aws

使用者指南

Amazon CodeCatalyst

Amazon CodeCatalyst: 使用者指南

Table of Contents

什麼是 Amazon CodeCatalyst？

Amazon CodeCatalyst 是一項整合服務，適用於在軟體開發流程中採用持續整合和部署實務的軟體開發團隊。 CodeCatalyst 將您需要的所有工具放在一個地方。您可以使用持續整合/持續交付 (CI/CD) 工具來規劃工作、協同編寫程式碼，以及建置、測試和部署應用程式。您還可以通過將您的空間連接到您的 CodeCatalyst 空間AWS 帳戶來將AWS資源與項目集成。透過單一工具管理應用程式生命週期的所有階段和層面，您可以快速且自信地交付軟體。

在中 CodeCatalyst，您可以建立空間來代表您的公司、部門或群組，然後建立包含支援開發團隊和任務所需資源的專案。CodeCatalyst資源是在生活在空間內的專案內部結構化的。為了協助團隊快速上手，請 CodeCatalyst 提供以語言或工具為基礎的專案藍圖。當您從專案藍圖建立專案時，專案會隨附包含範例程式碼、建置指令碼、部署動作、虛擬伺服器或無伺服器資源等資源的來源存放庫等資源。

我可以用什麼來做 CodeCatalyst？

您和您的開發團隊可以用 CodeCatalyst 來執行軟體開發的各個層面，從規劃工作到部署應用程式。您可以使用 CodeCatalyst 來：

- 重複執行程式碼並共同作業 — 與您的團隊協同處理程式碼，在原始程式碼儲存庫中使用分支、合併、提取要求和註解的程式碼。建立開發環境以快速處理程式碼，而不必複製或設定儲存庫的連線。
- 使用工作流程建置、測試及部署您的應用程式 — 使用建置、測試和部署動作來設定工作流程，以處理應用程式的持續整合與交付。您可以手動啟動工作流程，也可以將工作流程設定為根據事件 (例如程式碼推送或建立或關閉提取要求) 自動啟動。
- 使用問題追蹤排定團隊工作的優先順序 — 使用問題建立待辦項目，並透過看板監控進行中任務的狀態。創建和維護項目的健康積壓為您的團隊工作是開發軟件的重要組成部分。
- 設定監控和通知 — 監控團隊活動和資源狀態，並配置通知以隨時掌握重要變更的最新狀態。

我該如何開始使用 CodeCatalyst？

如果您沒有空間，或想了解如何設定和管理空間，建議您開始使用 Amazon 管理 CodeCatalyst 員指南。

如果您不熟悉專案或空間工作，我們建議您從下列步驟開始：

- 檢閱 CodeCatalyst 概念
- 建立支援 AWS 產生器 ID 使用者的空間

- 依照中的步驟建立您的第一個專案 教學課程：使用現代三層 Web 應用程式藍圖建立專案

進一步了解 CodeCatalyst

您可以 CodeCatalyst 在本使用者指南中深入瞭解的功能，以及下列資源：

- AWS DevOps 關於 Amazon 的文章 CodeCatalyst
- Amazon CodeCatalyst API 參考指南
- Amazon 行 CodeCatalyst 動開發套件開發人員指南
- CodeCatalyst 常見問
- 感言

CodeCatalyst 概念

熟悉關鍵概念，以協助您加速 Amazon 中的協作和應用程式開發 CodeCatalyst。這些概念包括原始檔控制、持續整合和持續交付 (CI/CD) 中使用的術語，以及模型化和設定自動發行程序。

如需其他概念資訊，請參閱下列主題：

- 來源儲存庫概念
- 工作流概念

主題

- AWS 建置器 ID 空間 CodeCatalyst
- 支援身分聯合的空間 CodeCatalyst
- 專案
- Blueprints (藍圖)
- 帳戶連線
- VPC 端連
- AWS 產生器 ID
- 使用者設定檔 CodeCatalyst
- 來源儲存庫
- 遞交
- 開發環境
- 工作流程
- 動作
- 問題
- 個人存取權杖 (PAT)
- 角色

AWS 建置器 ID 空間 CodeCatalyst

空間管理員通過從成員頁面發 CodeCatalyst 送單獨的邀請電子郵件來邀請用戶。受邀或註冊 CodeCatalyst 建立自己的 AWS Builder ID 的使用者。設定檔在 AWS Builder ID 中管理，並在中的使用者設定中顯示為使用者名稱和設定檔資訊 CodeCatalyst。

支援身分聯合的空間 CodeCatalyst

已新增至 IAM 身分中心執行個體的 SSO 使用者和群組，並在身分識別存放區中進行管理，並透過 IAM 身分中心邀請加入您的空間的使用者。S pace 管理員會同步 CodeCatalyst 成員頁面以取得最新的更新。使用者可使用在公司 IAM 身分中心執行個體中設定的 SSO 登入入口網站登入。支援身分識別聯合的空間會透過 Identity Center 應用程式及其對應至識別身分存放區 ID 連線至識別身分存放區執行個體。

專案

專案代表中 CodeCatalyst 支援開發團隊和任務的協同合作工作。建立專案後，您可以新增、更新或移除使用者和資源、自訂專案儀表板，以及監控團隊工作進度。您可以在一個空間中擁有多個專案。

如需專案的更多資訊，請參閱中的專案 CodeCatalyst。

Blueprints (藍圖)

藍圖是一種專案合成器，可為您產生並擴充應用程式支援檔案和相依性，並在主控台中建立 CodeCatalyst 專案。您可以從中選取的藍圖中選擇專案類型 CodeCatalyst、檢視 README 檔案，並預覽將要產生的專案存放庫和資源。您的專案是從藍圖指定的基本組態產生的。您可以定期合成專案藍圖，這樣會更新專案檔案，例如軟體相依性，並重新產生資源。專案使用名為 Projen 的工具，透過同步最新的專案更新並產生支援檔案來合成專案。根據您的應用程式類型和語言 `package.json` `Makefileeslint`，這些檔案可能包含、、及其他檔案。專案藍圖可以產生支援 AWS 資源的檔案，例如 CDK 建構、 AWS CloudFormation 範本和範本。 AWS Serverless Application Model

如需專案藍圖的詳細資訊，請參閱專案藍圖參考。

帳戶連線

帳戶連線會將 CodeCatalyst 空間與您的 AWS 帳戶. 設定帳戶連線後，即可供空間使用。 AWS 帳戶 然後，您可以將 IAM 角色新增到，以 CodeCatalyst 便它可以存取 AWS 帳戶. 您也可以將這些角色用於 CodeCatalyst 工作流程動作。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。

VPC 端連

VPC 連線是一種 CodeCatalyst 資源，其中包含存取 VPC 工作流程所需的所有組態。空間管理員可以代表空間成員在 Amazon CodeCatalyst 主控台中新增自己的 VPC 連線。透過新增虛擬私人雲端連線，Space 成員可以執行工作流程動作，並建立遵守網路規則並可存取相關 VPC 中的資源的開發環境。

如需 VPC 連線的詳細資訊，請參閱《管理CodeCatalyst 員指南》中的「管理 Amazon 虛擬私有雲端」。

AWS 產生器 ID

AWS Builder ID 是一種個人身分，您可以用來註冊和登入以 CodeCatalyst 及其他參與的應用程式。它是不一樣的 AWS 帳戶. 您的 AWS Builder ID 管理中繼資料，例如使用者別名和電子郵件地址 您的 AWS 產生器 ID 是唯一的身分，可支援中所有空間的使用者 CodeCatalyst。如需存取 AWS 產生器 ID 設定檔的相關資訊，請參閱更新您的設定檔。若要深入瞭解 AWS 產生器 ID，請參閱中的AWS 產生器 ID AWS 一般參考。

如需註冊和登入的詳細資訊，請參閱正在設定 CodeCatalyst。

使用者設定檔 CodeCatalyst

您可以從中 CodeCatalyst任何頁面的登入縮寫下方的下拉式清單中選擇設定檔選項，以存取您的 CodeCatalyst 使用者設定檔。您可以從設定檔頁面建立個人存取權杖 (PAT)，但您只能使用. AWS CLI 您的使用者名稱是您註冊時選擇的別名。您無法變更您的使用者名稱。要查看其他 CodeCatalyst 用戶的個人資料頁面，請轉到項目的「成員」標籤，然後選擇適當的用戶。

您可以檢視您的 AWS 設 CodeCatalyst 定檔，然後選擇前往產生器 ID 來存取您的 AWS Builder ID。您將被重定向到您的 AWS Builder ID 個人資料頁面。您的個人檔案的全名、電子郵件地址和密碼由您的 AWS Builder ID 管理，您可以使用 AWS Builder ID 頁面編輯該資訊。您在註冊時輸入了此資訊。當您準備好將 MFA 設定為使用驗證器應用程式進行登入時，您將使用「 AWS 產生器 ID」頁面。如需有關檢視 AWS 產生器 ID 設定檔的詳細資訊，請參閱更新您的設定檔。

如需註冊和登入的詳細資訊，請參閱正在設定 CodeCatalyst。

來源儲存庫

源存儲庫是您安全地存儲項目的代碼和文件的地方。它還存儲文件的版本歷史記錄。默認情況下，源存儲庫與 CodeCatalyst 項目中的其他用戶共享。您可以為一個項目擁有多個源存儲庫。您可以為中的專

案建立來源儲存庫 CodeCatalyst，或者如果已安裝的擴充功能支援該服務，您也可以選擇連結由其他服務託管的現有來源儲存庫。例如，您可以在安裝儲 GitHub 存庫擴充功能之後，將存放GitHub 庫連結至專案。如需詳細資訊，請參閱 使用中的來源儲存庫 CodeCatalyst 及 快速入門：在中使用 GitHub 儲存庫 CodeCatalyst。

來源儲存庫也是儲存 CodeCatalyst專案組態資訊的位置，例如定義 CI/CD 工作流程屬性和動作的組態檔案。如果您使用藍圖建立專案，則會建立一個來源存放庫，其中儲存了專案組態資訊。如果您建立空專案，則必須先建立來源儲存庫，才能建立需要設定資訊的資源，例如工作流程。

如需可協助您使用原始碼儲存庫和原始檔控制的更多概念，請參閱來源儲存庫概念。

遞交

提交是對一個文件或一組文件的更改。在 Amazon 主 CodeCatalyst控台中，提交會儲存您的變更，並將其推送至來源儲存庫。提交包含有關變更的資訊，包括進行變更的使用者身分、變更的時間和日期、提交標題，以及任何包含有關變更的訊息。如需詳細資訊，請參閱 在 Amazon 中處理提交 CodeCatalyst。

在中的來源儲存庫環境中 CodeCatalyst，認可是儲存庫內容變更的快照。每次使用者提交並推送變更時，都會 CodeCatalyst 儲存資訊，其中包括提交變更者、提交的日期和時間，以及作為提交一部分所做的變更。您還可以將 Git 標籤添加到提交中，以幫助識別特定的提交。

如需有關提交的詳細資訊，請參閱在 Amazon 中處理提交 CodeCatalyst。

開發環境

開發環境是一種雲端式開發環境，您可以使用它 CodeCatalyst 來快速處理儲存在專案原始碼儲存庫中的程式碼。開發環境中包含的項目工具和應用程序庫由項目源存儲庫中的 devfile 定義。如果您的源存儲庫中沒有 devfile，則會自動應用默認的 devfile。默認的 devfile 包括最常用的編程語言和框架的工具。根據預設，開發環境設定為具有 2 核心處理器、4 GB 的 RAM 和 16 GiB 的持續性儲存裝置。

工作流程

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要執行的一系列步驟或動作。工作流程也會定義導致工作流程啟動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

如需工作流程的相關詳細資訊，請參閱 使用中的工作流程來建置、測試和部署 CodeCatalyst。

動作

動作是工作流程的主要建置區塊，可定義工作流程執行期間要執行的邏輯工作單元。一般而言，工作流程包含多個依序執行或 parallel 執行的動作，具體取決於您設定它們的方式。

如需有關動作的更多資訊，請參閱使用動作。

問題

問題是追蹤與專案相關工作的記錄。您可以針對與專案相關的功能、工作、錯誤或任何其他工作主體建立問題。如果您正在使用敏捷開發，則問題還可以描述史詩或用戶故事。

如需問題的詳細資訊，請參閱中的問題 CodeCatalyst。

個人存取權杖 (PAT)

個人存取權杖 (PAT) 類似於密碼。它與您的使用者身分相關聯，以便在中的所有空間和專案中使用 CodeCatalyst。您可以使用 PAT 來存取包含整合式開發環境 (IDE) 和 Git 型來 CodeCatalyst 源存放庫的資源。PAT 代表您，您可 CodeCatalyst 以在使用者設定中管理它們。使用者可以擁有多個 PAT。個人存取權杖只會顯示一次。最佳做法是，請務必將它們安全地儲存在本機電腦上。依預設，PAT 會在一年後過期。

如需 PAT 的詳細資訊，請參閱在 Amazon 中管理個人訪問令牌 CodeCatalyst。

角色

角色定義使用者對專案或空間資源的存取權限，以及使用者可以執行的動作。當您邀請使用者加入專案時，您可以選擇使用者的角色。中有空間層級角色和專案層級角色。 CodeCatalyst具有正確層級管理角色的使用者可以變更指派的角色。例如，具有專案專案管理員角色的使用者可以完全控制該專案，

而且可以變更該專案中使用者的角色。如需有關哪些角色可用以及每個角色具有哪些權限的資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

正在設定 CodeCatalyst

您可以在兩種類型的空間中設定 CodeCatalyst：支援 AWS Builder ID 使用者的空間，以及建立支援身分聯合的空間，其中 SSO 使用者和群組在 IAM 身分中心管理。 AWS Builder ID 空間中的使用者使 CodeCatalyst 用其 AWS 產生器 ID 登入，而企業空間中的使用者則 CodeCatalyst 使用與空間相關聯之公司的 SSO 入口網站登入。

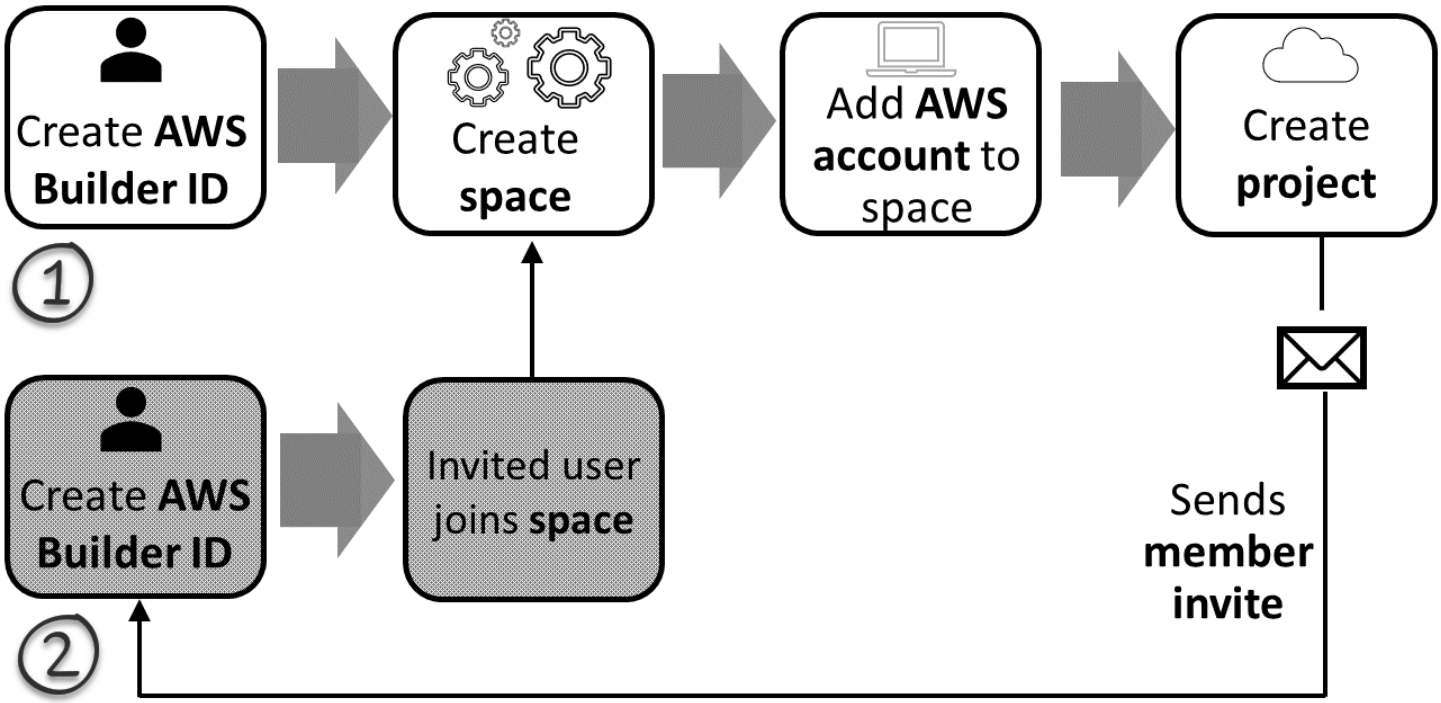
本指南提供了設定和管理 AWS 產生器 ID 空間的步驟。若要使用 Bu CodeCatalyst AWS ilder ID 空間，您將 CodeCatalyst 使用用於登入的使用者設定和 AWS 產生器 ID 進行設定 CodeCatalyst。

《管理CodeCatalyst 員指南》中提供了設定和管理支援身份聯合之空間的步驟。若要使用為聯合身分設定的空間，請參閱 Amazon CodeCatalyst 管理員指南中的 CodeCatalyst 空間設定和管理。

本節提供兩種通用途徑，讓您在 Amazon 中使 CodeCatalyst 用 AWS Builder ID 空間：以第一位使用者身分建立空間和專案，以及接受現有空間或專案的邀請。這些設置工作流程必然完全不同。下圖顯示了兩個註冊過程如下：

在 Amazon 中設定工作的通用途徑有兩種 CodeCatalyst：以第一位使用者身分建立空間和專案，以及接受現有空間或專案的邀請。這些設置工作流程必然完全不同。下圖顯示了兩個註冊過程如下：

1. 在第一種情況下，您要為公司、團隊或群組建立並設定 Space，並在邀請其他人使用這些資源之前建立專案。 AWS 帳戶 必須提供帳單用途，您仍然可以預設使用免費方案。
2. 在第二種情況下，如果您透 CodeCatalyst 過接受專案邀請加入，則其他人已經為您建立了空間和專案。但是，您仍然需要配置您的個人資料，以便準備開始與其他人合作。



> **Tip**
>
> CodeCatalyst 使用空間來分組專案和資源。首次註冊時 CodeCatalyst，系統會提示您建立空間和專案。

無論您是註冊建立空間和專案，還是註冊成為接受邀請的一部分，您都可以建立用來登入的 AWS Builder ID CodeCatalyst。若要建立 AWS 產生器 ID，請提供用來登入 AWS 應用程式的全名、密碼和電子郵件地址。在此 CodeCatalyst 之後，您可以使用電子郵件和密碼登入。您也可以使用此 AWS 產生器 ID 登入使用產 AWS 生器 ID 認證的其他應用程式。

在產 CodeCatalyst 生 AWS 器 ID 中，系統會根據您的登入資訊產生描述檔。您的個人檔案包含您 CodeCatalyst 專案中語言和通知設定的 CodeCatalyst 偏好設定。

> **Tip**
>
> 如果您在註冊 Amazon 個 CodeCatalyst 人檔案時遇到任何問題，請按照該頁面上提供的步驟進行操作。如果您需要其他協助，請參閱註冊時遇到問題。

主題

註冊以創建您的第一個空間和您的發展角色

您可以註冊 Amazon， CodeCatalyst 而無需邀請到現有空間或項目。當您這樣做時，您將在建立AWS產生器 ID 之後建立空間和專案。在建立空間時，您需要新增空間以AWS 帳戶進行計費。

> **Tip**
>
> 如果您在註冊 Amazon 個 CodeCatalyst 人檔案時遇到任何問題，請按照該頁面上提供的步驟進行操作。如果您需要其他協助，請參閱註冊時遇到問題。

對於在沒有項目或空間邀請的 CodeCatalyst 情況下開始的用戶，這是一個可能的流程。

Mary Major 是誰感興趣， CodeCatalyst 並決定嘗試一下開發人員。她導航到 CodeCatalyst 控制台，並選擇註冊和創建AWS生成器 ID 的選項。瑪麗提供了一個電子郵件地址和密碼來創建她的AWS生成器 ID。她將能夠使用自己的AWS生成器 ID 來登錄 CodeCatalyst 和其他應用程序。當被要求選擇別名時，她指定`MaryMajor`為將顯示在中的用 CodeCatalyst 戶名 CodeCatalyst ，其他項目成員將用於@mention Mary。

接下來，瑪麗將自動定向以創建空間。作為此流程的一部分，Mary 被要求AWS 帳戶與她正在創建的空間相關聯，以便她可以在第一個項目構建和部署中查看示例代碼。她會新增這些資訊並建立她的空間，並在其中選擇建立可用於新空間中專案的預覽開發角色的選項。Mary 選擇建立專案，然後檢視專案藍圖清單。在檢閱可用藍圖的資訊之後，她決定嘗試第一個專案的 Modern 三層 Web 應用程式藍圖。她填寫必填字段并創建項目。一旦專案準備就緒，她就會前往專案摘要頁面，其中包含最近的活動以及專案程式碼的連結，以及自動建置和部署該程式碼的工作流程。她探索程式碼和工作流程，包括檢

視已部署的範例 Web 應用程式。喜歡她所看到的東西，她決定邀請她的一些同事參加該項目開始探索 CodeCatalyst。

當她有片刻時，瑪麗配置她的AWS生成器 ID 以 CodeCatalyst 使用多因素身份驗證（MFA）登錄。配置 MFA 後，Mary 可以 CodeCatalyst 使用其 CodeCatalyst 密碼和來自已批准的第三方身份驗證應用程序的密碼或令牌來登錄。

建立您的第一個空間和 IAM 角色

請按照以下步驟註冊您的 Amazon 個人 CodeCatalyst 檔案、建立空間，以及為您的空間新增帳戶、支援角色和開發人員角色。

最後一個程序會建立並新增開發人員角色。開發人員角色是 AWS IAM 角色，可讓您的 CodeCatalyst 工作流程存取AWS資源。開發人員角色是用於管理AWS 服務的服務角色，將在已登入的帳戶中建立。服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。該角色將具有一個名稱`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。

> **Note**
>
> 作為安全性最佳實務，請僅將管理存取權指派給需要管理空間中AWS資源存取權的管理使用者和開發人員。

在開始之前，您必須準備好為具有管理權限的帳戶提供 AWS 帳戶 ID。準備好您的 12 位數 AWS 帳戶 ID。如需尋找 AWS 帳戶 ID 的相關資訊，請參閱您的 AWS 帳戶 ID 及其別名。

註冊為新使用者

1. 在您開始使用 CodeCatalyst 主控台之前AWS Management Console，請先開啟，然後確定您已使用您要用來建立空間的相同AWS 帳戶方式登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 在歡迎頁面上，選擇 [註冊]。[建立您的AWS產生器 ID] 頁面隨即顯示。您的AWS產生器 ID 是您建立用來登入的身分。它是不一樣的AWS 帳戶.
4. 在 [您的電子郵件地址] 中，輸入您要建立關聯的電子郵件地址 CodeCatalyst。然後選擇下一步。
5. 在 [您的名稱] 中，提供您要在使用 AWS Builder ID 的應用程式中顯示的名字和姓氏。允許使用空格。這將是您的AWS生成器 ID 配置文件名稱，例如瑪麗大專。您可以稍後變更名稱。

選擇下一步。電子郵件驗證頁面隨即顯示。

6. 驗證碼將發送到您指定的電子郵件地址。在 [驗證碼] 中輸入此驗證碼，然後選擇 [驗證]。如果您在 5 分鐘後仍未收到驗證碼，但在垃圾郵件或垃圾郵件資料夾中找不到驗證碼，請選擇「重新傳送驗證碼」。
7. 一旦我們驗證您的代碼，請在「密碼」和「確認密碼」中輸入符合要求的密碼。

   勾選確認您與「AWS客戶協議」與「AWS服務條款」之間的協議的核取方塊，然後選擇「建立AWS產生器 ID」。
8. 在 [建立 CodeCatalyst 別名] 頁面上，輸入您要在其中用於唯一使用者識別碼的別名CodeCatalyst。選擇不含空格的名稱縮短版本，例如MaryMajor。其他用 CodeCatalyst 戶將使用它來 @mention 您在評論和提取請求中。您的個 CodeCatalyst 人資料將包含您的 AWS Builder ID 中的全名和 CodeCatalyst 別名。您之後無法變更 CodeCatalyst 別名。

   您的全名和別名將顯示在中的不同區域 CodeCatalyst。例如，您的設定檔名稱會顯示在活動摘要中列出的活動，但專案成員會使用您的別名來 @mention 您。

   選擇下一步。頁面會更新以顯示 [建立您的 CodeCatalyst 空間] 區段。
9. 在命名您的空間中，輸入空間的名稱。您之後無法變更此設定。

   > **Note**
   >
   > 空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

10. 在AWS 區域下拉式選單中，選擇您要儲存空間和專案資料的區域。您之後無法變更此設定。
11. 選擇下一步。頁面會更新，以顯示要新增的頁面AWS 帳戶。此帳戶將用作該空間的帳單帳戶。
12. 在 AWS 帳戶ID 中，輸入您要連接到空間的帳戶的十二位數 ID。

    在AWS帳戶驗證令牌中，複製生成的令牌 ID。Token 會自動為您複製，但您可能想要在核准AWS連線要求時儲存它。
13. 選擇 [前往AWS主控台進行驗證]。
14. 「驗證 Amazon CodeCatalyst 空間」頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登入才能存取此頁面。

    在中AWS Management Console，請務必選擇您要建立空間的相同AWS 區域位置。

    要直接訪問該頁面，請登錄到 Amazon CodeCatalyst 空間AWS Management Console在 https://console.aws.amazon.com/codecatalyst/home/.

