控台登入作為作者名稱。
   - 在 [電子郵件地址] 中，輸入電子郵件地址，以便其他儲存庫使用者就此變更與您聯絡。
   - 在 Commit message (遞交訊息) 中，輸入變更的簡短描述。
6. 選擇 Commit changes (遞交變更)，儲存您對檔案所做的變更，並將變更遞交到儲存庫。

編輯或刪除檔案 (AWS CLI)

您可以使用AWS CLI和指put-file令對 CodeCommit 儲存庫中的檔案進行變更。若要將已變更的檔案存放在不同於原始檔案的位置，您也可以使用 put-file 命令為已變更的檔案新增目錄或路徑結構。如果您想要刪除整個檔案，可以使用 delete-file 命令。

> **Note**
>
> 若要搭配使用AWS CLI指令 CodeCommit，請安裝AWS CLI. 如需詳細資訊，請參閱命令列參考。

編輯儲存庫中的檔案

1. 使用檔案的本機副本，進行您要新增到 CodeCommit 儲存庫的變更。
2. 在終端機或命令列，執行 put-file 命令，並指定：
   - 要新增已編輯檔案的儲存庫。
   - 要新增已編輯檔案的分支。
   - 對該分支所進行最新遞交的完整遞交 ID，也稱為頂端或標頭遞交。
   - 檔案的本機位置。
   - 您要新增的已更新檔案名稱，包括已更新檔案在儲存庫中的存放路徑 (如果有)。
   - 要與此檔案變更建立關聯的使用者名稱和電子郵件。
   - 用以說明您所做變更的遞交訊息。

   使用者名稱、電子郵件地址和遞交訊息是選用的，但可協助其他使用者知道誰進行了變更以及原因。如果您未提供使用者名稱，則 CodeCommit 預設使用您的 IAM 使用者名稱或衍生主控台登入。

   例如，要將對名為 *ExampleSolution.py* 的文件所做的編輯添加*MyDemoRepo*到名為功能*####*的分支的存儲庫中，其最近提交的 ID 為 *4c925148 ##*：

   ```
   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**
   >
   > 如果您想要新增已變更的二進位檔案，請務必使用 `--file-content` 搭配標記法 **`fileb://`***`MyDirectory/MyFile`*`.raw`。

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
   "blobId": "2eb4af3bEXAMPLE",
   "commitId": "317f8570EXAMPLE",
   "treeId": "347a3408EXAMPLE"
}
```

若要刪除檔案，請使用 delete-file 命令。例如，要刪除名為 *main* 的分支中一個名為 *Readme.md* 的文件，最近提交 ID 為 *C5709475 ###*在存儲庫中名為 *MyDemoRepo*：

```
aws codecommit delete-file --repository-name MyDemoRepo --branch-name main --file-
path README.md --parent-commit-id c5709475EXAMPLE
```

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
  "blobId":"559b44fEXAMPLE",
  "commitId":"353cf655EXAMPLE",
  "filePath":"README.md",
  "treeId":"6bc824cEXAMPLE"
}
```

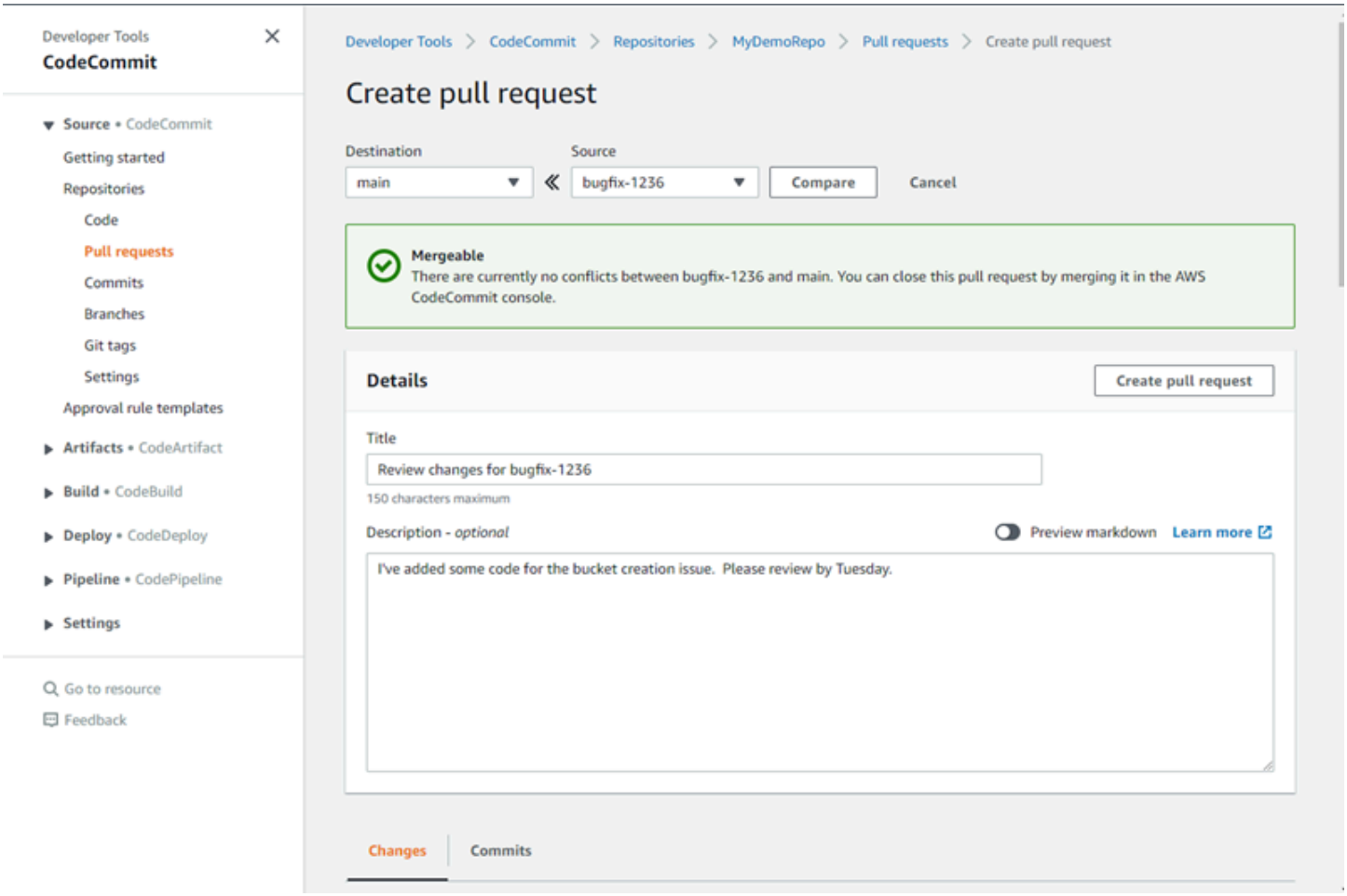
編輯檔案 (Git)

您可以編輯本地存儲庫中的文件，並將更改推送到 CodeCommit 存儲庫。如需詳細資訊，請參閱 開始使用 Git 和 AWS CodeCommit。

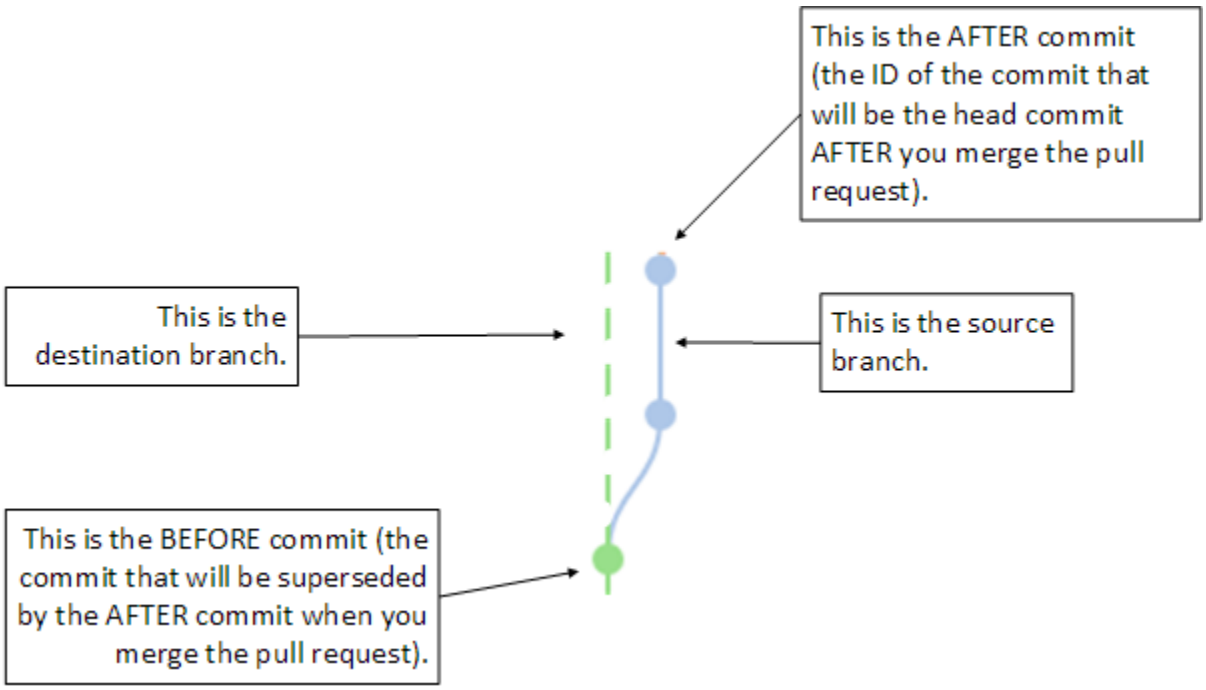
在中使用提取請求AWS CodeCommit儲存庫

為了讓您和其他儲存庫使用者可以檢閱程式碼變更、做註解，以及從一個分支合併到另一個分支，提取請求是主要方式。您可以使用提取請求來共同檢閱程式碼變更，以了解已發行軟體的次要變更或修正、新增的主要功能或新版本。以下是提取請求的一個可能的工作流程：

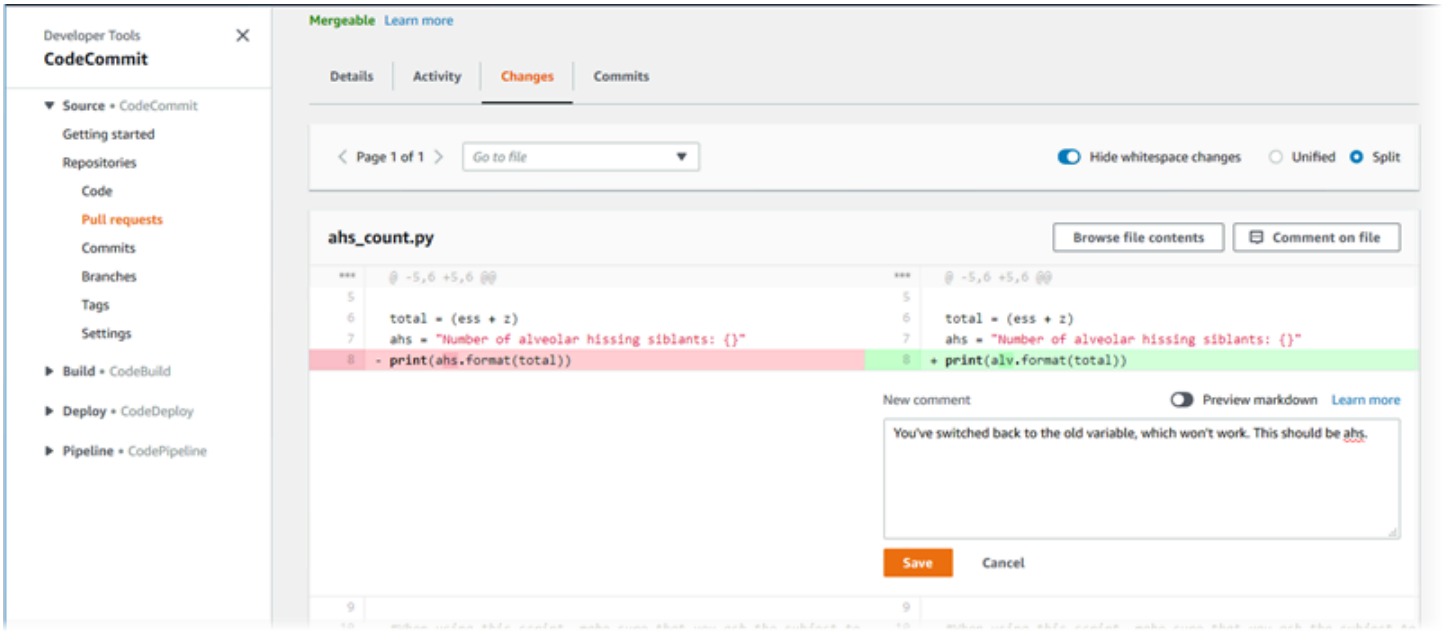
在名為的儲存庫中工作的開發人員 Li Juan，需要為即將發 MyDemoRepo 的產品版本開發新功能。為了將她的工作與可正式運作的程式碼分開，她在預設分支以外再建立一個分支，並命名為 *feature-randomizationfeature*。她撰寫程式碼、進行遞交，然後將新功能程式碼推送到這個分支。在將她的變更合併到預設分支之前，她希望其他儲存庫使用者能夠檢閱程式碼的品質。為了這樣做，她建立提取請求。提取請求包含她的工作分支與她想合併變更的程式碼分支 (在此案例中為預設分支) 之間的比較。她也可以建立核准規則，要求指定數目的使用者核准其提取請求。她甚至可以指定使用者的核准集區。其他使用者檢閱她的程式碼和變更，並新增註解和建議。她可能根據註解而以程式碼變更來多次更新她的工作分支。每當她在 CodeCommit 中將變更推送到該分支時，她的變更就會納入提取請求中。在提取請求已開啟的情況下，她也可能納入預定目的地分支中已發生的變更，讓使用者確信他們檢閱的是相關情境下所有提議的變更。當她和檢閱者都感到滿意，且已滿足核准規則 (如果有的話) 的條件時，她或其中一位檢閱者可合併程式碼並關閉提取請求。



提取請求需要兩個分支：來源分支 - 包含您希望檢閱的程式碼；目的地分支 - 在此處合併已檢閱的程式碼。來源分支包含「之後」遞交，此遞交包含您想要合併到目的地分支的變更。目的地分支包含「之前」遞交，這代表提取請求分支合併到目的地分支之前的程式碼狀態。合併策略的選擇會影響 CodeCommit 控制台中，提交在來源和目的地分支之間合併方式的詳細資訊。如需有關在 CodeCommit 中合併策略的詳細資訊，請參[合併提取請求（主控台）](#)。



提取請求會顯示建立提取請求時，在來源分支的頂端與目的地分支的最新遞交之間的差異，讓使用者可以檢視變更並做註解。您可以將變更遞交並推送到來源分支，以更新提取請求來回應註解。



在檢閱過您的程式碼，且已滿足核准規則需求 (如果有的話) 時，您可以使用下列其中一種方式來關閉提取請求：

- 在本機合併分支並推送您的變更。這會自動關閉請求如果使用向前快轉合併策略並且沒有合併衝突。

- 使用 AWS CodeCommit 主控台來關閉提取請求而不合併、解決合併中的衝突，或使用其中一個可用的合併策略來關閉和合併分支 (如果沒有衝突)。
- 使用 AWS CLI。

在您建立提取請求之前：

- 請確定已將您希望檢閱的程式碼變更遞交並推送到分支 (來源分支)。
- 為儲存庫設定通知，讓其他使用者收到有關提取請求及其變更的通知。(此為選用步驟，但建議執行。)
- 建立核准規則範本並與儲存庫建立關聯，以便為提取請求自動建立核准規則，以協助確保程式碼品質。如需詳細資訊，請參閱 使用核准規則範本。

當您在 Amazon Web Services 賬戶中為儲存庫用户設定 IAM 用户後，提取請求會更有效果。更容易識別哪個使用者發表哪個評論。另一個優點是使用者可以使用 Git 憑證存取儲存庫。如需詳細資訊，請參閱 步驟 1：初始配置 CodeCommit。您可以對其他類型的使用者 (包括聯合身分存取使用者) 使用提取請求。

如需有關在 CodeCommit 中使用儲存庫其他部分的資訊，請參使用儲存庫、使用核准規則範本、使用檔案、使用提交、使用分支，以及使用者偏好設定。

主題

- 建立提取請求
- 建立提取請求的核准規則
- 檢視提取請求AWS CodeCommit儲存庫
- 檢閱提取請求
- 更新提取請求
- 編輯或刪除提取請求的核準規則
- 提取請求的核可規則
- 合併中的提取請求AWS CodeCommit儲存庫
- 解決中提取請求的衝突AWS CodeCommit儲存庫
- 關閉中的提取請求AWS CodeCommit儲存庫

建立提取請求

建立提取請求可協助其他使用者在您將程式碼變更合併到另一個分支之前，查看和檢閱您的程式碼變更。首先，您會為程式碼變更建立分支。這稱為提取請求的來源分支。在您將變更遞交和推送至儲存庫之後，您可以建立一個提取請求，該請求會在提取請求關閉之後，將該分支的內容 (來源分支) 與您要合併變更所在的分支 (目的地分支) 比較。

您可以使用 AWS CodeCommit 主控台或 AWS CLI 來建立儲存庫的提取請求。

主題

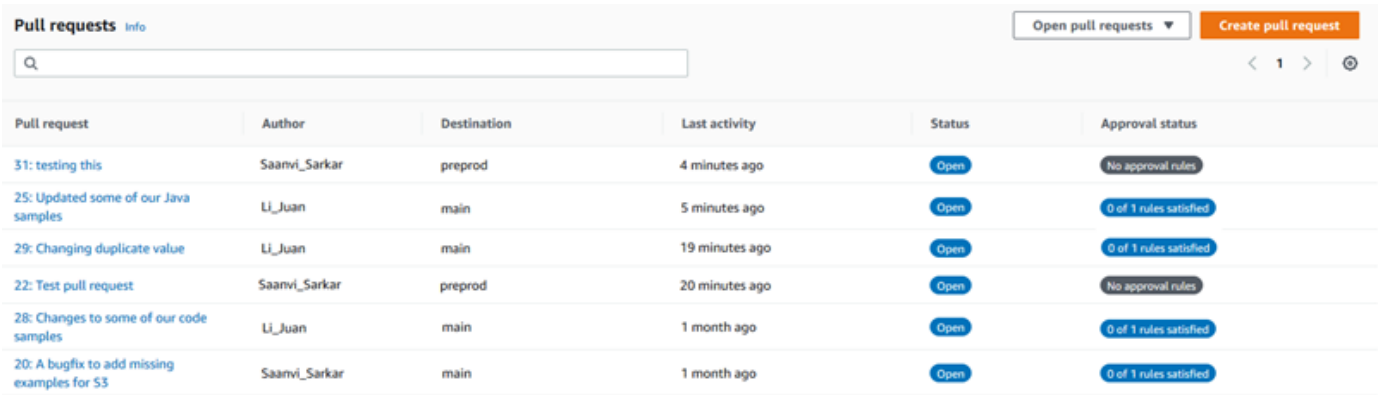
建立提取請求 (主控台)

您可以使用 CodeCommit 控制台在 CodeCommit 存儲庫中創建拉取請求。如果您的儲存庫已設定通知，當您建立提取請求時，訂閱的使用者會收到一封電子郵件。

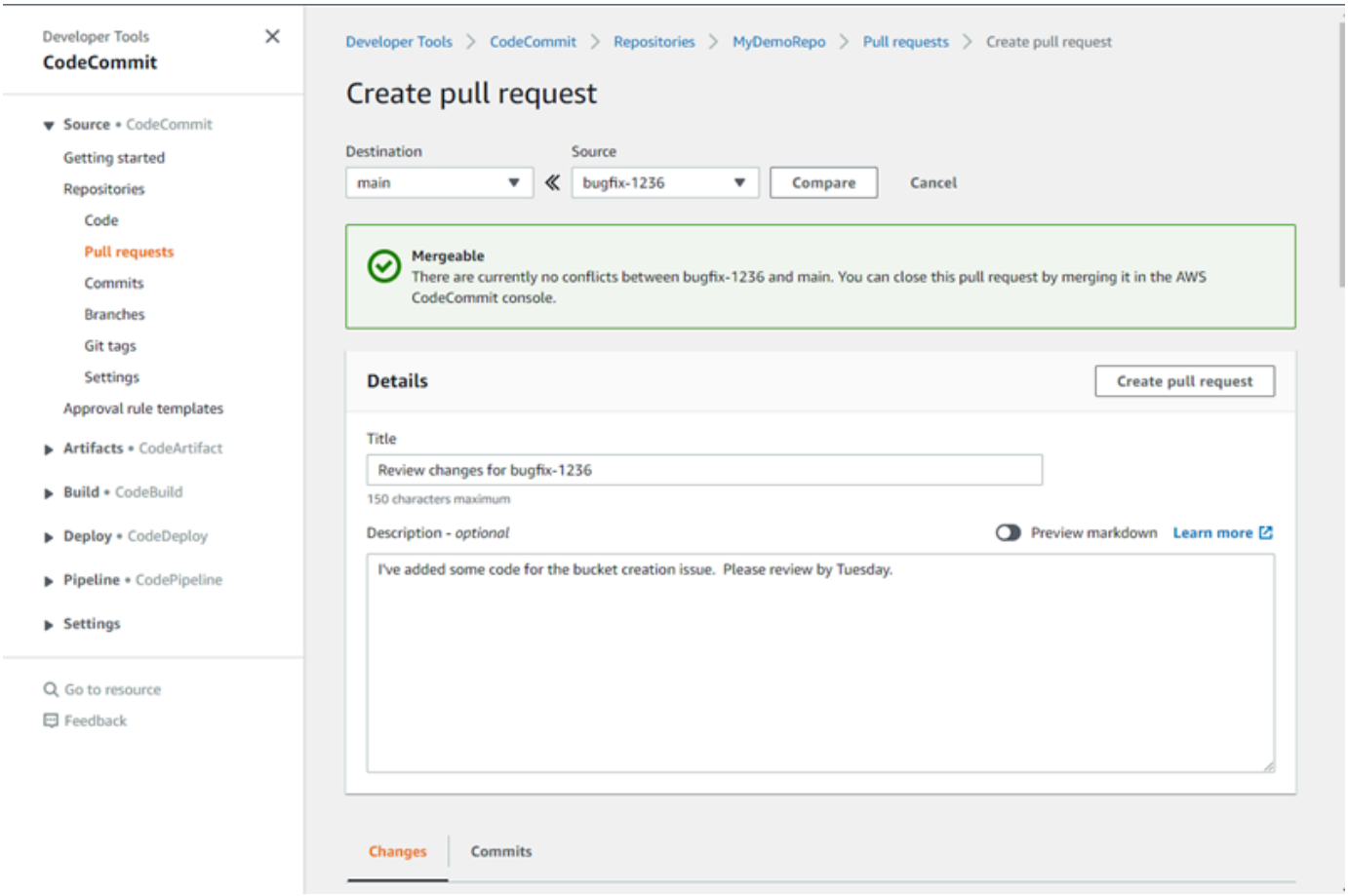
1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要建立提取請求所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。

> **Tip**
>
> 您也可以從 Branches (分支) 和 Code (程式碼) 建立提取請求。

4. 選擇 Create pull request (建立提取請求)。



5. 在 Create pull request (建立提取請求) 的 Source (來源) 中，選擇分支，其中包含您希望檢閱的變更。
6. InDestination (目的地)中，選擇您要在提取請求關閉時，合併程式碼變更所在的分支。
7. 選擇 Compare (比較)。系統會就這兩個分支執行比較，並顯示它們之間的差異。也會執行分析來判斷在提取請求關閉時，是否可自動合併這兩個分支。
8. 檢閱比較詳細資訊和變更，以確認提取請求包含您希望檢閱的變更和遞交。如果未包含，請調整您的來源和目的地分支選項，然後再次選擇 Compare (比較)。
9. 當您對提取請求的比較結果感到滿意時，請在 Title (標題) 中為此檢閱輸入簡短但具描述性的標題。這是在儲存庫的提取請求清單中顯示的標題。
10. (選用) 在 Description (描述) 中，輸入此檢閱的詳細資訊，以及對檢閱者有用的任何其他資訊。
11. 選擇 Create (建立)。



您的提取請求會出現在儲存庫的提取請求清單中。如果您設定通知時，Amazon SNS 主題的訂者會收到一封電子郵件，通知他們新建立的提取請求。

建立提取請求 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

若要使用 AWS for WordPressAWS CLI在 CodeCommit 儲存庫中建立提取請求

1. 執行 create-pull-request 命令，並指定：

   - 提取請求的名稱 (使用 --title 選項)。
   - 提取請求的描述(使用 --description 選項)。
   - create-pull-request 命令的目標清單，包括：
     - 建立提取請求所在的 CodeCommit 儲存庫的名稱 (使用repositoryName屬性)。
     - 分支名稱，其中包含您希望檢閱的程式碼變更，也稱為來源分支 (使用 sourceReference 屬性)。
     - (選用) 您要合併程式碼變更所在的分支名稱，也稱為目的地分支 (如果您不想合併到預設分支) (使用 destinationReference 屬性)。
   - 唯一的用戶端產生冪等符記 (使用 --client-request-token 選項)。

   此範例建立名為 *Pronunciation difficulty analyzer* 的提取請求，描述為 *Please review these changes by Tuesday*，而且以 *jane-branch* 來源分支為目標。拉取請求將被合併到默認分支*##*在 CodeCommit 庫中名為MyDemoRepo：

   ```
   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. 如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "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
    }
}
```

建立提取請求的核准規則

建立提取請求的核准規則會要求使用者核准提取請求，然後程式碼才能合併到目的地分支，有助於確保程式碼的品質。您可以指定必須核准提取請求的使用者數目。您也可以為規則指定使用者核准集區。如果您這麼做，則只有來自這些使用者的核准才計入規則所需的核准數目中。

> **Note**
>
> 您也可以建立核准規則範本，以協助您跨儲存庫 (適用於每個提取請求的核准規則) 的建立核准規則。如需詳細資訊，請參閱 使用核准規則範本。

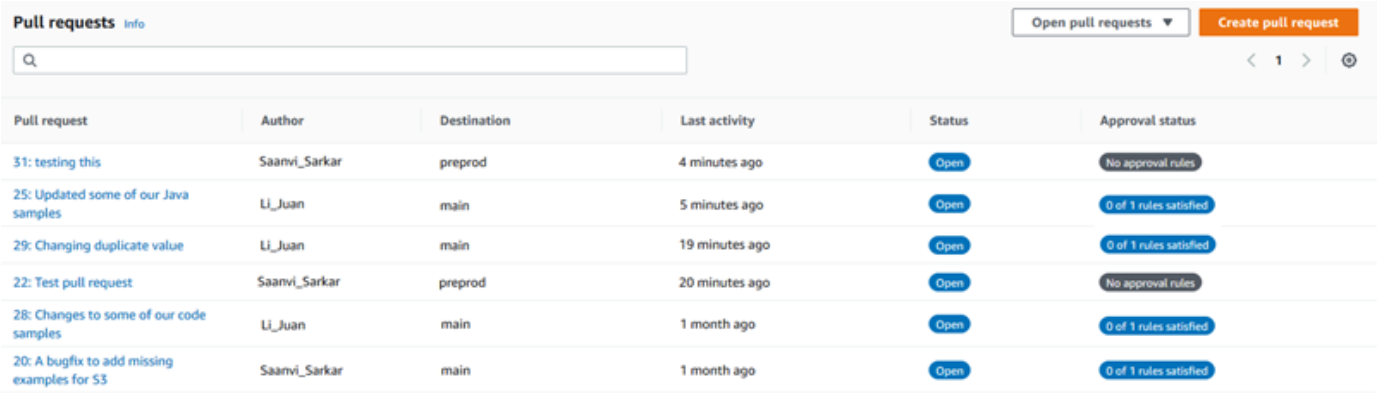
您可以使用 AWS CodeCommit 主控台或 AWS CLI 來為儲存庫建立分支。

主題

建立提取請求的核准規則 (主控台)

您可以使用 CodeCommit 主控台，建立核准規則 CodeCommit 儲存庫。

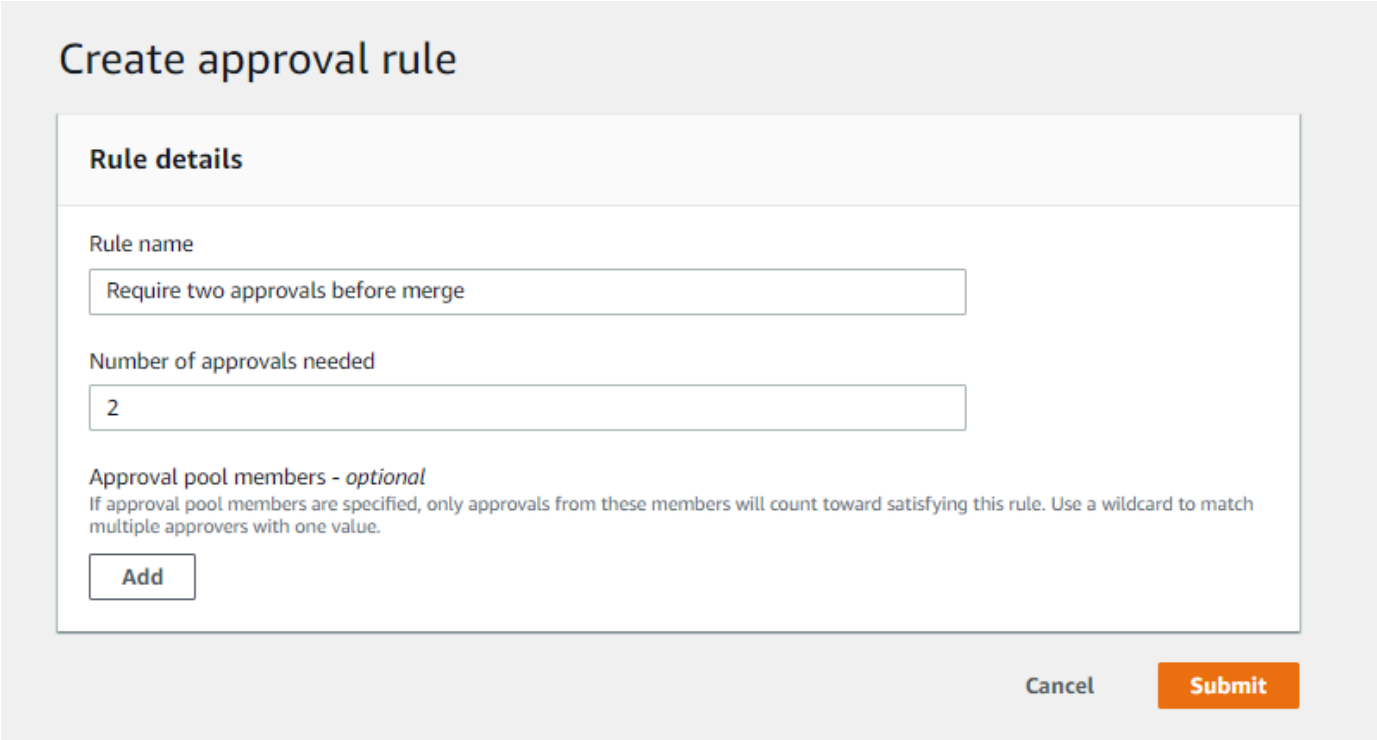
1. 開啟 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要為提取請求建立核准規則的儲存庫名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 從清單中選擇您要建立核准規則的提取請求。您只能對未結案的提取請求建立核准規則。



5. 在提取請求中，選擇 Approvals (核准)，然後選擇 Create approval rule (建立核准規則)。
6. 在 Rule name (規則名稱) 中，以描述性名稱來命名規則，讓您明白其用途。例如，如果需要兩個人先核准提取請求後，才能合併提取請求，您可以將規則命名為 **`Require two approvals before merge`**。

> **Note**
>
> 核准規則建立後就無法變更名稱。

在 Number of approvals needed (需要的核准數目) 中，輸入您要的數目。預設為 1。



7. (選擇性) 如果提取請求的核准必須來自特定使用者群組，請在 Approval rule members (核准規則成員) 中選擇 Add (新增)。在 Approver type (核准者類型) 中，選擇以下其中一項：

   - IAM 使用者名稱或假定角色：此選項會預先填入AWS帳戶 ID 與您用來登錄的帳戶，只需要一個名稱。對於名稱符合所提供名稱的 IAM 使用者和聯合存取使用者，可使用此選項。此選項非常強大，提供很大的靈活性。例如，如果您使用 Amazon Web Services 帳戶 123456789012 登入並選擇此選項，而且指定**`Mary_Major`**，以下全部算作來自該使用者的核准：
     - 帳戶中的 IAM 使用者 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中識別為 Major 的聯合身分使用者 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非您包含萬用字元 (`*Mary_Major`)，否則此選項無法識別擔任角色 **`CodeCommitReview`** 且角色工作階段名稱為 Mary_Major (`arn:aws:sts::123456789012:assumed-role/`

CodeCommitReview/Mary_Major) 的某人的作用中工作階段。您也可以明確指定角色名稱 (CodeCommitReview/Mary_Major)。

- 完全合格的 ARN：此選項可讓您指定 IAM 使用者或角色的完整 Amazon Resource Name (ARN)。此選項也可支援其他 AWS 服務所使用的擔任角色，例如 AWS Lambda 和 AWS CodeBuild。針對擔任的角色，ARN 格式應為 `arn:aws:sts::AccountID:assumed-role/RoleName` (若為角色) 和 `arn:aws:sts::AccountID:assumed-role/FunctionName` (若為函數)。

如果您選擇IAM 使用者名稱或假定角色作為核准者類型，數值」中，輸入 IAM 使用者或角色的名稱，或是使用者或角色的完整 ARN。再次選擇 Add (新增)，以新增更多使用者或角色，直到核准要計入所需核准數目中的所有使用者或角色，都已新增為止。

這兩種核准者類型都允許您在值中使用萬用字元 (*)。例如，如果您選擇IAM 使用者名稱或假定角色選項，且您指定**CodeCommitReview/***，所有擔任角色的使用者**CodeCommitReview**會計入核准集區中。個別角色工作階段名稱計入所需的核准者數目中。如此一來，Mary_Major 和 Li_Juan 在登入並擔任 CodeCommitReview 時都算作核准。如需 IAM ARN、萬用字元和格式的詳細資訊，請參閱IAM 識別碼。

> **Note**
>
> 核准規則不支援跨帳戶核准。

8. 核准規則設定完成後，請選擇 Submit (提交)。

建立提取請求的核准規則 (AWS CLI)

使用AWS CLI使用指令 CodeCommit，安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

在 中建立提取請求的核准規則 CodeCommit 儲存庫

1. 執行 create-pull-request-approval-rule 命令，並指定：

   - 提取請求的 ID (使用 --id 選項)。
   - 核准規則的名稱 (使用 --approval-rule-name 選項)。
   - 核准規則的內容 (使用 --approval-rule-content 選項)。

建立核准規則時，您有兩種方式在核准集區中指定核准者：

- CodeCommitApprovers：此選項僅需要 Amazon Web Services 帳戶和資源。對於名稱符合所提供資源名稱的 IAM 使用者和聯合存取使用者，可使用此選項。此選項非常強大，提供很大的靈活性。例如，如果您指定了 Amazon Web Services 帳戶 123456789012 和**Mary_Major**，以下全部算作來自該使用者的核准：
  - 帳戶中的 IAM 使用者 (arn:aws:iam::123456789012:user/Mary_Major)
  - 在 IAM 中識別為 Major 的聯合身分使用者 (arn:aws:sts::123456789012:federated-user/Mary_Major)

  除非您包含萬用字元 (*Mary_Major)，否則此選項無法識別擔任角色 **CodeCommitReview** 且角色工作階段名稱為 Mary_Major (arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major) 的某人的作用中工作階段。
- 完全合格的 ARN：此選項可讓您指定 IAM 使用者或角色的完整 Amazon Resource Name (ARN)。

如需 IAM ARN、萬用字元和格式的詳細資訊，請參閱IAM 識別碼。

下列範例建立名為的核准規則Require two approved approvers適用於 ID 為的提取請求27。此規則指定需要有來自核准集區的兩個核准。集區包括所有存取的使用者 CodeCommit 並承擔的作用**CodeCommitReview**中的123456789012Amazon Web Services 帳戶。還包括名為的 IAM 使用者或聯合身分使用者Nikhil_Jayashankar在同一 Amazon Web Services 帳戶：

```
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. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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"
    }
}
```

檢視提取請求AWS CodeCommit儲存庫

您可以使用 AWS CodeCommit 主控台或 AWS CLI 來檢視儲存庫的提取請求。依預設，您只能看到開啟中的提取請求，但您可以變更篩選以檢視所有提取請求、僅關閉的請求、僅您建立的提取請求等等。

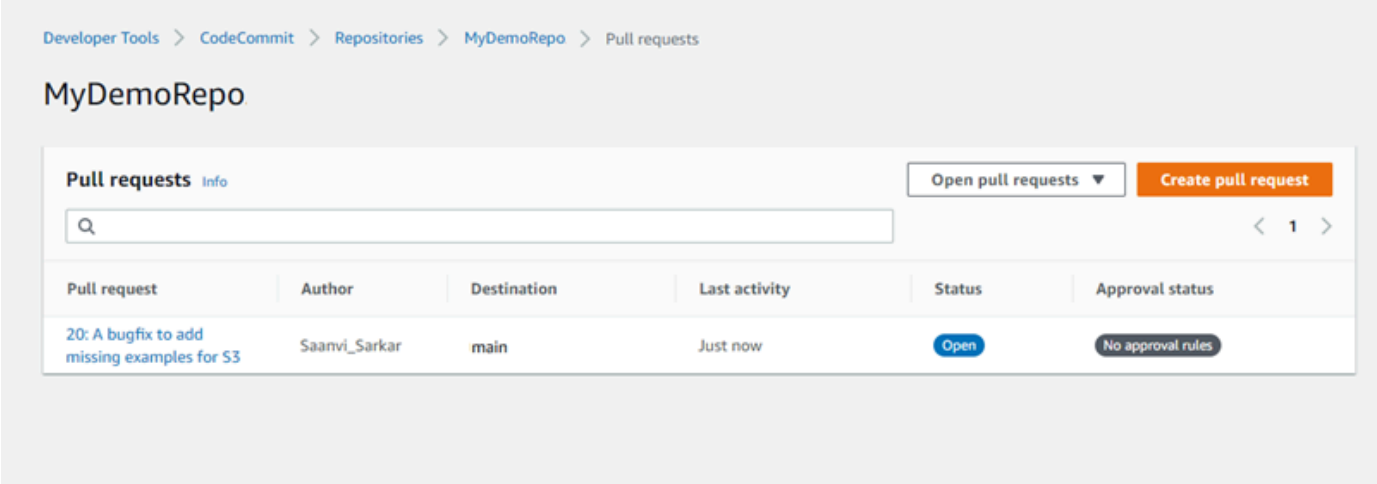
主題

- 檢視提取請求 (主控台)
- 檢視提取請求 (AWS CLI)

檢視提取請求 (主控台)

您可以使用AWS CodeCommit主控台來檢視 CodeCommit 儲存庫中提取請求的清單。透過變更篩選，您可以將清單顯示變更為僅向您顯示特定的一組提取請求。例如，您可以檢視您建立、狀態為 Open (開啟中) 的提取請求清單，或您可以選擇不同的篩選並檢視您所建立、狀態為 Closed (關閉) 的提取請求。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要檢視提取請求所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。



5. 若要變更顯示篩選，請從可用的篩選清單中選擇：

   - 開啟中提取請求(default): 顯示狀態為的所有提取請求開啟。
   - 所有提取請求：顯示所有提取請求。
   - 關閉提取請求：顯示狀態為的所有提取請求Closed。
   - 我的提取請求：顯示您建立的所有提取請求，而不論狀態為何。它不會顯示您加上評論或以其他方式參與的檢閱。
   - 我的開啟中提取請求：顯示您建立、狀態為開啟。
   - 我的關閉提取請求：顯示您建立、狀態為Closed。

6. 在顯示的清單中找到要檢視的提取請求時，請加以選擇。

檢視提取請求 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

請依照以下步驟，使用AWS CLI來檢視 CodeCommit 儲存庫中的提取請求。

1. 若要檢視儲存庫中提取請求的清單，請執行 list-pull-requests 命令，指定：

   - CodeCommit 儲存庫的名稱，您要檢視提取請求 (使用--repository-name選項）。
   - (選用) 提取請求的狀態 (使用 --pull-request-status 選項)。
   - （可選）創建提取請求的 IAM 用户的亞馬遜資源名稱 (ARN) (使用--author-arn選項）。
   - (選用) 可用來傳回結果批次的列舉字符 (使用 --next-token 選項)。
   - (選用) 每個請求傳回結果的數量有限制 (使用 --max-results 選項)。

例如，要列出由 IAM 用户使用 ARN 創建的拉取請求*arn: aws: iam# 1111111111:##/li_ #*和地位*##*在 CodeCommit 庫中名為MyDemoRepo：

```
aws codecommit list-pull-requests --author-arn arn:aws:iam::111111111111:user/
Li_Juan --pull-request-status CLOSED --repository-name MyDemoRepo
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
   "nextToken": "",
   "pullRequestIds": ["2","12","16","22","23","35","30","39","47"]
}
```

提取請求 ID 會以最新活動的順序顯示。

2. 若要檢視提取請求的詳細資訊，請執行 get-pull-request 命令加上 --pull-request-id 選項，指定提取請求的 ID。例如，若要檢視 ID 為 *27* 之提取請求的詳細資訊：

```
aws codecommit get-pull-request --pull-request-id 27
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要檢視提取要求的核准，請執行 get-pull-request-approval-state 命令，並指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 提取請求的修訂 ID (使用 --revision-id option)。您可以使用 get-pull-request 命令取得提取請求的目前修訂 ID。

   例如，若要檢視 ID 為 *8* 且修訂 ID 為 *9f29d167EXAMPLE* 的提取請求的核准：

   ```
   aws codecommit get-pull-request-approval-state --pull-request-id 8 --revision-
   id 9f29d167EXAMPLE
   ```

   如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "approvals": [
           {
               "userArn": "arn:aws:iam::123456789012:user/Mary_Major",
               "approvalState": "APPROVE"
           }
       ]
   ```

```
}
```

4. 若要檢視提取請求中的事件，請執行 describe-pull-request-events 命令加上 --pull-request-id 選項，指定提取請求的 ID。例如，若要檢視 ID 為 *8* 之提取請求的事件：

```
aws codecommit describe-pull-request-events --pull-request-id 8
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要檢視提取請求是否有任何合併衝突，請執行 get-merge-conflicts 命令，指定：

- CodeCommit 儲存庫的名稱（使用--repository-name選項）。
- 要在合併評估中使用的變更來源的分支、標籤、HEAD 或其他完整參考 (使用 --source-commit-specifier 選項)。
- 要在合併評估中使用的變更目的地的分支、標籤、HEAD 或其他完整參考 (使用 --destination-commit-specifier 選項)。
- 要使用的合併選項 (使用 --merge-option 選項)

例如，若要檢視名為######和一個名為##存儲庫中名為MyDemoRepo：

```
aws codecommit get-merge-conflicts --repository-name MyDemoRepo --source-commit-
specifier my-feature-branch --destination-commit-specifier main --merge-option
 FAST_FORWARD_MERGE
```

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "destinationCommitId": "fac04518EXAMPLE",
    "mergeable": false,
    "sourceCommitId": "16d097f03EXAMPLE"
}
```

檢閱提取請求

您可以使用主 AWS CodeCommit 控台來檢閱提取要求中包含的變更。您可以對請求、檔案，以及個別的程式碼行新增評論。您也可以回覆其他使用者的評論。如果您的儲存庫已設定通知，當使用者回覆您的評論或是當使用者對提取請求加上評論時，您會收到電子郵件。

您可以使用 AWS CLI 來評論提取請求並回覆註解。若要檢閱變更，您必須使用 CodeCommit 主控台、git diff指令或差異工具。

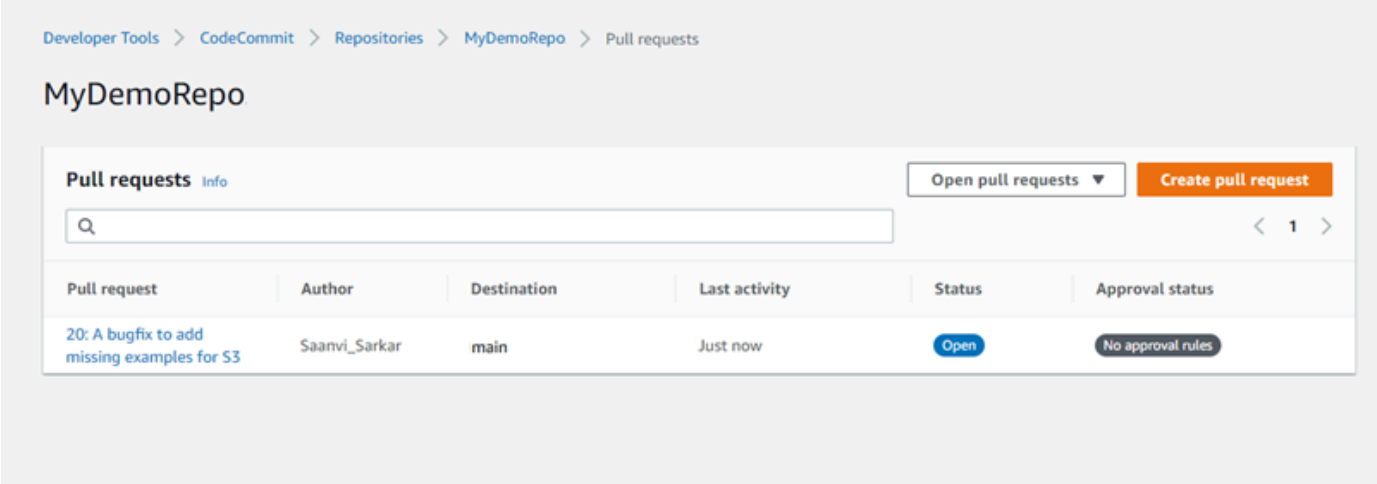
主題

- 檢閱提取要求 (主控台)
- 檢閱提取要求 (AWS CLI)

檢閱提取要求 (主控台)

您可以使用主 CodeCommit 控台來檢閱 CodeCommit 儲存庫中的提取要求。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。選擇您想要檢閱的開啟中提取請求。



> **Note**
>
> 您可以對已關閉或合併的提取請求作評論，但無法合併或重新開啟該提取請求。

5. 在提取請求中，選擇 Changes (變更)。
6. 執行以下任意一項：

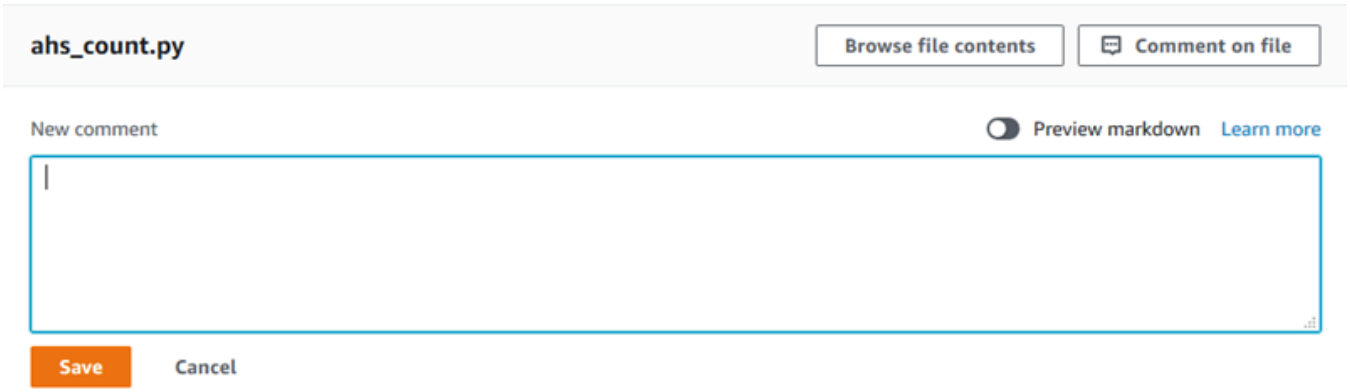
   - 若要對整個提取請求加上一般性評論，請在 Comments on changes (對變更加上評論) 的 New comment (新增評論) 中，輸入評論，然後選擇 Save (儲存)。您可以使用 Markdown，或也可以純文字輸入您的評論。

   - 若要對遞交中的檔案新增評論，請在 Changes (變更) 中找到檔案的名稱。選擇檔案名稱旁顯示的評論圖示

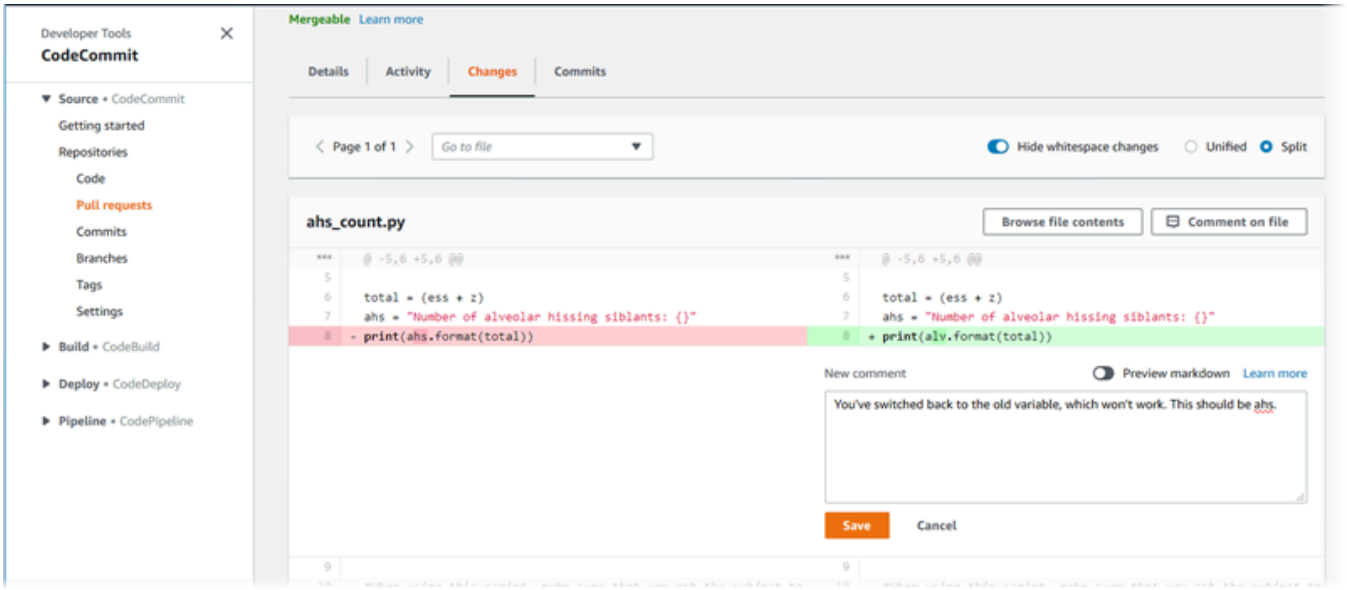
     ，輸入評論，然後選擇 Save (儲存)。



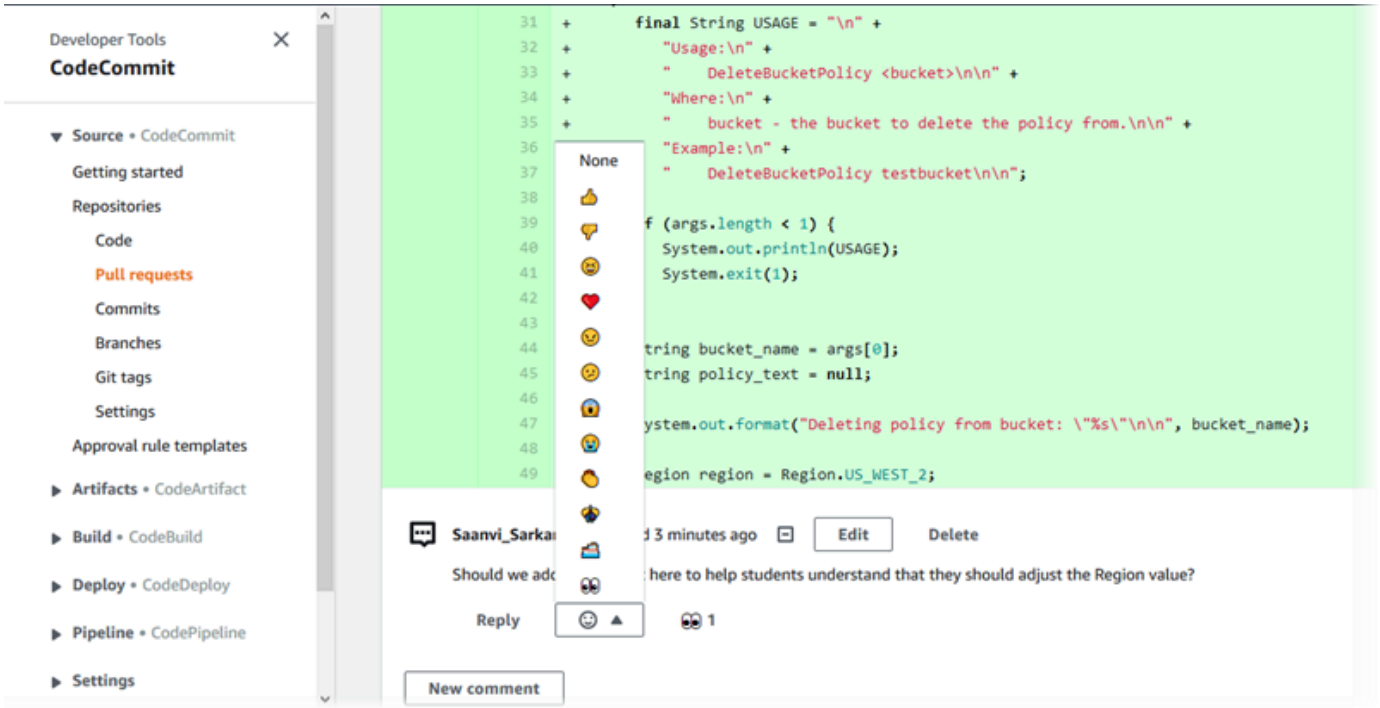
- 若要對提取請求中變更的行新增評論，在 Changes (變更) 中，移至您要加上評論的行。選擇該行顯示的評論圖示

  ，輸入評論，然後選擇 Save (儲存)。

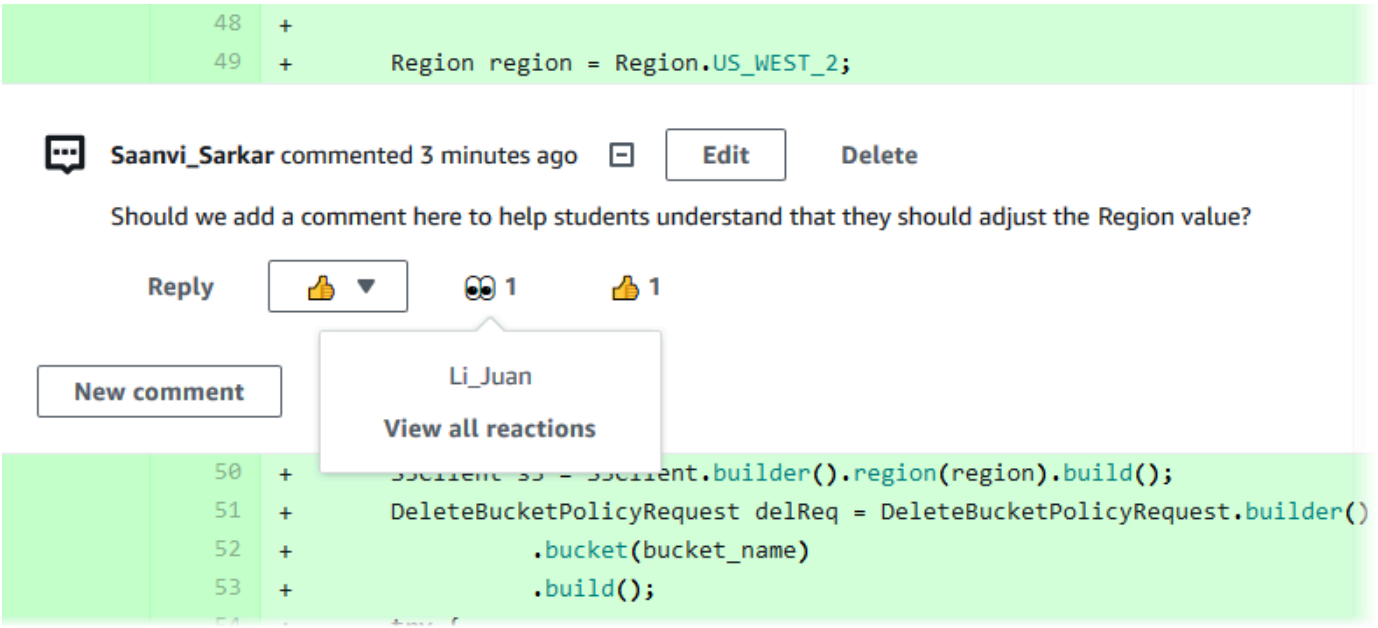


7. 若要回覆對遞交的評論，在 Changes (變更) 或 Activity (活動) 中，選擇 Reply (回覆)。您可以使用文字和表情符號回覆。



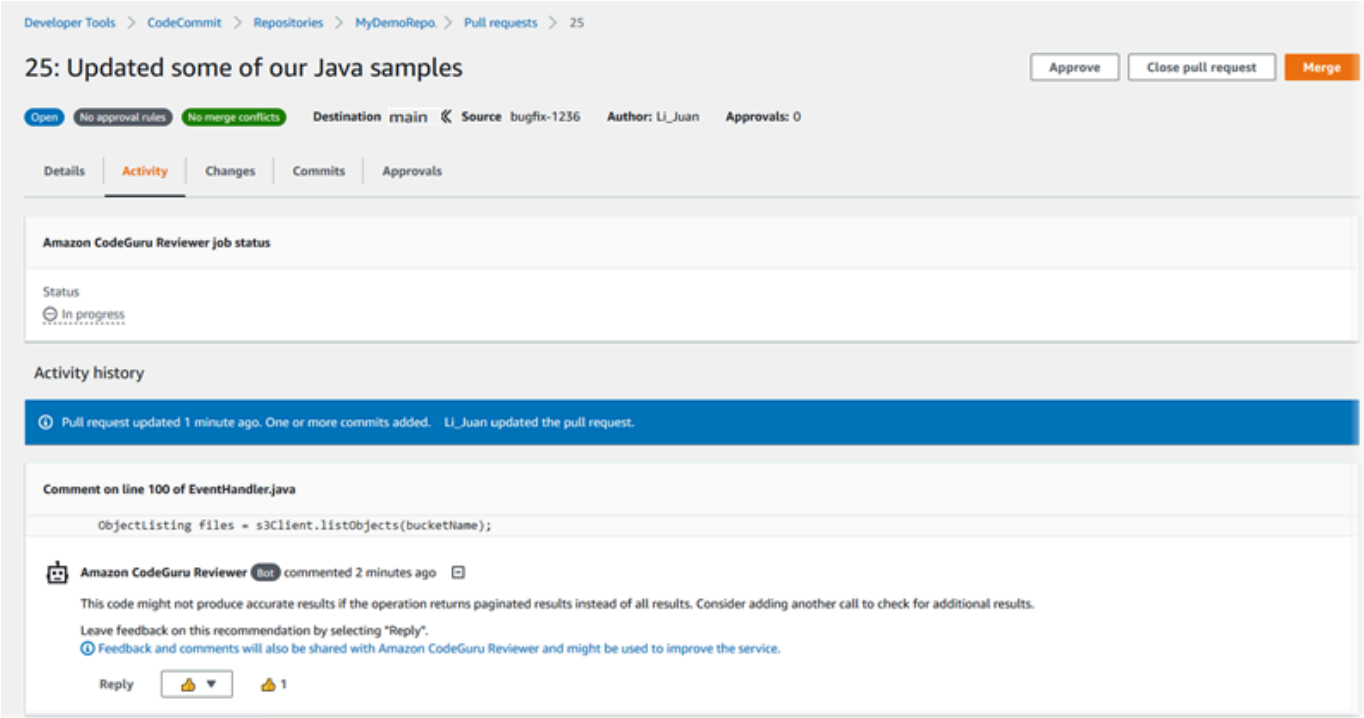
您可以通過選擇以特定表情符號反應回復來查看回應者的姓名。若要檢視所有表情符號反應和誰使用哪些表情符號回應的資訊，請選擇「檢視所有表情符 如果你已回覆留言的表情符號，你的回應會顯示在表情符號反應按鈕的圖示中。

> **Note**
>
> 控制台中顯示的反應計數在頁面加載時是準確的。如需表情符號反應計數的最新資訊，請重新整理頁面，或選擇 [檢視所有反應]。



8. (選擇性) 若要回覆 Amazon CodeGuru 審核者建立的建議，包括提供有關建議品質的意見反應，請選擇回覆。使用反應按鈕，提供有關您是否核准或拒絕建議的一般資訊。使用評論欄位提供有關反應的更多詳細資料。

> **Note**
>
> Amazon CodeGuru Reviewer 是一種自動化程式碼檢閱服務，它使用程式分析和機器學習來偵測常見問題，並建議 Java 或 Python 程式碼中的修正程式。
>
> - 如果您已將儲存庫與 Amazon CodeGuru 審核者相關聯、分析完成，且提取請求中的程式碼為 Java 或 Python 程式碼，則只有在您將儲存庫關聯時，才會看到 Amazon CodeGuru 審核者註解。如需詳細資訊，請參閱 將 AWS CodeCommit 儲存庫與 Amazon CodeGuru 審核者建立關聯或取消關聯。
> - 只有在提取請求的最 CodeGuru 新修訂版本中提出註解時，Amazon 審核者註解才會顯示在「變更」索引標籤中。它們始終顯示在「活動」標簽中。
> - 雖然您可以對 Amazon CodeGuru Reviewer 建議的任何可用表情符號反應做出回應，但只有豎起大拇指和豎起大拇指表情符號反應來評估推薦的有用性。

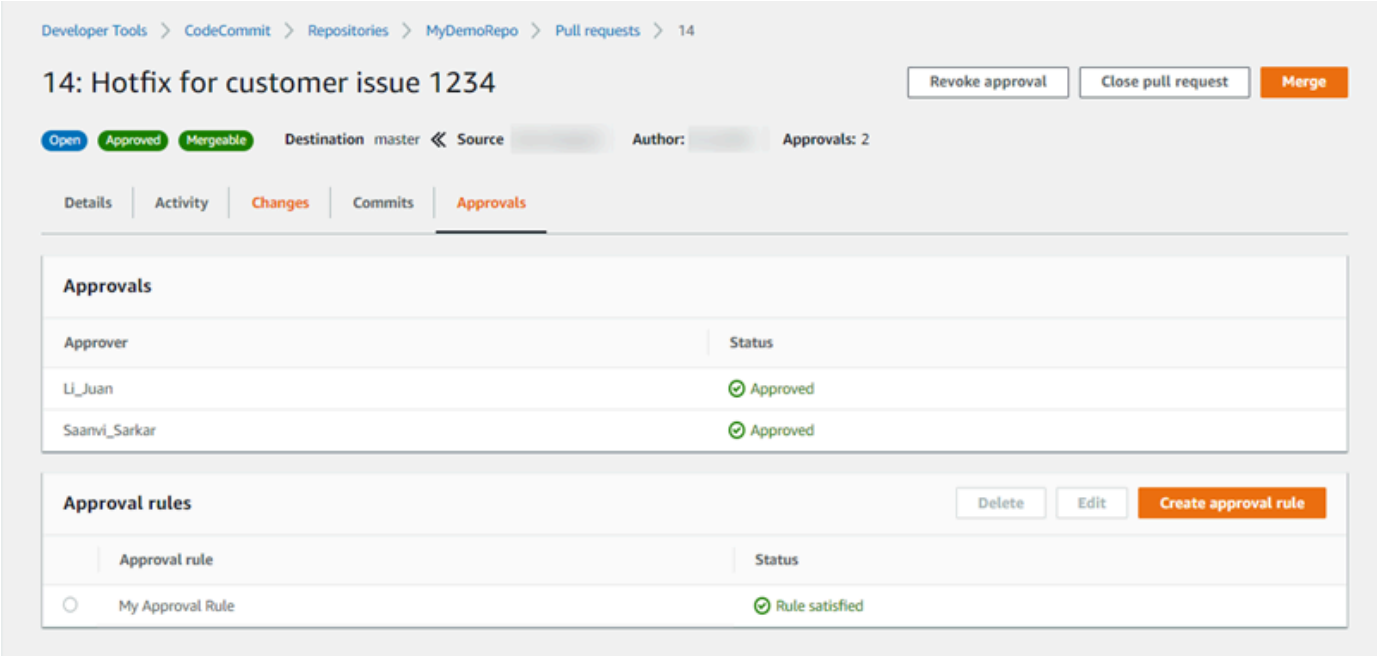


9. 若要核准提取請求中所做的變更，請選擇 Approve (核准)。

> **Note**
>
> 您無法核准您自己建立的提取請求。

您可以在 Approvals (核准) 中檢視核准、提取請求的核准規則、以及由核准規則範本建立的核准規則。如果您最後決定不要核准提取請求，您可以選擇 Revoke approval (撤銷核准)。

> **Note**
>
> 您只能核准或撤銷未結案提取請求的核准。如果提取請求的狀態為 Merged (已合併) 或 Closed (已關閉)，您無法核准或撤銷核准。



檢閱提取要求 (AWS CLI)

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

您可以使用下列 AWS CLI 指令來檢閱提取請求：

- post-comment-for-pull-request，將註解新增至提取請求
- get-comments-for-pull-request，以檢視在提取請求上留下的註解
- update-pull-request-approval-state，核准或撤銷提取要求的核准
- post-comment-reply，回覆提取要求中的註解

您也可以透過下列指令在提取要求中使用含有註解的表情符號：

- 若要使用表情符號回覆留言，請執行put-comment-reaction。
- 若要檢視表情符號對留言的反應，請執行get-comment-reactions。

使用檢 AWS CLI 閱 CodeCommit 儲存庫中的提取要求

1. 若要將評論新增至儲存庫中的提取請求，請執行 post-comment-for-pull-request 命令，指定：

- 提取請求的 ID (使用 --pull-request-id 選項)。
- 包含提取請求的儲存庫名稱 (使用 --repository-name 選項)。
- 將合併提取請求所在目的地分支中遞交的完整遞交 ID (使用 --before-commit-id 選項)。
- 當您張貼評論時，為提取請求分支目前頂端的來源分支中遞交的完整遞交 ID (使用 --after-commit-id 選項)。
- 唯一的用戶端產生冪等符記 (使用 --client-request-token 選項)。
- 評論的內容 (使用 --content 選項)。
- 可放置評論位置的清單位置相關資訊，包括：
  - 要比較的檔案名稱，包括其副檔名和子目錄 (如果有) (使用 filePath 屬性)。
  - 比較檔案中變更的行號 (使用 filePosition 屬性)。
  - 對變更的評論在來源與目的地分支之間比較中為 "before" 或 "after" (使用 relativeFileVersion 屬性)。

例如，使用此命令添加註釋 *##########################* 在一個名為的存儲庫中識別碼為 *47* 的提取請求中對 *ahs_count.py* 文件進行更改*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
```

如果此命令成功，則會產生類似如下的輸出：

```
{
        "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. 若要檢視提取請求的評論，請執行 get-comments-for-pull-request 命令，指定：

   - CodeCommit 存放庫的名稱 (含選--repository-name項)。
   - 系統產生的提取請求 ID (使用 --pull-request-id 選項)。
   - (選用) 用來傳回下一個批次結果的列舉符記 (使用 --next-token 選項)。
   - (選用) 一個非負整數，用來限制傳回的結果 (使用 --max-results 選項)。

   例如，使用這個命令來檢視 ID 為 42 的提取請求的評論。

   ```
   aws codecommit get-comments-for-pull-request --pull-request-id 42
   ```

   如果此命令成功，則會產生類似如下的輸出：

   ```
   {
      "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. 若要核准或撤銷提取請求的核准，請執行 update-pull-request-approval-state 命令，並指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 提取請求的修訂 ID (使用 --revision-id option)。您可以使用get-pull-request指令取得提取要求的目前修訂 ID。

- 您要套用的核准狀態 (使用 --approval-state) 選項。有效的核准狀態包括 APPROVE 和 REVOKE。

例如，使用此命令來核准 ID 為 *27* 且修訂 ID 為 *9f29d167EXAMPLE* 的提取請求。

```
aws codecommit update-pull-request-approval-state --pull-request-id 27 --revision-
id 9f29d167EXAMPLE --approval-state "APPROVE"
```

若成功，此命令不會傳回任何內容。

4. 若要對提取請求中的評論張貼回覆，請執行 post-comment-reply 命令，指定：

- 您要回覆的評論由系統產生的 ID (使用 --in-reply-to 選項)。
- 唯一的用戶端產生冪等符記 (使用 --client-request-token 選項)。
- 回覆的內容 (使用 --content 選項)。

例如，使用此命令添加回复 *## catch#########* 與系統生成的標識的 *AB* CD1234 示例 B5678EFGH 的評論。

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

如果此命令成功，則會產生類似如下的輸出：

```
{
    "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": []
    }
 }
```

更新提取請求

您可以通過推送提取請求的來源分支來使用進一步的代碼變更來更新提取請求。如需詳細資訊，請參閱在中創建提交 AWS CodeCommit。

您可以使用 AWS CodeCommit 主控台或 AWS CLI 來更新提取請求的標題或描述。您可能想要更新提取請求標題或描述，因為：

- 其他使用者無法了解描述，或原始標題會誤導。
- 您希望標頭或描述能反映對開啟中提取請求的來源分支進行的變更。

更新提取請求（主控台）

您可以使用 CodeCommit 主控台來更新 CodeCommit 儲存庫中提取請求的標題和描述。若要更新提取請求中的代碼，請推送提交至開啟中提取請求的來源分支。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇您要更新提取請求所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。選擇您想要更新的開啟中提取請求。
5. 在提取請求中，選擇 Details (詳細資訊)，然後選擇 Edit details (編輯詳細資訊)，以編輯標題或描述。

> **Note**
>
> 您無法更新已關閉或合併的提取請求的標題或描述。

更新提取請求 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

您可能也想要了解下列命令：

- update-pull-request-approval-state，核准或撤銷核准提取請求。
- create-pull-request-approval-rule，建立提取請求的核准規則。
- delete-pull-request-approval-rule，刪除提取請求的核准規則。

- 使用建立提交 AWS CLI或者使用 Git 客戶端創建提交來創建並推送其他代碼變更至開啟中提取請求的來源分支。

若要使用 AWS for WordPressAWS CLI來更新代碼提取儲存庫中的提取請求

1. 若要更新儲存庫中提取請求的標題，請執行 update-pull-request-title 命令，指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 提取請求的標題 (使用 --title 選項)。

   例如，若要更新 ID 為 *47* 提取請求的標題：

   ```
   aws codecommit update-pull-request-title --pull-request-id 47 --title
    "Consolidation of global variables - updated review"
   ```

   如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "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. 若要更新提取請求的描述，請執行 update-pull-request-description 命令，指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 描述 (使用 --description 選項)。

   例如，若要更新 ID 為 *47* 提取請求的描述：

```
aws codecommit update-pull-request-description --pull-request-id 47 --description
 "Updated the pull request to remove unused global variable."
```

   如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
   "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"
   }
}
```

編輯或刪除提取請求的核準規則

當您有提取請求的核准規則時，在符合條件之前，您無法合併該提取請求。您可以變更提取請求的核准規則，使之更容易滿足條件，或提高檢閱時的嚴謹性。您可以變更必須核准提取請求的使用者數目。您也可以在規則的使用者核准集區中，新增、移除或變更成員資格。最後，如果您不想再針對提取請求使用核准規則，可以刪除該核准規則。

> **Note**
>
> 您也可以覆寫提取請求的核准規則。如需詳細資訊，請參閱 提取請求的核可規則。

您可以使用 AWS CodeCommit 主控台或 AWS CLI 來編輯和刪除儲存庫的核准規則。

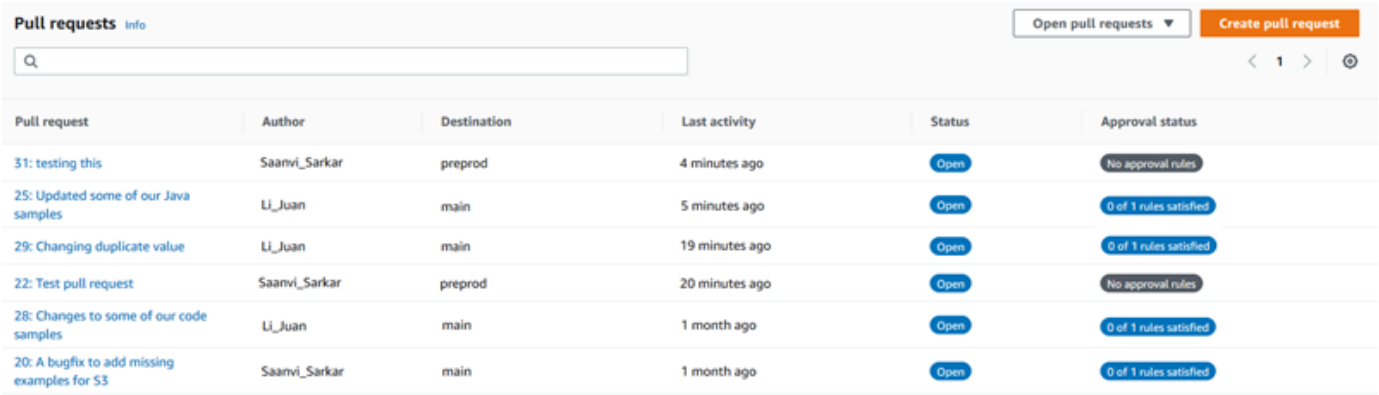
主題

- 編輯或刪除提取請求的核準規則（主控台）
- 編輯或刪除提取請求的核準規則 (AWS CLI)

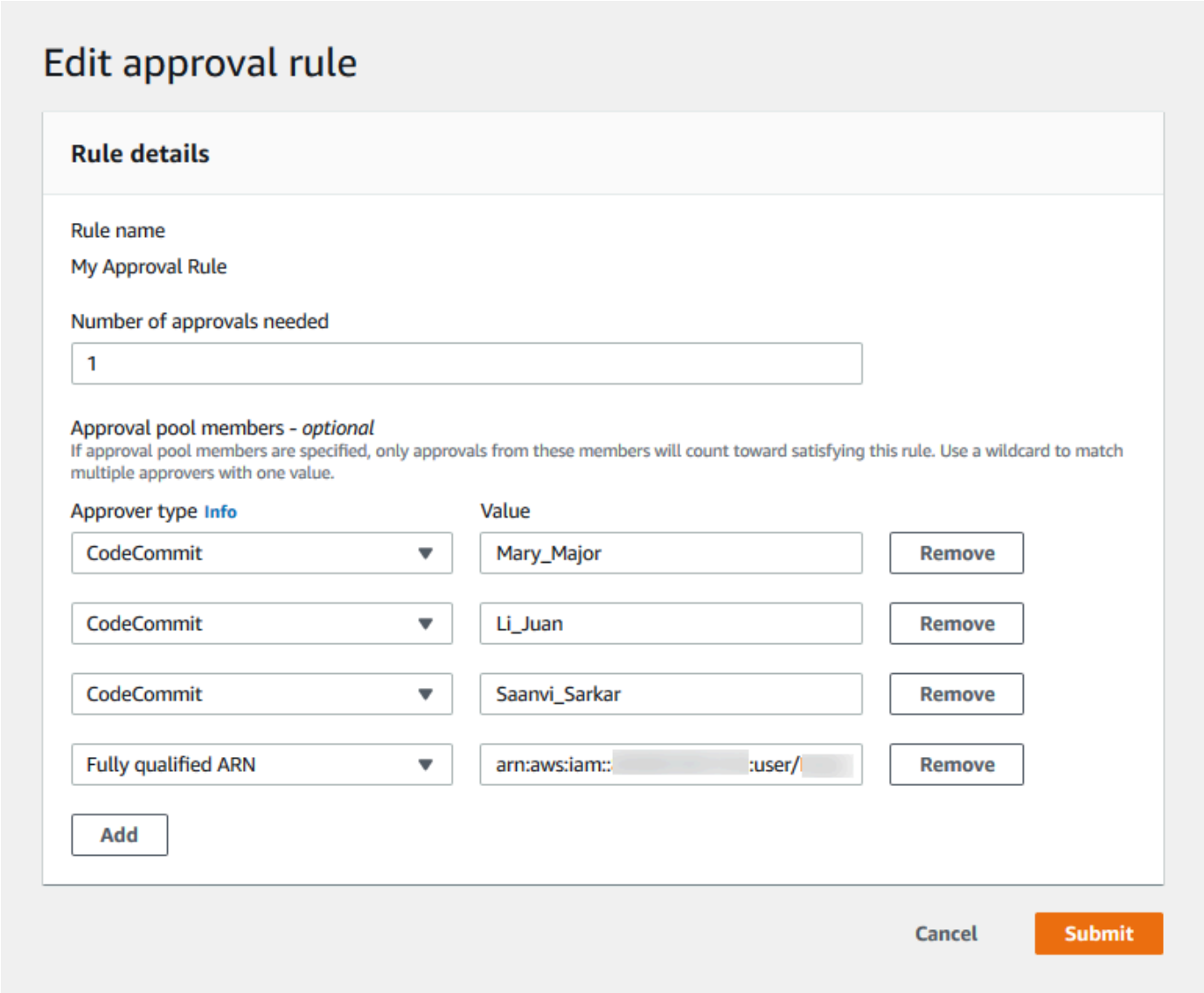
編輯或刪除提取請求的核準規則（主控台）

您可以使用 CodeCommit 主控台來編輯或刪除 CodeCommit 儲存庫中提取請求的核準規則。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱，您將於其中編輯或刪除提取請求的核准規則。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 選擇您要在哪個提取請求中編輯或刪除核准規則。您只能在未結案的提取請求中編輯及刪除核准規則。



5. 在提取請求中，選擇 Approvals (核准)，然後從清單中選擇要編輯或刪除的規則。執行下列任一步驟：
   - 如果想要編輯規則，請選擇 Edit (編輯)。
   - 如果要刪除規則，請選擇 Delete (刪除)，然後依照指示確認是否刪除規則。
6. 在 Edit approval rule (編輯核准規則) 中，變更規則，然後選擇 Submit (提交)。



7. 核准規則設定完成後，請選擇 Submit (提交)。

編輯或刪除提取請求的核準規則 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

您可以使用 AWS CLI 編輯核准規則的內容，以及刪除核准規則。

> **Note**
>
> 您可能也想要了解下列命令：
>
> - update-pull-request-approval-state，核准或撤銷核准提取請求。

- get-pull-request-approval-states，檢視提取請求的核准。
- evaluate-pull-request-approval-rules，判斷提取請求的核准規則是否已滿足條件。

若要使用 AWS for WordPressAWS CLI，編輯或刪除 CodeCommit 儲存庫中提取請求的核準規則

1. 若要編輯核准規則，請執行 update-pull-request-approval-rule-content 命令，並指定：

   - 提取請求的 ID (使用 --id 選項)。
   - 核准規則的名稱 (使用 --approval-rule-name 選項)。
   - 核准規則的內容 (使用 --approval-rule-content 選項)。

   此示例更新名為*#######*的提取請求，ID 為*27*。該規則需要批準池的一個用户批準，該池包括*123456789012*Amazon Web Services 帳户：

   ```
   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. 如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "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. 若要刪除核准規則，請執行 delete-pull-request-approval-rule 命令，並指定：

- 提取請求的 ID (使用 --id 選項)。
- 核准規則的名稱 (使用 --approval-rule-name 選項)。

例如，若要刪除名為*######*的提取請求，ID 為*15*：

```
aws codecommit delete-pull-request-approval-rule --pull-request-id 15 --approval-
rule-name "My Approval Rule"
```

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "approvalRuleId": "077d8e8a8-EXAMPLE"
}
```

提取請求的核可規則

在正常開發過程中，您希望在合併提取請求之前，使用者能夠符合核准規則的條件。不過，有時候您可能需要加速合併提取請求。例如，您可能想要將錯誤修正放入生產環境，但核准集區中沒有人可以核准提取請求。在這種情況下，您可以選擇覆寫提取請求的核准規則。您可以覆寫提取請求的所有核准規則，包括針對該提取請求而建立的核准規則，以及從核准規則範本產生的核准規則。您無法選擇性地覆寫特定的核准規則，只能覆寫所有規則。當您覆寫規則來忽略核准規則需求之後，您可以將提取請求合併到目的地分支。

當您覆寫提取請求的核准規則時，覆寫規則之使用者的相關資訊會記錄在提取請求的活動中。這樣您就可以回顧提取請求的歷史記錄，查看誰覆寫規則。如果提取請求仍未結案，您也可以選擇撤銷覆寫。合併提取請求之後，您就無法再撤銷覆寫。

主題

- 覆蓋批準規則（控制台）
- 覆蓋核可規則 (AWS CLI)

覆蓋批準規則（控制台）

在檢閱提取請求時，您可以在主控台覆寫提取請求的核准規則需求。如果改變主意，您可以撤銷覆寫，將會重新套用核准規則需求。只有在提取請求仍未結案時，您才能覆寫核准規則或撤銷覆寫。如果已合併或關閉，則無法變更其覆寫狀態。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。選擇您要覆寫核准規則需求或撤銷覆寫的提取請求。
4. 在 Approvals (核准) 索引標籤上，選擇 Override approval rules (覆寫核准規則)。將會忽略需求，按鈕文字會變更為 Revoke override (撤銷覆寫)。若要重新套用核准規則需求，請選擇 Revoke override (撤銷覆寫)。

覆蓋核可規則 (AWS CLI)

您可以使用 AWS CLI 來覆寫核准規則需求。也可以用來檢視提取請求的覆寫狀態。

覆寫提取請求的核准規則需求

1. 在終端機或命令列，執行 override-pull-request-approval-rules 命令，並指定：

   - 系統產生的提取請求 ID。
   - 提取請求的最新修訂 ID。若要檢視此資訊，請使用 get-pull-request。
   - 您要覆寫的狀態，OVERRIDE 或 REVOKE。REVOKE 狀態會移除 OVERRIDE 狀態，但不會儲存。

   例如，若要覆寫 ID 為 **34** 且修訂 ID 為 **927df8d8EXAMPLE** 的提取請求的核准規則：

   ```
   aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
   revision-id 927df8d8dEXAMPLE --override-status OVERRIDE
   ```

2. 如果成功，此命令不會傳回任何內容。
3. 若要撤銷 ID 為 **34** 且修訂 ID 為 **927df8d8EXAMPLE** 的提取請求的覆寫：

   ```
   aws codecommit override-pull-request-approval-rules --pull-request-id 34 --
   revision-id 927df8d8dEXAMPLE --override-status REVOKE
   ```

取得提取請求之覆寫狀態的相關資訊

1. 在終端機或命令列，執行 get-pull-request-override-state 命令，並指定：

   - 系統產生的提取請求 ID。
   - 提取請求的最新修訂 ID。若要檢視此資訊，請使用 get-pull-request。

   例如，若要檢視 ID 為 **34** 且修訂 ID 為 **927df8d8EXAMPLE** 的提取請求的覆寫狀態：

   ```
   aws codecommit get-pull-request-override-state --pull-request-id 34 --revision-
   id 927df8d8dEXAMPLE
   ```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "overridden": true,
       "overrider": "arn:aws:iam::123456789012:user/Mary_Major"
   }
   ```

合併中的提取請求AWS CodeCommit儲存庫

在檢閱過您的程式碼，且已滿足提取請求的所有核准規則 (如果有的話) 之後，您可以使用下列其中一種方式合併提取請求：

- 您可以在主控台使用其中一個可用的合併策略，將來源分支合併到目的地分支，這也會關閉提取請求。您也可以在主控台解決任何合併衝突。主控台會顯示訊息，指出提取請求是否可合併，或是否必須解決衝突。解決所有衝突後，如果您選擇 Merge (合併)，則會使用您選擇的合併策略來執行合併。向前快轉是預設的合併策略，這是 Git 的預設選項。根據來源和目的地分支中程式碼的狀態，該策略可能無法使用，但可能可以使用其他選項，例如 squash 或三向。
- 在 AWS CLI 中，您可以使用向前快轉、squash 或三向合併策略來合併和關閉提取請求。
- 在您的本機電腦上，您可以使用 git merge 命令來將來源分支合併到目的地分支，然後將合併的程式碼推送到目的地分支。這種方法有缺點，你應該仔細考慮。無論是否已滿足提取請求上的核准規則需求，都會避開這些控制而合併提取請求。如果使用向前快轉合併策略來合併提取請求，則合併並推送目的地分支也會自動關閉提取請求。這種方法的一個優點是git merge命令可讓您選擇 CodeCommit 主控台所沒有的合併選項或策略。如需 git merge 和合併選項的詳細資訊，請參閱 git-merge 或 Git 文件。

如果刪除提取請求的來源或目的地分支，CodeCommit 會自動關閉提取請求。

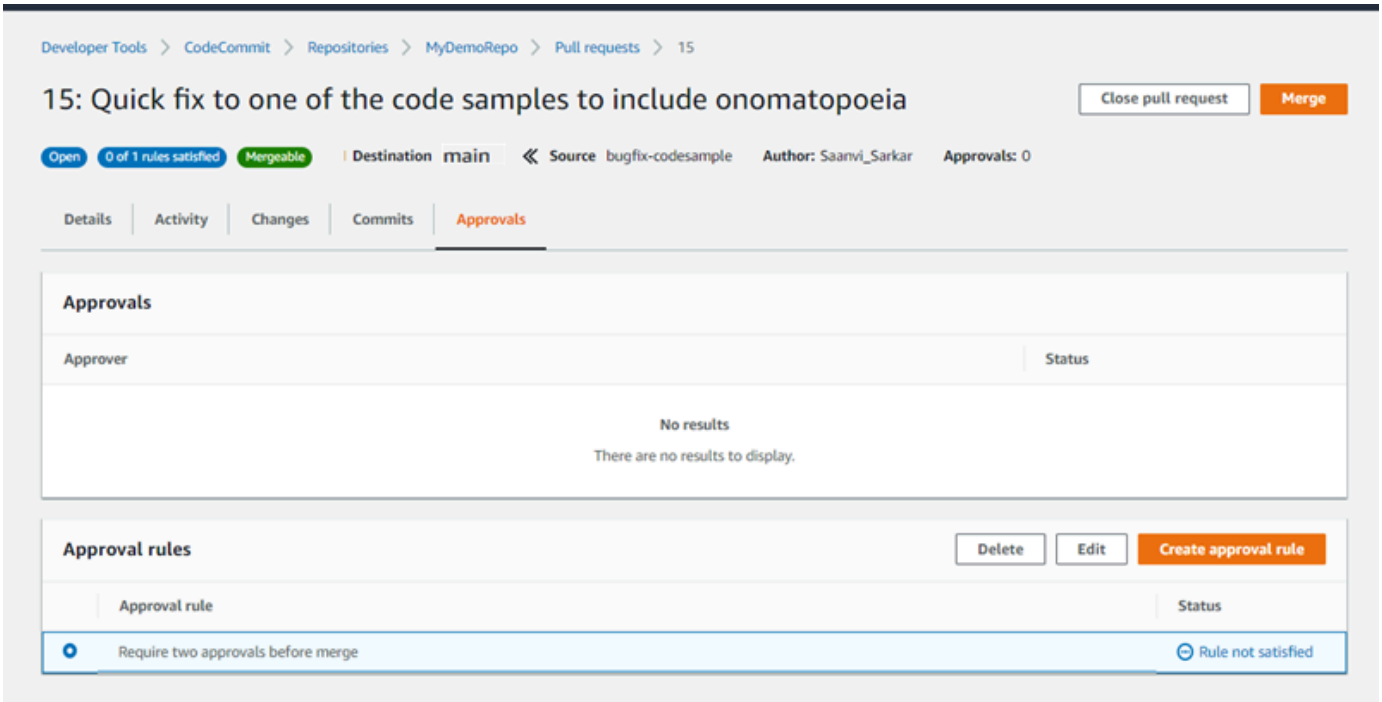
主題

合併提取請求（主控台）

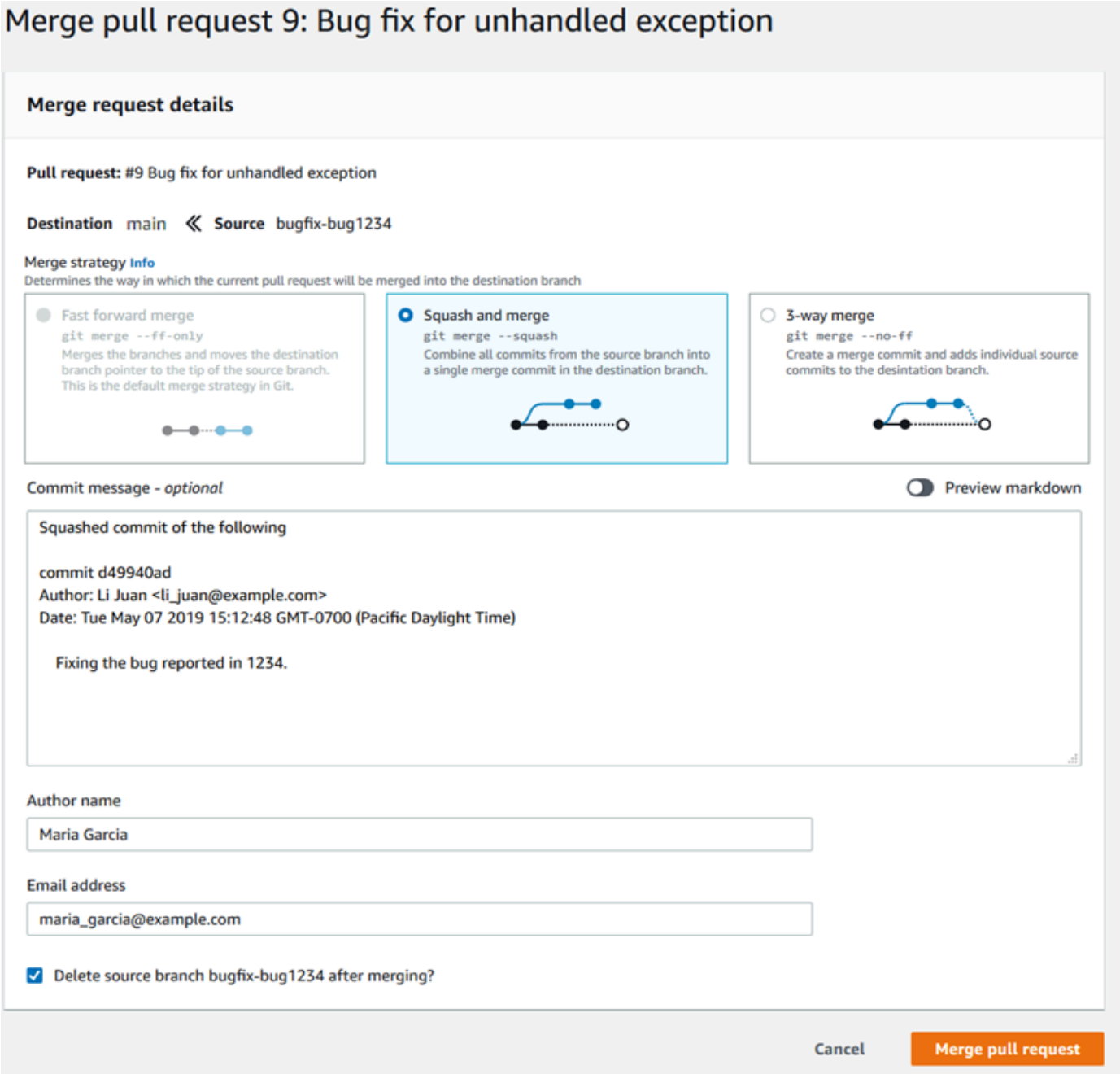
您可以使用 CodeCommit 主控台來合併 CodeCommit 儲存庫中的提取請求。提取請求的狀態變更為 Merged (已合併) 後，該提取請求就不會再出現在開啟的提取請求清單中。已合併的提取請求會歸類為已關閉。它無法變更回 Open (開啟)，但使用者仍然可以對變更進行評論並回覆評論。合併或關閉提取請求之後，您無法核准該請求、撤銷其核准，或覆寫已套用至提取請求的核准規則。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。選擇您想要合併的開啟中提取請求。
5. 在提取請求中，選擇 Approvals (核准)。檢閱核准者清單，並確認已滿足所有核准規則 (如果有的話) 的條件。如果一或多個核准規則的狀態為 Rule not satisfied (未滿足規則)，則您無法合併提取請求。如果沒有人核准提取請求，請考慮是否要合併，或是否要等待核准。

> **Note**
>
> 如果已針對提取請求建立核准規則，您可以編輯核准規則，或刪除核准規則以解除封鎖合併。如果核准規則是根據核准規則範本而建立，則您無法編輯或刪除核准規則。您只能選擇覆寫需求。如需詳細資訊，請參閱 提取請求的核可規則。



6. 選擇 Merge (合併)。
7. 在提取請求中，選擇可用的合併策略。無法套用的合併策略會呈現灰色。如果沒有可用的合併策略，您可以選擇在 CodeCommit 主控台中動解決衝突，或者使用 Git 用本機解決衝突。如需詳細資訊，請參閱 解決中提取請求的衝突AWS CodeCommit儲存庫。



- 向前快轉合併會將目的地分支的參考向前移動到來源分支的最新遞交。這是 Git 的預設行為 (如果可能)。不會建立合併遞交，但會保留來自來源分支的所有遞交歷史記錄，就好像發生在目的地分支一樣。在目的地分支歷史記錄的遞交視覺化檢視中，向前快轉合併不會顯示為分支合併，因為沒有建立合併遞交。來源分支的尖端會向前快轉到目的地分支的尖端。
- Squash 合併會建立一個遞交，其中包含來源分支的變更，並將該單一壓縮的遞交套用至目的地分支。預設情況下，該 squash 遞交的遞交訊息會包含來源分支中變更的所有遞交訊息。不會保留分支變更的個別遞交歷史記錄。這可協助簡化您的儲存庫歷史記錄，同時仍在目的地分支歷史記錄的 Commit visualizer 檢視中保留合併的圖形呈現方式。

- 三向合併會在目的地分支中建立合併的遞交訊息，同時也會在目的地分支的歷史記錄中保留來源分支中所做的個別遞交。這可協助保持儲存庫變更的完整歷史記錄。

8. 如果您選擇 squash 或三向合併策略，請檢閱自動產生的遞交訊息，如果您要變更資訊請加以修改。為遞交歷史記錄新增您的名稱和電子郵件地址。
9. (選擇性) 清除在合併時刪除來源分支的選項。預設是在合併提取請求時刪除來源分支。
10. 選擇 Merge pull request (合併提取請求) 完成合併。

合併提取請求 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

若要使用 AWS for WordPressAWS CLI合併 CodeCommit 儲存庫中的提取請求

1. 若要評估提取請求的所有核准規則是否已滿足，且已可合併，請執行 evaluate-pull-request-approval-rules 命令，並指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 提取請求的修訂 ID (使用 --revision-id option)。您可以使用 get-pull-request 命令取得提取請求的目前修訂 ID。

   例如，若要評估 ID 為 *27* 且修訂 ID 為 *9f29d167EXAMPLE* 的提取請求的核准規則狀態：

   ```
   aws codecommit evaluate-pull-request-approval-rules --pull-request-id 27 --
   revision-id 9f29d167EXAMPLE
   ```

   如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "evaluation": {
           "approved": false,
           "approvalRulesNotSatisfied": [
               "Require two approved approvers"
           ],
           "overridden": false,
           "approvalRulesSatisfied": []
       }
   }
   ```

> **Note**
>
> 此輸出指出提取請求無法合併，因為尚未滿足核准規則的需求。若要合併此提取請求，您可以請檢閱者核准它，以符合規則的條件。根據您的許可和規則的建立方式，您可能還可以編輯、覆寫或刪除規則。如需詳細資訊，請參閱 檢閱提取請求、提取請求的核可規則 及 編輯或刪除提取請求的核準規則。

2. 若要使用向前快轉合併策略來合併和關閉提取請求，請執行 merge-pull-request-by-fast-forward 命令，指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 來源分支頂端的完整遞交 ID (使用 --source-commit-id 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。

   例如，若要合併和關閉 ID 為*47*和源提交 ID*##*存儲庫中名為*MyDemoRepo*：

```
aws codecommit merge-pull-request-by-fast-forward --pull-request-id 47 --source-
commit-id 99132ab0EXAMPLE --repository-name MyDemoRepo
```

   如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要使用 squash 合併策略來合併和關閉提取請求，請執行 merge-pull-request-by-squash 命令，指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 來源分支頂端的完整遞交 ID (使用 --source-commit-id 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。
   - 您想要使用的衝突細節層次 (使用 --conflict-detail-level 選項)。如果未指定，會使用預設的 **FILE_LEVEL**。
   - 您要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。如果未指定，此值預設為 NONE，且必須手動解決衝突。
   - 要包含的遞交訊息 (使用 --commit-message 選項)。
   - 用於遞交的名稱 (使用 --author-name 選項)。
   - 用於遞交的電子郵件地址 (使用 --email 選項)。
   - 是否保留任何空資料夾 (使用 --keep-empty-folders 選項)。

   下列範例合併和關閉 ID 為*47*和源提交 ID*##*存儲庫中名為*MyDemoRepo*。使用的衝突細節為 LINE_LEVEL，衝突解決策略為 ACCEPT_SOURCE：

```
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，這個命令會產生與透過向前快轉合併相同的輸出類型，輸出類似於以下內容：

```
{
    "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. 若要使用三向合併策略來合併和關閉提取請求，請執行 merge-pull-request-by-three-way 命令，指定：

   - 提取請求的 ID (使用 --pull-request-id 選項)。
   - 來源分支頂端的完整遞交 ID (使用 --source-commit-id 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。
   - 您想要使用的衝突細節層次 (使用 --conflict-detail-level 選項)。如果未指定，會使用預設的 **FILE_LEVEL**。
   - 您要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。如果未指定，此值預設為 NONE，且必須手動解決衝突。
   - 要包含的遞交訊息 (使用 --commit-message 選項)。
   - 用於遞交的名稱 (使用 --author-name 選項)。
   - 用於遞交的電子郵件地址 (使用 --email 選項)。
   - 是否保留任何空資料夾 (使用 --keep-empty-folders 選項)。

   下列範例合併和關閉 ID 為*47*和源提交 ID*##*存儲庫中名為*MyDemoRepo*。衝突細節和衝突解決策略都使用預設選項：

   ```
   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，這個命令會產生與透過向前快轉合併相同的輸出類型，類似於以下內容：

   ```
   {
       "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"
    }
}
```

解決中提取請求的衝突AWS CodeCommit儲存庫

如果您的提取請求有衝突且無法合併，您可以使用多種方式來嘗試解決衝突：

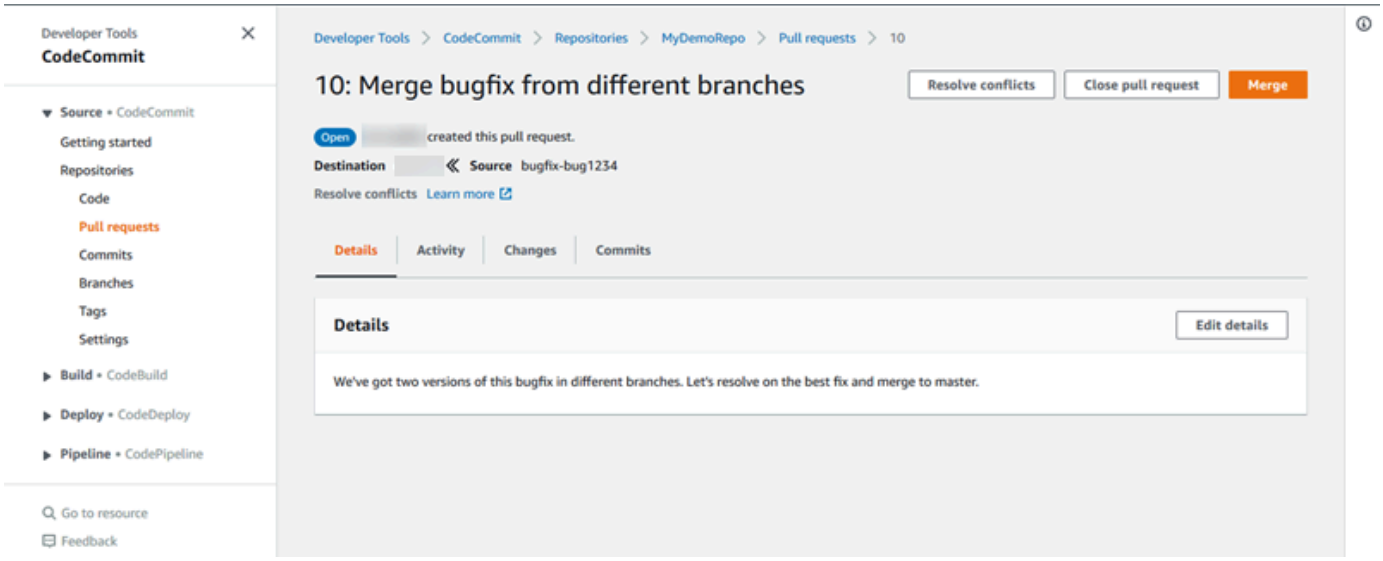
- 在您的本機電腦，可以使用 git diff 命令來尋找兩個分支之間的衝突並進行變更來解決衝突。您也可以使用不同的工具或其他軟體，協助您尋找和解決差異。解決問題並感到滿意後，您可以推送包含已解決衝突之變更的來源分支，來更新提取請求。如需 git diff 和 git difftool 的詳細資訊，請參閱 Git 文件。
- 在主控台，您可以選擇 Resolve conflicts (解決衝突)。這會開啟純文字編輯器，以類似 git diff 命令的方式來顯示衝突。您可以手動檢閱每個檔案中的衝突、進行變更，然後以您的變更來更新提取請求。
- 在 AWS CLI 中，您可以使用 AWS CLI 取得合併衝突的相關資訊，並建立未參照的合併遞交來測試合併。

主題

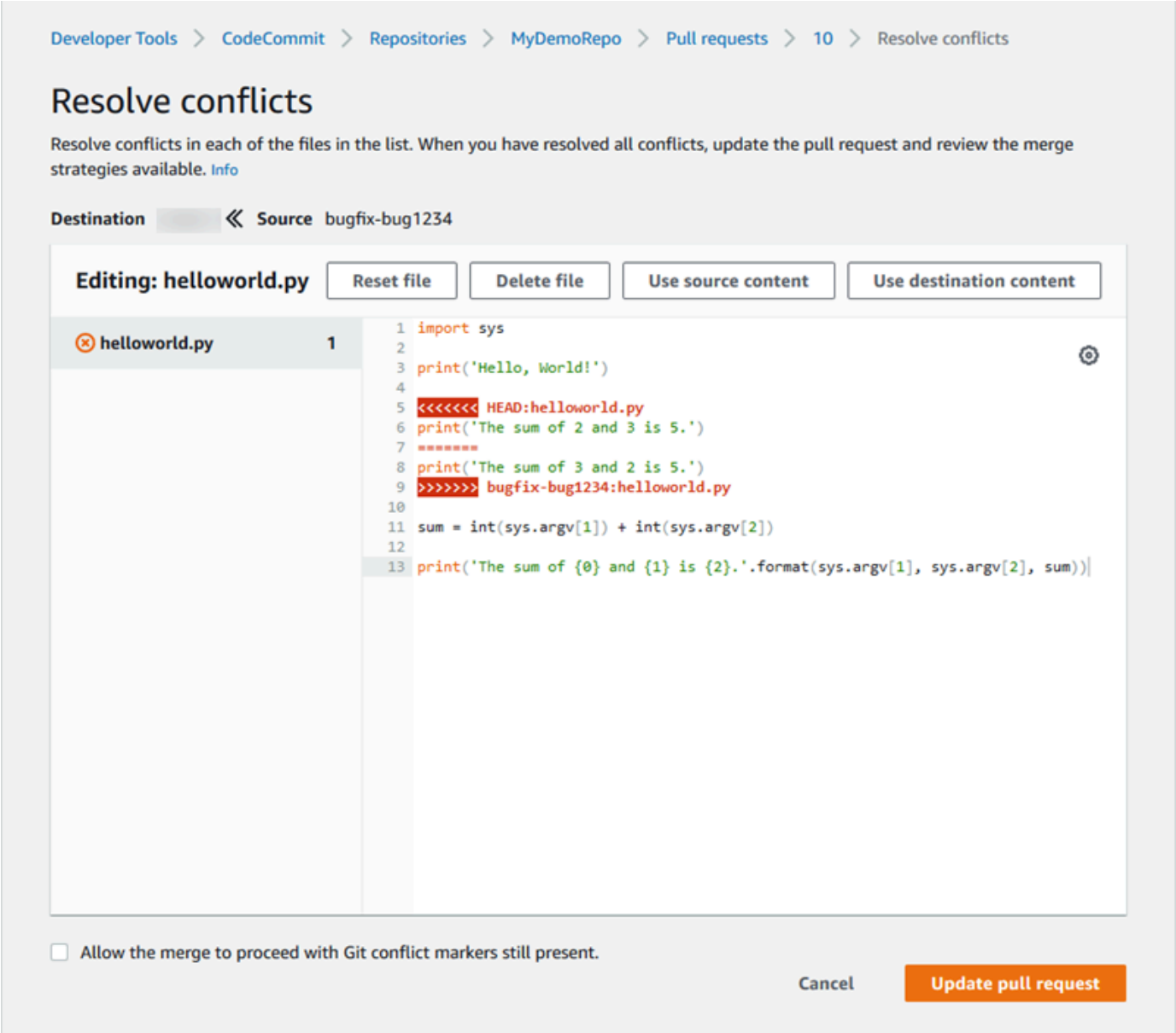
解決拉取請求中的衝突（控制台）

您可以使用 CodeCommit 主控台，解決 CodeCommit 儲存庫中提取請求的衝突。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。選擇您想要合併但包含衝突的開啟中提取請求。
5. 在提取請求中，選擇 Resolve conflicts (解決衝突)。提取請求中有必須解決然後才能合併的衝突時，才會顯示此選項。



6. 會開啟衝突解決視窗，列出有必須解決之衝突的每個檔案。選擇清單中的每個檔案來檢閱衝突，並進行任何必要的變更，直到所有衝突都解決。



- 您可以選擇使用來源檔案內容、目的地檔案內容，或者如果檔案不是二進位檔案，則手動編輯檔案內容來讓其只包含您想要的變更。標準 git diff 標記用來顯示檔案中目的地 (HEAD) 和來源分支之間的衝突。
- 如果檔案是二進位檔案、Git 子模組，或是有檔案/資料夾名稱衝突，您必須選擇使用來源檔案或目的地檔案來解決衝突。您無法在 CodeCommit 主控台中查看或編輯二進位檔案。
- 如果有檔案模式衝突，您可透過選擇來源檔案的檔案模式和目的地檔案的檔案模式，來查看可解決該衝突的選項。
- 如果您決定要放棄對檔案的變更並恢復為衝突狀態，請選擇 Reset file (重設檔案)。這可讓您以不同的方法來解決衝突。

7. 對所做的變更感到滿意之後，請選擇 Update pull request (更新提取請求)。

> **Note**
>
> 您必須先解決檔案中的所有衝突，然後才能使用您的變更來成功更新提取請求。

8. 提取請求會根據您的變更加以更新並變成可合併。您可透過該合併頁面。您可以選擇立即合併提取請求，也可以返回提取請求清單。

解決提取請求中的衝突（AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

沒有單一AWS CLI命令可讓您解決提取請求中的衝突和合併該請求。不過，您可以使用個別命令來探索衝突、嘗試解決問題，並測試提取請求是否可以合併。您可以使用：

- get-merge-options，來了解針對兩個遞交指標之間的合併，可使用哪些合併選項。
- get-merge-conflicts，傳回在兩個遞交指標之間的合併中具有合併衝突的檔案清單。
- batch-describe-merge-conflicts，使用指定的合併策略，取得在兩個遞交之間的合併中檔案的所有合併衝突相關資訊。
- describe-merge-conflicts，使用指定的合併策略，取得在兩個遞交之間特定檔案的合併衝突詳細資訊。
- create-unreferenced-merge-commit，使用指定的合併策略，測試合併兩個遞交指標的結果。

1. 為了探索兩個遞交指標之間的合併有哪些可用的合併選項，請執行 get-merge-options 命令，指定：

   - 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
   - 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。
   - (選用) 要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。
   - (選用) 所需的任何衝突的細節層次 (使用 --conflict-detail-level 選項)。

   例如，要確定可用於合併名為*######*，目標分支名為*##*存儲庫中名為*MyDemoRepo*：

```
aws codecommit get-merge-options --source-commit-specifier bugfix-1234 --
destination-commit-specifier main --repository-name MyDemoRepo
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "mergeOptions": [
        "FAST_FORWARD_MERGE",
        "SQUASH_MERGE",
        "THREE_WAY_MERGE"
    ],
    "sourceCommitId": "d49940adEXAMPLE",
    "destinationCommitId": "86958e0aEXAMPLE",
    "baseCommitId": "86958e0aEXAMPLE"
}
```

2. 若要取得包含兩個遞交指標間合併之合併衝突的檔案清單，請執行 get-merge-conflicts 命令，指定：

- 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
- 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。
- 儲存庫的名稱 (使用 --repository-name 選項)。
- 您想要使用的合併選項 (使用 --merge-option 選項)。
- (選用) 所需的任何衝突的細節層次 (使用 --conflict-detail-level 選項)。
- (選用) 要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。
- (選用) 傳回的含衝突檔案數目上限 (使用 --max-conflict-files 選項)。

例如，若要在名為 MyDemoRepo 的儲存庫中，使用三向合併策略，取得包含名為 master 之來源分支和名為 main 之目的地分支之間合併衝突的檔案清單：

```
aws codecommit get-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-option
 THREE_WAY_MERGE --repository-name MyDemoRepo
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要取得兩個遞交指標之間的合併中，所有檔案或部分檔案之合併衝突的相關資訊，請執行 batch-describe-merge-conflicts 命令，指定：

- 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
- 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。
- 您想要使用的合併選項 (使用 --merge-option 選項)。
- 儲存庫的名稱 (使用 --repository-name 選項)。
- (選用) 要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。
- (選用) 所需的任何衝突的細節層次 (使用 --conflict-detail-level 選項)。
- (選用) 傳回的合併 hunk 數目上限 (使用 --max-merge-hunks 選項)。
- (選用) 傳回的含衝突檔案數目上限 (使用 --max-conflict-files 選項)。
- (選用) 目標檔案路徑，用來描述衝突 (使用 --file-paths 選項)。

例如，要確定合併名為*###-#####*，目標分支名為*##*使用*####*策略在名為*MyDemoRepo*：

```
aws codecommit batch-describe-merge-conflicts --source-commit-specifier feature-
randomizationfeature --destination-commit-specifier main --merge-
option THREE_WAY_MERGE --repository-name MyDemoRepo
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要取得兩個遞交指標之間的合併中，特定檔案之任何合併衝突的相關資訊，請執行 describe-merge-conflicts 命令，指定：

   - 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
   - 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。

- 您想要使用的合併選項 (使用 --merge-option 選項)。
- 目標檔案路徑，用來描述衝突 (使用 --file-path 選項)。
- 儲存庫的名稱 (使用 --repository-name 選項)。
- (選用) 要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。
- (選用) 所需的任何衝突的細節層次 (使用 --conflict-detail-level 選項)。
- (選用) 傳回的合併 hunk 數目上限 (使用 --max-merge-hunks 選項)。
- (選用) 傳回的含衝突檔案數目上限 (使用 --max-conflict-files 選項)。

例如，要確定名為*## .md*的源分支中名為*###-#####*，目標分支名為*##*使用*####*策略在名為*MyDemoRepo*：

```
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
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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. 若要建立未參照遞交來代表合併兩個遞交指標的結果，請執行 create-unreferenced-merge-commit 命令，指定：

   - 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
   - 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。
   - 您想要使用的合併選項 (使用 --merge-option 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。
   - (選用) 要使用的衝突解決策略 (使用 --conflict-resolution-strategy 選項)。
   - (選用) 所需的任何衝突的細節層次 (使用 --conflict-detail-level 選項)。
   - (選用) 要包含的遞交訊息 (使用 --commit-message 選項)。

- (選用) 用於遞交的名稱 (使用 --name 選項)。
- (選用) 用於遞交的電子郵件地址 (使用 --email 選項)。
- (選用) 是否保留任何空資料夾 (使用 --keep-empty-folders 選項)。

例如，要確定合併名為*######*，目標分支名為*##*使用名為*MyDemoRepo*：

```
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."
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

關閉中的提取請求AWS CodeCommit儲存庫

如果您想關閉提取請求而不合併程式碼，您有多種方式可以使用：

- 在主控台中，您可以關閉提取請求而不需合併程式碼。如果您想要使用 git merge 命令來手動合併分支，或如果提取請求來源分支中的程式碼，不是您要合併到目的地分支的程式碼，則可能想要這樣做。
- 您可以刪除拉取請求中指定的源分支。如果刪除提取請求的來源或目的地分支，CodeCommit 會自動關閉提取請求。
- 在 AWS CLI 中，您可以將提取請求的狀態從 OPEN 更新為 CLOSED。這將會關閉提取請求而不合併程式碼。

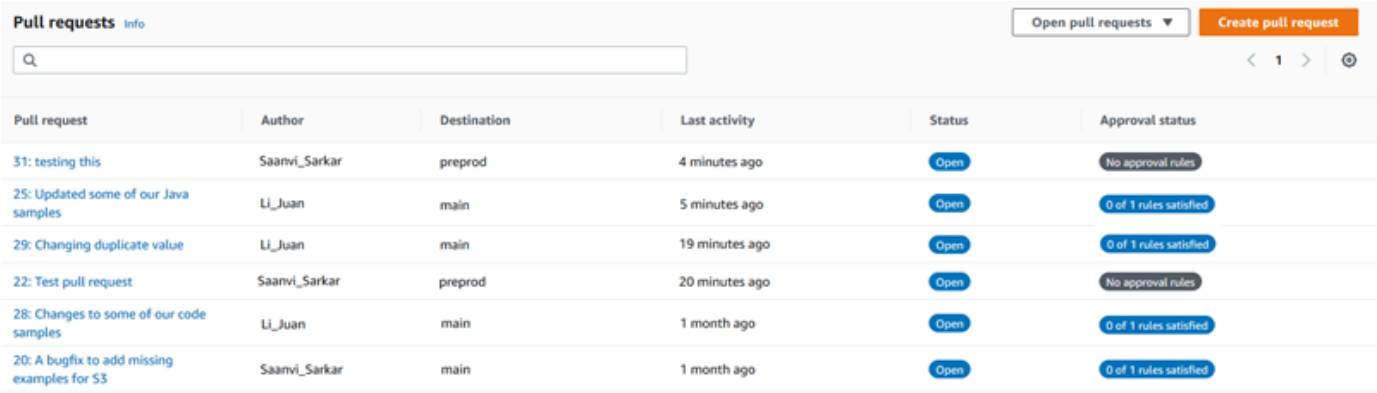
主題

- 關閉提取請求
- 關閉提取請求 (AWS CLI)

關閉提取請求

您可以使用 CodeCommit 主控台來關閉 CodeCommit 儲存庫中的提取請求。提取請求的狀態變更為 Closed (已關閉) 後，就無法再變更回 Open (開啟)，但使用者仍然可以對變更進行評論並回覆評論。

1. 開啟位於的 CodeCommit 主控台https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在 Repositories (儲存庫) 中，選擇儲存庫的名稱。
3. 在導覽窗格中，選擇 Pull requests (提取請求)。
4. 依預設，會顯示所有開啟的提取請求清單。選擇您想要關閉的開啟中提取請求。



5. 在提取請求中，選擇 Close pull request (關閉提取請求)。此選項會關閉提取請求，而不會嘗試將來源分支合併到目的地分支。此選項不會提供方式在關閉提取請求時刪除來源分支，但您可以在請求關閉之後自行執行該動作。

關閉提取請求 (AWS CLI)

使用AWS CLI命令，請安裝AWS CLI。如需詳細資訊，請參閱 命令列參考。

若要使用 AWS for WordPressAWS CLI關閉 CodeCommit 取請求

- 若要將儲存庫中提取請求的狀態從 OPEN 更新為 CLOSED，請執行 update-pull-request-status 命令，指定：
  - 提取請求的 ID (使用 --pull-request-id 選項)。
  - 提取請求的狀態 (使用 --pull-request-status 選項)。

  例如，若要更新 ID 提取請求的狀態*42*設置為*##*在 CodeCommit 庫中名為MyDemoRepo：

```
aws codecommit update-pull-request-status --pull-request-id 42 --pull-request-
status CLOSED
```

如果此命令成功執行，您會看到類似如下的輸出產生：

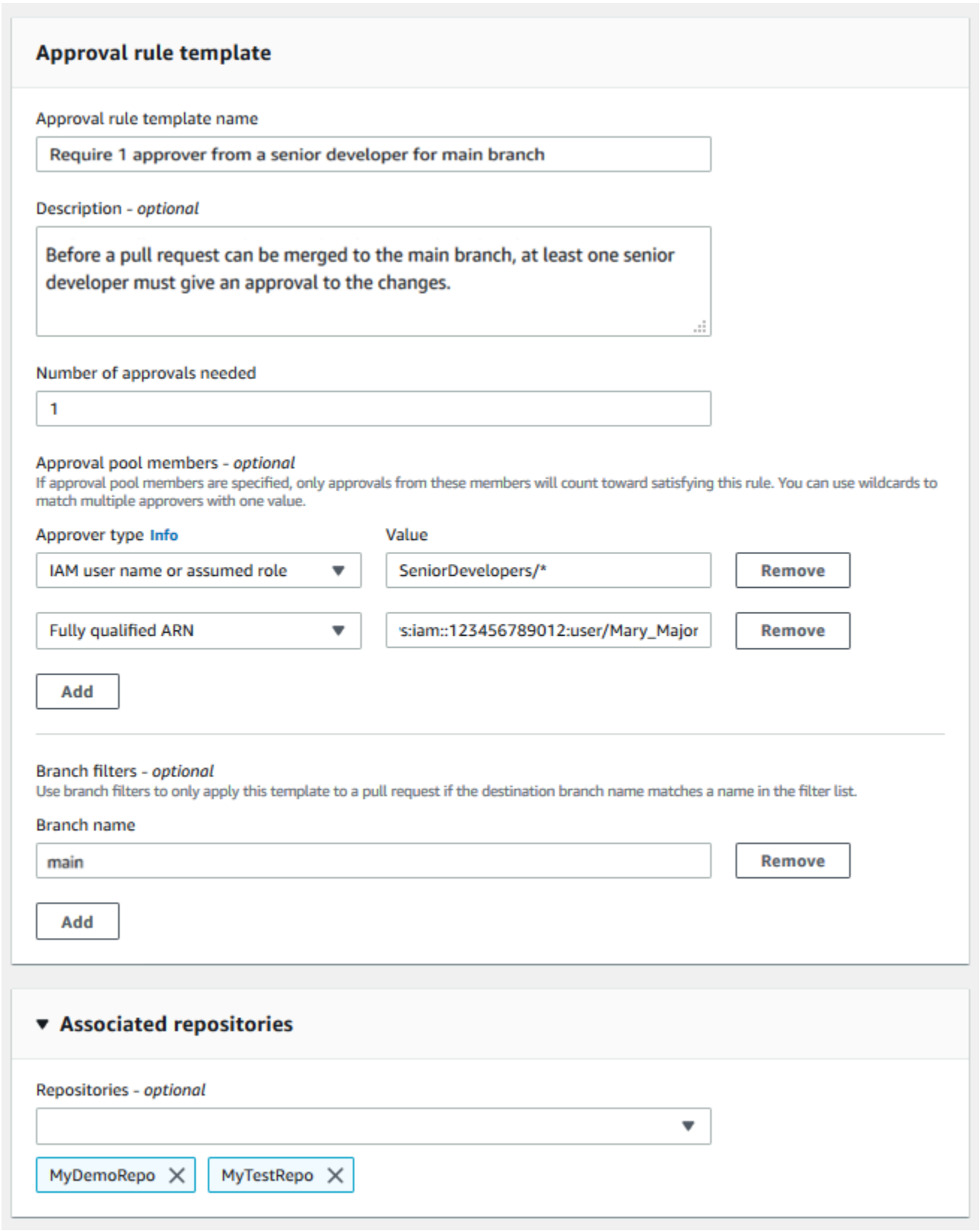
```
{
    "pullRequest": {
        "approvalRules": [
            {
                "approvalRuleContent": "{\"Version\": \"2018-11-08\",\"Statements
\": [{\"Type\": \"Approvers\",\"NumberOfApprovalsNeeded\": 2,\"ApprovalPoolMembers
\": [\"arn:aws:sts::123456789012:assumed-role/CodeCommitReview/*\"]}]}",
                "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"
    }
```

```
}
```

使用核准規則範本

您可以建立提取請求的核准規則。若要自動將核准規則套用至儲存庫中建立的部分或所有提取要求，請使用核准規則範本。核准規則範本協助您自訂跨儲存庫的開發工作流程，讓不同分支能具有適當等級的核准和控制。您可以為生產和開發分支定義不同的規則。每次建立符合規則條件的提取請求時，就會套用這些規則。如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

您可以將核准規則範本與建立核准規則範本的一或多個儲存庫產生關聯。AWS 區域當範本與儲存庫建立關聯後，在該儲存庫中建立提取請求時，就會自動為提取請求建立核准規則。就像單一核准規則一樣，核准規則範本定義核准規則結構，包括所需的核准數目，以及必須發出核准的選擇性使用者集區。與核准規則不同，您也可以定義目的地參考 (一或多個分支)，也稱為「分支篩選條件」。如果您定義目的地參考，則只有當提取請求的目的地分支名稱符合範本中指定的分支名稱 (目的地參考) 時，才會為提取請求建立規則。因此，假設您指定 **refs/heads/main** 作為目的地參考，則只有在目的地分支為 `main` 時，範本中定義的核准規則才會套用至提取請求。


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

主題

建立核准規則範本

您可以建立一或多個核准規則範本，以協助您自訂跨儲存庫的開發工作流程。您可以建立多個範本，以設定自動建立核准規則，讓不同分支具有適當等級的核准和控制。例如，您可以為生產和開發分支建立不同的範本，並將這些範本套用至一或多個儲存庫。當使用者在這些儲存庫中建立提取請求時，將根據這些範本來評估請求。如果請求符合所套用範本中的條件，則會為提取請求建立核准規則。

您可以使用主控台或 AWS CLI 建立核准規則範本。如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

主題

建立核准規則範本 (主控台)

依預設，核准規則範本不會與任何儲存庫相關聯。您可以在建立範本時建立範本與一或多個儲存庫之間的關聯，或者稍後再新增關聯。

建立核准規則範本 (主控台)

1. 請在以下位置開啟CodeCommit主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 選擇 Approval rule templates (核准規則範本)，然後選擇 Create template (建立範本)。
3. 在 Approval rule template name (核准規則範本名稱) 中，以描述性名稱來命名範本，讓您明白其用途。例如，如果一群資深開發人員中必須有一人核准提取請求，才能合併提取請求，您可以將規則命名為 **`Require 1 approver from a senior developer`**。
4. (選擇性) 在 Description (描述) 中，提供此範本的用途描述。這可以幫助其他人決定此範本是否適用於其儲存庫。

5. 在 Number of approvals needed (需要的核准數目) 中，輸入您要的數目。預設為 1。
6. (選擇性) 如果提取請求的核准必須來自特定使用者群組，請在 Approval rule members (核准規則成員) 中選擇 Add (新增)。在 Approver type (核准者類型) 中，選擇以下其中一項：

   - IAM 使用者名稱或假定角色：此選項會為您用來登入的帳戶預先填入 Amazon Web Services 帳戶 ID，而且只需要輸入名稱。它可用於名稱與所提供名稱相符的 IAM 使用者和聯合存取使用者。此選項非常強大，提供很大的靈活性。例如，如果您選擇此選項並使用 Amazon Web Services 帳戶 123456789012 登入，並指定**Mary_Major**，則以下所有內容都會計為來自該使用者的核准：

     - 帳戶中的 IAM 使用者 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中識別為瑪麗 () 的聯合身分使用者 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非您包含萬用字元 (`*Mary_Major`)，否則此選項無法識別擔任角色 **CodeCommitReview** 且角色工作階段名稱為 Mary_Major (`arn:aws:sts::123456789012:assumed-role/CodeCommitReview/Mary_Major`) 的某人的作用中工作階段。您也可以明確指定角色名稱 (`CodeCommitReview/Mary_Major`)。
   - 完全合格的 ARN：此選項可讓您指定 IAM 使用者或角色的完整 Amazon 資源名稱 (ARN)。此選項也可支援其他 AWS 服務所使用的擔任角色，例如 AWS Lambda 和 AWS CodeBuild。針對擔任的角色，ARN 格式應為 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`RoleName`* (若為角色) 和 `arn:aws:sts::`*`AccountID`*`:assumed-role/`*`FunctionName`* (若為函數)。

   如果您選擇 IAM 使用者名稱或假定的角色做為核准者類型，請在值中輸入 IAM 使用者或角色的名稱或使用者或角色的完整 ARN。再次選擇 Add (新增)，以新增更多使用者或角色，直到核准要計入所需核准數目中的所有使用者或角色，都已新增為止。

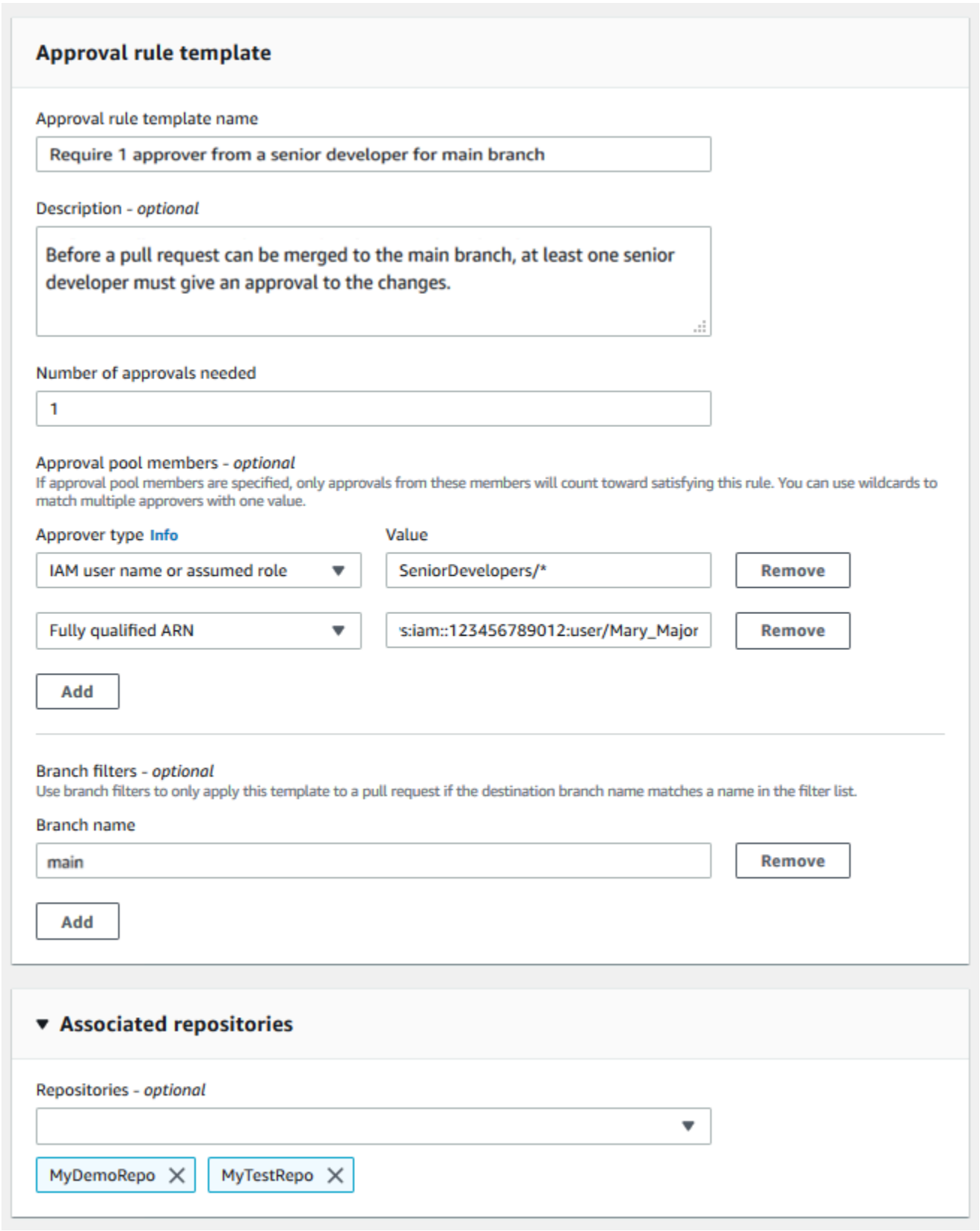
   這兩種核准者類型都允許您在值中使用萬用字元 (*)。例如，如果您選擇 IAM 使用者名稱或假定的角色選項，並指定**CodeCommitReview/***，則所有擔任該角色的使用者**CodeCommitReview**都會計入核准集區中。個別角色工作階段名稱計入所需的核准者數目中。如此一來，Mary_Major 和 Li_Juan 在登入並擔任 `CodeCommitReview` 時都算作核准。如需 IAM ARN、萬用字元和格式的詳細資訊，請參閱 IAM 識別符。

   > **Note**
   >
   > 核准規則不支援跨帳戶核准。

7. (選擇性) 在 Branch filters (分支篩選條件) 中，輸入目的地分支名稱，以用來篩選能否建立核准規則。例如，如果您指定 `main`，則只有當提取請求的目的地分支是名為 `main` 的分支時，才會為關聯儲存庫中的提取請求建立核准規則。您可以在分支名稱中使用萬用字元 (*)，將核准規則套用至符合萬用字元案例的所有分支名稱。不過，您不能在分支名稱的開頭使用萬用字元。您最多可以指定 100 個分支名稱。如果您未指定任何篩選條件，範本會套用至相關聯儲存庫中的所有分支。
8. (選擇性) 在關聯的儲存庫的儲存庫清單中，選擇您要與AWS 區域此核准規則產生關聯的儲存庫。

> **Note**
>
> 您可以選擇在建立範本之後建立儲存庫的關聯。如需詳細資訊，請參閱建立核准規則範本與儲存器的關聯。

9. 選擇 Create (建立)。


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

建立核准規則範本 (AWS CLI)

您可以使用 AWS CLI 來建立核准規則範本。使用 AWS CLI 時，您可以指定範本的目的地參考，只在提取請求的目的地分支符合範本中的目的地分支時，才套用範本。

建立核准規則範本 (AWS CLI)

1. 在終端機或命令列，執行 create-approval-rule-template 命令，並指定：

   - 核准規則範本的名稱。考慮使用能描述用途的名稱。
   - 核准規則範本的描述。與名稱一樣，請考慮提供詳細的描述。
   - 核准規則範本的 JSON 結構。此結構可包含目的地參考的需求 (套用核准規則之提取請求的目的地分支)，以及核准集區成員 (核准計入所需核准數目中的使用者)。