中的驗證權杖欄AWS Management Console位會自動填入中產生的權杖 CodeCatalyst。

15. (選擇性) 在「授權付費方案」下，選擇「授權付費方案」(標準、企業)，為您的帳單帳戶開啟付費方案。

> **Note**
>
> 這不會將計費層級升級為付費層級。不過，這會設定，以AWS 帳戶便您可以在中隨時變更空間的計費層級。 CodeCatalyst您可以隨時開啟付費方案。如果不進行此變更，空間只能使用免費方案。

16. 選擇驗證空間。

    會顯示「帳戶驗證」成功訊息，顯示帳戶已新增至空間。

17. 保留在驗證 Amazon CodeCatalyst 空間頁面上。選擇下列連結：若要為此空間新增 IAM 角色，請檢視空間詳細資訊。

    包含CodeCatalyst 空間詳細資訊的連線頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。

18. 返回 CodeCatalyst 頁面，然後選擇 [下一步]。

19. 建立您的空間時，會顯示狀態訊息。建立空間後，會顯示以 CodeCatalyst 下訊息：您的空間已準備就緒。您的最後一步是創建一個項目。 。您可以執行下列任一作業：

    - 現在選擇「跳過」。
    - 選擇為您的空間建立第一個專案。如需展示如何使用藍圖建立專案的教學課程，請參閱 教學課程：使用現代三層 Web 應用程式藍圖建立專案

> **Note**
>
> 如果顯示權限錯誤或橫幅，請重新整理頁面並嘗試再次檢視頁面。

若要建立並新增 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-***spaceName***

1. 在 CodeCatalyst 主控台中啟動之前，請開啟AWS Management Console，然後確定您已使用相同 AWS 帳戶的空間登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 選擇您要建立角色之AWS 帳戶位置的連結。AWS 帳戶詳細資訊頁面隨即顯示。
5. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。
6. 選擇在 IAM 中建立 CodeCatalyst 開發管理員角色。此選項會建立包含開發角色之權限原則和信任原則的服務角色。該角色將具有一個名稱`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***`spaceName`***務角色。

   > **Note**
   >
   > 此角色僅建議與開發人員帳戶搭配使用，並使用`AdministratorAccess`AWS受管理的政策，讓其具有完整存取權，以便在其中建立新政策和資源AWS 帳戶。
7. 選擇 [建立開發角色]。
8. 在「連線」頁面的「可用的 IAM 角色」下 CodeCatalyst**`CodeCatalystWorkflowDevelopmentRole-`*****`spaceName`***，檢視新增至您帳戶的 IAM 角色清單中的角色。
9. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

若要建立並新增 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在 CodeCatalyst 主控台中啟動之前，請開啟AWS Management Console，然後確定您已使用相同 AWS 帳戶的空間登入。
2. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
3. 選擇您要建立角色之AWS 帳戶位置的連結。AWS 帳戶詳細資訊頁面隨即顯示。
4. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登入才能存取此頁面。
5. 在CodeCatalyst 空間詳細資料下，選擇新增 S CodeCatalyst upport 角色。此選項會建立包含預覽開發角色的權限原則和信任原則的服務角色。該角色將具有附加唯AWSRoleForCodeCatalystSupport一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服AWSRoleForCodeCatalystSupport務角色。

6. 在 [新增 Sup CodeCatalyst port 角色] 頁面上，保持選取的預設值，然後選擇 [建立角色]。
7. 在「可用的 IAM 角色」下 CodeCatalyst**CodeCatalystWorkflowDevelopmentRole-*spaceName***，檢視新增至您帳戶的 IAM 角色清單中的角色。
8. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

建立 AWS Builder ID、建立第一個空間並新增帳戶後，您就可以建立專案。如需詳細資訊，請參閱 在 Amazon 創建一個項目 CodeCatalyst。如果這是您第一次使用 CodeCatalyst，我們建議您從教學課程：使用現代三層 Web 應用程式藍圖建立專案.

接受邀請並建立您的AWS產生器 ID

您可以註冊 Amazon CodeCatalyst 作為接受專案或空間邀請的一部分。在接受邀請的過程中，系統會提示您建立AWS產生器 ID。您將使用您的AWS產生器 ID 來存取中的資源 CodeCatalyst。

> **Tip**
>
> 如果您需要其他協助，請參閱註冊時遇到問題。

以下是一個可能的流程，讓使用者從專案或空間的邀請開始。 CodeCatalyst

Sanvi Sarkar 是一位開發人員，已收到邀請以專案管理員身分加入 CodeCatalyst 專案。Sanvi 接受邀請，該邀請會打開的登錄頁面。 CodeCatalyst她選擇註冊並提供電子郵件地址和密碼來創建她的 AWS Builder ID。Sanvi 將能夠使用她的AWS生成器 ID 來登錄 CodeCatalyst 和其他應用程序。之後，她可以編輯個人資料以變更登入電子郵件地址或密碼。當系統要求選擇別名時，Sanvi 會指定`SaanviSarkar`為將顯示在其中的 CodeCatalyst 別名， CodeCatalyst 而其他專案成員將用於 @mention Sanvi。註冊後，Sanvi 還將能夠將其登錄憑據用於使用生AWS成器 ID 憑據的其他應用程序。

完成註冊後，Sanvi 會自動加入邀請中指定的 CodeCatalyst 項目和空間。邀請還為她在專案和空間中的角色提供預先確定的權限。在專案設定中，Sanvi 的別名會以她指派的專案角色顯示在成員清單中。若要使用中的來源儲存庫 CodeCatalyst，Sanvi 需要一點時間來建立個人存取權杖 (PAT)。進行來源變更或需要驗 CodeCatalyst 證 Token 的動作時，將在中使用 PAT 進行驗證。

當 Saanvi 在專案上工作時，她的別名會列在專案的工作活動記錄中。Sanvi 的問題和評論將顯示她的別名，其他項目成員可以在其中 @mention 她回復。若要 @mention 另一個專案成員，Sanvi 會在他們的個人檔案中查找他們的別名。 CodeCatalyst

當她有片刻時，Sanvi 會配置她的AWS生成器 ID 以 CodeCatalyst 使用多因素身份驗證（MFA）登錄。配置 MFA 後，Sanvi 可以 CodeCatalyst 使用她的密碼和來自已批准的第三方身份驗證應用程序的 CodeCatalyst 密碼或令牌的組合登錄。

接受邀請並建立AWS產生器 ID

當您受邀加入 Amazon 的專案或空間時 CodeCatalyst，您會收到來自 notify@codecatalyst.aws 的電子郵件，要求您接受邀請。如果您已經有 AWS Builder ID 且已登入 CodeCatalyst，選擇「接受邀請」會自動開啟瀏覽器索引標籤中的專案或空間。如果您尚未登入主機，但擁有 AWS Builder ID，系統會將您導向登入頁面。如需詳細資訊，請參閱使用您的AWS生成器 ID 登錄。

如果您沒有建置AWS器 ID，選擇「接受邀請」會帶您前往登入頁面，您應該在其中選擇建立 AWS Builder ID 的選項。

接受邀請並建立AWS產生器 ID

1. 在邀請電子郵件中，選擇「接受邀請」。
2. 在「登入」頁面上，選擇「未註冊嗎？」 創建您的AWS生成器 ID。

   > **Tip**
   >
   > 您的AWS產生器 ID 是您建立用來登入的身分。它是不一樣的AWS 帳戶.

3. 在 [建立您的AWS產生器 ID] 頁面的 [電子郵件地址] 中，輸入您要用於 AWS Builder ID 的電子郵件地址。

   在 [您的名稱] 中，提供您要在使用 AWS Builder ID 的應用程式中顯示的名字和姓氏。允許使用空格。這將是您的AWS生成器 ID 配置文件名稱，例如瑪麗大專。您可以稍後變更名稱。

   選擇下一步。

   驗證碼將發送到您指定的電子郵件。在 [驗證碼] 中輸入此驗證碼，然後選擇 [驗證]。如果您在 5 分鐘後仍未收到驗證碼，但在垃圾郵件或垃圾郵件資料夾中找不到驗證碼，請選擇「重新傳送驗證碼」。
4. 驗證您的代碼後，請輸入符合「密碼」和「確認密碼」中要求的密碼。
5. 選擇「建立AWS產生器 ID」。
6. 在 [建立別名] 頁面上，輸入您要在其中用於唯一使用者識別碼的別名 CodeCatalyst。選擇不含空格的名稱縮短版本，例如MaryMajor。其他用 CodeCatalyst 戶將使用它來 @mention 您在評論和

提取請求中。您的個 CodeCatalyst 人資料將包含您的 AWS Builder ID 中的全名和 CodeCatalyst 別名。您無法變更 CodeCatalyst 別名。

您的全名和別名將顯示在中的不同區域 CodeCatalyst。例如，您的設定檔名稱會顯示在活動摘要中列出的活動，但專案成員會使用您的別名來 @mention 您。

選擇 [建立別名]。您將被帶到受邀參加的項目或空間。

使用您的AWS生成器 ID 登錄

請按照以下步驟登錄到您的 Amazon CodeCatalyst 個人資料。

> **Note**
>
> 您是否已為多因素身份驗證（MFA）註冊了設備？ 我們強烈建議您在 Amazon 中設定 MFA CodeCatalyst 以提高安全性。如需詳細資訊，請參閱如何註冊設備以使用多因素身份驗證。

使用您的AWS產生器 ID 登入

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 輸入您的電子郵件地址。或者，如果您想要儲存電子郵件地址以供 future 登入使用，請選擇 [儲存我的電子郵件地址]。選擇 Continue (繼續)。
3. 輸入您的密碼。選擇 Sign In (登入)。如果您忘記密碼，請按照中的步驟操作我忘記了密碼。

信任的裝置

在您從登入頁面選擇 [這是受信任的裝置] 選項後，Amazon CodeCatalyst 會將該裝置的所有 future 登入都視為已授權。只要您使用該受信任的裝置，Amazon 就不 CodeCatalyst 會提供輸入 MFA 代碼的選項。某些例外情況包括從新的瀏覽器登入，或當您的裝置發出未知的 IP 位址時。

接受使用 SSO 登入的電子郵件邀請

新增至支援身分聯盟之空間的使用者會收到包含登入入口網站連結和設定資訊的電子郵件。請使用下列步驟接受邀請並使用 SSO 登入。

若要改用您的AWS產生器 ID 登入，請參閱使用您的AWS生成器 ID 登錄。

使用 SSO 接受並登入

1. 選擇電子郵件中接受請求的按鈕。使用電子郵件中提供的連結，前往與您公司相關聯之空間的登入入口網站。
2. 在使用者名稱和密碼中，輸入您的認證。
3. 選擇 Sign In (登入)。

使用 SSO 登入

請按照以下步驟使用 SSO 登錄 Amazon CodeCatalyst。

若要改用您的AWS產生器 ID 登入，請參閱使用您的AWS生成器 ID 登錄。

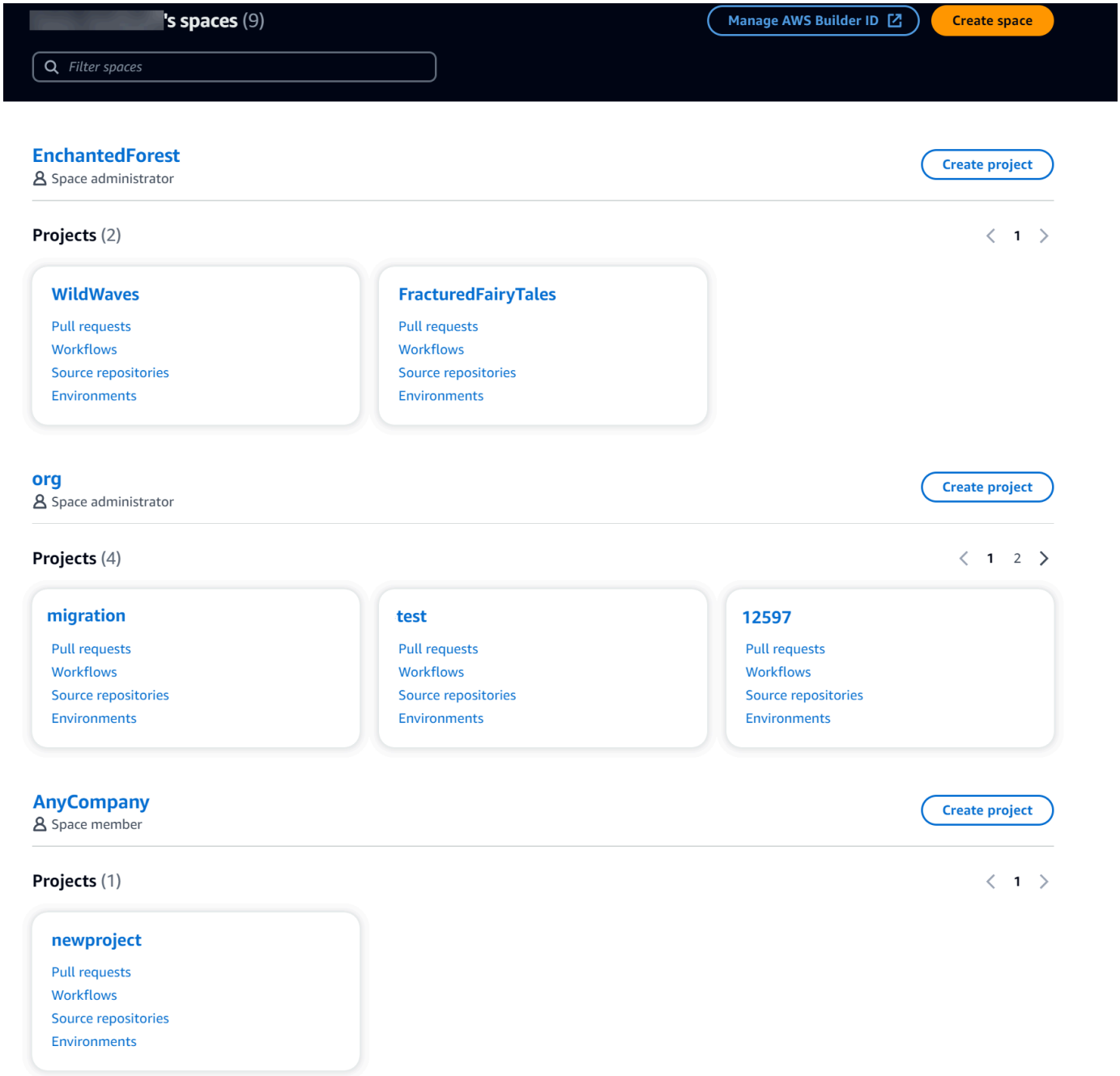
使用 SSO 登入

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在 [選擇登入選項] 底下，選擇 [使用單一登入 (SSO)]。
3. 在AWS識別中心應用程式名稱中，輸入身分識別同盟管理員提供的應用程式名稱。
4. 選擇繼續前往 IAM 身分中心。

檢視使用者的所有空間和專案

您可以在使用者首頁上檢視您的空間和專案清單。使用者首頁會顯示使用者所屬之每個空間的清單、該空間中的使用者角色 (例如 Space 管理員)，以及使用者擁有成員資格的每個空間中的專案。

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在瀏覽器中，輸入以下地址：https://codecatalyst.aws/home



3. 選擇您要開啟的空間或專案。如果您沒有看到預期看到的空間或專案，您可能需要以其他使用者身分登入。

檢視和管理 CodeCatalyst 設定檔

您可以在 Amazon 中檢視使用者設定檔， CodeCatalyst 以取得電子郵件地址和 CodeCatalyst 別名等資訊。您也可以更新您的個人資料和AWS生成器 ID。如果您忘記密碼，可以要求重設密碼。

檢視您的 CodeCatalyst 設定檔

您在註冊時提供的信息將用作登錄到 Amazon 的憑據， CodeCatalyst 並在您的個人資料中管理。這包括您的姓名、暱稱，以及您用來登入的電子郵件地址 CodeCatalyst。

> **Note**
>
> 建AWS立者 ID 暱稱不是您的 CodeCatalyst 別名。您在註冊時選擇了 CodeCatalyst 別名。

若要檢視您的 CodeCatalyst 設定檔

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [我的設定]。 CodeCatalyst 我的設定」頁面隨即開啟。
3. 若要更新您的建置AWS器 ID 電子郵件地址或密碼，或設定 MFA，請選擇管理AWS建置器 ID。「AWS產生器 ID」頁面隨即開啟。

檢視其他使用者的 CodeCatalyst設定檔

若要檢視其他使用者的 CodeCatalyst 設定檔

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在側邊導覽中，選擇 [專案設定]。選擇「成員」頁標。檢視 CodeCatalyst專案的成員清單。
3. 選擇您要查詢的成員名稱或 @mention。「我的設定」 頁面會顯示使用者的別名、電子郵件地址和全名。將 CodeCatalyst 別名用於 @mention 專案成員。

   > **Note**
   >
   > 使用者的AWS建立者 ID 暱稱不是他們的 CodeCatalyst別名。他們在註冊時選擇了他們的 CodeCatalyst 別名。

   若要檢視專案中其他使用者的設定檔，請在清單中選擇他們的名稱。

更新您的設定檔

在中 CodeCatalyst，您的設定檔包含由 AWSBuilder ID 管理的個人資訊，以及中管理的設定 CodeCatalyst。

- 您設定檔的全名、電子郵件地址和密碼由 AWSBuilder ID 管理。您在註冊時輸入了此資訊。當您將 MFA 設定為使用驗證器應用程式進行應用程式登入時，會 CodeCatalyst 帶您前往AWS產生器 ID 頁面。
- CodeCatalyst 您的個人存取權杖 (PAT)、 CodeCatalyst 通知和語言偏好設定的設定可在中的 「我的設定」 頁面中管理 CodeCatalyst。如需詳細資訊，請參閱在 Amazon 中管理個人訪問令牌 CodeCatalyst。

> **Note**
>
> 您可以更新您的AWS產生器 ID 全名 (CodeCatalyst 顯示名稱) 和名字。但是，您無法變更 CodeCatalyst 別名。

更新您的AWS生成器 ID 或電子郵件地址

更新您的AWS 建構家 ID或電子郵件地址

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [我的設定]。 CodeCatalyst我的設定」頁面隨即開啟。
3. 在設定檔頁面上，選擇「管理AWS建置器 ID」。AWS生成器 ID 頁面打開。
4. 在頁面左側，選擇 [我的詳細資料]。
5. 在「設定檔資訊」下，選擇「編輯」 以更新您的姓名或暱稱。如果您沒有指定暱稱，暱稱欄位會反映全名中的名字。這不是你的 CodeCatalyst 別名。

   > **Note**
   >
   > 這會更新AWS產生器 ID 的完整名稱和名字。這不會更新您的 CodeCatalyst 別名。

   在 [聯絡資訊] 下方，選擇 [編輯] 以更新您的電子郵件

> **Note**
>
> 這會更新您用來登入的電子郵件地址 CodeCatalyst。

變更您的 CodeCatalyst 密碼

若要變更您的 CodeCatalyst 密碼

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [使用者設定檔]。 CodeCatalyst 我的設定」 頁面隨即開啟。
3. 在設定檔頁面上，選擇「管理AWS建置器 ID」。「AWS產生器 ID」頁面隨即開啟。
4. 在頁面左側，選擇 [安全性]。
5. 選擇變更密碼，然後依照指示操作。

設定以使用AWS CLI與 CodeCatalyst

Amazon 主 CodeCatalyst 控台是您處理大部分日常任務的地方。但是，您可能需要在中AWS CLI設置和配置開發環境，個人訪問令牌或事件日誌 CodeCatalyst。您必須先安裝AWS CLI並設定設定檔，才能與配合使用 CodeCatalyst。

若要設定用AWS CLI於 CodeCatalyst

1. 安裝最新版本的AWS CLI. 如果您已經AWS CLI安裝了版本，請確定它是最新版本並包含的指令 CodeCatalyst，並在需要時更新。若要確認您安裝的版本包含命 CodeCatalyst 令，請開啟命令提示字元並執行下列命令：

   ```
   aws codecatalyst help
   ```

   如果您看到 CodeCatalyst 指令清單，表示您有支援的版本 CodeCatalyst。如果無法辨識指令，請將您的版本更新AWS CLI為最新版本。若要取得更多資訊，請參閱《使用指南》AWS CLI中的〈安裝或更新最新版本的AWS Command Line Interface〉。

2. 如果您沒有設定檔，或想要使用專門用於的具名設定檔，請執行aws configure指令來建立設定檔CodeCatalyst。建議您建立具名的設定檔以特別搭配使用 CodeCatalyst，但您也可以使用預設紀要。如需詳細資訊，請參閱組態基本知識。
3. 編輯配置config文件的文件以添加要連接的部分， CodeCatalyst 如下所示。config 檔案在 Linux 或 macOS 上位於 ~/.aws/config，在 Windows 上位於 C:\Users*USERNAME*\.aws\config。

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

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

4. 儲存檔案。
5. 嘗試執行任何命 CodeCatalyst 令之前，請開啟新的終端機或命令提示字元，然後執行下列命令來要求並擷取認證以執行命aws codecatalyst令。如codecatalyst有需要，請以您的設定檔名稱取代。

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

若要檢視codecatalyst命令範例，請參閱下列主題：

- 在 Amazon 中管理個人訪問令牌 CodeCatalyst
- 存取記錄的事件 CodeCatalyst

入門教學課程

Amazon CodeCatalyst 提供許多不同的範本來協助您開始使用專案。您也可以選擇從空專案開始，然後在其中加入資源。請遵循這些自學課程中的步驟，瞭解您可以使用的一些方式 CodeCatalyst。

如果這是您第一次使用 CodeCatalyst，我們建議您從教學課程：使用現代三層 Web 應用程式藍圖建立專案.

> **Note**
>
> 若要遵循這些自學課程，您必須先完成設定。如需詳細資訊，請參閱 正在設定 CodeCatalyst。

主題

- 教學課程：使用現代三層 Web 應用程式藍圖建立專案
- 教學課程：從空白專案開始，並手動新增資源
- 教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作
- 教學課程：建立包含可組合式 PDK 藍圖的完整堆疊應用程式

如需專注於特定功能區域的其他教學課程 CodeCatalyst，請參閱：

- 開始使用 Slack 通知
- 開始使用 CodeCatalyst 來源儲存庫和單頁應用程式藍圖
- 開始使用中的工作流程 CodeCatalyst
- 開始使用自訂藍圖
- 開始使用 Amazon 動 CodeCatalyst 作開發人員指南

如需深入教學課程，請參閱：

- 教學課程：將成品上傳到 Amazon S3
- 教學課程：使用部署無伺服器應用程式 AWS CloudFormation
- 教學課程：將應用程式部署到 Amazon ECS
- 教學課程：將應用程式部署到 Amazon EKS

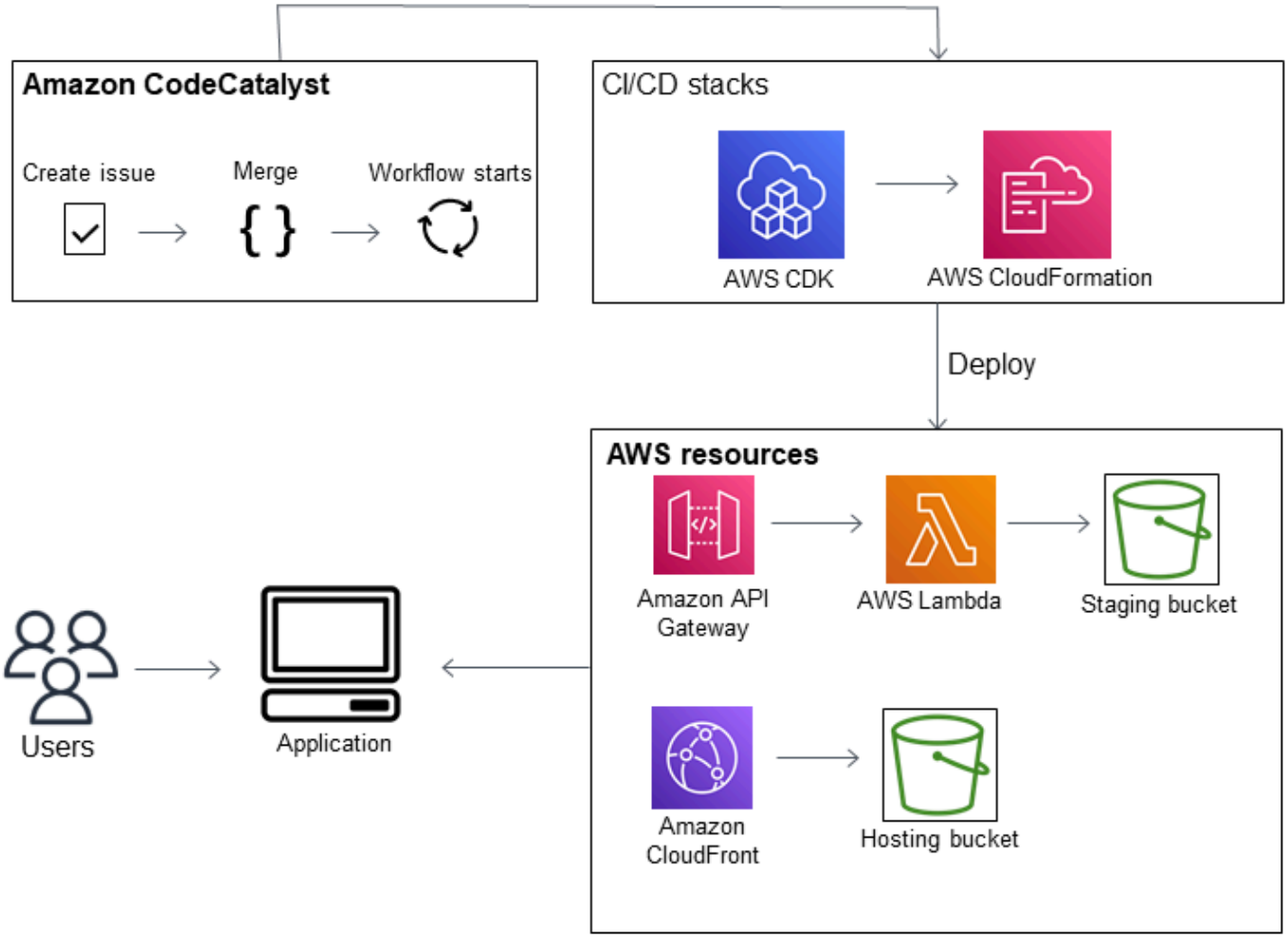
- 教學課程：使用動作的 Lint 程式 GitHub碼
- 教學課程：建立和更新 React 應用程式

教學課程：使用現代三層 Web 應用程式藍圖建立專案

透過使用藍圖建立專案，您可以更快地開始開發軟體。使用藍圖建立的專案包括您需要的資源，包括用來管理程式碼的來源存放庫，以及建置和部署應用程式的工作流程。在本教程中，我們將引導您使用現代三層 Web 應用程序藍圖在 Amazon CodeCatalyst 中創建項目。此教學課程也包括檢視已部署的範例、邀請其他使用者使用該範例，以及變更具有提取要求的程式碼，這些提取要求會在合併提取要求 AWS 帳戶 時自動建置並部署至連線的資源。在哪裡使用報告、活動摘要和其他工具 CodeCatalyst 建立專案，您的藍圖會在與專案 AWS 帳戶 相關聯的 AWS 資源中建立資源。您的藍圖檔案可讓您建置和測試範例現代應用程式，並將其部署到 AWS 雲端.

下圖說明如何使用中 CodeCatalyst 的工具來建立問題以追蹤、合併和自動建置變更，然後在 CodeCatalyst專案中啟動工作流程，該工作流程會執行動作以允許 AWS CDK 和 AWS CloudFormation 佈建您的基礎結構。

這些動作會在關聯的中產生資源， AWS 帳戶 並將您的應用程式部署至具有 API Gateway 端點的無伺服器 AWS Lambda功能。此 AWS Cloud Development Kit (AWS CDK) 動作會將一或多個 AWS CDK 堆疊轉換為 AWS CloudFormation 範本，並將堆疊部署至您的 AWS 帳戶. 堆疊中的資源包括用於分發動態 Web 內容的 Amazon CloudFront 資源、應用程式資料的 Amazon DynamoDB 執行個體，以及支援部署應用程式的角色和政策。



當您使用 Modern 三層 Web 應用程式藍圖建立專案時，會使用下列資源建立專案：

在 CodeCatalyst 項目中：

- 具有範例程式碼和工作流程 YAML 的來源儲存庫
- 一種工作流程，每當對默認分支進行更改時構建和部署示例代碼
- 您可以用來規劃和追蹤工作的問題板和待處理
- 包含範例程式碼中包含自動報告的測試報告套件

在相關的 AWS 帳戶：

- 建立應用程式所需資源的三個 AWS CloudFormation 堆疊。

若要取得有關將在中 AWS 建立的資源的展開詳細資訊，請參閱〈〉參考資料。 CodeCatalyst

> **Note**
>
> 專案中包含的資源和範例取決於您選取的藍圖。Amazon CodeCatalyst 提供數個專案藍圖，用來定義與其定義的語言或架構相關的資源。若要深入瞭解藍圖，請參閱專案藍圖參考。

主題

必要條件

若要在本教學課程中建立新式應用程式專案，您必須已完成中的工作，正在設定 CodeCatalyst如下所示：

- 有一個用於登錄的 AWS 生成器 ID CodeCatalyst。
- 屬於空間，並在該空間中為您指派 Space 管理員或超級使用者角色。如需詳細資訊，請參閱建立支援 AWS 產生器 ID 使用者的空間、管理空間使用者及空間管理員角色。
- 與您的空間建立 AWS 帳戶 關聯，並擁有您在註冊期間建立的 IAM 角色。例如，在註冊期間，您可以選擇使用稱為角色原則的角色原則來建立服務CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。該角色將具有附加唯CodeCatalystWorkflowDevelopmentRole-*spaceName*一

標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-*spaceName*務角色。如需建立角色的步驟，請參閱為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。

步驟 1：建立現代化的三層 Web 應用程式專案

建立專案之後，您的專案就是開發和測試程式碼、協調開發工作，以及檢視專案指標的地方。您的專案也包含您的開發工具和資源。

在本教學課程中，您將使用 Modern 三層 Web 應用程式藍圖來建立互動式應用程式。作為項目一部分自動創建和運行的工作流程將構建和部署應用程序。只有在為您的空間配置了所有角色和帳戶資訊之後，工作流程才會成功執行。成功執行工作流程後，您可以造訪端點 URL 以查看應用程式。

使用藍圖建立專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在主 CodeCatalyst 控台中，導覽至您要建立專案的空間。
3. 選擇建立專案。
4. 選擇從藍圖開始。
5. 在搜尋列中，輸入 **modern**。
6. 選取新式三層 Web 應用程式藍圖，然後選擇 [下一步]。
7. 在為您的專案命名中，輸入專案名稱。例如：

   MyExampleProject.

   > **Note**
   >
   > 名稱在您的空間中必須是唯一的。

8. 在帳戶中，選擇 AWS 帳戶 您在註冊時添加的。藍圖會將資源安裝到此帳戶。
9. 在部署角色中，選擇您在註冊期間新增的角色。例如，選擇 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。

   如果沒有列出角色，請新增一個角色。若要新增角色，請選擇新增 IAM 角色，然後將角色新增至您的 AWS 帳戶。如需詳細資訊，請參閱 管理 AWS 帳戶 空間。
10. 在運算平台中，選擇 Lambda。

11. 在前端託管選項中，選擇 Amplify 託管。如需相關資訊 AWS Amplify，請參閱何謂 AWS Amplify 主機？ 在《AWS Amplify 使用者指南》中。
12. 在部署區域中，輸入您希望藍圖部 AWS 區域 署 Mysfits 應用程式和支援資源的地區代碼。如需區域代碼的清單，請參閱 AWS 一般參考.
13. 在應用程式名稱中，保留預設值`mysfits`*`string`*。
14. (選擇性) 在 [產生專案預覽] 下，選擇 [檢視程式碼] 以預覽藍圖將安裝的來源檔案。選擇檢視工作流程以預覽藍圖將安裝的 CI/CD 工作流程定義檔。預覽會根據您的選取動態更新。
15. 選擇建立專案。

專案工作流程會在您建立專案後立即啟動。這將需要一點時間來完成構建和部署代碼。在此期間，繼續並邀請其他人加入您的項目。

步驟 2：邀請某人加入您的項目

現在您已經設定好專案，請邀請其他人與您合作。

邀請某人加入您的專案

1. 導覽至您要邀請使用者的專案。
2. 在導覽窗格中，選擇 [專案設定]。
3. 在 [成員] 索引標籤上選擇 [邀請]。
4. 輸入您要邀請成為專案使用者的人員的電子郵件地址。您可以輸入多個電子郵件地址，並以空格或逗號分隔。您也可以從非專案成員的空間成員中進行選擇。
5. 選擇使用者的角色。

   完成新增使用者後，請選擇 [邀請]。

步驟 3：建立要共同作業和追蹤工作的問題

CodeCatalyst 幫助您跟踪功能，任務，錯誤以及與問題相關的項目的任何其他工作。您可以創建問題以跟踪所需的工作和想法。默認情況下，當您創建問題時，它會添加到您的積壓。您可以將問題移至追蹤進行中工作的看板。您也可以將問題指派給特定專案成員。

若要建立專案的問題

1. 在導覽窗格中，選擇 [問題]。

2. 選擇 [建立問題]。
3. 在問題標題中，提供問題的名稱。選擇性地提供問題的描述。在此範例中，使用 **make a change in the src/mysfit_data.json file.**
4. 選擇優先順序、估計值、狀態和標籤。在受指派人下，選擇 [+ 新增我]，將問題指派給您自己。
5. 選擇 [建立問題]。這個問題現在可以在主機板上看到。選擇卡片，將問題移至「進行中」欄。

如需詳細資訊，請參閱 中的問題 CodeCatalyst。

步驟 4：檢視您的來源儲存庫

您的藍圖會安裝包含檔案的來源存放庫，以定義和支援您的應用程式或服務。源存儲庫中一些值得注意的目錄和文件是：

- .cloud9 目錄 — 包含 AWS Cloud9 開發環境的支援檔案。
- .codealyst 目錄 — 包含藍圖中包含的每個YAML工作流程的工作流程定義檔案。
- .idea 目錄 — 包含 JetBrains 開發環境的支援檔案。
- .vscode 目錄 — 包含視覺工作室程式碼開發環境的支援檔案。
- CDkSt ack 目錄 — 包含定義中基礎結構的 AWS CDK 堆疊檔案。 AWS 雲端
- src 目錄 — 包含應用程式原始程式碼。
- test 目錄 — 包含整數和單元測試的檔案，這些檔案會在您建置和測試應用程式時執行的自動化 CI/CD 工作流程的一部分執行。
- Web 目錄-包含前端源代碼。其他文件包括項目文件，例如包含有關項目的重要元數據的文件，網站的`index.html`頁面，用於鏈接代碼的`.eslintrc.cjs`文件，以及用於指定根文件和編譯器選項的文件。`package.json` `tsconfig.json`
- `Dockerfile`file — 說明應用程式的容器。
- `README.md`file — 包含專案的配置資訊。

若要導覽至專案的來源儲存庫

1. 導覽至您的專案，然後執行下列其中一項作業：
   - 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。
   - 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。在來源儲存庫中，從清單中選擇存放庫的名稱。您可以在篩選列中輸入部分存放庫名稱來篩選儲存庫清單。

2. 在儲存區域的首頁上，檢視儲存區域的內容以及相關資源的相關資訊，例如提取要求數目和工作流程。依預設，會顯示預設分支的內容。您可以從下拉式清單中選擇不同的分支來變更檢視。

第 5 步：使用測試分支創建開發環境並快速更改代碼

您可以建立開發環境，快速處理原始碼儲存庫中的程式碼。在本教程中，我們假設您將：

- 建立 AWS Cloud9 開發環境。
- 創建開發環境時，選擇在主分支之外的新分支中工作的選項。
- 使用此新分支的名稱`test`。

在稍後的步驟中，您將使用開發環境來變更程式碼並建立提取要求。

使用新分支建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要創建開發環境的項目。
3. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]、選擇 [原始碼儲存庫]，然後選擇您要建立開發環境的存放庫。
4. 在儲存庫首頁上，選擇建立開發環境。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 選擇要複製的存放庫，選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。
7. 選擇性地新增開發環境的別名。
8. 您也可以選擇 [開發環境] 設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
9. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。將在您選擇的 IDE 中打開一個新選項卡，並顯示您的開發環境。您可以編輯代碼並提交和推送更改。

在本節中，您將在中使用產生的範例應用程式，方 CodeCatalyst 法是變更具有提取要求的程式碼，這些提取要求會在合併提取要求 AWS 帳戶 時自動建置並部署至連線的資源。

若要在**src/mysfit_data.json**檔案中進行變更

1. 導航到您的項目開發環境。在中 AWS Cloud9，展開側邊導覽功能表以瀏覽檔案。展開mysfitssrc、和開啟src/mysfit_data.json。
2. 在檔案中，將"Age":欄位的值從 6 變更為 12。您的行應如下所示：

```
    {
        "Age": 12,
        "Description": "Twilight's personality sparkles like the night sky and is
 looking for a forever home with a Greek hero or God. While on the smaller side
 at 14 hands, he is quite adept at accepting riders and can fly to 15,000 feet.
 Twilight needs a large area to run around in and will need to be registered with
 the FAA if you plan to fly him above 500 feet. His favorite activities include
 playing with chimeras, going on epic adventures into battle, and playing with a
 large inflatable ball around the paddock. If you bring him home, he'll quickly
 become your favorite little Pegasus.",
        "GoodEvil": "Good",
        "LawChaos": "Lawful",
        "Name": "Twilight Glitter",
        "ProfileImageUri": "https://www.mythicalmysfits.com/images/
pegasus_hover.png",
        "Species": "Pegasus",
        "ThumbImageUri": "https://www.mythicalmysfits.com/images/pegasus_thumb.png"
    },
```

3. 儲存檔案。
4. 使用**cd /projects/mysfits**命令更改為 mysfits 存儲庫。
5. 使用 git add，git 提交和 git 推送命令添加，提交和推送更改。

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

步驟 6：檢視建置現代應用程式的工作流程

建立新式應用程式專案之後， CodeCatalyst 會代表您產生數個資源，包括工作流程。工作流程是在 .yaml 檔案中定義的自動化程序，描述如何建置、測試和部署程式碼。

在此自學課程中， CodeCatalyst 建立了一個工作流程，並在您建立專案時自動啟動它。(工作流程可能仍在執行，具體取決於您建立專案的多久之前。) 使用下列程序來檢查工作流程的進度、檢閱產生的記錄檔和測試報告，最後瀏覽至已部署應用程式的 URL。

若要檢查工作流程進度

1. 在 CodeCatalyst 主控台的功能窗格中，選擇 CI/CD，然後選擇 [工作流程]。

   這時系統顯示工作流程列表。這些是 CodeCatalyst 藍圖在您建立專案時產生並啟動的工作流程。

2. 觀察工作流程清單。您應該看到四個：

   - 頂端的兩個工作流程對應於您先前在中建立的`test`分支第 5 步：使用測試分支創建開發環境並快速更改代碼。這些工作流程是複製`main`分支上的工作流程。未處ApplicationDeploymentPipeline於作用中狀態，因為它已設定為與`main`分支搭配使用。工OnPullRequest作流程未執行，因為沒有提取要求。
   - 底部的兩個工作流程對應於先前執行藍圖時建立的`main`分支。ApplicationDeploymentPipeline工作流程處於使用中狀態，且具有進行中 (或已完成) 的執行。

     > **Note**
     >
     > 如果ApplicationDeploymentPipeline執行失敗並出現 Build @cdk_bootstrap 或DeployBackend錯誤，可能是因為您先前執行了 Modern 三層 Web 應用程式，並留下舊資源與目前藍圖的衝突。您需要刪除這些舊資源，然後重新執行工作流程。如需詳細資訊，請參閱 清除資源。

3. 在底部選擇與`main`分支相關聯的ApplicationDeploymentPipeline工作流程。此工作流程是使用`main`分支上的原始程式碼執行的。

   工作流程圖表隨即出現。該圖顯示了幾個塊，每個塊代表一個任務或動作。大多數動作都是垂直排列的，最上方的動作會在下列動作之前執行。並排排列的動作並行執行。群組在一起的動作必須全部成功執行，下面的動作才能啟動。

   主要區塊是：

   - WorkflowSource— 此區塊代表您的來源儲存庫。除其他資訊外，還會顯示來源儲存庫名稱 (mysfits) 和自動啟動工作流程執行的提交。 CodeCatalyst 創建項目時生成此提交。
   - 建置 — 此區塊代表兩個動作的群組，這兩個動作必須成功完成，下一個動作才能啟動。

- DeployBackend— 此區塊代表將應用程式的後端元件部署到 AWS 雲端的動作。
- 測試-此塊表示兩個測試操作的分組，這兩個測試操作都必須成功完成，下一個操作才能啟動。
- DeployFrontend— 此區塊代表將應用程式前端元件部署到雲端的動作 AWS 。

4. 選擇 [定義] 索引標籤 (靠近頂端)。工作流程定義檔會顯示在右側。該文件具有以下值得注意的部分：

   - 一個`Triggers`部分，在頂部。本節指出每當程式碼推送至來源儲存庫的`main`分支時，工作流程都必須啟動。推送至其他分支 (例如`test`) 不會啟動此工作流程。工作流程會使用分支上的檔案執`main`行。
   - 區`Actions`段，在`Triggers`。本節定義您在工作流程圖中看到的動作。

5. 選擇「最新」狀態標籤 (靠近頂端)，然後在工作流程圖表中選擇任何動作。
6. 在右側，選擇「組態」索引標籤，以查看上次執行期間動作所使用的組態設定。每個組態設定在工作流程定義檔案中都有相符的屬性。
7. 讓主控台保持開啟狀態，然後繼續下一個程序。

若要檢閱建置記錄和測試報告

1. 選擇「最新」狀態標籤。
2. 在工作流程圖中，選擇DeployFrontend動作。
3. 等待動作完成。注意「進行中」圖示
   ( )
   變更為「成功」圖示
   ( )。
4. 選擇構建後端操作。
5. 選擇 [記錄檔] 索引標籤，然後展開幾個區段，以檢視這些步驟的記錄訊息。您可以看到與後端設置相關的消息。
6. 選擇 [報表] 索引標籤，然後選擇`backend-coverage.xml`報表。 CodeCatalyst 顯示關聯的報告。此報告會顯示已執行的程式碼涵蓋範圍測試，並指出測試成功驗證的程式碼行比例，例如 80%。

   如需測試報告的詳細資訊，請參閱使用工作流程測試 CodeCatalyst。

> **Tip**
>
> 您也可以在導覽窗格中選擇 [報告] 來檢視測試報告。

7. 讓 CodeCatalyst 主機保持開啟狀態，然後繼續下一個程序。

確認現代應用程式已成功部署

1. 返回ApplicationDeploymentPipeline工作流程，然後選擇最新執行的「執行**##**」連結。
2. 在工作流程圖中，找到DeployFrontend動作，然後選擇 [檢視應用程式] 連結。我們的網站隨即出現。

> **Note**
>
> 如果您在DeployFrontend動作中沒有看到「檢視」應用程式連結，請確定您已選擇執行 ID 連結。

3. 搜索名為暮光閃光的飛馬 Mysfit。請注意年齡的值。它是6。您將進行代碼更改以更新年齡。

步驟 7：要求其他人檢閱您的變更

現在，您在名為的分支中發生了變更`test`，您可以要求其他人透過建立提取請求來檢閱它們。執行下列步驟來建立提取要求，以將`test`分支的變更合併到`main`分支中。