   建立核准規則的內容時，您有兩種方式在核准集區中指定核准者：

   - CodeCommitApprovers：此選項僅需要 Amazon Web Services 帳戶和資源。它可用於名稱與提供的資源名稱相符的 IAM 使用者和聯合存取使用者。此選項非常強大，提供很大的靈活性。例如，如果您指定AWS帳戶 123456789012**Mary_Major**，並且以下所有項目都會計為來自該使用者的核准：
     - 帳戶中的 IAM 使用者 (`arn:aws:iam::123456789012:user/Mary_Major`)
     - 在 IAM 中識別為瑪麗 () 的聯合身分使用者 (`arn:aws:sts::123456789012:federated-user/Mary_Major`)

     除非您包含萬用字元 ()，否則此選項無法辨識假定角色為 *Mary_Maj* or (`arn:aws:sts::123456789012:assumed-role/`*SeniorDevelopers*`/`*Mary_Major*) 角色工作階段名稱的*SeniorDevelopers*使用中工作階段。`*Mary_Major`
   - 完全合格的 ARN：此選項可讓您指定 IAM 使用者或角色的完整 Amazon 資源名稱 (ARN)。

   如需 IAM ARN、萬用字元和格式的詳細資訊，請參閱 IAM 識別符。

   下列範例建立名為 **2-approver-rule-for-main** 且描述為 **Requires two developers from the team to approve the pull request if the destination branch is main** 的核准規則範本。此範本需要兩位擔任角色 **CodeCommitReview** 使用者核准任何提取請求，提取請求才能合併至 **main** 分支：

```
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. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "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"
    }
}
```

建立核准規則範本與儲存器的關聯

核准規則範本建立在特定的範本中AWS 區域，但在其關聯AWS 區域之前，它們不會影響其中的任何儲存庫。若要將範本套用至一或多個儲存庫，您必須建立該範本與一或多個儲存庫的關聯。您可以將單一範本套用至中的多個儲存庫AWS 區域。這可讓您建立一致的條件來核准和合併提取請求，協助您將儲存庫中的開發工作流程自動化和標準化。

您只能將核准規則範本與建立核准規則範本所在的儲存庫產生關聯。AWS 區域

如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

主題

建立核准規則範本 (主控台)

您在建立核准規則範本時，可能已與儲存庫建立關聯。(此步驟為選用。) 您可以編輯範本來新增或移除關聯。

建立核准規則範本與儲存庫的關聯

1. 請在以下位置開啟CodeCommit主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 選擇 Approval rule templates (核准規則範本)。選擇範本，然後選擇 Edit (編輯)。
3. 在 Associated Repositories (相關聯的儲存庫) 中，從 Repositories (儲存庫) 清單中選擇儲存庫。每個相關聯的儲存庫都顯示在清單方塊下方。
4. 選擇 儲存 。核准規則現在會套用至在這些相關聯儲存庫中建立的任何提取請求。

建立核准規則範本 (AWS CLI)

您可以使用 AWS CLI 建立核准規則範本與一或多個儲存庫的關聯。

建立範本與單一儲存庫的關聯

1. 在終端機或命令列，執行 associate-approval-rule-template-with-repository 命令，並指定：
   - 要與儲存庫建立關聯的核准規則範本名稱。
   - 要與核准規則範本相關聯的儲存庫名稱。

   例如，若要將名為 *2-* 的核准規則範本approver-rule-for-main與名為下列項目的存放庫產生關聯 *MyDemoRepo*：

   ```
   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，此命令不會傳回任何內容。

將範本與多個儲存庫建立關聯

1. 在終端機或命令列，執行 batch-associate-approval-rule-template-with-repositories 命令，並指定：

   - 要與儲存庫建立關聯的核准規則範本名稱。
   - 要與核准規則範本相關聯的儲存庫名稱。

   例如，將名為 **2-approver-rule-for-main** 的核准規則範本與名為 **MyDemoRepo** 和 **MyOtherDemoRepo** 的儲存庫建立關聯：

   ```
   aws codecommit batch-associate-approval-rule-template-with-repositories --
   repository-names "MyDemoRepo", "MyOtherDemoRepo" --approval-rule-template-name 2-
   approver-rule-for-main
   ```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

   ```
   {
       "associatedRepositoryNames": [
           "MyDemoRepo",
           "MyOtherDemoRepo"
       ],
       "errors": []
   }
   ```

管理核准規則範本

您可以在中管理核准規則範本，AWS 區域以協助瞭解它們的使用方式及其用途。例如，您可以編輯核准規則範本的名稱和描述，以協助其他人瞭解其用途。您可以在中列出所有核准規則範本AWS 區域，並取得範本內容和結構的相關資訊。您可以檢閱哪些範本與儲存庫相關聯，以及哪些儲存庫與範本相關聯。

如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

管理核准規則範本 (主控台)

您可以在 CodeCommit 主控台檢視和管理核准規則範本。

管理核准規則範本

1. 請在以下位置開啟CodeCommit主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 選擇「核准規則範本」，以檢視您登入之AWS 區域處的核准規則範本清單。

   > **Note**
   >
   > 核准規則範本 (核准規則範本)。AWS 區域

3. 如果您要變更範本，請從清單中選擇範本，然後選擇 Edit (編輯)。
4. 進行變更，然後選擇 Save (儲存變更)。

管理核准規則範本 (AWS CLI)

您可以使用下列 AWS CLI 命令來管理核准規則範本：

- list-approval-rule-templates，檢視所有核准規則範本 (AWS 區域
- get-approval-rule-template，檢視核准規則範本的內容
- update-approval-rule-template-content，變更核准規則範本的內容
- update-approval-rule-template-name，變更核准規則範本的名稱
- update-approval-rule-template-description，變更核准規則範本的描述
- list-repositories-for-approval-rule-template，檢視與核准規則範本相關聯的所有儲存庫
- list-associated-approval-rule-templates-for-repository，檢視與儲存庫相關聯的所有核准規則範本

列出所有核准規則範本 (AWS 區域

1. 在終端機或命令列上執行 list-approval-rule-templates 命令。例如，列出美國東部 (俄亥俄) 區域中所有核准規則範本 (俄亥俄) 區域：

   ```
   aws codecommit list-approval-rule-templates --region us-east-2
   ```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

   ```
   {
       "approvalRuleTemplateNames": [
           "2-approver-rule-for-main",
   ```

```
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

取得核准規則範本的內容

1. 在終端機或命令列，執行 get-approval-rule-template 命令，並指定核准規則範本的名稱：

```
aws codecommit get-approval-rule-template --approval-rule-template-name 1-approver-
rule-for-all-pull-requests
```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "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."
    }
}
```

更新核准規則範本的內容

1. 在終端機或命令提示字元下，執行 update-approval-rule-template-content 命令，並指定範本的名稱和變更的內容。例如，變更名為 **1-approver-rule** 的核准規則範本的內容，以將核准集區重新定義為擔任角色 **CodeCommitReview** 的使用者：

```
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. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "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"
    }
}
```

更新核准規則範本的名稱

1. 在終端機或命令提示字元下，執行 update-approval-rule-template-name 命令，並指定目前名稱和要變成的名稱。例如，將核准規則範本的名稱從 **1-approver-rule** 變更為 **1-approver-rule-for-all-pull-requests**：

```
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. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "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"
    }
}
```

更新核准規則範本的描述

1. 在終端機或命令列，執行 update-approval-rule-template-description 命令，並指定核准規則範本的名稱和新描述：

```
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. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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"
    }
}
```

列出與範本相關聯的所有儲存庫

1. 在命令列或終端機，執行 list-repositories-for-approval-rule-template 命令，並指定範本的名稱：

```
aws codecommit list-repositories-for-approval-rule-template --approval-rule-
template-name 2-approver-rule-for-main
```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "repositoryNames": [
        "MyDemoRepo",
        "MyClonedRepo"
    ]
}
```

列出與儲存庫相關聯的所有範本

1. 在命令列或終端機，執行 list-associated-approval-rule-templates-for-repository 命令，並指定儲存庫的名稱：

```
aws codecommit list-associated-approval-rule-templates-for-repository --repository-
name MyDemoRepo
```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "approvalRuleTemplateNames": [
        "2-approver-rule-for-main",
        "1-approver-rule-for-all-pull-requests"
    ]
}
```

取消核准規則範本

如果核准規則範本所產生的核准規則對儲存庫中的團隊工作流程不再有意義，您可以取消範本與該儲存庫的關聯。取消範本的關聯不會移除範本與儲存庫產生關聯時所建立的任何核准規則。

如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

取消核准規則範本 (主控台) (主控台)

您可以使用主控台移除儲存庫與核准規則範本之間的關聯。

取消核准規則範本與儲存庫的關聯

1. 請在以下位置開啟CodeCommit主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 選擇 Approval rule templates (核准規則範本)。選擇要與一或多個儲存庫取消關聯的範本，然後選擇 Edit (編輯)。
3. 在 Associated repositories (相關聯的儲存庫) 中，選擇您要取消關聯的儲存庫旁邊的 X。儲存庫名稱不再出現。
4. 選擇 儲存 。核准規則不會套用至這些儲存庫中建立的提取請求。規則仍會套用至關聯存在時所提出的提取請求。

取消核准規則範本 (AWS CLI)

您可以使用 AWS CLI 來取消一或多個儲存庫與核准規則範本的關聯。

取消核准規則範本與儲存庫的關聯

1. 在終端機或命令列，執行 disassociate-approval-rule-template-from-repository 命令，並指定：
   - 核准規則範本的名稱。
   - 儲存庫的名稱。

   例如，取消名為 **1-approver-rule-for-all-pull-requests** 的核准規則範本與名為 **MyDemoRepo** 的儲存庫的關聯：

   ```
   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，此命令不會傳回任何內容。

取消核准規則範本與多個儲存庫的關聯

1. 在終端機或命令列，執行 batch-disassociate-approval-rule-template-from-repositories 命令，並指定：
   - 核准規則範本的名稱。
   - 儲存庫的名稱。

例如，取消名為 **1-approver-rule-for-all-pull-requests** 的核准規則範本與名為 **MyDemoRepo** 和名為 **MyOtherDemoRepo** 的儲存庫的關聯：

```
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. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "disassociatedRepositoryNames": [
        "MyDemoRepo",
        "MyOtherDemoRepo"
    ],
    "errors": []
}
```

刪除核准規則範本

如果您在任何儲存庫中未使用某個核准規則範本，則可以該範本。刪除未使用的核准規則範本有助於保持範本井然有序，並且更容易找到適合工作流程的範本。

如需有關核准規則範本的受管理原則和權限的詳細資訊，請參閱核准規則範本動作的權限和 CodeCommit 的 AWS 受管政策。

主題

- 刪除核准規則範本 (主控台)
- 刪除核准規則範本 (AWS CLI)

刪除核准規則範本 (主控台)

如果核准規則範本不再與您的開發工作相關，您可以刪除該範本。當您使用主控台刪除核准規則範本時，在刪除程序期間，該範本會取消與任何儲存庫的關聯。

刪除核准規則範本

1. 請在以下位置開啟CodeCommit主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 選擇 Approval rule templates (核准規則範本)。選擇您要刪除的範本，然後選擇 Delete (刪除)。

刪除核准規則範本 (AWS CLI)

如果核准規則已與所有儲存庫取消關聯，您可以使用 AWS CLI 刪除此核准規則。如需詳細資訊，請參閱取消核准規則範本 (AWS CLI)。

刪除核准規則範本

1. 在終端機或命令列，執行 delete-approval-rule-template 命令，並指定您要刪除的核准規則範本名稱：

```
aws codecommit delete-approval-rule-template --approval-rule-template-name 1-approver-for-all-pull-requests
```

2. 如果成功，此命令傳回的輸出會類似如下。如果已刪除核准規則範本，則此命令不會傳回任何結果。

```
{
    "approvalRuleTemplateId": "41de97b7-EXAMPLE"
}
```

在 AWS CodeCommit 存儲庫中使用提交

遞交是儲存庫的內容和內容變更的快照。每當使用者遞交和推送變更時，該資訊就會儲存和存放起來。因此，儲存的資訊也包括誰遞交變更、遞交的日期和時間，以及在遞交時所做的變更。您也可以將遞交加上標籤，以輕鬆地識別特定的遞交。在中 CodeCommit，您可以：

- 檢閱遞交。
- 在圖表中檢視遞交的歷史記錄。
- 將遞交與其父系或另一個指標做比較。
- 對遞交新增註解，並回覆其他人所做的註解。

您必須先將本機電腦設定為連線至 CodeCommit 存放庫，才能將提交推送至儲存庫。如需最簡單的方式，請參閱適用於使用 Git 認證的 HTTPS 使用者。

如需有關在中使用儲存庫其他方面的資訊 CodeCommit 使用儲存庫使用檔案，請參閱使用提取請求使用分支、、和使用者偏好設定。

主題

- 在中創建提交 AWS CodeCommit
- 檢視提交詳細資訊 AWS CodeCommit
- 比較提交 AWS CodeCommit

- 評論提交 AWS CodeCommit
- 在中建立一個 Git 標籤 AWS CodeCommit
- 在中檢視 Git 標籤詳細資料 AWS CodeCommit
- 在中刪除一個 Git 標籤 AWS CodeCommit

在中創建提交 AWS CodeCommit

當您建立新儲存庫的第一次提交時，您可以使用 AWS CLI 和指put-file令。這將創建第一次提交，它允許您創建和指定新存儲庫的默認分支。您可以使用 Git 或在 AWS CLI CodeCommit 儲存庫中建立提交。如果本地存儲庫連接到 CodeCommit 存儲庫，則可以使用 Git 將提交從本地存儲庫推送到 CodeCommit 存儲庫。若要直接在 CodeCommit 主控台中建立提交，請參閱建立檔案或將檔案新增至 AWS CodeCommit儲存庫和編輯AWS CodeCommit儲存庫中檔案的內容。

> **Note**
>
> 最佳作法是，建議您使用最新的受支援版本的 AWS CLI、Git 和其他軟體。如果您使用 AWS CLI，請確定您已安裝最新版本，以確保您使用的是包含`create-commit`指令的版本。

主題

- 使用 AWS CLI
- 使用 Git 客戶端創建提交
- 使用建立提交 AWS CLI

使用 AWS CLI

您可以使用 AWS CLI 和命`put-file`令來建立儲存庫的第一次提交。使用put-file創建一個第一次提交，將文件添加到您的空存儲庫中，並使用您指定的名稱創建一個分支。它將新分支指定為存儲庫的默認分支。

> **Note**
>
> 若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用 AWS CLI

1. 在本機電腦上，建立要新增為第一個檔案至 CodeCommit儲存庫的檔案。常見的做法是建立一個 README.md markdown 檔案，向其他儲存庫使用者說明此儲存庫的用途。如果您包含README.md檔案，檔案的內容會自動顯示在 CodeCommit主控台中儲存庫的「代碼」頁底部。
2. 在終端機或命令列，執行 put-file 命令，並指定：

   - 您要新增第一個檔案的存放庫名稱。
   - 要創建為默認分支的分支的名稱。
   - 檔案的本機位置。用於此位置的語法因您的本機作業系統而有所不同。
   - 您要新增的檔案名稱，包括儲存庫中已更新檔案的路徑。
   - 您要與此檔案建立關聯的使用者名稱和電子郵件。
   - 說明為何新增此檔案的遞交訊息。

   使用者名稱、電子郵件地址和提交訊息是選擇性的，但可以協助其他使用者瞭解進行變更的使用者和原因。如果您未提供使用者名稱，則 CodeCommit 預設使用您的 IAM 使用者名稱或衍生您的主控台登入作為作者名稱。

   例如，若要新增名為 *Readme.md* 的檔案，其內容為「歡迎來到我們的團隊資料庫！」 到一個名為*##MyDemoRepo*的分支的存儲庫：

   ```
   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."
   ```

   如果此命令成功執行，您會看到類似如下的輸出傳回：

   ```
   {
       "commitId": "724caa36EXAMPLE",
       "blobId": "a8a94062EXAMPLE",
       "treeId": "08b2fc73EXAMPLE"
   }
   ```

使用 Git 客戶端創建提交

您可以使用安裝在本機電腦上的 Git 用戶端來建立提交，然後將這些提交推送至您的 CodeCommit 儲存庫。

1. 完成必要條件，包括設定 。

   > **Important**
   >
   > 如果尚未完成設定，您無法使用 Git 連接到或遞交至儲存庫。

2. 確定您是在正確的分支中建立遞交。若要查看可用的分支清單，並了解您目前設定使用的分支，請執行 git branch。將會顯示所有分支。您目前的分支旁會顯示星號 (*)。若要切換到不同的分支，請執行 git checkout ***branch-name***。如果這是您的第一次提交，請運行git config 命令來配置 Git 客戶端，以使用您想用於該分支的名稱創建一個初始分支。例如，如果您希望默認分支具有名稱*##*：

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

   > **Tip**
   >
   > 此命令僅適用於 Git 版本 2.28 及更新版本。
   > 您還可以運行此命令，將所有新創建的存儲庫的默認分支名稱設置**development**為：
   >
   > ```
   > git config --global init.defaultBranch development
   > ```

3. 對分支進行變更 (例如，新增、修改或刪除檔案)。

   例如，在本機存放庫中，建立一個以下列文字命名`bird.txt`的檔案：

   ```
   bird.txt
   --------
   Birds (class Aves or clade Avialae) are feathered, winged, two-legged, warm-
   blooded, egg-laying vertebrates.
   ```

4. 執行 git status，應該會指出 `bird.txt` 尚未在任何擱置中包含的遞交：

   ```
   ...
   Untracked files:
   ```

```
  (use "git add <file>..." to include in what will be committed)

        bird.txt
```

5. 執行 git add bird.txt 以在擱置中遞交中包含新的檔案。
6. 如果您再次執行 git status，應該會看到類似如下的輸出：這表示 `bird.txt` 現在已屬於擱置中遞交或暫存供遞交：

```
...
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   bird.txt
```

7. 若要完成遞交，請執行 git commit 搭配 `-m` 選項 (例如， git commit -m "***Adding bird.txt to the repository.***")。`-m` 選項會建立遞交訊息。
8. 如果您再次執行 git status，應該會看到類似如下的輸出：它表明提交已準備好從本地回購推送到 CodeCommit 存儲庫：

```
...
nothing to commit, working directory clean
```

9. 在將最終提交從本地回購推送到 CodeCommit 存儲庫之前，您可以通過運行查看正在推送的內容git diff --stat ***remote-name***/***branch-name***，其中*####*是本地回購用於 CodeCommit 存儲庫的暱稱，而 branch *name* *#######*名稱。

> **Tip**
>
> 若要取得別名，請執行 git remote。若要取得分支名稱的清單，請執行 git branch。目前的分支旁會顯示星號 (*)。您也可以執行 git status 以取得目前的分支名稱。

> **Note**
>
> 如果您從本地存儲庫的角度克隆了存儲庫，則*####*不是存儲庫的名稱。 CodeCommit 複製儲存庫時，*remote-name* 會自動設定為 `origin`。

例如，git diff --stat origin/main 會顯示與下列類似的輸出：

```
bird.txt | 1 +
1 file changed, 1 insertion(+)
```

輸出假設您已將本地回購連接到 CodeCommit 存儲庫。(如需指示，請參閱連接到儲存庫。)

10. 當您準備將提交從本地回購推送到 CodeCommit 存儲庫時，請運行git push ***remote-name branch-name***，其中####是本地存儲庫用於 CodeCommit 存儲庫的暱稱，而分###是要推送到存儲庫的分支的名稱。 CodeCommit

    例如，執行 git push origin main 會顯示與下列類似的輸出：

    針對 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
    ```

    針對 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**
    >
    > 如果您將 -u 選項新增至 git push (例如，git push -u origin main)，則未來只需執行 git push，因為已設定上游追蹤資訊。若要取得上游追蹤資訊，請執行 git remote show ***remote-name*** (例如，git remote show origin)。

如需更多選項，請參閱 Git 文件。

使用建立提交 AWS CLI

您可以使用 AWS CLI 和命`create-commit`令，在指定分支的尖端上建立儲存庫的提交。您也可以建立未參照的合併遞交來代表合併兩個遞交指標的結果。如需詳細資訊，請參閱建立未參照遞交。

> **Note**
>
> 若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

建立遞交

1. 在您的本機電腦上，進行您要提交至 CodeCommit儲存庫的變更。
2. 在終端機或命令列，執行 create-commit 命令，並指定：
   - 您要遞交變更的儲存庫。
   - 您要遞交變更的分支。
   - 對該分支所進行最新遞交的完整遞交 ID，也稱為頂端或標頭遞交，或父遞交 ID。
   - 如果您所做的變更會刪除資料夾的內容，是否保留任何空的資料夾。根據預設，此值為 false。
   - 您想要新增、變更或刪除的檔案的相關資訊。
   - 要與這些變更建立關聯的使用者名稱和電子郵件。
   - 用以說明您為什麼做這些變更的遞交訊息。

   使用者名稱、電子郵件地址和遞交訊息是選用的，但可協助其他使用者知道進行變更的人員及原因。如果您未提供使用者名稱，則 CodeCommit 預設使用您的 IAM 使用者名稱或衍生您的主控台登入作為作者名稱。

   例如，要為存儲庫創建一個提交，該存儲庫將README.md文件添加到*#*分支*MyDemoRepo*中命名的存儲庫中。該文件的內容位於 Base64 中，內容為「歡迎來到我們的團隊存儲庫！」：

```
aws codecommit create-commit --repository-name MyDemoRepo --
branch-name main --parent-commit-id 4c925148EXAMPLE --put-files
 "filePath=README.md,fileContent=V2VsY29tZSB0byBvdXIgdGVhbSByZXBvc2l0b3J5IQo="
```

> **Tip**
>
> 要獲取父提交 ID，請運行 get-branch 命令。

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
    "commitId": "4df8b524-EXAMPLE",
    "treeId": "55b57003-EXAMPLE",
    "filesAdded": [
        {
            "blobId": "5e1c309dEXAMPLE",
            "absolutePath": "meeting.md",
            "fileMode": "NORMAL"
        }
    ],
    "filesDeleted": [],
    "filesUpdated": []
}
```

file1.py # file2.txt ################ picture.png #### image1.png####### images ###################### .py ########## ExampleSolutionMyDemoRepo.py ###MyFeatureBranch###### ID # 4c925148###

```
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**
>
> --put-files區段的語法會根據您的作業系統而有所不同。上面的例子是針對 Linux，macOS 或 Unix 用戶和 Windows 用戶進行了優化使用 Bash 模擬器。採用命令列或 Powershell 的 Windows 使用者，應該使用該系統適用的語法。

如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
   "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"
```

```
        }
    ]
}
```

檢視提交詳細資訊 AWS CodeCommit

您可以使用 AWS CodeCommit 控制台瀏覽存儲庫中提交的歷史記錄。這可協助您識別儲存庫中進行的變更，包括：

- 變更何時以及由誰進行。
- 特定遞交何時合併到分支。

檢視分支遞交的歷史記錄可能也有助於了解分支之間的差異。如果您使用標記，也可以快速地檢視加上標籤的遞交和該標記遞交的父項。在命令列中，您可以使用 Git 來檢視本機存放庫或 CodeCommit 儲存庫中提交的詳細資訊。

瀏覽存儲庫中的提交

您可以使用 AWS CodeCommit 控制台瀏覽存儲庫的提交歷史記錄。您也可以隨著時間檢視儲存庫中遞交及其分支的圖表。這可協助您了解儲存庫的歷史記錄，包括進行變更的時間。

> **Note**
>
> 使用 git rebase 命令來重設儲存庫的基準會變更儲存庫的歷史記錄，這可能造成遞交不按順序顯示。如需詳細資訊，請參閱 Git 分支重設基準或您的 Git 文件。

主題

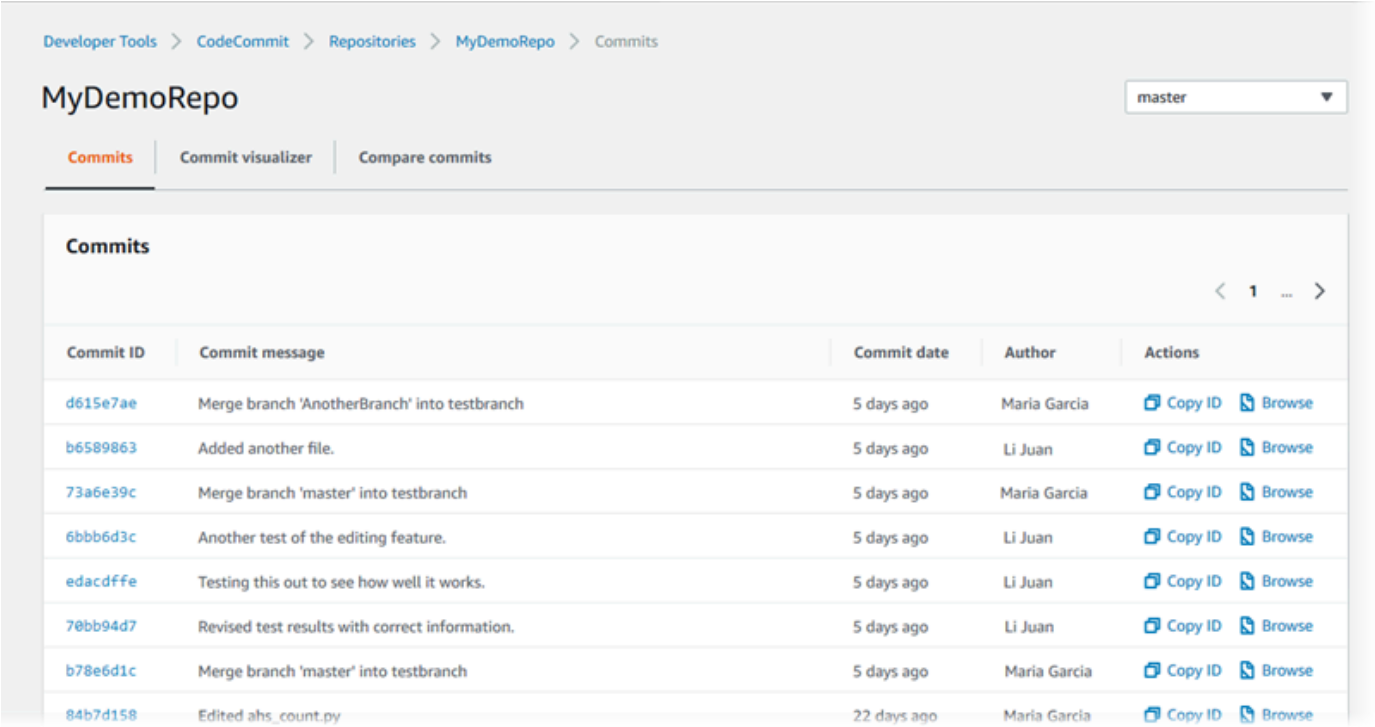
- 瀏覽存儲庫的提交歷史記錄
- 查看存儲庫提交歷史記錄的圖表

瀏覽存儲庫的提交歷史記錄

您可以瀏覽特定分支或儲存庫標籤的遞交歷史記錄，包含遞交者和遞交訊息的詳細資訊。您也可以檢視遞交的程式碼。

瀏覽遞交的歷史記錄

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢閱遞交歷史記錄的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)。在遞交歷史記錄檢視中，會顯示預設分支中儲存庫遞交的歷史記錄，以遞交日期的相反時間順序顯示。日期與時間採用國際標準時間 (UTC)。您可以檢視不同分支遞交的歷史記錄，方法是選擇檢視選取器按鈕，然後從清單中選擇分支。如果您使用的是您的儲存庫中的標籤，您可以透過在檢視選取器按鈕中選擇該標籤，以檢視具有特定標籤的遞交和及父系。



4. 若要檢視遞交與其父系之間的差異，並選擇遞交的 ID 來查看對於變更的相關評論。如需詳細資訊，請參閱 將提交與其父提交進行比較 及 對遞交做註解。若要檢視遞交與任何其他遞交指標 (包括分支、標籤或遞交 ID) 之間的差異，請參閱比較任兩個遞交指標。
5. 執行下列其中一項或多項：
   - 若要檢視進行變更的日期和時間，請將滑鼠的游標移到遞交日期上。
   - 若要檢視完整遞交 ID，請複製然後將它貼到文字編輯器或其他位置。若要複製它，請選擇「複製 ID」。
   - 若要檢視程式碼在遞交時的狀態，請選擇 Browse (瀏覽)。儲存庫在該遞交時的內容會顯示在 Code (程式碼) 檢視中。檢視選取器按鈕會顯示縮寫的遞交 ID 而非分支或標籤。

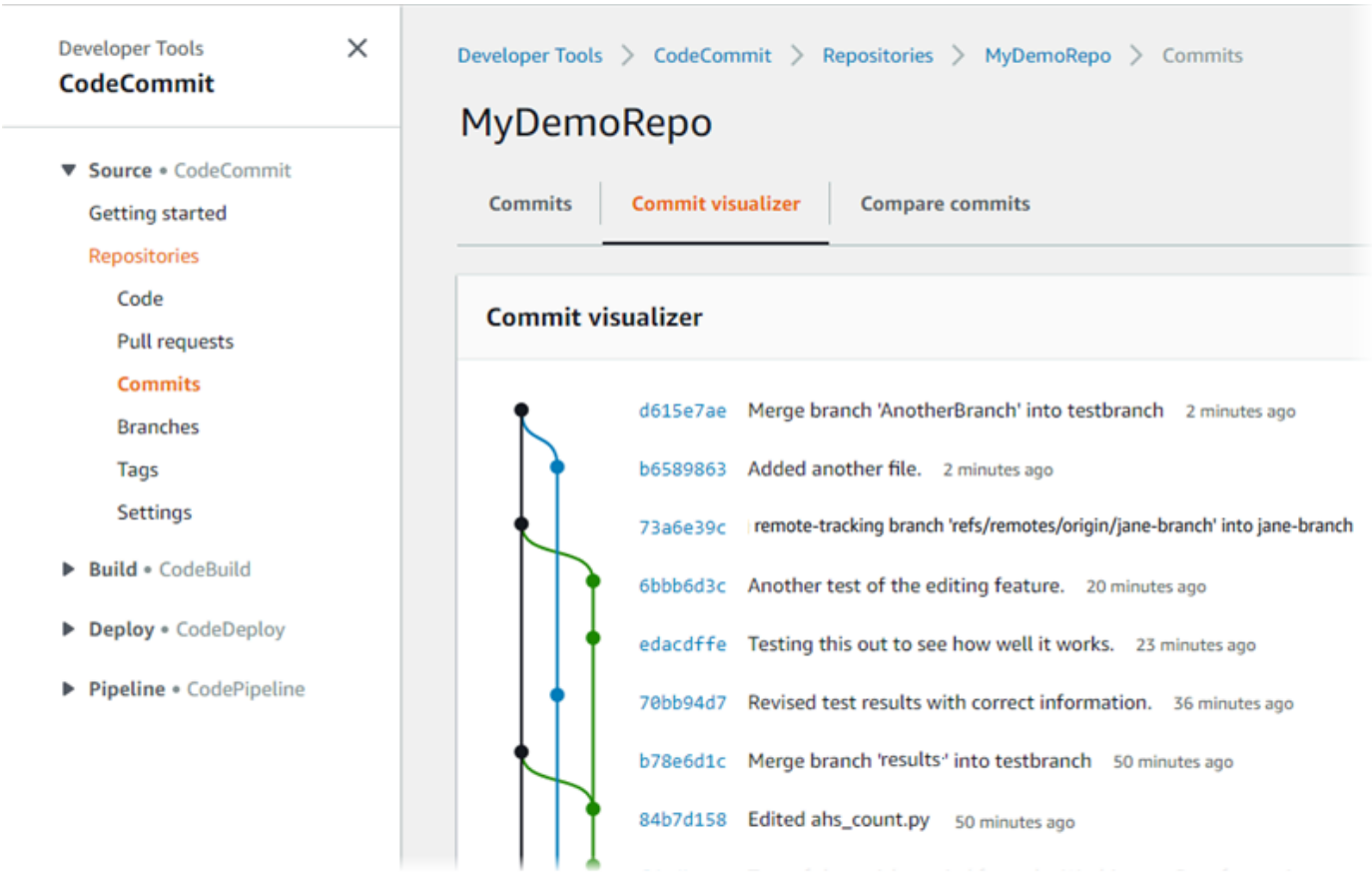
查看存儲庫提交歷史記錄的圖表

您可以檢視對儲存庫所做的遞交圖表。Commit Visualizer 檢視是對儲存庫的分支所進行所有遞交的有向無環圖 (DAG) 呈現。此圖形呈現可協助您了解新增或合併遞交和關聯功能的時間。它也可以協助您了解與其他變更相關的變更是何時進行。

> **Note**
>
> 在遞交圖表中，使用向前快轉方法合併的遞交不會以單獨線條的形式顯示。

檢視遞交的圖表

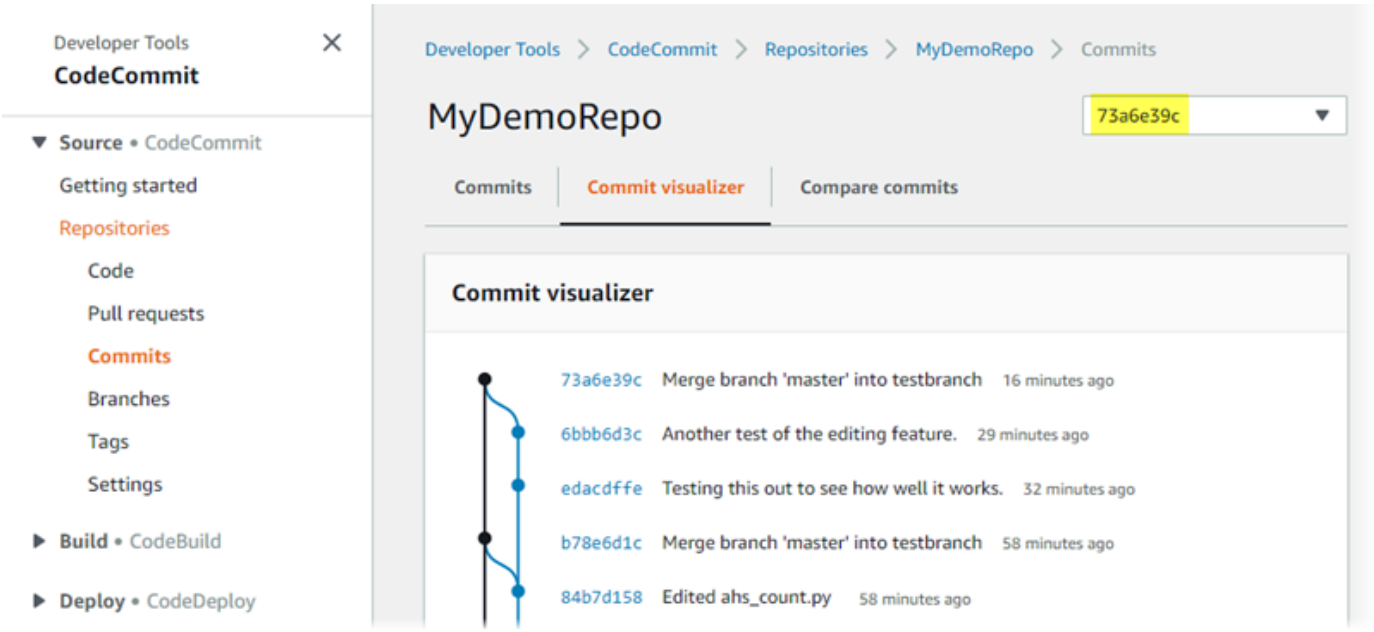
1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢視遞交圖表的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)，然後選擇 Commit visualizer (遞交視覺化工具) 索引標籤。



在提交圖形中，縮寫的提交 ID 和每個提交訊息的主旨會顯示在圖形中該點的旁邊。

> **Note**
>
> 圖表在一頁上最多可以顯示 35 個分支。如果超過 35 個分支，則圖表會太複雜而無法顯示。您可以透過兩個方式將檢視簡化：
>
> - 使用檢視選取器按鈕來顯示特定分支的圖形。
> - 透過貼上完整的遞交 ID 至搜尋方塊，從該遞交轉譯圖形。

4. 若要從遞交轉譯圖形，請在圖形中選擇與該遞交對應的點。檢視選取器按鈕會變更為縮寫的遞交 ID。



檢視提交詳細資訊 (AWS CLI)

Git 可讓您檢視遞交的詳細資訊。您也可以執行下列命令，使用 AWS CLI 來檢視本機存放庫或 CodeCommit 儲存庫中提交的詳細資訊：

- 若要檢視一個遞交的相關資訊，請執行 aws codecommit get-commit。
- 若要檢視多個遞交的相關資訊，請執行 aws codecommit batch-get-commits。
- 若要檢視合併遞交的相關資訊，請執行 aws codecommit get-merge-commit。
- 若要檢視遞交指標 (分支、標籤、HEAD 或其他完整參考，例如遞交 ID) 的變更的相關資訊，請執行 aws codecommit get-differences。
- 若要檢視儲存庫中 Git Blob 物件的 base64 編碼內容，請執行 aws codecommit get-blob。

檢視遞交的詳細資訊

1. 執行 aws codecommit get-commit 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選--repository-name項)。
   - 完整遞交 ID。

   例如，若要在名為的 CodeCommit 儲存庫中檢視 ID 317f8570EXAMPLE 的提交相關資訊MyDemoRepo：

   ```
   aws codecommit get-commit  --repository-name MyDemoRepo  --commit-id
    317f8570EXAMPLE
   ```

2. 如果成功，此命令的輸出包含下列：

   - 遞交作者 (如 Git 所設定) 的詳細資訊，包括採用時間戳記格式的日期和國際標準時間 (UTC) 位移。
   - 遞交者 (如 Git 所設定) 的詳細資訊，包括採用時間戳記格式的日期和 UTC 位移。
   - 遞交存在位置的 Git 樹狀目錄 ID。
   - 父遞交的遞交 ID。
   - 遞交訊息。

   以下是基於上述範例命令的一些範例輸出：

   ```
   {
       "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"
    }
}
```

檢視合併遞交的相關資訊

1. 執行 get-merge-commit 命令，並指定：

   - 合併來源的遞交指標 (使用 --source-commit-specifier 選項)。
   - 合併目的地的遞交指標 (使用 --destination-commit-specifier 選項)。
   - 您想要使用的合併選項 (使用 --merge-option 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。

   例如，若要檢視名為*########-bug1234* 的來源分支合併提交的相關資訊，其目標分支使用名為 *main* 的儲存庫中的 *THREE* _WAY_MERGE 策略：*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，此命令的輸出會傳回類似如下的資訊：

   ```
   {
       "sourceCommitId": "c5709475EXAMPLE",
       "destinationCommitId": "317f8570EXAMPLE",
       "baseCommitId": "fb12a539EXAMPLE",
       "mergeCommitId": "ffc4d608eEXAMPLE"
   }
   ```

檢視多個遞交的相關資訊

1. 執行 batch-get-commits 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選`--repository-name`項)。
   - 您要檢視其資訊的每個遞交的完整遞交 ID 清單。

例如，若要檢視使用 ID 317f8570EXAMPLE 和名稱為的 CodeCommit 儲存庫4c925148EXAMPLE中的提交資訊MyDemoRepo：

```
aws codecommit batch-get-commits  --repository-name MyDemoRepo  --commit-ids
 317f8570EXAMPLE 4c925148EXAMPLE
```

2. 如果成功，此命令的輸出包含下列：

   - 遞交作者 (如 Git 所設定) 的詳細資訊，包括採用時間戳記格式的日期和國際標準時間 (UTC) 位移。
   - 遞交者 (如 Git 所設定) 的詳細資訊，包括採用時間戳記格式的日期和 UTC 位移。
   - 遞交存在位置的 Git 樹狀目錄 ID。
   - 父遞交的遞交 ID。
   - 遞交訊息。

   以下是基於上述範例命令的一些範例輸出：

```
{
    "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"
    }
}
```

檢視遞交指標變更的詳細資訊

1. 執行 aws codecommit get-differences 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選`--repository-name`項)。
   - 您要取得資訊的遞交指標。只有 `--after-commit-specifier` 為必要。如果您不指定 `--before-commit-specifier`，將會顯示在 `--after-commit-specifier` 時當時的所有檔案。

   例如，若要檢視 ID 317f8570EXAMPLE 的提交與名稱為的 CodeCommit 儲存庫之間差異4c925148EXAMPLE的相關資訊MyDemoRepo：

   ```
   aws codecommit get-differences  --repository-name MyDemoRepo  --before-commit-specifier 317f8570EXAMPLE --after-commit-specifier 4c925148EXAMPLE
   ```

2. 如果成功，此命令的輸出包含下列：

   - 差異的清單，包含變更類型 (A 代表已新增，D 代表已刪除，或 M 代表已修改)。
   - 檔案變更類型的模式。

- 包含變更的 Git Blob 物件 ID。

以下是基於上述範例命令的一些範例輸出：

```
{
    "differences": [
        {
            "afterBlob": {
                "path": "blob.txt",
                "blobId": "2eb4af3bEXAMPLE",
                "mode": "100644"
            },
            "changeType": "M",
            "beforeBlob": {
                "path": "blob.txt",
                "blobId": "bf7fcf28fEXAMPLE",
                "mode": "100644"
            }
        }
    ]
}
```

檢視 Git Blob 物件的詳細資訊

1. 執行 aws codecommit get-blob 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選`--repository-name`項)。
   - Git Blob 的 ID (使用 `--blob-id` 選項)。

   例如，若要檢視 Git blob 的相關資訊，其`2eb4af3bEXAMPLE`識別碼為`MyDemoRepo`：CodeCommit

   ```
   aws codecommit get-blob  --repository-name MyDemoRepo  --blob-id 2eb4af3bEXAMPLE
   ```

2. 如果成功，此命令的輸出包含下列：

   - Blob 的 base64 編碼內容，通常是一個檔案。

例如，前一個命令的輸出應類似以下：

```
{
    "content": "QSBCaW5hcnkgTGFyToEXAMPLE="
}
```

檢視提交詳細資訊 (Git)

在執行這些步驟之前，您應該已經將本地存放庫連接到 CodeCommit 存儲庫並提交了更改。如需說明，請參閱連接到儲存庫。

若要顯示最近提交至儲存庫的變更，請執行git show指令。

```
git show
```

此命令會產生類似下列的輸出：

```
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**
>
> 在此範例和以下範例中，遞交 ID 已縮寫。不會顯示完整的遞交 ID。

若要檢視發生的變更，請使用 git show 命令搭配遞交 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.
```

要查看兩次提交之間的差異，請運行git diff命令並包含兩個提交 ID。

```
git diff ce22850d 4f8c6f9d
```

在此範例中，這兩個遞交之間的差異是已新增了兩個檔案。此命令會產生類似下列的輸出：

```
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 查看有關本地存儲庫中提交的詳細信息，請運行以下git log命令：

```
git log
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
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
```

若只要顯示遞交 ID 和訊息，請執行 git log --pretty=oneline 命令：

```
git log --pretty=oneline
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
94ba1e60 Added horse.txt
4c925148 Added cat.txt and dog.txt
```

如需更多選項，請參閱 Git 文件。

比較提交 AWS CodeCommit

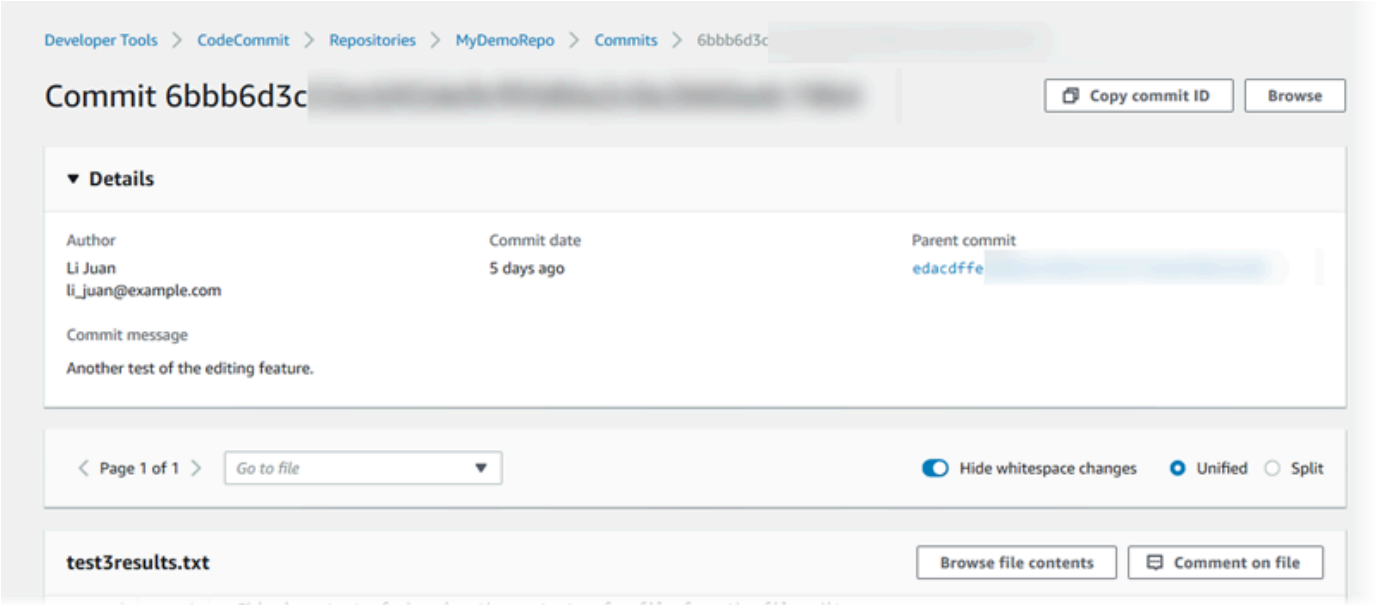
您可以使用 CodeCommit 控制台查看 CodeCommit存儲庫中提交說明符之間的差異。您可以快速檢視遞交與其父系之間的差異。您也可以比較任兩個參考，包括遞交 ID。

主題

將提交與其父提交進行比較

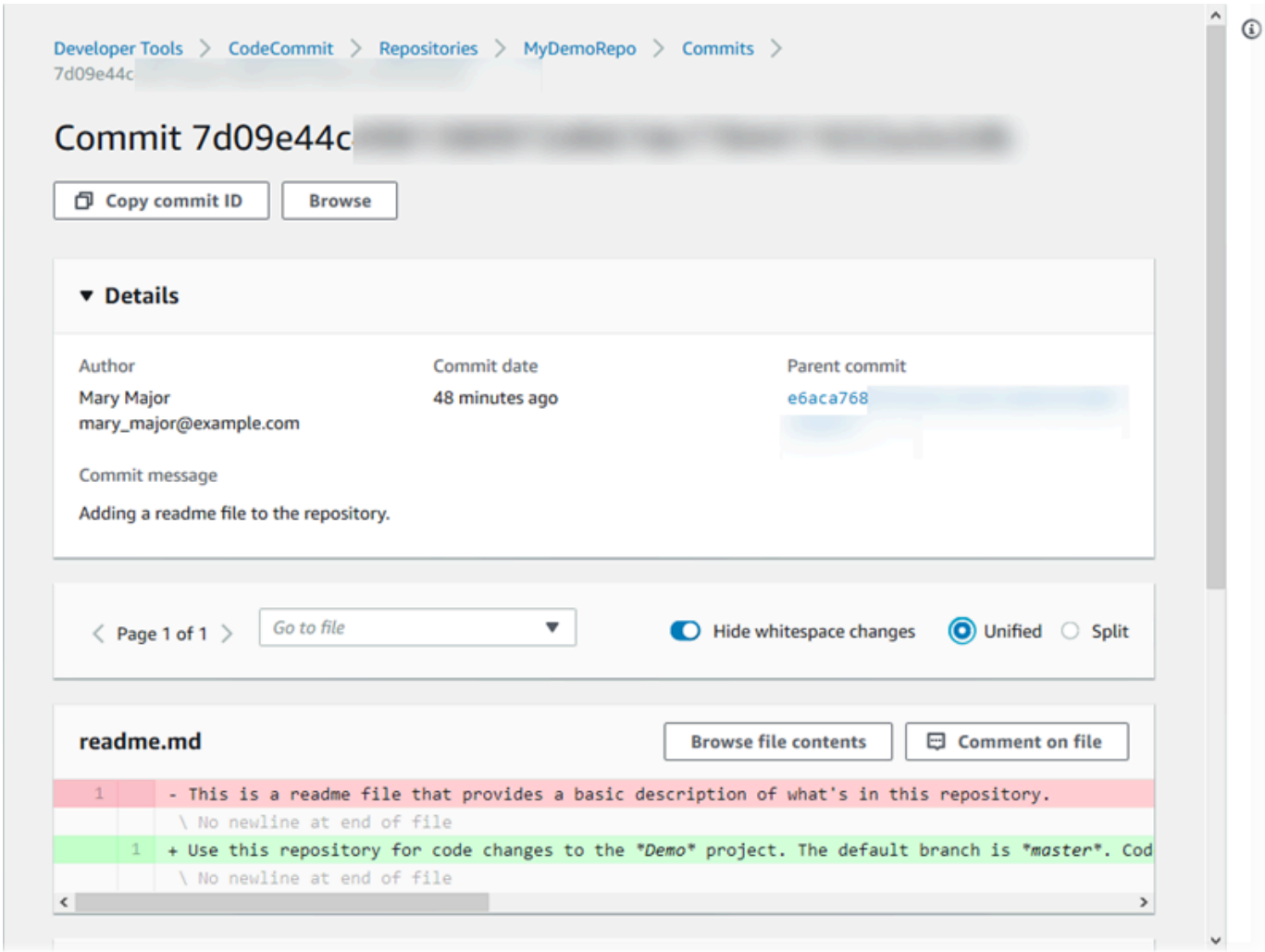
您可以快速檢視遞交與其父系之間的差異，以檢閱遞交訊息、遞交者和變更內容。

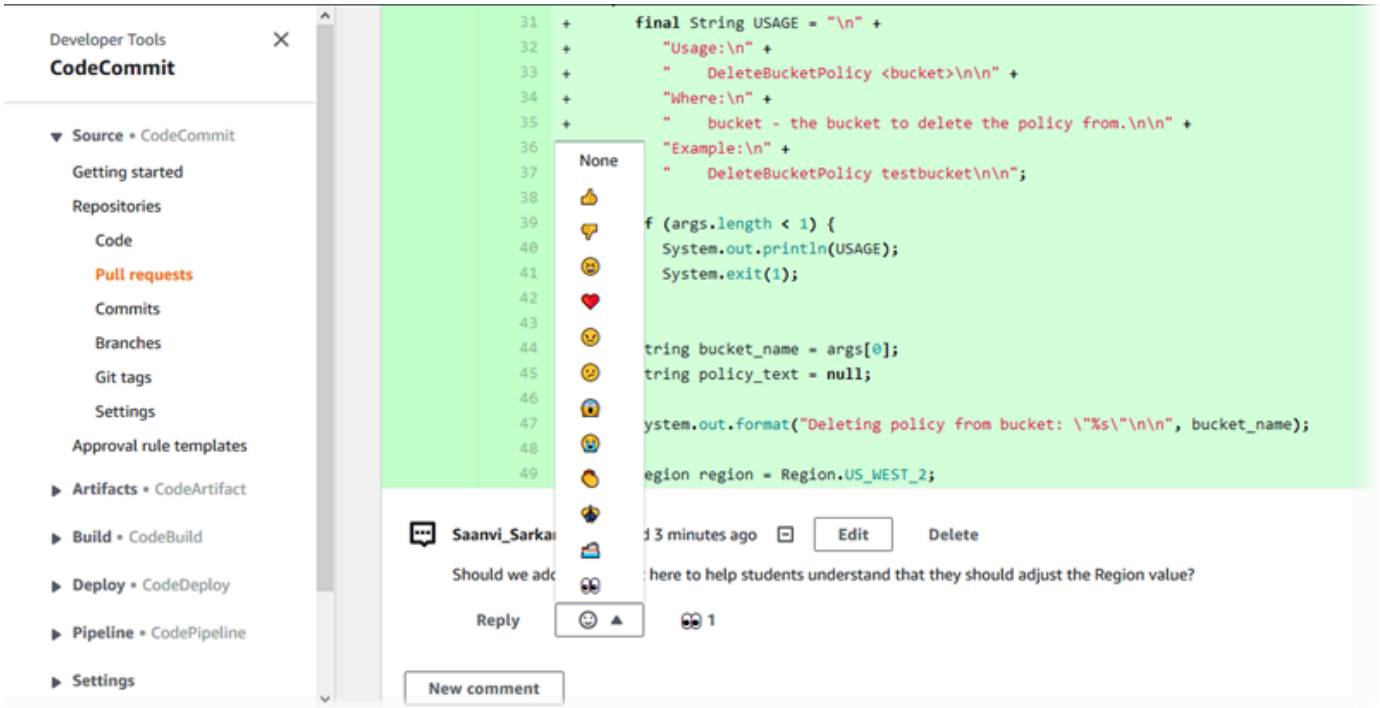
1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 頁面上，選擇您要檢視遞交與其父系之間差異所在的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)。
4. 選擇清單中任何縮寫的遞交 ID。檢視會變更為顯示其遞交的詳細資訊，包括它與其父遞交之間的差異。



您可以透過左右並排 (Split (分割) 檢視) 或內嵌 (Unified (統一) 檢視) 來顯示變更。您也可以隱藏或顯示空格的變更。您也可以新增評論。如需詳細資訊，請參閱 對遞交做註解。

> **Note**
>
> 用於檢視程式碼的偏好設定及其他主控台設定只要有變動，就會儲存為瀏覽器 Cookie。如需詳細資訊，請參閱 使用者偏好設定。


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



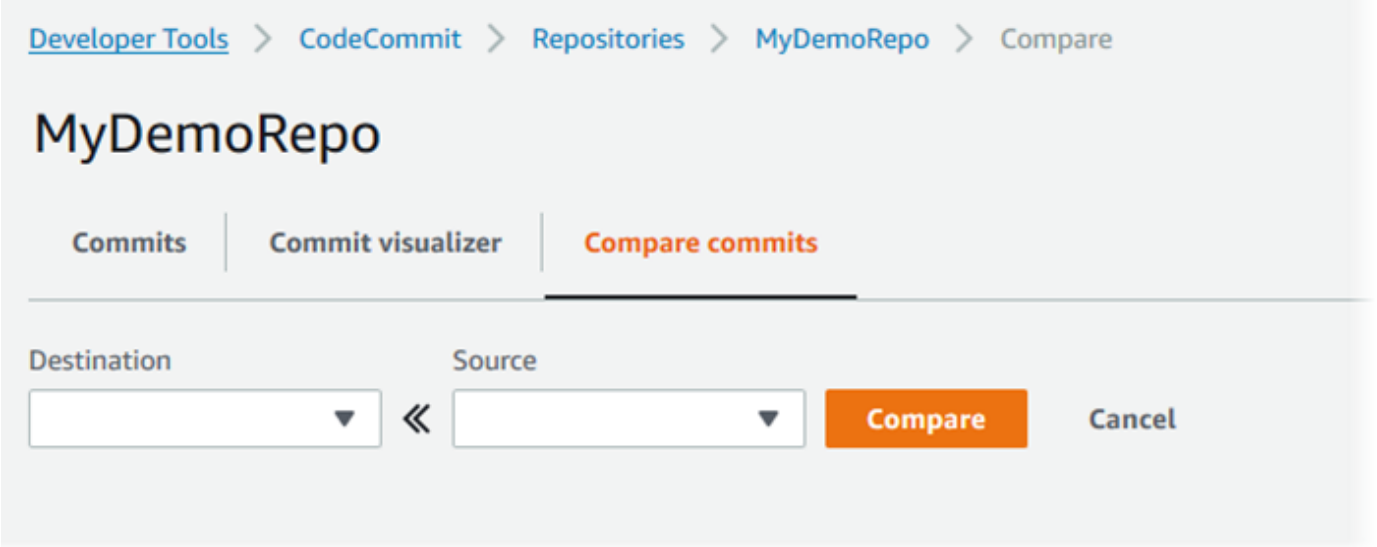
> **Note**
>
> 根據行結尾樣式，您的程式碼編輯器和其他因素，您可能看見新增或刪除整行，而不是行中的特定變更。詳細資訊層級符合 git show 或 git diff 命令中傳回的內容。

5. 若要將遞交與其父系比較，請從 Commit visualize (遞交視覺化工具) 索引標籤，選擇縮寫的遞交 ID。將顯示遞交詳細資訊，包括遞交與其父系之間的變更。

比較任兩個遞交指標

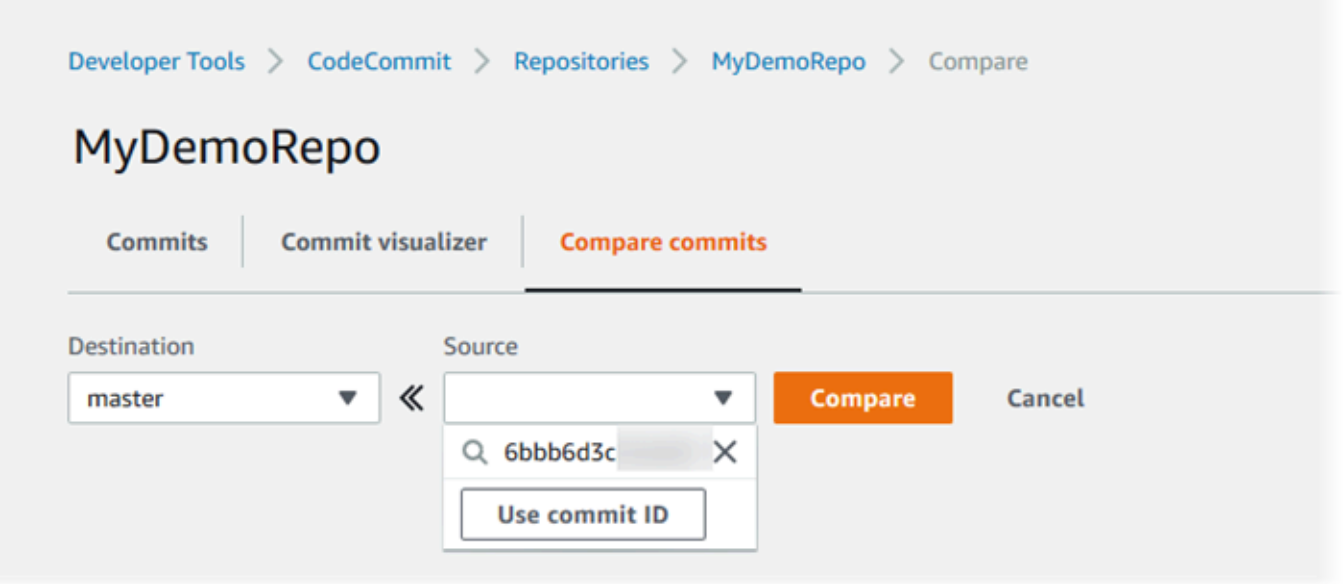
您可以在 CodeCommit 控制台中查看任何兩個提交說明符之間的差異。遞交指標為參考，例如分支、標籤和遞交 ID。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 頁面上，選擇您要比較遞交、分支或附加標籤之遞交所在的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)，然後選擇 Compare commits (比較遞交)。

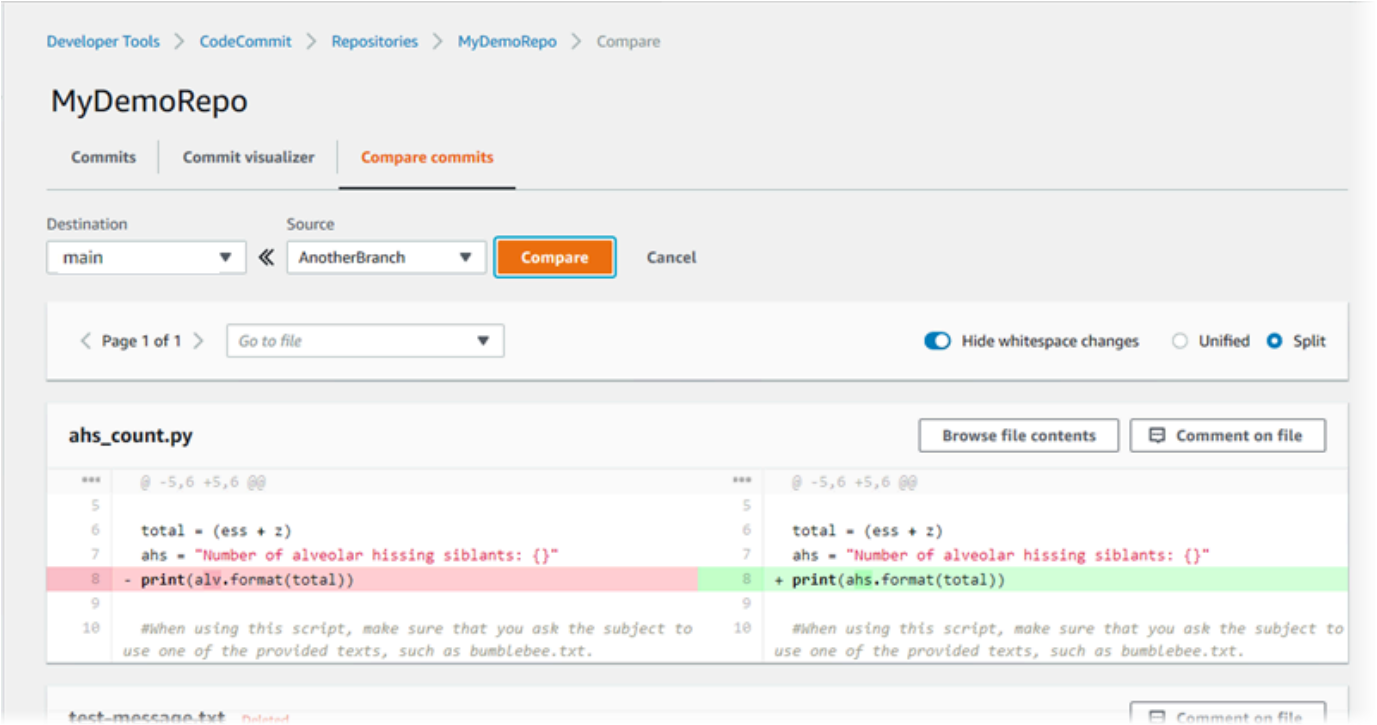


4. 使用方塊來比較兩個遞交指標。

   - 若要比較分支的頂端，請從清單中選擇分支名稱。這會從用於比較的該分支選取最新的遞交。
   - 若要將遞交與關聯的特定標籤比較，請從清單選擇標籤名稱 (如果有)。這會選取用於比較的之加上標籤的遞交。
   - 若要比較特定的遞交，請在方塊中輸入或貼上遞交 ID。若要取得完整的遞交 ID，請選擇導覽列中的 Commits (遞交)，並從清單複製遞交 ID。在「比較」提交頁面上，將完整的提交 ID 貼到文字方塊中，然後選擇「使用提交 ID」。



5. 選取指標之後，選擇 Compare (比較)。



您可以透過左右並排 (Split (分割) 檢視) 或內嵌 (Unified (統一) 檢視) 來顯示差異。您也可以隱藏或顯示空格的變更。

6. 若要清除您的比較選項，請選擇 Cancel (取消)。

評論提交 AWS CodeCommit

您可以使用 CodeCommit 控制台對存儲庫中的提交進行註釋，以及查看和回復其他用戶對提交的評論。這可協助您討論對儲存庫中進行的變更，包括：

- 為什麼進行變更。
- 是否需要更多變更。
- 是否應將變更合併到另一個分支。

您可以對整體遞交、對遞交中的檔案或檔案內的特定行或變更進行評論。您還可以通過選擇該行，然後在瀏覽器中複製生成的 URL 來鏈接到一行代碼。

> **Note**
>
> 若要取得最佳結果，請在以 IAM 使用者身分登入時使用註解。未針對使用根帳戶登入資料、聯合存取或臨時登入資料登入的使用者最佳化評論功能。

主題

查看存儲庫中提交的註釋

您可以使用 CodeCommit 控制台查看提交的註釋。

檢視對遞交的評論

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢閱對遞交評論的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)。選擇您想要檢視任何評論的遞交之遞交 ID。

   隨即會顯示該遞交的頁面，以及任何評論。

在存儲庫中添加和回復提交的註釋

您可以使用 CodeCommit 控制台將註釋添加到提交和父代的比較中，或者添加到兩個指定的提交之間的比較。您也可以使用表情符號、自己的留言或兩者來回覆留言。

在提交（控制台）上添加和回復評論

您可以使用文字和表情符號在提交中新增和回覆註解。您的註解和表情符號會標記為屬於您用來登入主控台的 IAM 使用者或角色。

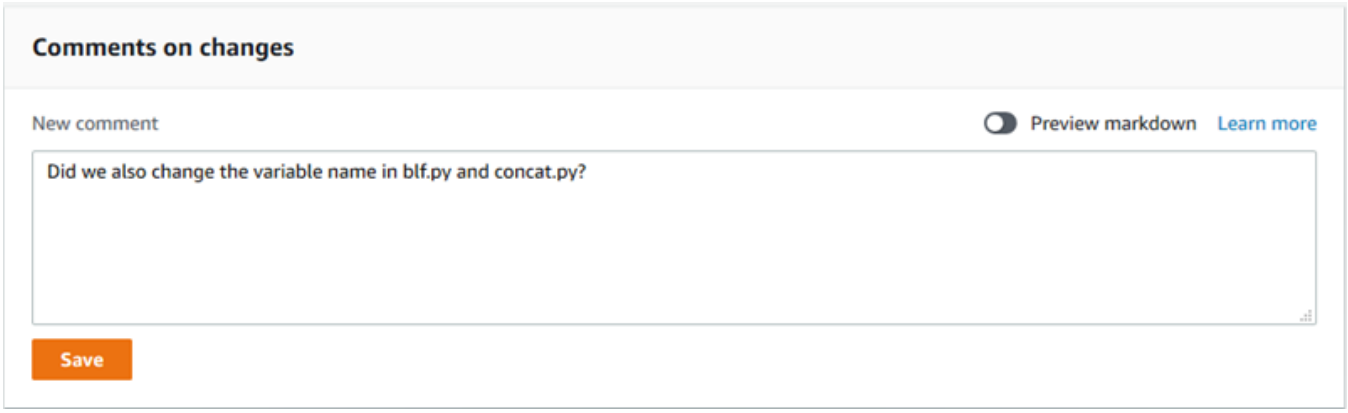
新增和回覆對遞交的評論

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home

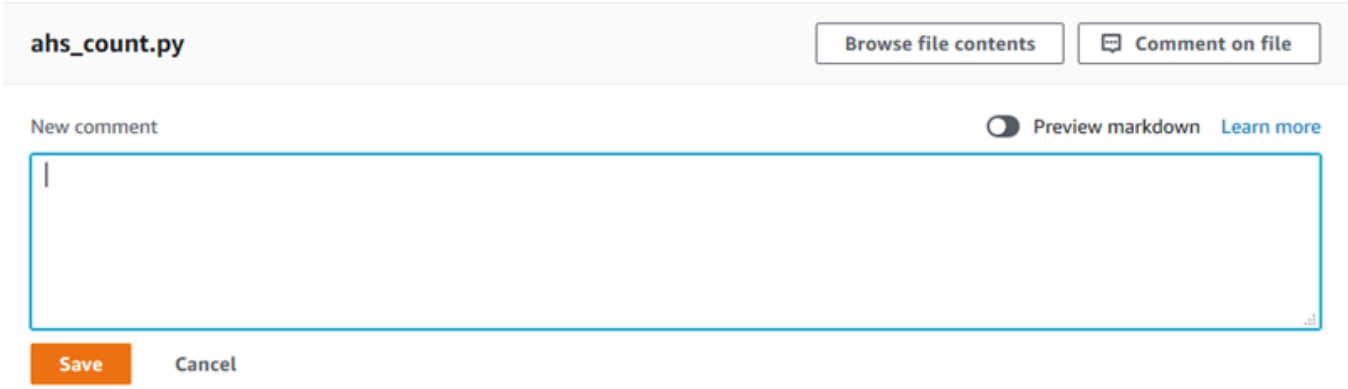
2. 在 Repositories (儲存庫) 中，選擇您要對遞交評論所在的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)。選擇您想要新增或回覆評論所在的遞交之遞交 ID。

   隨即會顯示該遞交的頁面，以及任何評論。
4. 若要新增評論，請執行以下其中一項：

   - 若要新增一般評論，請在 Comments on changes (對變更加上評論) 中，輸入評論，然後選擇 Save (儲存)。您可以使用 Markdown，或也可以純文字輸入您的評論。

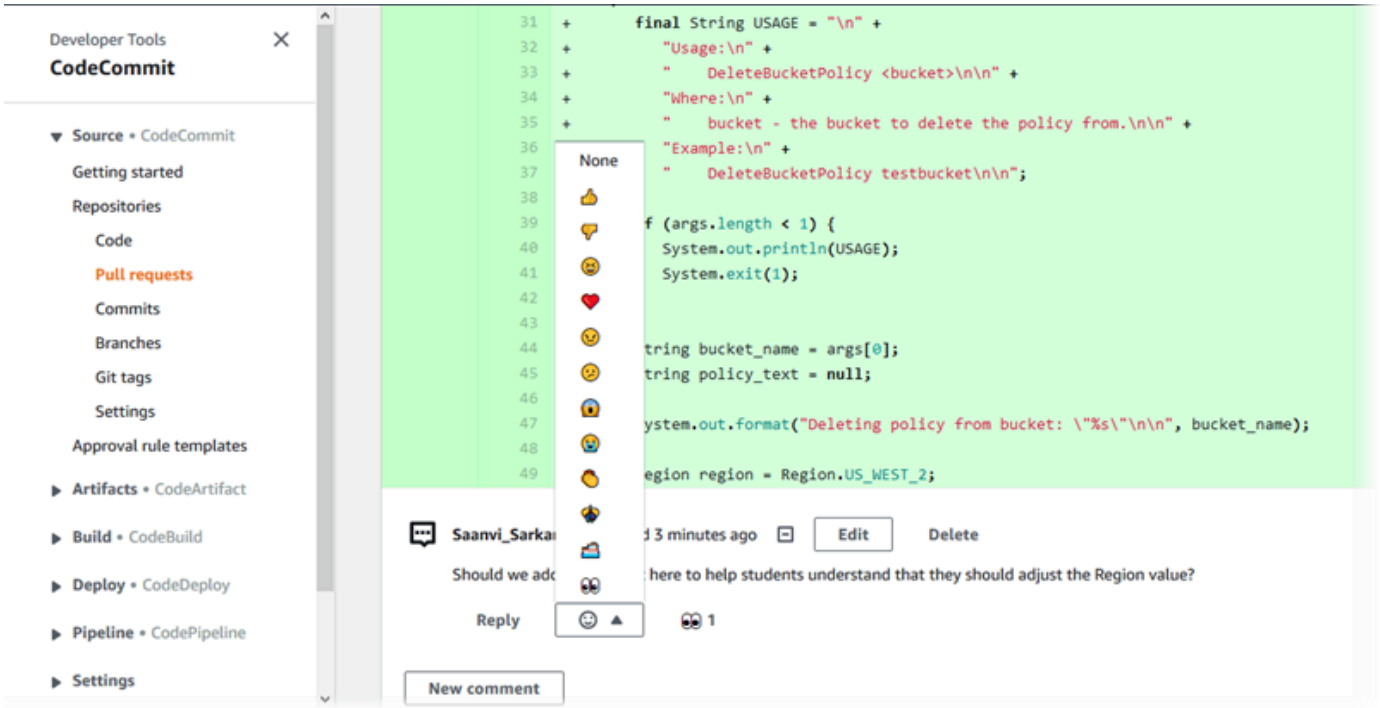
   - 若要對遞交中的檔案新增評論，請找到檔案的名稱。選擇 Comment on file (對檔案加上評論)，輸入您的評論，然後選擇 Save (儲存)。

   - 若要對遞交中變更的一行新增評論，請前往變更出現的行。選擇評論氣泡
     輸入您的評論，然後選擇 Save (儲存)。

> **Note**
>
> 在儲存評論之後，您可以編輯評論，也可以刪除其內容。該評論將保留一則訊息，指出內容已遭刪除。考慮為您的評論使用 Preview markdown (預覽 Markdown) 模式，之後再進行儲存。

5. 若要回覆對遞交的評論，請選擇 Reply (回覆)。若要回覆含有表情符號的留言，請從清單中選擇您想要的表情符號。每個留言只能選擇一個表情符號。如果您想要變更表情符號反應，請從清單中選擇不同的表情符號反應，或選擇「無」以移除您的反應。

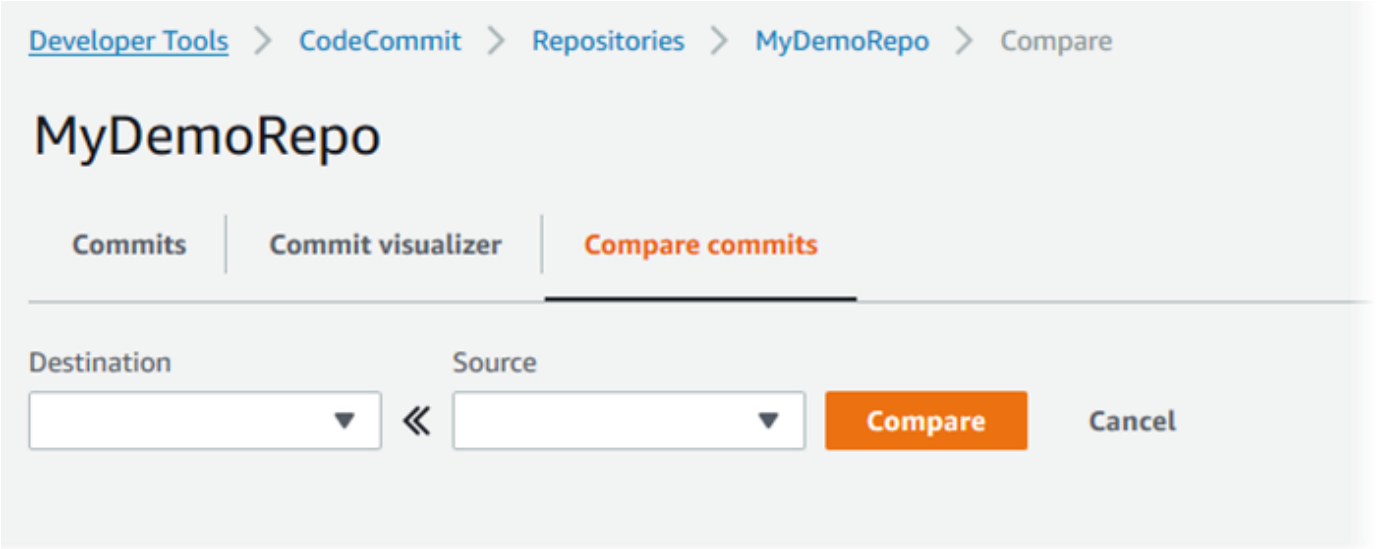


比較兩個提交說明符時添加和回復註釋

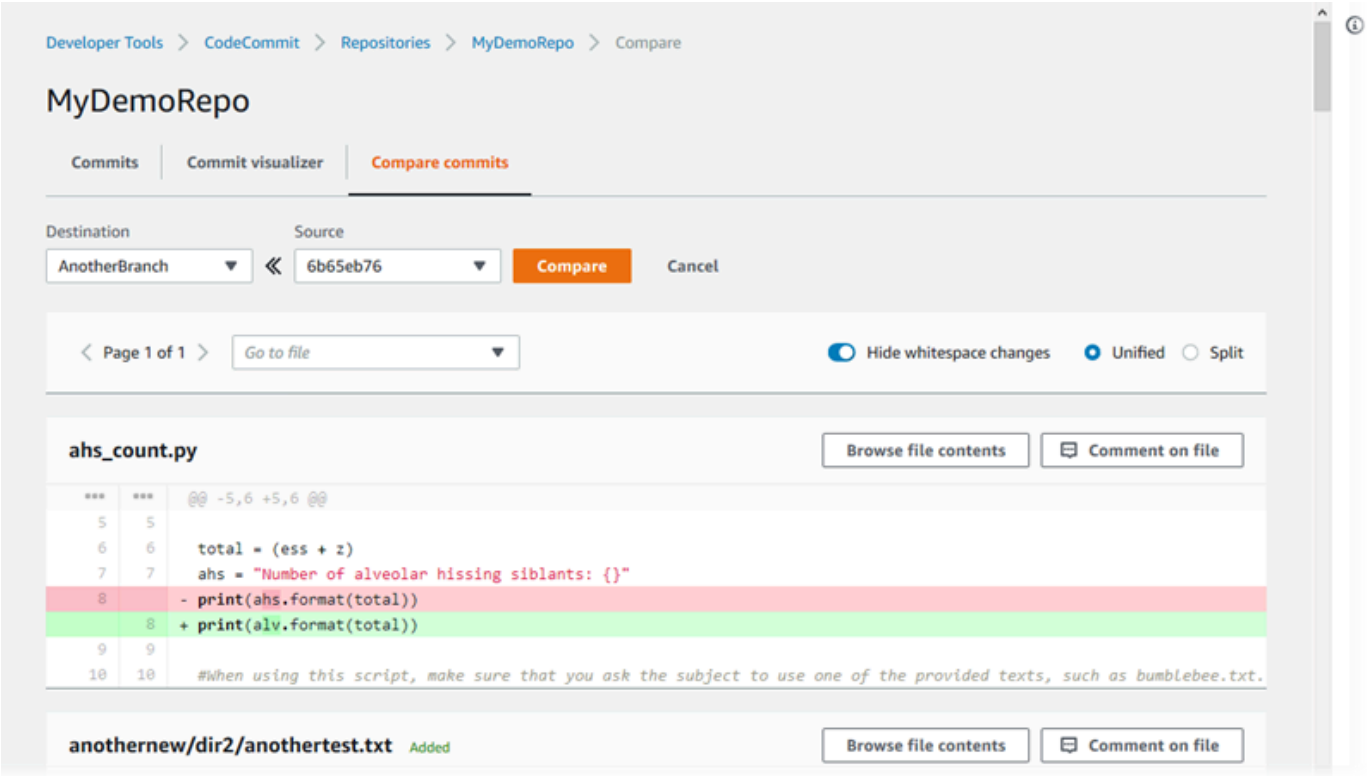
您可以對分支、標籤或遞交之間的比較新增評論。

比較遞交指標時新增和回覆評論

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要比較遞交、分支或附加標籤之遞交所在的儲存庫。
3. 在導覽窗格中，選擇 Commits (遞交)，然後選擇 Compare commits (比較遞交) 索引標籤。



4. 使用 Destination (目的地) 和 Source (來源) 欄位來比較兩個遞交指標。使用下拉式清單或貼上遞交 ID。選擇 Compare (比較)。



5. 執行下列其中一項或多項：

   - 若要將新增評論到檔案或行，請選擇評論氣泡

     。
   - 若要對比較的變更新增一般評論，請移至 Comments on changes (對變更的評論)。

檢視、新增、更新和回覆傳送 ()AWS CLI

您可以透過執行下列命令來檢視、新增、回覆、更新和刪除評論的內容：

- 若要檢視兩個遞交間比較的評論，請執行 get-comments-for-compared-commit。
- 若要檢視評論的詳細資訊，請執行 get-comment。
- 若要刪除您建立的評論內容，請執行 delete-comment-content。
- 若要對兩個遞交間的比較建立評論，請執行 post-comment-for-compared-commit。
- 若要更新評論，請執行 update-comment。
- 若要回覆註解，請執行post-comment-reply。
- 若要使用表情符號回覆留言，請執行put-comment-reaction。
- 若要檢視表情符號對留言的反應，請執行get-comment-reactions。

檢視對遞交的評論

1. 執行 get-comments-for-compared-commit 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選`--repository-name`項)。
   - 之後遞交的完整遞交 ID，以建立比較的方向性 (使用 `--after-commit-id option`)。
   - 之前遞交的完整遞交 ID，以建立比較的方向性 (使用 `--before-commit-id` 選項)。
   - (選用) 用來傳回下一個批次結果的列舉符記 (使用 `--next-token` 選項)。
   - (選用) 一個非負整數，用來限制傳回的結果 (使用 `--max-results` 選項)。

   例如，若要檢視名為的儲存庫中兩次提交比較時所做的註解 *MyDemoRepo*：

   ```
   aws codecommit get-comments-for-compared-commit --repository-name MyDemoRepo --
   before-commit-ID 6e147360EXAMPLE --after-commit-id 317f8570EXAMPLE
   ```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
      "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"
```

```
}
```

檢視對遞交評論的詳細資訊

1. 執行 get-comment 命令，指定系統產生的評論 ID。例如：

```
aws codecommit get-comment --comment-id ff30b348EXAMPLEb9aa670f
```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
{
   "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
         }
   }
}
```

刪除對遞交評論的內容

1. 執行 delete-comment-content 命令，指定系統產生的評論 ID。例如：

```
aws codecommit delete-comment-content --comment-id ff30b348EXAMPLEb9aa670f
```

> **Note**
>
> 如果您已套用 AWSCodeCommitFullAccess 原則，或將DeleteCommentContent權限設定為 [允許]，您才能刪除註解的內容。

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
   "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
         }
   }
}
```

建立對遞交的評論

1. 執行 post-comment-for-compared-commit 命令，並指定：

   - CodeCommit 存放庫的名稱 (含選--repository-name項)。
   - 之後遞交的完整遞交 ID，以建立比較的方向性 (使用 --after-commit-id 選項)。
   - 之前遞交的完整遞交 ID，以建立比較的方向性 (使用 --before-commit-id 選項)。
   - 唯一的用戶端產生冪等符記 (使用 --client-request-token 選項)。
   - 評論的內容 (使用 --content 選項)。
   - 可放置評論位置的清單位置相關資訊，包括：
     - 要比較的檔案名稱，包括其副檔名和子目錄 (如果有) (使用 filePath 屬性)。
     - 比較檔案內變更的行號 (使用 filePosition 屬性)。

- 對變更的評論在來源與目的地分支之間的比較中為之前或之後 (使用 relativeFileVersion 屬性)。

例如，要添加註釋 *###############* 在名為的存儲庫中的兩個提交之間進行比較時對 *cl_sample.js* 文件的更改 *MyDemoRepo*：

```
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. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
        "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"
 }
```

更新對遞交的評論

1. 執行 update-comment 命令，指定系統產生的評論 ID 和您要用來取代任何現有內容的內容。

   例如，要添加內容 *################* 若要加上識別碼為 *442* B498 的註解，請執行下列動作：

```
aws codecommit update-comment --comment-id 442b498bEXAMPLE5756813 --content "Fixed
 as requested. I'll update the pull request."
```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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
          }
    }
}
```

回覆對遞交的評論

1. 若要對提取請求中的評論張貼回覆，請執行 post-comment-reply 命令，指定：

   - 您要回覆的評論由系統產生的 ID (使用 --in-reply-to 選項)。
   - 唯一的用戶端產生冪等符記 (使用 --client-request-token 選項)。
   - 回覆的內容 (使用 --content 選項)。

   例如，要添加回复 *## catch#########* 使用系統生成的標識符的評論示例 *B* 5678EFGH：

```
aws codecommit post-comment-reply --in-reply-to abcd1234EXAMPLEb5678efgh --
content "Good catch. I'll remove them." --client-request-token 123Example
```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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": []
    }
 }
```

使用表情符號回覆提交的留言

1. 若要使用表情符號回覆提取要求中的註解，或變更表情符號反應的值，請執行指put-comment-reaction令，並指定：

   - 系統產生的註解 ID，您要使用表情符號回覆的註解。
   - 您要新增或更新的反應值。可接受的值包括支援的表情符號、簡碼和 Unicode 值。

   中的表情符號支援下列值 CodeCommit：

表情符號	ShortCode	Unicode
#	:thumbsup:	U+1F44D
#	:thumbsdown:	U+1F44E
#	:smile:	U+1F604

表情符號	ShortCode	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
	無	U+0000

########## ABCD1234Example B5678efgh #####

```
aws codecommit put-comment-reaction --comment-id abcd1234EXAMPLEb5678efgh --
reaction-value :thumbsup:
```

2. 如果成功，此命令不會產生輸出。

若要檢視表情符號對留言的反應

1. 若要檢視留言的表情符號反應，包括使用這些表情符號回應的使用者，請執行指get-comment-reactions令，並指定系統產生的留言 ID。

   例如，要使用系統生成的標識符號為 *AB* CD1234 查看對評論的表情符號反應，例如 B5678efgh：

```
aws codecommit get-comment-reactions --comment-id abcd1234EXAMPLEb5678efgh
```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "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"
                ]
            }
        ]
    }
}
```

在中建立一個 Git 標籤 AWS CodeCommit

您可以使用 Git 標籤來標記遞交，以協助其他儲存庫使用者了解其重要性。若要在 CodeCommit 儲存庫中建立 Git 標籤，您可以從連線到儲存庫的本機存放庫使用 CodeCommit Git。在本機存放庫中建立 Git 標籤之後，您可以使用git push --tags將其推送至 CodeCommit 儲存庫。

如需詳細資訊，請參閱 檢視標籤詳情。

使用 Git 創建一個標籤

請依照下列步驟使用本機 CodeCommit 存放庫中的 Git，在儲存庫中建立 Git 標籤。

在這些步驟中，我們假設您已經將本地存放庫連接到 CodeCommit 存儲庫。如需說明，請參閱連接到儲存庫。

1. 運行git tag ***new-tag-name commit-id***命令，其中*new-tag-name*是新的 Git 標籤的名稱，提*# ID* 是與 Git 標籤關聯的提交的 ID。

   例如，以下命令會建立名為 `beta` 的 Git 標籤，並將它與遞交 ID `dc082f9a...af873b88` 建立關聯：

   ```
   git tag beta dc082f9a...af873b88
   ```

2. 要將新的 Git 標籤從本地回購推送到 CodeCommit 存儲庫，請運行該git push ***remote-name new-tag-name***命令，其中*####*是 CodeCommit 存儲庫的名稱，並且*new-tag-name*是新的 Git 標籤的名稱。

   例如，要將名為的新 Git 標籤推送`beta`到名為的 CodeCommit 存儲庫`origin`：

   ```
   git push origin beta
   ```

> **Note**
>
> 要將所有新的 Git 標籤從本地存儲庫推送到 CodeCommit 存儲庫，請運行git push --tags。為了確保您的本地 CodeCommit 存儲庫已使用存儲庫中的所有 Git 標籤進行更新，請運行git fetch後跟git fetch --tags。

如需更多選項，請參閱 Git 文件。

在中檢視 Git 標籤詳細資料 AWS CodeCommit

在 Git 中，標籤是一個標記，可套用到參考 (例如遞交)，以將它標示為對其他儲存庫使用者可能重要的資訊。例如，您可以使用標籤 **beta** 來標記屬於專案 Beta 版發行點的遞交。如需詳細資訊，請參閱 使用 Git 創建一個標籤。Git 標籤與儲存庫標籤不同。如需有關如何使用儲存庫標籤的詳細資訊，請參閱新增標籤至儲存庫。

您可以使用 AWS CodeCommit 主控台來檢視儲存庫中 Git 標籤的相關資訊，包括每個 Git 標籤所參考之提交的日期和提交訊息。從主控台，您可以使用儲存庫的預設分支標頭來比較標籤參考的遞交。如同任何其他遞交，您也可以檢視該 Git 標籤時間點的程式碼。

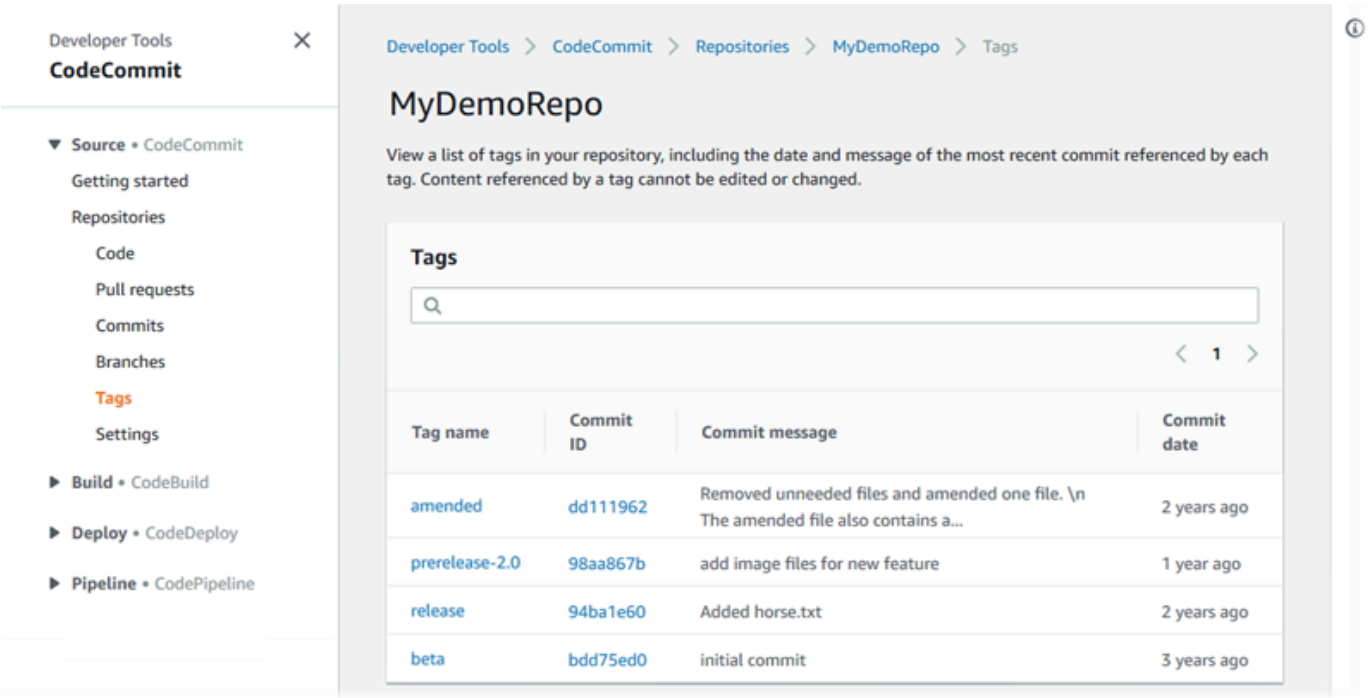
您也可以從終端機或命令列使用 Git 來檢視本機存放庫中 Git 標籤的詳細資料。

主題

檢視標籤詳細資料 (主控台)

使用主 AWS CodeCommit 控台快速檢視儲存庫的 Git 標籤清單，以及 Git 標籤所參考之提交的詳細資料。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢視標籤所在的儲存庫名稱。
3. 在導覽窗格中，選擇 Git tags (Git 標籤)。



4. 執行以下任意一項：

   - 若要檢視程式碼在遞交時的狀態，請選擇 Git 標籤名稱。
   - 若要檢視遞交的詳細資訊 (包括完整的遞交訊息、遞交者和作者)，請選擇縮寫的遞交 ID。

檢視 Git 標籤詳細資料 (Git)

若要使用 Git 檢視本機存放庫中 Git 標籤的詳細資料，請執行下列其中一個指令：

- git tag 可檢視 Git 標籤名稱的清單。
- git show 可檢視特定 Git 標籤的詳細資訊。
- git LS-遠程查看有關存儲庫中 Git 標籤的信息。 CodeCommit

> **Note**
>
> 為了確保您的本地 CodeCommit 存儲庫已使用存儲庫中的所有 Git 標籤進行更新，請運行git fetch後跟git fetch --tags。

在以下步驟中，我們假設您已將本地存放庫連接到 CodeCommit 存儲庫。如需說明，請參閱連接到儲存庫。

若要檢視本機存放庫中的 Git 標籤清單

1. 執行 git tag 命令：

```
git tag
```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
beta
release
```

> **Note**
>
> 如果沒有定義任何標籤，git tag 不會傳回任何內容。

如需更多選項，請參閱 Git 文件。

若要檢視本機存放庫中 Git 標籤的相關資訊

1. 執行 git show ***tag-name*** 命令。例如，若要檢視名為 beta Git 標籤的詳細資訊，請執行：

```
git show beta
```

2. 如果此命令成功執行，您會看到類似如下的輸出產生：

```
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**
>
> 若要結束 Git 標籤資訊的輸出，請輸入 :q。

如需更多選項，請參閱 Git 文件。

若要檢視儲 CodeCommit 存庫中 Git 標籤的相關資訊

1. 執行 git ls-remote --tags 命令。

```
git ls-remote --tags
```

2. 如果成功，這個命令會在 CodeCommit 儲存庫中產生 Git 標籤的輸出清單：

```
129ce87a...70fbffba    refs/tags/beta
785de9bd...59b402d8    refs/tags/release
```

   如果沒有定義任何 Git 標籤，git ls-remote --tags 會傳回空白的行。

如需更多選項，請參閱 Git 文件。

在中刪除一個 Git 標籤 AWS CodeCommit

若要刪除儲存庫中的 CodeCommit Git 標籤，請使用連線至儲存庫的本機存放庫中的 CodeCommit Git。

使用 Git 刪除一個 Git 標籤

請依照下列步驟使用本機存放庫中的 Git 來刪除儲存庫中的 CodeCommit Git 標籤。

這些步驟是假設您已經將本地存儲庫連接到 CodeCommit 存儲庫的假設編寫的。如需說明，請參閱連接到儲存庫。

1. 要從本地存儲庫中刪除 Git 標籤，請運行git tag -d ***tag-name***命令，其中標###是要刪除的 Git 標籤的名稱。

> **Tip**
>
> 若要取得 Git 標籤名稱的清單，請執行 git tag。

例如，要刪除名為的本地存儲庫中的 Git 標籤`beta`：

```
git tag -d beta
```

2. 要從 CodeCommit 存儲庫中刪除 Git 標籤，請運行git push ***`remote-name`*** --delete ***`tag-name`***命令，其中`####`是本地存儲庫用於 CodeCommit 存儲庫的暱稱，標`###`是要從存儲庫中刪除的 Git 標籤的名稱。 CodeCommit

   > **Tip**
   >
   > 若要取得 CodeCommit 儲存庫名稱及其 URL 的清單，請執行git remote -v指令。

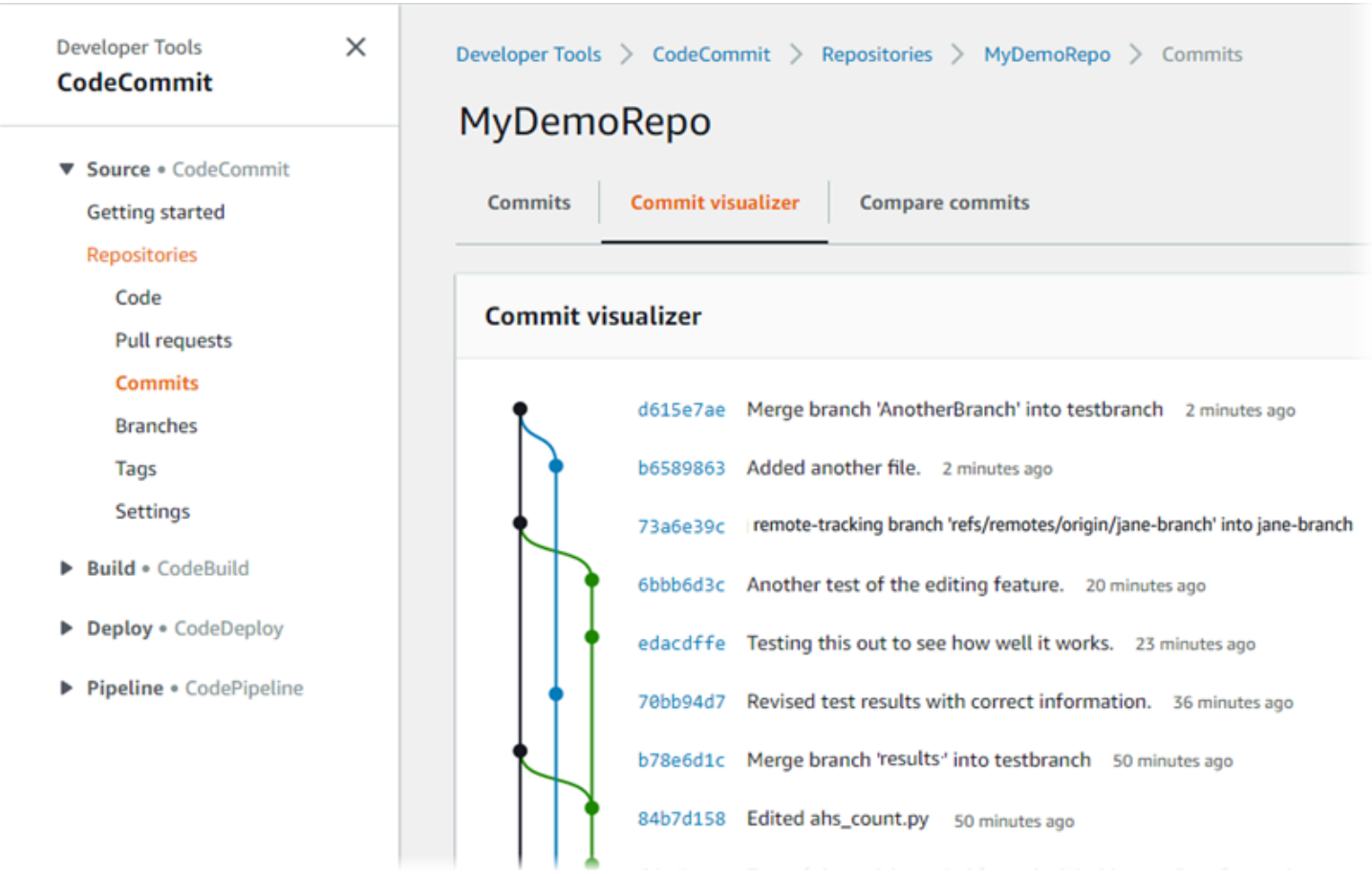
   例如，若要刪除名為的儲存庫`beta`中名為的 CodeCommit Git 標籤`origin`：

   ```
   git push origin --delete beta
   ```

使用 AWS CodeCommit 儲存庫中的分支

什麼是分支？ 在 Git 中，分支是指向提交的指針或引用。在開發時，分支可讓您的組織工作更輕鬆。您可以使用分支來分隔新版或不同版本檔案的工作，而不會影響其他分支中的工作。您可以使用分支來開發新功能、存放來自特定遞交的特定專案版本等等。當您創建第一次提交時，將為您創建一個默認分支。此預設分支是當使用者複製存放庫時，在本機存放庫 (Repos) 中用作基底或預設分支的分支。該默認分支的名稱取決於您創建第一次提交的方式。如果您使用 CodeCommit 控制台、或其中一個 SDK 將第一個檔案新增至儲存庫，則該預設分支的名稱為 main。 AWS CLI這是本指南範例中使用的預設分支名稱。如果您使用 Git 客戶端推送第一次提交，則默認分支的名稱就是 Git 客戶端指定為其默認分支的名稱。考慮將 Git 客戶端配置為使用 main 作為初始分支的名稱。

在中 CodeCommit，您可以變更存放庫的預設分支。您也可以建立和刪除分支，以及檢視分支的詳細資訊。您可以快速比較某個分支和預設分支 (或任何兩個分支) 的差異。要查看存儲庫中分支和合併的歷史記錄，可以使用提交視覺效果，如下圖所示。



如需有關在中使用儲存庫其他方面的資訊 CodeCommit 使用儲存庫使用檔案，請參閱使用提取請求使用提交、、和使用者偏好設定。

主題

在中創建一個分支 AWS CodeCommit

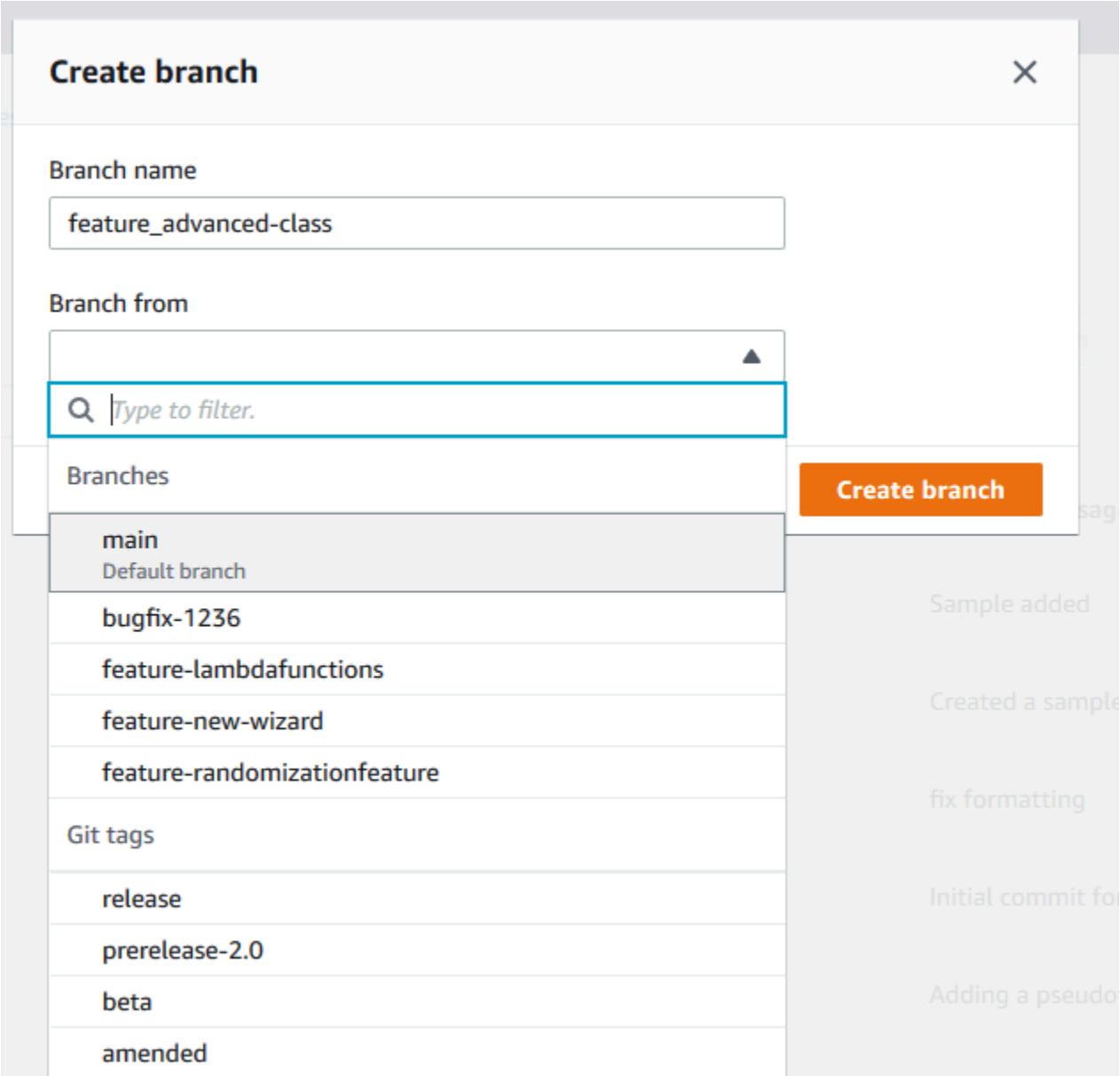
您可以使用 CodeCommit 主控台或 AWS CLI 建立儲存庫的分支。這是要區隔新的或不同版本的檔案，而不影響預設分支中工作的快速方式。在 CodeCommit 控制台中創建分支後，您必須將該更改拉到本地回購中。或者，您可以在本地創建一個分支，然後從連接到存儲庫的本地存儲庫中使用 CodeCommit Git 來推送該更改。

主題

創建一個分支（控制台）

您可以使用 CodeCommit 控制台在 CodeCommit 存儲庫中創建分支。當使用者下一次從儲存庫提取變更之後，他們會看到新的分支。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要建立分支所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Branches (分支)。
4. 選擇 Create branch (建立分支)。



在 Branch name (分支名稱) 中，輸入分支的名稱。在 Branch from (分支來源) 中，從清單中選擇分支或標籤，或貼上遞交 ID。選擇 Create branch (建立分支)。

創建一個分支（Git）

請按照以下步驟使用本地倉庫中的 Git 在本地存儲庫中創建一個分支，然後將該分支推送到 CodeCommit 存儲庫。

這些步驟是假設您已經將本地存儲庫連接到 CodeCommit 存儲庫的假設編寫的。如需說明，請參閱連接到儲存庫。

1. 通過運行git checkout -b ***new-branch-name***命令在本地回購中創建一個分支，其中*new-branch-name*是新分支的名稱。

   例如，以下命令創建一個在本地回購MyNewBranch中命名的分支：

   ```
   git checkout -b MyNewBranch
   ```

2. 若要將新分支從本機存放庫推送至 CodeCommit 儲存庫，請執行指git push令，同時指定***remote-name***和***new-branch-name***.

   例如，要將名為的本地回購中的新分支推送MyNewBranch到帶有暱稱的 CodeCommit 存儲庫中origin：

   ```
   git push origin MyNewBranch
   ```

> **Note**
>
> 如果您將 -u 選項新增至 git push (例如，git push -u origin main)，那麼，在未來您可以執行 git push 而不帶 *remote-name branch-name*。上游追蹤資訊即已設定。若要取得上游追蹤資訊，請執行 git remote show ***remote-name*** (例如，git remote show origin)。
> 若要查看所有本機和遠端追蹤分支的清單，請執行 git branch --all。
> 要在本地存儲庫中設置連接到 CodeCommit 存儲庫中分支的分支，請運行git checkout ***remote-branch-name***。

如需更多選項，請參閱 Git 文件。

創建一個分支（AWS CLI）

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

請依照下列步驟使 AWS CLI 用在 CodeCommit 存放庫中建立分支，然後將該分支推送至 CodeCommit 存放庫。如需建立初始提交並指定空白儲存庫預設分支名稱的步驟，請參閱使用 AWS CLI.

1. 執行 create-branch 命令，並指定：

   - 建立分支的 CodeCommit 儲存庫名稱 (使用選--repository-name項)。

> **Note**
>
> 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

- 新分支的名稱 (使用 --branch-name 選項)。
- 新分支指向的遞交 ID (使用 --commit-id 選項)。

例如，要創建一個名為的分支，MyNewBranch該分支指向名為的 CodeCommit 存儲庫317f8570EXAMPLE中提交 IDMyDemoRepo：

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-name MyNewBranch
 --commit-id 317f8570EXAMPLE
```

只有在發生錯誤時，此命令才會產生輸出。

2. 要使用新的遠程分支名稱更新本地存 CodeCommit 儲庫中可用的存儲庫分支列表，請運行git remote update ***remote-name***。

   例如，要使用暱稱更新 CodeCommit 存儲庫的可用分支列表origin：

   ```
   git remote update origin
   ```

   > **Note**
   >
   > 或者，您也可以執行 git fetch 命令。您也可以通過運行查看所有遠程分支git branch --all，但是在更新本地存儲庫的列表之前，您創建的遠程分支不會出現在列表中。
   > 如需更多選項，請參閱 Git 文件。

3. 要在本地存儲庫中設置連接到 CodeCommit 存儲庫中的新分支的分支，請運行git checkout ***remote-branch-name***。

> **Note**
>
> 若要取得 CodeCommit 儲存庫名稱及其 URL 的清單，請執行git remote -v指令。

限制推送和合併到分支 AWS CodeCommit

根據預設，任何具有足夠權限將程式碼推送至 CodeCommit 儲存庫的存放庫使用者，都可以對該儲存庫中的任何分支做出貢獻。無論您將分支新增到儲存庫的方式為何：使用主控台、命令列或 Git，都是如此。不過，您可能想要設定分支，使得只有部分儲存庫使用者可以將程式碼推送或合併至該分支。例如，您可能想要設定用於生產程式碼的一個分支，使得只有一組資深開發人員可以將變更推送或合併至該分支。其他開發人員仍可以從分支提取、製作自己的分支，以及建立提取請求，但無法將變更推送或合併至該分支。您可以透過建立條件政策來設定此存取權，該政策使用 IAM 中一或多個分支的內容金鑰。

> **Note**
>
> 若要完成本主題中的某些程序，您必須使用具有足夠權限來設定和套用 IAM 政策的管理員使用者登入。如需詳細資訊，請參閱建立 IAM 管理員使用者和群組。

主題

設定 IAM 政策以限制推送和合併到分支

您可以在 IAM 中建立政策，防止使用者更新分支，包括將提交推送至分支，以及將提取請求合併到分支。若要這樣做，您的政策會使用條件式陳述式，使得只有在符合條件時，才會套用 Deny 陳述式的效果。您在 Deny 陳述式中包含的 API 會決定不允許的動作。您可以將此政策設定為僅套用至儲存庫中的一個分支、儲存庫中的多個分支，或套用至 Amazon Web Services 帳戶中所有儲存庫中符合條件的所有分支。

為分支建立條件式政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇政策。
3. 選擇 Create policy (建立政策)。

4. 選擇 JSON，然後將下列範例政策貼上。將 Resource 的值取代為包含您要限制存取分支儲存庫的 ARN。將 codecommit:References 的值以您要限制存取之一或多個分支的參考取代。例如，此原則會拒絕推送認可、合併分支、刪除分支、合併提取要求，以及將檔案新增至名為的分支，以*main*及名為以下命名的儲存庫*prod*中名為的分支：*MyDemoRepo*

```
{
    "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 中的分支只是標頭遞交 SHA-1 值的指標 (參考)，這是條件使用 References 的原因。在效果為 Deny 且 GitPush 是其中一個動作的任何政策中，Null 陳述式為必要。這是必需的，因為 Git git-receive-pack 在將更改從本地回購推送到 CodeCommit.

> **Tip**
>
> 若要建立適用於 Amazon Web Services 帳戶中所有儲存庫中名為 main 的所有分支的政策，請將儲存庫 ARN 的`Resource`值變更為星號 (*)。

5. 選擇檢閱政策。更正政策陳述式中的任何錯誤，然後繼續 Create policy (建立政策)。
6. 當 JSON 經過驗證，隨即會顯示 Create policy (建立政策) 頁面。Summary (摘要) 區段中會出現警告，建議您此政策將不會授予許可。這是預期的行為。
   - 在 Name (名稱) 中，輸入此政策的名稱，例如 **`DenyChangesToMain`**。
   - 在 Description (描述) 中，輸入政策目的之描述。此為選用操作，但建議您採用。
   - 選擇建立政策。

將 IAM 政策套用至 IAM 群組或角色

您已建立限制推送和合併到分支的政策，但在您將其套用至 IAM 使用者、群組或角色之前，該政策才會生效。最佳做法是考慮將政策套用至 IAM 群組或角色。將政策套用至個別 IAM 使用者並不能很好地擴展。

將條件式政策套用至群組或角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在瀏覽窗格中，如果要將政策套用至 IAM 群組，請選擇 [群組]。如果要將政策套用至使用者所擔任的角色，請選擇 [角色]。選擇群組或角色的名稱。
3. 在 [權限] 索引標籤上，選擇 [連接政策]。
4. 選取您從策略清單建立的條件政策，然後選擇 Attach policy (附加政策)。

如需詳細資訊，請參閱附加和取消 IAM 政策。

測試原則

您應該測試您對群組或角色套用政策的效果，以確保它可如預期運作。您有很多種方法可以執行此操作。例如，若要測試類似以上所示的政策，您可以：

- 使用 IAM 使用者 (身為已套用政策的 IAM 群組成員) 登入 CodeCommit 主控台，或擔任已套用該政策的角色。在主控台中，於套用限制的分支上新增檔案。嘗試將檔案儲存或上傳到該分支時，您應該會看到錯誤訊息。將檔案新增到不同分支。操作應該會成功。
- 使用 IAM 使用者 (身為已套用政策的 IAM 群組成員) 登入 CodeCommit 主控台，或擔任已套用該政策的角色。建立提取請求，其將合併到限制適用所在的分支。您應該可以建立提取請求，但在嘗試合併它時發生錯誤。
- 從終端或命令行，在適用限制的分支上創建一個提交，然後將該提交推送到 CodeCommit 存儲庫。您應該會看到錯誤訊息。從其他分支進行的遞交和推送應該可以照常運作。

檢視分行詳細資訊 AWS CodeCommit

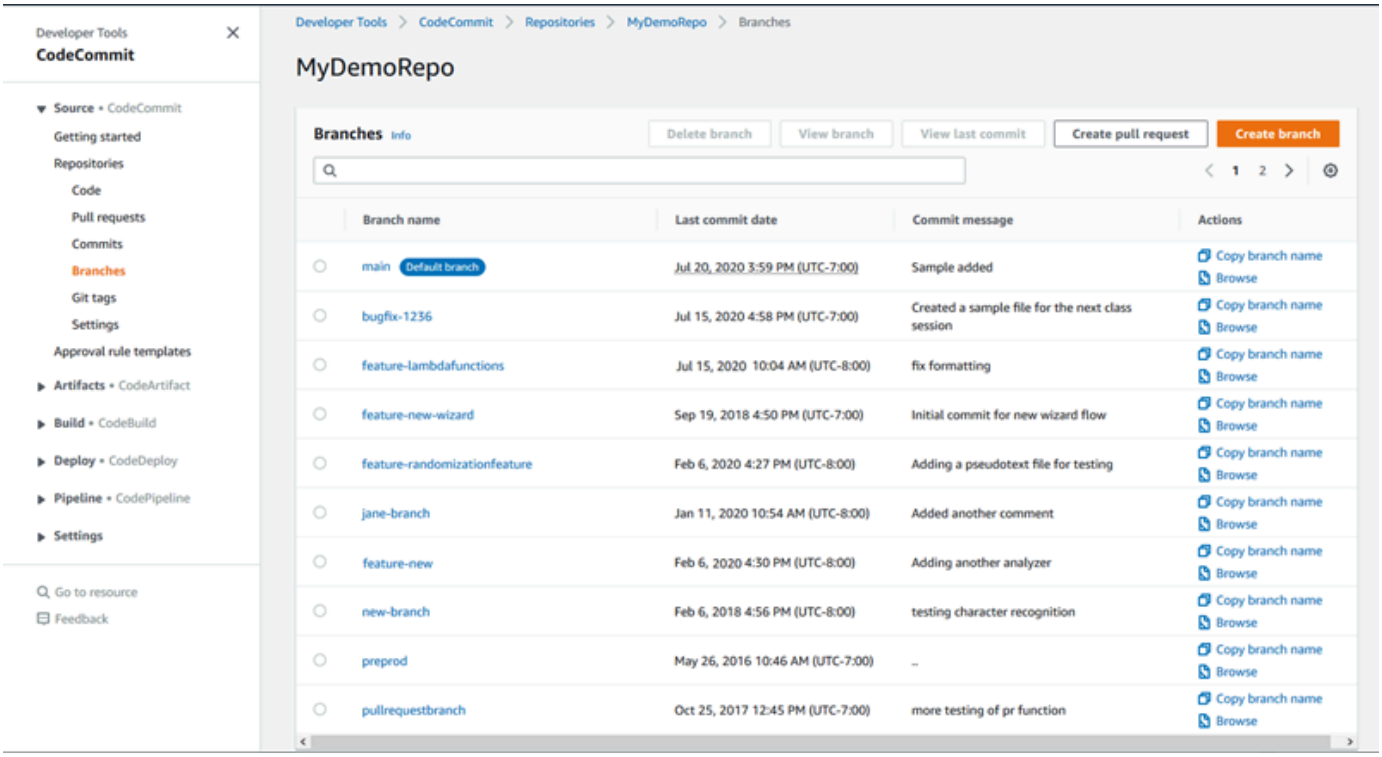
您可以使用 CodeCommit 控制台查看有關 CodeCommit 存儲庫中分支的詳細信息。您可以檢視上次對分支遞交的日期、遞交訊息等等。您也可以使用連線至儲存庫的本機存放庫中的 AWS CLI 或 CodeCommit Git。

主題

查看分支詳細信息（控制台）

使用 CodeCommit 控制台快速查看存儲庫的分支列表以及有關分支的詳細信息。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要檢視分支詳細資訊所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Branches (分支)。



4. 用作儲存庫預設分支的分支名稱會顯示在「預設」(D efault) 分支旁邊。若要查看對分支的最新遞交的詳細資訊，請選擇該分支，然後選擇 View last commit (檢視上次的遞交)。若要檢視分支中的檔案和程式碼，請選擇分支名稱。

檢視分支詳細資料 (Git)

若要使用本機存放庫中的 Git 來檢視 CodeCommit 儲存庫的本機和遠端追蹤分支的詳細資料，請執行git branch指令。

以下步驟是假設您已將本地存儲庫連接到 CodeCommit 存儲庫的假設編寫的。如需說明，請參閱連接到儲存庫。

1. 執行 git branch 命令，指定 --all 選項：

```
git branch --all
```

2. 如果此命令成功執行，您會看到類似如下的輸出傳回：

```
  MyNewBranch
* main
  remotes/origin/MyNewBranch
```

```
  remotes/origin/main
```

星號 (*) 會顯示在目前所開啟分支的名稱旁。後面的項目為遠端追蹤參考。

> **Tip**
>
> git branch 顯示本機分支。
> git branch -r 顯示遠端分支。
> git checkout ***existing-branch-name*** 會切換至指定的分支名稱，並且，如果之後立即執行 git branch，顯示它時會加上星號 (*)。
> git remote update ***remote-name***使用可用的 CodeCommit 存儲庫分支列表更新您的本地回購。若要取得 CodeCommit 儲存庫名稱及其 URL 的清單，請執行git remote -v指令。)

如需更多選項，請參閱 Git 文件。

檢視分行詳細資料 (AWS CLI)

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用 AWS CLI 檢視有關 CodeCommit 儲存庫中分支的詳細資訊，請執行下列一或多個命令：

- 要查看分支名稱列表，請運行列表分支。
- 若要檢視特定分支的相關資訊，請執行 get- 分支。

檢視分支名稱的清單

1. 運行list-branches命令，指定 CodeCommit 存儲庫的名稱（使用`--repository-name`選項）。

   > **Tip**
   >
   > 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

   例如，若要檢視名為的 CodeCommit 儲存庫中分支的詳細資訊MyDemoRepo：

   ```
   aws codecommit list-branches --repository-name MyDemoRepo
   ```

2. 如果成功，此命令會輸出 `branchNameList` 物件，每個分支一個項目。

以下是基於上述範例命令的一些範例輸出：

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

檢視特定分支的詳細資訊

1. 執行 get-branch 命令，並指定：

   - 儲存庫名稱 (使用 --repository-name 選項)。
   - 分支名稱 (使用 --branch-name 選項)。

   例如，若要檢視在名為的 CodeCommit 儲存庫MyNewBranch中命名的分支的相關資訊MyDemoRepo：

   ```
   aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
   ```

2. 如果成功，此命令會輸出分支的名稱和上次對分支進行遞交的 ID。

   以下是基於上述範例命令的一些範例輸出：

   ```
   {
       "branch": {
             "branchName": "MyNewBranch",
             "commitID": "317f8570EXAMPLE"
       }
   }
   ```

比較和合併分支 AWS CodeCommit

您可以使用 CodeCommit 控制台來比較 CodeCommit 存儲庫中的分支。比較分支有助於快速檢視某個分支和預設分支的差異，或檢視任兩個分支間的差異。

主題

將分支與默認分支進行比較

使用 CodeCommit 控制台快速查看分支和存儲庫的默認分支之間的差異。

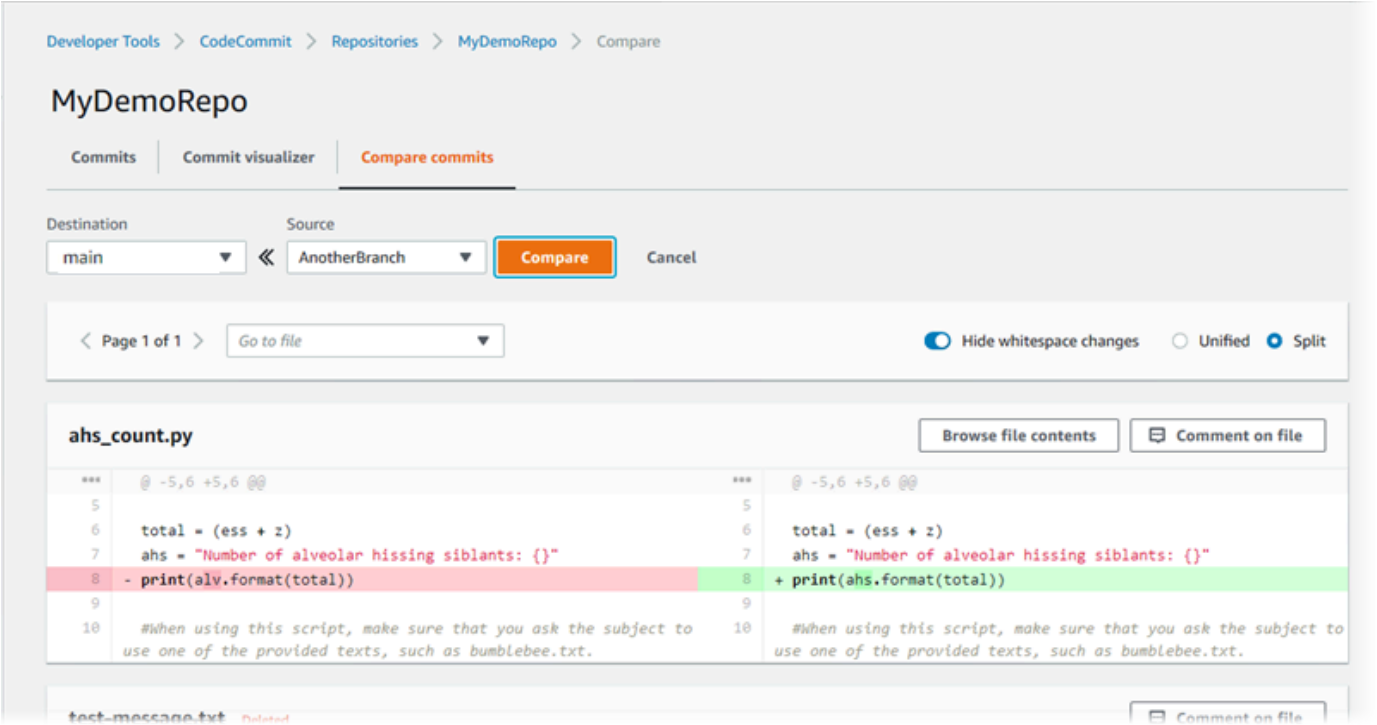
1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要比較分支所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Commits (遞交)，然後選擇 Compare commits (比較遞交) 索引標籤。
4. 在 Destination (目的地) 中，選擇預設分支的名稱。在 Source (來源) 中，選擇您想要與預設分支比較的分支。選擇 Compare (比較)。

比較兩個特定的分支

使用 CodeCommit 控制台查看要比較的兩個分支之間的差異。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要比較分支所在儲存庫的名稱。
3. 在導覽窗格中，選擇 Commits (遞交)，然後選擇 Compare commits (比較遞交) 索引標籤。
4. 在 Destination (目的地) 和 Source (來源) 中，選擇要比較的兩個分支，然後選擇 Compare (比較)。若要檢視變更的檔案清單，請展開變更的檔案清單。您可以透過左右並排 (分割檢視) 或內嵌 (統一檢視) 來檢視檔案中的變更。

> **Note**
>
> 如果您以 IAM 使用者身分登入，則可以設定並儲存喜好設定，以便檢視程式碼和其他主控台設定。如需詳細資訊，請參閱 使用者偏好設定。



合併兩個分支（AWS CLI）

您可以使用其中一個可用的合併策略來合併 CodeCommit 儲存庫中的兩個分支，方法是執行下列其中一個指令： AWS CLI

- 若要使用向前快轉合併策略來合併兩個分支，請執行 merge-branches-by-fast-forward 命令。
- 若要使用 squash 合併策略來合併兩個分支，請執行 merge-branches-by-squash 命令。
- 若要使用三向合併策略來合併兩個分支，請執行 merge-branches-by-three-way 命令。

您也可以執行 create-unreferenced-merge-commit 命令來測試合併。如需詳細資訊，請參閱解決提取請求中的衝突。

> **Note**
>
> 若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用 AWS CLI 合併 CodeCommit 儲存庫中的兩個分支

1. 若要使用向前快轉合併策略來合併兩個分支，請執行 merge-branches-by-fast-forward 命令，並指定：

   - 包含您要合併之變更的來源分支名稱 (使用 --source-commit-specifier 選項)。
   - 您要合併變更的目的地分支名稱 (使用 --destination-commit-specifier 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。

   例如，要將名為*#### -1234* 的源分支合併到名為 *prepro* d 的存儲庫中的目標分支：*MyDemoRepo*

   ```
   aws codecommit merge-branches-by-fast-forward --source-commit-specifier bugfix-bug1234 --destination-commit-specifier preprod --repository-name MyDemoRepo
   ```

   如果此命令成功執行，您會看到類似如下的輸出產生：

   ```
   {
       "commitId": "4f178133EXAMPLE",
       "treeId": "389765daEXAMPLE"
   }
   ```

2. 若要使用 squash 合併策略來合併兩個分支，請執行 merge-branches-by-squash 命令，指定：

   - 包含您要合併之變更的來源分支名稱 (使用 --source-commit-specifier 選項)。
   - 您要合併變更的目的地分支名稱 (使用 --destination-commit-specifier 選項)。
   - 儲存庫的名稱 (使用 --repository-name 選項)。
   - 要包含的遞交訊息 (使用 --commit-message 選項)。
   - 用於遞交的名稱 (使用 --name 選項)。
   - 用於遞交的電子郵件地址 (使用 --email 選項)。

   例如，要將名為錯誤修正 *-bug1234 ##########*正的目標分支合併到名為錯*#*修正的存儲庫中：*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
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

3. 若要使用三向合併策略來合併兩個分支，請執行 merge-branches-by-three-way 命令，並指定：

- 包含您要合併之變更的來源分支名稱 (使用 --source-commit-specifier 選項)。
- 您要合併變更的目的地分支名稱 (使用 --destination-commit-specifier 選項)。
- 儲存庫的名稱 (使用 --repository-name 選項)。
- 要包含的遞交訊息 (使用 --commit-message 選項)。
- 用於遞交的名稱 (使用 --name 選項)。
- 用於遞交的電子郵件地址 (使用 --email 選項)。

例如，要將名為 *main* 的源分支與名為 *bug ## -1234* 的目標分支合併到一個名為：*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
```

如果此命令成功執行，您會看到類似如下的輸出產生：

```
{
    "commitId": "4f178133EXAMPLE",
    "treeId": "389765daEXAMPLE"
}
```

變更分支設定 AWS CodeCommit

您可以在 AWS CodeCommit 主控台中變更要用作預設分支的分支，或使用 AWS CLI. 例如，如果您使用 Git 客戶端創建了第一次提交，該客戶端將默認分支設置為 master，則可以創建一個名為 main 的分支，然後更改分支設置，以便將新分支設置為存儲庫的默認分支。要更改其他分支設置，您可以使用連接到存儲庫的本地存儲庫中的 CodeCommit Git。

主題

更改默認分支（控制台）

您可以在 AWS CodeCommit 控制台中指定哪個分支是 CodeCommit 存儲庫中的默認分支。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要變更設定所在的儲存庫名稱。
3. 在導覽窗格中，選擇設定。
4. 在 Default branch (預設分支) 中選擇分支下拉式清單，然後選擇不同的分支。選擇儲存。

> **Tip**
>
> - 如果您在下拉式清單中沒有看到其他分支，表示您尚未建立任何其他分支。如果存放庫只有一個分支，則無法變更存放庫的預設分支。如需詳細資訊，請參閱 在中創建一個分支 AWS CodeCommit。
> - 如果您看不到 [預設分支] 區段，而是看到通知規則和連線的項目，則表示您位於主控台的 [一般設定] 功能表中。儲存庫的設定選單會列在與程式碼和提取要求相同層級的儲存庫下。

更改默認分支（AWS CLI）

若要搭配使用 AWS CLI 指令 CodeCommit，請安裝 AWS CLI. 如需詳細資訊，請參閱 命令列參考。

若要使用 AWS CLI 變更儲存庫中存放庫的分支設定，請執行下列命令： CodeCommit

- update-default-branch以變更預設的分支。

變更預設的分支

1. 執行 update-default-branch 命令，並指定：

   - 更新預設分支的 CodeCommit 儲存庫名稱 (使用選--repository-name項)。

     > **Tip**
     > 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

   - 新預設分支的名稱 (使用 --default-branch-name 選項)。

     > **Tip**
     > 要獲取分支的名稱，請運行列表分支命令。

2. 例如，要將默認分支更MyNewBranch改為 CodeCommit 名為MyDemoRepo：

   ```
   aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-
   name MyNewBranch
   ```

   只有在發生錯誤時，此命令才會產生輸出。

如需更多選項，請參閱 Git 文件。

刪除分支 AWS CodeCommit

您可以使用 CodeCommit 控制台刪除存儲庫中的分支。刪除中的分支 CodeCommit並不會刪除本地存儲庫中的該分支，因此用戶可能會繼續擁有該分支的副本，直到下次提取更改為止。要在本地刪除分支並將該更改推送到 CodeCommit 存儲庫，請從連接到存儲庫的本地存儲庫中使用 CodeCommit Git。

刪除分支不會刪除任何遞交，但會刪除對該分支中遞交的所有參考。如果您刪除分支 (其中包含尚未合併到儲存庫中另一個分支的遞交)，除非您有其完整的遞交 ID，否則無法擷取這些遞交。

> **Note**
>
> 您不能使用這個主題中的指示來刪除儲存庫的預設分支。如果您要刪除預設分支，必須建立分支，讓新分支成為預設分支，然後刪除舊分支。如需詳細資訊，請參閱 建立分支 及 變更分支設定。

主題

刪除分支（控制台）

您可以使用 CodeCommit 控制台刪除存 CodeCommit 儲庫中的分支。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在 Repositories (儲存庫) 中，選擇您要刪除分支所在的儲存庫名稱。
3. 在導覽窗格中，選擇 Branches (分支)。
4. 尋找您要刪除的分支名稱，然後選擇 Delete branch (刪除分支) 並確認您的選擇。

刪除分支（AWS CLI）

如果該 AWS CLI 分支不是 CodeCommit 存放庫的預設分支，您可以使用刪除存放庫中的分支。如需有關安裝和使用的更多資訊 AWS CLI，請參閱命令列參考。

1. 在終端機或命令列，執行 delete-branch 命令，並指定：
   - 要刪除分支的 CodeCommit 儲存庫名稱 (使用選--repository-name項)。

     > **Tip**
     >
     > 要獲取存儲 CodeCommit 庫的名稱，請運行列表庫命令。

   - 要刪除的分支的名稱 (使用 branch-name 選項)。

> **Tip**
>
> 要獲取分支的名稱，請運行列表分支命令。

2. 例如，要刪除名為的 CodeCommit 存儲庫MyNewBranch中名為的分支MyDemoRepo：

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

此命令會傳回有關已刪除分支的詳細資訊，包括已刪除的分支名稱和屬於分支標頭之遞交的完整遞交 ID。例如：

```
"deletedBranch": {
    "branchName": "MyNewBranch",
    "commitId": "317f8570EXAMPLE"
}
```

刪除一個分支（Git）

請按照以下步驟使用本地存儲庫中的 Git 刪除 CodeCommit 存儲庫中的分支。

這些步驟是假設您已經將本地存儲庫連接到 CodeCommit 存儲庫的假設編寫的。如需說明，請參閱連接到儲存庫。

1. 要從本地回購中刪除分支，請運行該git branch -D ***branch-name***命令，其中####是要刪除的分支的名稱。

> **Tip**
>
> 若要取得分支名稱的清單，請執行 git branch --all。

例如，要刪除名為以下內容的本地回購中的分支MyNewBranch：

```
git branch -D MyNewBranch
```

2. 要從 CodeCommit 存儲庫中刪除分支，請運行git push ***remote-name*** --delete ***branch-name***命令，其中####是本地存儲庫用於 CodeCommit 存儲庫的暱稱，而分###是要從存儲庫中刪除的分支的名稱。 CodeCommit

> **Tip**
>
> 若要取得 CodeCommit 儲存庫名稱及其 URL 的清單，請執行git remote -v指令。

例如，若要刪除 CodeCommit 儲存庫MyNewBranch中名為的分支origin：