若要建立提取請求

1. 導航到您的項目。
2. 執行以下任意一項：
   - 在瀏覽窗格中，選擇 [程式碼]，選擇 [提取要求]，然後選擇 [建立提取要求]。
   - 在儲存區域首頁上，選擇 [其他]，然後選擇 [建立提取要求]。
   - 在專案頁面上，選擇 [建立提取要求]。
3. 在源存儲庫中，確保指定的源存儲庫是包含提交代碼的存儲庫。僅當您未從存放庫的主頁面建立提取要求時，此選項才會顯示。
4. 在「目的地」分支中，選擇要在檢閱程式碼之後將其合併到的分支。
5. 在原始碼分支中，選擇包含已提交程式碼的分支。

6. 在提取請求標題中，輸入一個標題，以幫助其他用戶了解需要審查的內容以及原因。
7. (選擇性) 在提取要求說明中，提供問題連結或變更說明等資訊。

> **Tip**
>
> 您可以選擇 [為我撰寫說明]， CodeCatalyst 自動產生提取要求中所包含變更的描述。您可以在將自動產生的描述新增至提取請求後，對其進行變更。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

8. (選擇性) 在問題中，選擇「連結問題」，然後從清單中選擇問題或輸入其 ID。若要取消連結問題，請選擇「取消連結」圖示。
9. (選擇性) 在 [必要的複查者] 中，選擇 [新增必要的審核者]。從專案成員清單中選擇要新增的專案成員。必要的審核者必須先核准變更，才能將提取請求合併到目的地分支。

> **Note**
>
> 您無法同時將審核者新增為必要審核者和選擇性複查者。您無法將自己新增為審核者。

10. (選擇性) 在 [選擇性複查者] 中，選擇 [新增選用複查者]。從專案成員清單中選擇要新增的專案成員。選用審核者不需要核准變更作為需求，才能將提取請求合併到目的地分支。
11. 查看分支之間的差異。提取請求中顯示的差異在於源分支中的修訂版本和合併基礎之間的更改，這是在提取請求創建時目的地分支的頭部提交。如果沒有顯示任何更改，則分支可能相同，或者您可能已經為源和目標選擇了相同的分支。
12. 當您滿意提取請求包含您要檢閱的程式碼和變更時，請選擇 [建立]。

> **Note**
>
> 建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整個提取請求。您可以使用 @符號後面接到檔案名稱來新增資源連結，例如檔案。

當您建立提取要求時，OnPullRequest工作流程會開始使用`test`分支中的來源檔案。審核者核准程式碼變更時，您可以透過選擇工作流程並檢視測試輸出來觀察結果。

檢閱變更之後，您可以合併程式碼。將程式碼合併到預設分支會自動啟動建置和部署變更的工作流程。

若要從 CodeCatalyst 主控台合併提取要求

1. 瀏覽至您的現代應用程式專案。
2. 在專案頁面的「開啟提取請求」下，選擇您要合併的提取請求。如果您沒有看到提取請求，請選擇 [檢視全部]，然後從清單中選擇。選擇 Merge (合併)。
3. 從提取請求的可用合併策略中選擇。選擇性地選取或取消選取要在合併提取請求之後刪除來源分支的選項，然後選擇「合併」。

> **Note**
>
> 如果「合併」按鈕未處於作用中狀態，或者您看到「不可合併」標籤，表示一或多位必要的複查者尚未核准提取請求，或者無法在主控台中合併提取請求。 CodeCatalyst 未核准提取請求的審核者會在 [提取請求詳細資料] 區域的 [概觀] 中以時鐘圖示表示。如果所有必要的複查者都已核准提取請求，但「合併」按鈕仍處於作用中狀態，則您可能會發生合併衝突。您可以在 CodeCatalyst控制台中解決目的地分支的合併衝突，然後合併提取請求，或者您可以解決衝突並在本地合併，然後將包含合併的提交推送到 CodeCatalyst。如需詳細資訊，請參閱合併一個提取請求（Git）和您的 Git 文件。

將分支中的變更合併到`test`分支之**`main`**後，變更會自動啟動建立和部署變更的ApplicationDeploymentPipeline工作流程。

若要查看合併的提交，請在 ApplicationDeploymentPipeline 工作流程中執行

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 在「工作流程」中 ApplicationDeploymentPipeline，展開「最近的執行」。您可以看到合併提交開始執行的工作流程。（可選）選擇它來監視運行進度。
3. 當運行完成時，重新加載您之前訪問的 URL。檢視飛馬座以確認年齡已變更。

步驟 8：關閉問題

問題解決後，可以在 CodeCatalyst 主控台上關閉問題。

若要關閉專案的問題

1. 導航到您的項目。
2. 在導覽窗格中，選擇 [問題]。
3. D rag-and-drop 問題到「完成」列。

如需詳細資訊，請參閱 中的問題 CodeCatalyst。

清除資源

清理 CodeCatalyst 並從您 AWS 的環境中移除本自學課程的痕跡。

您可以選擇繼續使用本自學課程所使用的專案，也可以刪除專案及其關聯的資源。

> **Note**
>
> 刪除此項目將刪除項目中所有成員的所有存儲庫，問題和成品。

刪除專案

1. 導覽至您的專案，然後選擇 [專案設定]。
2. 選擇「一般」頁籤。
3. 在專案名稱下，選擇 [刪除專案]。

刪除 AWS CloudFormation 和 Amazon S3 中的資源

1. AWS Management Console 使用您新增至 CodeCatalyst空間的相同帳戶登入。
2. 轉到服AWS CloudFormation務。
3. 刪除字符#堆棧。
4. ##########
5. 選擇（但不刪除）CDK 工具包堆棧。選擇 Resources (資源) 標籤。選擇StagingBucket連結，然後刪除 Amazon S3 中的儲存貯體和儲存貯體內容。

   > **Note**
   >
   > 如果您未手動刪除此值區，則在重新執行 Modern 三層 Web 應用程式藍圖時可能會看到錯誤訊息。

6. (選擇性) 刪除 CDK 工具組堆疊。

參考資料

現代三層 Web 應用程式藍圖會將資源部署到您的 CodeCatalyst 空間和雲端中的 AWS AWS 帳戶。這些資源是：

- 在您的 CodeCatalyst 空間中：
  - 包含下列資源的 CodeCatalyst 專案：
    - 源代碼庫-此存儲庫包含「Mysfits」Web 應用程序的示例代碼。
    - 工作流程 — 每當對默認分支進行更改時，此工作流程構建和部署 Mysfits 應用程序代碼
    - 問題板與待處理項目 — 此主機板與待處理項目可用來規劃和追蹤工作。
    - 測試報告套件 — 此套件包含示例代碼中包含的自動報告。
- 在相關的 AWS 帳戶：
  - CDKToolkit 堆疊 — 此堆疊會部署下列資源：

- Amazon S3 暫存貯體、儲存貯體政策和用於加密儲存貯體的 AWS KMS 金鑰。
- 部署動作的 IAM 部署角色。
- AWS 支援堆疊中資源的 IAM 角色和政策。

> **Note**
>
> CDKToolkit 不會拆除並重新建立每個部署。這是在每個帳戶中啟動的堆疊，以支援 AWS CDK.

- 開發神秘的###BackEnd堆棧-此堆棧部署以下後端資源：
  - Amazon API Gateway 端點。
  - AWS 支援堆疊中資源的 IAM 角色和政策。
  - AWS Lambda 函數和層為現代應用程式提供無伺服器運算平台。
  - 儲存貯體部署和 Lambda 函數的 IAM 政策和角色。
- 一個 mysfits ###堆棧-此堆棧部署 AWS Amplify 前端應用程序。

另請參閱

如需有關在此教學課程中建立資源之 AWS 服務的詳細資訊，請參閱下列內容：

- Amazon S3 — 將前端資產存放在物件儲存服務上的服務，提供業界領先的可擴展性、資料高可用性、安全性和效能。如需詳細資訊，請參閱 Amazon S3 使用者指南。
- Amazon API Gateway — 用於建立、發佈、維護、監控和保護任何規模的 REST、HTTP 和 WebSocket API 的服務。如需詳細資訊，請參閱 API Gateway 開發人員指南。
- Amplify — 託管您的前端應用程式的服務。如需詳細資訊，請參閱AWS Amplify 主機使用者指南。
- AWS Cloud Development Kit (AWS CDK)— 用於在代碼中定義雲基礎架構並通過進行佈建的框架 AWS CloudFormation。 AWS CDK 包括 AWS CDK Toolkit，這是一個命令行工具，用於與 AWS CDK 應用程序和堆棧進行交互。如需詳細資訊，請參閱《AWS Cloud Development Kit (AWS CDK) 開發人員指南》。
- Amazon DynamoDB — 用於存放資料的全受管 NoSQL 資料庫服務。如需詳細資訊，請參閱 Amazon DynamoDB 開發人員指南。
- AWS Lambda— 一種在高可用性計算基礎結構上叫用程式碼的服務，而無需佈建或管理伺服器。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》。

- AWS IAM — 用於安全控制存取權限 AWS 及其資源的服務。如需詳細資訊，請參閱 IAM 使用者指南。

教學課程：從空白專案開始，並手動新增資源

您可以在建立專案時選擇 Empty 專案藍圖，以建立不含任何預先定義資源的空白專案。建立空白專案之後，您可以根據專案需求建立並加入資源。由於沒有藍圖建立的專案在建立時是空的，因此此選項需要有關建立和設定 CodeCatalyst 資源的更多知識才能開始使用。

主題

必要條件

若要建立空白專案，您必須將 S pace 管理員或超級使用者角色指派給您。如果這是您第一次登入 CodeCatalyst，請參閱正在設定 CodeCatalyst。

創建一個空的項目

創建一個項目是能夠一起工作的第一步。如果您想要建立自己的資源 (例如來源儲存庫和工作流程)，您可以從空白專案開始。

建立空專案的步驟

1. 導覽至您要在其中建立專案的空間。
2. 在空間儀表板上，選擇建立專案。
3. 選擇「從頭開始」。
4. 在「為您的專案命名」下，輸入您要指派給專案的名稱。名稱在您的空間中必須是唯一的。
5. 選擇建立專案。

現在您有一個空的項目，下一步是創建一個源存儲庫。

建立來源儲存庫

創建一個源存儲庫以存儲和協作項目的代碼。專案成員可以將此儲存庫複製到其本機電腦，以處理程式碼。或者，您可以選擇連結託管在支援服務中的存放庫，但本教學課程並未涵蓋這些內容。如需詳細資訊，請參閱 連結來源儲存庫。

若要建立來源儲存庫

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
5. 在存放庫名稱中，提供存放庫的名稱。在本指南中，我們使用 *codecatalyst-source-repository*，但您可以選擇不同的名稱。存放庫名稱在專案中必須是唯一的。如需有關存放庫名稱需求的詳細資訊，請參閱來源儲存庫的配額 CodeCatalyst。
6. (選擇性) 在說明中，新增存放庫的說明，以協助專案中的其他使用者瞭解存放庫的用途。
7. (選擇性) 針對您計劃推送的程式碼類型新增`.gitignore`檔案。
8. 選擇建立。

> **Note**
>
> CodeCatalyst 創建`README.md`文件時將文件添加到存儲庫中。 CodeCatalyst還會在名為 main 的默認分支中為存儲庫創建一個初始提交。您可以編輯或刪除 README.md 檔案，但無法變更或刪除預設分支。

您可以建立開發環境，在儲存庫中快速新增程式碼。在本教學課程中，我們建議您使用建立開發環境 AWS Cloud9，並選擇在建立開發環境時從主分支建立分支的選項。我們使用此分支的名稱，但您可以根據**`test`**需要輸入不同的分支名稱。

使用新分支建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要創建開發環境的項目。

3. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]、選擇 [原始碼儲存庫]，然後選擇您要建立開發環境的存放庫。
4. 在儲存庫首頁上，選擇建立開發環境。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 選擇要複製的存放庫，選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。
7. 選擇性地新增開發環境的別名。
8. 您也可以選擇 [開發環境] 設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
9. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。將在您選擇的 IDE 中打開一個新選項卡，並顯示您的開發環境。您可以編輯代碼並提交和推送更改。

建立工作流程以建置、測試和部署程式碼變更

在中 CodeCatalyst，您可以在工作流程中組織應用程式或服務的建置、測試和部署。工作流程由動作組成，可設定為在發生指定的來源儲存庫事件 (例如程式碼推送或開啟或更新提取要求) 之後自動執行。如需工作流程的相關詳細資訊，請參閱 使用中的工作流程來建置、測試和部署 CodeCatalyst。

依照中的指示建立開始使用中的工作流程 CodeCatalyst您的第一個工作流程。

邀請某人加入您的項目

現在，您已經設置了自定義項目，請邀請其他人與您合作。

若要邀請某人加入您的專案

1. 導覽至您要邀請使用者的專案。
2. 在導覽窗格中，選擇 [專案設定]。
3. 在 [成員] 索引標籤上選擇 [邀請]。
4. 輸入您要邀請成為專案使用者的人員的電子郵件地址。您可以輸入多個電子郵件地址，並以空格或逗號分隔。您也可以從非專案成員的空間成員中進行選擇。
5. 選擇使用者的角色。

   完成新增使用者後，請選擇 [邀請]。

建立問題以協同合作並追蹤工作

CodeCatalyst 幫助您跟踪功能，任務，錯誤以及與問題相關的項目的任何其他工作。您可以創建問題以跟踪所需的工作和想法。默認情況下，當您創建問題時，它會添加到您的積壓。您可以將問題移至追蹤進行中工作的看板。您也可以將問題指派給特定專案成員。

若要為專案建立問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

   請確定您要在要建立問題的專案中瀏覽。若要檢視所有專案，請在導覽窗格中選擇 Amazon CodeCatalyst，並視需要選擇 [檢視所有專案]。選擇您要在其中建立或處理問題的專案。
2. 在功能窗格中，選擇 [追蹤]，然後選擇 [積壓]。
3. 選擇 [建立問題]。
4. 在問題標題中，提供問題的名稱。選擇性地提供問題的描述。如有需要，請選擇問題的狀態、優先順序和估計值。您也可以從專案成員清單中將問題指派給專案成員。

   > **Tip**
   >
   > 您可以選擇將問題分配給 Amazon Q，讓 Amazon Q 嘗試解決問題。如果嘗試成功，則會建立提取要求，且問題的狀態會變更為「審閱中」，以便您可以檢閱並測試程式碼。如需詳細資訊，請參閱 教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作。
   > 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。
5. 選擇儲存。

建立問題之後，您可以將問題指派給專案成員、預估問題，並在看板上追蹤問題。如需更多詳細資訊，請參閱 中的問題 CodeCatalyst。

教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作

Amazon 的生成式 AI 功能 CodeCatalyst 正在預覽版中，可能會有所變更。它們僅在美國西部 (奧勒岡) 區域提供。生成式 AI 功能的使用方式因層級而異。如需詳細資訊，請參閱 定價。

如果您在啟用生成 AI 功能的空間中，Amazon CodeCatalyst 中有一個專案和來源儲存庫，則可以使用這些功能來協助加速軟體開發。開發人員經常有更多的任務要做，而不是時間來完成它們。在建立提取

要求以檢閱這些變更時，他們通常不會花時間向團隊成員解釋他們的程式碼變更，因此希望其他使用者能夠找到不言自明的變更。提取請求建立者和審核者也沒有時間徹底尋找和閱讀提取請求上的所有註解，特別是在提取請求有多個修訂時。 CodeCatalyst 包括生成式 AI 功能，可以幫助團隊成員更快地完成任務，並增加他們必須專注於工作中最重要部分的時間。

> **Note**
>
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於「為我撰寫說明」、「建立內容摘要」和「指派問題給 Amazon Q」功能開發功能功能建置在 Amazon 基岩上，因此使用者可以充分利用 Amazon Bedrock 中實作的控制項來執行安全、安全性和負責任的人工智慧 (AI) 使用。

在本教學課程中，您將學習如何使用中的生成 AI 功能 CodeCatalyst 來協助您摘要建立提取請求時分支之間的變更，並總結提取要求上留下的註解。您也將學習如何針對簡單的程式碼變更或改善想法建立問題，並將其指派給 Amazon Q。

必要條件

若要使用本自學課程中的 CodeCatalyst 功能，您必須先完成並可存取下列資源：

- 您擁有用於登入的 AWS 產生器 ID 或單一登入 (SSO) 身分 CodeCatalyst。
- 您的專案位於已啟用生成 AI 功能的空間中。如需詳細資訊，請參閱管理生成 AI 功能。
- 您在該空間中的專案中具有參與者或專案管理員角色。
- 該項目至少配置了一個源存儲庫。不支援連結的儲存庫。
- 將問題指派給由生成 AI 建立的初始解決方案時，無法使用 Jira Software 延伸功能來設定專案。此功能不支援擴充功能。

如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間、中的問題 CodeCatalyst、中的擴展 CodeCatalyst 和 在 Amazon 中使用角色 CodeCatalyst。

本教程是基於使用 Python 現代三層 Web 應用程序藍圖創建的項目。如果您使用以不同藍圖建立的專案，您仍然可以遵循這些步驟，但是某些細節會有所不同，例如範例程式碼和語言。

創建提取請求時，創建分支之間的代碼更改摘要

提取請求是您和其他專案成員可以檢閱、註解和合併從一個分支到另一個分支的程式碼變更的主要方式。您可以使用提取請求協同檢閱程式碼變更，以瞭解次要變更或修正、主要新增功能或發行軟體的新

版本。總結程式碼變更和變更背後的意圖，做為提取要求說明的一部分，對於檢閱程式碼的其他人來說很有幫助，也有助於歷史瞭解程式碼隨著時間的變更。但是，開發人員通常依賴他們的代碼來解釋自己或提供模糊的細節，而不是用足夠的細節來描述他們的更改，以便審閱者了解他們正在審查的內容或代碼更改背後的意圖。

您可以在建立提取請求時使用「為我寫入描述」功能，讓 Amazon Q 為提取請求中包含的變更建立說明。選擇此選項時，Amazon Q 會分析包含程式碼變更的來源分支與您要合併這些變更的目的地分支之間的差異。然後，它會建立這些變更內容的摘要，以及它對這些變更的意圖和影響的最佳解釋。

您可以在建立的任何提取要求中試用這項功能，但在本教學課程中，我們會對以 Python 為基礎的 Modern Modern 三層 Web 應用程式藍圖中所建立的專案中所包含的程式碼進行一些簡單的變更，進行測試。

> **Tip**
>
> 如果您使用的是使用不同藍圖或您自己的程式碼建立的專案，您仍然可以按照本教學課程進行操作，但是本教學課程中的範例不會與專案中的程式碼相符。而不是下面建議的例子，在分支中對項目的代碼進行簡單的更改，然後創建一個提取請求來測試功能，如以下步驟所示。

首先，您將在源存儲庫中創建一個分支。然後，您將使用控制台中的文本編輯器對該分支中的文件進行快速更改代碼。然後，您將創建一個提取請求，並使用為我寫入描述功能來總結您所做的更改。

要創建一個分支（控制台）

1. 在主 CodeCatalyst 控台中，導覽至來源儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇您要在其中創建分支的存儲庫。
4. 在儲存庫的概觀頁面上，選擇 [其他]，然後選擇 [建立分支]。
5. 輸入分支的名稱。
6. 選擇要從中建立分支的分支，然後選擇 [建立]。

一旦你有了一個分支，只需簡單的更改編輯該分支中的文件。在此範例中，您將編輯`test_endpoint.py`檔案，將測試的重試次數從變更**3**為 5 次。

> **Tip**
>
> 您也可以選擇建立或使用開發環境來變更此程式碼。如需詳細資訊，請參閱 建立開發環境。

在主控台中編輯**test_endpoint.py**檔案

1. 在**mysfits**來源存放庫的概觀頁面上，選擇分支下拉式清單，然後選擇您在上一個程序中建立的分支。
2. 在「檔案」中，導覽至您要編輯的檔案。例如，若要編輯test_endpoint.py檔案，請展開測試、展開 integ，然後選擇test_endpoint.py。
3. 選擇編輯。
4. 在第 7 行中，變更所有測試將重試的次數：

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

   至:

   ```
   def test_list_all(retry=5):
   ```
5. 選擇提交並將更改提交到分支。

現在您已經有了變更的分支，您可以建立提取要求。

建立包含變更摘要的提取要求

1. 在儲存區域的總覽頁面上，選擇 [其他]，然後選擇 [建立提取要求]。
2. 在「目標」分支中，選擇要在檢閱程式碼之後將其合併到的分支。

   > **Tip**
   >
   > 在上一個程序中選擇您從中建立分支的分支，以進行此功能的最簡單示範。例如，如果您是從儲存庫的預設分支建立分支，請選擇該分支作為提取要求的目標分支。
3. 在 S ource 分支中，選擇包含您剛剛提交到test_endpoint.py文件的更改的分支。
4. 在提取請求標題中，輸入一個標題，以幫助其他用戶了解需要審查的內容以及原因。
5. 在 [提取要求說明] 中，選擇 [為我撰寫說明]， CodeCatalyst 建立提取要求中包含之變更的描述。

6. 這時系統顯示「變更」的摘要。檢閱建議的文字，然後選擇「接受」並新增至說明。
7. 選擇性地修改摘要，以更好地反映您對程式碼所做的變更。您也可以選擇將審核者或連結問題新增至此提取請求。當您完成所需的其他變更之後，請選擇 [建立]。

建立提取要求中程式碼變更留下的註解摘要

當使用者檢閱提取要求時，他們通常會在該提取要求中的變更留下多個註解。如果有很多評論者的評論很多，在意見反應中挑出共同的主題可能很困難，甚至可以確定您已經審核了所有修訂版本中的所有評論。您可以使用「建立註解摘要」功能，讓 Amazon Q 分析提取請求中程式碼變更留下的所有註解，並建立這些註解的摘要。

> **Note**
>
> 註釋摘要是暫時的。如果您重新整理提取要求，摘要將會消失。內容摘要不包含整體提取要求的註解，只包含提取要求修訂版本中程式碼差異的註解。

若要在提取要求中建立註解摘要

1. 導覽至您在上一個程序中建立的提取請求。

   > **Tip**
   >
   > 如果您願意，您可以在項目中使用任何打開的提取請求。在導覽列中，選擇「程式碼」，選擇「提取要求」，然後選擇任何開啟的提取要求。

2. 如果提取請求尚未有註解，請在「變更」中新增一些註解至提取要求。
3. 在概觀中，選擇建立注釋摘要。完成後，「註解摘要」區段會展開。
4. 檢閱提取請求修訂版本中程式碼變更所留下的註解摘要，並將其與提取請求中的註解進行比較。

建立問題並將其指派給 Amazon Q

開發團隊創建問題來跟踪和管理他們的工作，但有時問題仍然存在，因為要么不清楚誰應該對它進行研究，或者問題需要對代碼庫的特定部分進行研究，否則必須首先參加其他緊急工作。 CodeCatalyst 包括與名為 Amazon Q 的生成 AI 助理整合，該助理可以根據問題的標題和說明分析問題。如果您將問題指派給 Amazon Q，它會嘗試建立草稿解決方案供您評估。這可以幫助您和您的團隊專注於需要注意的問題並最佳化您的工作，而 Amazon Q 則可針對您沒有資源可立即解決的問題提供解決方案。

> **Tip**
>
> Amazon Q 在簡單問題和簡單的問題上表現最佳。為了獲得最佳效果，請使用簡單的語言清楚地解釋您想要完成的工作。

當您將問題指派給 Amazon Q 時， CodeCatalyst 會將問題標示為已封鎖，直到您確認您希望 Amazon Q 如何處理此問題為止。它要求您回答三個問題，然後才能繼續：

- 無論您是要確認所採取的每個步驟，還是要在沒有反饋的情況下繼續進行。如果您選擇確認每個步驟，您可以回覆 Amazon Q，並提供有關其建立方法的意見反應，以便在需要時重複執行方法。如果您選擇此選項，Amazon Q 也可以檢閱使用者在其建立的任何提取請求上留下的意見反應。如果您選擇不確認每個步驟，Amazon Q 可能會更快地完成其工作，但不會審核您在問題或其建立的任何提取請求中提供的任何意見反應。
- 您是否要允許其更新工作流程檔案作為其工作的一部分。您的專案可能已設定為在提取要求事件上開始執行的工作流程。如果是這樣，Amazon Q 建立的任何提取請求 (包括建立或更新工作流程 YAML) 都可能會開始執行提取請求中包含的工作流程。最佳做法是，請勿選擇允許 Amazon Q 處理工作流程檔案，除非您確定專案中沒有工作流程會自動執行這些工作流程，然後再檢閱並核准其建立的提取請求。
- 您希望它在哪個源存儲庫中工作。即使您的專案有多個來源儲存庫，Amazon Q 也只能處理一個來源儲存庫中的程式碼。不支援連結的儲存庫。

完成並確認您的選擇後，Amazon Q 會將問題移至「進行中」狀態，同時嘗試根據問題標題、說明以及指定存放庫中的程式碼判斷請求的內容。它將創建一個固定的評論，在其中將提供有關其工作狀態的更新。檢閱資料之後，Amazon Q 將制定解決方案的潛在方法。Amazon Q 會記錄其動作，方法是更新其釘選註解，並在每個階段評論問題的進展。與固定的評論和回復不同，它不會保留其工作的嚴格按時間順序進行記錄。相反，它將與其工作相關的最相關信息放在固定評論的頂層。它將嘗試根據其方法和對存儲庫中已經存在的代碼的分析來創建代碼。如果它成功生成了一個潛在的解決方案，它將創建一個分支並提交代碼到該分支。然後創建一個拉取請求，將該分支與默認分支合併。Amazon Q 完成工作後，會將問題移至「審核中」，讓您和您的團隊知道有可供您評估的程式碼。

> **Note**
>
> 此功能僅可透過「問題」使用。如果您已將專案設定為搭配 Jira 軟體擴充功能使用 Jira，則無法使用此功能。此外，如果您已自訂主機板的版面配置，問題可能不會變更狀態。為獲得最佳結果，請僅對具有標準電路板配置的專案使用此功能。

將問題指派給 Amazon Q 後，就無法變更問題的標題或說明，也無法將問題指派給其他人。如果您從問題中取消指派 Amazon Q，它將完成目前的步驟，然後停止工作。一旦取消指派問題，就無法繼續工作或重新指派問題。

在教學課程的這一部分中，您將根據使用 Modern 三層 Web 應用程式藍圖所建立之專案中所包含的程式碼潛在功能來建立三個問題：一個用來新增 mysfit 生物、一個用來新增排序功能，以及一個用來更新工作流程以包含名為的分支。**`test`**

> **Note**
>
> 如果您正在使用不同程式碼的專案中工作，請建立包含與該程式碼庫相關的標題和描述的問題。

若要建立問題並產生解決方案供您評估

1. 在導覽窗格中，選擇 [問題]，並確定您在 [看板] 檢視中。
2. 選擇 [建立問題]。
3. 為問題提供一個標題，以簡單的語言解釋您想要執行的操作。例如，針對此問題，請輸入的標題**`Create another mysfit named Quokkapus`**。在說明中，提供下列詳細資訊：

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

Name: Quokkapus

Species: Quokka-Octopus hybrid

Good/Evil: Good

Lawful/Chaotic: Chaotic

Age: 216

Description: Australia is full of amazing marsupials, but there's nothing there
 quite like the Quokkapus.
She's always got a friendly smile on her face, especially when she's using her
 eight limbs to wrap you up
```