```
git push origin --delete MyNewBranch
```

> **Tip**
>
> 如果分支是預設分支，則此命令不會刪除該分支。

如需更多選項，請參閱 Git 文件。

使用者偏好設定

您可以使用 AWS CodeCommit 主控台來設定部分預設設定。例如，您可以將檢視程式碼變更的偏好設定，變更為內嵌或以分割檢視。當您變更其中一項設定時，AWS CodeCommit 主控台會在瀏覽器中設定 Cookie，並在您每次使用主控台時自動存放和套用您的選擇。只要您使用該瀏覽器來存取 AWS CodeCommit 主控台，這些偏好設定就會套用到所有區域中的所有儲存庫。這些偏好設定不專屬於特定儲存庫或特定區域。這些偏好設定完全不影響您與 AWS CLI、AWS CodeCommit API 或其他與 AWS CodeCommit 互動的服務進行互動。

> **Note**
>
> 使用者偏好設定 Cookie 專屬於特定瀏覽器。如果您從瀏覽器清除 Cookie，則會清除偏好設定。同樣地，如果您使用不同的瀏覽器來存取儲存庫，該瀏覽器將無法存取其他瀏覽器的 Cookie。不會保留您的偏好設定。

使用者偏好設定包括：

- 當檢視程式碼的變更時，是否使用 Unified (統一) 或 Split (分割) 檢視，以及是否顯示或隱藏空格變更。
- 當檢視、編輯或編寫程式碼時，在程式碼編輯器視窗中是否使用淺色背景或深色背景。

偏好設定並沒有設定頁面。反之，每當您在主控台變更偏好設定時，例如，如何檢視程式碼變更，該變更就會儲存並適時套用。

遷移至 AWS CodeCommit

有一些方法可以將 GCodeCommit 儲存庫遷移到：複製、鏡射、遷移所有或僅某些分支等等。您也可以將電腦上的本機、無版本控制的內容遷移到 CodeCommit。

下列主題示範一些可用來遷移儲存庫的方式。您的步驟可能會有所不同，這取決於儲存庫的類型、樣式或複雜度，以及您對於要遷移的儲存庫和遷移方式所做的決策。對於非常大型的儲存庫，您可以考慮漸進遷移。

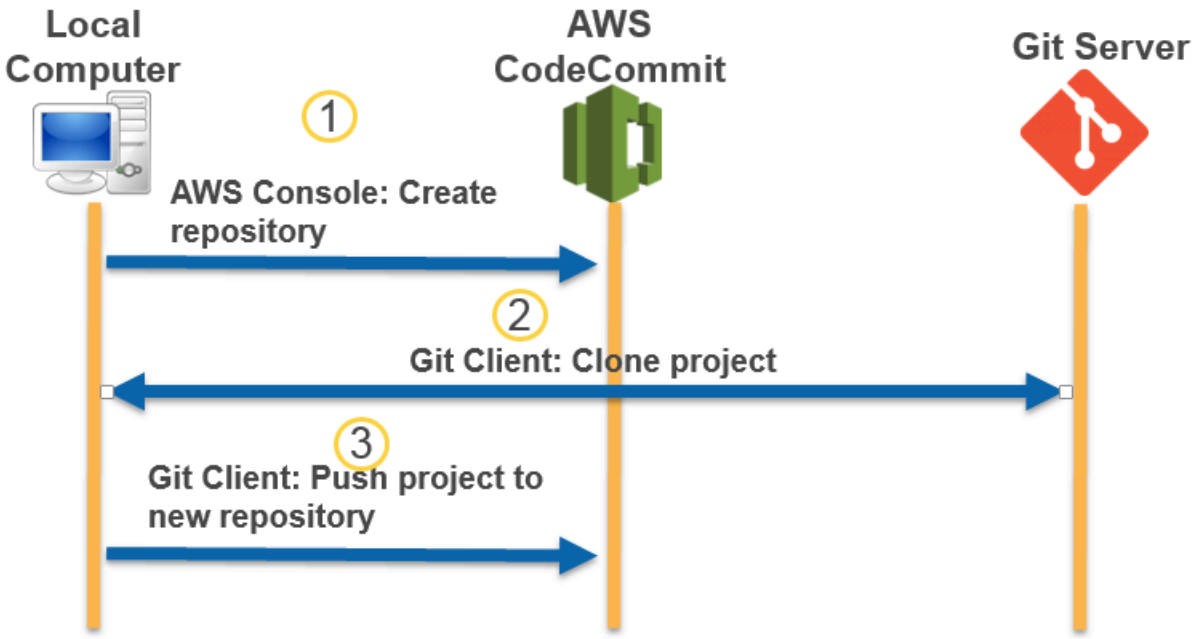
> **Note**
>
> 您可以從其他版本控制系統 (例如 Perforce、Subversion 或 TFS) 遷移到 CodeCommit，但必須先遷移到 Git。
> 如需更多選項，請參 Git 文件。
> 或者，您可以查看遷移到 Git中的專業 Git本書由斯科特·查肯和本·斯特勞布。

主題

將 Git 儲存庫遷移到 AWS CodeCommit

您可以將現有的 Git 儲存庫遷移到 CodeCommit 儲存庫。此主題中的程序將說明如何將於另一個 Git 儲存庫託管的專案遷移到 CodeCommit。隨著此程序，您會完成：

- 完成所需的初始設定 CodeCommit。
- 創建一個 CodeCommit 存儲庫。
- 克隆存儲庫並將其推送到 CodeCommit。
- 檢視儲 CodeCommit 存庫中的檔案。
- 與您的團隊共用 CodeCommit 儲存庫。



主題

步驟 0：存取所需的設定 CodeCommit

您必須先建立和設定 IAM 使用者 CodeCommit，並設定本機電腦以進行存取， CodeCommit 然後才能將儲存庫移轉至。您也應該安裝 AWS CLI 來管理 CodeCommit。雖然你可以在沒有它的情況下執行大多數 CodeCommit 任務，但在命令行或終端機上使用 Git 時，它AWS CLI提供了靈活性。

如果您已設置 CodeCommit，則可以跳到步驟 1：建立 CodeCommit 儲存庫。

若要建立和設定 IAM 使用者以存取 CodeCommit

1. 通過轉到 http://aws.amazon.com 並選擇註冊創建一個 Amazon Web Services 帳戶。

2. 在您的 Amazon Web Services 帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的 Amazon Web Services 帳戶中建立 IAM 使用者。

   > **Note**
   >
   > CodeCommit 需要AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS執行所需的動作 CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇 [使用者]，然後選擇要設定用於 CodeCommit 存取的 IAM 使用者。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從策略清單中，選取AWSCodeCommitPowerUser或另一個受管理的策略以進行 CodeCommit 存取。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的政策後，請選擇 [下一步：檢閱] 以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策以及與其他群組和使用者共用存放庫存取權的詳細資訊，請參閱共用儲存庫和AWS CodeCommit 的身分驗證與存取控制。

安裝及設定 AWS CLI

1. 在本機電腦上，下載並安裝 AWS CLI。這是從命令列進行互動 CodeCommit 的先決條件。我們建議您安裝AWS CLI版本 2。它是最新的主要版本，AWS CLI並支援所有最新功能。它是唯一支援使用 root 帳戶、同盟存取或暫時登入資料的git-remote-codecommit版本。AWS CLI

   如需詳細資訊，請參閱開始設定 AWS 命令列界面。

   > **Note**
   >
   > CodeCommit 僅適用於 1.7.38 及AWS CLI更高版本。根據最佳實務，安裝 AWS CLI 或將其升級到可用的最新版本。若要判斷您已安裝的 AWS CLI 版本，請執行 aws --version 命令。

若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 執行此命令以確認是否已安裝AWS CLI的命 CodeCommit 令。

```
aws codecommit help
```

此命令返回命 CodeCommit 令列表。

3. 使AWS CLI用configure指令來設定使用設定檔，如下所示：

```
aws configure
```

出現提示時，指定AWS要與之 IAM 使用者搭配使用的存取金鑰和AWS秘密存取金鑰 CodeCommit。此外，請務必指定存放庫的存在AWS 區域位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

```
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
```

如需建立和設定與 AWS CLI 搭配使用之描述檔的詳細資訊，請參閱下列內容：

- 命名設定檔
- 在中使用 IAM 角色 AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源AWS 區域，您必須AWS CLI使用預設的 Region 名稱重新配置。支援的預設地區名稱 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

若要取得有關 CodeCommit 和的更多資訊AWS 區域，請參閱區域和 Git 連線端點。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得登入資料？ 以及管理 IAM 使用者的存取金鑰。如需 AWS CLI 和描述檔的詳細資訊，請參閱具名描述檔。

接著，您必須安裝 Git。

- 對於 Linux，macOS 系統或 Unix：

  若要處理 CodeCommit 儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

  若要安裝 Git，我們建議使用 Git 下載等網站。

  > **Note**
  >
  > Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其使用的方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱疑難排解.

- 用於 Windows：

  若要處理 CodeCommit 儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

  要安裝 Git，我們建議您使用 Git 之類的網站。如果您使用此連結來安裝 Git，您可以接受所有安裝預設設定，但下列項目除外：

  - 在調整 PATH 環境步驟期間出現提示時，請從命令列選擇要使用 Git 的選項。
  - (選擇性) 如果您想要搭配使用 HTTPS 的認證協助程式 (AWS CLI而非為其設定 Git 認證) CodeCommit，請在 [設定額外選項] 頁面上，確定已清除 [啟用 Git 認證管理員] 選項。只有當 IAM 使用者設定 Git 認證時 CodeCommit ，Git 認證管理員才能相容。如需詳細資訊，請參閱 適用於使用 Git 認證的 HTTPS 使用者 及 適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)。

  > **Note**
  >
  > Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其使用的方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱疑難排解.

CodeCommit 支持 HTTPS 和安全殼層身份驗證。若要完成設定，您必須設定 Git 認證 CodeCommit (HTTPS，建議大多數使用者使用)、存取時要使用的 SSH key pair CodeCommit (SSH)、git-remote-codecommit (建議使用聯合存取的使用者使用) 或 AWS CLI (HTTPS) 中包含的認證協助程式。

- 關於所有支援作業系統上的 Git 登入資料，請參閱第 3 步：為 HTTPS 連接創建 Git 憑據 CodeCommit。
- 如需在 Linux、macOS 或 Unix 上使用的 SSH 資訊，請參閱安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰CodeCommit。
- 關於 Windows 上的 SSH，請參閱 步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰。
- 若為 git-remote-codecommit，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。
- 如需 Linux、macOS 或 Unix 上的認證協助程式，請參閱設定認證協助程式 (Linux、macOS 或 Unix)。
- 關於 Windows 上的登入資料協助程式，請參閱設定登入資料協助程式 (Windows)。

步驟 1：建立 CodeCommit 儲存庫

在本節中，您將使用主 CodeCommit 控台來建立用於本教學課程其餘部分的 CodeCommit 存放庫。若要使用 AWS CLI 來建立儲存庫，請參閱建立儲存庫 (AWS CLI)。

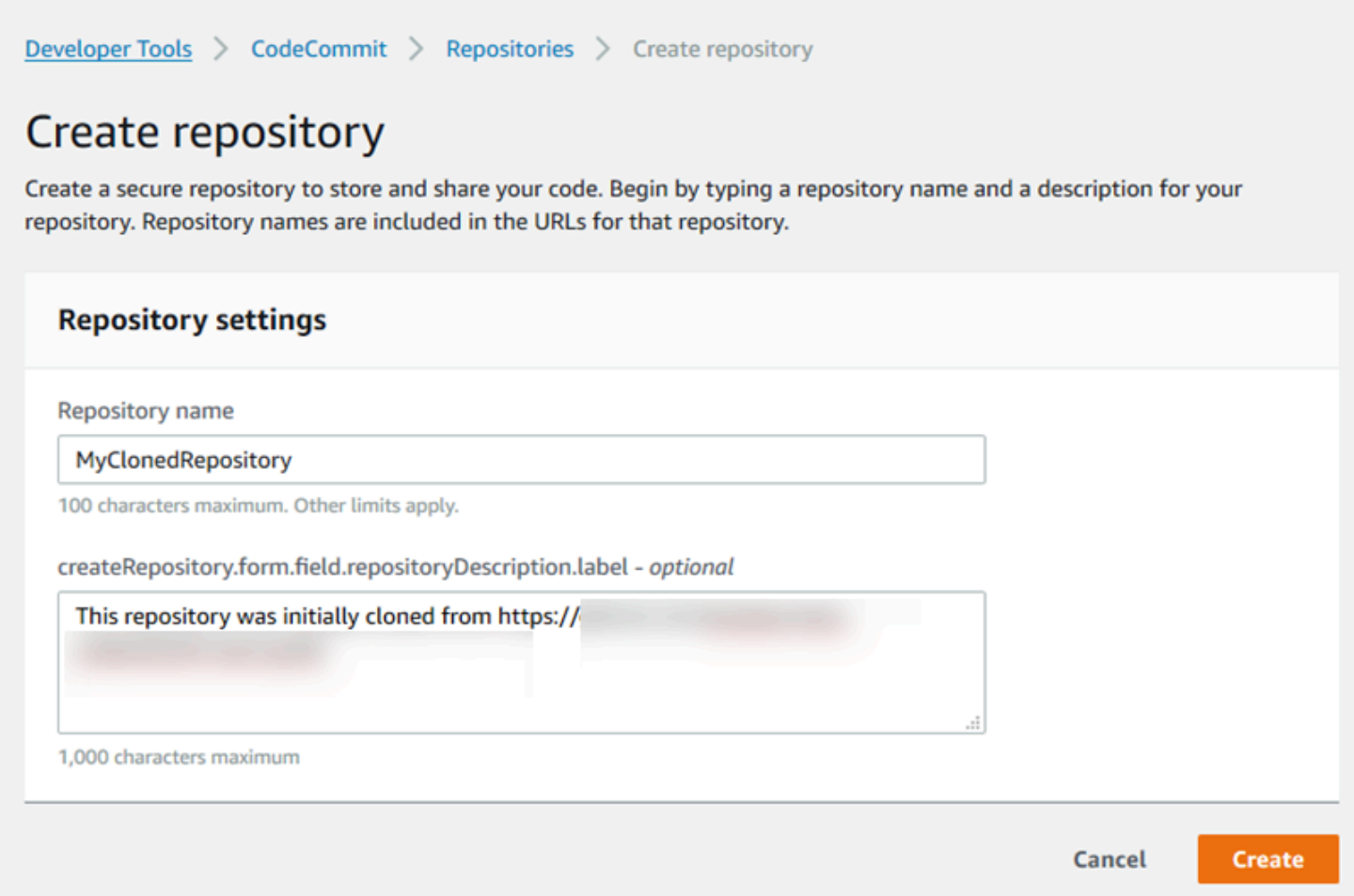
1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選取器中，選擇您AWS 區域要建立儲存庫的位置。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 請在 Repositories (儲存庫) 頁面上，選擇 Create repository (建立儲存庫)。
4. 在 Create repository (建立儲存庫) 頁面的 Repository name (儲存庫名稱) 中，輸入儲存庫的名稱。

   > **Note**
   >
   > 儲存庫名稱需區分大小寫。該名稱在您的 Amazon Web Services 帳戶中必須是唯一的。AWS 區域

5. (選用) 在 Description (描述) 中，輸入儲存庫的描述。這可協助您和其他使用者識別儲存庫的用途。

> **Note**
>
> 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的應用程式開發人員，且計劃在 Web 瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

6. (選用) 選擇 Add tag (新增標籤)，以新增一或多個儲存庫標籤 (自訂屬性標籤，可協助您整理和管理您的 AWS 資源) 到您的儲存庫。如需詳細資訊，請參閱標記儲存庫 AWS CodeCommit。
7. (選擇性) 展開其他組態以指定是使用預設金鑰AWS 受管金鑰還是您自己的客戶管理金鑰來加密和解密此儲存庫中的資料。如果您選擇使用自己的客戶管理金鑰，則必須確定該金鑰可在您建立儲存庫的AWS 區域位置使用，且金鑰處於作用中狀態。如需詳細資訊，請參閱AWS Key Management Service和AWS CodeCommit存儲庫的加密。
8. (選擇性) 如果此儲存庫包含 Java 或 Python 程式碼，且您希望 CodeGuru 審核者對其進行分析，請選取啟用 Java 和 Python 的 Amazon CodeGuru 審核者。 CodeGuru Reviewer 使用多種機器學習模型來尋找程式碼瑕疵，並在提取要求中提供改善和修正建議。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南。
9. 選擇建立。



建立儲存庫之後，該儲存庫會出現在 Repositories (儲存庫) 清單中。在 URL 欄中，選擇複製圖示，然後選擇要用來連接到 CodeCommit 的通訊協定 (SSH 或 HTTPS)。複製 URL。

例如，如果您為儲存庫命名，*MyClonedRepository*而且在美國東部 (俄亥俄) 區域使用具有 HTTPS 的 Git 認證，則 URL 如下所示：

```
https://git-codecommit.us-east-2.amazonaws.com/MyClonedRepository
```

稍後在步驟 2：克隆存儲庫並推送到 CodeCommit 存儲庫中，您需要用到此 URL。

步驟 2：克隆存儲庫並推送到 CodeCommit 存儲庫

在本節中，您會將 Git 儲存庫複製到本機電腦，建立所謂的本機儲存庫。然後，您將本地回購的內容推送到之前創建的 CodeCommit 存儲庫中。

1. 從本機電腦上的終端機或命令提示字元執行git clone命令，並--mirror選擇將遠端存放庫的裸副本複本複製到名為的新資料夾中*aws-codecommit-demo*。這是一個僅用於遷移的裸機儲存

庫。它不是與中遷移的存儲庫進行交互的本地存儲庫 CodeCommit。您可以在遷移完成後稍後創 CodeCommit 建它。

下列範例會將託管於 GitHub (*https://github.com/awslabs/ aws-demo-php-simple-app.git*) 的示範應用程式複製到名為的目錄中的本機存放庫。*aws-codecommit-demo*

```
git clone --mirror https://github.com/awslabs/aws-demo-php-simple-app.git aws-codecommit-demo
```

2. 將目錄變更為您進行複製的目錄。

```
cd aws-codecommit-demo
```

3. 運行git push命令，指定目標 CodeCommit 存儲庫的 URL 和名稱以及選--all項。(這是您在步驟 1：建立 CodeCommit 儲存庫中複製的 URL)。

例如，如果您為存放庫命名，*MyClonedRepository*並且設定為使用 HTTPS，則可以執行下列命令：

```
git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyClonedRepository
 --all
```

> **Note**
>
> --all 選項只會推送儲存庫的所有分支。不會推送其他參考，例如標籤。如果您想要推送標籤，請等候到初始推送完成，然後再推送一次，但這次使用 --tags 選項：
>
> ```
> git push ssh://git-codecommit.us-east-2.amazonaws.com/v1/
> repos/MyClonedRepository --tags
> ```
>
> 如需詳細資訊，請參閱 Git 網站上的 Git 推送。如需有關推送大型儲存庫的資訊，尤其是一次推送所有參考 (例如，使用 --mirror 選項)，請參閱以增量方式移轉儲存庫。

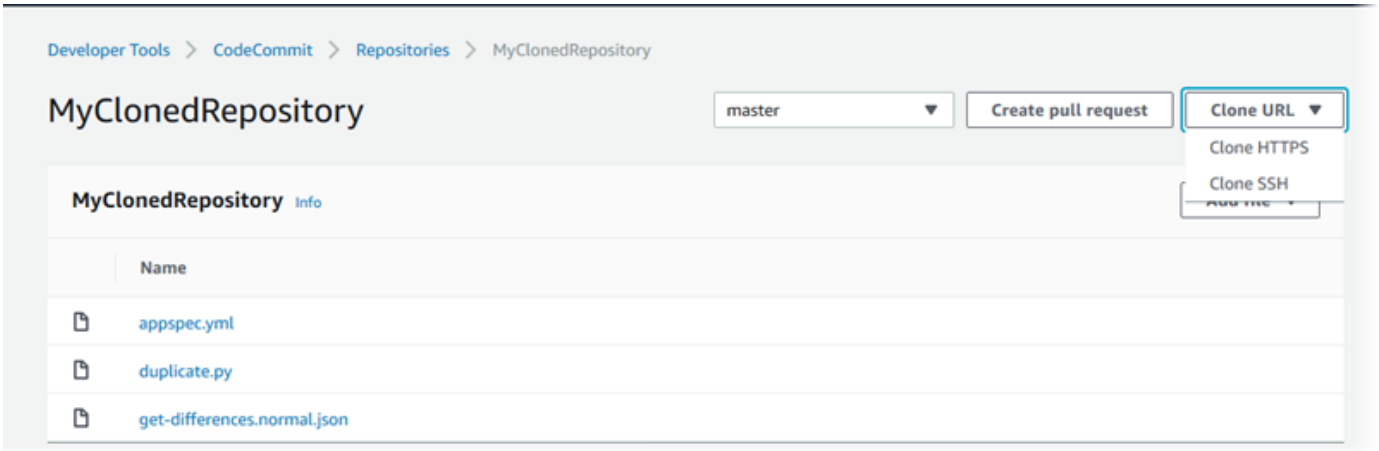
您可以在將存放庫移轉至之後刪除*aws-codecommit-demo*資料夾及其內容 CodeCommit。若要使用中使用儲存庫的所有正確參考來建立本機存放庫 CodeCommit，請執行不含`--mirror`選項的`git clone`指令：

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

步驟 3：查看文件 CodeCommit

推送目錄的內容後，您可以使用 CodeCommit 控制台快速查看該存儲庫中的所有文件。

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在存放庫中，選擇存放庫的名稱 (例如，*MyClonedRepository*)。
3. 檢視儲存庫中的檔案，以查看分支、複製 URL、設定等等。



步驟 4：共享存 CodeCommit 儲庫

當您在中建立存放庫時 CodeCommit，會產生兩個端點：一個用於 HTTPS 連線，另一個用於 SSH 連線。兩者都提供網路上的安全連線。您的使用者可以使用任一通訊協定。無論您建議使用者採用哪個通訊協定，這兩個端點都保持在作用中。在與他人共用儲存庫之前，您必須先建立允許其他使用者存取您的儲存庫的 IAM 政策。提供這些存取指示給您的使用者。

為您的儲存庫建立客戶受管政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。
3. 在 [建立原則] 頁面上，選擇 [匯入受管理的原則]。
4. 在 [匯入受管原則] 頁面的 [篩選器策略] 中，輸入**AWSCodeCommitPowerUser**。選擇原則名稱旁邊的按鈕，然後選擇 [匯入]。
5. 在 Create policy (建立政策) 頁面上，選擇 JSON。將 CodeCommit 動作`Resource`行的「*」部分取代為 CodeCommit 儲存庫的 Amazon 資源名稱 (ARN)，如下所示：

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

> **Tip**
>
> 若要尋找 CodeCommit 存放庫的 ARN，請移至 CodeCommit 主控台，從清單中選擇存放庫名稱，然後選擇 [設定]。如需詳細資訊，請參閱檢視儲存庫詳細。

如果您希望此政策套用到多個儲存庫，請指定儲存庫的 ARN，將每個儲存庫新增為資源。在每個資源陳述式之間包含逗號，如下所示：

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

完成編輯後，請選擇 [檢閱原則]。

6. 在 [檢閱原則] 頁面的 [名稱] 中，輸入原則的新名稱 (例如 *AWSCodeCommitPowerUser-MyDemoRepo*)。選擇性地提供此原則的說明。
7. 選擇建立政策。

若要管理存放庫的存取權，請為其使用者建立 IAM 群組，將 IAM 使用者新增至該群組，然後附加您在上一步中建立的客戶受管政策。附加存取所需的任何其他原則，例如 IAMUserSSHKeys 或. IAMSelfManageServiceSpecificCredentials

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Groups (群組)，然後選擇 Create New Group (建立新的群組)。
3. 在 Set Group Name (設定群組名稱) 頁面的 Group Name (群組名稱) 中，輸入群組的名稱 (例如，*MyDemoRepoGroup*)，然後選擇 Next Step (下一步)。請考慮將儲存庫名稱包含於群組名稱中。

> **Note**
>
> 此名稱在 Amazon Web Services 帳戶中必須是唯一的。

4. 選取您在上一節建立的客戶受管政策旁的方塊 (例如，AWSCodeCommitPowerUser-MyDemoRepo)。
5. 在 Review (檢閱) 頁面上，選擇 Create Group (建立群組)。IAM 會使用已附加的指定政策建立此群組。該群組會出現在與您的 Amazon Web Services 帳戶關聯的群組清單中。
6. 從清單中選擇您的群組。
7. 在群組摘要頁面上，選擇 Users (使用者) 標籤，然後選擇 Add Users to Group (新增使用者到群組)。在顯示與 Amazon Web Services 帳戶關聯的所有使用者的清單中，選取您要允許存取 CodeCommit 存放庫的使用者旁邊的核取方塊，然後選擇 [新增使用者]。

> **Tip**
>
> 您可以使用 [Search (搜尋)] 方塊，依名稱快速尋找使用者。

8. 新增使用者後，請關閉 IAM 主控台。

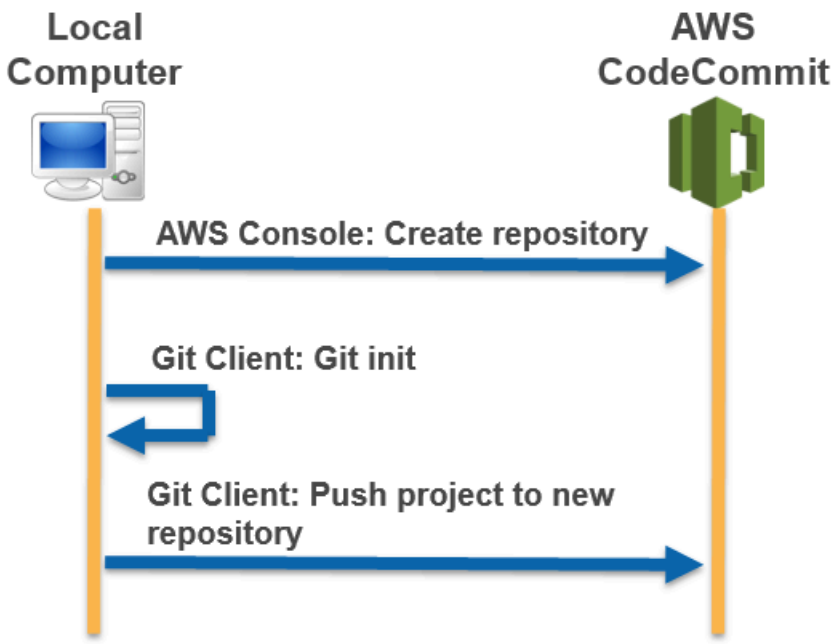
建立要使用您設定的政策群組和政策存取的 IAM CodeCommit 使用者之後，請將連線到存放庫所需的資訊傳送給該使用者。

1. 開啟主 CodeCommit 控台，網址為 https://console.aws.amazon.com/codesuite/codecommit/home。
2. 在區域選取器中，選擇建立儲存庫的AWS 區域位置。儲存庫特定於AWS 區域. 如需詳細資訊，請參閱區域和 Git 連線端點。
3. 在 Repositories (儲存庫) 頁面上，選擇您要共用的儲存庫。
4. 在 Clone URL (複製 URL) 中，選擇您要讓使用者使用的通訊協定。這樣會複製該連線通訊協定的複製 URL。
5. 將複製 URL 連同其他任何指示傳送給使用者，例如安裝 AWS CLI、設定描述檔或安裝 Git。請務必包含連線通訊協定的組態資訊 (例如 HTTPS)。

將本機或未版本控制的內容移轉至 AWS CodeCommit

本主題中的程序說明如何將電腦上的現有專案或本機內容移轉至 CodeCommit 存放庫。隨著此程序，您會完成：

- 完成所需的初始設定 CodeCommit。
- 創建一個 CodeCommit 存儲庫。
- 將本地文件夾放在 Git 版本控制下，並將該文件夾的內容推送到 CodeCommit 存儲庫中。
- 檢視儲 CodeCommit 存庫中的檔案。
- 與您的團隊共用 CodeCommit 儲存庫。



主題

步驟 0：存取所需的設定 CodeCommit

您必須先為本機電腦建立並設定 IAM 使用者 CodeCommit，並設定本機電腦進行存取 CodeCommit ，才能將本機內容移轉至。您也應該安裝 AWS CLI 來管理 CodeCommit。雖然您可以在沒有它的情況下執行大多數 CodeCommit 任務，但是在使用 Git 時可以AWS CLI提供彈性。

如果您已設置 CodeCommit，則可以跳到步驟 1：建立 CodeCommit 儲存庫。

若要建立和設定 IAM 使用者以存取 CodeCommit

1. 通過轉到 http://aws.amazon.com 並選擇註冊創建一個 Amazon Web Services 帳戶。
2. 在您的 Amazon Web Services 帳戶中建立 IAM 使用者，或使用現有的使用者。確定您擁有與該 IAM 使用者相關聯的存取金鑰 ID 和秘密存取金鑰。如需詳細資訊，請參閱在您的 Amazon Web Services 帳戶中建立 IAM 使用者。

   > **Note**
   >
   > CodeCommit 需要AWS Key Management Service。如果您使用現有的 IAM 使用者，請確定沒有附加給使用者的政策明確拒絕AWS KMS執行所需的動作 CodeCommit。如需詳細資訊，請參閱AWS KMS和加密。

3. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
4. 在 IAM 主控台的導覽窗格中，選擇 [使用者]，然後選擇要設定用於 CodeCommit 存取的 IAM 使用者。
5. 在 Permissions (許可) 標籤上，選擇 Add Permissions (新增許可)。
6. 在 Grant permissions (授予許可) 中，選擇 Attach existing policies directly (直接連接現有政策)。
7. 從策略清單中，選取AWSCodeCommitPowerUser或另一個受管理的策略以進行 CodeCommit 存取。如需詳細資訊，請參閱 CodeCommit 的 AWS 受管政策。

   選取要附加的政策之後，請選擇 [下一步：檢閱] 以檢閱要附加至 IAM 使用者的政策清單。如果清單正確，請選擇 Add permissions (新增許可)。

   如需有關 CodeCommit 受管政策以及與其他群組和使用者共用存放庫存取權的詳細資訊，請參閱共用儲存庫和AWS CodeCommit 的身分驗證與存取控制。

安裝及設定 AWS CLI

1. 在本機電腦上，下載並安裝 AWS CLI。這是從命令列進行互動 CodeCommit 的先決條件。我們建議您安裝AWS CLI版本 2。它是最新的主要版本，AWS CLI並支援所有最新功能。它是唯一支援使用 root 帳戶、同盟存取或暫時登入資料的git-remote-codecommit版本。AWS CLI

   如需詳細資訊，請參閱開始設定 AWS 命令列界面。

   > **Note**
   >
   > CodeCommit 僅適用於 1.7.38 及AWS CLI更高版本。根據最佳實務，安裝 AWS CLI 或將其升級到可用的最新版本。若要判斷您已安裝的 AWS CLI 版本，請執行 aws --version 命令。
   > 若要將舊版的 AWS CLI 升級為最新版本，請參閱安裝 AWS Command Line Interface。

2. 執行此命令以確認是否已安裝AWS CLI的命 CodeCommit 令。

   ```
   aws codecommit help
   ```

   此命令返回命 CodeCommit 令列表。

3. 使AWS CLI用configure指令來設定使用設定檔，如下所示：

   ```
   aws configure
   ```

   出現提示時，指定AWS要與之 IAM 使用者搭配使用的存取金鑰和AWS秘密存取金鑰 CodeCommit。此外，請務必指定存放庫的存在AWS 區域位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

   ```
   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
   ```

   如需建立和設定與 AWS CLI 搭配使用之描述檔的詳細資訊，請參閱下列內容：

   - 命名設定檔

- 在中使用 IAM 角色 AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源AWS 區域，您必須AWS CLI使用預設的 [區域] 名稱重新配置。支援的預設地區名稱 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

若要取得有關 CodeCommit 和的更多資訊AWS 區域，請參閱區域和 Git 連線端點。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得登入資料？ 以及管理 IAM 使用者的存取金鑰。如需 AWS CLI 和描述檔的詳細資訊，請參閱具名描述檔。

接著，您必須安裝 Git。

- 對於 Linux，macOS 系統或 Unix：

  若要處理 CodeCommit 儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

  若要安裝 Git，我們建議使用 Git 下載等網站。

  > **Note**
  >
  > Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其使用的方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱中的資訊疑難排解。

- 用於 Windows：

  若要處理 CodeCommit 儲存庫中的檔案、認可和其他資訊，您必須在本機電腦上安裝 Git。CodeCommit 支援 Git 版本 1.7.9 及更高版本。Git 版本 2.28 支持配置初始提交的分支名稱。我們建議您使用最新版本的 Git。

  要安裝 Git，我們建議您使用 Git 之類的網站。如果您使用此連結來安裝 Git，您可以接受所有安裝預設設定，但下列項目除外：

  - 在調整 PATH 環境步驟期間出現提示時，請從命令列選擇要使用 Git 的選項。
  - (選擇性) 如果您想要搭配使用 HTTPS 的認證協助程式 (AWS CLI而非為其設定 Git 認證) CodeCommit，請在 [設定額外選項] 頁面上，確定已清除 [啟用 Git 認證管理員] 選項。只有在 IAM 使用者設定 Git 認證時，才能與 CodeCommit Git 認證管理員相容。如需詳細資訊，請參閱 適用

於使用 Git 認證的 HTTPS 使用者 及 適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)。

> **Note**
>
> Git 是一個不斷發展的，定期更新的平台。有時候，功能變更可能會影響其使用的方式 CodeCommit。如果您在使用特定 Git 版本時遇到問題 CodeCommit，請檢閱中的資訊疑難排解。

CodeCommit 支持 HTTPS 和安全殼層身份驗證。若要完成設定，您必須設定 Git 認證 CodeCommit (HTTPS，建議大多數使用者使用)、存取時要使用的 SSH key pair git-remote-codecommit (SSH) CodeCommit、(建議使用聯合存取的使用者使用) 或. AWS CLI

- 關於所有支援作業系統上的 Git 登入資料，請參閱第 3 步：為 HTTPS 連接創建 Git 憑據 CodeCommit。
- 如需在 Linux、macOS 或 Unix 上使用的 SSH 資訊，請參閱安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰CodeCommit。
- 關於 Windows 上的 SSH，請參閱 步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰。
- 若為 git-remote-codecommit，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。
- 如需 Linux、macOS 或 Unix 上的認證協助程式，請參閱設定認證協助程式 (Linux、macOS 或 Unix)。
- 關於 Windows 上的登入資料協助程式，請參閱設定登入資料協助程式 (Windows)。

步驟 1：建立 CodeCommit 儲存庫

在本節中，您將使用主 CodeCommit 控台來建立用於本教學課程其餘部分的 CodeCommit 存放庫。若要使用 AWS CLI 來建立儲存庫，請參閱建立儲存庫 (AWS CLI)。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在區域選取器中，選擇您AWS 區域要建立儲存庫的位置。如需詳細資訊，請參閱區域和 Git 連線端點。
3. 請在 Repositories (儲存庫) 頁面上，選擇 Create repository (建立儲存庫)。

4. 在 Create repository (建立儲存庫) 頁面的 Repository name (儲存庫名稱) 中，輸入儲存庫的名稱。

   > **Note**
   >
   > 儲存庫名稱需區分大小寫。該名稱在您的 Amazon Web Services 帳戶中必須是唯一的。AWS 區域

5. (選用) 在 Description (描述) 中，輸入儲存庫的描述。這可協助您和其他使用者識別儲存庫的用途。

   > **Note**
   >
   > 描述欄位會在主控台中顯示降價，並接受所有 HTML 字元和有效的 Unicode 字元。如果您是使用`GetRepository`或 `BatchGetRepositories` API 的應用程式開發人員，且計劃在網頁瀏覽器中顯示存放庫描述欄位，請參閱 CodeCommit API 參考。

6. (選用) 選擇 Add tag (新增標籤)，以新增一或多個儲存庫標籤 (自訂屬性標籤，可協助您整理和管理您的 AWS 資源) 到您的儲存庫。如需詳細資訊，請參閱標記儲存庫 AWS CodeCommit。
7. (選擇性) 展開其他組態以指定是使用預設金鑰AWS 受管金鑰還是您自己的客戶管理金鑰來加密和解密此儲存庫中的資料。如果您選擇使用自己的客戶管理金鑰，則必須確定該金鑰可在您建立儲存庫的AWS 區域位置使用，且金鑰處於作用中狀態。如需詳細資訊，請參閱AWS Key Management Service和AWS CodeCommit存儲庫的加密。
8. (選擇性) 如果此儲存庫包含 Java 或 Python 程式碼，且您希望 CodeGuru 審核者對其進行分析，請選取啟用 Java 和 Python 的 Amazon CodeGuru 審核者。 CodeGuru Reviewer 使用多種機器學習模型來尋找程式碼瑕疵，並在提取要求中提供改善和修正建議。如需詳細資訊，請參閱 Amazon CodeGuru 審核者使用者指南。
9. 選擇建立。

建立儲存庫之後，該儲存庫會出現在 Repositories (儲存庫) 清單中。在 URL 欄中，選擇複製圖示，然後選擇要用來連接到 CodeCommit 的通訊協定 (HTTPS 或 SSH)。複製 URL。

例如，如果您為存放庫命名，*`MyFirstRepo`*並且使用 HTTPS，則 URL 將如下所示：

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

稍後在步驟 2：將本機內容移轉至 CodeCommit 儲存庫中，您需要用到此 URL。

步驟 2：將本機內容移轉至 CodeCommit 儲存庫

現在您有了 CodeCommit 儲存庫，您可以在本機電腦上選擇要轉換成本機 Git 儲存庫的目錄。git init 命令可用於將現有的、無版本控制的內容轉換為 Git 儲存庫，或者，如果您尚無檔案或內容，則用於初始化新的空白儲存庫。

1. 從本機電腦上的終端機或命令列，將目錄變更到您要做為儲存庫來源的目錄。
2. 執行下列命令，將 Git 設定為使用名為的預設分支**main**：

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

   您還可以運行此命令，將所有新創建的存儲庫的默認分支名稱設置**main**為：

   ```
   git config --global init.defaultBranch main
   ```
3. 執行 git init 命令，在此目錄中初始化 Git 版本控制。這會在目錄的根建立 .git 子目錄，以啟用版本控制追蹤。此 .git 資料夾也包含儲存庫的所有必要中繼資料。

   ```
   git init
   ```
4. 執行下列命令來檢查初始化目錄的狀態：

   ```
   git status
   ```

   新增您要加入版本控制中的檔案。在此教學中，您會執行 `git add` 命令搭配 `.` 指標，以新增這個目錄中的所有檔案。關於其他選項，請參閱 Git 文件。

   ```
   git add .
   ```
5. 以遞交訊息為新增的檔案建立遞交。

   ```
   git commit -m "Initial commit"
   ```
6. 運行git push命令，指定目標 CodeCommit 存儲庫的 URL 和名稱以及選`--all`項。(這是您在步驟 1：建立 CodeCommit 儲存庫中複製的 URL。)

   例如，如果您為存放庫命名，*MyFirstRepo*並且設定為使用 HTTPS，則可以執行下列命令：

   ```
   git push https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyFirstRepo --all
   ```

步驟 3：查看文件 CodeCommit

推送目錄的內容後，您可以使用 CodeCommit 控制台快速查看存儲庫中的所有文件。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在存放庫中，從清單中選擇存放庫的名稱 (例如，*MyFirstRepository*)。
3. 檢視儲存庫中的檔案，以查看分支、複製 URL、設定等等。

步驟 4：共享存 CodeCommit 儲庫

當您在中建立存放庫時 CodeCommit，會產生兩個端點：一個用於 HTTPS 連線，另一個用於 SSH 連線。兩者都提供網路上的安全連線。您的使用者可以使用任一通訊協定。無論您建議使用者採用哪個通訊協定，這兩個端點都保持在作用中。在與他人共用儲存庫之前，您必須先建立允許其他使用者存取您的儲存庫的 IAM 政策。提供這些存取指示給您的使用者。

為您的儲存庫建立客戶受管政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。
3. 在 [建立原則] 頁面上，選擇 [匯入受管理的原則]。
4. 在 [匯入受管原則] 頁面的 [篩選器策略] 中，輸入**AWSCodeCommitPowerUser**。選擇原則名稱旁邊的按鈕，然後選擇 [匯入]。
5. 在 Create policy (建立政策) 頁面上，選擇 JSON。將 CodeCommit 動作`Resource`行的「*」部分取代為 CodeCommit 儲存庫的 Amazon 資源名稱 (ARN)，如下所示：

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

> **Tip**
>
> 若要尋找 CodeCommit 存放庫的 ARN，請移至 CodeCommit 主控台，從清單中選擇存放庫名稱，然後選擇 [設定]。如需詳細資訊，請參閱檢視儲存庫詳細。

如果您希望此政策套用到多個儲存庫，請指定儲存庫的 ARN，將每個儲存庫新增為資源。在每個資源陳述式之間包含逗號，如下所示：

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

完成編輯後，請選擇 [檢閱原則]。

6. 在 [檢閱原則] 頁面的 [名稱] 中，輸入原則的新名稱 (例如 *AWSCodeCommitPowerUser-MyDemoRepo*)。選擇性地提供此原則的說明。
7. 選擇建立政策。

若要管理存放庫的存取權，請為其使用者建立 IAM 群組，將 IAM 使用者新增至該群組，然後附加您在上一個步驟中建立的客戶受管政策。附加存取所需的任何其他政策，例如 `IAMSelfManageServiceSpecificCredentials` 或 `IAMUserSSHKeys`。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 Dashboard (儀表板) 導覽區域中，選擇 Groups (群組)，然後選擇 Create New Group (建立新的群組)。
3. 在 Set Group Name (設定群組名稱) 頁面的 Group Name (群組名稱) 中，輸入群組的名稱 (例如，*MyDemoRepoGroup*)，然後選擇 Next Step (下一步)。請考慮將儲存庫名稱包含於群組名稱中。

> **Note**
>
> 此名稱在 Amazon Web Services 帳戶中必須是唯一的。

4. 選取您在上一節建立的客戶受管政策旁的方塊 (例如，AWSCodeCommitPowerUser-MyDemoRepo)。
5. 在 Review (檢閱) 頁面上，選擇 Create Group (建立群組)。IAM 會使用已附加的指定政策建立此群組。該群組會出現在與您的 Amazon Web Services 帳戶關聯的群組清單中。
6. 從清單中選擇您的群組。

7. 在群組摘要頁面上，選擇 Users (使用者) 標籤，然後選擇 Add Users to Group (新增使用者到群組)。在顯示與 Amazon Web Services 帳戶關聯的所有使用者的清單中，選取您要允許存取 CodeCommit 存放庫的使用者旁邊的核取方塊，然後選擇 [新增使用者]。

   > **Tip**
   >
   > 您可以使用 [Search (搜尋)] 方塊，依名稱快速尋找使用者。

8. 新增使用者後，請關閉 IAM 主控台。

建立要使用您設定的政策群組和政策存取的 IAM CodeCommit 使用者後，請將連線到存放庫所需的資訊傳送給該使用者。

1. 請在以下位置開啟 CodeCommit 主控台。 https://console.aws.amazon.com/codesuite/codecommit/home
2. 在區域選取器中，選擇建立儲存庫的AWS 區域位置。儲存庫特定於AWS 區域. 如需詳細資訊，請參閱區域和 Git 連線端點。
3. 在 Repositories (儲存庫) 頁面上，選擇您要共用的儲存庫。
4. 在 Clone URL (複製 URL) 中，選擇您要讓使用者使用的通訊協定。這樣會複製該連線通訊協定的複製 URL。
5. 將複製 URL 連同其他任何指示傳送給使用者，例如安裝 AWS CLI、設定描述檔或安裝 Git。請務必包含連線通訊協定的組態資訊 (例如 HTTPS)。

以增量方式移轉儲存庫

遷移到 AWS CodeCommit 時，請考慮依增量或區塊推送儲存庫，以減少因為間歇性網路問題或網路效能降低而造成整個推送失敗的可能性。如果採用遞增推送並搭配如此處所含的指令碼，您可以重新開始遷移，並只推送稍早嘗試時未成功的遞交。

此主題中的程序說明如何建立和執行指令碼，以增量方式遷移儲存庫，且只重新推送未成功的增量，直到遷移完成為止。

這些指示的撰寫是假設您已完成設定 和建立 儲存庫中的步驟。

主題

- 步驟 0：判斷是否要以累加方式移轉
- 步驟 1：安裝必要條件並將 CodeCommit 存放庫新增為遠端

步驟 0：判斷是否要以累加方式移轉

需要考量幾個因素，以決定儲存庫的整體大小及是否漸進遷移。最明顯的因素是儲存庫中成品的整體大小。像是儲存庫的累積歷史記錄，這種因素也會影響大小。具有多年的歷史記錄和分支的儲存庫可能會非常龐大，即使個別資產並不大。您可以運用多種策略來讓遷移這些儲存庫更簡單且更有效率。例如，您可以在複製開發歷史記錄很長的儲存庫時使用淺層複製策略，或對大型二進位檔案停用差異壓縮。您可以透過查詢 Git 文件來研究選項，或者您可以選擇設定和配置遞增推送，用於使用此主題 `incremental-repo-migration.py` 隨附的範例指令碼來遷移您的儲存庫。

如果以下一或多個條件成立，您可能想要設定遞增推送：

- 您要遷移的儲存庫有超過 5 年的歷史記錄。
- 您的網際網路連線受限於不穩定的中斷、捨棄的封包、緩慢回應或其他服務中斷。
- 儲存庫的整體大小大於 2 GB，而您要遷移整個儲存庫。
- 儲存庫包含的大型成品或二進位檔案未正確壓縮，例如具有超過 5 個追蹤版本的大型映像檔案。
- 您之前曾嘗試移轉至， CodeCommit 並收到「內部服務錯誤」訊息。

即使上述條件均不符合，您仍然可以選擇以遞增方式推送。

步驟 1：安裝必要條件並將 CodeCommit 存放庫新增為遠端

您可以建立自己的自訂指令碼，它有其自己的先決條件。如果使用此主題中的範例，您必須：

- 安裝其必要項目。
- 將儲存庫複製到本機電腦。
- 將 CodeCommit 存放庫新增為您要移轉之存放庫的遠端。

設置為運行 incremental-repo-migration .py

1. 在您的本機電腦上安裝 Python 2.6 或更新版本。如需詳細資訊以及最新的版本，請參閱 Python 網站。

2. 在同一台計算機上，安裝 GitPython，這是一個用於與 Git 存儲庫交互的 Python 庫。如需詳細資訊，請參閱 GitPython文件。

3. 使用 git clone --mirror 命令來複製您要遷移到本機電腦的儲存庫。從終端機 (Linux、macOS 或 Unix) 或命令提示字元 (Windows)，使用git clone --mirror指令建立儲存庫的本機存放庫，包括您要在其中建立本機存放庫的目錄。舉例來說，若要將一個名*MyMigrationRepo*為 *https://example.com/my-repo/* 的 Git 儲存庫複製到名為 *my-repo* 的目錄中：

```
git clone --mirror https://example.com/my-repo/MyMigrationRepo.git my-repo
```

您應該會看到類似以下的輸出，這表示已將儲存庫複製到名為 my-repo 的本機儲存庫：

```
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. 將您剛複製的儲存庫目錄切換到本機儲存庫 (例如，*my-repo*)。從該目錄，使用 git remote add ***DefaultRemoteName RemoteRepositoryURL*** 命令來將 CodeCommit 儲存庫新增為遠端儲存庫的本機儲存庫。

> **Note**
>
> 推送大型儲存庫時，請考慮使用 SSH，而不是 HTTPS。推送大型變更、大量變更，或大型儲存庫時，長時間執行的 HTTPS 連線經常會因為網路連線問題或防火牆設定提前終止。如需有關設定 SSH CodeCommit 的詳細資訊，請參閱Linux Linux或適用於 Windows 上的 SSH 連線。

例如，使用以下命令為名為的遠端 CodeCommit存放庫新增名 MyDestinationRepo 為遠端存放庫的 SSH 端點`codecommit`：

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

> **Tip**
>
> 由於這是複製，預設的遠端名稱 (origin) 已在使用中。您必須使用其他遠端名稱。雖然範例使用 codecommit，您可以使用您要的任何名稱。使用 git remote show 命令來檢閱為您的本機儲存庫設定的遠端清單。

5. 使用 git remote -v 命令來顯示本機儲存庫的擷取和推送設定，並確認已正確設定。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**
>
> 如果您仍看到不同遠端儲存庫的擷取和推送項目 (例如，來源項目)，請使用 git remote set-url --delete 命令來移除它們。

步驟 2：建立用於逐步移轉的指令碼

這些步驟的撰寫是假設您使用 incremental-repo-migration.py 範例指令碼。

1. 開啟文字編輯器，並將範例指令碼的內容貼上至空的文件中。
2. 將文件儲存在文件目錄 (而非本機儲存庫的工作目錄) 並將它命名為 incremental-repo-migration.py。確定您選擇的目錄是在您的本機環境或路徑變數中設定的目錄，使得您可以從命令列或終端機執行 Python 指令碼。

步驟 3：執行指令碼並以遞增方式移轉至 CodeCommit

現在您已經建立了incremental-repo-migration.py指令碼，您可以使用它將本機存放庫逐步遷移到 CodeCommit 儲存庫。依預設，指令碼將以每個批次 1,000 個遞交的方式推送遞交，並嘗試使用目錄的 Git 設定，藉此它會執行做為本機儲存庫和遠端儲存庫的設定。必要時，您可以使用 incremental-repo-migration.py 中包含的選項來設定其他設定。

1. 從終端機或命令提示字元，將目錄切換到您要遷移的本機儲存庫。
2. 從該目錄中，執行以下命令：

```
python incremental-repo-migration.py
```

3. 指令碼會執行，並在終端機或命令提示字元中顯示進度。有些大型儲存庫顯示進度的速度會慢下來。如果單一推送失敗了三次，指令碼會停止。然後您可以重新執行指令碼，指令碼從失敗的批次開始。您可以重新執行指令碼，直到所有推送成功且遷移完成。

Tip

您可以從任何目錄執行 `incremental-repo-migration.py`，只要您使用 `-l` 和 `-r` 選項來指定要使用的本機和遠端設定。例如，若要從任何目錄使用指令碼將位於 /tmp/*my-repo* 的本機儲存庫遷移到別名為 *codecommit* 的遠端：

```
python incremental-repo-migration.py -l "/tmp/my-repo" -r "codecommit"
```

在遞增推送時，您可能還想要使用 `-b` 選項來變更使用的預設批次大小。例如，如果您定期推送的儲存庫具有經常變動的非常大型二進位檔案，並且從具有網路頻寬限制的位置作業，您可能希望使用 `-b` 選項來將批次大小變更為 500，而非 1,000。例如：

```
python incremental-repo-migration.py -b 500
```

這將以每個批次 500 個遞交的方式遞增推送本機儲存庫。如果在遷移儲存庫時，您決定再次變更批次大小 (例如，如果您決定減少在嘗試失敗後減少批次大小)，請記得使用 `-c` 選項來移除批次標籤，之後再使用 `-b` 來重設批次大小：

```
python incremental-repo-migration.py -c
python incremental-repo-migration.py -b 250
```

Important

如果您想要在失敗後重新執行指令碼，請勿使用 `-c` 選項。`-c` 選項會移除用於將遞交分批次的標籤。只在您想要變更批次大小並重新開始時，或如果您決定不再使用該指令碼時，才使用 `-c` 選項。

附錄：範例指令集 **incremental-repo-migration.py**

為方便起見，我們開發了一個範例 Python 指令碼 incremental-repo-migration.py，用於以遞增的方式推送儲存庫。此指令碼是開放原始碼範例，並以現狀提供。

```
# 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

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同的責任模式將其稱為雲端的安全性和雲端中的安全性：

- 雲端本身的安全：AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計劃的一部分。了解合規計劃適用於AWS CodeCommit，請參AWS合規計劃的服務範圍。
- 雲端內部的安全 – 您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 CodeCommit 時應用共同責任模型。下列主題將演示如何將 CodeCommit 設定為滿足您的安全與合規目標。您也將了解如何使用其他AWS服務，協助您監控並保護 CodeCommit 資源。

主題

AWS CodeCommit 中的資料保護

是一項受管服務，受 AWS 全球網路安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務以設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可使用 AWS 發佈的 API 呼叫，透過網路存取。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

CodeCommit 儲存庫會在靜態時自動加密。不需要客戶採取任何行動。 CodeCommit 也會加密傳輸中的儲存庫資料。您可以使用 HTTPS 通訊協定、SSH 通訊協定，或兩者搭配 CodeCommit 儲存庫使用。如需詳細資訊，請參閱設定 AWS CodeCommit。您也可以設定儲存庫的跨帳戶 CodeCommit 存取權限。

主題

- AWS Key Management Service和AWS CodeCommit存儲庫的加密
- 使用旋轉認證連線至AWS CodeCommit儲存庫

AWS Key Management Service和AWS CodeCommit存儲庫的加密

CodeCommit 儲存庫中的資料在傳輸和靜態時都會加密。將資料推入儲存 CodeCommit 庫時 (例如，透過呼叫git push)，會在收到的資料儲存在儲存庫中時 CodeCommit 加密該資料。從 CodeCommit 儲存庫中提取資料時 (例如，透過呼叫git pull)，將資料 CodeCommit 解密，然後將其傳送給呼叫者。這假設與推送或提取請求相關聯的 IAM 使用者已經過驗證AWS。傳送或接收的資料是利用 HTTPS 或 SSH 加密網路通訊協定來傳輸。

您可以使用AWS 受管金鑰或客戶管理的金鑰來加密和解密儲存庫中的資料。如需有關客戶受管金鑰之間差異的詳細資訊AWS 受管金鑰，請參閱客戶受管金鑰和 AWS 受管金鑰. 如果您未指定客戶管理的金鑰， CodeCommit 將使AWS 受管金鑰用來加密和解密儲存庫中的資料。這AWS 受管金鑰是自動為您在AWS 帳戶. 第一次在 Amazon Web Services 帳戶的新 CodeCommit 儲存庫AWS 區域中建 CodeCommit 立儲存庫時，如果您未指定客戶受管金鑰，請在 AWS 受管金鑰 () 中建立相同AWS 區域的`aws/codecommit`金鑰 AWS Key Management Service (金鑰AWS KMS)。此`aws/codecommit`金鑰僅供使用 CodeCommit。它存儲在您的 Amazon Web Services 帳戶中。根據您指定的內容，可 CodeCommit 以使用客戶管理的金鑰，或使用AWS 受管金鑰來加密和解密儲存庫中的資料。

> **Important**
>
> CodeCommit AWS KMS對用於加密和解密儲存庫中資料的AWS KMS金鑰執行下列動作。如果您使用的是AWS 受管金鑰，使用者不需要明確的權限來執行這些動作，但使用者不得有任何拒絕`aws/codecommit`金鑰執行這些動作的附加原則。如果您使用的客戶管理金鑰的 AWS 帳戶 ID 已設定為該金鑰的原則主體，則必須將這些權限明確設定為`allow`。具體來說，當您建立第一個儲存區域時，如果您更新儲存庫的金鑰，則不得將下列任何權限設定為AWS 受管金鑰，`deny`如果您使用的是原則主體的 Customer Managed 金鑰，則必須設定為：`allow`

- "kms:Encrypt"
- "kms:Decrypt"
- "kms:ReEncrypt"（根據上下文，這可能需要kms:ReEncryptFrom，kms:ReEncryptTo，或kms:ReEncrypt*不設置為拒絕）
- "kms:GenerateDataKey"
- "kms:GenerateDataKeyWithoutPlaintext"
- "kms:DescribeKey"

如果您想要使用自己的客戶管理金鑰，則該金鑰必須位於存放庫所AWS 區域在的位置。 CodeCommit 支援使用單一和多區域客戶管理金鑰。雖然支援所有金鑰材料原點類型，但我們建議您使用預設的 KMS 選項。使用外部金鑰存放區選項的客戶可能會遇到其商店供應商的延遲。此外。 CodeCommit 客戶受管金鑰具有下列需求：

- CodeCommit 僅支持使用對稱密鑰。
- 金鑰使用類型必須設定為加密和解密。

如需有關建立客戶管理金鑰的詳細資訊，請參閱概念和建立金鑰。

若要查看AWS 受管金鑰產生者的相關資訊 CodeCommit，請執行下列動作：

1. 請登入 AWS Management Console，並開啟 AWS Key Management Service (AWS KMS) 主控台 (網站：https://console.aws.amazon.com/kms)。
2. 若要變更 AWS 區域，請使用頁面右上角的區域選取器。
3. 在服務導覽窗格中，選擇AWS 受管金鑰。請確定您已登入您AWS 區域要檢閱金鑰的位置。
4. 在加密密鑰列表中，選擇AWS 受管金鑰與別名 aws/代碼提交。將顯示有關的AWS 擁有的金鑰基本資訊。

您無法變更或刪除此項AWS 受管金鑰。

如何使用加密演算法來加密儲存庫資料

CodeCommit 使用兩種不同的方法來加密數據。小於 6 MB 的個別 Git 物件是使用 AES-GCM-256 進行加密，此方法會提供資料完整性驗證。對於單一 Blob，介於 6 MB 到最大 2 GB 之間的物件會使用 AES-CBC-256 加密。 CodeCommit 一律驗證加密內容。

加密內容

每一個與 AWS KMS 整合的服務會指定適用於加密和解密操作的加密環境。加密環境是 AWS KMS 用來檢查資料完整性的額外驗證資訊。如果為加密操作指定，則在解密操作中也必須指定。否則，解密會失敗。 CodeCommit 使用 CodeCommit 儲存庫 ID 作為加密內容。您可以使用get-repository命令或 CodeCommit 控制台來查找存儲庫 ID。搜尋AWS CloudTrail記錄檔中的 CodeCommit 存放庫 ID，以瞭解針對哪個金鑰進行了哪些加密作業，AWS KMS以加密或解密 CodeCommit 儲存庫中的資料。

如需 AWS KMS 的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》https://docs.aws.amazon.com/kms/latest/developerguide/。

使用旋轉認證連線至AWS CodeCommit儲存庫

您可以授予使用者存取您的AWS CodeCommit儲存庫，而無需為他們設定 IAM 使用者或使用存取金鑰和秘密金鑰。若要聯合身分，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。您也可以為 IAM 使用者設定以角色為基礎的存取，以CodeCommit存取個別 Amazon Web Services 帳戶中的儲存庫 (一種稱為跨帳戶存取的技術)。如需設定儲存庫跨帳戶存取的逐步解說，請參閱 使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取。

您可以為想要或必須透過下列方法來驗證身分的使用者設定存取：

- 安全性聲明標記語言 (SAML)
- 多重要素驗證 (MFA)
- 聯合
- 登入 Amazon
- Amazon Cognito
- Facebook
- Google
- OpenID Connect (OIDC) 相容的身分提供者

> **Note**
>
> 以下資訊僅適用於使用 git-remote-codecommit 或 AWS CLI 登入資料協助程式來連線到 CodeCommit 儲存庫。由於建議使用的暫時性或聯合存取CodeCommit 的方法需要設定 git-

remote-codecommit，因此本主題提供使用該公用程式的範例。如需詳細資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。
您不能使用 SSH 或 Git 登入資料和 HTTPS 並搭配輪換或暫時性存取登入資料來連線到 CodeCommit 儲存庫。

如果符合下列所有要求，您不需要完成這些步驟：

- 您使用 Amazon EC2 EC2 EC2 EC2 EC2。
- 您使用 Git 和 HTTPS 搭配AWS CLI登入資料協助程式，以便從 Amazon EC2 執行個體連線到 CodeCommit儲存庫。
- Amazon EC2 執行個體具有附加的 IAM 執行個體設定檔，其中包含對於在 Linux、蘋果系統或 Unix 上使用 HTTPS 連接AWS CLI憑證助手或中所述的存取許可對於在視窗上使用 HTTPS 連線AWS CLI憑證助手。
- 您已在 Amazon EC2 執行個體上安裝並設定 Git 登入資料協助程式，如對於在 Linux、蘋果系統或 Unix 上使用 HTTPS 連接AWS CLI憑證助手或中所述對於在視窗上使用 HTTPS 連線AWS CLI憑證助手。

符合上述要求的 Amazon EC2 執行個體已設定為代表您通訊臨時存CodeCommit取登入資料。

> **Note**
>
> 您可以git-remote-codecommit在 Amazon EC2 執行個體上設定和使用。

為了讓使用者臨時存取您的 CodeCommit 儲存庫，請完成以下步驟。

步驟 1：完成先決條件

完成設定步驟，讓使用者能夠使用輪換登入資料存取 CodeCommit 儲存庫：

- 如需跨帳戶存取，請參閱逐步解說：使用 IAM 角色和在 Amazon Web Services 帳戶之間委派存取使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取。
- 關於 SAML 和聯合，請參閱使用組織的身分驗證系統以授予存取 AWS 資源和關於以 AWS STS SAML 2.0 為基礎的聯合。
- 對於 MFA，請參閱搭配使用 Multi-Factor Authentication (MFA) 裝置AWS和建立臨時安全登入資料以啟用 IAM 使用者的存取權。

- 若要使用 Amazon、Amazon Cognito、Facebook、Google 或任何與身分提供者相容的 OIDC，請參閱關於AWS STS Web 身分提供者。

使用AWS CodeCommit 的身分驗證與存取控制中的資訊，指定您要授予使用者的 CodeCommit 許可。

步驟 2：取得角色名稱或存取憑證

如果您要讓使用者擔任角色以存取儲存庫，請提供使用者該角色的 Amazon Resource Name (ARN)。否則，根據您設定存取的方式而定，使用者可以透過下列其中一種方法取得輪換登入資料：

- 對於跨帳戶訪問，請調用AWS CLI假設角色命令或調用AWS STS AssumeRoleAPI。
- 對於 SAML，請呼叫AWS CLIassume-role-with-saml命令或AWS STS AssumeRoleWithSAML API。
- 對於聯合，請呼叫AWS CLI假設角色或get-federation-token命令AWS STSAssumeRole或或GetFederationTokenAPI。
- 對於 MFA，請呼叫AWS CLIget-session-token命令或AWS STS GetSessionTokenAPI。
- 對於使用亞馬遜，Amazon Cognito，臉書，谷歌或任何 OID 兼容的身份提供商登錄，請調用AWS CLI assume-role-with-web-身份命令或AWS STS AssumeRoleWithWebIdentityAPI。

步驟 3：安裝git-remote-codecommit和配置AWS CLI

您必須將本機電腦設定為使用存取身份證明，方法是在中安裝git-remote-codecommit和設定設定檔AWS CLI。

1. 遵循設定 中的指示來設定 AWS CLI。使用 aws configure 命令設定一或多個描述檔。考慮建立一個具名描述檔，以在使用輪換登入資料連線到 CodeCommit 儲存庫時使用。
2. 您可以使用下列其中一種方法，將登入資料與使用者的 AWS CLI 具名描述檔相關聯。

    - 如果您擔任某個可存取 CodeCommit 的角色，請使用擔任該角色所需的資訊來設定具名描述檔。例如，如果您想要假設 Amazon Web Services 帳戶 1111111111 *CodeCommitAccess*中命名的角色，您可以設定使用其他AWS資源時要使用的預設設定檔，以及在擔任該角色時要使用的具名設定檔。下列指令會建立具名的具名設定檔 *CodeAccess*，該設定檔具有名為的角色*CodeCommitAccess*。使用者名稱 *Maria_Garcia* 與工作階段相關聯，預設設定檔會設定為其認AWS證來源：

```
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
```

若要驗證變更，請手動檢視或編輯 ~/.aws/config 檔案 (若為 Linux) 或 %UserProfile %.aws\config 檔案 (若為 Windows)，並檢閱具名描述檔下的資訊。例如，您的檔案看起來可能會類似以下內容：

```
[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
```

設定完具名描述檔之後，您可以接著使用該具名描述檔透過git-remote-codecommit 公用程式來複製 CodeCommit 儲存庫。例如，若要複製名為 *MyDemoRepo* 的儲存庫：

```
git clone codecommit://CodeAccess@MyDemoRepo
```

- 如果您是使用 Web Identity Federation 和 OpenID Connect (OIDC)，請設定具名描述檔，讓 AWS Security Token Service (AWS STS) AssumeRoleWithWebIdentity API 代表您發出呼叫，以重新整理暫時性登入資料。使用 aws configure set 命令或手動編輯 ~/.aws/credentials 檔案 (若為 Linux) 或 %UserProfile%.aws\credentials 檔案 (若為 Windows)，以使用必要的設定值來新增 AWS CLI 具名描述檔。例如，要創建一個擔任該*CodeCommitAccess*角色並使用 Web 身份令牌文件的配置文件~/*####/my-token-file*：

```
[CodeCommitWebIdentity]
role_arn = arn:aws:iam::111111111111:role/CodeCommitAccess
web_identity_token_file=~/my-credentials/my-token-file
role_session_name = Maria_Garcia
```

如需詳細資訊，請參閱《使用AWS Command Line Interface者指南》中的〈設定AWS Command Line InterfaceAWS CLI和使用 IAM 角色〉。

步驟 4：存取CodeCommit儲存庫

假設您的使用者已按照連接到儲存庫中的指示連線到 CodeCommit 儲存庫，那麼使用者就會使用 git-remote-codecommit 提供的擴充功能和 Git 來呼叫 git clone、git push 和 git pull，以複製、推送和提取使用者可存取的 CodeCommit 儲存庫。例如，若要複製儲存庫：

```
git clone codecommit://CodeAccess@MyDemoRepo
```

Git 遞交、推送和提取命令會使用常規 Git 語法。

當使用者使用 AWS CLI 並指定與輪換存取登入資料相關聯的 AWS CLI 具名描述檔時，則會傳回該描述檔範圍限定內的結果。

AWS CodeCommit 的身分和存取權管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取權。IAM 管理員控制哪些人可以驗證 (登入) 和授權 (具有權限) 以使用 CodeCommit 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

- 對象
- 使用身分驗證
- 使用政策管理存取權
- AWS CodeCommit 的身分驗證與存取控制
- AWS CodeCommit 搭配 IAM 的運作方式
- CodeCommit 資源型政策
- 基於 CodeCommit 標籤的授權
- CodeCommit IAM 角色
- AWS CodeCommit 身分型政策範例
- 對 AWS CodeCommit 身分與存取進行疑難排解

對象

AWS Identity and Access Management (IAM) 的使用方式會不同，需視您在 CodeCommit 中所執行的工作而定。

服務使用者 — 如果您使用 CodeCommit 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 CodeCommit 功能來完成工作時，您可能需要其他權限。瞭解存取許可的管理方式可協助您向管理員請求正確的許可。若您無法存取 CodeCommit 中的某項功能，請參閱 對 AWS CodeCommit 身分與存取進行疑難排解。

服務管理員 — 如果您負責公司的 CodeCommit 資源，您可能擁有完整的存取權 CodeCommit。決定您的服務使用者應該存取哪些 CodeCommit 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步瞭解貴公司如何搭配使用 IAM CodeCommit，請參閱AWS CodeCommit 搭配 IAM 的運作方式。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 CodeCommit 存取權的詳細資訊。若要檢視可在 IAM 中使用的 CodeCommit 基於身分的政策範例，請參閱。AWS CodeCommit 身分型政策範例

使用身分驗證

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者身分，或擔任 IAM 角色進行 驗證（登入至 AWS）。

您可以使用透過身分來源 AWS IAM Identity Center 提供的憑證，以聯合身分登入 AWS。(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。您 AWS 藉由使用聯合進行存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入至 AWS 的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 以提高帳戶的安全。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證（例如密碼和存取金鑰）的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。

- 跨帳户存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人（信任的委託人）存取您帳戶中的資源。角色是授予跨帳户存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源（而非使用角色作為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_forward_access_sessions.html。
  - 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。
  - 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其附加到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體（使用者、根使用者或角色工作階段）發出請求時評估這些政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文

件形式儲存在 AWS 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具備該政策的使用者便可以從 AWS Management Console、AWS CLI 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分（例如 IAM 使用者、使用者群組或角色）的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策則是獨立的政策，您可以將這些政策附加到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需瞭解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人（帳戶成員、使用者或角色）擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體（IAM 使用者或角色）的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。
- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需組織和 SCP 的更多相關資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱《IAM 使用者指南》中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS CodeCommit 的身分驗證與存取控制

存取 AWS CodeCommit 需要憑證。這些認證必須具有存取AWS資源的權限，例如 CodeCommit 儲存庫和 IAM 使用者，這些使用者可用來管理 Git 認證或用於建立 Git 連線的 SSH 公開金鑰。以下各節提供如何使用 AWS Identity and Access Management(IAM) 以及 CodeCommit 協助安全存取資源的詳細資訊：

- 身分驗證
- 存取控制

身分驗證

由於 CodeCommit 儲存庫是以 GIT 為基礎且支援 Git 的基本功能 (包括 Git 認證)，因此建議您在使用時使用 IAM 使用者。 CodeCommit您可以使 CodeCommit 用其他身分類型存取，但其他身分類型會受到限制，如下所述。

身分類型：

- IAM 使用者 — IAM 使用者是您 Amazon Web Services 帳戶中具有特定自訂許可的身分。例如，IAM 使用者可以擁有建立和管理 Git 登入資料以 CodeCommit 存取儲存庫的權限。這是建議使用的使用者類型 CodeCommit。您可以使用 IAM 使用者名稱和密碼登入安全 AWS 網頁，例如，AWS Management Console、AWS 開發論壇或 AWS Support 中心。

  您可以產生 Git 登入資料，或將 SSH 公開金鑰與 IAM 使用者建立關聯，也可以安裝和設定git-remote-codecommit。這些是設定 Git 與您的 CodeCommit 儲存庫搭配使用的最簡單方法。使用 Git 認證，您可以在 IAM 中產生靜態使用者名稱和密碼。然後，使用這些登入資料，對 Git 及支援 Git 使用者名稱和密碼身分驗證的任何第三方工具，建立 HTTPS 連線。透過 SSH 連線，您可以在本機電腦上建立 Git 並 CodeCommit 用於 SSH 驗證的公開和私密金鑰檔案。您可以將公開金鑰與 IAM 使用者建立關聯，然後將私密金鑰儲存在本機電腦上。 git-remote-codecommit擴展 Git 本身，並且不需要為用戶設置 Git 憑據。

  此外，您可以為每個使用者產生存取金鑰。當您以程式設計的方式存取 AWS 服務時 (無論是透過其中一個 AWS 軟體開發套件 或使用 AWS Command Line Interface (AWS CLI))，請使用這些存取金鑰。軟體開發套件和 CLI 工具會使用存取金鑰，以加密方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。 CodeCommit 支援簽章版本 4，這是一種驗證傳入 API 要求的通訊協定。如需有關驗證請求的詳細資訊，請參閱《AWS 一般參考》中的 Signature 第 4 版簽署程序。

- Amazon Web Services 帳戶根使用者 — 當您註冊時AWS，您會提供與您的 Amazon Web Services 帳戶相關聯的電子郵件地址和密碼。這些是您的根憑證，可完整存取您的所有 AWS 資源。根帳號使用者無法使用某些 CodeCommit 功能。此外，將 Git 與根帳戶搭配使用的唯一方法是，安裝並設定 git-remote-codecommit (建議使用) 或設定 AWS CLI 所隨附的 AWS 登入資料協助程式。Git 登入資料或 SSH 公有私有金鑰對不能與根帳戶使用者一起使用。基於這些原因，我們不建議您在與之互動時使用 root 帳戶使用者 CodeCommit。

> **Important**
>
> 基於安全理由，建議您只在建立管理員使用者時使用根憑證，這是擁有 AWS 帳戶完整許可的 IAM 使用者。然後，您可以使用此管理員使用者建立其他 IAM 使用者和角色，並授予有限許可。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/

latest/UserGuide/best-practices.html#create-iam-users中的IAM 最佳實務和建立 Admin (管理員) 使用者和群組。

- IAM 身分中心和 IAM 身分中心中的使用者 — AWS IAM Identity Center 擴展的功能，AWS Identity and Access Management以提供集中的位置，將使用者的管理及其對雲端應用程式的存取AWS 帳戶整合在一起。雖然 IAM 身分中心目前並未提供 Git 登入資料或安全殼層金鑰配對的機制AWS，但建議做為最佳作法。這些使用者可以安裝並設定git-remote-codecommit為本機複製 CodeCommit儲存庫，但並非所有整合式開發環境 (IDE) 都支援複製、推送或提取git-remote-codecommit。

  最佳實務是要求人類使用者（包括需要管理員存取權的使用者）搭配身分提供者使用聯合功能，使用暫時憑證來存取 AWS 服務。

  聯合身分是來自您企業使用者目錄的使用者、Web 身分供應商、AWS Directory Service、Identity Center 目錄或透過身分來源提供的憑證來存取 AWS 服務 的任何使用者。聯合身分存取 AWS 帳戶時，會擔任角色，並由角色提供暫時憑證。

  對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分來源中的一組使用者和群組，以便在您的所有 AWS 帳戶和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱《AWS IAM Identity Center 使用者指南》中的什麼是 IAM Identity Center？。

- IAM 角色 — 與 IAM 使用者一樣，IAM 角色是您可以在帳戶中建立以授予特定許可的 IAM 身分。

  IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

  使用暫時憑證的 IAM 角色在下列情況中非常有用：

  - 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。
  - 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。

- 跨帳户存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人（信任的委託人）存取您帳戶中的資源。角色是授予跨帳户存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源（而非使用角色作為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_forward_access_sessions.html。
  - 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。
  - 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

> **Note**
>
> Git 登入資料或 SSH 公有私有金鑰對不能與聯合身分使用者一起使用。此外，使用者偏好設定不適用於聯合身分使用者。如需如何使用聯合存取設定連線的相關資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

存取控制

您可以擁有有效的認證來驗證您的請求，但除非您擁有權限，否則您無法建立或存取 CodeCommit 資源。例如，您必須有許可來檢視儲存庫、推送程式碼、建立和管理 Git 登入資料等等。

下列各節說明如何管理的權限 CodeCommit。我們建議您先閱讀概觀。

- 管理資 CodeCommit 源存取權限概觀
- 使用以身分為基礎的政策 (IAM 政策) CodeCommit
- CodeCommit 許可參考

管理資 CodeCommit 源存取權限概觀

每個AWS資源都由一個 Amazon Web Services 帳戶擁有。建立或存取資源的許可由許可政策所控管。帳戶管理員可以將許可政策連接到 IAM 身分 (即使用者、群組和角色)。某些服務 (例如 AWS Lambda) 也支援將許可政策連接到資源。

> **Note**
>
> 帳戶管理員 (或管理員使用者) 是具有管理員權限的使用者。如需詳細資訊，請參《IAM 使用者指南》中的 IAM 最佳實務。

授予許可時，您會決定誰取得許可、為了哪些資源而取得許可，以及您允許對這些資源進行的特定動作。

主題

- CodeCommit 資源與營運
- 了解資源所有權

- 管理資源存取
- 資源範圍 CodeCommit
- 指定政策元素：資源、動作、效果和委託人
- 在政策中指定條件

CodeCommit 資源與營運

在中 CodeCommit，主要資源是存放庫。每個資源都有一個相關聯的唯一 Amazon Resource Name (ARN)。在政策中，您使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。如需 ARN 的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 Amazon Resource Name (ARN) 與 AWS 服務命名空間。 CodeCommit 目前不支援稱為子資源的其他資源類型。

下表說明如何指定 CodeCommit 資源。

資源類型	ARN 格式
儲存庫	arn:aws:codecommit:*region*:*account-id* :*repository-name*
所有 CodeCommit 儲存庫	arn:aws:codecommit:*
指定帳戶所擁有的所有 CodeCommit 儲存庫 AWS 區域	arn:aws:codecommit:*region*:*account-id* :*

> **Note**
>
> 大多數 AWS 服務將 ARN 中的冒號 (:) 或斜線 (/) 視為相同字元。但是， CodeCommit 需要完全匹配的資源模式和規則。在建立事件模式時，請務必使用正確的 ARN 字元，使這些字元在資源中符合 ARN 語法。

例如，您可以在您的陳述式中使用其 ARN 指定特定的儲存庫 (*MyDemoRepo*)，如下所示：

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:MyDemoRepo"
```

若要指定屬於特定帳戶的所有儲存庫，請使用萬用字元 (*)，如下所示：

```
"Resource": "arn:aws:codecommit:us-west-2:111111111111:*"
```

若要指定所有資源，或如果特定的 API 動作不支援 ARN，請在 Resource 元素中使用萬用字元 (*)，如下所示：

```
"Resource": "*"
```

您也可以使用萬用字元 (*)，以指定與儲存庫名稱局部相符的所有資源。例如，下列 ARN 指定任何以名稱開頭MyDemo且已註冊至 Amazon Web Services 帳戶111111111111的 CodeCommit儲存庫：us-east-2AWS 區域

```
arn:aws:codecommit:us-east-2:111111111111:MyDemo*
```

如需與 CodeCommit 資源搭配使用的可用作業清單，請參閱CodeCommit 許可參考。

了解資源所有權

Amazon Web Services 帳戶擁有帳戶中建立的資源，無論是誰建立的資源。具體來說，資源擁有者是對資源建立請求進行驗證的主體實體 (即根帳戶、IAM 使用者或 IAM 角色) 的 Amazon Web Services 帳戶。下列範例說明其如何運作：

- 如果您在 Amazon Web Services 帳戶中建立 IAM 使用者，並授與建立 CodeCommit 資源的許可給該使用者，則該使用者可以建立 CodeCommit 資源。但是，使用者所屬的 Amazon Web Services 帳戶擁有資 CodeCommit 源。
- 如果您使用 Amazon Web Services 帳戶的根帳戶登入資料建立規則，則您的 Amazon Web Services 帳戶就是該 CodeCommit 資源的擁有者。
- 如果您在具有建立 CodeCommit 資源許可的 Amazon Web Services 帳戶中建立 IAM 角色，則任何可以擔任該角色的人都可以建立 CodeCommit 資源。角色所屬的 Amazon Web Services 帳戶擁有資 CodeCommit 源。

管理資源存取

若要管理 AWS 資源的存取，請使用許可政策。許可政策描述誰可以存取哪些資源。以下部分說明用來建立許可政策的選項。

> **Note**
>
> 本節討論在的內容中使用 IAM CodeCommit。它不提供 IAM 服務的詳細資訊。如需 IAM 的詳細資訊，請參閱什麼是 IAM？ 在 IAM 使用者指南中。如需有關 IAM 政策語法和說明的資訊，請參閱IAM 使用者指南中的 IAM 政策參考。

附加至 IAM 身分的許可政策稱為身分型政策 (IAM 政策)。連接到資源的許可政策稱為資源型政策。目前僅 CodeCommit支援以身分識別為基礎的政策 (IAM 政策)。

主題

- 身分類型政策 (IAM 政策)
- 資源型政策

身分類型政策 (IAM 政策)

若要管理AWS資源的存取權，請將許可政策附加至 IAM 身分。在中 CodeCommit，您可以使用以身分識別為基礎的原則來控制存放庫的存取。例如，您可以執行下列動作：

- 將權限原則附加至您帳戶中的使用者或群組 — 若要授與使用者檢視 CodeCommit 主控台中 CodeCommit 資源的權限，請將以身分識別為基礎的權限原則附加至使用者所屬的使用者或群組。
- 將權限原則附加至角色 (以授與跨帳號權限) — 委派 (例如當您要授與跨帳號存取權時) 涉及在擁有資源的帳號 (信任帳號) 與包含需要存取資源 (受信任帳號) 之使用者的帳號之間設定信任。許可政策授予角色的使用者對資源執行預定任務所需的許可。信任政策指定允許哪些受信任帳戶授予其使用者擔任角色的許可。如需詳細資訊，請參閱 IAM 術語和概念。

  若要授予跨帳戶許可，請將身分型許可政策附加到 IAM 角色。例如，帳戶 A 中的管理員可以建立角色，將跨帳戶許可授與另一個 Amazon Web Services 帳戶 (例如，帳戶 B) 或AWS服務，如下所示：

  1. 帳戶 A 管理員建立 IAM 角色，並將許可政策連接到可授與帳戶 A 中資源許可的角色。
  2. 帳戶 A 管理員將信任政策連接至該角色，識別帳戶 B 做為可擔任該角的委託人。
  3. 帳戶 B 管理員即可將擔任該角色的許可，委派給帳戶 B 中的任何使用者。這麼做可讓帳戶 B 的使用者建立或存取帳戶 A 的資源。如果您想要將擔任該角色的許可授予 AWS 服務，則信任政策中的委託人也可以是 AWS 服務委託人。如需詳細資訊，請參閱 IAM 術語和概念中的委派。

  如需使用 IAM 來委派許可的詳細資訊，請參閱《IAM 使用者指南》中的存取管理。

以下範例政策允許使用者在名為 *MyDemoRepo* 的儲存庫中建立分支：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:CreateBranch"
      ],
      "Resource" : "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
    }
  ]
}
```

若要限制帳戶中使用者可存取的呼叫和資源，請建立特定的 IAM 政策，然後將這些政策附加到 IAM 使用者。如需有關如何建立 IAM 角色和探索 IAM 政策陳述式範例的詳細資訊 CodeCommit，請參閱客戶管理的身分政策範例。

資源型政策

某些服務 (例如 Amazon S3) 也支援以資源為基礎的許可政策。例如，您可以將資源型政策附加到 S3 儲存貯體，以管理該儲存貯體的存取權限。 CodeCommit 不支援以資源為基礎的政策，但您可以使用標籤來識別資源，然後您可以在 IAM 政策中使用這些資源。如需以標籤為基礎的政策範例，請參閱身分類型政策 (IAM 政策)。

資源範圍 CodeCommit

在中 CodeCommit，您可以將以身分識別為基礎的原則和權限範圍設定為資源，如中所述。CodeCommit 資源與營運但是，您無法將 `ListRepositories` 許可的範圍限定於某個資源。反之，您必須將其範圍開放給所有資源 (使用萬用字元 *)。否則，動作會失敗。

所有其他 CodeCommit 權限可設定為資源的範圍。

指定政策元素：資源、動作、效果和委託人

您可以建立策略以允許或拒絕使用者存取資源，或允許或拒絕使用者對這些資源採取特定動作。CodeCommit 定義一組公用 API 作業，定義使用者如何使用服務，無論是透過 CodeCommit 主控台、SDK AWS CLI、或直接呼叫這些 API。若要授與這些 API 作業的權限，請 CodeCommit 定義您可以在政策中指定的一組動作。

某些 API 操作會需要多個動作的許可。如需資源與 API 操作的詳細資訊，請參閱 CodeCommit 資源與營運 與 CodeCommit 許可參考.

以下是政策的基本元素：

- 資源 — 若要識別政策適用的資源，請使用 Amazon 資源名稱 (ARN)。如需詳細資訊，請參閱CodeCommit 資源與營運。
- 動作 — 若要識別您要允許或拒絕的資源作業，請使用動作關鍵字。例如，根據指定的`Effect`，權限可以允`codecommit:GetBranch`許或拒絕使用者執行`GetBranch`作業，該作業會取得有關 CodeCommit 存放庫中分支的詳細資訊。
- 效果 — 您可以指定當使用者請求特定動作時所發生的效果 (允許或拒絕)。如果您未明確授予存取 (允許) 資源，則隱含地拒絕存取。您也可以明確拒絕存取資源，以確保即使另一個政策授予存取，使用者也無法存取該資源。
- 主體 — 在以身分為基礎的政策 (IAM 政策) 中，唯一 CodeCommit 支援的政策類型，而該政策附加的使用者是隱含的主體。

若要進一步了解 IAM 政策語法，請參閱 IAM 使用者指南中的 IAM 政策參考。

如需顯示所有 CodeCommit API 動作及其套用資源的表格，請參閱CodeCommit 許可參考。

在政策中指定條件

授與許可時，您可以使用 IAM 的存取政策語言來指定政策應在哪些條件下生效。例如，建議只在特定日期之後套用政策。如需有關以政策語言指定條件的詳細資訊，請參閱《IAM 使用者指南》中的條件和政策文法。

欲表示條件，您可以使用預先定義的條件金鑰。沒有 CodeCommit 特定的條件金鑰。不過，您可以使用適合的 AWS 通用條件金鑰。如需全 AWS 通用金鑰的清單，請參閱 IAM 使用者指南中的可用的條件金鑰。

使用以身分為基礎的政策 (IAM 政策) CodeCommit

以下身分型政策範例示範帳戶管理員如何將許可政策附加至 IAM 身分 (使用者、群組和角色)，以授與對資源執行作業的 CodeCommit 權限。

> **Important**
>
> 我們建議您先檢閱介紹性主題，其中說明可用於管理 CodeCommit 資源存取權的基本概念和選項。如需詳細資訊，請參閱管理資 CodeCommit 源存取權限概觀。

主題

- 使用 CodeCommit 主控台所需的許可
- 在主控台檢視資源
- CodeCommit 的 AWS 受管政策
- 客戶受管政策範例

以下是身分型許可政策範例：

```
{
  "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*"
      ]
    }
  ]
}
```

此原則有一個陳述式，可讓使用者取得名為的儲 CodeCommit存 CodeCommit 庫以`MyDestinationRepo`及以該**`us-east-2`**區域`MyDemo`中名稱開頭的所有儲存庫的相關資訊。

使用 CodeCommit 主控台所需的許可

若要查看每個 CodeCommit API 作業的必要權限，以及有關作 CodeCommit 業的詳細資訊，請參閱CodeCommit 許可參考。

若要允許使用者使用主 CodeCommit 控台，系統管理員必須授與其 CodeCommit 動作權限。例如，您可以將AWSCodeCommitPowerUser受管理策略或其對等策略附加到使用者或群組。

除了透過身分型政策而授予使用者的許可， CodeCommit 還需要 AWS Key Management Service (AWS KMS) 動作的許可。IAM 使用者不需要這些動作的明確`Allow`許可，但使用者不得附加任何將下列權限設定為的政策`Deny`：

```
        "kms:Encrypt",
```

```
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:DescribeKey"
```

如需加密的更多資訊 CodeCommit，請參閱AWS KMS和加密。

在主控台檢視資源

主 CodeCommit 控台需要ListRepositories獲得許可，才能顯示您登入之 Amazon Web Services 帳戶的AWS 區域儲存庫清單。主控台還包含 Go to resource (移至資源) 功能，可快速執行不區分大小寫的資源搜尋。此搜尋會在您登入的 Amazon Web Services 帳戶中執行。AWS 區域將會跨以下服務來顯示以下資源：

- AWS CodeBuild：組建專案
- AWS CodeCommit：儲存庫
- AWS CodeDeploy：應用程式
- AWS CodePipeline：管道

若要跨所有服務中的資源執行此搜尋，您必須擁有以下許可：

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

如果您沒有某項服務的許可，則不會傳回該服務之資源的結果。即使您有許可來檢視資源，但如果有明確的 `Deny` 而無法檢視特定資源，則不會傳回這些資源。

CodeCommit 的 AWS 受管政策

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自己撰寫政策更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用 AWS 受管政策。這些政策涵蓋常見的使用案例，並可在您的 AWS 帳戶 中使用。如需有關 AWS 受管政策的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 服務會維護和更新 AWS 受管政策。您無法更改 AWS 受管政策中的許可。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管政策中移除許可，因此政策更新不會破壞您現有的許可。

此外，AWS 支援跨越多項服務之任務職能的受管政策。例如，ReadOnlyAccess AWS 受管政策提供針對所有 AWS 服務和資源的唯讀存取權限。當服務啟動新功能時，AWS 會為新的操作和資源新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的 AWS 受管政策。

AWS 透過提供獨立的 IAM 政策來解決許多常用案例，這些政策由 AWS 所建立與管理。這些 AWS 受管政策授予常用案例所需的許可。的受管政策 CodeCommit 還提供許可，以便在其他服務 (例如 IAM、Amazon SNS 和 Amazon E CloudWatch vents) 中執行操作，視已授予相關政策的使用者的責任所需。例如，該 AWSCodeCommitFullAccess 政策是一個管理層級的使用者政策，允許具有此政策的使用者為存放庫建立和管理 CloudWatch 事件規則 (名稱前綴為的規則`codecommit`) 和 Amazon SNS 主題，以取得有關存放庫相關事件 (名稱前綴為的主題`codecommit`) 的通知，以及管理中的存放庫。CodeCommit

以下可連接到您帳戶中使用者的 AWS 受管政策專屬於 CodeCommit。

主題

- AWS受管理的策略： AWSCodeCommitFullAccess
- AWS受管理的策略： AWSCodeCommitPowerUser
- AWS受管理的策略： AWSCodeCommitReadOnly
- CodeCommit 受管理的策略和通知
- AWS CodeCommit受管政策和 Amazon CodeGuru 審閱者
- CodeCommit AWS受管理策略的更新

AWS受管理的策略： AWSCodeCommitFullAccess

您可將 `AWSCodeCommitFullAccess` 政策連接到 IAM 身分。此政策授予的完整存取權限 CodeCommit。僅將此政策套用至您想要授予對 Amazon Web Services 帳戶中 CodeCommit 儲存庫和相關資源 (包括刪除存放庫的能力) 的完整控制權的管理層級使用者。

該 AWSCodeCommitFullAccess 策略包含以下政策聲明：

```
{
  "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受管理的策略： AWSCodeCommitPowerUser

您可將 AWSCodeCommitPowerUser 政策連接到 IAM 身分。此政策允許使用者存取 CodeCommit 與儲存庫相關的所有功能，但不允許使用者刪除 CodeCommit 儲存庫或在其他服務 (例如 Amazon Events) 中建立或刪除儲存庫相關資源。AWS CloudWatch 建議將此政策套用到大多數使用者。

該 AWSCodeCommitPowerUser 策略包含以下政策聲明：

```
  {
    "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受管理的策略： AWSCodeCommitReadOnly

您可將 AWSCodeCommitReadOnly 政策連接到 IAM 身分。此政策授予對其他AWS服務中資源的唯讀存取權 CodeCommit 和存放庫相關資源，以及建立和管理自己相 CodeCommit關資源的能力 (例如 Git 登入資料和讓 IAM 使用者在存取儲存庫時使用的 SSH 金鑰)。將此政策套用至使用者，以授予他們讀取儲存庫內容的權利 (但不得變更內容)。

該 AWSCodeCommitReadOnly 策略包含以下政策聲明：

```
{
   "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 受管理的策略和通知

AWS CodeCommit 支援通知，可通知使用者儲存庫有重要變更。 CodeCommit 包含通知功能的政策聲明的受管理策略。如需詳細資訊，請參閱什麼是通知？。

完整存取受管政策中的通知相關許可

AWSCodeCommitFullAccess 受管政策包含下列陳述式，允許對通知的完整存取權限。套用此受管政策的使用者也可以針對通知建立和管理 Amazon SNS 主題、訂閱和取消訂閱使用者主題、列出要選擇作為通知規則目標的主題，以及列出針對 Slack 設定的用AWS Chatbot戶端。

```
    {
        "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": "*"
    }
```

唯讀受管政策中的通知相關許可

AWSCodeCommitReadOnlyAccess 受管政策包含下列陳述式，允許對通知的唯讀存取權限。適用此受管政策的使用者可以檢視資源的通知，但無法建立、管理或訂閱通知。

```
  {
        "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": "*"
    }
```

其他受管政策中的通知相關許可

AWSCodeCommitPowerUser 受管政策包含下列陳述式，允許使用者建立、編輯和訂閱通知。使用者無法刪除通知規則或管理資源的標籤。

```
    {
        "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": "*"
    }
```

如需 IAM 和通知的詳細資訊，請參閱通知的 Identity and Access Management。AWS CodeStar

AWS CodeCommit受管政策和 Amazon CodeGuru 審閱者

CodeCommit 支援 Amazon CodeGuru Reviewer，這是一種自動化程式碼檢閱服務，可使用程式分析和機器學習來偵測常見問題，並在 Java 或 Python 程式碼中建議修正程式。 CodeCommit 包含 CodeGuru 審核者功能的原則陳述式的受管理原則。如需詳細資訊，請參閱什麼是 Amazon CodeGuru 審核者。

中與 CodeGuru 審核者相關的權限 AWSCodeCommitFullAccess

`AWSCodeCommitFullAccess`受管理的政策包含下列陳述式，可讓 CodeGuru 審核者與儲存庫產生關聯，以及取消關聯 CodeCommit 性。套用此受管理原則的使用者也可以檢視 CodeCommit儲存庫與 CodeGuru Reviewer 之間的關聯狀態，以及檢視提取要求的檢閱工作狀態。

```
    {
      "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"
        }
      }
    }
```

中與 CodeGuru 審核者相關的權限 AWSCodeCommitPowerUser

AWSCodeCommitPowerUser受管理的原則包含下列陳述式，可讓使用者將儲存區域與 CodeGuru Reviewer 產生關聯，並取消關聯性、檢視關聯狀態，以及檢視提取要求的檢閱工作狀態。

```
    {
      "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"
        }
      }
    }
```

中與 CodeGuru 審核者相關的權限 AWSCodeCommitReadOnly

AWSCodeCommitReadOnlyAccess受管理的原則包含下列陳述式，可讓您以唯讀方式存取「 CodeGuru 審核者」關聯狀態，並檢視提取要求的檢閱工作狀態。套用此受管原則的使用者無法建立或取消儲存庫的關聯。

```
     {
      "Sid": "AmazonCodeGuruReviewerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
            "codeguru-reviewer:DescribeRepositoryAssociation",
            "codeguru-reviewer:ListRepositoryAssociations",
            "codeguru-reviewer:DescribeCodeReview",
            "codeguru-reviewer:ListCodeReviews"
      ],
      "Resource": "*"
    }
```

Amazon CodeGuru 審核者服務連結角色

當您將儲存庫與 CodeGuru 審核者建立關聯時，會建立服務連結角色，以便 CodeGuru 審核者可以偵測問題，並針對提取要求中的 Java 或 Python 程式碼建議修正程式。服務連結角色名為 AWSServiceRoleForAmazonCodeGuruReviewer。如需詳細資訊，請參閱為 Amazon CodeGuru 審核者使用服務連結角色。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

CodeCommit AWS受管理策略的更新

檢視 CodeCommit 自此服務開始追蹤這些變更以來的AWS受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請在訂閱 RSS 摘要AWS CodeCommit 使用者指南文件歷史記錄。

變更	描述	日期
AWS受管理的策略：AWSCodeCommitFullAccess和 AWS受管理的策略：AWSCodeCommitPowerUser — 更新現有政策	CodeCommit 為這些策略添加了權限，以支持使用的其他通知類型AWS Chatbot。 AWSCodeCommitPowerUser 和 AWSCodeCommitFullAccess 原則已變更為新增權限`chatbot:ListMicrosoftTeamsChannelConfigurations` 。	2023 年 5 月 16 日
AWS受管理的策略：AWSCodeCommitReadOnly – 更新現有政策	CodeCommit 從策略中移除重複的權限。 AWSCodeCommitReadOnly 已變更為移除重複的權限，`"iam:ListAccessKeys"` 。	2021 年 8 月 18 日
CodeCommit 開始追蹤變更	CodeCommit 開始追蹤其AWS受管理策略的變更。	2021 年 8 月 18 日

客戶受管政策範例

您可以建立自己的自訂 IAM 政策，以允許 CodeCommit 動作和資源的許可。您可以將這些自訂政策連接至需要這些許可的 IAM 使用者或群組。您也可以建立自己的自訂 IAM 政策，以便在 CodeCommit 與其他AWS服務之間進行整合。

主題

- 客戶管理的身分政策範例
- 客戶管理的整合政策範例

客戶管理的身分政策範例

下列 IAM 政策範例會針對各種 CodeCommit 動作授予許可。使用它們來限制 CodeCommit IAM 使用者和角色的存取權限。這些政策控制能否使用 CodeCommit 主控台、API、AWS 軟體開發套件或 AWS CLI 來執行動作。

> **Note**
>
> 所有範例皆使用美國西部 (奧勒岡) 區域 (us-west-2) 並包含虛構帳戶 ID。

範例

- 範例 1：允許使用者在單一 CodeCommit 執行作業 AWS 區域
- 範例 2：允許使用者將 Git 用於單一儲存庫
- 範例 3：允許使用者從指定的 IP 位址範圍連線存取儲存庫
- 範例 4：拒絕或允許對分支執行動作
- 範例 5：拒絕或允許對具有標籤的儲存庫執行動作

範例 1：允許使用者在單一 CodeCommit 執行作業 AWS 區域

下列權限原則會使用萬用字元 (`"codecommit:*"`) 來允許使用者在 us-east-2 區域中執行所有 CodeCommit動作，而不是從其他地區執行。AWS 區域

```
{
    "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"
                }
            }
        }
    ]
}
```

範例 2：允許使用者將 Git 用於單一儲存庫

在中 CodeCommit，`GitPull`IAM 政策許可適用於從中擷取資料的任何 Git 用戶端命令 CodeCommit git fetchgit clone，包括、等等。同樣地，`GitPush`IAM 政策許可適用於資料傳送至的任何 Git 用戶端命令 CodeCommit。例如，如果 `GitPush` IAM 政策權限設定為`Allow`，則使用者可以使用 Git 通訊協定推送刪除分支。該推送不受該 IAM 使用者`DeleteBranch`作業套用的任何許可影響。`DeleteBranch` 許可適用於以主控台、AWS CLI、軟體開發套件和 API 執行的動作，但不包括 Git 通訊協定。

下列範例允許指定的使用者從名為的 CodeCommit 存放庫中提取及推送至`MyDemoRepo`：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codecommit:GitPull",
        "codecommit:GitPush"
      ],
```

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

範例 3：允許使用者從指定的 IP 位址範圍連線存取儲存庫

您可以建立政策，只允許 IP 地址在特定 IP 地址範圍內的使用者才能連接到 CodeCommit 儲存庫。這有兩種同樣有效的方法。如果使用者的 Deny IP 位址不在特定區塊內，您可以建立不允許 CodeCommit 作業的原則，或者您可以建立Allow策略，以便在使用者的 IP 位址位於特定區塊內時允許 CodeCommit 作業。

您可以建立 Deny 政策，以拒絕所有不在特定 IP 範圍內的使用者存取。例如，您可以將 AWSCodeCommitPowerUser 受管政策和客戶受管政策，附加到所有需要存取儲存庫的使用者。下列範例原則會拒絕其 IP 位址不在指定 IP 位址區塊 203.0.113.0/16 內的使用者的所有 CodeCommit 權限：

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": [
            "codecommit:*"
         ],
         "Resource": "*",
         "Condition": {
            "NotIpAddress": {
               "aws:SourceIp": [
                  "203.0.113.0/16"
               ]
            }
         }
      }
   ]
}
```

下列範例原則只 MyDemoRepo 有在指定的 IP 位址位於 203.0.113.0/16 的指定位址區塊內時，才允許指定的使用者 CodeCommit存取以 AWSCodeCommitPowerUser 受管理策略對等權限命名的存放庫：

```
{
   "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"
               ]
            }
         }
      }
   ]
}
```

範例 4：拒絕或允許對分支執行動作

您可以建立政策，對使用者拒絕在一或多個分支執行您指定動作的許可。或者，您可以建立政策，以允許他們原本在儲存庫的其他分支上可能沒有的一或多個分支上執行動作。您可以使用這些政策搭配適當的受管 (預先定義) 政策。如需詳細資訊，請參閱限制推送和合併到分支 AWS CodeCommit。

例如，您可以建立一個Deny策略，拒絕使用者在名為的存放庫中對名為 main 的分支進行變更，包括刪除該分支。*MyDemoRepo*您可以使用此政策搭配 AWSCodeCommitPowerUser 受管政策。套用這兩個原則的使用者將能夠建立和刪除分支、建立提取要求，以及所有其他允許的動作 AWSCodeCommitPowerUser，但是他們無法將變更推送到名為 main 的分支、在主 CodeCommit 控台的 main 分支中新增或編輯檔案，或將分支或提取要求合併到主分支。由於 Deny 套用到

GitPush，您必須在政策中包含 Null 陳述式，以便使用者從本機儲存庫推送時，允許分析初始 GitPush 呼叫的有效性。

> **Tip**
>
> 如果您想要建立適用於 Amazon Web Services 帳戶中所有儲存庫中名為 main 的所有分支的政策，請針對Resource指定星號 (*) 而不是儲存庫 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"
                }
            }
        }
    ]
}
```

下列範例政策允許使用者變更 Amazon Web Services 帳戶中所有儲存庫中名為 main 的分支。它不允許更改任何其他分支。您可以將此原則與 AWSCodeCommitReadOnly 受管理的原則搭配使用，以允許自動推送至主分支中的存放庫。由於 Effect 是 Allow，這個範例政策無法搭配 AWSCodeCommitPowerUser 之類的受管政策一起使用。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:GitPush",
                "codecommit:Merge*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "codecommit:References": [
                        "refs/heads/main"
                    ]
                }
            }
        }
    ]
}
```

範例 5：拒絕或允許對具有標籤的儲存庫執行動作

您可以建立政策，根據與這些儲存庫相關聯的AWS標記，允許或拒絕存放庫上的動作，然後將這些政策套用至為管理 IAM 使用者設定的 IAM 群組。*##################AWS#### S tatus ###### Secret ########## CodeCommit ##################### (####) ### IAM ###*然後，您需要確定在這些標記存放庫上工作的開發人員不是該一般*###*員群組的成員，而是屬於未套用限制性政策的其他 IAM 群組 (SecretDevelopers)。

下列範例會拒絕儲存庫上標記為 S tatus 索引鍵和 Secret 索引鍵值的所有 CodeCommit 動作：

```
{
  "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"
        }
      }
    }
  ]
}
```

您可以將特定存放庫 (而非所有儲存庫) 指定為資源，進一步精簡此策略。您也可以建立政策，允許 CodeCommit 對未使用特定標記標記的所有儲存庫執行動作。例如，下列原則允許對等 CodeCommit 動作使用AWSCodeCommitPowerUser權限，但它只允許對未標記指定標記的儲存庫 CodeCommit 執行動作：

> **Note**
>
> 此原則範例僅包含的動作 CodeCommit。它不包括AWSCodeCommitPowerUser受管理策略中包含的其他AWS服務的處理行動。如需詳細資訊，請參閱 AWS受管理的策略：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": "*"
    }
  ]
}
```

客戶管理的整合政策範例

本節提供範例由客戶管理的使用者政策，這些政策授與 CodeCommit 與其他AWS服務之間的整合權限。關於允許跨帳戶存取 CodeCommit 儲存庫的特定政策範例，請參閱使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取。

> **Note**
>
> 所有範例都會在需要時使用美國西部 (奧勒岡) 區域 (us-west-2)，並包含虛擬帳戶 ID。AWS 區域

範例

- 範例 1：建立允許跨帳戶存取 Amazon SNS 主題的政策
- 範例 2：建立 Amazon Simple Notification Service (Amazon SNS) 主題政策，以允許 Amazon CloudWatch 事件將 CodeCommit 事件發佈到主題
- 範例 3：建立與 CodeCommit 觸發器AWS Lambda整合的策略

範例 1：建立允許跨帳戶存取 Amazon SNS 主題的政策

您可以設定 CodeCommit 儲存庫，讓程式碼推送或其他事件觸發動作，例如從 Amazon 簡單通知服務 (Amazon SNS) 傳送通知。如果您使用用於建立 CodeCommit儲存庫的相同帳戶建立 Amazon SNS 主題，則不需要設定其他 IAM 政策或許可。您可以建立主題，然後為儲存庫建立觸發。如需詳細資訊，請參閱為 Amazon SNS 主題創建觸發器。

但是，如果您想要將觸發器設定為在另一個 Amazon Web Services 帳戶中使用 Amazon SNS 主題，則必須先使用允許發佈 CodeCommit 到該主題的政策來設定該主題。從該其他帳戶開啟 Amazon SNS 主控台，從清單中選擇主題，然後針對「其他主題動作」 選擇 「編輯主題政策」。在 [進階] 索引標籤上，修改主題的原則以允許發佈 CodeCommit 至該主題。例如，如果該政策是預設政策，您可以按如下方式修改政策，以######變更項目，以符合儲存庫、Amazon SNS 主題和帳戶的值：

```
{
  "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"
        }
      }
    }
  ]
}
```

範例 2：建立 Amazon Simple Notification Service (Amazon SNS) 主題政策，以允許 Amazon CloudWatch 事件將 CodeCommit 事件發佈到主題

您可以將 CloudWatch 事件設定為在事件發生時 (包括 CodeCommit 事件) 發佈到 Amazon SNS 主題。若要這麼做，您必須確定 E CloudWatch vents 具有將事件發佈到 Amazon SNS 主題的權限，方法是為主題建立政策或修改類似下列主題的現有政策：

```
{
  "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"
    }
  ]
```

```
}
```

如需 CodeCommit 和 CloudWatch 事件的詳細資訊，請參閱來自支援服務的CloudWatch 事件事件範例。如需 IAM 和政策語言的詳細資訊，請參閱 IAM JSON 政策語言的文法。

範例 3：建立與 CodeCommit 觸發器AWS Lambda整合的策略

您可以設定 CodeCommit 存放庫，讓程式碼推送或其他事件觸發動作，例如叫用中的函數。AWS Lambda如需詳細資訊，請參閱為 Lambda 函數建立觸發器。此資訊僅適用於觸發程序，而非「 CloudWatch 事件」。

如果您希望觸發器直接執行 Lambda 函數 (而不是使用 Amazon SNS 主題叫用 Lambda 函數)，而不是在 Lambda 主控台中設定觸發器，則必須在函數的以資源為基礎的政策中包含類似以下內容的陳述式：

```
{
  "Statement":{
     "StatementId":"Id-1",
     "Action":"lambda:InvokeFunction",
     "Principal":"codecommit.amazonaws.com",
     "SourceArn":"arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo",
     "SourceAccount":"111111111111"
  }
}
```

手動設定叫用 Lambda 函數的 CodeCommit 觸發器時，您也必須使用 Lambda AddPermission命令授與呼叫函數 CodeCommit 的權限。如需範例，請參閱為現有 Lambda 函數建立觸發器的若要允 CodeCommit 許執行 Lambda 函數一節。

如需 Lambda 函數之資源政策的詳細資訊，請參閱AddPermissionAWS Lambda開發人員指南中的提取/推送事件模型。

CodeCommit 許可參考

下表列出每個 CodeCommit API 作業、您可以授與權限的對應動作，以及用於授與權限的資源 ARN 格式。 CodeCommit API 會根據該 API 允許的動作範圍分組到表格中。在設定存取控制和撰寫可附加至 IAM 身分 (身分型政策) 的許可政策時，請參閱此說明。

當您建立許可政策時，您需要在政策的 `Action` 欄位中指定動作。您需要在政策的 `Resource` 欄位中指定資源值做為 ARN，可包含或不含萬用字元 (*)。

若要表示 CodeCommit 原則中的條件，請使用AWS寬條件金鑰。如需 AWS 通用金鑰的完整清單，請參閱《IAM 使用者指南》中的可用的金鑰。如需 IAM 政策中的動作、資源和條件金鑰 CodeCommit 的完整資訊，請參閱的動作、資源和條件金鑰AWS CodeCommit。

> **Note**
>
> 若要指定動作，請使用 `codecommit:` 字首，後面接著 API 操作名稱 (例如 `codecommit:GetRepository` 或 `codecommit:CreateRepository`)。

使用萬用字元

若要指定多個動作或資源，請在 ARN 中使用萬用字元 (*)。例如，`codecommit:*`指定所有 CodeCommit 動作，並`codecommit:Get*`指定以該字開頭的所有 CodeCommit 動作`Get`。以下範例授予對所有以 `MyDemo` 為名稱開頭之儲存庫的存取權。

```
arn:aws:codecommit:us-west-2:111111111111:MyDemo*
```

您只能針對下表列出的 *`repository-name`* 資源使用萬用字元：您不能對 *`region`* 或 *`account-id`* 資源使用萬用字元。如需有關萬用字元的詳細資訊，請參閱 IAM 使用者指南中的 IAM 識別碼。

主題

- Git 客戶端命令所需的權限
- 在分支上執行動作的權限
- 合併動作的權限
- 提取要求動作的權限
- 核准規則範本動作的權限
- 對個別檔案執行動作的權限
- 註解動作的權限
- 對已提交程式碼的動作權限
- 儲存庫動作的權限
- 對標籤動作的權限
- 觸發器上動作的權限

- CodePipeline 整合動作的權限

Git 客戶端命令所需的權限

在中 CodeCommit，GitPullIAM 政策許可適用於從中擷取資料的任何 Git 用戶端命令 CodeCommit git fetchgit clone，包括、等等。同樣地，GitPushIAM 政策許可適用於資料傳送至的任何 Git 用戶端命令 CodeCommit。例如，如果 GitPush IAM 政策權限設定為Allow，則使用者可以使用 Git 通訊協定推送刪除分支。該推送不受該 IAM 使用者DeleteBranch作業套用的任何許可影響。DeleteBranch 許可適用於以主控台、AWS CLI、軟體開發套件和 API 執行的動作，但不包括 Git 通訊協定。

GitPull並且GitPush是 IAM 政策許可。不是 API 動作。

CodeCommit Git 用戶端命令動作所需的權限

GitPull

動作：codecommit:GitPull

需要將信息從 CodeCommit 存儲庫提取到本地回購。這只是 IAM 政策許可，不是 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GitPush

動作：codecommit:Git Push

需要將信息從本地回購推送到存 CodeCommit 儲庫。這只是 IAM 政策許可，不是 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

在分支上執行動作的權限

下列權限允許或拒絕 CodeCommit 儲存庫中分支的動作。這些權限僅與在 CodeCommit 主控台和 CodeCommit API 中執行的動作有關，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。例如，git show-branch -r 命令會顯示儲存庫的遠端分支清單，及其使用 Git 通訊協定所做的遞交。它不受作業的任何權限影 CodeCommit ListBranches響。

如需有關分支原則的詳細資訊，請參閱限制推送和合併到分支 AWS CodeCommit和客戶受管政策範例。

CodeCommit 在分支上執行動作的 API 作業和必要權限

CreateBranch

動作：codecommit:CreateBranch

在 CodeCommit 存放庫中建立分支所需。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DeleteBranch

動作：codecommit:DeleteBranch

需要從 CodeCommit 存放庫中刪除分支。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetBranch

動作：codecommit:GetBranch

需要獲取有關 CodeCommit 存儲庫中分支的詳細信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

ListBranches

動作：codecommit:ListBranches

需要獲取存 CodeCommit 儲庫中的分支列表。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesByFastForward

動作：codecommit:MergeBranchesByFastForward

需要使用 CodeCommit 儲存庫中的快進合併策略來合併兩個分支。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesBySquash

動作：codecommit:ListBranches

在 CodeCommit 儲存庫中使用 squash 策略合併兩個分支時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergeBranchesByThreeWay

動作：codecommit:ListBranches

在 CodeCommit 儲存庫中使用三向策略合併兩個分支時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdateDefaultBranch

動作：codecommit:UpdateDefaultBranch

需要變更 CodeCommit 存放庫中的預設分支。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

合併動作的權限

下列權限允許或拒絕 CodeCommit 儲存庫中合併的動作。這些權限與使用 CodeCommit 控制台和 CodeCommit API 執行的操作有關，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。如需分支的相關許可，請參閱在分支上執行動作的權限。如需提取請求的相關許可，請參閱提取要求動作的權限。

CodeCommit 合併命令動作的 API 作業和必要權限

BatchDescribeMergeConflicts

動作：codecommit:BatchDescribeMergeConflicts

需要返回有關 CodeCommit 存儲庫中提交之間合併中衝突的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreateUnreferencedMergeCommit

動作：codecommit:CreateUnreferencedMergeCommit

需要在兩個分支之間創建未引用的提交或 CodeCommit 存儲庫中的提交，以便比較它們並識別任何潛在的衝突。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DescribeMergeConflicts

動作：codecommit:DescribeMergeConflicts

傳回 CodeCommit 儲存庫中潛在合併之檔案的基礎、來源和目的地版本之間合併衝突的相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeCommit

動作：codecommit:GetMergeCommit

需要返回有關 CodeCommit 存儲庫中源和目標提交之間合併的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeOptions

動作：codecommit:GetMergeOptions

需要返回有關 CodeCommit 存儲庫中兩個分支之間的可用合併選項或提交說明符的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

提取要求動作的權限

以下許可允許或拒絕在 CodeCommit 儲存庫的提取請求上執行動作。這些權限與使用 CodeCommit 控制台和 CodeCommit API 執行的操作有關，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。如需註解的相關許可，請參閱註解動作的權限。

CodeCommit 提取要求動作的 API 作業和必要權限

BatchGetPullRequests

動作：codecommit:BatchGetPullRequests

傳回 CodeCommit 儲存庫中一或多個提取請求的相關資訊時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreatePullRequest

動作：codecommit:CreatePullRequest

需要在 CodeCommit 儲存庫中建立提取要求。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreatePullRequestApprovalRule

動作：codecommit:CreatePullRequestApprovalRule

為 CodeCommit 儲存庫中的提取請求建立核准規則時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DeletePullRequestApprovalRule

動作：codecommit:DeletePullRequestApprovalRule

對 CodeCommit 儲存庫中的提取請求刪除核准規則時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DescribePullRequestEvents

動作：codecommit:DescribePullRequestEvents

需要返回有關 CodeCommit 存儲庫中一個或多個提取請求事件的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

EvaluatePullRequestApprovalRules

動作：codecommit:EvaluatePullRequestApprovalRules

評估提取請求是否符合 CodeCommit 儲存庫中其相關核准規則中指定的所有條件是必要的。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentsForPullRequest

動作：codecommit:GetCommentsForPullRequest

傳回在提取請求上所做的註解時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommitsFromMergeBase

動作：codecommit:GetCommitsFromMergeBase

在潛在合併情況下傳回遞交之間差異的相關資訊時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetMergeConflicts

動作：codecommit:GetMergeConflicts

需要返回有關提取請求中源和目的地分支之間合併衝突的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequest

動作：codecommit:GetPullRequest

傳回 CodeCommit 儲存庫中提取請求的相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequestApprovalStates

動作：codecommit:GetPullRequestApprovalStates

傳回所指定提取請求的核准狀態相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetPullRequestOverrideState

動作：codecommit:GetPullRequestOverrideState

需要傳回有關是否已為提取請求設定 (覆寫) 核准規則的資訊，如果是，則傳回使用者的 Amazon 資源名稱 (ARN) 或覆寫規則及其對提取請求的要求的資訊。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

ListPullRequests

動作：codecommit:ListPullRequests

列出儲存庫中的提取請求時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestByFastForward

動作：codecommit:MergePullRequestByFastForward

關閉提取請求和嘗試使用向前快轉合併策略將提取請求的來源分支合併到目的地分支時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestBySquash

動作：codecommit:MergePullRequestBySquash

關閉提取請求和嘗試使用 squash 合併策略將提取請求的來源分支合併到目的地分支時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

MergePullRequestByThreeWay

動作：codecommit:MergePullRequestByThreeWay

關閉提取請求和嘗試使用三方合併選項將提取請求的來源分支合併到目的地分支時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

OverridePullRequestApprovalRules

動作：codecommit:OverridePullRequestApprovalRules

需要為 CodeCommit 儲存庫中的提取要求預留所有核准規則需求。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentForPullRequest

動作：codecommit:PostCommentForPullRequest

在 CodeCommit 儲存庫中的提取請求上張貼註解時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestApprovalRuleContent

動作：codecommit:UpdatePullRequestApprovalRuleContent

需要變更存 CodeCommit 放庫中提取要求的核准規則結構。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestApprovalState

動作：codecommit:UpdatePullRequestApprovalState

更新 CodeCommit 儲存庫中提取要求的核准狀態所需。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestDescription

動作：codecommit:UpdatePullRequestDescription

變更 CodeCommit 儲存庫中提取請求的描述時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestStatus

動作：codecommit:UpdatePullRequestStatus

變更 CodeCommit 儲存庫中提取請求的狀態時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdatePullRequestTitle

動作：codecommit:UpdatePullRequestTitle

變更 CodeCommit 儲存庫中提取請求的標題時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

核准規則範本動作的權限

以下許可允許或拒絕在 CodeCommit 儲存庫的核准規則範本上執行動作。這些權限僅適用於在 CodeCommit主控台、 CodeCommit API 中執行的動作，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。如需提取請求的相關許可，請參閱提取要求動作的權限。

CodeCommit 核准規則範本動作的 API 作業和必要權限

AssociateApprovalRuleTemplateWithRepository

動作：codecommit:AssociateApprovalRuleTemplateWithRepository

需要將範本與 Amazon Web Services 帳戶中的指定儲存庫建立關聯。建立關聯後，這會自動建立核准規則，讓指定儲存庫中建立的每個提取請求都符合範本條件。

資源：*

BatchAssociateApprovalRuleTemplateWithRepositories

動作：codecommit:BatchAssociateApprovalRuleTemplateWithRepositories

需要將範本與 Amazon Web Services 帳戶中的一或多個指定儲存庫建立關聯。

資源：*

BatchDisassociateApprovalRuleTemplateFromRepositories

動作：codecommit:BatchDisassociateApprovalRuleTemplateFromRepositories

必須取消範本與 Amazon Web Services 帳戶中一或多個指定儲存庫的關聯。

資源：*

CreateApprovalRuleTemplate

動作：codecommit:CreateApprovalRuleTemplate

需要建立核准規則範本，然後可以與 Amazon Web Services 帳戶中的一個或多個儲存庫建立關聯。

資源：*

DeleteApprovalRuleTemplate

動作：codecommit:DeleteApprovalRuleTemplate

從 AWS 帳戶中刪除核准規則範本時需要。

資源：*

DisassociateApprovalRuleTemplateFromRepository

動作：codecommit:DisassociateApprovalRuleTemplateFromRepository

必須取消指定範本與 Amazon Web Services 帳戶中的儲存庫的關聯。在已使用範本建立的提取請求上不會移除核准規則。

資源：*

GetApprovalRuleTemplate

動作：codecommit:GetApprovalRuleTemplate

必須傳回 Amazon Web Services 帳戶中核准規則範本的相關資訊。

資源：*

ListApprovalRuleTemplates

動作：codecommit:ListApprovalRuleTemplates

必須在 Amazon Web Services 帳戶中列出核准規則範本。

資源：*

ListAssociatedApprovalRuleTemplatesForRepository

動作：codecommit:ListAssociatedApprovalRuleTemplatesForRepository

必須列出與 Amazon Web Services 帳戶中指定儲存庫相關聯的所有核准規則範本。

資源：*

ListRepositoriesForApprovalRuleTemplate

動作：codecommit:ListRepositoriesForApprovalRuleTemplate

必須列出與 Amazon Web Services 帳戶中指定核准規則範本相關聯的所有儲存庫。

資源：*

UpdateApprovalRuleTemplateContent

動作：codecommit:UpdateApprovalRuleTemplateContent

需要更新 Amazon Web Services 帳戶中核准規則範本的內容。

資源：*

UpdateApprovalRuleTemplateDescription

動作：codecommit:UpdateApprovalRuleTemplateDescription

需要更新 Amazon Web Services 帳戶中核准規則範本的說明。

資源：*

UpdateApprovalRuleTemplateName

動作：codecommit:UpdateApprovalRuleTemplateName

更新 AWS 帳戶中核准規則範本的名稱時需要。

資源：*

對個別檔案執行動作的權限

以下許可允許或拒絕在 CodeCommit 儲存庫的個別檔案上執行動作。這些權限僅適用於在 CodeCommit主控台、 CodeCommit API 中執行的動作，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。例如，`git push` 命令使用 Git 通訊協定將新的和變更的檔案推送到 CodeCommit 儲存庫。它不受作業的任何權限影 CodeCommit `PutFile`響。

CodeCommit 針對個別檔案執行動作的 API 作業和必要權限

DeleteFile

動作：`codecommit:DeleteFile`

需要從 CodeCommit 控制台從 CodeCommit存儲庫中的指定分支中刪除指定的文件。

資源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetBlob

動作：`codecommit:GetBlob`

需要從 CodeCommit 主控台檢視 CodeCommit儲存庫中個別檔案的編碼內容。

資源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetFile

動作：`codecommit:GetFile`

需要從 CodeCommit 主控台檢視 CodeCommit 存放庫中指定檔案的編碼內容及其中繼資料。

資源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

GetFolder

動作：`codecommit:GetFolder`

需要從 CodeCommit 主控台檢視 CodeCommit存放庫中指定資料夾的內容。

資源： `arn:aws:codecommit:`*`region`*`:`*`account-id`*`:`*`repository-name`*

PutFile

動作：codecommit:PutFile

需要從 CodeCommit 主控台、 CodeCommit API 或將新檔案或修改過的檔案新增至 CodeCommit 儲存庫AWS CLI。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

註解動作的權限

下列權限允許或拒絕 CodeCommit 儲存庫中註解的動作。這些權限與使用 CodeCommit 控制台和 CodeCommit API 執行的操作有關，以及使用AWS CLI. 如需提取請求中註解的相關許可，請參閱提取要求動作的權限。

CodeCommit 存放庫上動作的 API 作業和必要權限

DeleteCommentContent

動作：codecommit:DeleteCommentContent

將儲存庫中的變更、檔案或遞交所做的註解內容刪除時需要。無法刪除註解，但可以移除註解的內容 (如果使用者具有此許可)。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetComment

動作：codecommit:GetComment

需要返回有關在 CodeCommit 存儲庫中對更改，文件或提交的註釋的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentReactions

動作：codecommit:GetCommentReactions

需要將有關表情符號反應的信息返回到對 CodeCommit 存儲庫中的更改，文件或提交的註釋。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommentsForComparedCommit

動作：codecommit:GetCommentsForComparedCommit

需要返回有關 CodeCommit 存儲庫中兩個提交之間進行比較的註釋的信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentForComparedCommit

動作：codecommit:PostCommentForComparedCommit

對 CodeCommit 儲存庫中兩個遞交之間的比較做註解時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PostCommentReply

動作：codecommit:PostCommentReply

需要創建對提交之間的比較或 CodeCommit 存儲庫中提取請求的評論的回复。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PutCommentReaction

動作：codecommit:PutCommentReaction

需要在提交或 CodeCommit 存儲庫中的拉取請求回复帶有表情符號的評論。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdateComment

動作：codecommit:UpdateComment

在遞交之間的比較或提取請求上編輯註解時需要。只有註解作者才能編輯註解。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

對已提交程式碼的動作權限

以下許可允許或拒絕在遞交到 CodeCommit 儲存庫的程式碼上執行動作。這些權限與使用 CodeCommit 控制台和 CodeCommit API 執行的操作有關，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。例如，git commit 命令使用 Git 通訊協定為儲存庫中的分支建立遞交。它不受作業的任何權限影 CodeCommit CreateCommit響。

明確拒絕其中一些權限可能會導致 CodeCommit 主控台產生非預期的後果。例如，將 GetTree 設為 Deny 可防止使用者在主控台瀏覽儲存庫的內容，但無法阻止使用者檢視儲存庫中某個檔案的內容 (例

如，如果以電子郵件將檔案的連結傳送給他們)。將 GetBlob 設為 Deny 可防止使用者檢視檔案的內容，但無法阻止使用者瀏覽儲存庫的結構。將 GetCommit 設為 Deny 可防止使用者擷取遞交的詳細資訊。將 GetObjectIdentifier 設為 Deny 可封鎖程式碼瀏覽的大部分功能。如果您Deny在策略中將這三個動作全部設定為，則具有該策略的使用者無法在 CodeCommit 主控台中瀏覽程式碼。

CodeCommit 針對已提交程式碼執行動作的 API 作業和必要權限

BatchGetCommits

動作：codecommit:BatchGetCommits

傳回 CodeCommit 儲存庫中一或多個遞交的相關資訊時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreateCommit

動作：codecommit:CreateCommit

建立遞交時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommit

動作：codecommit:GetCommit

傳回遞交的相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommitHistory

動作：codecommit:GetCommitHistory

傳回儲存庫中遞交歷史記錄的相關資訊時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetDifferences

動作：codecommit:GetDifferences

傳回遞交指標 (例如，分支、標籤、HEAD、遞交 ID，或其他完整參考) 中的差異相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetObjectIdentifier

動作：codecommit:GetObjectIdentifier

將 Blob、樹狀結構和遞交解析為其識別符時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetReferences

動作：codecommit:GetReferences

傳回所有參考時需要，例如分支和標籤。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetTree

動作：codecommit:GetTree

需要從 CodeCommit 控制台查看 CodeCommit存儲庫中指定樹的內容。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

儲存庫動作的權限

下列權限允許或拒絕 CodeCommit 存放庫上的動作。這些權限與使用 CodeCommit 控制台和 CodeCommit API 執行的操作有關，以及使用AWS CLI. 不適用於可使用 Git 通訊協定執行的類似動作。

CodeCommit 存放庫上動作的 API 作業和必要權限

BatchGetRepositories

動作：codecommit:BatchGetRepositories

需要獲取有關 Amazon Web Services 帳戶中多個 CodeCommit 存儲庫的信息。在中Resource，您必須指定允許 (或拒絕) 使用者資訊之所有 CodeCommit 存放庫的名稱。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CreateRepository

動作：codecommit:CreateRepository

建立存 CodeCommit 放庫所需。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

DeleteRepository

動作：codecommit:DeleteRepository

刪除存 CodeCommit 放庫所需。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetRepository

動作：codecommit:GetRepository

取得單一 CodeCommit 儲存庫相關資訊所需。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

ListRepositories

動作：codecommit:ListRepositories

需要取得 Amazon Web Services 帳戶多個 CodeCommit儲存庫的名稱和系統 ID 清單。針對此動作，Resource 唯一允許的值是所有儲存庫 (*)。

資源： *

UpdateRepositoryDescription

動作：codecommit:UpdateRepositoryDescription

需要變更存 CodeCommit 放庫的描述。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UpdateRepositoryName

動作：codecommit:UpdateRepositoryName

需要變更存 CodeCommit 放庫的名稱。在中Resource，您必須指定允許變更的存放 CodeCommit 庫和新的存放庫名稱。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

對標籤動作的權限

下列權限允許或拒絕 CodeCommit 資源AWS標籤上的動作。

CodeCommit 標籤動作的 API 作業和必要權限

ListTagsForResource

動作：codecommit:ListTagsForResource

傳回在 CodeCommit 的資源上所設定 AWS 標籤的相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

TagResource

動作：codecommit:TagResource

新增或編輯儲存庫的 AWS 標籤時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UntagResource

動作：codecommit:UntagResource

需要從中的資源中移除AWS標籤 CodeCommit。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

觸發器上動作的權限

以下許可允許或拒絕在 CodeCommit 儲存庫的觸發上執行動作。

CodeCommit 觸發器上動作的 API 作業和必要權限

GetRepositoryTriggers

動作：codecommit:GetRepositoryTriggers

傳回為儲存庫而設定的觸發相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

PutRepositoryTriggers

動作：codecommit:PutRepositoryTriggers

為儲存庫建立、編輯或刪除觸發時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

TestRepositoryTriggers

動作：codecommit:TestRepositoryTriggers

將資料傳送到為觸發而設定的主題或函數以測試儲存庫觸發的功能時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CodePipeline 整合動作的權限

CodePipeline 若要在管線的來源動作中使用 CodeCommit 存放庫，您必須將下表中列出的所有權限授與的服務角色 CodePipeline。如果服務角色中未設定這些許可，或設定為 **Deny**，則變更儲存庫時不會自動執行管道，也無法手動發佈變更。

CodeCommit CodePipeline 整合動作的 API 作業和必要權限

GetBranch

動作：codecommit:GetBranch

需要獲取有關 CodeCommit 存儲庫中分支的詳細信息。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetCommit

動作：codecommit:GetCommit

傳回遞交的相關資訊時需要。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

UploadArchive

動作：codecommit:UploadArchive

允許的服務角色將存放庫變更上載 CodePipeline 至管線的必要項目。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

GetUploadArchiveStatus

動作：codecommit:GetUploadArchiveStatus

判斷存檔上傳的狀態時需要，無論是進行中、完成、已取消或發生錯誤。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

CancelUploadArchive

動作：codecommit:CancelUploadArchive

取消將存檔上傳至管道時需要。這僅是 IAM 政策許可，而不是您可以呼叫的 API 動作。

資源： arn:aws:codecommit:*region*:*account-id*:*repository-name*

AWS CodeCommit 搭配 IAM 的運作方式

在您使用 IAM 管理存取權限之前 CodeCommit，您應該瞭解哪些 IAM 功能可搭配使用 CodeCommit。若要取得 CodeCommit 和其他 AWS 服務如何使用 IAM 的詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 的 AWS 服務。

主題

條件金鑰

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素（或 Condition 區塊）可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式（例如等於或小於），來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授與該 IAM 使用者。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定的條件金鑰。若要查看 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的 AWS 全域條件內容金鑰。

CodeCommit 定義了它自己的一組條件鍵，並且還支持使用一些全局條件鍵。若要查看 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的 AWS 全域條件內容金鑰。

某些 CodeCommit 動作支援`codecommit:References`條件索引鍵。有關使用此金鑰的政策範例，請參閱範例 4：拒絕或允許對分支執行動作。

若要查看 CodeCommit 條件金鑰清單，請參閱 IAM 使用者指南AWS CodeCommit中的條件金鑰。若要了解您可以針對何種動作及資源使用條件索引鍵，請參閱 AWS CodeCommit 定義的動作。

範例

若要檢視以 CodeCommit 身分為基礎的原則範例，請參閱。AWS CodeCommit 身分型政策範例

CodeCommit 資源型政策

CodeCommit 不支援以資源為基礎的政策。

基於 CodeCommit 標籤的授權

您可以將標籤附加至 CodeCommit 資源，或將要求中的標籤傳遞給 CodeCommit。若要根據標籤控制存取，請使用 `codecommit:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。如需標記 CodeCommit 資源的更多資訊，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。如需有關標記策略的詳細資訊，請參閱標記 AWS 資源。

CodeCommit 也支援以工作階段標記為基礎的策略。如需詳細資訊，請參閱工作階段標籤。

使用標籤提供身份信息 CodeCommit

CodeCommit 支援使用工作階段標籤，這些標籤是當您擔任 IAM 角色、使用臨時登入資料或在 AWS Security Token Service () AWS STS 中聯合使用者時傳遞的索引鍵值配對屬性。您也可以將這類標籤與 IAM 使用者建立關聯。您可以使用這些標籤中提供的資訊，以便更輕鬆地識別誰進行了變更或造成了事件。 CodeCommit 在 CodeCommit 事件中包含具有下列索引鍵名稱的標籤值：

金鑰名稱	Value
displayName	您要顯示並與使用者建立關聯之可供人閱讀的名稱 (如 Mary Major 或 Saanvi Sarkar)。
emailAddress	您想要顯示並與使用者建立關聯的電子郵件地址 (如 mary_major@example.com 或 saanvi_sarkar@example.com)。

如果提供此信息，請將其 CodeCommit 包含在發送到 Amazon EventBridge 和 Amazon CloudWatch 活動的事件中。如需詳細資訊，請參閱監控 CodeCommit Amazon 事件 EventBridge 和 Amazon CloudWatch 活動。

角色的政策必須包含設為 Allow 的 sts:TagSession 許可，才能夠使用工作階段標記。如果您是使用聯合身分存取，便可在設定時配置顯示名稱與電子郵件標籤資訊。例如，如果您使用的是 Azure Active Directory，則可提供下列宣告資訊：

宣告名稱	Value
https://aws.amazon.com/SAML/Attributes/PrincipalTag:displayNam e	user.displayname
https://aws.amazon.com/SAML/Attributes/PrincipalTag:emailAddress	user.mail

您可以利用 AWS CLI 來透過 AssumeRole 傳遞 displayName 和 emailAddress 的工作階段標籤。例如，某位使用者想擔任名為 *Developer* 的角色，並將該角色與她的名字 *Mary Major* 建立關聯，則可使用類似如下的 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
```

如需詳細資訊，請參閱AssumeRole。

您可以使用 AssumeRoleWithSAML 操作來傳回一組包含 displayName 和 emailAddress 標籤的暫時登入資料。當您在存取 CodeCommit 儲存庫時，就能使用這些標籤。您的公司或群組必須已將第三方 SAML 解決方案與 AWS 整合。若是如此，您便可以將 SAML 屬性做為工作階段標籤傳遞。例如，如果您想將名為 *Saanvi Sarkar* 的使用者之顯示名稱和電子郵件身分屬性做為工作階段標籤傳遞：

```
<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>
```

如需詳細資訊，請參閱使用 AssumeRoleWith SAML 傳遞工作階段標記。

您可以使用 AssumeRoleWithIdentity 操作來傳回一組包含 displayName 和 emailAddress 標籤的暫時登入資料。當您在存取 CodeCommit 儲存庫時，就能使用這些標籤。如要從 OpenID Connect (OIDC) 傳遞工作階段標籤，您必須在 JSON Web 權杖 (JWT) 中包含工作階段標籤。例如，用於呼叫AssumeRoleWithWebIdentity的解碼 JWP 權杖，其中包含名為 *Li* Juan 的使用者的displayName和emailAddress工作階段標籤：

```
{
    "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"
        ]
    }
```

```
}
```

如需詳細資訊，請參閱使用 AssumeRoleWithWebIdentity.