```
in a great big hug. She exists on a diet of code bugs and caffeine. If you've got
 some gnarly code that needsa
assistance, adopt Quokkapus and put her to work - she'll love it! Just make sure
 you leave enough room for
her to grow, and keep that coffee coming.
```

4. (選擇性) 附加影像以用作問題 mysfit 的縮圖和個人檔案圖片。如果您這麼做，請更新說明以包含您要使用的影像和原因的詳細資料。例如，您可以在說明中添加以下內容：「mysfit 需要將圖像文件部署到網站上。將這些圖像添加到源存儲庫。」

> **Note**
>
> 在本教程中，附加的圖像將不會部署到網站。您可以自行將影像新增至網站，然後為 Amazon Q 留下註解，以便在建立提取請求後更新其程式碼，以指向您要使用的映像。

請檢閱說明，並確定其中包含所有可能需要的詳細資料，然後再繼續進行下一個步驟。

5. 在受指派人中，選擇指派給 Amazon Q。
6. 在來源儲存庫中，選擇包含專案程式碼的來源儲存庫。
7. 在每個步驟之後滑動「需要 Amazon Q 停止」，並等待其工作選擇器的審核到使用中狀態。

> **Note**
>
> 選擇讓 Amazon Q 在每個步驟之後停止的選項可讓您對問題發表評論，並且可以選擇讓 Amazon Q 根據您的意見變更方法最多三次。如果您選擇不讓 Amazon Q 在每個步驟後停止，以便您可以檢閱其工作，因為 Amazon Q 不等待您的意見反應，因此工作可能會更快地繼續進行，但是您無法透過留下評論來影響 Amazon Q 所採取的方向。如果您選擇該選項，Amazon Q 也不會回應提取請求中留下的註解。

8. 將允許 Amazon Q 修改工作流程檔案選擇器保留為非使用中狀態。
9. 選擇 [建立問題]。您的檢視會變更為「問題」看板。
10. 選擇創建問題以創建另一個問題，這次是標題**`Change the get_all_mysfits() API to return mysfits sorted by the Age attribute`**。將此問題指派給 Amazon Q 並建立問題。
11. 選擇創建問題以創建另一個問題，這次是標題**`Update the OnPullRequest workflow to include a branch named test in its triggers`**。選擇性地連結至描述中的工作流程。

將此問題指派給 Amazon Q，但這次請確保將允許 Amazon Q 修改工作流程檔案選取器設定為使用中狀態。建立問題以返回「問題」看板。

> **Tip**
>
> 您可以輸入 at 符號 (@) 並輸入檔案名稱來搜尋包括工作流程檔案在內的檔案。

建立並指派問題之後，問題就會進入進行中。Amazon Q 將在固定評論中添加跟踪問題進度的評論。如果它能夠定義解決方案的方法，它將更新問題的描述，其中包含其對代碼庫的分析的背景部分和一個方法部分，詳細說明了其建立解決方案的建議方法。如果 Amazon Q 成功提出問題中所述問題的解決方案，它將在該分支中建立分支和程式碼變更，以實作其提議的解決方案。程式碼準備就緒後，就會建立提取要求，以便您檢閱建議的程式碼變更、將該提取要求的連結新增至問題，然後將問題移至審核中。

> **Important**
>
> 在合併提取請求之前，您應該始終檢查提取請求中的任何代碼更改。如同任何其他程式碼變更，合併 Amazon Q 所做的程式碼變更可能會對程式碼基底和基礎設施程式碼造成負面影響，如果合併的程式碼未正確檢閱且在合併時包含錯誤。

檢閱包含 Amazon Q 所做變更的問題和連結的提取請求

1. 在問題中，選擇指派給正在進行中的 Amazon Q 的問題。檢閱註解以監控機器人的進度。如果存在，請檢閱背景並將其記錄在問題描述中，然後選擇 X 以關閉問題窗格。
2. 現在選擇指派給 Amazon Q 的問題正在審查中。審查背景和接近它記錄在問題的描述。檢閱註解以瞭解其執行的動作。在 [提取要求] 中，選擇 [開啟] 標籤旁的提取要求連結，以檢閱程式碼。
3. 在提取請求中，檢閱程式碼變更。如需詳細資訊，請參閱 檢閱提取要求。如果您希望 Amazon Q 變更任何建議的程式碼，請在提取請求上留下註解。為 Amazon Q 留下評論以獲得最佳效果時，請特別注意。

   例如，當您檢閱針對建立的提取要求時`Create another mysfit named Quokkapus`，您可能會注意到說明中有錯字。您可以為 Amazon Q 發表評論，該評論指出「通過在「需求」和「a」之間添加空格來更改描述以修復錯字「needsa」。 或者，您可以發表評論，告訴 Amazon Q 更新說明，並提供整個修改後的說明供其合併。

   如果您將新 mysfit 的圖像上傳到網站上，則可以為 Amazon Q 發表評論，以使用指向圖像和縮略圖的指針更新 mysfit 以用於新的 mysfit。

> **Note**
>
> Amazon Q 不會回復個人評論。如果您在建立問題時選擇在每個步驟之後停止等待核准的預設選項，Amazon Q 才會納入提取請求中留下的意見反應。

4. (選擇性) 在您和其他專案使用者保留所有您想要變更程式碼的註解之後，請選擇 [建立修訂]，讓 Amazon Q 建立包含您在註解中請求的變更的提取請求的修訂版本。Amazon Q 將在概觀中報告修訂建立進度的進度，而不是在變更中報告。請務必重新整理瀏覽器，以檢視 Amazon Q 建立修訂版的最新更新。

> **Note**
>
> 只有建立問題的使用者才能建立提取請求的修訂版本。您只能請求一個提取請求的修訂版本。在選擇 [建立修訂版本] 之前，請確定您已解決所有有關注釋的問題，並且對注釋的內容感到滿意。

5. 此範例專案中的每個提取要求都會執行工作流程。在合併提取請求之前，請確定工作流程執行成功。您也可以選擇建立其他工作流程和環境，以便在合併程式碼之前測試程式碼。如需詳細資訊，請參閱 開始使用中的工作流程 CodeCatalyst。
6. 當您滿意提取請求的最新版次時，請選擇「合併」。

清除資源

完成此教學課程後，請考慮採取下列動作來清除您在本教學課程中建立的任何不再需要的資源。

- 從不再處理的任何問題中取消指派 Amazon Q。如果 Amazon Q 已完成問題的工作或找不到解決方案，請確保取消指派 Amazon Q，以避免達到生成 AI 功能的最大配額。如需詳細資訊，請參閱管理生成式 AI 功能和定價。
- 將工作完成的任何問題移至「完成」。
- 如果不再需要該專案，請刪除該專案。

教學課程：建立包含可組合式 PDK 藍圖的完整堆疊應用程式

Amazon CodeCatalyst 提供許多不同的藍圖，協助您快速開始使用專案。使用藍圖建立的專案包含您所需的資源，包括來源存放庫、原始程式碼範例、CI/CD 工作流程、建置與測試報告，以及整合式問題追蹤工具。不過，有時您可能想要逐步建置專案，或將功能新增至由藍圖建立的現有專案。您也可以使

用藍圖來執行此操作。本教學課程示範如何開始使用單一藍圖，該藍圖奠定了基礎，並允許您將所有專案程式碼儲存在單一存放庫中。從那裡，您可以在方便的時候在初始藍圖之上套用其他藍圖，靈活地整合其他資源和基礎結構。通過這種構建塊方法，您可以滿足多個項目中的特定需求。

本教學說明如何將多個 AWS 專案開發套件 (AWS PDK) 藍圖組成，以建立包含 React 網站、Smithy API 和支援 CDK 基礎設施的應用程式，以便將其部署到 AWS。AWS PDK 提供常見模式的建置區塊，以及管理和建置專案的開發工具。如需詳細資訊，請參閱 AWS PDK GitHub 來源儲存庫。

下列 PDK 藍圖的設計是為了彼此搭配使用，以組合方式建置應用程式：

- Monorepo-創建一個根級項目，用於管理 monorepo 中項目之間的相互依賴關係。該項目還提供構建緩存和依賴可視化。
- 類型安全 API-創建一個可以在鐵匠鋪或 OpenAPI v3 中定義的 API，並管理構建時代碼生成，以允許您以類型安全的方式實現和與 API 進行交互。排出 CDK 結構，用於管理將 API 部署到 API Gateway 並配置自動輸入驗證。
- Cloudscape 反應網站-創建一個使用 Cloudscape 構建的基於反應的網站，該網站預先與 Cognito 身份驗證和（可選）您創建的 API 集成，從而使您能夠安全地調用 API。
- 基礎架構-建立一個專案，以設定部署應用程式所需的所有 CDK 相關基礎結構。它還預先配置為每次構建時基於 CDK 代碼生成圖表。
- DevOps-建立與 AWS 專案開發套件 (AWS PDK) 中的建構相容的 DevOps 工作流程。

本教學還包括如何檢視已部署的應用程式、邀請其他使用者使用該應用程式，以及使用提取請求來變更程式碼的步驟，這些請求會在合併提取請求時自動建置並部署到連線 AWS 帳戶中的資源。

當您建立由 PDK 藍圖組成的專案時，會使用專案中的下列資源來建立專案： CodeCatalyst

- 一個配置為一個單一的源存儲庫。
- 執行靜態程式碼分析和授權檢查的工作流程，以及在對預設分支進行變更時建置和部署範例程式碼。每次您對程式碼進行變更時，都會產生架構圖。
- 您可以用來規劃和追蹤工時的問題板和待處理項目。
- 具有自動報告的測試報表套裝。

主題

- 必要條件
- 第 1 步：創建一個專案

- 第 2 步：將類型安全 API 添加到項目中
- 第 3 步：為項目添加雲景反應網站
- 步驟 4：產生基礎設施以將應用程式部署到 AWS 雲端
- 步驟 5：設定 DevOps 工作流程以部署專案
- 步驟 6：確認發布工作流程並查看您的網站
- 在 PDK 專案上共同作業和重複執行

必要條件

若要建立與更新專案，您必須已完成中的作業，正在設定 CodeCatalyst如下所示：

- 有一個用於登錄的 AWS 生成器 ID CodeCatalyst。
- 屬於空間，並在該空間中為您指派 Space 管理員或超級使用者角色。如需詳細資訊，請參閱建立支援 AWS 產生器 ID 使用者的空間、管理空間使用者及空間管理員角色。
- 擁有與您的空間相關聯的 AWS 帳戶，並擁有您在註冊時建立的 IAM 角色。例如，在註冊期間，您可以選擇使用稱為 CodeCatalystWorkflowDevelopmentRole-S ***paceName*** 角色原則的角色原則來建立服務角色。該角色將具有附加唯`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。如需建立角色的步驟，請參閱為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。

第 1 步：創建一個專案

從 PDK-Monorepo 藍圖開始，創建您的單個回購代碼庫作為基礎，允許您添加額外的 PDK 藍圖。

使用 PDK-Monorepo 藍圖建立專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在主 CodeCatalyst 控台中，導覽至您要建立專案的空間。
3. 在空間儀表板上，選擇建立專案。
4. 選擇從藍圖開始。
5. 選擇 PDK-Monorepo 藍圖，然後選擇 [下一步]。
6. 在「為您的專案命名」下，輸入您要指派給專案的名稱及其相關聯的資源名稱。名稱在您的空間中必須是唯一的。

7. 在「專案資源」下，執行下列操作：

   a. 在 [主要程式設計語言] 下，選擇您要用來開發專案程式碼的語言。您可以選 TypeScript擇 Java 或 Python。

   b. 選擇代碼配置

   c. 在「來源儲存庫」文字輸入欄位中，輸入來源儲存庫的名稱，以建立新存放庫，或從現有的連結儲存庫中選取。現有的存放庫必須是空的。如需詳細資訊，請參閱 連結來源儲存庫。

   d. (選擇性) 從「Package 件管理員」下拉式選單中，選擇套件管理員。只有當您選擇 TypeScript 作為主要程式設計語言時，才需要這樣做。

8. (選擇性) 若要預覽將根據您所選取的專案參數產生的程式碼，請從 [產生專案預覽] 中選擇 [檢視程式碼]。

9. (選擇性) 從藍圖的卡片中選擇檢視詳細資料以檢視有關藍圖的特定詳細資料，例如藍圖架構概觀、所需的連線和權限，以及藍圖建立的資源類型。

10. 選擇創建項目來創建您的 monorepo 項目。創建的根級項目管理 monorepo 中項目之間的相互依賴關係，並提供構建緩存和依賴管理。

    如需專案藍圖的詳細資訊，請參閱專案藍圖參考。

PDK-Monorepo 藍圖只會產生專案的基礎。要使用藍圖創建可行的應用程序，您需要添加其他 PDK 藍圖，例如類型安全 API，Cloudscape 反應網站，基礎架構或. DevOps 在下一步中，您將應用類型安全 API 到項目。

第 2 步：將類型安全 API 添加到項目中

PDK-類型安全 API 藍圖允許您使用鐵匠鋪或 OpenAI v3 來定義 API。它會根據您的 API 定義產生執行階段套件，其中包括用於與 API 互動的用戶端，以及用於實作 API 的伺服器端程式碼。藍圖也會為每個 API 作業產生具有類型安全性的 CDK 建構。您可以將藍圖套用至現有的 PDK 龍專案，以便將 API 功能新增至專案。

若要套用 PDK-類型安全的 API 藍圖

1. 在 monorepo 專案的導覽窗格中，選擇 [藍圖]，然後選擇 [套用藍圖]。
2. 選擇 [PDK]-[類型安全 API] 藍圖，然後選擇 [下一步]。
3. 在設定藍圖下，設定藍圖參數：

   - 在模型語言下，選擇 API 模型定義的語言。
   - 在「命名空間」文字輸入欄位中，輸入 API 的命名空間。

- 在 API 名稱文字輸入欄位中，輸入 API 的名稱。
- 在 CDK 語言下，選擇您慣用的語言來編寫 CDK 基礎結構以部署 API。
- 選擇 [處理常式語言] 下拉式功能表，然後選擇要在其中實作處理常式的語言。
- 選擇 [文件格式] 下拉式功能表，然後選擇您要產生 API 文件的格式。

4. 在 [程式碼變更] 索引標籤中，檢閱提議的變更。提取請求中顯示的差異顯示了在創建提取請求時對您的項目進行的更改。
5. 如果對套用藍圖時將進行的提議變更感到滿意，請選擇套用藍圖。

   建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整體的提取請求。您可以使用@符號 (後面接著檔案名稱) 來新增資源的連結，例如檔案。

   > **Note**
   >
   > 在核准並合併提取要求之前，不會套用藍圖。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

6. 從 [狀態] 欄中，針對 PDK-[類型安全 API 藍圖] 列選擇 [擱置中] 提取要求，然後選擇開啟提取要求的連結。
7. 選擇「合併」，選擇您偏好的合併策略，然後選擇「合併」以合併套用藍圖中的變更。

   合併提取請求後，在 monorepo 項目中生成一個新`packages/apis/mypdkapi`文件夾，其中包含了配置類型安全 API 的所有 API 相關源代碼。
8. 在導覽窗格中，選擇 [藍圖] 以確認 PDK 的 [狀態-類型安全 API] 顯示 [最新]。

第 3 步：為項目添加雲景反應網站

PDK-雲景反應網站藍圖生成一個網站。您可以關聯可選參數（Type Safe API），以自動配置您的網站以設置經過身份驗證的類型安全客戶端以及交互式 API 資源管理器來測試您的各種 API。

要應用 PDK-雲景反應網站藍圖

1. 在 monorepo 專案的導覽窗格中，選擇 [藍圖]，然後選擇 [套用藍圖]。
2. 選擇 PDK-雲景反應網站藍圖，然後選擇 [下一步]。
3. 在設定藍圖下，設定藍圖參數：

   - 在網站名稱文字輸入欄位中，輸入您網站的名稱。

- 選擇「類型安全 API」下拉式功能表，然後選擇要在網站中整合的 API 藍圖。傳入 API 會設定經過驗證的用戶端，並新增必要的相依性、API 總管和其他功能。

4. 在 [程式碼變更] 索引標籤中，檢閱提議的變更。提取請求中顯示的差異顯示了在創建提取請求時對您的項目進行的更改。
5. 如果對套用藍圖時將進行的提議變更感到滿意，請選擇套用藍圖。

   建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整體的提取請求。您可以使用@符號 (後面接著檔案名稱) 來新增資源的連結，例如檔案。

   > **Note**
   >
   > 在核准並合併提取要求之前，不會套用藍圖。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

6. 從 [狀態] 欄中，針對 PDK-Cloudscape React 網站藍圖列選擇擱置中提取要求，然後選擇開啟的提取要求的連結。
7. 選擇「合併」，選擇您偏好的合併策略，然後選擇「合併」以合併套用藍圖中的變更。

   合併提取請求後，在 monorepo 項目中生成一個新`packages/websites/`*`my-website-name`*文件夾，其中包含您新網站的所有源代碼。
8. 在功能窗格中，選擇「藍圖」以確認 PDK-雲景反應網站的狀態顯示為最新狀態。

接下來，您將套用 PDK-基礎設施藍圖來產生基礎設施，以將網站部署到 AWS 雲端。

步驟 4：產生基礎設施以將應用程式部署到 AWS 雲端

PDK-基礎架構藍圖會設定包含所有 CDK 程式碼的套件，以部署您的網站和 API。它還提供了圖生成和一致性的原型 nag 包默認情況下。

若要套用 PDK-基礎架構藍圖

1. 在 monorepo 專案的導覽窗格中，選擇 [藍圖]，然後選擇 [套用藍圖]。
2. 選擇 PDK-基礎結構藍圖，然後選擇 [下一步]。
3. 在設定藍圖下，設定藍圖參數：

   - 在 CDK 語言下，選擇您要用來開發基礎架構的語言。
   - 在 [堆疊名稱] 文字輸入欄位中，輸入為藍圖產生之 CloudFormation 堆疊的名稱。

> **Note**
>
> 在下一個步驟中記下此堆疊名稱，您將在其中設定 DevOps 工作流程。

- 選擇「類型安全 API」下拉式功能表，然後選擇要在網站中整合的 API 藍圖。
- 選擇雲景反應 TS 網站下拉式選單，然後選擇您想要在基礎架構中部署的網站藍圖（例如 PDK-雲景反應網站）。

4. 在 [程式碼變更] 索引標籤中，檢閱提議的變更。提取請求中顯示的差異顯示了在創建提取請求時對您的項目進行的更改。
5. 如果對套用藍圖時將進行的提議變更感到滿意，請選擇套用藍圖。

   建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整體的提取請求。您可以使用@符號 (後面接著檔案名稱) 來新增資源的連結，例如檔案。

> **Note**
>
> 在核准並合併提取要求之前，不會套用藍圖。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

6. 從 [狀態] 欄中，針對 PDK-基礎結構藍圖列選擇擱置提取要求，然後選擇開啟提取要求的連結。
7. 選擇「合併」，選擇您偏好的合併策略，然後選擇「合併」以合併套用藍圖中的變更。

   合併提取請求之後，會在 monorepo 專案中產生一個新`packages/infra`資料夾，其中包含將您的專案部署到 AWS 雲端的基礎設施。
8. 在導覽窗格中，選擇 [藍圖] 以確認 PDK-基礎結構的 [狀態] 顯示 [最新]。

接下來，您將套用 PDK- DevOps 藍圖來部署應用程式。

步驟 5：設定 DevOps 工作流程以部署專案

PDK- DevOps 藍圖會產生使用組態中指定的 AWS 帳戶和角色建置和部署專案所需的 DevOps 工作流程。

若要套用 PDK- DevOps 藍圖

1. 在 monorepo 專案的導覽窗格中，選擇 [藍圖]，然後選擇 [套用藍圖]。
2. 選擇 PDK- DevOps 藍圖，然後選擇 [下一步]。

3. 在設定藍圖下，設定藍圖參數：

   - 在當前環境中選擇引導 CDK。
   - 在「堆疊名稱」文字輸入欄位中，輸入您要部署的 CloudFormation 堆疊名稱。這應該符合步驟 4：產生基礎設施以將應用程式部署到 AWS 雲端針對 PDK-基礎結構藍圖中設定的堆疊名稱。
   - 選擇 AWS 帳戶連線下拉式功能表，然後選擇要用於資源的 AWS 帳戶。如需詳細資訊，請參閱新增 AWS 帳戶 至空間。
   - 選擇用於部署應用程式下拉式功能表的角色，然後選擇要用於部署專案應用程式的 IAM 角色。

     > **Note**
     >
     > 建立 IAM 角色時，`SourceArn`請限制在專案設定中目前`ProjectID`找到的角色。如需詳細資訊，請參閱 了解服CodeCatalystWorkflowDevelopmentRole-*`spaceName`*務角色。

   - 選擇「地區」下拉式選單，然後選擇您要部署 monorepo 專案的區域。部署僅適用於所需 AWS 服務存在的區域。如需詳細資訊，請參閱各區域的 AWS 服務。

4. 在 [程式碼變更] 索引標籤中，檢閱提議的變更。提取請求中顯示的差異顯示了在創建提取請求時對您的項目進行的更改。

5. 如果對套用藍圖時將進行的提議變更感到滿意，請選擇套用藍圖。

   建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整體的提取請求。您可以使用@符號 (後面接著檔案名稱) 來新增資源的連結，例如檔案。

   > **Note**
   >
   > 在核准並合併提取要求之前，不會套用藍圖。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

6. 從 [狀態] 欄中，針對 PDK-基礎結構藍圖列選擇擱置提取要求，然後選擇開啟提取要求的連結。

7. 選擇「合併」，選擇您偏好的合併策略，然後選擇「合併」以合併套用藍圖中的變更。

   合併提取請求後，在 monorepo 項目中生成一個新的`.codecatalyst/workflows`文件夾。

8. 在導覽窗格中，選擇 [藍圖] 以確認 PDK 的 [狀態]- DevOps 顯示 [最新]。

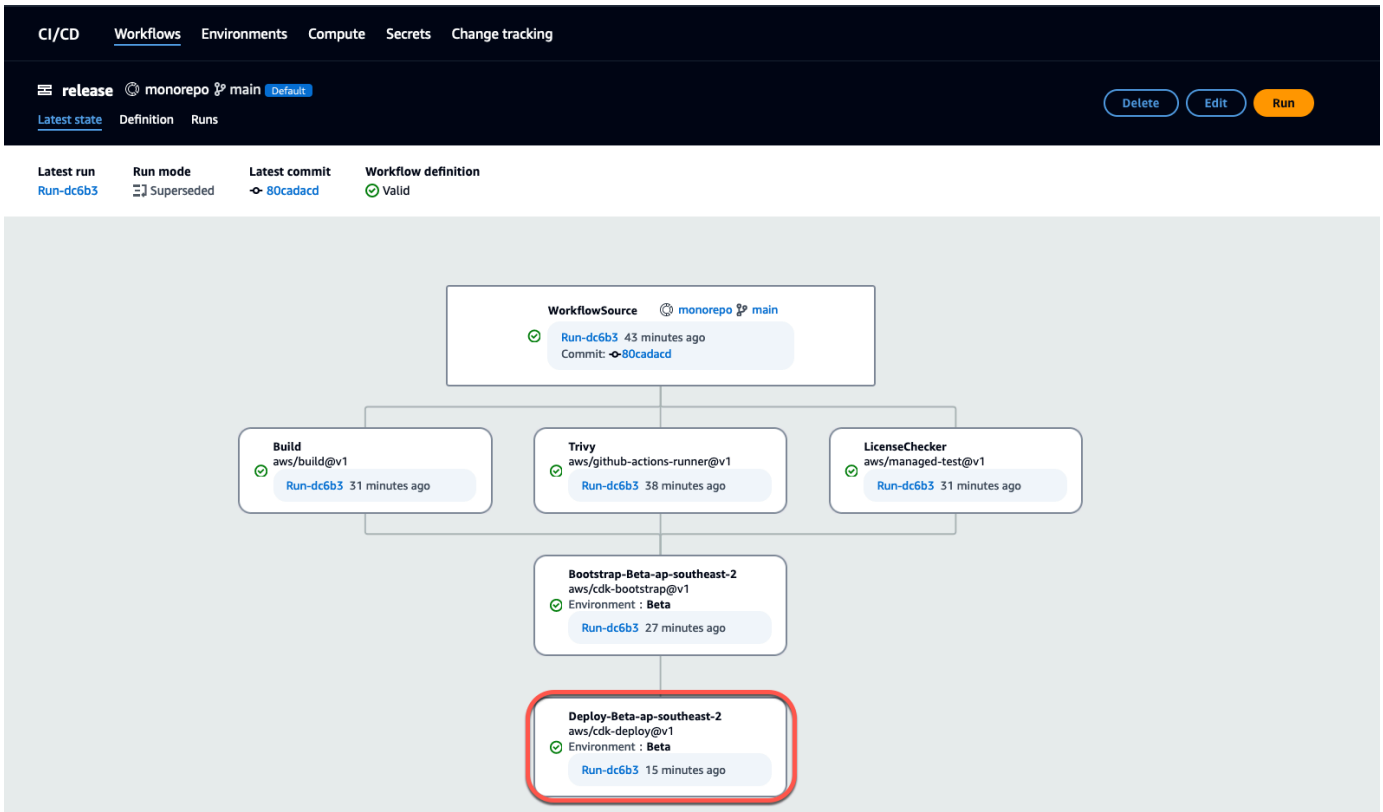
> **Note**
>
> PDK- DevOps 藍圖以及 PDK 藍圖的所有後續變更將大幅減慢，因為會產生幕後鎖定檔案，以確保 future 可重複建置和部署。它將為任何支持的語言中的所有軟件包生成鎖定文件。

步驟 6：確認發布工作流程並查看您的網站

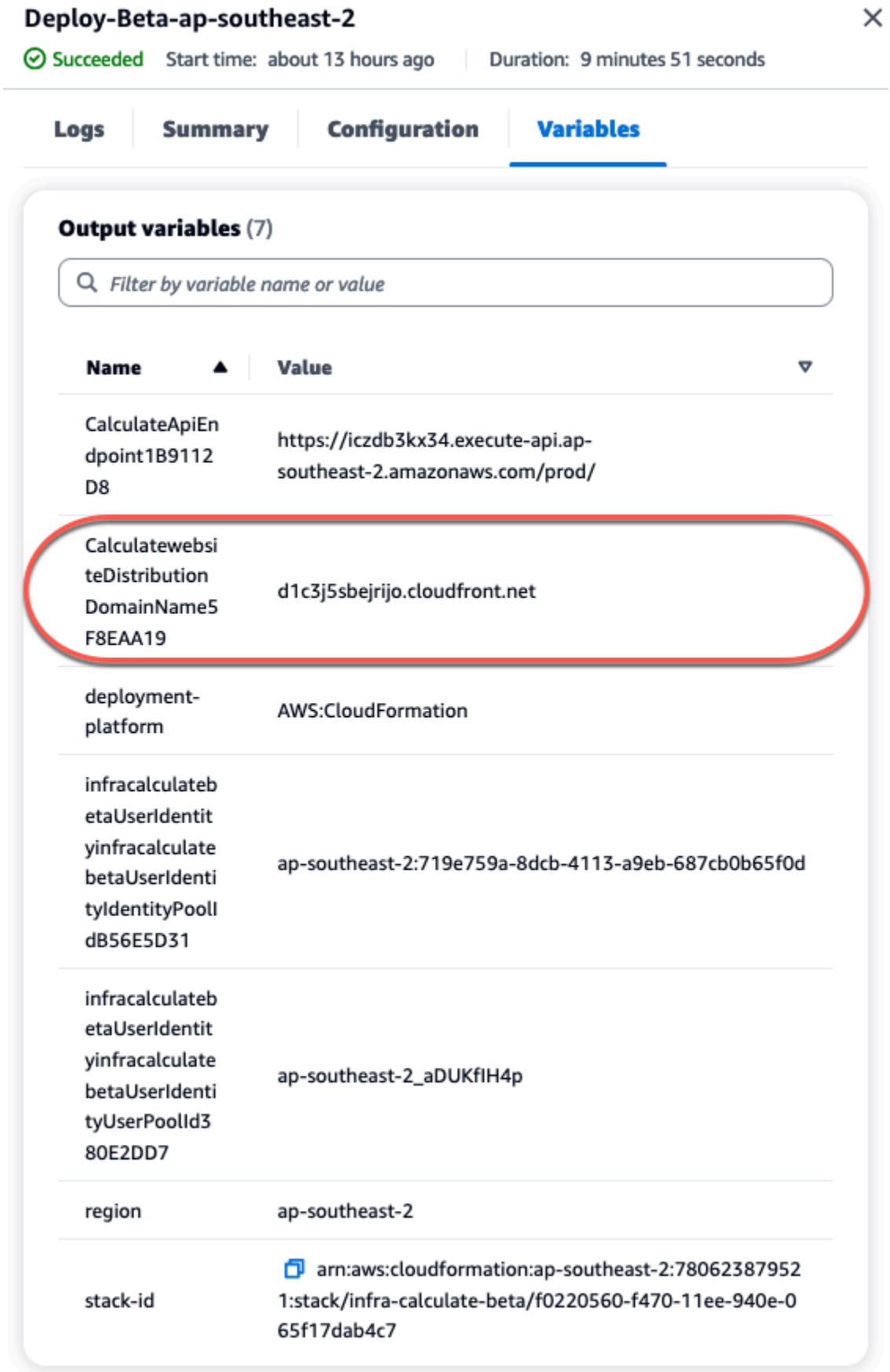
完成前面的步驟後，您可以確認發行工作流程，以確保專案正在建置中。

確認發行工作流程並檢視您的網站

1. 在 monorepo 專案的導覽窗格中，選擇 CI/CD，然後選擇「工作流程」。
2. 對於核發工作流程，請選擇最新的工作流程執行以檢視詳細資訊。如需詳細資訊，請參閱 檢視單次執行的狀態和詳細資訊。
3. 順利完成工作流程執行後，選擇工作流程中的最後一個動作 (例如，部署-B eta-ap-souteast -2，然後選擇 [變數]。



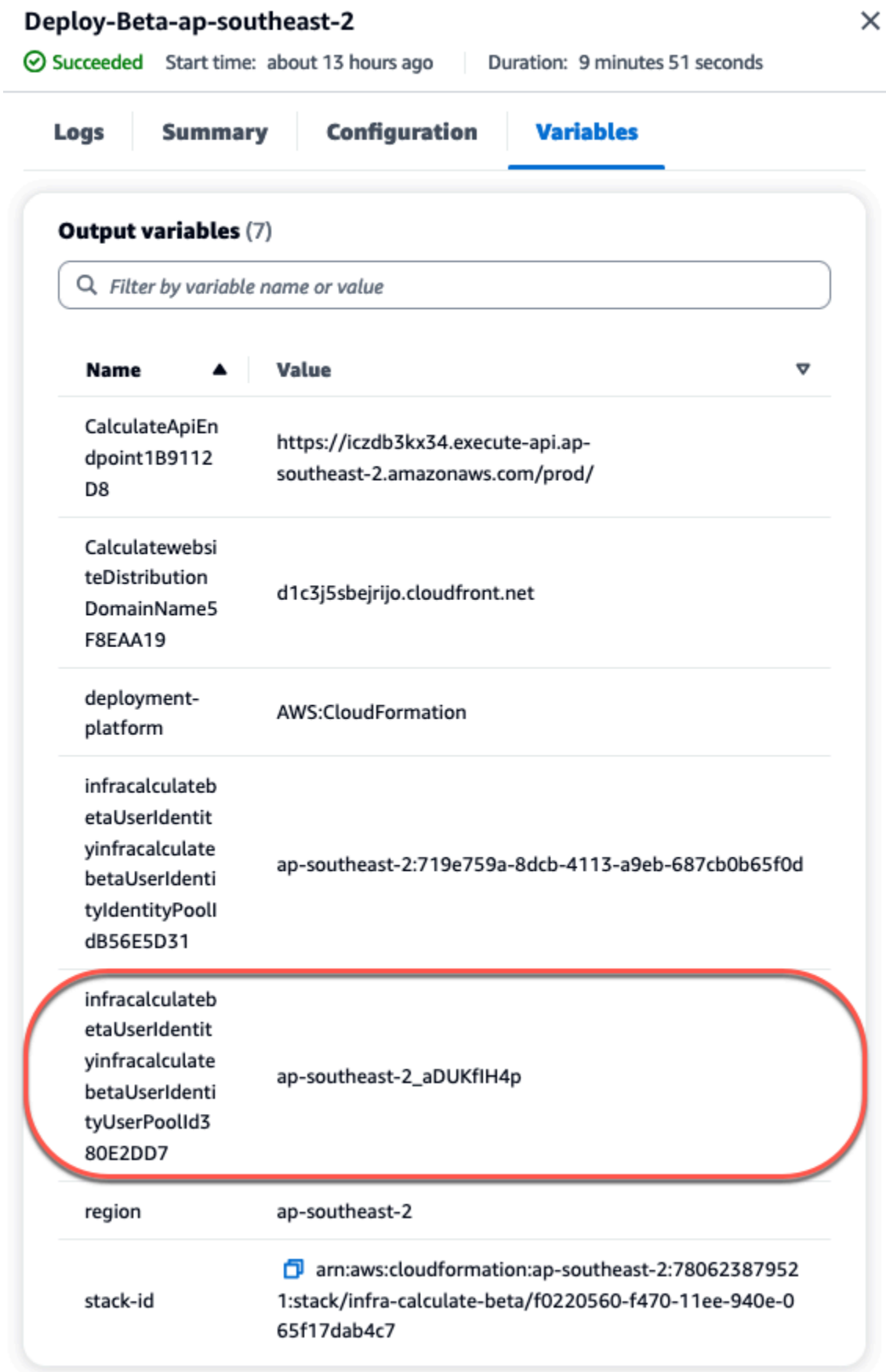
4. 將「變數」表格中的連結 (例如 ***MyPDKAPI*** websiteDistributionDomain NameXxxxx) 複製並貼上到新的瀏覽器視窗，以檢視已部署的網站。


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

您需要一個 Amazon Cognito 帳戶才能登錄到您的網站。根據預設，使用者集區不會設定為允許自我註冊。

a. 瀏覽至 AWS Cognito 主控台。

b. 從「使用者集區」表中，選擇與 PDK- DevOps 藍圖建立的使用者集區相符的「使用者集區」名稱，該名稱可在「變數」表格中找到 (例如，「基礎***###betaUserIdentityinfra#betaUserIdentity IdentityPoolId*** XXXXX」。如需詳細資訊，請參閱使用者集區入門。


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

c. 選擇 Create user (建立使用者)。

d. 設定使用者資訊參數：

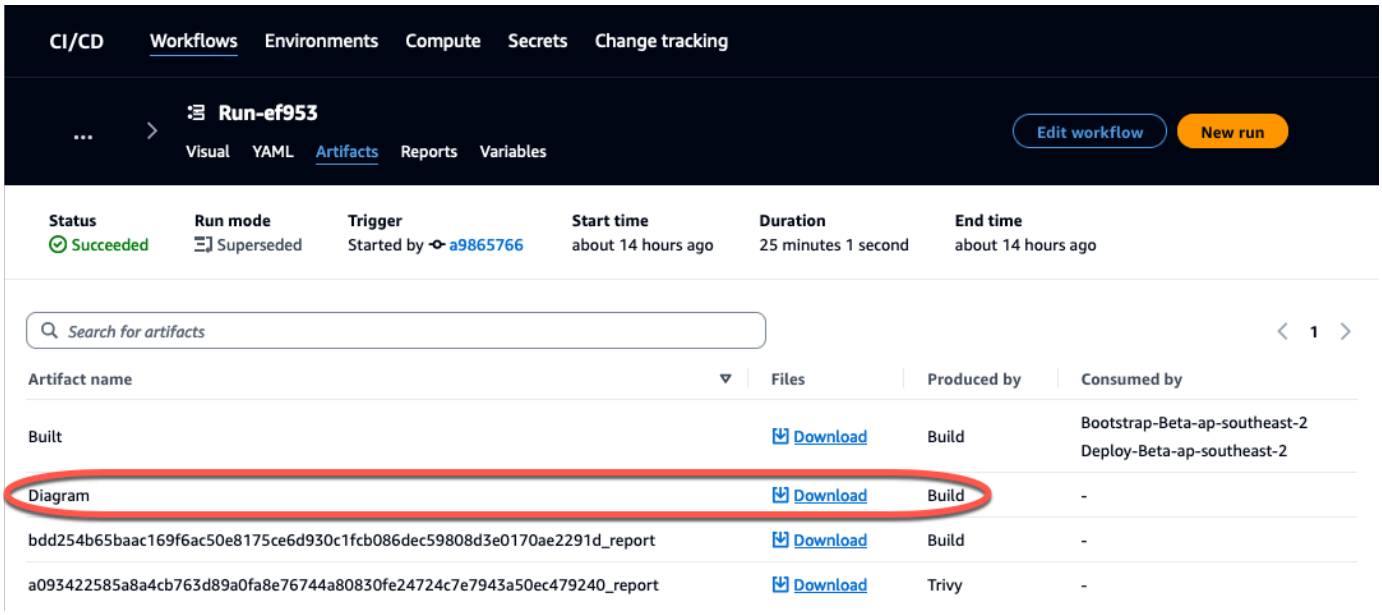
- 在「邀請訊息」下，選擇「傳送電子郵件邀請」
- 在 [使用者名稱] 文字輸入欄位中，輸入使用者名稱。
- 在 [電子郵件地址] 文字輸入欄位中，輸入使用者名稱。
- 在 [暫時密碼] 下方，選擇 [產生密碼]。

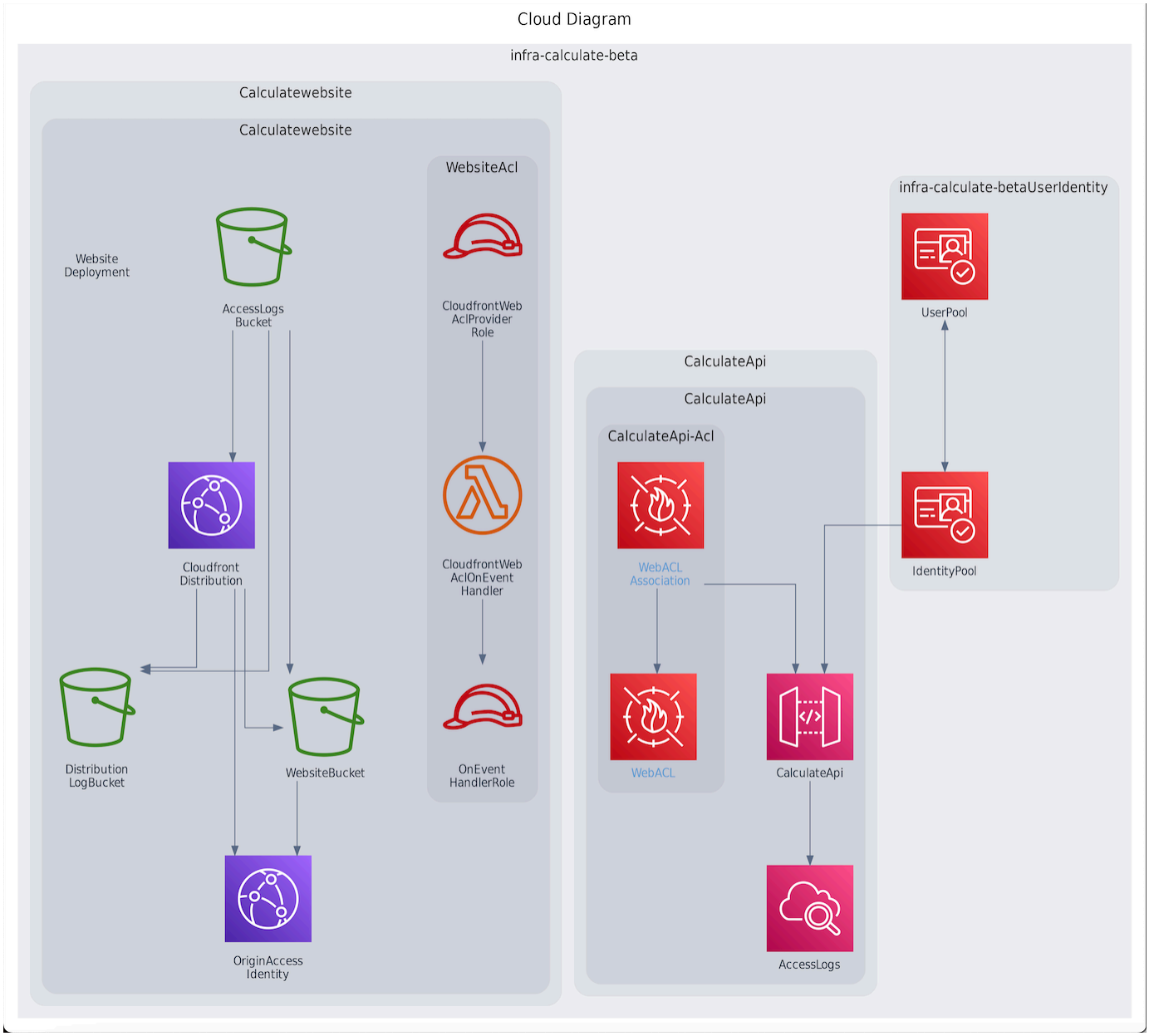
e. 選擇 Create user (建立使用者)。

f. 導航到您為用戶信息參數輸入的電子郵件帳戶，打開一封帶有臨時密碼的電子郵件。記下密碼。

g. 瀏覽回部署的網站，輸入您建立的使用者名稱和收到的暫時密碼，然後選擇 [登入]。

5. (選擇性) 順利完成工作流程執行之後，您也可以檢視產生的圖表。在中選擇「成品」標籤 CodeCatalyst，為「圖表」列選擇「下載」，然後開啟下載的檔案。





在 PDK 專案上共同作業和重複執行

專案設定完成後，您可以對原始程式碼進行變更。您也可以邀請其他 Space 成員來處理專案。PDK 藍圖可讓您反覆建置應用程式，只在需要時新增所需的應用程式，同時保留對每個藍圖組態的完整控制權。

主題

- 第 1 步：邀請成員加入您的項目
- 步驟 2：建立問題以協同合作和追蹤工作

- 步驟 3：檢視您的來源儲存庫
- 步驟 4：建立開發環境並變更程式碼
- 步驟 5：推送和合併程式碼變更

第 1 步：邀請成員加入您的項目

您可以使用主控台邀請使用者加入您的專案。您可以邀請您的空間成員，或從空間外部新增名稱。

若要邀請使用者加入您的專案，您必須以專案管理員或 Space 管理員角色登入。

您不需要邀請具有 S pace 管理員角色的使用者加入您的專案，因為他們已經具有對空間中所有專案的隱含存取權。

當您邀請使用者加入您的專案時 (未指派 S pace 系統管理員角色)，使用者將顯示在專案底下的 [專案成員] 表格中，以及顯示在 [專案成員] 表格中的空間下。

若要從 [專案設定] 索引標籤邀請成員加入您的專案

1. 導航到您的項目。

   > **Tip**
   >
   > 您可以選擇要在頂部導航欄中查看的項目。

2. 在導覽窗格中，選擇 [專案設定]。
3. 選擇「成員」頁標。
4. 在 [專案成員] 中，選擇 [邀請新成員]。
5. 輸入新成員的電子郵件地址，選擇此成員的角色，然後選擇 [邀請]。如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

從項目概述頁面邀請成員加入您的項目

1. 導航到您的項目。

   > **Tip**
   >
   > 您可以選擇要在頂部導航欄中查看的項目。

2. 選擇「成員 +」按鈕。
3. 輸入新成員的電子郵件地址，選擇此成員的角色，然後選擇 [邀請]。如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

步驟 2：建立問題以協同合作和追蹤工作

CodeCatalyst 幫助您跟踪功能，任務，錯誤以及與問題相關的項目的任何其他工作。您可以創建問題以跟踪所需的工作和想法。默認情況下，當您創建問題時，它會添加到您的積壓。您可以將問題移至追蹤進行中工作的看板。您也可以將問題指派給特定專案成員。在此步驟中，請建立問題以對 PDK 專案進行變更。

若要建立問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您要創建問題的 monorepo 項目。
3. 在專案首頁上，選擇 [建立問題]。或者，在導覽窗格中，選擇「問題」。
4. 選擇 [建立問題]。

> **Note**
>
> 您也可以在使用網格檢視時以內嵌方式新增問題。

5. 輸入問題的標題。
6. (選擇性) 輸入「說明」。針對此問題，請輸入下列說明：`a change in the src/mysfit_data.json file.`。您可以使用降價來添加格式。
7. (選擇性) 選擇問題的 [狀態]、[優先順序] 和 [估計]。
8. (選擇性) 新增現有標籤或建立新標籤，然後選擇 + 新增標籤來加入標籤。
   a. 若要新增現有標示，請從清單中選擇標示。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有標籤。
   b. 若要建立新標籤並新增標籤，請在搜尋欄位中輸入要建立的標籤名稱，然後按 Enter 鍵。
9. (選擇性) 選擇 + 新增工作負責人以新增工作負責人。您可以選擇 + 加入我，快速將自己新增為受指派人。

> **Tip**
>
> 您可以選擇將問題分配給 Amazon Q，讓 Amazon Q 嘗試解決問題。如需詳細資訊，請參閱 教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

10. (選擇性) 新增現有的自訂欄位或建立新的自訂欄位。問題可以有多個自定義字段。

    a. 若要新增現有的自訂欄位，請從清單中選擇自訂欄位。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有自訂欄位。

    b. 要創建一個新的自定義字段並添加它，請在搜索字段中輸入要創建的自定義字段的名稱，然後按 Enter 鍵。然後選擇要創建的自定義字段的類型並設置一個值。

11. 選擇 [建立問題]。右下角會顯示通知：如果問題已成功建立，便會顯示確認訊息，指出問題已成功建立。如果問題未成功建立，則會顯示錯誤訊息，其中包含失敗原因。然後，您可以選擇「重試」來編輯並重試建立問題，或選擇「捨棄」來捨棄問題。這兩個選項都會關閉通知。

> **Note**
>
> 您無法在建立提取要求時將其連結至問題。不過，您可以在建立後對其進行編輯，以新增提取請求的連結。

如需詳細資訊，請參閱 中的問題 CodeCatalyst。

步驟 3：檢視您的來源儲存庫

您可以在 Amazon 中檢視與專案關聯的來源儲存庫 CodeCatalyst。對於中的來源儲存庫 CodeCatalyst，儲存庫的概觀頁面會提供該儲存區域中資訊和活動的快速概觀，包括：

- 存放庫的描述 (如果有的話)
- 儲存庫中的分支數
- 儲存庫的開啟提取要求數目
- 儲存庫的相關工作流程數
- 預設分支或您選擇的分支中的檔案和資料夾
- 上次提交到顯示分支的標題，作者和日期
- 在降價中呈現的 README.md 文件的內容，如果包含任何 README.md 文件

此頁面也提供儲存區域認可、分支和提取要求的連結，以及開啟、檢視和編輯個別檔案的快速方法。

> **Note**
>
> 您無法在控制台中查看有關鏈接存儲庫的此信 CodeCatalyst 息。若要檢視連結儲存區域的相關資訊，請選擇儲存區域清單中的連結，以便在代管該儲存區域的服務中開啟該儲存區域。

若要導覽至專案的來源儲存庫

1. 導覽至您的專案，然後執行下列其中一項作業：
   - 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。
   - 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。在來源儲存庫中，從清單中選擇存放庫的名稱。您可以在篩選列中輸入部分存放庫名稱來篩選儲存庫清單。
2. 在儲存區域的首頁上，檢視儲存區域的內容以及相關資源的相關資訊，例如提取要求數目和工作流程。依預設，會顯示預設分支的內容。您可以從下拉式清單中選擇不同的分支來變更檢視。

> **Tip**
>
> 您也可以從專案摘要頁面中選擇 [查看專案程式碼]，快速瀏覽至專案的儲存庫。

步驟 4：建立開發環境並變更程式碼

在此步驟中，創建一個開發環境並進行代碼更改，然後將其合併到主分支中。本教學會逐步引導您完成簡單的 AWS PDK 專案，但您也可以遵循 AWS PDK GitHub 儲存庫中提供的更複雜的範例。

使用新分支建立開發環境

1. 在 monorepo 項目的導航窗格中，執行以下操作之一：
   - 選擇 [概觀]，然後瀏覽至 [我的開發環境] 區段。
   - 選擇 [程式碼]，然後選擇 [開發環境]。
   - 選擇 [程式碼]，選擇 [原始碼儲存庫]，然後選擇您要建立開發環境的 monorepo 儲存庫。
2. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
3. 選擇複製儲存庫。

4. 選擇要複製的存放庫，選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。

   > **Note**
   >
   > 如果您從 [來源儲存庫] 頁面或特定的來源儲存庫建立開發環境，則不需要選擇存放庫。開發環境將從您從「來源儲存庫」頁面選擇的來源儲存庫建立。

5. (選擇性) 在別名-選擇性中，輸入開發環境的別名。
6. (選擇性) 選擇開發環境設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
7. （可選）在 Amazon Virtual Private Cloud (Amazon VPC) 中-可選，從下拉式功能表中選取您要與開發環境建立關聯的 VPC 連線。

   如果為您的空間設置了默認 VPC，則您的開發環境將運行連接到該 VPC。您可以透過關聯不同的 VPC 連線來覆寫此項目。此外，請注意，連接 VPC 人雲端的開發環境不支援 AWS 工具組。

   > **Note**
   >
   > 當您使用 VPC 連線建立開發環境時，VPC 內會建立新的網路介面。 CodeCatalyst 使用關聯的 VPC 角色與此介面互動。此外，請確定您的 IPv4 CIDR 區塊未設定為 `172.16.0.0/12` IP 位址範圍。

8. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。

執行開發環境之後，您可以在中使用產生的範例應用程式 CodeCatalyst ，方法是變更具有提取請求的程式碼，這些請求會在合併提取請求時自動建置並部署到連線 AWS 帳戶中的資源。monorepo 提供了一個開發文件，因此所有必需的全局依賴關係和運行時都會自動出現。

若要變更專案中的程式碼

1. 在開發環境的工作終端中，導航到您的 monorepo 項目，然後通過運行以下命令來安裝項目依賴項：

   ```
   npx projen install
   ```

2. 瀏覽至定packages/apis/*mypdkapi*/model/src/main/smithy/operations/say-hello.smithy義 API 作業範例的「」。在本教程中，您將構建一個簡單的Calculate操作，將兩個數字相加在一起。變更程式碼以定義此作業，包括其輸入和輸出。

   範例：

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

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

   該@handler特徵告訴類型安全 API，您將以寫入的 AWS Lambda 處理常式的形式實作此操作 TypeScript。類型安全 API 將為此操作生成存根，供您在中 TypeScript實現。該@required特徵已添加，這意味著它將在運行時由部署的 API 網關強制執行。如需詳細資訊，請參閱 S mithy 文件。

3. 使用與程式碼變更對齊的/say-hello.smithy檔案名稱重新命名 (例如，calculate.smithy)。

4. 導覽至packages/apis/*mypdkapi*/model/src/main/smithy/main.smithy，然後變更程式碼以連接作業。您可以透過在此檔案的operations欄位中列/calculate.smithy出該Calculate作業來公開中定義的作業。

   範例：

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

use aws.protocols#restJson1
```

```
/// A sample smithy api
@restJson1
service MyPDKApi {
    version: "1.0"
    operations: [Calculate]
    errors: [
      BadRequestError
      NotAuthorizedError
      InternalFailureError
    ]
}
```