您可以使用 GetFederationToken 操作來傳回一組包含 displayName 和 emailAddress 標籤的暫時登入資料。當您在存取 CodeCommit 儲存庫時，就能使用這些標籤。例如，若要使用 AWS CLI 取得包含 displayName 和 emailAddress 標籤的聯合字符：

```
aws sts get-federation-token \
--name my-federated-user \
--tags key=displayName,value="Nikhil Jayashankar"
 key=emailAddress,value=nikhil_jayashankar@example.com
```

如需詳細資訊，請參閱使用 GetFederationToken.

CodeCommit IAM 角色

IAM 角色是您 Amazon Web Services 帳戶中具備特定許可的實體。

使用臨時登入資料 CodeCommit

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫AssumeRole或等 AWS STS API 作業來取得臨時安全登入資料GetFederationToken。

CodeCommit 支援使用臨時認證。如需詳細資訊，請參閱使用旋轉認證連線至AWS CodeCommit儲存庫。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

CodeCommit 不使用服務連結角色。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

CodeCommit 不使用服務角色。

AWS CodeCommit 身分型政策範例

根據預設，IAM 使用者和角色不具備建立或修改 CodeCommit 資源的許可。他們也無法使用 AWS Management Console、AWS CLI 或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 操作的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

如需政策的範例，請參閱下列主題：

- 範例 1：允許使用者在單一 CodeCommit 執行作業 AWS 區域
- 範例 2：允許使用者將 Git 用於單一儲存庫
- 範例 3：允許使用者從指定的 IP 位址範圍連線存取儲存庫
- 範例 4：拒絕或允許對分支執行動作
- 範例 5：拒絕或允許對具有標籤的儲存庫執行動作
- 使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 索引標籤上建立政策。

主題

- 政策最佳實務
- 使用控 CodeCommit 制台
- 允許使用者檢視他們自己的許可
- 根據標籤檢視 CodeCommit 儲存庫

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 CodeCommit 資源。這些動作可能會讓您的 AWS 帳戶 產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並朝向最低權限許可的目標邁進：如需開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的 AWS 受管政策。

- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務（例如 AWS CloudFormation）使用條件。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA)：如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 最佳安全實務。

使用控 CodeCommit 制台

若要存取 AWS CodeCommit 主控台，您必須擁有最低的一組許可。這些許可必須允許您列出並檢視 Amazon Web Services 帳戶中 CodeCommit 資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (IAM 使用者或角色) 而言，主控台就無法如預期運作。

若要確保這些實體仍可使用 CodeCommit 主控台，請同時將下列AWS受管理的原則附加至實體。如需更多資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

如需詳細資訊，請參閱使用以身分為基礎的政策 (IAM 政策) CodeCommit。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許其最基本主控台許可。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台上，或是使用 AWS CLI 或 AWS API 透過編寫程式的方式完成此動作的許可。

```
{
    "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": "*"
        }
    ]
}
```

根據標籤檢視 CodeCommit

您可以在身分型政策中使用條件，以根據標籤控制存取 CodeCommit 資源。如需示範作法的政策範例，請參閱範例 5：拒絕或允許對具有標籤的儲存庫執行動作。

如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。

對 AWS CodeCommit 身分與存取進行疑難排解

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 CodeCommit 常見問題。

主題

- 我沒有執行操作的授權 CodeCommit
- 我沒有授權執行 iam：PassRole
- 我想要檢視我的存取金鑰
- 我是系統管理員，想要允許其他人存取 CodeCommit
- 我想允許 Amazon Web Services 帳戶以外的人訪問我的 CodeCommit 資源

我沒有執行操作的授權 CodeCommit

若 AWS Management Console 告知您並未獲得執行動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是為您提供登入憑證的人員。

如需更多資訊，請參閱 使用 CodeCommit 主控台所需的許可

我沒有授權執行 iam：PassRole

如果您收到錯誤，告知您未獲授權執行 `iam:PassRole` 動作，您的政策必須更新，允許您將角色傳遞給 CodeCommit。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 `marymajor` 的 IAM 使用者嘗試使用主控台在 CodeCommit 中執行動作時，發生下列範例錯誤。但是，該動作要求服務具備服務角色授與的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我想要檢視我的存取金鑰

在您建立 IAM 使用者存取金鑰後，您可以隨時檢視您的存取金鑰 ID。但是，您無法再次檢視您的私密存取金鑰。若您遺失了密碼金鑰，您必須建立新的存取金鑰對。

存取金鑰包含兩個部分：存取金鑰 ID (例如 `AKIAIOSFODNN7EXAMPLE`) 和私密存取金鑰 (例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)。如同使用者名稱和密碼，您必須一起使用存

取金鑰 ID 和私密存取金鑰來驗證您的請求。就如對您的使用者名稱和密碼一樣，安全地管理您的存取金鑰。

> **Important**
>
> 請勿將您的存取金鑰提供給第三方，甚至是協助尋找您的標準使用者 ID。透過執行此操作，可能會讓他人永久存取您的 AWS 帳戶。

建立存取金鑰對時，您會收到提示，要求您將存取金鑰 ID 和私密存取金鑰儲存在安全位置。私密存取金鑰只會在您建立它的時候顯示一次。若您遺失了私密存取金鑰，您必須將新的存取金鑰新增到您的 IAM 使用者。您最多可以擁有兩個存取金鑰。若您已有兩個存取金鑰，您必須先刪除其中一個金鑰對，才能建立新的金鑰對。若要檢視說明，請參閱《IAM 使用者指南》中的管理存取金鑰。

我是系統管理員，想要允許其他人存取 CodeCommit

若要允許其他人存取 CodeCommit，您必須為需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。您接著必須將政策連接到實體，在 CodeCommit 中授予他們正確的許可。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

我想允許 Amazon Web Services 帳戶以外的人訪問我的 CodeCommit 資源

如需詳細資訊，請參閱 使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取。

AWS CodeCommit 中的恢復能力

AWS 全球基礎架構是以 AWS 區域 與可用區域為中心建置的。AWS 區域 提供多個分開且隔離的實際可用區域，並以具備低延遲、高輸送量和高度備援特性的聯網相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

一個 CodeCommit repository 或 CodeCommit 核准規則範本存在於AWS 區域它被創建的地方。如需詳細資訊，請參閱 區域和 Git 連線端點 AWS CodeCommit。對於儲存庫中的彈性，您可以將 Git 用戶端設定為一次推送至兩個儲存庫。如需詳細資訊，請參閱 將提交推送到額外的 Git 存儲庫。

如需 AWS 區域 與可用區域的詳細資訊，請參閱AWS全球基礎架構。

AWS CodeCommit 中的基礎設施安全

作為託管服務，AWS CodeCommit受AWS全局網絡安全過程，詳情請參閱Amazon Web Services：安全流程概觀白皮書。

您使用AWS發佈的 API 呼叫，透過網路存取 CodeCommit。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。建議使用 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

請求必須使用存取金鑰 ID 和與 IAM 委託人相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 產生臨時安全憑證來簽署請求。

您可從任何網路位置呼叫這些 API 操作，但 CodeCommit 確實根據來源 IP 地址來支援限制。您也可以使用 CodeCommit 政策來控制從特定 Amazon Virtual Private Cloud (Amazon VPC) 的端點或特定 VPC 的存取。實際上，這只會隔離網路中的特定 VPC 對特定 CodeCommit 資源的網路存取。AWS網路。

如需詳細資訊，請參閱下列內容：

- 範例 1：允許使用者在單一 CodeCommit 執行作業 AWS 區域
- 範例 3：允許使用者從指定的 IP 位址範圍連線存取儲存庫
- AWS CodeCommit 與介面 VPC 端點搭配使用

監控 AWS CodeCommit

監控是維護可靠性、可用性與效能的重要環節 CodeCommit 和你的另一個AWS解決方案。AWS提供下列監控工具監看 CodeCommit，在發現錯誤時回報，並適時自動採取動作：

- 亞馬遜 EventBridge 可用於自動化您的AWS服務，並自動回應系統事件 (例如應用程式可用性問題或資源變更)。的事件AWS服務交付給 EventBridge 在接近即時的情況下。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。如需詳細資訊，請參閱「」亞馬遜 EventBridge 使用者指南和監控 CodeCommit Amazon 事件 EventBridge 和 Amazon CloudWatch 活動。
- 亞馬遜 CloudWatch 事件串流，描述所發生的變更AWS的費用。 CloudWatch 事件驅動運算，因為您可以在其他事件與在其他事件與在其他事件中觸發自動化動作的規則AWS這些事件發生時的服務。如需詳細資訊，請參閱 。亞馬遜 CloudWatch 事件使用者指南和監控 CodeCommit Amazon 事件 EventBridge 和 Amazon CloudWatch 活動。
- 亞馬遜 CloudWatch 日誌可用來監控、存放及存取日誌檔案 CloudTrail 和其他來源的資料。CloudWatch 日誌可監控日誌檔案中的資訊，並在達到特定閾值時向您發出通知。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 。亞馬遜 CloudWatch 日誌使用者指南。
- AWS CloudTrail可擷取 Amazon Web Services vice 帳戶發出的 API 呼叫和相關事件，並傳送日誌檔案至您指定的 Amazon S3 儲存貯體。您可以找出哪些使用者和帳戶呼叫 AWS、發出呼叫的來源 IP 地址，以及呼叫的發生時間。如需詳細資訊，請參閱 。AWS CloudTrail使用者指南和使用 AWS CloudTrail 記錄 AWS CodeCommit API 呼叫。

監控 CodeCommit Amazon 事件 EventBridge 和 Amazon CloudWatch 活動

您可以監控AWS CodeCommit事件在 EventBridge，它可以從您自己的應用程序提供實時數據流，software-as-a-service (SaaS) 應用程式，以及AWS服務。 EventBridge將資料路由至目標，例如AWS LambdaAmazon Simple Notification Service。這些事件與 Amazon 上出現的事件相同 CloudWatch 事件串流，說明所發生的變更AWS的費用。

下列範例說明多種事件 CodeCommit。

> **Note**
>
> CodeCommit 支持提供`displayName`和`emailAddress`包含在事件中，工作階段標籤所包含的資訊 (若有提供該資訊)。如需詳細資訊，請參閱工作階段標籤和使用標籤提供身份信息 CodeCommit。

主題

- 反應已建立事件
- 反應更新事件

referenceCreated 事件

在此範例事件中，系統已在名為 MyDemoRepo 的儲存庫內建立名為 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

在此範例事件中，系統已透過在名為 MyDemoRepo 的儲存庫內進行合併來更新名為 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

在此範例事件中，系統已在名為 MyDemoRepo 的儲存庫內刪除名為 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"
  }
}
```

unreferencedMergeCommit已建立事件

在此範例事件中，系統已在名為 MyDemoRepo 的儲存庫內建立未參照的合併遞交。

```
{
  "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"
  }
}
```

commentOnCommit已建立事件

在此範例事件中，名為 Mary_Major 的聯合身分使用者會對遞交進行評論。在此範例中，她的聯合身分供應商已設定 displayName 和 emailAddress 的工作階段標籤，而該資訊會包含在事件中。

```
{
  "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"
  }
}
```

commentOnCommit更新事件

在此範例事件中，擔任 Admin 角色且角色工作階段名稱為 Mary_Major 的使用者編輯了對遞交的評論。在此範例中，該角色會隨附為 displayName 和 emailAddress 所設定的工作階段標籤，而該資訊會包含在事件中。

```
{
  "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

在此範例事件中，名為 Saanvi_Sarkar 的聯合身分使用者會對提取請求進行評論。在此範例中，她的聯合身分供應商已設定 displayName 和 emailAddress 的工作階段標籤，而該資訊會包含在事件中。

```
{
  "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

在此範例事件中，名為 Saanvi_Sarkar 的聯合身分使用者編輯了對提取請求的評論。在此範例中，她的聯合身分供應商已設定 displayName 和 emailAddress 的工作階段標籤，而該資訊會包含在事件中。

```
{
  "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

在此範例事件中，擔任 Admin 角色且角色工作階段名稱為 Mary_Major 的使用者已在名為 MyDemoRepo 的儲存庫內建立提取請求。使用者未提供任何工作階段標籤資訊，因此事件中不會包含該資訊。

```
{
  "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

在此範例事件中，擔任 Admin 角色且角色工作階段名稱為 Mary_Major 的使用者，已針對 ID 為 1 的提取請求更新名為 test-branch 的來源分支。

```
{
  "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"
  }
}
```

pullRequestStatus變更事件

在此範例事件中，擔任 Admin 角色且角色工作階段名稱為 Mary_Major 的使用者已關閉 ID 為 1 的提取請求。該提取請求並不會合併。

```
{
  "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

在此範例事件中，擔任 Admin 角色且工作階段名稱為 Mary_Major 的使用者已合併 ID 為 1 的提取請求。

```
{
  "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"
  }
}
```

approvalRuleTemplate已建立事件

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major已建立名為的核准規則範本2-approvers-required-for-main。

```
{
    "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": {}
    }
}
```

approvalRuleTemplate更新事件

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major編輯名為的核准規則範本2-approvers-required-for-main。該核准規則範本不會與任何儲存庫相關聯。

```
{
  "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": {}
  }
}
```

approvalRuleTemplate已刪除事件

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major已刪除名為的核准規則範本2-approvers-required-for-main。該核准規則範本不會與任何儲存庫相關聯。

```
{
  "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major關聯名為的核准規則範本2-approvers-required-for-main使用名為的存儲庫MyDemoRepo。

```
{
    "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major取消名為的核准規則範本2-approvers-required-for-main從名為的存儲庫MyDemoRepo。

```
{
    "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major批次關聯名為的核准規則範本2-approvers-required-for-main使用名為的存儲庫MyDemoRepo和一個名為的存儲庫MyTestRepo。

```
{
    "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major批次取消名為的核准規則範本2-approvers-required-for-main從名為的存儲庫MyDemoRepo和一個名為的存儲庫MyTestRepo。

```
{
    "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major已建立名為的核准規則1-approver-needed針對 ID 為的提取請求227。

```
{
    "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

在此範例事件中，使用者名稱為的使用者為的使用者為的使用者已為Mary_Major刪除名為的核准規則1-approver-needed針對 ID 為的提取請求227。具有名稱的 IAM 使用者Saanvi_Sarkar原本已撰寫核准規則。

```
{
    "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

在此範例事件中，使用者名稱為的使用者已將提取請求的核准規則需求的核准規則需求的核准規則需求的核准規則需求的核准規則需求Mary_Major。提取請求是由具有 IAM 使用者名稱為的使用者所撰寫提取請求Li_Juan。

```
{
    "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"
    }
}
```

在此範例事件中，提取請求的核准規則需求已恢復 (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

在此範例事件中，使用者名稱為的使用者已核准提取請求Mary_Major。

```
{
    "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"
    }
}
```

在此範例事件中，使用者名稱為的使用者已撤銷提取請求的核准提取請求的核准提取請求的核准Mary_Major。

```
{
    "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

在此範例事件中，使用者名稱為的使用者已編輯提取請求的核准規則Mary_Major。她也是撰寫提取請求的使用者。

```
{
    "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"
    }
}
```

反應已建立事件

在此範例事件中，使用者名稱為的使用者已加入對註解的使用者已新增為的使用者Mary_Major。

```
{
   "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"
   }
}
```

反應更新事件

在此範例事件中，使用者名稱為的使用者已更新為的使用者Mary_Major。使用者只能更新自己的反應。

```
{
   "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 CloudTrail 記錄 AWS CodeCommit API 呼叫

CodeCommit 與整合AWS CloudTrail，這項服務可提供由使用者、角色或中AWS服務所採取之動作的記錄 CodeCommit。 CloudTrail 擷取 CodeCommit 為事件的所有 API 呼叫，包括來自 CodeCommit 主控台的呼叫、Git 用戶端以及來自對 API 發出的程 CodeCommit 式碼呼叫。若您建立線索，便可將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括的事件 CodeCommit。即使您未設定線索，依然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新事件。使用由收集的資訊 CloudTrail，您就可以判斷傳送至的請求 CodeCommit、提出請求的 IP 地址、提出請求的對象、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

CodeCommit 中的資訊 CloudTrail

CloudTrail 在您建立帳戶時，系統即會在 Amazon Web Services 帳戶中啟用。當中發生活動時 CodeCommit，系統便會將該活動記錄至 CloudTrail 事件，並將其他AWS服務事件記錄到事件歷史記錄中。您可以檢視、搜尋和下載 Amazon Web Services 帳戶中的最近事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

如需 Amazon Web Services 帳戶中正在進行事件的記錄 (包含的事件) CodeCommit，請建立線索。線索能 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立權杖時，權杖會套用到所有區域。線索會記錄來自 AWS 分割區中所有區域的事件，然後將所有日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還能設定其他AWS服務，以進一步分析和處理 CloudTrail 日誌中收集的事件資料。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

在 Amazon Web Services 帳戶中啟用日 CloudTrail 誌檔案時，對 CodeCommit動作發出的 API 呼叫會在 CloudTrail 日誌檔案中追蹤，與其他AWS服務記錄編寫在一起。 CloudTrail 根據期間和檔案大小，決定何時建立和寫入新檔案。

所有 CodeCommit 操作都由記錄 CloudTrail，包括一些（例如`GetObjectIdentifier`），這些操作當前未記錄在 AWS CodeCommitAPI 參考中，但會被引用為訪問權限並在中記錄CodeCommit 許可參考。例如，呼叫`ListRepositories`(在AWS CLI、`aws codecommit list-repositories`)、()和`CreateRepositoryPutRepositoryTriggers`(`aws codecommit create-repositoryaws codecommit put-repository-triggers`) 動作會在 CloudTrail 記錄檔中產生項目，以及 Git 用戶端呼叫`GitPull`和`GitPush`。此外，如果您在中將 CodeCommit 存放庫配置為管道的來源 CodePipeline，則會看到對存 CodeCommit 取權限動作的呼叫，例如`UploadArchive`從 CodePipeline。由於 CodeCommit 使用 AWS Key Management Service 來加密和解密儲存庫，您也會在 CloudTrail 日誌中看到從 CodeCommit 對 `Encrypt` 和來自 AWS KMS 之 `Decrypt` 動作的呼叫。

每個日誌項目都會包含產生要求之人員的資訊。日誌記錄中的使用者身分資訊，可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者憑證提出
- 該請求是以角色或聯合身分使用者的臨時安全登入資料提出，還是由擔任的角色提出
- 該請求是否由另一項 AWS 服務提出

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

日誌檔可存放於 Amazon S3 儲存貯體任意長時間，但您也可以定義 Amazon S3 生命週期規則，自動封存或刪除日誌檔案。根據預設，您的日誌檔案預設會使用 Amazon S3 伺服器端加密 (SSE) 進行加密。

了解 CodeCommit 日誌檔案項目

CloudTrail 日誌檔案可包含一個或多個日誌項目。每一項目均列出多個 JSON 格式的事件。一個日誌事件為任何來源提出的單一請求，並包含請求動作、動作的日期和時間、請求參數等資訊。日誌項目並非公有 API 呼叫的有序堆疊追蹤，因此不會以任何特定順序顯示。

> **Note**
>
> 此範例已格式化，以提升可讀性。在 CloudTrail 日誌檔案中，所有項目和事件會合併為單一列。這個範例中受限於單一 CodeCommit 項目。在真實的 CloudTrail 日誌檔案中，您將看到來自多個AWS服務的項目和事件。

內容

範例：列出 CodeCommit 儲存庫的記錄項目

以下範例顯示的是展示`ListRepositories`動作的 CloudTrail 日誌項目。

> **Note**
>
> 雖然會`ListRepositories`傳回儲存庫清單，但是不可變的回應不會記 CloudTrail 錄在記錄檔中，因`responseElements`此會顯示`null`在記錄檔中。


```
{
  "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"
```

```
}
```

範例：建立 CodeCommit 存放庫的記錄項目

以下範例顯示的是展示CreateRepository動作的 CloudTrail 日誌項目。

```
{
  "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"
}
```

範例：對 CodeCommit 儲存庫執行 Git 提取呼叫的日誌項目

以下範例顯示的是展示本機存放庫所在之`GitPull`動作的 CloudTrail 日誌項目 up-to-date。

```
{
  "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"
}
```

下列範例顯示的 CloudTrail 記錄項目會示範本機存放庫不在的`GitPull`動作 up-to-date ，因此資料會從 CodeCommit 儲存庫傳輸至本機存放庫。

```
{
  "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"
}
```

範例：成功推送至 CodeCommit 儲存庫的記錄項目

以下範例顯示的是展示GitPush動作的 CloudTrail 日誌項目。GitPush 動作在日誌項目中顯示兩次表示推送成功。

```
{
  "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"
}
```

透過 AWS CloudFormation 建立 CodeCommit 資源

AWS CodeCommit 已與 AWS CloudFormation 整合，這項服務可協助您建立 AWS 資源的模型和設定，以減少建立和管理資源和基礎設施的時間。您可以建立一個範本，描述所有所需的AWS資源 (例如儲存庫)，就會為您AWS CloudFormation佈建和設定這些資源。

當您使用 AWS CloudFormation 時，您可以重複使用您的範本，重複、一致的設定您的 CodeCommit 資源。只需描述一次您的資源，即可在多個 AWS 帳戶 帳戶與區域內重複佈建相同資源。

CodeCommit 和 AWS CloudFormation 範本

若要佈建和配置 CodeCommit 與相關服務的資源，您必須了解 AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的什麼是 AWS CloudFormation Designer？。

CodeCommit支援在中建立儲存庫AWS CloudFormation與從主控台或命令列建立儲存庫不同，您可AWS CloudFormation以使用建立儲存庫，並從 Amazon S3 儲存貯體中指定的 .zip 檔案自動將程式碼提交至新建立的存放庫。如需詳細資訊，包括存放庫的 JSON 和 YAML 範本範例，請參閱AWS::CodeCommit::Repository。

使用建立CodeCommit儲存庫時AWS CloudFormation，只要在 AWS:: 存放庫程式碼中設定屬性，即可選擇在建立過程中將程式碼提交至該儲存庫。CodeCommit您可以指定存放程式碼的 Amazon S3 儲存貯體，並選擇性地使用該BranchName屬性來指定將在該程式碼初始提交時建立的預設分支名稱。這些屬性僅用於初始存儲庫創建，並且在堆棧更新時被忽略。您無法使用這些屬性對儲存庫進行其他認可，或在初始提交之後變更預設分支的名稱。

> **Note**
>
> 2021 年 1 月 19 日，將預設分支的名稱CodeCommit從主分支AWS變更為主分支。此名稱變更CodeCommit會影響使用主CodeCommit控台、CodeCommit API、AWS SDK 和AWS CLI. 自 2021 年 3 月 4 日起，AWS CDK使用AWS CloudFormation或具有初始代碼提交作為創建的一部分創建的存儲庫與此更改保持一致。此變更不會影響現有的儲存庫或分支。使用本機 Git 用戶端建立初始提交的客戶，其預設分支名稱會遵循這些 Git 用戶端的設定。如需詳細資訊，請參閱使用分支、建立提交和變更分支設定。

您也可以建立建立相關資源的範本，例如儲存庫的通知規則、AWS CodeBuild建置專案、AWS CodeDeploy應用程式和AWS CodePipeline管道。

Template examples

下列範例會建立名為的CodeCommit儲存庫*MyDemoRepo*。*############### Amazon S3 ###### #####MySourceCodeBucket#########################*

> **Note**
>
> Amazon S3 Web 服務帳戶中的名稱，其包含的 ZIP 檔案具有要遞交給新儲存器的內容。Amazon S3 開發人員指南中定義的 Amazon S3 物件金鑰。

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
```

如需更多範例，請參閱 AWS::CodeCommit::Repository。

AWS CloudFormationCodeCommit、和AWS Cloud Development Kit (AWS CDK)

使用其創建中的使AWS CDK用AWS CloudFormation功能創建的存儲庫。了解AWS CloudFormation範本如何運用CodeCommit資源，可協助您建立及管理AWS CDK程式碼。如需有關的詳細資訊AWS CDK，請參閱AWS Cloud Development Kit (AWS CDK)開發人員指南和 AWS CDKAPI 參考。

下面的打AWS CDK字稿示例創建一個名為CodeCommit存儲庫*MyDemoRepo*。*###############Amazon S3 ##########MySourceCodeBucket#########################*

```
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"
              }
            },
        }
     );
  }
}
```

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- 《AWS CloudFormation 使用者指南》https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
- 《AWS CloudFormation 命令列介面使用者指南》https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html

AWS CodeCommit 疑難排解

以下資訊可能有助於診斷 AWS CodeCommit 內的常見問題。

主題

對 Git 憑據和 HTTPS 連接進行故障排除AWS CodeCommit

下列資訊可幫助您對使用 Git 登入資料和 HTTPS 來連接到 AWS CodeCommit 儲存庫時的常見問題進行故障診斷。

主題

的 Git 登入資料AWS CodeCommit：當我在終端機或命令列連接到我的 CodeCommit 儲存庫時，我持續看見輸入登入登入資料的提示

問題：嘗試從終端機或命令列推送、提取或以其他方式與之互動時，系統提示您提供使用者名稱和密碼，而您必須為 IAM 用户提供 Git 登入資料。

可能的修正：此錯誤最常見的原因是本機電腦執行的作業系統不支援登入資料管理、它未安裝登入資料管理公用程式，或是使用者的 Git 登入資料尚未儲存到這些登入資料管理系統的其中一個。根據作業

系統和本機環境，您可能需要安裝登入資料管理工具、設定作業系統中包含的登入資料管理工具，或自訂本機環境以使用登入資料儲存體。例如，如果您的電腦執行的是 macOS，您可以使用 Keychain Access 公用程式來存放登入資料。如果您的電腦執行 Windows，您可以使用隨著適用於 Windows 的 Git 安裝的 Credential Manager。如需詳細資訊，請參閱 Git 文件中的適用於使用 Git 認證的 HTTPS 使用者和Credential Storage。

的 Git 登入資料AWS CodeCommit：我設定 Git 登入資料，但我的系統未使用它們

問題：嘗試使用搭配 Git 用户端時，用户端似乎未使用使用 CodeCommit 的 Git 登入資料。

可能的修正：此錯誤最常見的原因是，您先前將電腦設定為使用隨附的登入資料協助程式AWS CLI。請檢查您的 .gitconfig 檔案中類似以下的區段，並且將它們移除：

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

儲存檔案，然後開啟新的命令列或終端機工作階段，之後再次嘗試連接。

您也可能已在電腦上設定多個登入資料協助程式或管理程式，那麼您的系統可能會預設使用另一個組態。若要將使用的登入資料協助程式重設為預設值，執行 git config 命令時您可以使用 --system 選項，而不是 --global 或 --local。

如需詳細資訊，請參閱 Git 文件中的適用於使用 Git 認證的 HTTPS 使用者和Credential Storage。

針對 git-remote-codecommit 和 AWS CodeCommit 進行故障診斷

以下資訊可以協助您對與 AWS CodeCommit 儲存庫連線時可能會遇到的 git-remote-codecommit 相關問題進行故障診斷。

主題

- 我看到一個錯誤：git：'遠程代碼提交' 不是一個 git 命令
- 我看到一個錯誤：致命的：無法找到「代碼提交」的遠程幫助程序
- 複製錯誤：我無法從 IDE 複製 CodeCommit 儲存庫
- 推送或拉出錯誤：我無法從 IDE 將遞交推送或提取到 CodeCommit 儲存庫

我看到一個錯誤：git：'遠程代碼提交' 不是一個 git 命令

問題：當你嘗試使用 git 遠程代碼提交時，你會看到一個錯誤，即 git 遠程代碼提交不是 git 命令。請參閱 'git —help'」。

可能的修正：此錯誤最常見的原因的優點在於，您要麼沒有將 git-遠程/解碼提交可執行文件添加到 PATH，或者字符串包含語法錯誤。這可能會發生在 git 和遠程代碼提交之間缺少連字符的情況下，或者當一個額外的 git 放在 git-遠程代碼提交之前。

如需設定和使用 git-remote-codecommit 的詳細資訊，請參HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

我看到一個錯誤：致命的：無法找到「代碼提交」的遠程幫助程序

問題：當您嘗試使用 git-remote-codecommit 時，出現到指出「致命：找不到「代碼提交」的遠程幫助程序。

可能的修正：此錯誤最常見的原因為：

- git-remote-codecommit 的設定未完成
- 您已將 git-Remote 代碼提交安裝在路徑中不在或未配置為`Path`環境變數
- Python 不在你的路徑中，或者沒有配置為`Path`環境變數
- 您正在使用的終端或命令行窗口，該窗口自完成 git-遠程代碼提交安裝以來尚未重新啟動

如需設定和使用 git-remote-codecommit 的詳細資訊，請參HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

複製錯誤：我無法從 IDE 複製 CodeCommit 儲存庫

問題：當您嘗試在 IDE 中複製 CodeCommit 儲存庫時，出現到指出端點或 URL 無效的錯誤。

可能的修正：並非所有 IDE 都支援git-remote-codecommit複製期間。透過終端機或命令列從本機複製儲存庫，然後將該本機儲存庫新增到 IDE。如需詳細資訊，請參閱 步驟 3：連接到CodeCommit控制台並克隆存儲庫。

推送或拉出錯誤：我無法從 IDE 將遞交推送或提取到 CodeCommit 儲存庫

問題：當您嘗試從 IDE 提取或推送程式碼時，出現連線錯誤。

可能的修正：此錯誤最常見的原因是，IDE 與等 Git 遠端協助程式不相容git-remote-codecommit。透過命令列或終端機使用 Git 命令手動更新本機儲存庫，而不要使用 IDE 功能來遞交、推送和提取程式碼。

如需遠端協助程式和 Git 的詳細資訊，請參閱 Git 文件。

疑難排解 SSH 連線至AWS CodeCommit

以下資訊可以協助您對使用 SSH 來連接 CodeCommit 儲存庫時的常見問題進行故障診斷。

主題

存取錯誤：公開金鑰已成功上傳至 IAM，但在 Linux、macOS 或 Unix 系統上連線失敗

問題：當您嘗試連接到 SSH 端點以與CodeCommit存儲庫，無論是在測試連接或克隆存儲庫時，連接都會失敗或被拒絕。

可能的修正：指派給 IAM 中公開金鑰的 SSH 金鑰 ID 可能與您的連線嘗試不相關聯。您可能尚未配置配置文件，您可能無法存取組態檔案，另一個設定可能會阻止成功讀取設定檔、您可能提供了錯誤的金鑰 ID，或者您可能提供 IAM 使用者的 ID 而非金鑰 ID。

您可以在 IAM 使用者的設定檔中的 IAM 主控台中找到 SSH 金鑰 ID：

> **Note**
>
> 如果您已上傳一個以上的 SSH 金鑰 ID，則會依金鑰 ID 的字母順序列出金鑰，而非依上傳日期。請確定您已複製的金鑰 ID 與正確的上傳日期相關聯。

嘗試使用以下命令測試連線：

```
ssh Your-SSH-Key-ID@git-codecommit.us-east-2.amazonaws.com
```

確認連線後如果您看到成功訊息，則您的 SSH 金鑰 ID 有效。編輯您的設定檔，將您的連線嘗試與 IAM 中的公開金鑰建立關聯。如果您不想要編輯組態檔，您可以嘗試使用 SSH 金鑰 ID，將所有連線嘗試加到您的儲存庫。例如，如果您想克隆一個名為的存儲庫*MyDemoRepo*如果不修改配置文件來關聯您的連接嘗試，則可以運行以下命令：

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

如需詳細資訊，請參閱Linux Linux。

存取錯誤：公開金鑰已成功上傳至 IAM，並成功測試 SSH，但 Windows 系統上的連線失敗

問題：當您嘗試使用 SSH 端點來克隆或與CodeCommit儲存庫時，會出現包含該片語的錯誤訊息`No supported authentication methods available`。

可能的修正：此錯誤最常見的原因是，當您嘗試使用 SSH 時，您的 Windows 系統環境變數集將 Windows 導向使用另一個程式。例如，您可能已設定 GIT_SSH 變數以指向其中一個 PuTTY 工具組 (plink.exe)。這可能是傳統的組態，也可能需要在您的電腦上安裝一或多個其他程式。如果您確定不需要此環境變數，您可以開啟系統屬性將其移除。

若要解決這個問題，請開啟 Bash 模擬器，然後重試 SSH 連線，但要包含 `GIT_SSH_COMMAND="SSH"` 做為字首。例如，若要使用 SSH 來複製儲存庫：

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

在 Windows 命令列透過 SSH 連接時，如果您的 Windows 版本規定連接字串必須包含 SSH 金鑰 ID，就可能發生類似的問題。請再次嘗試連線，這次包括從 IAM 複製的 SSH 金鑰 ID 做為命令的一部分。例如：

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

身分驗證挑戰：連接到 CodeCommit 儲存庫時無法建立主機的真偽

問題：當您嘗試使用 SSH 端點與CodeCommit儲存庫時，會出現包含該片語的警告訊息The `authenticity of host 'host-name' can't be established.`

可能的修正：您的登入資料可能未正確設定。請遵循Linux Linux或適用於 Windows 上的 SSH 連線中的指示。

如果您按照這些步驟操作並且問題仍然存在，則可能有人嘗試man-in-the-middle攻擊。當您看到下列訊息時，請輸入 no 並按 Enter 鍵。

```
Are you sure you want to continue connecting (yes/no)?
```

請確定 CodeCommit 的指紋和公有金鑰與 SSH 設定主題中所述相符，再繼續進行連線。

的公共指紋CodeCommit

伺服器	密碼雜湊類型	指紋
git 代碼提交. 美東 2. 亞馬遜	MD5	a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e
git 代碼提交. 美東 2. 亞馬遜	SHA256	3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ
git 代碼提交. 我東-1. 亞馬遜	MD5	a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84

伺服器	密碼雜湊類型	指紋
git 代碼提交. 我東-1. 亞馬遜	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git 代碼提交. 我們西部-亞馬遜	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git 代碼提交. 我們西部-亞馬遜	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git 代碼提交. 歐盟西部-亞馬遜	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git 代碼提交. 歐盟西部-亞馬遜	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git 代碼提交. AP-東北-1. 亞馬遜	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git 代碼提交. AP-東北-1. 亞馬遜	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git 代碼提交. AP-東南部-亞馬遜	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git 代碼提交. AP-東南部-亞馬遜	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno

伺服器	密碼雜湊類型	指紋
git 代碼提交. AP-東南部-亞馬遜	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0
git 代碼提交. AP-東南部-亞馬遜	SHA256	nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec
git 代碼提交. AP-東南部-3. 亞馬遜	MD5	64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6
git 代碼提交. AP-東南部-3. 亞馬遜	SHA256	ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g
git 代碼提交. 我中心 1. 亞馬遜	MD5	bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76
git 代碼提交. 我中心 1. 亞馬遜	SHA256	grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA
git 代碼提交. 歐盟中央-1. 亞馬遜	MD5	74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d
git 代碼提交. 歐盟中央-1. 亞馬遜	SHA256	MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY
git 代碼提交. AP-東北-2. 亞馬遜	MD5	9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed

伺服器	密碼雜湊類型	指紋
git 代碼提交. AP-東北-2. 亞馬遜	SHA256	eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko
git 代碼提交. SA-東-1. 亞馬遜	MD5	74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51
git 代碼提交. SA-東-1. 亞馬遜	SHA256	kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU
git 代碼提交. 我們西部-亞馬遜	MD5	3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1
git 代碼提交. 我們西部-亞馬遜	SHA256	gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg
git 代碼提交. 歐盟西部-亞馬遜	MD5	a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3
git 代碼提交. 歐盟西部-亞馬遜	SHA256	r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs
git 代碼提交. AP-南 1. 亞馬遜	MD5	da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21
git 代碼提交. AP-南 1. 亞馬遜	SHA256	hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE

伺服器	密碼雜湊類型	指紋
git 代碼提交. AP-南 2. 亞馬遜	MD5	bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce
git 代碼提交. AP-南 2. 亞馬遜	SHA256	Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig
git 代碼提交. CA-中央-1. 亞馬遜	MD5	9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23
git 代碼提交. CA-中央-1. 亞馬遜	SHA256	Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w
git 代碼提交. 歐盟西部-3. 亞馬遜	MD5	1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76
git 代碼提交. 歐盟西部-3. 亞馬遜	SHA256	uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI
git 代碼提交。us-gov-west-1. 亞馬遜	MD5	9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69
git 代碼提交。us-gov-west-1. 亞馬遜	SHA256	djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4
git 代碼提交。us-gov-east-1. 亞馬遜	MD5	00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2

伺服器	密碼雜湊類型	指紋
git 代碼提交。us-gov-east-1. 亞馬遜	SHA256	fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98
地址代碼提交. 歐盟北 1. 亞馬遜	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
地址代碼提交. 歐盟北 1. 亞馬遜	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git 代碼提交. 我-南	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git 代碼提交. 我-南	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git 代碼提交. AP-東-1. 亞馬遜	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git 代碼提交. AP-東-1. 亞馬遜	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git 代碼提交. CN-北 1. 亞馬遜	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git 代碼提交. CN-北 1. 亞馬遜	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8

伺服器	密碼雜湊類型	指紋
git 代碼提交. CN-西北部-1. 亞馬遜	MD5	`2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6`
git 代碼提交. CN-西北部-1. 亞馬遜	SHA256	`wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ`
GIT 代碼提交歐盟南部 1. 亞馬遜	MD5	`b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11`
GIT 代碼提交歐盟南部 1. 亞馬遜	SHA256	`lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI`
git 代碼提交. AP-東北-3. 亞馬遜	MD5	`25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29`
git 代碼提交. AP-東北-3. 亞馬遜	SHA256	`2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA`
GIT 代碼提交. AF-南-亞馬遜	MD5	`21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29`
GIT 代碼提交. AF-南-亞馬遜	SHA256	`C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM`
git 代碼提交. 中央-1. 亞馬遜	MD5	`04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9`

伺服器	密碼雜湊類型	指紋
git 代碼提交. 中央-1. 亞馬遜	SHA256	uFxhp51kUWhleTLeYb xQVYm4RnNLNZ5Dbdm1 cgdSl/8

IAM 錯誤：嘗試將公鑰添加到 IAM 時出現「格式無效」

問題：在 IAM 中，當嘗試設置使用 SSH 時CodeCommit時，會出現包含該片語的錯誤訊息`Invalid format`當您嘗試添加您的公鑰時。

可能的修正：IAM 要求公開金鑰必須以 ssh-rsa 格式或 PEM 格式進行編碼。它僅接受 OpenSSH 格式的公鑰，並且具有中規定的其他要求搭配使用 SSH 金鑰CodeCommit在IAM 使用者指南。如果您提供另一種格式的公有金鑰，或金鑰未包含必要的位元數，您會看到此錯誤。

- 當您複製 SSH 公開金鑰時，您的作業系統可能已經引入換行符號。請確定您新增至 IAM 的公開金鑰中沒有換行符號。
- 某些 Windows 作業系統不會使用 OpenSSH 格式。若要產生金鑰配對並複製 IAM 所需的 OpenSSH 格式，請參閱the section called “步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰”。

如需 IAM 中 SSH 金鑰需求的詳細資訊，請參閱搭配使用 SSH 金鑰CodeCommit在IAM 使用者指南。

我需要訪問CodeCommit使用 SSH 登入資料存放在多個亞馬遜網路服務帳戶

問題：我想設置 SSH 訪問CodeCommit儲存庫位於多個亞馬遜網路服務帳戶中。

可能的修正：您可以為每個 Amazon 網路服務帳戶建立唯一的 SSH 公開/私密金鑰配對，並使用每個金鑰設定 IAM。然後，您可以使用與公鑰關聯的每個 IAM 用戶 ID 的相關信息配置一 /.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.
```

在這種配置中，您將能夠用「代碼提交 2」替換「Git 代碼提交。例如，若要在第二個 Amazon 網路服務帳戶中複製儲存庫：

```
git clone ssh://codecommit-2/v1/repos/YourRepositoryName
```

要為您的存儲庫設置遠程，請運行git remote add。例如：

```
git remote add origin ssh://codecommit-2/v1/repos/YourRepositoryName
```

如需更多範例，請參閱這個論壇帖子和這個貢獻GitHub。

Windows 上的 Git：嘗試使用 SSH 連接時，Bash 模擬器或命令列凍結

問題：在您設定適用於 Windows 的 SSH 存取，並於命令列或終端機確認連線之後，您看到訊息指出登錄中沒有快取伺服器的主機金鑰，而當您在命令提示字元或 Bash 模擬器中嘗試使用 git pull、git push 或 git clone 等命令時，嘗試將金鑰存放在快取中會遭到凍結 (不接受 y/n/return 鍵輸入)。

可能的修正：此錯誤最常見的原因是 Git 環境設定為使用 OpenSSH 以外的方法進行身分驗證 (可能是 PuTTY)。已知這在某些組態中快取金鑰時會造成問題。若要修正此問題，請嘗試下列其中一項：

- 開啟 Bash 模擬器，並在 Git 命令之前新增 `GIT_SSH_COMMAND="ssh"` 參數。例如，如果您嘗試推送至儲存庫，不要輸入 git push，請輸入：

  ```
  GIT_SSH_COMMAND="ssh" git push
  ```

- 如果您安裝了膩子，請打開膩子，然後在主機名稱 (或 IP 位址)，輸入CodeCommit您想要達到的端點 (例如，git 代碼提交. 選擇 Open (開啟)。當出現 PuTTY 安全提醒的提示時，請選擇 Yes (是)，以永久快取金鑰。
- 重新命名或刪除 `GIT_SSH` 環境變數 (如果您已不再使用它)。然後開啟新的命令提示字元或 Bash 模擬器工作階段，並再次嘗試您的命令。

關於其他解決方案，請參閱 Stack Overflow 上Git 複製/提取在快取的存放區金鑰中持續凍結。

公鑰格式需要 Linux 的某些發行版中的規範

問題：當您嘗試設定公用-私密金鑰組時，您會收到錯誤訊息。

可能的修正：某些 Linux 發行版需要在`~/.ssh/config`指定接受的公開金鑰類型的檔案。如需詳細資訊，請參閱您發行版本的相關文件`PubkeyAcceptedKeyTypes`。

訪問錯誤：連接到一個 SSH 公鑰被拒絕CodeCommit儲存庫

問題：當您嘗試使用 SSH 端點與CodeCommit儲存庫時，會出現包含該片語的錯誤訊息`Error: public key denied`。

可能的修正：此錯誤最常見的原因是您尚未完成 SSH 連接的設定。設定公開和私密安全殼層金鑰配對，然後將公開金鑰與 IAM 使用者建立關聯。如需設定 SSH 的詳細資訊，請參閱Linux Linux和適用於 Windows 上的 SSH 連線。

對憑據幫助程序和 HTTPS 連接進行故障排除AWS CodeCommit

下列資訊可幫助您對使用隨附的登入資料協助程式來解決常見問題進行故障診斷AWS CLI和 HTTPS 連接到 CodeCommit 存儲庫。

> **Note**
>
> 雖然登入資料協助程式是可支援使用聯合存取、身分供應商或臨時性登入資料連線到 CodeCommit 的方法，但建議採用的方法是安裝並使用git-remote-codecommit公用程式。如需詳細資訊，請參閱 HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit。

主題

- 執行git config命令配置憑據幫助程序
- 我在 Windows 中使用登入資料協助程式時發生找不到命令的錯誤
- 當我連接到 CodeCommit 儲存庫時提示我輸入使用者名稱
- 適用於 macOS 的 Git：我成功設定登入資料協助程式，但現在拒絕我存取儲存庫 (403)
- 適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)

執行git config命令配置憑據幫助程序

問題：當您嘗試運行 git config 命令來配置憑據助手以與 CodeCommit 存儲庫通信時，您會看到一個錯誤，即參數太少，或者提示 Git 配置命令和語法的使用提示符。

可能的修正：出現此錯誤的最常見原因是 Windows 操作系統上的命令使用單引號，或者在 Linux、macOS 或 Unix 操作系統中對命令使用雙引號。正確的語法如下：

- Windows：`git config --global credential.helper "!aws codecommit credential-helper $@"`
- Linux、macOS 或 Unix：`git config --global credential.helper '!aws codecommit credential-helper $@'`

我在 Windows 中使用登入資料協助程式時發生找不到命令的錯誤

問題：更新AWSCLI，登入資料協助程式到 CodeCommit 儲存庫的連線失敗，並出現`aws codecommit credential-helper $@ get: aws: command not found`。

原因：此錯誤最常見的原因是您的AWSCLI 版本已更新到使用 Python 3 的版本。這是 MSI 套件的已知問題。若要驗證您是否具有受影響的版本，請開啟命令列並執行下列命令：`aws --version`

如果輸出 Python 版本開頭為 3，表示您具有受影響的版本。例如：

```
aws-cli/1.16.62 Python/3.6.2 Darwin/16.7.0 botocore/1.12.52
```

可能的修正：您可執行下列其中一項來解決此問題：

- 在 Windows 使用 Python 和 pip 安裝和設定 AWS CLI，而不是使用 MSI。如需詳細資訊，請參閱在 Windows 上安裝 Python、pip 和 AWS CLI。
- 手動編輯 `.gitconfig` 檔案，變更 `[credential]` 區段來明確指向本機電腦上的 `aws.cmd`。例如：

  ```
  [credential]
      helper = !"\C:\\Program Files\\Amazon\\AWSCLI\\bin\\aws.cmd\" codecommit
   credential-helper $@
      UseHttpPath = true
  ```

- 執行 git config 命令來更新 `.gitconfig` 檔案以明確參考 `aws.cmd`，並視需要手動更新 PATH 環境變數來包含命令的路徑。例如：

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

當我連接到 CodeCommit 儲存庫時提示我輸入使用者名稱

問題：嘗試使用登入資料協助程式來與 CodeCommit 儲存庫通訊時，出現訊息，提示您輸入使用者名稱。

可能的修正：設定您的AWS設定資料，或確保您使用的是您設定使用 CodeCommit 的設定。如需設定的詳細資訊，請參閱HTTPS 連線的設定步驟AWS CodeCommitLinux、MacOS 或 Unix 上的儲存庫AWS CLI憑證助手或HTTPS 連線的設定步驟AWS CodeCommit使用視窗上的儲存庫AWS CLI憑證助手。如需 IAM、存取金鑰和私有金鑰的詳細資訊，請參見管理 IAM 使用者的存取金鑰和如何獲取登入資料？

適用於 macOS 的 Git：我成功設定登入資料協助程式，但現在拒絕我存取儲存庫 (403)

問題：在 macOS 上，登入資料協助程式似乎未如預期存取或使用您的登入資料。這可能是因為兩個不同的問題所導致：

- 所以此AWS CLI配置為AWS 區域與儲存庫所在位置不同。
- Keychain Access 公用程式所儲存的登入資料那時已過期。

可能的修正：若要驗證AWS CLI配置為正確的區域，請運行aws configure命令，然後查看顯示的信息。如果 CodeCommit 存儲庫位於AWS 區域不同於顯示的AWS CLI，您必須執行aws configure命令並將值更改為適合該區域的值。如需詳細資訊，請參閱 步驟 1：初始配置CodeCommit。

在 OS X 和 macOS 上發佈的預設 Git 版本使用 Keychain Access 公用程式來保存生成的登入資料。基於安全理由，為存取 CodeCommit 儲存庫而產生的密碼是臨時性，因此，存放在金鑰鏈中的登入資料大約 15 分鐘之後會失效。如果您僅使用 CodeCommit 來存取 Git，請嘗試下列：

1. 在終端機，執行 git config 命令來尋找 Keychain Access 公用程式定義所在的 Git 組態檔案 (`gitconfig`)。根據您的本機系統和偏好設定，您可能會有多個 `gitconfig` 檔案。

   ```
   git config -l --show-origin | grep credential
   ```

在此命令的輸出中，搜索結果類似於：

```
file:/path/to/gitconfig  credential.helper=osxkeychain
```

在此行開始處所列的檔案為您必須編輯的 Git 組態檔案。

2. 若要編輯 Git 設定檔，請使用純文字編輯器或執行下列命令：

```
nano /usr/local/git/etc/gitconfig
```

3. 使用下列其中一種策略來修改設定：

- 註釋掉或刪除包含`helper = osxkeychain`。例如：

```
# helper = osxkeychain
```

- 同時更新`aws credential helper`和`osxkeychain`憑據幫助器部分以具有上下文。例如，如果`osxkeychain`用於對 GitHub 進行身份驗證：

```
[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，Git 將使用`osxkeychain`幫助程序，當遠程主機匹配」`https://github.com`」以及遠程主機匹配時的憑據幫助程序」`https://git-codecommit\.us-east-1\.amazonaws.com`「。

- 在憑據幫助程序之前包含一個空字符串幫助程序。例如：

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

或者，如果您要繼續使用 Keychain Access 公用程式來快取其他 Git 儲存庫的登入資料，請修改標頭而非將它變更為註解。例如，若要允許 GitHub 快取的登入資料，您可以修改標頭，如下所示：

```
[credential "https://github.com"]
   helper = osxkeychain
```

如果您使用 Git 來存取其他儲存庫，您可以設定 Keychain Access 公用程式，使得它不會為您的 CodeCommit 儲存庫提供登入資料。若要設定 Keychain Access 公用程式：

1. 開啟 Keychain Access 公用程式。(您可以使用 Finder 找到它。)
2. 搜尋`git-codecommit.us-east-2.amazonaws.com`並替換`us-east-2`與AWS 區域存儲庫存在的位置。反白此列，開啟操作功能表 (按一下滑鼠右鍵)，然後選擇 Get Info (取得資訊)。
3. 選擇 Access Control (存取控制) 標籤。
4. 在 Confirm before allowing access (允許存取之前確認) 中，選擇 `git-credential-osxkeychain`，然後選擇減號從清單移除它。

> **Note**
>
> 從清單移除 `git-credential-osxkeychain` 之後，每當您執行 Git 命令時就會看到對話方塊。選擇 Deny (拒絕) 以繼續。如果您覺得快顯的干擾性太大，以下是一些替代選項：
>
> - 使用 SSH Connect 到 CodeCommit或 Git 登入資料，而不是使用的登入資料協助程式 HTTPS。如需更多詳細資訊，請參閱 Linux 及 使用 Git 認證的 HTTPS 使用者進行設定。
> - 在 Keychain Access 公用程式中，在存取控制標籤`git-codecommit.us-east-2.amazonaws.com`中，選擇允許所有應用程序訪問此項目（對此項目的訪問不受限制）選項。這可防止快顯，但登入資料最終仍會過期 (平均大約 15 分鐘)，然後您會看到 403 錯誤訊息。發生這種情況時，您必須刪除金鑰鏈項目才能還原功能。
> - 安裝一個依預設不使用金鑰鏈的 Git 版本。
> - 考慮用於刪除金鑰鏈項目的一個指令碼解決方案。要查看社區生成的腳本解決方案示例，請參閱可定期刪除 OS X 憑證存放區中快取登入資料的 Mac OS X 的指令碼在產品和服務整合。

如果想要讓 Git 完全無法使用 Keychain Access 公用程式，您可以設定 Git 以停止使用 osxkeychain 做為登入資料協助程式。例如，如果您開啟終端機並執行命令 `git config --system`

credential.helper，而它傳回 osxkeychain，就表示 Git 設定為使用 Keychain Access 公用程式。您可以執行下列命令來加以變更：

```
git config --system --unset credential.helper
```

請注意，通過使用--system選項會改變 Git 在系統範圍內對所有使用者的行為，如果您除了 CodeCommit 之外還使用其他儲存庫服務，這可能對其他使用者或其他儲存庫產生意外的影響。另外，此方法可能需要使用 sudo，而且您的帳戶可能沒有可套用此變更的足夠系統許可。請務必再次執行 git config --system credential.helper 命令來驗證已成功套用命令。如需詳細資訊，請參閱自訂 Git - Git 組態和 Stack Overflow 上的此文章。

適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)

問題：在 Windows 上，登入資料協助程式似乎未如預期存取或使用您的登入資料。這可能是因為不同的問題所導致：

- 所以此AWS CLI配置為AWS 區域與儲存庫所在位置不同。
- 在預設情況下，適用於 Windows 的 Git 會安裝 GCodeCommit 入資料管理工具，它與使用AWS登入資料協助程式。安裝時，它會導致與儲存庫的連線失敗，即使登入資料協助程式已隨AWS CLI並配置為 CodeCommit 的連接。
- 某些版本的適用於 Windows 的 Git 登入資料可能無法完全符合 RFC 2617 和 RFC 4559，這可能會導致 Git 登入資料和 AWS CLI 隨附的登入資料協助程式的問題。如需詳細資訊，請參閱版本 2.11.0(3) 未請求輸入使用者名稱/密碼。

可能的修正：

- 如果您嘗試使用 AWS CLI 隨附的登入資料協助程式，請考慮使用 Git 登入資料透過 HTTPS 連接，而不是使用登入資料協助程式。與適用於 Windows 的 Git 登入資料管理工具不同，與AWS CodeCommit。如需詳細資訊，請參閱 適用於使用 Git 認證的 HTTPS 使用者。

  如果想要使用登入資料協助程式來驗證AWS CLI配置為正確的AWS 區域，執行aws configure命令，然後查看顯示的信息。如果 CodeCommit 存儲庫位於AWS 區域不同於顯示的AWS CLI，您必須執行aws configure命令並將值更改為適合該區域的值。如需詳細資訊，請參閱 步驟 1：初始配置 CodeCommit。
- 如果可能，請解除安裝並重新安裝適用於 Windows 的 Git。安裝適用於 Windows 的 Git 時，請清除在安裝 Git Credential Manager 公用程式時所用之選項的核取方塊。此 Credential Manager 與 AWS

CodeCommit 的登入資料協助程式不相容。如果您已安裝 Git Credential Manager 或另一個登入資料管理公用程式，而且您不想將它解除安裝，您可以修改.gitconfig文件，併為 CodeCommit 添加憑據管理：

1. 開啟控制面板，選擇登入資料管理工具，並刪除 CodeCommit 的任何儲存的登入資料。
2. 在任何純文字編輯器 (例如記事本) 中開啟 .gitconfig 檔案。

> **Note**
>
> 如果您使用多個 Git 設定檔，您可能同時有本機和全域 .gitconfig 檔案。請務必編輯適當的檔案。

3. 將以下區段新增到您的 .gitconfig 檔案：

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

4. 儲存檔案，然後開啟新的命令列工作階段，之後再次嘗試連接。

如果您想要使用登入資料協助程式來AWS CodeCommit連接到 CodeCommit 儲存庫和另一個登入資料管理系統時，您可以在連接到其他託管儲存庫 (例如 GitHub 儲存庫) 時。

若要將使用的登入資料協助程式重設為預設值，您可以在執行 git config 命令時使用 --system 選項，而不是 --global 或 --local。

- 如果是在 Windows 電腦上使用 Git 登入資料，您可以嘗試解決任何 RFC 不相符的問題，方法是在連線字串中包括您的 Git 登入資料使用者名稱。例如，若要解決問題，並複製名為*MyDemoRepo*在美國東部 (俄亥俄) 區域：

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

> **Note**
>
> 如果在您的 Git 登入資料使用者名稱中有一個 @ 字元，則這種方法將沒有作用。您必須對該字元使用 URL 編碼 (也稱為 URL 逸出或 % 編碼)。

對 Git 用户端和進行故AWS CodeCommit

以下資訊可以協助您對使用 Git 搭配 AWS CodeCommit 儲存庫時的常見問題進行故障診斷。如需對使用 HTTPS 或 SSH 時 Git 用戶端的相關問題進行故障診斷，也請參閱 對 Git 登入資料進行故障診斷、疑難排解 SSH 連線和 對登入資料協助程式 (HTTPS) 進行故障診斷。

主題

Git 錯誤：錯誤：RPC 失敗；結果 =56，HTTP 代碼 = 200 嚴重：遠程端意外掛機

問題：推送大型變更、大量變更，或大型儲存庫時，長時間執行的 HTTPS 連線經常會因為網路連線問題或防火牆設定提前終止。

可能的修正：改為使用 SSH 推送，或是在您遷移大型儲存庫時遵循以增量方式移轉儲存庫。此外，請確定您未超過個別檔案的大小限制。如需詳細資訊，請參閱 配額。

Git 錯誤：太多個參考更新命令

問題：每個推送的參考更新數量上限為 4,000 個。當推送包含超過 4,000 個參考更新時會顯示此錯誤。

可能的修正：嘗試使用`git push --all`和`git push --tags`。如果您有太多個標籤，請將標籤分割成多個推送。如需詳細資訊，請參閱 配額。

Git 錯誤：Git 通過 HTTPS 推送在某些版本的 Git 中會損壞

問題：Curl 更新至 7.41.0 的問題導致基於 SSI 的摘要身份驗證失敗。已知受影響的 Git 版本包含 1.9.5.msysgit.1。某些版本的適用於 Windows 的 Git 登入資料可能無法完全符合 RFC 2617 和 RFC 4559，這可能會導致使用 Git 登入資料或 AWS CLI 隨附的登入資料協助程式的 HTTPS 連線發生問題。

可能的修正：檢查您的 Git 版本是否存取已知問題或使用早期或更新版本的版本。如需 mysysgit 的更多資訊，請參閱 GitHub 論壇中的對 HTTPS 的推送已損壞。如需適用於 Windows 的 Git 版本問題的更多資訊，請參閱 版本 2.11.0(3) 未要求輸入使用者名稱/密碼。

Git 錯誤：'gnutls_handshake () 失敗'

問題：在 Linux 中，當您嘗試使用 Git 來與 CodeCommit 儲存庫通訊時，出現含有語句的錯誤訊息。`error: gnutls_handshake() failed`。

可能的修正：針對 OpenSSL 編譯 Git。針對一個方法，請參閱 Ask Ubuntu 論壇中的連接到 HTTPS 伺服器時出現「錯誤：gnutls_handshake () 失敗」。

或者，請使用 SSH 而不是 HTTPS 來與 CodeCommit 儲存庫通訊。

Git 錯誤：Git 找不到 CodeCommit 儲存庫或沒有可存取儲存庫的許可

問題：連線字串中的結尾斜線可能導致連線嘗試失敗。

可能的修正：確定您已為儲存庫提供正確的名稱和連線字串，並且沒有結尾斜線。如需詳細資訊，請參閱 連接到儲存庫。

窗口上的 Git：沒有支援的身份驗證方法可供使用（公鑰）

問題：在您為 Windows 設定 SSH 存取之後，嘗試使用命令 (例如git pull、git push, 或git clone。

可能的修正：此錯誤的最常見原因是您的電腦上存在 GIT_SSH 環境變數，並且已設定可支援其他連線公用程式，例如 PuTTY。若要修正此問題，請嘗試下列其中一項：

- 開啟 Bash 模擬器，並在 Git 命令之前新增 `GIT_SSH_COMMAND="ssh"` 參數。例如，如果您嘗試複製儲存庫，不要輸入 git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo my-demo-repo，請輸入：

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

- 重新命名或刪除 `GIT_SSH` 環境變數 (如果您已不再使用它)。然後開啟新的命令提示字元或 Bash 模擬器工作階段，並再次嘗試您的命令。

如需有關在 Windows 上使用 SSH 時對 Git 問題進行故障診斷的詳細資訊，請參閱 疑難排解 SSH 連線。

故障診斷存取錯誤和AWS CodeCommit

以下資訊可以協助您對與 AWS CodeCommit 儲存庫連線時可能會看到的存取錯誤進行故障診斷。

主題

存取錯誤：當我從 Windows 連接到 CodeCommit 儲存庫時提示我輸入使用者名稱和密碼

問題：當您嘗試使用 Git 與 CodeCommit 儲存庫通訊，您看到對話方塊提示您輸入使用者名稱和密碼。

可能的修正：這可能是 Windows 的內建登入資料管理系統。根據組態的不同，執行以下其中一項：

- 如果使用 HTTPS 搭配 Git 登入資料，系統不會將 Git 登入資料存放在系統中。請提供 Git 登入資料然後繼續。系統應該不會再次提示您輸入。如需詳細資訊，請參閱 適用於使用 Git 認證的 HTTPS 使用者。
- 如果對 AWS CodeCommit 使用 HTTPS 搭配登入資料協助程式，它與 Windows 登入資料管理系統不相容。選擇 Cancel (取消)。

  這也可能表示您在安裝適用於 Windows 的 Git 時也安裝 Git Credential Manager。Git Manager 與包含的 CodeCommit 的登入資料協助程式不相容AWS CLI。請考慮解除安裝 Git Credential Manager。您也可以安裝並配置git-remote-codecommit作為使用 CodeCommit 的登入資料協助程式的替代方法。

  如需詳細資訊，請參閱 HTTPS 連線的設定步驟AWS CodeCommit與git-remote-codecommit、對於在視窗上使用 HTTPS 連線AWS CLI憑證助手 及 適用於窗口的 Git：我已安裝適用於 Windows 的 Git，但現在拒絕我存取儲存庫 (403)。

存取錯誤：在連接到 CodeCommit 儲存庫時公有金鑰遭拒

問題：當您嘗試使用 SSH 端點來與 CodeCommit 儲存庫通訊，出現含有語句的錯誤訊息`Error: public key denied`。

可能的修正：此錯誤最常見的原因是您尚未完成 SSH 連接的設定。配置公有和私有 SSH 金 key pair，然後將公有金鑰與您的 IAM 用户關聯。如需設定 SSH 的詳細資訊，請參閱Linux Linux和適用於 Windows 上的 SSH 連線。

存取錯誤：當連接到 CodeCommit 儲存庫時出現「超過費率」或「429」訊息

問題：當您嘗試與 CodeCommit 儲存庫通訊，出現訊息指出「超過費率」或包含「429」錯誤碼。通訊大幅減慢或失敗。

原因：對 CodeCommit 的所有調用，無論是從應用程序，AWS CLI、Git 客户端或AWS Management Console每秒請求數上限和總體活動請求數上限。您不能超過任何 Amazon Web Services 帳户允許的最大請求率AWS 區域。如果請求超過最大費率，您收到錯誤，您的 Amazon Web Services 帳户的進一步呼叫都會暫時受到調節。在調節期間，您與 CodeCommit 的連線會減慢並可能失敗。

可能的修正：採取步驟減少對 CodeCommit 的連線或呼叫數量或分散請求。一些可考慮的方法：

- 在請求中實作抖動，特別是定期輪詢請求

  如果您的應用程式會定期輪詢 CodeCommit 且此應用程式在多個 Amazon EC2 執行個體上執行，則引進抖動 (隨機延遲量) 可讓不同的 Amazon EC2 執行個體不會在同一秒輪詢。我們建議從 0 到 59 秒的亂數，在一分鐘的時間範圍內平均分配輪詢機制。
- 使用事件型架構，而不是輪詢

  改用事件型架構來取代輪詢，只在發生事件時才進行呼叫。考慮使用 CloudWatch Events 通知AWS CodeCommit事件觸發工作流。
- 實作 API 和自動化 Git 動作的錯誤重試與指數退避

  錯誤重試與指數退避可協助限制呼叫率。每個 AWS 開發套件實作自動重試邏輯和指數退避演算法。對於自動化 Git 推送和 Git 提取，您可能需要實作自己的重試邏輯。如需詳細資訊，請參閱「」中的錯誤重試與指數退避AWS。
- 請求 CodeCommit 服務配額提高AWS支援中心

  若想收到服務限制提高，您必須確認您已遵守此處提供的建議，包括實作錯誤重試或指數退避方法。在您的請求中，您也必須提供AWS 區域、Amazon Web Services 帳户和受到調節問題影響的時間範圍。

故障排除配置錯誤和AWS CodeCommit

以下資訊可以協助您對與 AWS CodeCommit 儲存庫連線時可能會看到的組態錯誤進行故障診斷。

主題

組態錯誤：無法配置AWS CLImacOS 上的憑據

問題：當你運行`aws configure`配置AWS CLI，您會看到`ConfigParseError`訊息。

可能的修正：此錯誤最常見的原因是登入資料檔案已存在。瀏覽到 ~/.aws 並尋找名為 `credentials` 的檔案。重新命名或刪除該檔案，然後再次執行 aws configure。

解決控制台錯誤和 AWS CodeCommit

以下資訊可以協助您對使用 AWS CodeCommit 儲存庫時的主控台錯誤進行故障診斷。

主題

存取錯誤：從主控台或 AWS CLI 對 CodeCommit 儲存庫的加密金鑰存取遭拒

問題：當您嘗試 CodeCommit 從主控台或存取時AWS CLI，會出現包含該片語`EncryptionKeyAccessDeniedException`或的錯誤訊息`User is not authorized for the KMS default key for CodeCommit 'aws/codecommit' in your account`。

可能的修正：造成此錯誤的最常見原因是您的 Amazon Web Services 帳戶未訂閱AWS Key Management Service，這是所需的 CodeCommit。開啟主AWS KMS控台，選擇AWS受管理的金鑰，然後選擇 [立即開始使用]。如果您看到訊息說明您目前尚未訂閱 AWS Key Management Service 服務，請按照該頁面上的指示進行訂閱。若要取得有關 CodeCommit 和的更多資訊AWS Key Management Service，請參閱AWS KMS和加密。

加密錯誤：無法解密存儲庫

問題：當您嘗試從主控台或存取 CodeCommit 存放庫時AWS CLI，會出現包含該片語的錯誤訊息`Repository can't be decrypted`。

可能的修正：造成此錯誤的最常見原因是，用於加密和解密此儲存庫資料的AWS KMS金鑰不在作用中或擱置刪除。需要使用中AWS 受管金鑰或客戶管理AWS Key Management Service的金鑰 CodeCommit。開啟AWS KMS主控台、選擇AWS 受管金鑰或 Custo mer Managed 金鑰，並確定儲存庫所使用的金鑰存在於儲存庫所在的AWS 區域位置，且其狀態為「作用中」。若要取得有關 CodeCommit 和的更多資訊AWS Key Management Service，請參閱AWS KMS和加密。

> **Important**
>
> 如果用於加密和解密儲存庫資料的金鑰已永久刪除或無法存取，則無法存取使用該金鑰加密的儲存庫中的資料。

控制台錯誤：無法從控制台瀏覽 CodeCommit 存儲庫中的代碼

問題：嘗試從主控台瀏覽儲存庫的內容時，出現拒絕存取的錯誤訊息。

可能的修正：發生此錯誤的最常見原因是，套用至 Amazon Web Services 帳戶的 IAM 政策拒絕從 CodeCommit 主控台瀏覽程式碼所需的一或多個許可。如需有關 CodeCommit 存取權限和瀏覽的詳細資訊，請參閱AWS CodeCommit 的身分驗證與存取控制。

顯示錯誤：無法檢視檔案或檔案之間的比較

問題：當您嘗試在 CodeCommit 主控台中檢視檔案或檔案的兩個版本之間進行比較時，會出現錯誤，指出檔案或差異太大而無法顯示。

可能的修正：造成此錯誤的最常見原因是檔案太大而無法顯示、包含一或多行超出檔案中單行字元限制的行，或檔案兩個版本之間的差異超過行限制。如需詳細資訊，請參閱配額。若要檢視檔案或檔案版本之間的差異，您可以在偏好的 IDE 中在本機開啟檔案、使用 Git diff 工具或執行`git diff`指令。

觸發和的疑難排解AWS CodeCommit

以下資訊可能有助於您對 AWS CodeCommit 中的觸發問題進行疑難排解。

主題

- 觸發器錯誤: 儲存庫觸發未如預期樣地執行

觸發器錯誤: 儲存庫觸發未如預期樣地執行

問題：為儲存庫設定的一或多個觸發似乎未如預期樣執行。

可能的修正：如果觸發器的目標是AWS Lambda函數，請確保您已經將函數的資源政策設定為供CodeCommit。如需詳細資訊，請參閱 範例 3：建立與 CodeCommit 觸發器AWS Lambda整合的策略。

或者，編輯觸發，並確保已選取要據以觸發動作的事件，而且觸發的分支包含您想看到動作回應的分支。嘗試將觸發的設定變更為 All repository events (所有儲存庫事件) 和 All branches (所有分支)，然後測試觸發。如需詳細資訊，請參閱編輯儲存庫的觸發程式。

開啟偵錯

問題：我想要開啟偵錯，以取得儲存庫和 Git 儲存庫執行命令情況的更多資訊。

可能的修正：嘗試下列項目：

1. 在終端機或命令提示字元中，於您的本機電腦上執行下列命令，之後再執行 Git 命令：

   在 Linux、macOS 系統或 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**
   >
   > 設定 `GIT_CURL_VERBOSE` 僅適用 HTTPS 連線。SSH 不會使用 `libcurl` 程式庫。

2. 要獲取有關 Git 存儲庫的更多信息，我們建議您安裝最新版本的 git- sizer。依照指示安裝適合您作業系統和環境的公用程式。安裝完成後，在命令列或終端機上，將目錄變更為本機儲存庫，然後執行下列命令：

```
git-sizer --verbose
```

> **Tip**
>
> 請考慮將命令的輸出儲存至檔案，以便在疑難排解問題時 (特別是隨著時間的推移) 輕鬆與其他人共用。

AWS CodeCommit 參考

下列參考主題可協助您進一步瞭解 Git CodeCommit AWS 區域、服務限制等。

主題

- 區域和 Git 連線端點 AWS CodeCommit
- AWS CodeCommit 與介面 VPC 端點搭配使用
- 配額 AWS CodeCommit
- AWS CodeCommit 指令行參考
- 基本的 Git 命令

區域和 Git 連線端點 AWS CodeCommit

每個 CodeCommit 儲存庫都與 AWS 區域. CodeCommit 提供區域端點以向服務提出請求。此外，在每個可用的區域中，為 SSH 和 HTTPS 通訊協定 CodeCommit 提供 Git 連線端點。 CodeCommit

本指南中的所有範例都使用與美國東部 (俄亥俄) 的 Git 相同端點 URL: `git-codecommit.us-east-2.amazonaws.com`。不過，當您使用 Git 並設定連線時，請務必選擇與託管 CodeCommit 儲存庫的 Git 連線端點相符。 AWS 區域 例如，如果您想要與美國東部 (維吉尼亞北部) 的存放庫建立連線，請使用的端點 URL `git-codecommit.us-east-1.amazonaws.com`。這也適用於 API 呼叫。當您使用 AWS CLI 或 SDK 連線到 CodeCommit 存放庫時，請確定您使用存放庫的正確地區端點。

主題

- 支 AWS 區域 援 CodeCommit
- Git 連接端點
- 伺服器指紋 CodeCommit

支 AWS 區域 援 CodeCommit

您可以在下列項目中建立和使用 CodeCommit 儲存庫 AWS 區域：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)

- 美國西部 (奧勒岡)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- Europe (Paris)
- 歐洲 (法蘭克福)
- 歐洲 (斯德哥爾摩)
- 歐洲 (米蘭)
- 非洲 (開普敦)
- 以色列 (特拉維夫)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 亞太區域 (雅加達)
- 中東 (阿拉伯聯合大公國)
- 亞太區域 (首爾)
- 亞太區域 (大阪)
- 亞太區域 (孟買)
- 亞太區域 (海德拉巴)
- 亞太區域 (香港)
- 南美洲 (聖保羅)
- Middle East (Bahrain)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- AWS GovCloud (美國西部)
- AWS GovCloud (美國東部)

CodeCommit 在某些地區增加了對聯邦信息處理標準（FIPS）140-2 出版物政府標準的支持。如需有關 FIPS 和 FIPS 端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。對於支援 FIPS 的 Git 連線端點的詳細資訊，請參閱 Git 連接端點。

如需有關地區端點 AWS CLI、服務和 API 呼叫的詳細資訊 CodeCommit，請參閱AWS CodeCommit 端點和配額。

Git 連接端點

當您設定 Git 連線至 CodeCommit儲存庫時，請使用下列 URL：

Git 連接端點 AWS CodeCommit

區域名稱	區域	端點 URL	通訊協定
美國東部 (俄亥俄)	us-east-2	https://git-codecommit.us-east-2.amazonaws.com	HTTPS
美國東部 (俄亥俄)	us-east-2	ssh: //git 代碼提交. 我東-2. 亞馬遜	SSH
美國東部 (俄亥俄)	us-east-2	美國東部-亞馬遜git-codecommit-fips網站	HTTPS
美國東部 (維吉尼亞北部)	us-east-1	https://git-codecommit.us-east-1.amazonaws.com	HTTPS
美國東部 (維吉尼亞北部)	us-east-1	ssh: //git 代碼提交. 我東-1. 亞馬遜	SSH
美國東部 (維吉尼亞北部)	us-east-1	亞馬遜網站://git-codecommit-fips. 美國東部	HTTPS
美國西部 (奧勒岡)	us-west-2	https://git-codecommit.us-west-2.amazonaws.com	HTTPS
美國西部 (奧勒岡)	us-west-2	ssh: //git 代碼提交. 我們西部-2. 亞馬遜	SSH
美國西部 (奧勒岡)	us-west-2	美國西部-亞馬遜git-codecommit-fips網站	HTTPS

區域名稱	區域	端點 URL	通訊協定
美國西部 (加利佛尼亞北部)	us-west-1	https://git-codecommit.us-west-1.amazonaws.com	HTTPS
美國西部 (加利佛尼亞北部)	us-west-1	ssh: //git 代碼提交. 我-西部-1. 亞馬遜	SSH
美國西部 (加利佛尼亞北部)	us-west-1	美國西部-1git-codecommit-fips. 亞馬遜	HTTPS
歐洲 (愛爾蘭)	eu-west-1	https://git-codecommit.eu-west-1.amazonaws.com	HTTPS
歐洲 (愛爾蘭)	eu-west-1	ssh: //git 代碼提交. 歐盟西部-1. 亞馬遜	SSH
亞太區域 (東京)	ap-northeast-1	https://git-codecommit.ap-northeast-1.amazonaws.com	HTTPS
亞太區域 (東京)	ap-northeast-1	ssh: //git 代碼提交. AP-東北-1. 亞馬遜	SSH
亞太區域 (新加坡)	ap-southeast-1	https://git-codecommit.ap-southeast-1.amazonaws.com	HTTPS
亞太區域 (新加坡)	ap-southeast-1	ssh: //git 代碼提交. AP-東南部-	SSH
亞太區域 (悉尼)	ap-southeast-2	https://git-codecommit.ap-southeast-2.amazonaws.com	HTTPS

區域名稱	區域	端點 URL	通訊協定
亞太區域 (悉尼)	ap-southeast-2	ssh: //git 代碼提交. AP-東南部-	SSH
亞太區域 (雅加達)	ap-southeast-3	https://git-codecommit.ap-southeast-3.amazonaws.com	HTTPS
亞太區域 (雅加達)	ap-southeast-3	ssh: //git 代碼提交. AP-東南部	SSH
中東 (阿拉伯聯合大公國)	me-central-1	https://git-codecommit.me-central-1.amazonaws.com	HTTPS
中東 (阿拉伯聯合大公國)	me-central-1	ssh：//git 代碼提交. 我中心 -1.Amazonaws.com	SSH
歐洲 (法蘭克福)	eu-central-1	https://git-codecommit.eu-central-1.amazonaws.com	HTTPS
歐洲 (法蘭克福)	eu-central-1	ssh: //git 代碼提交. 歐盟中央 -1.Amazonaws.com	SSH
亞太區域 (首爾)	ap-northeast-2	https://git-codecommit.ap-northeast-2.amazonaws.com	HTTPS
亞太區域 (首爾)	ap-northeast-2	ssh: //git 代碼提交. AP-東北-2.	SSH
南美洲 (聖保羅)	sa-east-1	https://git-codecommit.sa-east-1.amazonaws.com	HTTPS

區域名稱	區域	端點 URL	通訊協定
南美洲 (聖保羅)	sa-east-1	ssh: //git 代碼提交. SA-東-阿馬遜	SSH
歐洲 (倫敦)	eu-west-2	https://git-codecommit.eu-west-2.amazonaws.com	HTTPS
歐洲 (倫敦)	eu-west-2	ssh: //git 代碼提交. 歐盟西部-2. 亞馬遜	SSH
亞太區域 (孟買)	ap-south-1	https://git-codecommit.ap-south-1.amazonaws.com	HTTPS
亞太區域 (孟買)	ap-south-1	ssh: //git 代碼提交. AP-南 1.	SSH
亞太區域 (海德拉巴)	ap-south-2	https://git-codecommit.ap-south-2.amazonaws.com	HTTPS
亞太區域 (海德拉巴)	ap-south-2	ssh: //git 代碼提交. AP-南 2. 亞馬遜	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 代碼提交. CA-中央 -1.Amazonaws.com	SSH
加拿大 (中部)	ca-central-1	https://git-codecommit-fips.ca-中央-1. 亞馬遜	HTTPS

區域名稱	區域	端點 URL	通訊協定
歐洲 (巴黎)	eu-west-3	https://git-codecommit.eu-west-3.amazonaws.com	HTTPS
歐洲 (巴黎)	eu-west-3	ssh: //git 代碼提交. 歐盟西部-3.	SSH
AWS GovCloud (美國西部)	us-gov-west-1	https://git-codecommit.us-gov-west-1. 亞馬遜	HTTPS
AWS GovCloud (美國西部)	us-gov-west-1	ssh：//git 代碼提交。us-gov-west-1. 亞馬遜	SSH
AWS GovCloud (美國西部)	us-gov-west-1	網址：//git-codecommit-fips。 us-gov-west-1. 亞馬遜	HTTPS
AWS GovCloud (美國東部)	us-gov-east-1	https://git-codecommit.us-gov-east-1. 亞馬遜	HTTPS
AWS GovCloud (美國東部)	us-gov-east-1	ssh：//git 代碼提交。us-gov-east-1. 亞馬遜	SSH
AWS GovCloud (美國東部)	us-gov-east-1	網址：//git-codecommit-fips。 us-gov-east-1. 亞馬遜	HTTPS
歐洲 (斯德哥爾摩)	eu-north-1	https://git-codecommit.eu-north-1.amazonaws.com	HTTPS
歐洲 (斯德哥爾摩)	eu-north-1	ssh: //git 代碼提交. 歐盟北部.	SSH
中東 (巴林)	me-south-1	https://git-codecommit.me-south-1.amazonaws.com	HTTPS

區域名稱	區域	端點 URL	通訊協定
中東 (巴林)	me-south-1	ssh: //git 代碼提交. 我南 1. 亞馬遜	SSH
亞太區域 (香港)	ap-east-1	https://git-codecommit.ap-east-1.amazonaws.com	HTTPS
亞太區域 (香港)	ap-east-1	ssh: //git 代碼提交. AP-東-1. 亞馬遜	SSH
中國 (北京)	cn-north-1	https://git-codecommit.cn-north-1.amazonaws.com.cn	HTTPS
中國 (北京)	cn-north-1	ssh: //git 代碼提交. CN-北-阿馬遜.	SSH
中國 (寧夏)	cn-northwest-1	https://git-codecommit.cn-northwest-1.amazonaws.com.cn	HTTPS
中國 (寧夏)	cn-northwest-1	ssh: //git 代碼提交. CN-西北-1. 亞馬遜	SSH
歐洲 (米蘭)	eu-south-1	https://git-codecommit.eu-south-1.amazonaws.com	HTTPS
歐洲 (米蘭)	eu-south-1	ssh: //git 代碼提交. 歐盟南部 1.	SSH
亞太區域 (大阪)	ap-northeast-3	https://git-codecommit.ap-northeast-3.amazonaws.com	HTTPS
亞太區域 (大阪)	ap-northeast-3	ssh: //git 代碼提交. AP-東北-3.	SSH

區域名稱	區域	端點 URL	通訊協定
非洲 (開普敦)	af-south-1	https://git-codecommit.af-south-1.amazonaws.com	HTTPS
非洲 (開普敦)	af-south-1	ssh: //git 代碼提交. AF-南部	SSH
以色列 (特拉維夫)	il-central-1	https://git-codecommit.il-central-1.amazonaws.com	HTTPS
以色列 (特拉維夫)	il-central-1	ssh: //git 代碼提交. 中央-1.Amazonaws.com	SSH

伺服器指紋 CodeCommit

下表列出中 Git 連線端點的公用指紋 CodeCommit。將端點新增到已知主機檔案時，在驗證過程中會顯示這些伺服器指紋。

的公共指紋 CodeCommit

Server	密碼雜湊類型	指紋
git 代碼提交. 美東 2. 亞馬遜	MD5	`a9:6d:03:ed:08:42:21:be:06:e1:e0:2a:d1:75:31:5e`
git 代碼提交. 美東 2. 亞馬遜	SHA256	`3lBlW2g5xn/NA2Ck6dyeJIrQOWvn7n8UEs56fG6ZIzQ`
git 代碼提交. 我東-1. 亞馬遜	MD5	`a6:9c:7d:bc:35:f5:d4:5f:8b:ba:6f:c8:bc:d4:83:84`

Server	密碼雜湊類型	指紋
git 代碼提交. 我東-1. 亞馬遜	SHA256	eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY
git 代碼提交. 我們西部-亞馬遜	MD5	a8:68:53:e3:99:ac:6e:d7:04:7e:f7:92:95:77:a9:77
git 代碼提交. 我們西部-亞馬遜	SHA256	0pJx9SQpkbPUAHwy58UVIq0IHcyo1fwCpOOuVgcAWPo
git 代碼提交. 歐盟西部-亞馬遜	MD5	93:42:36:ea:22:1f:f1:0f:20:02:4a:79:ff:ea:12:1d
git 代碼提交. 歐盟西部-亞馬遜	SHA256	tKjRkOL8dmJyTmSbeSdN1S8F/f0iql3RlvqgTOP1UyQ
git 代碼提交. AP-東北-1. 亞馬遜	MD5	8e:a3:f0:80:98:48:1c:5c:6f:59:db:a7:8f:6e:c6:cb
git 代碼提交. AP-東北-1. 亞馬遜	SHA256	Xk/WeYD/K/bnBybzhiuu4dWpBJtXPf7E30jHU7se4Ow
git 代碼提交. AP-東南部-1. 亞馬遜	MD5	65:e5:27:c3:09:68:0d:8e:b7:6d:94:25:80:3e:93:cf
git 代碼提交. AP-東南部-1. 亞馬遜	SHA256	ZIsVa7OVzxrTIf+Rk4UbhPv6Es22mSB3uTBojfPXIno

Server	密碼雜湊類型	指紋
git 代碼提交. AP-東南部-2. 亞馬遜	MD5	7b:d2:c1:24:e6:91:a5:7b:fa:c1:0c:35:95:87:da:a0
git 代碼提交. AP-東南部-2. 亞馬遜	SHA256	nYp+gHas80HY3DqbP4yanCDFhqDVjseefVbHEXqH2Ec
git 代碼提交. AP-東南部-3. 亞馬遜	MD5	64:d9:e0:53:19:4f:a8:91:9a:c3:53:22:a6:a8:ed:a6
git 代碼提交. AP-東南部-3. 亞馬遜	SHA256	ATdkGSFhpqIu7RqUVT/1RZo6MLxxxUW9NoDVMbAc/6g
git 代碼提交. 我中心 1. 亞馬遜	MD5	bd:fa:e2:f9:05:84:d6:39:6f:bc:d6:8d:fe:de:61:76
git 代碼提交. 我中心 1. 亞馬遜	SHA256	grceUDWubo4MzG1NoaKZKUfrgPvfN3ijliOnQr1lTZA
git 代碼提交. 歐盟中央-1. 亞馬遜	MD5	74:5a:e8:02:fc:b2:9c:06:10:b4:78:84:65:94:22:2d
git 代碼提交. 歐盟中央-1. 亞馬遜	SHA256	MwGrkiEki8QkkBtlAgXbYt0hoZYBnZF62VY5RzGJEUY
git 代碼提交. AP-東北-2. 亞馬遜	MD5	9f:68:48:9b:5f:fc:96:69:39:45:58:87:95:b3:69:ed

Server	密碼雜湊類型	指紋
git 代碼提交. AP-東北-2. 亞馬遜	SHA256	eegAPQrWY9YsYo9ZHIKOmxetfXBHzAZd8Eya53Qcwko
git 代碼提交. SA-東-阿馬遜	MD5	74:99:9d:ff:2b:ef:63:c6:4b:b4:6a:7f:62:c5:4b:51
git 代碼提交. SA-東-阿馬遜	SHA256	kW+VKB0jpRaG/ZbXkgbtMQbKgEDK7JnISV3SVoyCmzU
git 代碼提交. 我們西部-亞馬遜	MD5	3b:76:18:83:13:2c:f8:eb:e9:a3:d0:51:10:32:e7:d1
git 代碼提交. 我們西部-亞馬遜	SHA256	gzauWTWXDK2u5KuMMi5vbKTmfyerdIwgSbzYBODLpzg
git 代碼提交. 歐盟西部-亞馬遜	MD5	a5:65:a6:b1:84:02:b1:95:43:f9:0e:de:dd:ed:61:d3
git 代碼提交. 歐盟西部-亞馬遜	SHA256	r0Rwz5k/IHp/QyrRnfiM9j02D5UEqMbtFNTuDG2hNbs
git 代碼提交. AP-南 1. 亞馬遜	MD5	da:41:1e:07:3b:9e:76:a0:c5:1e:64:88:03:69:86:21
git 代碼提交. AP-南 1. 亞馬遜	SHA256	hUKwnTj7+Xpx4Kddb6p45j4RazIJ4IhAMD8k29itOfE

Server	密碼雜湊類型	指紋
git 代碼提交. AP-南 2. 亞馬遜	MD5	bc:cc:9f:15:f8:f3:58:a2:68:65:21:e2:23:71:8d:ce
git 代碼提交. AP-南 2. 亞馬遜	SHA256	Xe0CyZEOvgR5Xa2YUGqf+jn8/Ut7l7nX/CmslSFNEig
git 代碼提交. CA-中央-1. 亞馬遜	MD5	9f:7c:a2:2f:8c:b5:74:fd:ab:b7:e1:fd:af:46:ed:23
git 代碼提交. CA-中央-1. 亞馬遜	SHA256	Qz5puafQdANVprLlj6r0Qyh4lCNsF6ob61dGcPtFS7w
git 代碼提交. 歐盟西部-3. 亞馬遜	MD5	1b:7f:97:dd:d7:76:8a:32:2c:bd:2c:7b:33:74:6a:76
git 代碼提交. 歐盟西部-3. 亞馬遜	SHA256	uw7c2FL564jVoFgtc+ikzILnKBsZz7t9+CFdSJjKbLI
git 代碼提交。 us-gov-west-1. 亞馬遜	MD5	9f:6c:19:3b:88:cd:e8:88:1b:9c:98:6a:95:31:8a:69
git 代碼提交。 us-gov-west-1. 亞馬遜	SHA256	djXQoSIFcg8vHe0KVH1xW/gOF9X37tWTqu4Hkng75x4
git 代碼提交。 us-gov-east-1. 亞馬遜	MD5	00:8d:b5:55:6f:05:78:05:ed:ea:cb:3f:e6:f0:62:f2

Server	密碼雜湊類型	指紋
git 代碼提交。 us-gov-east-1. 亞馬遜	SHA256	fVb+R0z7qW7minenW+rUpAABRCRBTCzmETAJEQrg98
地址代碼提交. 歐盟北 1. 亞馬遜	MD5	8e:53:d8:59:35:88:82:fd:73:4b:60:8a:50:70:38:f4
地址代碼提交. 歐盟北 1. 亞馬遜	SHA256	b6KSK7xKq+V8jl7iuAcjqXsG7zkqoUZZmmhYYFBq1wQ
git 代碼提交. 我-南 1. 亞馬遜	MD5	0e:39:28:56:d5:41:e6:8d:fa:81:45:37:fb:f3:cd:f7
git 代碼提交. 我-南 1. 亞馬遜	SHA256	O+NToCGgjrHekiBuOl0ad7ROGEsz+DBLXOd/c9wc0JU
git 代碼提交. AP-東-1. 亞馬遜	MD5	a8:00:3d:24:52:9d:61:0e:f6:e3:88:c8:96:01:1c:fe
git 代碼提交. AP-東-1. 亞馬遜	SHA256	LafadYwUYW8hONoTRpojbjNs9IRnbEwHtezD3aAIBX0
git 代碼提交. CN-北 1. 亞馬遜.	MD5	11:7e:2d:74:9e:3b:94:a2:69:14:75:6f:5e:22:3b:b3
git 代碼提交. CN-北 1. 亞馬遜.	SHA256	IYUXxH2OpTDsyYMLIp+JY8CTLS4UX+ZC5JVZXPRaxc8

Server	密碼雜湊類型	指紋
git 代碼提交. CN-西北部-1. 亞馬遜	MD5	2e:a7:fb:4c:33:ac:6c:f9:aa:f2:bc:fb:0a:7b:1e:b6
git 代碼提交. CN-西北部-1. 亞馬遜	SHA256	wqjd6eHd0+mOBx+dCNuL0omUoCNjaDtZiEpWj5TmCfQ
地址代碼提交歐盟南部 1. 亞馬遜	MD5	b9:f6:5d:e2:48:92:3f:a9:37:1e:c4:d0:32:0e:fb:11
地址代碼提交歐盟南部 1. 亞馬遜	SHA256	lyXrWbCg3uQmJrl1XxB/ASR7ugW1Ysf5yzYOJbudHsI
git 代碼提交. AP-東北-3. 亞馬遜	MD5	25:17:40:da:b9:d4:18:c3:b6:b3:fb:ed:1c:20:fe:29
git 代碼提交. AP-東北-3. 亞馬遜	SHA256	2B815B9F0AvwLnRxSVxUz4kDYmtEQUGGdQYP8OQLXhA
GIT 代碼提交. AF-南方 1. 亞馬遜	MD5	21:a0:ba:d7:c1:d1:b5:39:98:8d:4d:7c:96:f5:ca:29
GIT 代碼提交. AF-南方 1. 亞馬遜	SHA256	C34ji3x/cnsDZjUpyNGXdE5pjHYimqJrQZ3leTgqJHM
git 代碼提交. 中央-1. 亞馬遜	MD5	04:74:89:16:98:7a:61:b1:69:46:42:3c:d1:b4:ac:a9

Server	密碼雜湊類型	指紋
git 代碼提交. 中央-1. 亞馬遜	SHA256	`uFxhp51kUWhleTLeYb xQVYm4RnNLNZ5Dbdm1 cgdSl/8`

AWS CodeCommit 與介面 VPC 端點搭配使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 來託管 AWS 資源，則可以在 VPC 和. CodeCommit 您可以使用此連線 CodeCommit 來啟用與 VPC 上的資源進行通訊，而無需透過公用網際網路。

Amazon VPC 是一項 AWS 服務，可用於在您定義的虛擬網路中啟動 AWS 資源。您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。使用 VPC 端點時，VPC 和 AWS 服務之間的路由由由 AWS 網路處理，您可以使用 IAM 政策來控制對服務資源的存取。

若要將 VPC 連接到 CodeCommit，您需要為的定義介面 VPC 端點。 CodeCommit介面端點是具有私有 IP 位址的 elastic network interface，可做為傳送至支援 AWS 服務之流量的進入點。端點提供可靠、可擴充的連線能力， CodeCommit 無需網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連線。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC。

> **Note**
>
> 提供 VPC 支援並與 CodeCommit之整合的其他 AWS 服務 (例如 AWS CodePipeline) 可能不支援使用 Amazon VPC 端點進行該整合。例如， CodePipeline 和之間的流量 CodeCommit 不能限制在 VPC 子網路範圍內。支援整合的服務，例如 AWS Cloud9，可能需要額外的服務，例如 AWS Systems Manager.

介面 VPC 私人雲端端點的支援是一種 AWS 技術 AWS PrivateLink，可使用具有私有 IP 位址的 elastic network interface，在 AWS 服務之間進行私人通訊。如需詳細資訊，請參閱AWS PrivateLink。

下列步驟適用於 Amazon VPC 的使用者。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的入門。

可用性

CodeCommit 目前支援下列各項的 VPC 端點： AWS 區域

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- Europe (Paris)
- 歐洲 (法蘭克福)
- 歐洲 (斯德哥爾摩)
- 歐洲 (米蘭)
- 非洲 (開普敦)
- 以色列 (特拉維夫)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 亞太區域 (雅加達)
- 中東 (阿拉伯聯合大公國)
- 亞太區域 (首爾)
- 亞太區域 (大阪)
- 亞太區域 (孟買)
- 亞太區域 (海德拉巴)
- 亞太區域 (香港)
- 南美洲 (聖保羅)
- Middle East (Bahrain)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- AWS GovCloud (美國西部)

- AWS GovCloud (美國東部)

為以下項目建立 VPC 端點 CodeCommit

若要開 CodeCommit 始使用您的 VPC，請為. CodeCommit CodeCommitGit 操作和 CodeCommit API 操作需要單獨的端點。根據您的商業需求，您可能需要建立多個 VPC 端點。為建立 VPC 端點時 CodeCommit，請選擇AWS 服務，然後在服務名稱中選擇下列選項：

- COM. 亞馬遜。 ## .git-code 認可：如果您想要為具有儲存庫的 Git 作業建立 VPC 端點，請選擇此選項。 CodeCommit 例如，如果您的使用者使用 Git 用戶端和指令 (例如、)`git pull`，以及與 CodeCommit 儲存庫互動`git push`時`git commit`，請選擇此選項。
- COM. 亞馬遜。 ##。 git-codecommit-fips：如果您想要使用符合聯邦資訊處理標準 (FIPS) 出版物 140-2 美國政府標準的 CodeCommit 儲存庫建立適用於 Git 作業的 VPC 端點，請選擇此選項。

> **Note**
>
> 並非所有 AWS 區域都可以使用適用於 Git 的 FIPS 端點。如需詳細資訊，請參閱 Git 連接端點。

- COM. 亞馬遜。 ## .codecommit：如果您要建立用 CodeCommit 於 API 作業的 VPC 端點，請選擇此選項。例如，如果您的使用者使用 AWS CLI、 CodeCommit API 或 AWS SDK 來與作業 (例如、和`PutFile`) 互 CodeCommit 動 `CreateRepository`、`ListRepositories`，請選擇此選項。
- COM. 亞馬遜。 ## .codecommit-fips：如果您要建立符合聯邦資訊處理標準 (FIPS) 出版物 140-2 美國政府標準的 CodeCommit API 作業的 VPC 端點，請選擇此選項。

> **Note**
>
> 並非所有 AWS 區域均提供 FIPS 端點。如需詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀 AWS CodeCommit 中的項目。

為以下項目建立 VPC 端點原則 CodeCommit

您可以為 Amazon VPC 端點建立政策，您可以 CodeCommit 在其中指定：

- 可執行動作的主體。
- 可執行的動作。

- 可對其執行動作的資源。

例如，一家公司可能需要限制只能從 VPC 的網路地址範圍存取儲存庫。您可以在此處查看這類政策的範例：範例 3：允許使用者從指定的 IP 位址範圍連線存取儲存庫 。該公司為美國東部 (俄亥俄) 區域設定了兩個 Git VPC 端點：com.amazonaws.us-east-2.codecommit和com-amazonaws.us-east-2.git-codecommit-fips. 他們希望允許代碼推送到*MyDemoRepo*僅在 FIPS 兼容端點上名為的 CodeCommit 存儲庫。為了強制此行為，他們在 com.amazonaws.us-east-2.codecommit 端點上設定類似如下的政策，以明確拒絕 Git 推送動作：

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "codecommit:GitPush",
            "Effect": "Deny",
            "Resource": "arn:aws:codecommit:us-west-2:123456789012:MyDemoRepo",
            "Principal": "*"
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立界面端點。

配額 AWS CodeCommit

下表說明中的配額 CodeCommit。如需可變更之配額的相關資訊，請參閱 AWS CodeCommit 端點和配額。如需要請求增加 Service Quotas 的相關資訊，請參閱AWS 服務配額。如需 Git 和其他軟體所需版本的相關資訊，請參閱 CodeCommit、Git 和其他元件的相容性。

核准規則與核准規則範本名稱	字母、數字、句號、空格、底線和連字號的任何組合，長度介於 1 到 100 個字元之間。名稱區分大小寫。名稱的結尾不可以是 .git，並且不得包含以下任何字元：！ ？ @ # $ % ^ &

	* ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
核准規則內容長度	3000 個字元
核准規則範本描述長度	1000 個字元
核准規則範本目的地參考	100
核准規則範本	千在一 AWS 區域
提取請求的核准規則	最多 30 個。其中最多 25 個可以來自核准規則範本。
從核准規則範本建立的提取請求上的核准規則	25
提取請求的核准	200
核准集區中的核准者	50
分支名稱	長度介於 1 到 256 個字元之間的任何允許字元組合，但分支名稱只有 40 個十六進位字元不允許。分支名稱不可以： • 開頭或結尾為斜線 (/) 或句號 (.)。 • 包含單一字元 @ • 包含兩或多個連續的句號 (..)、正斜線 (//) 或以下字元的組合：@{ • 包含空格或以下任何字元：? ^ * [\ ~ : 分支名稱為參考。分支名稱的許多限制是根據 Git 參考標準。如需詳細資訊，請參閱 Git 內部和 git-check-ref-format.
註解長度	最多可輸入 1、2 個字元。
觸發的自訂資料	這是最多為 1,000 個字元的字串欄位。無法用於傳遞任何動態參數。

在主控台中顯示	在下列情況下，可能無法在主控台中檢視檔案或檔案之間的比較： • 檔案大於 2 MB • 該文件在一行中包含超過 25,000 個字符 • 一個比較包含超過 6,500 行差異
在主控台中進行遞交的電子郵件地址	允許字元的任何組合，長度介於 1 到 256 個字元之間。不會對電子郵件地址進行驗證。

檔案路徑	允許的字元的任何組合，長度介於 1 到 4,096 個字元之間。檔案路徑必須是明確的名稱，指定檔案和檔案的確切位置。檔案路徑的深度不能超過 20 個目錄。此外，檔案路徑不可以：
	• 包含空白字串
	• 是相對檔案路徑
	• 包含以下任何字元組合：
	`/./`
	`/../`
	`//`
	• 結尾為尾端斜線或反斜線
	
	檔案名稱和路徑必須是完整合格。本機電腦上檔案的名稱和路徑必須遵循該作業系統的標準。在 CodeCommit 儲存庫中指定檔案的路徑時，請使用 Amazon Linux 的標準。

檔案大小	使用 CodeCommit 主控台、API 或. 時，任何個別檔案的最大 6 MB AWS CLI。

Git Blob 大小	上限為 2 GB。 **Note** 在單一遞交中，所有檔案的數目或大小總計沒有限制，只要中繼資料不超過 6 MB 且單一 Blob 不超過 2 GB 就沒問題。
Commit Visualizer 中的分支圖表顯示	每頁 35 個。如果單一頁面上超過 35 個分支，則不會出現圖表。
提交的元數據	使用 CodeCommit 主控台、API 或 AWS CLI. **Note** 單次提交中所有檔案的數量或總大小沒有限制，只要中繼資料不超過 6 MB、個別檔案不超過 6 MB，且單一 blob 不超過 2 GB 即可。
提交中的檔案數	最多可容納 100 人。
開啟的提取要求數目	最多一千
單一推送中的參考數目	最多 4,000 個，包括建立、刪除和更新。儲存庫中參考的整體數目不受限制。
儲存庫數量	每個 Amazon Web Services 帳戶最多 5,000。此限制可以變更。如需詳細資訊，請參閱AWS CodeCommit 端點和配額和 AWS Service Quotas。
儲存庫中的觸發數目	上限為 10。

區域	CodeCommit 可在以下內容中使用 AWS 區域： • 美國東部 (俄亥俄) • 美國東部 (維吉尼亞北部) • 美國西部 (加利佛尼亞北部) • 美國西部 (奧勒岡) • 歐洲 (愛爾蘭) • 歐洲 (倫敦) • Europe (Paris) • 歐洲 (法蘭克福) • 歐洲 (斯德哥爾摩) • 歐洲 (米蘭) • 非洲 (開普敦) • 以色列 (特拉維夫) • 亞太區域 (東京) • 亞太區域 (新加坡) • 亞太區域 (悉尼) • 亞太區域 (雅加達) • 中東 (阿拉伯聯合大公國) • 亞太區域 (首爾) • 亞太區域 (大阪) • 亞太區域 (孟買) • 亞太區域 (海德拉巴) • 亞太區域 (香港) • 南美洲 (聖保羅) • Middle East (Bahrain) • 加拿大 (中部) • 中國 (北京) • 中國 (寧夏) • AWS GovCloud (美國西部) • AWS GovCloud (美國東部)

	如需詳細資訊，請參閱 區域和 Git 連線端點。
儲存庫描述	字元的任何組合，長度介於 0 到 1,000 個字元之間。儲存庫的描述為選用。
儲存庫名稱	字母、數字、句號、底線和連字號的任何組合，長度介於 1 到 100 個字元之間。名稱區分大小寫。儲存庫名稱的結尾不可以是 .git，並且不得包含以下任何字元：! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
儲存庫標籤金鑰名稱	Unicode 字母、數字、空格，以及 UTF-8 與 1 之間允許字元的任何組合，長度為 128 個字元。允許的字元為 + - = . _ : / @ 標籤金鑰名稱必須是唯一的，而且每個金鑰只能有一個值。標籤不能： • 以 aws: 開頭 • 只包含空格 • 以空格結尾 • 包含表情圖示或以下任何字元：? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;
儲存庫標籤值	Unicode 字母、數字、空格，以及 UTF-8 與 1 之間允許字元的任何組合，長度為 256 個字元。允許的字元為 + - = . _ : / @ 金鑰只能有一個值，但多個金鑰可以有相同的值。標籤不能： • 只包含空格 • 以空格結尾 • 包含表情圖示或以下任何字元： ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

儲存庫標籤	標籤會區分大小寫。每個資源的上限為 50。不允許只有 40 個十六進位字元的標籤名稱。
觸發名稱	字母、數字、句號、底線和連字號的任何組合，長度介於 1 到 100 個字元之間。觸發名稱不可以包含空格或逗號。
在主控台中進行遞交的使用者名稱	允許的字元的任何組合，長度介於 1 到 1,024 個字元之間。

AWS CodeCommit 指令行參考

此參考協助您了解如何使用 AWS CLI。

若要安裝和設定 AWS CLI

1. 在您的本機電腦上，下載並安裝 AWS CLI. 這是從命令列進行互動 CodeCommit 的先決條件。我們建議您安裝 AWS CLI 版本 2。它是最新的主要版本， AWS CLI 並支援所有最新功能。它是唯一支援使用 root 帳戶、同盟存取或暫時登入資料的git-remote-codecommit版本。 AWS CLI

   若要取得更多資訊，請參閱〈使用 AWS 指令行介面進行設置〉。

   > **Note**
   >
   > CodeCommit 僅適用於 1.7.38 及 AWS CLI 更高版本。最佳作法是安裝或升級 AWS CLI 至可用的最新版本。若要判斷 AWS CLI 您已安裝的版本，請執行aws --version指令。若要將舊版的升級 AWS CLI 到最新版本，請參閱安裝 AWS Command Line Interface.

2. 執行此命令以確認是否已安裝 AWS CLI 的命 CodeCommit 令。

   ```
   aws codecommit help
   ```

   此命令返回命 CodeCommit 令列表。

3. 使 AWS CLI 用configure指令來設定使用設定檔，如下所示：

   ```
   aws configure
   ```

出現提示時，指定 AWS 要與之 IAM 使用者搭配使用的存取金鑰和 AWS 秘密存取金鑰 CodeCommit。此外，請務必指定存放庫的存在 AWS 區域 位置，例如us-east-2。系統提示您輸入預設輸出格式時，請指定 json。例如，如果您要為 IAM 使用者設定描述檔：

```
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
```

如需建立和配置要搭配使用之設定檔的詳細資訊 AWS CLI，請參閱下列內容：

- 命名設定檔
- 在中使用 IAM 角色 AWS CLI
- 設定命令
- 使用旋轉認證連線至AWS CodeCommit儲存庫

若要連線至儲存庫或另一個儲存庫中的資源 AWS 區域，您必須 AWS CLI 使用預設的 [區域] 名稱重新配置。支援的預設地區名稱 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

若要取得有關 CodeCommit 和的更多資訊 AWS 區域，請參閱區域和 Git 連線端點。如需 IAM、存取金鑰和秘密金鑰的詳細資訊，請參閱如何取得登入資料？ 以及管理 IAM 使用者的存取金鑰。如需 AWS CLI 和設定檔的詳細資訊，請參閱命名設定檔。

若要檢視所有可用命 CodeCommit 令的清單，請執行下列命令：

```
aws codecommit help
```

```
###### CodeCommit ########################## (##create-repository)#
```

```
aws codecommit command-name help
```

請參閱以下章節以檢視 AWS CLI中的命令描述和範例用法：

- associate-approval-rule-template-與存儲庫
- 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
- 創建分支
- create-commit
- create-pull-request
- create-pull-request-approval-規則
- 創建存儲庫
- create-unreferenced-merge-commit
- delete-approval-rule-template
- delete-branch
- delete-comment-content
- delete-file
- 刪除存儲庫
- describe-merge-conflicts
- delete-pull-request-approval-規則
- describe-pull-request-events
- disassociate-pull-request-approval-rule-template-from-repository
- evaluate-pull-request-approval-規則
- get-approval-rule-template
- get-blob
- 獲取分支
- get-comment
- get-comment-reactions
- get-comments-for-compared-提交
- get-comments-for-pull-請求

- get-commit
- get-differences
- get-merge-commit
- get-merge-conflicts
- get-merge-options
- get-pull-request
- get-pull-request-approval-狀態
- get-pull-request-override-狀態
- 獲取存儲庫
- get-repository-triggers
- list-approval-rule-templates
- list-associated-approval-rule-templates-for-repository
- 列表分支
- list-pull-requests
- 列表存儲庫
- list-repositories-for-approval-規則模板
- list-tags-for-resource
- merge-branches-by-fast-向前
- merge-branches-by-squash
- merge-branches-by-three-方式
- merge-pull-request-by-快進
- merge-pull-request-by-壁球
- merge-pull-request-by-三路
- override-pull-request-approval-規則
- post-comment-for-compared-提交
- post-comment-for-pull-請求
- post-comment-reply
- put-comment-reaction
- put-file

- put-repository-triggers
- tag-resource
- test-repository-triggers
- untag-resource
- update-approval-rule-template-內容
- update-approval-rule-template-說明
- update-approval-rule-template-姓名
- update-comment
- update-default-branch
- update-pull-request-approval-規則內容
- update-pull-request-approval-狀態
- update-pull-request-description
- update-pull-request-status
- update-pull-request-title
- update-repository-description
- update-repository-name

基本的 Git 命令

您可以使用 Git 來處理本機 CodeCommit 存放庫以及您已連接本機存放庫的儲存庫。

以下是一些經常使用的 Git 命令基本範例。

如需更多選項，請參閱您的 Git 文件。

主題

- 組態變數
- 遠端儲存庫
- 遞交
- 分支
- 標籤

組態變數

列出所有組態變數。	`git config --list`
僅列出本機組態變數。	`git config --local -l`
僅列出系統組態變數。	`git config --system -l`
僅列出全域組態變數。	`git config --global -l`
設定指定組態檔案中的組態變數。	`git config [--local | --global | --system] variable-name variable-value`
當對尚未具有默認分支的存儲庫進行初始提交時，將所有本地存儲庫的默認分支名稱設置為 main	git config --global init.defaultBranch main
直接編輯組態檔案。也可以用來探索特定組態檔案的位置。若要結束編輯模式，通常您會輸入 `:q` (結束而不儲存變更) 或 `:wq` (儲存變更然後結束)，然後按 Enter 鍵。	`git config [--local | --global | --system] --edit`

遠端儲存庫

初始化本地存儲庫以準備將其連接到 CodeCommit 存儲庫。	`git init`
可用來使用本機軟體庫對儲存庫所擁有的指定暱稱以及儲存 CodeCommit 庫的指定 URL，在本機存放庫與遠端 CodeCommit 儲存庫 (例如儲存 CodeCommit 庫) 之間建立連線。	`git remote add remote-name remote-url`
在本機電腦上目前資料夾的指定子資料夾中，透過在指定 URL 複製 CodeCommit 存放庫來建立本機存放庫。此命令還會為克隆 CodeCommit 存儲庫中的每個分支創建一個遠程跟踪分支，並	`git clone remote-url local-sub folder-name`

創建並簽出從克隆存儲庫中當前默認分支分支分支的初始分支。 CodeCommit	
顯示本機軟體庫用於 CodeCommit 儲存庫的暱稱。	`git remote`
顯示本地存儲庫用於提取和推送到存儲庫的暱稱和 URL。 CodeCommit	`git remote -v`
使用本地倉庫對 CodeCommit 存儲庫和指定分支的指定暱稱，將最終的提交從本地回購推送到 CodeCommit 存儲庫。還可以在推送期間為本地回購設置上游跟踪信息。	`git push -u` *`remote-name branch-name`*
設置上游跟踪信息後，將最終的提交從本地回購推送到 CodeCommit 存儲庫。	`git push`
使用本地倉庫對 CodeCommit 存儲庫和指定分支的指定暱稱，從存儲 CodeCommit 庫中提取最終提交到本地回購	`git pull` *`remote-name branch-name`*
設置上游跟踪信息後，從 CodeCommit 存儲庫中提取已完成的提交到本地回購。	`git pull`
使用本地存儲庫對 CodeCommit 存儲庫的指定暱稱，從存儲庫中斷本地 CodeCommit 存儲庫的連接。	`git remote rm` *`remote-name`*

遞交

顯示已經或尚未新增到本機儲存庫之待處理遞交中的項目。	`git status`
以簡潔的格式顯示在本地回購中已添加或未添加到待處理提交中的內容。 (M = 已修改，A = 已新增，D = 已刪除等等)	`git status -sb`

顯示待處理提交和本地回購中最新提交之間的更改。	`git diff HEAD`
將特定文件添加到本地回購中的待處理提交中。	`git add [file-name-1 file-name-2 file-name-N | file-pattern ]`
將所有新的、修改的和刪除的檔案新增至本機儲存庫中的待處理遞交。	`git add`
開始在本地回購中完成待處理的提交，它顯示一個編輯器來提供一個提交消息。輸入訊息之後，待處理遞交即完成。	`git commit`
在本地回購中完成待處理的提交，包括同時指定提交消息。	`git commit -m "Some meaningful commit comment"`
列出本地回購中最近的提交。	`git log`
以圖形格式列出本地回購中的最近提交。	`git log --graph`
以預先定義的簡明格式列出本地回購中的最近提交。	`git log --pretty=oneline`
以預先定義的簡明格式列出本地回購中的最近提交，並帶有圖形。	`git log --graph --pretty=oneline`
以自定義格式列出本地回購中的最近提交，並帶有圖形。 (如需選項的詳細資訊，請參閱 Git 基礎 - 檢視遞交歷史記錄)	`git log --graph --pretty=format:"%H (%h) : %cn : %ar : %s"`

分支

列出本地存儲庫中的所有分支，並在當前分支旁邊顯示一個星號（*）。	`git branch`

將 CodeCommit 存儲庫中所有現有分支的信息提取到本地回購。	`git fetch`
列出本地回購中的所有分支和本地回購中的遠程跟踪分支。	`git branch -a`
僅列出本地回購中的遠程跟踪分支。	`git branch -r`
使用指定的分支名稱在本地回購中創建一個新分支。	`git branch new-branch-name`
使用指定的分支名稱切換到本地回購中的另一個分支。	`git checkout other-branch-name`
使用指定的分支名稱在本地回購中創建一個新分支，然後切換到它。	`git checkout -b new-branch-name`
使用本地存儲庫對 CodeCommit 存儲庫的指定暱稱和指定的分支名稱將新分支從本地回購推送到 CodeCommit 存儲庫。還在推送期間為本地回購中的分支設置上游跟踪信息。	`git push -u remote-name new-branch-name`
使用指定的分支名稱在本地回購中創建一個新分支。然後使用本地存儲庫對 CodeCommit 存儲庫的指定暱稱和指定的分支名稱將本地回購中的新分支連接到 CodeCommit 存儲庫中的現有分支。	`git branch --track new-branch-name remote-name /remote-branch-name`
將本地回購中另一個分支的更改合併到本地回購中的當前分支。	`git merge from-other-branch-name`
刪除本地存儲庫中的分支，除非它包含尚未合併的工作。	`git branch -d branch-name`
使用本地 CodeCommit 存儲庫對存儲庫的指定暱稱和指定的分支名稱刪除 CodeCommit 存儲庫中的分支。(注意冒號 (:) 的使用方式。)	`git push remote-name :branch-name`

標籤

列出本地回購中的所有標籤。	`git tag`
從 CodeCommit 存儲庫中提取所有標籤到本地回購。	`git fetch --tags`
顯示有關本地回購中特定標籤的信息。	`git show tag-name`
在本地回購中創建一個「輕量級」標籤。	`git tag tag-name commit-id-to-point-tag-at`
使用本地存儲庫對 CodeCommit 存儲庫和指定標籤名稱的指定暱稱將特定標籤從本地回購推送到 CodeCommit 存儲庫。	`git push remote-name tag-name`
使用本地存儲庫對 CodeCommit 存儲庫的指定暱稱將所有標籤從本地回購推送到 CodeCommit 存儲庫。	`git push remote-name --tags`
刪除本地存儲庫中的標籤。	`git tag -d tag-name`
使用本機軟體 CodeCommit 庫對儲存庫的指定暱稱和指定的標籤名稱，刪除 CodeCommit 儲存庫中的標籤。(注意冒號 (:) 的使用方式。)	`git push remote-name :tag-name`

AWS CodeCommit 使用者指南文件歷史記錄

下表說明的文件的重要變更 CodeCommit。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- API 版本:

變更	描述	日期
CodeCommit 現在支援使用客戶管理金鑰	您現在可以使用客戶管理的金鑰或用AWS 受管金鑰於加密和解密儲存庫中的資料。如需詳細資訊，請參閱AWS KMS和加密、建立存放庫和變更存放庫設定。	2023 年 12 月 21 日
CodeCommit 現已在以色列（特拉維夫）上市；	CodeCommit 現已在以色列（特拉維夫）上市。如需詳細資訊，請參閱區域和 Git 連線端點。	2023 年 8 月 28 日
下列項目的受管理策略變 CodeCommit	AWSCodeCommitPowerUser 和原 AWSCodeCommitFullAccess 則已使用其他權限進行更新。如需詳細資訊，請參閱 AWS 受管政策的 CodeCommit 更新項目。	2023 年 5 月 16 日
CodeCommit 現在有三個額外的可用 AWS 區域	CodeCommit 現在還有三個額外提供AWS 區域：亞太區域 (雅加達)、中東 (阿聯酋) 和亞太區域 (海德拉巴)。如需詳細資訊，請參閱區域和 Git 連線端點。	2023 年 2 月 28 日
CodeCommit 現已在非洲 (開普敦) 推出	CodeCommit 現在是在一個額外的可用AWS 區域:非洲 (開普	2021 年 9 月 15 日

	敦). 如需詳細資訊，請參閱區域和 Git 連線端點。	
下列項目的受管理策略變 CodeCommit	現在提供 CodeCommit 有關受AWS管理策略更新的詳細資料。如需詳細資訊，請參閱 AWS 受管政策的 CodeCommit 更新項目。	2021 年 8 月 18 日
CodeCommit 亞太區域 (大阪) 現已推出	CodeCommit 現在還有一個額外的AWS 區域：亞太區域（大阪）。如需詳細資訊，請參閱區域和 Git 連線端點。	2021 年 4 月 14 日
AWS CloudFormation並AWS Cloud Development Kit (AWS CDK)更改默認分支的命名行為 CodeCommit	使用AWS CloudFormation或使用初始代碼提交創建的存儲庫現在使用 main 的默認分支名稱。AWS CDK此變更不會影響現有的儲存庫或分支。使用本機 Git 用戶端建立初始提交的客戶，會有一個預設的分支名稱，遵循這些 Git 用戶端的設定。如需詳細 CodeCommit 資訊，請參閱使用 AWS CloudFormation.	2021 年 3 月 4 日
CodeCommit 變更預設分支的命名行為	自 2021 年 1 月 19 日起，初始提交 CodeCommit 存儲庫創建的默認分支名稱是主要的。此變更不會影響現有的儲存庫或分支。使用本機 Git 用戶端建立初始提交的客戶，會有一個預設的分支名稱，遵循這些 Git 用戶端的設定。如需詳細資訊，請參閱使用分支、建立提交和變更分支設定。	2021 年 1 月 19 日

CodeCommit 歐洲 (米蘭) 現已推出	CodeCommit 現在是在一個額外的可用AWS 區域:歐洲 (米蘭). 如需詳細資訊，請參閱區域和 Git 連線端點。	2020 年 9 月 16 日
CodeCommit 為評論添加對表情符號反應的支持	CodeCommit 現在支援使用表情符號回應其他使用者的留言。如需詳細資訊，請參閱註解提交和檢閱提取要求。	2020 年 6 月 24 日
CodeCommit 現已在中國（北京）和中國（寧夏）上市	CodeCommit 現在還有兩個額外提供AWS 區域：中國（北京）和中國（寧夏）。如需詳細資訊，請參閱區域和 Git 連線端點。	2020 年 4 月 23 日
CodeCommit 增加了支持 git-remote-codecommit	CodeCommit 支持通過 HTTPS 連接到 CodeCommit 存儲庫git-remote-codecommit，這是一個可修改 Git 的實用程序。對於存放庫的聯合或臨時 CodeCommit 存取連線，建議使用這種方法。您也可以與 IAM 使git-remote-codecommit用者搭配使用。 git-remote-codecommit不需要您為用戶設置 Git 憑據。如需詳細資訊，請參閱AWS CodeCommit使用的 HTTPS 連線的設定步驟 git-remote-codecommit。	2020 年 3 月 4 日

CodeCommit 支持會話標籤	CodeCommit 支援使用工作階段標籤，這些標籤是當您擔任 IAM 角色、使用臨時登入資料或在 AWS Security Token Service () AWS STS 中聯合使用者時傳遞的索引鍵值配對屬性。您能夠使用這些標籤中提供的資訊，以利識別進行變更或導致事件發生的使用者。如需詳細資訊，請參閱監控 CodeCommit 和在 CodeCommit 中使用標籤來提供身分識別資訊。	2019 年 12 月 19 日
CodeCommit 適用於亞太區域 (香港)	您現在可以 CodeCommit 在亞太區域（香港）使用。如需包含 Git 連線端點的詳細資訊，請參閱區域。	2019 年 12 月 11 日
CodeCommit 支持 Amazon 評 CodeGuru 論	CodeCommit 支援 Amazon CodeGuru Reviewer，這是一種自動化程式碼檢閱服務，可使用程式分析和機器學習來偵測常見問題，並在 Java 或 Python 程式碼中建議修正程式。如需詳細資訊，請參閱將儲存庫與 Amazon CodeGuru 審核者建立關聯或取消關聯以及使用提取請求。	2019 年 12 月 3 日
CodeCommit 支援核准規則	您現在可以使用核准規則，協助您自訂跨儲存庫的開發工作流程，讓不同分支能以適當等級來核准和控制提取請求。如需詳細資訊，請參閱使用核准規則樣版和使用提取請求。	2019 年 11 月 20 日

CodeCommit 支援通知規則	您現在可以使用通知規則，向使用者通知儲存庫中的重要變更。此功能的作用取代 2019 年 11 月 5 日之前建立的通知。如需詳細資訊，請參閱建立通知規則。	2019 年 11 月 5 日
CodeCommit 在中東（巴林）可用	您現在可以在 CodeCommit 中東（巴林）使用。如需包含 Git 連線端點的詳細資訊，請參閱區域。	2019 年 10 月 30 日
CodeCommit 添加對檢索有關多個提交的信息的支持	您可以使用中的 batch-get-commits 命令來取得有關多次提交的資訊AWS CLI。如需詳細資訊，請參閱檢視遞交詳細資訊。	2019 年 8 月 15 日
CodeCommit 可在歐洲 (斯德哥爾摩)	您現在可以 CodeCommit 在歐洲（斯德哥爾摩）使用。如需包含 Git 連線端點的詳細資訊，請參閱區域。	2019 年 7 月 31 日
CodeCommit 增加了對 CodeCommit 控制台中標記存儲庫的支持	您現在可以新增、管理和移除存放庫的標籤，以協助您從 CodeCommit 主控台管理AWS資源。如需詳細資訊，請參閱標記儲存庫。	2019 年 7 月 2 日
CodeCommit 增加了對其他 Git 合併策略的支持	您現在在 CodeCommit 中合併提取請求時，可以選擇 Git 合併策略。您也可以在 CodeCommit 主控台中解決合併衝突。如需詳細資訊，請參閱使用提取請求。	2019 年 6 月 10 日

CodeCommit 可在 AWS GovCloud (美國東部)	您現在可以 CodeCommit 在 AWS GovCloud（美國東部）中使用。如需包含 Git 連線端點的詳細資訊，請參閱區域。	2019 年 5 月 31 日
CodeCommit 添加對標記存儲庫的支持	您現在可以新增、管理和移除儲存庫的標籤，以協助您管理 AWS 資源。如需詳細資訊，請參閱標記儲存庫。	2019 年 5 月 30 日
在主控台中尋找資源	您想在可以快速搜尋您的資源，例如儲存庫、組建專案、部署應用程式，以及管道。選擇 Go to resource (移至資源)，或按 / 鍵，然後輸入資源名稱。如需詳細資訊，請參閱CodeCommit 自學課程。	2019 年 5 月 14 日
CodeCommit 可用在 AWS GovCloud (美國西部)	您現在可以 CodeCommit 在 AWS GovCloud（美國西部）中使用。如需包含 Git 連線端點的詳細資訊，請參閱區域。	2019 年 4 月 18 日
CodeCommit 增加了對 Amazon VPC 端點的支持	您現在可以在 VPC 和 CodeCommit. 之間建立私人連線。如需詳細資訊，請參閱 CodeCommit 搭配介面 VPC 端點使用。	2019 年 3 月 7 日
CodeCommit 添加了一個新的 API	CodeCommit 添加了用於創建提交的 API。如需詳細資訊，請參閱建立遞交。	2019 年 2 月 20 日
內容更新	本指南中的內容已更新次要修正和其他故障診斷指南。	2019 年 1 月 2 日

內容更新	本指南中的內容已更新，以支援全新的 CodeCommit 主機體驗。	2018 年 10 月 30 日
CodeCommit 和聯邦資訊處理標準 (FIPS)	CodeCommit 在某些地區增加了對聯邦信息處理標準 (FIPS) 140-2 出版物政府標準的支持。如需有關 FIPS 和 FIPS 端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。如需 Git 連線端點的詳細資訊，請參閱區域。	2018 年 10 月 25 日
CodeCommit 增加了三個 API	CodeCommit 添加了三個 API 來支持使用文件。如需 Git 連線端點的詳細資訊，請參閱針對個別檔案執行動作的權限和 AWS CodeCommitAPI 參考。	2018 年 9 月 27 日
CodeCommit 文檔歷史通知可通過 RSS 提要	您現在可以透過訂閱 RSS 摘要來接收有關 CodeCommit 文件更新的通知。	2018 年 6 月 29 日

舊版更新

下表說明 2018 年 6 月 29 日之前的文件重要變更。

變更	描述	變更日期
新主題	已新增限制推送和合併到分支主題。已更新 CodeCommit 許可參考主題。	2018 年 5 月 16 日
新增 章節	已新增在中使用檔案AWS CodeCommit儲存庫章節。已更新CodeCommit 許可參考和開始使用 AWS CodeCommit主題。	2018 年 2 月 21 日

變更	描述	變更日期
新主題	已新增使用角色設定儲存庫的跨帳戶 AWS CodeCommit 存取主題。	2018 年 2 月 21 日
新主題	已新增將 AWS Cloud9 與 AWS CodeCommit 整合主題。主產品和服務整合題已更新為相關資訊AWS Cloud9。	2017 年 12 月 1 日
新增 章節	已新增在中使用提取請求AWS CodeCommit儲存庫章節。AWS CodeCommit 的身分驗證與存取控制章節中已更新與提取請求和註解的相關許可資訊。還包含更新的受管政策陳述式。	2017 年 11 月 20 日
更新主題	本產品和服務整合主題已更新，包括希望更新現有管道以使用 Amazon E CloudWatch vents 啟動管道以回應 CodeCommit 儲存庫變更的客戶連結。	2017 年 10 月 11 日
新增主題	已新增AWS CodeCommit 的身分驗證與存取控制章節。它會取代「存取許可參考」主題。	2017 年 9 月 11 日
更新主題	已更新管理儲存庫的觸發章節，以反映觸發組態的變更。整本指南已更新主題和影像，以反映導覽列的變更。	2017 年 8 月 29 日
新主題	已新增使用者偏好設定主題。已更新 檢視標籤詳情主題。主產品和服務整合題已更新為與 Amazon CloudWatch 活動整合的相關資訊。	2017 年 8 月 3 日
新增主題	已新增Eclipse 與 AWS CodeCommit 整合 和整合視覺工作室與AWS CodeCommit 主題。	2017 年 6 月 29 日
更新主題	CodeCommit 現已在另外兩個區域提供：亞太區域 (孟買) 和加拿大 (中部)。已更新 區域和 Git 連線端點主題。	2017 年 6 月 29 日
更新主題	CodeCommit 目前在另外四個區域提供：亞太區域 (首爾)、南美洲 (聖保羅)、美國西部 (加利佛尼亞北部) 和歐洲 (倫敦)。已更新 區域和 Git 連線端點主題。	2017 年 6 月 6 日

變更	描述	變更日期
更新主題	CodeCommit 現已在其他四個區域提供：亞太區域 (東京)、亞太區域 (新加坡)、亞太區域 (雪梨) 和歐洲 (法蘭克福)。區域和 Git 連線端點本主題已更新，以提供有關的 Git 連線端點和支援區域的資訊 CodeCommit。	2017 年 5 月 25 日
新主題	已新增比較和合併分支主題。已更新使用分支章節的內容，來提供如何透過 CodeCommit 主控台，使用儲存庫中分支的相關資訊。	2017 年 5 月 18 日
新主題	已新增比較遞交主題來提供有關比較遞交的資訊。已針對使用儲存庫、遞交和分支而更新使用者指南的結構。	2017 年 3 月 28 日
更新主題	已更新檢視遞交詳細資訊主題，來提供在主控台檢視遞交和其父系之間差異的相關資訊，以及在 AWS CLI 中使用 get-differences 命令來檢視遞交之間差異的相關資訊。	2017 年 1 月 24 日
新主題	已新增使用 AWS CloudTrail 記錄 AWS CodeCommit API 呼叫主題，其中包含有關記錄 CodeCommit 使用的連線的資訊AWS CloudFormation。	2017 年 1 月 11 日
新主題	已新增適用於使用 Git 認證的 HTTPS 使用者主題，其中包含設定透過 HTTPS CodeCommit 使用 Git 認證的連線的相關資訊。	2016 年 12 月 22 日
更新主題	已更新產品和服務整合主題來包含 AWS CodeBuild 的整合相關資訊。	2016 年 12 月 5 日
更新主題	CodeCommit 現已在另一個地區推出，歐洲（愛爾蘭）。區域和 Git 連線端點本主題已更新，以提供有關的 Git 連線端點和支援區域的資訊 CodeCommit。	2016 年 11 月 16 日
更新主題	CodeCommit 目前已在美國西部 (奧勒岡) 另一個區域推出。區域和 Git 連線端點本主題已更新，以提供有關的 Git 連線端點和支援區域的資訊 CodeCommit。	2016 年 11 月 14 日

變更	描述	變更日期
新主題	本為 Lambda 函數建立觸發器主題已更新，以反映在建立 Lambda 函數時建立 CodeCommit 觸發器的能力。這個簡化的程序可簡化觸發器的建立，並使用叫用 Lambda 函數所需的權限自動設定觸發器。 CodeCommit 已新增為現有 Lambda 函數建立觸發器主題，其中包含在主控台中為現有 Lambda 函數建立觸發程序的相關資 CodeCommit 訊。	2016 年 10 月 19 日
新主題	CodeCommit 現已在另一個區域推出，美國東部 (俄亥俄)。已新增區域和 Git 連線端點主題，以提供 Git 連線端點和支援區域的相關資訊 CodeCommit。	2016 年 10 月 17 日
主題更新	已更新產品和服務整合主題來包含 AWS Elastic Beanstalk 的整合相關資訊。	2016 年 10 月 13 日
主題更新	已更新產品和服務整合主題來包含 AWS CloudFormation 的整合相關資訊。	2016 年 10 月 6 日
主題更新	已修訂適用於 Windows 上的 SSH 連線主題，來提供在 Windows 上使用 Bash 模擬器 (而不是 PuTTY 工具組) 執行 SSH 連線的相關指導。	2016 年 9 月 29 日
主題更新	檢視遞交詳細資訊和主開始使用 CodeCommit 題已更新，在主 CodeCommit 控台中包含「提交視覺化檢視」的相關資訊。已更新配額主題，來增加單一推送所允許的參考數目。	2016 年 9 月 14 日
主題更新	檢視遞交詳細資訊和主開始使用 CodeCommit 題已更新，包含有關在主 CodeCommit 控台中檢視認可歷程記錄的資訊。	2016 年 7 月 28 日
新增主題	已新增將 Git 儲存庫遷移到 AWS CodeCommit 和將本機或未版本控制的內容移轉至 AWS CodeCommit 主題。	2016 年 6 月 29 日
主題更新	疑難排解 和對於在視窗上使用 HTTPS 連線AWS CLI憑證助手主題已完成次要更新。	2016 年 6 月 22 日
主題更新	產品和服務整合和「存取權限參照」主題已更新，以包含與整合的相關資訊 CodePipeline。	2016 年 4 月 18 日

變更	描述	變更日期
新增主題	已新增管理儲存庫的觸發章節。新的主題包含如何建立、編輯和刪除觸發的範例 (包括政策和程式碼範例)。	2016 年 3 月 7 日
新主題	已新增產品和服務整合主題。疑難排解已完成次要更新。	2016 年 3 月 7 日
主題更新	針對 Linux和 適用於 Windows 上的 SSH 連線，除了 MD5 伺服器指紋，還增加適用於 CodeCommit 的 SHA256 伺服器指紋。	2015 年 12 月 9 日
新主題	已新增瀏覽存儲庫中的文件主題。疑難排解中已增加新的問題。整本使用者指南已完成次要改進和修正。	2015 年 10 月 5 日
新主題	已新增對於 SSH 用户不使用AWS CLI主題。已簡化設定 章節中的主題。已提供指導，此指導可協助使用者依其作業系統和慣用的通訊協定，以決定該遵循的步驟。	2015 年 8 月 5 日
主題更新	在 安全殼層和 Linux、macOS 或 Unix：設定 Git 和私密金鑰CodeCommit和 步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰中，已新增對 SSH 金鑰 ID 步驟的澄清和範例。	2015 年 7 月 24 日
主題更新	中的步驟步驟 3：設定 Git 和 CodeCommit 的公有和私有金鑰已更新，以解決 IAM 的問題並儲存公開金鑰檔案。	2015 年 7 月 22 日
主題更新	已更新疑難排解，來提供導覽輔助工具。關於登入資料金鑰鏈問題，已新增更多疑難排解資訊。	2015 年 7 月 20 日
主題更新	在 AWS KMS和加密和「存取許可參考」主題中，已新增與 AWS Key Management Service 許可相關的詳細資訊。	2015 年 7 月 17 日
主題更新	疑難排解中已多增加一節，提供針對 AWS Key Managemen t Service 的問題進行疑難排解的相關資訊。	2015 年 7 月 10 日
初始版本	此為 CodeCommit 使用者指南的初始版本。	2015 年 7 月 9 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。