5. 執行下列命令來建置變更：

```
npx projen build
```

> **Note**
>
> 或者，您可以傳入--parallel X標誌，這將跨X核心分配構建。

由於添加了@handler特徵，因此在構建完成後會生成以下文件：

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

6. 瀏覽至packages/apis/*mypdkapi*/handlers/typescript/src/*calculate.ts*，然後變更程式碼。此檔案是針對 API 叫用的伺服器處理常式。

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

/**
 * Type-safe handler for the Calculate operation
 */
export const calculate: CalculateChainedHandlerFunction = async (request) => {
```

```
  LoggingInterceptor.getLogger(request).info('Start Calculate Operation');

  const { input } = request;

  return Response.success({
    result: input.body.numberA + input.body.numberB,
  });
};

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

7. 導航到該/packages/apis/*mypdkapi*/handlers/typescript/test/*calculate.test.ts*文件，並對代碼進行更改以更新單元測試。

   範例：

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

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

describe('Calculate', () => {
```

```
  it('should return correct sum', async () => {
    const response = await calculate({
      ...requestArguments,
      input: {
        requestParameters: {},
        body: {
          numberA: 1,
          numberB: 2
        }
      },
    });

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

});
```

8. 瀏覽至/packages/infra/main/src/constructs/apis/*mypdkapi.ts*檔案，然後變更程式碼，以便在 CDK 基礎架構中新增Calculate作業的整合。API 構造具有集成屬性，您可以在其中傳入之前添加的實施。由於您正在使用 Smithy 模型中的@handler特徵進Calculate行操作，因此可以使用已預先配置的生成 CalculateFunction CDK 構造來指向處理程序實現。

   範例：

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

/**
 * Api construct props.
 */
export interface CalculateApiProps {
```

```
  /**
   * Instance of the UserIdentity.
   */
  readonly userIdentity: UserIdentity;
}

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

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

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

```
            actions: ["execute-api:Invoke"],
            resources: ["execute-api:/*"],
          }),
          // Open up OPTIONS to allow browsers to make unauthenticated preflight
 requests
          new PolicyStatement({
            effect: Effect.ALLOW,
            principals: [new AnyPrincipal()],
            actions: ["execute-api:Invoke"],
            resources: ["execute-api:/*/OPTIONS/*"],
          }),
        ],
      }),
    });

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

9. 執行下列命令來建置變更：

```
npx projen build
```

專案完成建置後，您可以檢視已更新產生的圖表，您可以在中找到/packages/infra/main/cdk.out/cdkgraph/diagram.png。該圖顯示了該函數如何添加並連接到創建的 API。隨著 CDK 代碼被修改，這個圖表也會被更新。

現在，您可以通過將更改推送和合併到存儲庫的主分支來部署更改。

步驟 5：推送和合併程式碼變更

提交並推送您的代碼更改，然後可以將其合併到源代碼庫的主分支中。

將變更推送至特徵分支

- 通過運行以下命令將更改提交並推送到功能分支：

```
git add .
```

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

```
git push
```

推送變更會觸發針對功能分支執行的新工作流程，您可以在 CodeCatalyst 主控台中檢視該工作流程。然後，您可以創建一個提取請求，將更改合併到源存儲庫的主分支。將功能分支合併到主分支會觸發發行工作流程。您也可以將提取請求鏈接到您的問題。

若要建立提取請求並將其連結至您的問題

1. 在您的龍斷項目中，執行以下操作之一：
   - 在瀏覽窗格中，選擇 [程式碼]，選擇 [提取要求]，然後選擇 [建立提取要求]。
   - 在儲存區域首頁上，選擇 [其他]，然後選擇 [建立提取要求]。
   - 在專案頁面上，選擇 [建立提取要求]。
2. 在源存儲庫中，確保指定的源存儲庫是包含提交代碼的存儲庫。只有在您未從存放庫的主頁面建立提取要求時，才會顯示此選項。
3. 在「目標」分支中，選擇要在檢閱程式碼之後將其合併到的主要分支。
4. 在原始碼分支中，選擇包含已提交程式碼的功能分支。
5. 在提取請求標題中，輸入一個標題，以幫助其他用戶了解需要審查的內容以及原因。
6. (選擇性) 在提取要求說明中，提供問題連結或變更說明等資訊。

> **Tip**
>
> 您可以選擇 [為我撰寫說明]， CodeCatalyst 自動產生提取要求中所包含變更的描述。您可以在將自動產生的描述新增至提取請求後，對其進行變更。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱在 Amazon 中管理生成 AI 功能 CodeCatalyst。

7. 在 [問題] 中，選擇 [連結問題]，然後選擇您在中建立的問題步驟 2：建立問題以協同合作和追蹤工作。若要取消連結問題，請選擇「取消連結」圖示。

8. (選擇性) 在 [必要的複查者] 中，選擇 [新增必要的審核者]。從專案成員清單中選擇要新增的專案成員。必要的審核者必須先核准變更，才能將提取請求合併到目的地分支。

   > **Note**
   >
   > 您無法同時將審核者新增為必要的審核者和選用的審核者。您無法將自己新增為審核者。

9. (選擇性) 在 [選擇性複查者] 中，選擇 [新增選用複查者]。從專案成員清單中選擇要新增的專案成員。選用審核者不需要核准變更作為需求，才能將提取請求合併到目的地分支中。
10. 您的提取請求必須由審核者或您自己審核並合併到主要分支中。如需詳細資訊，請參閱 合併提取請求。

    當您的變更合併到來源儲存庫的主分支時，會自動觸發新的工作流程。
11. 合併完成後，您可以將問題移至「完成」。

    a. 在導覽窗格中，選擇 [問題]。

    b. 選擇建立於中的問題步驟 2：建立問題以協同合作和追蹤工作，選擇 [狀態] 下拉式清單，然後選擇 [完成]。

發行工作流程會在成功執行後部署您的應用程式，以便您檢視變更。

確認發行工作流程並檢視您的網站

1. 在 monorepo 專案的導覽窗格中，選擇 CI/CD，然後選擇「工作流程」。
2. 對於核發工作流程，請選擇最新的工作流程執行以檢視詳細資訊。如需詳細資訊，請參閱 檢視單次執行的狀態和詳細資訊。
3. 順利完成工作流程執行後，選擇工作流程中的最後一個動作 (部署-B eta-ap-souteast -2)，然後選擇 [變數]。
4. 通過將 ***MyPDKAPI*** websiteDistributionDomain NameXxxxx 行中的鏈接複製並粘貼到新的瀏覽器窗口中，以查看已部署的網站。
5. 輸入您在中建立的使用者名稱和密碼步驟 6：確認發布工作流程並查看您的網站，然後選擇 [登入]。
6. (選擇性) 測試應用程式中的變更。

   a. 選擇「POST」下拉式選單。

   b. 輸入和的兩個值`number B`，`numberA`然後選擇「執行」。

c. 確認回應內文中的結果。

隨著時間的推移，PDK 藍圖的目錄版本可能會變更。您可以將專案的藍圖變更為目錄版本，以隨時掌握最新變更。您可以在變更專案的藍圖版本之前檢視程式碼變更和受影響的環境。如需更多詳細資訊，請參閱 更新專案中的藍圖。

中的空格 CodeCatalyst

您可以建立代表您、您的公司、部門或群組的空間，並提供開發團隊可以管理專案的位置。您必須建立空間來新增專案、成員和您在 Amazon 中建立的關聯雲端資源 CodeCatalyst。

> **Note**
>
> 空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

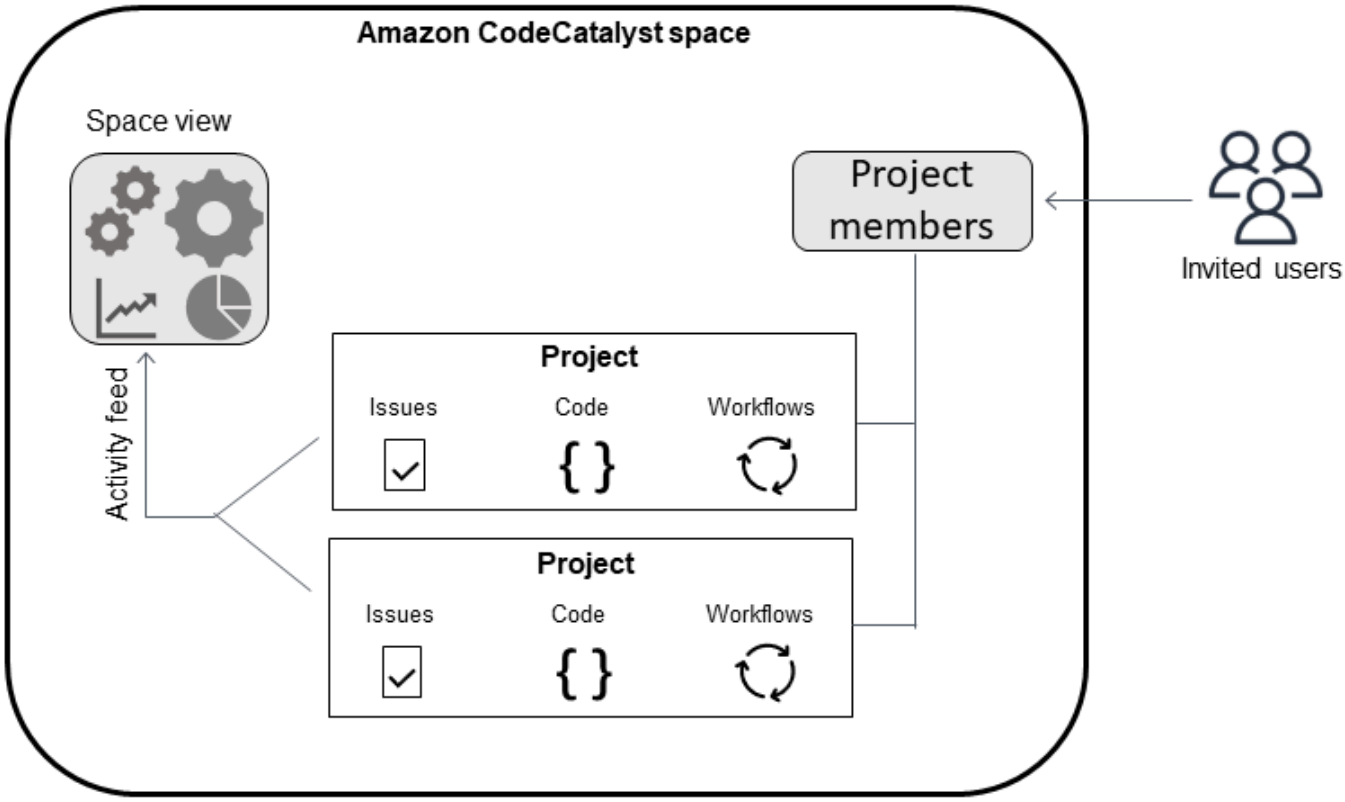
當您建立空間時，系統會自動為您指派 S pace 管理員角色。您可以將此角色新增至空間中的其他使用者。

您可以使用 S pace 管理員角色來管理空間，如下所示：

- 將其他空間管理員新增至空間
- 變更成員角色和權限
- 編輯或刪除空間
- 建立專案並邀請成員加入專案
- 檢視空間中所有專案的清單
- 檢視空間中所有專案的活動摘要

當您建立空間時，系統會自動將您新增至具有兩個角色的空間：S pace 管理員角色，以及您在建立空間時所建立之專案的專案管理員角色。當其他使用者接受專案邀請時，會自動將其新增為空間的成員。空間中的此成員資格不會授與空間中的任何權限。使用者可以在空間中執行的動作取決於使用者在特定專案中的角色。

如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。



以下是新增帳戶的其他注意事項：

- 有空間的帳戶連 AWS 帳戶 線 one-to-one 對應。單個 AWS 帳戶 可以添加到多個不同的空間。您部署的 AWS 帳戶不需要是唯一的，而且可供多個空間使用。
- AWS 帳戶 加入至 CodeCatalyst 空間可用於該空間中的任何專案。
- 雖然每個環境都可以支援多個 AWS 帳戶，但在一個動作中，您只能在每個環境中使用一個帳戶。
- 計費是在空間層級設定。可以設定多個帳戶進行計費，但一個 CodeCatalyst 空間中只能有一個帳戶處於作用中狀態。中某個空間只 AWS 帳戶 能使用一個帳單帳戶 CodeCatalyst。如果某個帳戶已用於空間，您必須使用不同的帳單帳戶作為額外空間。
- 建立連線後，如果您的工作流程必須透 AWS 過您的 CodeCatalyst 環境存取這些 IAM 角色，則必須在連線中新增 IAM 角色。如需如何使用環境的更多資訊，請參閱使用環境。

主題

- 建立支援 AWS 產生器 ID 使用者的空間
- 編輯空間
- 刪除中的空格 CodeCatalyst

- 監視空間的活動
- 管理 AWS 帳戶 空間
- 管理連線帳戶的 IAM 角色
- 管理空間使用者
- 管理團隊
- 管理機器資源
- 管理空間的開發環境
- 中空格的配額 CodeCatalyst

建立支援 AWS 產生器 ID 使用者的空間

當您第一次 CodeCatalyst 使用您的 AWS 生成器 ID 在 Amazon 註冊時，您需要創建一個空間。如需詳細資訊，請參閱 正在設定 CodeCatalyst。您可以選擇建立額外的空間來滿足您的業務需求。

> **Note**
>
> 空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

本指南中提供的資訊用於在其中 CodeCatalyst 建立支援 AWS Builder ID 使用者的空間。《管理 CodeCatalyst 員指南》中提供了設定和管理支援身份聯合之空間的步驟。若要使用為聯合身分設定的空間，請參閱 Amazon CodeCatalyst 管理員指南中的 CodeCatalyst 空間設定和管理。

若要建立支援 AWS Builder ID 使用者的其他空間，您必須被指派為 Space 管理員角色。

> **Note**
>
> 當您建立額外的空間時，系統不會提示您建立專案。若要瞭解如何在空間中建立專案，請參閱在 Amazon 創建一個項目 CodeCatalyst。

建立其他空間的步驟

1. 在中 AWS Management Console，請確定您使用您想要與您的 CodeCatalyst 空間建立關聯的相同 AWS 帳戶 登入。

2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 導覽至您的空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

4. 選擇 [建立空間]。
5. 在 [建立空間] 頁面上的 [空間名稱] 中，輸入空間的名稱。您之後無法變更此設定。

> **Note**
>
> 空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

6. 在中 AWS 區域，選擇您要儲存空間和專案資料的區域。您之後無法變更此設定。
7. 在 AWS 帳戶 ID 中，輸入您要連接到空間的帳戶的十二位數 ID。

   在AWS 帳戶驗證令牌中，複製生成的令牌 ID。Token 會自動為您複製，但您可能想要在核准 AWS 連線要求時儲存它。
8. 選擇驗證於 AWS。
9. 「驗證 Amazon CodeCatalyst 空間」頁面會在中開啟 AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登入才能存取此頁面。

   在中 AWS Management Console，請務必選擇您要建立空間的相同 AWS 區域 位置。

   要直接訪問該頁面，請登錄到 Amazon CodeCatalyst 空間 AWS Management Console 在 https://console.aws.amazon.com/codecatalyst/home/.

   驗證令牌會自動輸入驗證令牌中。成功橫幅會顯示一則訊息，指出權杖是有效的權杖。
10. 選擇驗證空間。

    會顯示「帳戶驗證」成功訊息，顯示帳戶已新增至空間。
11. 保留在驗證 Amazon CodeCatalyst 空間頁面上。選擇下列連結：若要為此空間新增 IAM 角色，請檢視空間詳細資訊。

    CodeCatalyst 空間詳細資訊頁面會在中開啟 AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。
12. 在「可用的 IAM 角色」下 CodeCatalyst，選擇「新增 IAM 角色」。

[新增 IAM 角色可供使用 CodeCatalyst] 頁面隨即顯示。

13. 選擇在 IAM 中建立 CodeCatalyst 開發管理員角色。此選項會建立包含開發角色之權限原則和信任原則的服務角色。

    開發人員角色是 AWS IAM 角色，可讓您的 CodeCatalyst 工作流程存取 AWS Amazon S3、Lambda 和 AWS CloudFormation. 該角色將具有附加唯`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。

14. 選擇 [建立開發角色]。
15. 在連線頁面的可用 IAM 角色下 CodeCatalyst，在新增至您帳戶的 IAM 角色清單中檢視開發人員角色。
16. 選擇去 Amazon CodeCatalyst。
17. 在中的建立頁面上 CodeCatalyst，選擇建立空間。

編輯空間

您可以變更空間的描述，以協助使用者更好地瞭解空間的用途。

您必須具有 S pace 管理員角色才能編輯空間詳細資料。

本指南中提供的資訊用於編輯支援 AWS Builder ID 使用者的空間。 CodeCatalyst 若要進一步了解設定和管理支援身分聯合的空間的步驟，請參閱 Amazon 管理 CodeCatalyst 員指南中的 CodeCatalyst 空間設定和管理。

編輯空間描述

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。

   > **Tip**
   >
   > 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 在空間設定索引標籤上，選擇編輯。對空間描述進行您想要的變更，然後選擇 [儲存]。

刪除中的空格 CodeCatalyst

您可以刪除空間以移除對空間所有資源的存取權。您必須具有 S pace 管理員角色才能刪除空間。

> **Note**
>
> 您無法復原刪除空間。

刪除空間後，所有空間成員將無法存取空間資源。空間資源的計費也將停止，並停止第三方來源存放庫提示的任何工作流程。

> **Note**
>
> 空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

本指南中提供的資訊可用於刪除支援 AWS Builder ID 使用者的空格。 CodeCatalyst 若要進一步了解設定和管理支援身分聯合的空間的步驟，請參閱 Amazon 管理 CodeCatalyst 員指南中的 CodeCatalyst 空間設定和管理。

刪除空間的步驟

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。

   > **Tip**
   >
   > 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 請選擇 [設定]，然後選擇 [刪除]。
4. 鍵入**`delete`**以確認刪除。
5. 選擇刪除。

   > **Note**
   >
   > 如果您屬於多個空間，系統會將您重新導向至空間概觀頁面。如果您屬於一個空間，系統會將您重新導向至空間建立頁面。

監視空間的活動

若要查看最近建立的專案和狀態更新，您可以使用 CodeCatalyst 主控台檢視顯示空間資源更新的活動摘要。

在活動摘要中，您可以檢視指標，例如工作流程執行失敗和建立的專案。

若要檢視您空間中的活動

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 選擇 Activity (活動)。
4. 檢視「活動」中的資訊。
5. 若要依活動篩選，請選擇右上角的選取器。
6. 若要檢視空間中的所有活動，請選擇 [任何活動類型]。

管理 AWS 帳戶 空間

您可以使用您 AWS 帳戶 在 Amazon CodeCatalyst 空間中的資源。若要執行此操作，您必須在中設定 AWS 帳戶 與空間之間的連線 CodeCatalyst。建立類似這樣的連線意味著您 CodeCatalyst 空間中的專案和工作流程可以與 AWS 帳戶. 您必須為每個 AWS 帳戶 要與 CodeCatalyst 空間搭配使用的連線建立一個連線。

建立連線後，您可以選擇將 AWS IAM 角色與其建立關聯。

主題

- 新增 AWS 帳戶 至空間
- 將 IAM 角色新增至帳戶連線
- 將帳戶連線和 IAM 角色新增至您的部署環境
- 檢視帳戶連線
- 從空間移除帳號 (在中 CodeCatalyst)

您可以 AWS 帳戶 通過將帳戶添加 CodeCatalyst 到您的空間來設置使用授權。通過添加 AWS 帳戶 到您的 CodeCatalyst 空間，您可以為項目工作流程提供 AWS 帳戶 資源和計費配置的訪問權限。

新增會 AWS 帳戶 建立授權 CodeCatalyst 使用此帳戶的連線。您可以使用添加 AWS 帳戶 來執行以下操作：

- 設定 CodeCatalyst 空間的帳單。請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。
- CodeCatalyst 允許假設 IAM 角色存取 AWS 資源並部署到帳戶 AWS 服務 中。請參閱管理 AWS 帳戶 空間。

透過完成授權來建立帳戶連線 AWS 帳戶。建立連線後，您可以透過新增 IAM 角色來進一步設定要使用的工作流程和專案的連線。

新增 AWS 帳戶 至空間

您可以使用 CodeCatalyst 主控台和 AWS Management Console 將您的空間連接到 AWS 帳戶.

在中新增 AWS 帳戶 空間之前 CodeCatalyst，請先完成下列先決條件：

- 建立 AWS 帳戶 並取得許可，以在您要連線的帳戶中建立 AWS IAM 角色。
- 建立要與帳戶連線建立關聯的 IAM 角色，包括具有角色許可的 IAM 政策。
- 在您要建立連線的 CodeCatalyst空間中取得 Space 管理員角色。

主題

步驟 1：建立連線要求

在 CodeCatalyst 控制台中創建連接請求會生成一個連接令牌，您可以用它來完成授權。

您必須在要建立連線的空間中具有 S CodeCatalyst pace 管理員或超級使用者角色。您還必須具有要新增 AWS 帳戶 的管理權限。

建立連線

1. 在中 AWS Management Console，請確定您使用要建立連線的相同帳戶登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 選擇 [新增] AWS 帳戶。
5. 在「 AWS 帳戶 與 Amazon 關聯 CodeCatalyst」頁面的 AWS 帳戶 ID 中，輸入您要連接到空間的帳戶的十二位數 ID。如需尋找 AWS 帳戶 ID 的相關資訊，請參閱您的 AWS 帳戶 ID 及其別名。
6. 在 Amazon CodeCatalyst 顯示名稱中，輸入帳戶的參考名稱。
7. (選擇性) 在連線說明中，輸入帳戶說明，以協助您選擇要套用帳戶和角色的專案。
8. 選擇關聯 AWS 帳戶。
9. 頁面會返回顯示成功橫幅的AWS 帳戶 詳細資訊頁面。

步驟 2：接受帳戶連接請求

在 CodeCatalyst 主控台中提交要求以連線到您的之後 AWS 帳戶，您可以與 AWS 管理員合作，透過使用提供的連線 Token 提交連線要求來接受連線要求。

請確定您擁有帳戶的系統管理員權限，而且您已使 AWS Management Console 用建立連線 AWS 帳戶的相同權限登入。

核准連線要求 (主控台)

1. 在中 AWS Management Console，請確定您使用要建立連線的相同帳戶登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 在AWS 帳戶 詳細資訊頁面上，選擇中的 [完成設定] AWS Management Console。
5. 「驗證 Amazon CodeCatalyst 空間」頁面會在中開啟 AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。

   要直接訪問該頁面，請登錄到 Amazon CodeCatalyst 空間 AWS Management Console 在 https://console.aws.amazon.com/codecatalyst/home/.

   驗證令牌會自動輸入驗證令牌中。成功訊息會顯示該權杖是有效權杖的訊息。
6. (選擇性) 在「授權付費方案」下，選擇「授權付費方案」(標準、企業)，為您的帳單帳戶開啟付費方案。

> **Note**
>
> 這不會將計費層級升級為付費層級。不過，這會設定，以 AWS 帳戶 便您可以在中隨時變更空間的計費層級。 CodeCatalyst您可以隨時開啟付費方案。如果不進行此變更，空間只能使用免費方案。

7. 選擇驗證空間。

   會顯示「帳戶驗證」成功訊息，顯示帳戶已新增至空間。

步驟 3：檢閱已核准的連線

獲得核准連線後，您可以在主控台中檢視連線，以及新增到其中的 IAM 角色。

檢閱核准的連線

1. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
2. 帳戶連線會與建立日期一起列出。
3. 選擇帳戶顯示名稱。AWS 帳戶 詳細資訊頁面隨即顯示。

步驟 4：將 IAM 角色新增至您的連線

如果您使用針對部 CodeCatalyst 署動作設定的 IAM 角色，請將該角色新增至您的部署環境。如需詳細資訊，請參閱 將 IAM 角色新增至帳戶連線。

後續步驟：為您的帳戶連線建立其他 IAM 角色

建立連線後，您可以建立要新增的其他 IAM 角色。您新增的 IAM 角色取決於您的工作流程。例如，CodeCatalyst組建動作需要 CodeCatalyst 組建角色。

若要連接您的帳戶，您需要針對您建立的角色提供 Amazon 資源名稱 (ARN)。複製您角色的 ARN，如此詳細說明。如需針對 IAM 角色使用 ARN 的詳細資訊，請參閱 Amazon 資源名稱 (ARN)。

若要存取您的 IAM 角色 ARN

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。

3. 在搜尋方塊中，輸入您要新增的角色名稱。
4. 從清單中選擇角色。

   此時會顯示角色的 [摘要] 頁面。
5. 在頂端複製角色 ARN 值。

將 IAM 角色新增至帳戶連線

建立帳戶連線的一部分包括新增要用於 CodeCatalyst 空間中專案的 IAM 角色或角色。

> **Note**
>
> 若要在帳戶連線中使用 IAM 角色，請確定已更新信任政策以使用 CodeCatalyst 服務主體。

將 IAM 角色新增至帳戶連線 (主控台)

1. 在中 AWS Management Console，請確定您使用您要管理的相同帳戶登入。
2. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 選擇帳戶連線的 Amazon CodeCatalyst 顯示名稱，然後選擇 [管理角色來源] AWS Management Console。

   將顯示「將 IAM 角色新增至 Amazon CodeCatalyst 空間」頁面。
5. 執行以下任意一項：

   - 若要建立包含開發人員角色之許可政策和信任政策的服務角色，請選擇在 IAM 中建立開 CodeCatalyst 發管理員角色。該角色將具有附加唯`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***`spaceName`***務角色。

     選擇 [建立開發角色]。
   - 若要新增您已在 IAM 中建立的角色，請選擇 [新增現有的 IAM 角色]。在選取現有 IAM 角色中，從下拉式清單中選擇角色。

     選擇 Add role (新增角色)。

頁面會在中開啟 AWS Management Console。您可能需要登錄才能訪問該頁面。

6. 在 Amazon CodeCatalyst 空間頁面導覽窗格中，選擇「空間」。

   要直接訪問該頁面，請登錄到 Amazon CodeCatalyst 空間 AWS Management Console 在 https://console.aws.amazon.com/codecatalyst/home/.

7. 選擇為您的 CodeCatalyst 空間新增的帳戶。連線頁面隨即顯示。
8. 在連線頁面的可用 IAM 角色下 CodeCatalyst，檢視新增至您帳戶的 IAM 角色清單。選擇將 IAM 角色關聯至 CodeCatalyst。
9. 在「關聯 IAM 角色」彈出式視窗的「角色 ARN」中，輸入要與您 CodeCatalyst 的空間建立關聯的 IAM 角色的 Amazon 資源名稱 (ARN)。

   在「目的」下，選擇一個角色用途，說明您要如何在帳戶連線中使用該角色。指定用RUNNER於在工作流程中執行動作的角色。指定您用來存取其他服務SERVICE的角色。

   您可以指定多個用途。

   > **Note**
   >
   > 選擇角色 ARN 的目的是必需的。

10. 選擇關聯 IAM 角色。針對其他 IAM 角色重複這些步驟。

將帳戶連線和 IAM 角色新增至您的部署環境

若要存取 AWS 資源 (例如 Amazon ECS 或用於部署的 AWS Lambda 資源)， CodeCatalyst 建立和部署動作需要具有許可的 IAM 角色才能存取這些資源。使用 S pace 管理員或超級使用者角色，您可以將 CodeCatalyst 帳戶連線 AWS 帳戶 到建立資源的位置。然後，您可以將 IAM 角色新增至您的帳戶連線。對於部署動作，您必須接著將 IAM 角色新增至 CodeCatalyst 環境。

您必須新增要用於專案部署環境的 IAM 角色。將角色新增至帳戶連線並不會將角色和連線新增至專案部署環境。若要將您的帳戶連線和 IAM 角色新增至部署環境，請確定已建立帳戶連線和角色，如中所述步驟 4：將 IAM 角色新增至您的連線。

然後，使用 CodeCatalyst 主控台中的「環境」頁面，將您的帳戶連線和 IAM 角色新增至專案中的部署環境。

> **Note**
>
> 只有當 IAM 角色用於需要 IAM 角色的 CodeCatalyst 動作時，才能將 IAM 角色新增至環境。所有需要 IAM 角色的工作動作 (包括建置動作) 都必須使用環境。 CodeCatalyst

將帳戶連線和 IAM 角色新增至部署環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至具有要新增帳戶連線和 IAM 角色的部署環境的專案。
3. 展開 CI/CD，然後選擇 [環境]。
4. 選擇您的環境，然後顯示其他標籤。
5. 選擇AWS 帳戶 連線索引標籤。在 [連線名稱] 底下，會列出已新增至環境的帳戶 (如果有的話)。
6. 選擇關聯 AWS 帳戶。將顯示「關 AWS 帳戶 聯於<environment_name>」頁面。
7. 在 [連線] 底下，選擇與您要新增之 IAM 角色的帳戶連線名稱。選擇關聯。

檢視帳戶連線

您可以檢視連線清單，並檢視有關每個連線的詳細資料。

您必須具有 S pace 管理員或超級使用者角色，才能管理您的空間連線。

檢視 CodeCatalyst 空間的所有連線

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至具有您要檢視之帳戶連線的空間。
3. 選擇「AWS 帳戶」標籤。
4. 在AWS 帳戶下，檢視空間的帳戶連線清單，包括每個連線的帳戶 ID 和狀態。

檢視帳戶連線詳細資料

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
3. 在 Amazon CodeCatalyst 顯示名稱中，選擇連接名稱。在 [詳細資料] 頁面上，檢視與連線相關聯的 IAM 角色清單以及其他詳細資料。

從空間移除帳號 (在中 CodeCatalyst)

您可以刪除不再需要的帳戶連線。對於此程序，您將使 CodeCatalyst 用刪除先前新增至空間的帳戶連線。這會從您的空間刪除帳戶連線，前提是該帳戶不是該空間的帳單帳戶。

> **Important**
>
> 刪除帳戶連線後，您就無法重新連線。您必須建立新的帳戶連線，然後根據需要建立 IAM 角色和環境關聯，或設定帳單。

即使 CodeCatalyst 空間的使用量不超過免費方案，也必須為您的空間指定帳戶。您必須先為空間新增另一個帳戶，才能移除屬於指定帳單帳戶的空間。請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。

> **Important**
>
> 雖然您可以使用這些步驟來移除帳戶，但不建議這樣做。帳戶也可能設定為支援中的工作流程 CodeCatalyst。

若要管理空間的帳戶連線，您必須具有 S pace 管理員或超級使用者角色。

刪除帳戶連線

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
3. 在 Amazon CodeCatalyst 顯示名稱下，選擇您要移除的帳戶連線旁邊的選擇器。
4. 選擇移除 AWS 帳戶。在欄位中輸入名稱以確認刪除，然後選擇 [移除]。

   會顯示成功橫幅，且帳戶連線會從連線清單中移除。

管理連線帳戶的 IAM 角色

您可以在 AWS Identity and Access Management (IAM) 中為要新增的帳戶建立角色 CodeCatalyst。如果您要新增帳單帳戶，則不需要建立角色。

在您的中 AWS 帳戶，您必須具有為要新增至空間的 AWS 帳戶 角色建立角色的權限。如需 IAM 角色和政策的詳細資訊，包括 IAM 參考和範例政策，請參閱Identity and Access Management 和 Amazon CodeCatalyst。如需有關中使用之信任原則與服務主體的詳細資訊 CodeCatalyst，請參閱CodeCatalyst 信任模型。

在中 CodeCatalyst，您必須使用 Space 管理員角色登入，才能完成將帳戶 (以及角色 (如果適用) 新增至您的空間的步驟。

您可以使用下列其中一種方法，將角色新增至您的帳戶連線。

- 若要建立包含角色之權限原則和信任原則的服務CodeCatalystWorkflowDevelopmentRole-***spaceName***角色，請參閱CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色。
- 如需建立角色和新增原則以從藍圖建立專案的範例，請參閱建立 IAM 角色並使用 CodeCatalyst信任政策。
- 如需建立 IAM 角色時要使用的範例角色政策清單，請參閱Amazon CodeCatalyst 可存取AWS資源的 IAM 角色。
- 如需建立工作流程動作角色的詳細步驟，請參閱該動作的工作流程教學課程，如下所示：
  - 教學課程：將成品上傳到 Amazon S3
  - 教學課程：使用部署無伺服器應用程式 AWS CloudFormation
  - 教學課程：將應用程式部署到 Amazon ECS
  - 教學課程：使用動作的 Lint 程式 GitHub碼

主題

- CodeCatalystWorkflowDevelopmentRole-spaceName 角色
- AWSRoleForCodeCatalystSupport 角色
- 建立 IAM 角色並使用 CodeCatalyst信任政策

CodeCatalystWorkflowDevelopmentRole-***spaceName*** 角色

您可以在 IAM 中以一鍵式角色建立開發人員角色。您必須在要新增帳戶的空間中具有 Space 管理員或超級使用者角色。您還必須具有要新增 AWS 帳戶 的管理權限。

開始下列程序之前，您必須使 AWS Management Console 用您要新增至 CodeCatalyst 空間的相同帳戶登入。否則，控制台將返回未知的帳戶錯誤。

若要建立並新增 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-***spaceName***

1. 在 CodeCatalyst 主控台中啟動之前，請開啟 AWS Management Console，然後確定您已使用相同 AWS 帳戶 的空間登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 選擇您要建立角色之 AWS 帳戶 位置的連結。AWS 帳戶 詳細資訊頁面隨即顯示。
5. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟 AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。
6. 選擇在 IAM 中建立 CodeCatalyst 開發管理員角色。此選項會建立包含開發角色之權限原則和信任原則的服務角色。該角色將具有一個名稱`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。

   > **Note**
   >
   > 此角色僅建議與開發人員帳戶搭配使用，並使用`AdministratorAccess` AWS 受管理的政策，讓其具有完整存取權，以便在其中建立新政策和資源 AWS 帳戶。
7. 選擇 [建立開發角色]。
8. 在「連線」頁面的「可用的 IAM 角色」下 CodeCatalyst**`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`***，檢視新增至您帳戶的 IAM 角色清單中的角色。
9. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

AWSRoleForCodeCatalystSupport 角色

您可以在 IAM 中以一鍵式角色建立支援角色。您必須在要新增帳戶的空間中具有 Space 管理員或超級使用者角色。您還必須具有要新增 AWS 帳戶 的管理權限。

開始下列程序之前，您必須使 AWS Management Console 用您要新增至 CodeCatalyst 空間的相同帳戶登入。否則，控制台將返回未知的帳戶錯誤。

若要建立並新增 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在 CodeCatalyst 主控台中啟動之前，請開啟 AWS Management Console，然後確定您已使用相同 AWS 帳戶 的空間登入。
2. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
3. 選擇您要建立角色之 AWS 帳戶 位置的連結。AWS 帳戶 詳細資訊頁面隨即顯示。
4. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟 AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登入才能存取此頁面。
5. 在CodeCatalyst 空間詳細資料下，選擇新增 S CodeCatalyst upport 角色。此選項會建立包含預覽開發角色的權限原則和信任原則的服務角色。該角色將具有附加唯AWSRoleForCodeCatalystSupport一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服AWSRoleForCodeCatalystSupport務角色。
6. 在 [新增 Sup CodeCatalyst port 角色] 頁面上，保持選取的預設值，然後選擇 [建立角色]。
7. 在「可用的 IAM 角色」下 CodeCatalyst**CodeCatalystWorkflowDevelopmentRole-*spaceName***，檢視新增至您帳戶的 IAM 角色清單中的角色。
8. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

建立 IAM 角色並使用 CodeCatalyst信任政策

要 CodeCatalyst 搭配 AWS 帳戶 連線使用的 IAM 角色必須設定為使用此處提供的信任政策。使用這些步驟建立 IAM 角色並附加政策，以便您從中 CodeCatalyst的藍圖建立專案。

或者，您可以建立包含角色權限原則和信任原則的服務`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色。如需詳細資訊，請參閱 將 IAM 角色新增至帳戶連線。

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. 選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 [自訂信任原則]。
4. 在 [自訂信任原則] 表單下，貼上下列信任原則。

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

5. 選擇下一步。
6. 在 [新增權限] 下，搜尋並選取您已在 IAM 中建立的自訂政策。
7. 選擇下一步。
8. 在角色名稱中，輸入角色的名稱，例如：`codecatalyst-project-role`
9. 選擇建立角色。
10. 複製角色 Amazon 資源名稱（ARN）。將角色新增至您的帳戶連線或環境時，您必須提供此資訊。

管理空間使用者

您可以透過檢視、新增、移除或變更加入空間的使用者角色來管理空間的成員。

本指南中提供的資訊適用於邀請和管理支援 AWS Builder ID 使用者 CodeCatalyst 的空間中的使用者。若要進一步了解設定和管理支援身分聯合的空間的步驟，請參閱 Amazon 管理 CodeCatalyst 員指南中的 CodeCatalyst 空間設定和管理。

檢視空間中的成員

您可以檢視空間中的使用者，包括其顯示名稱、別名以及他們對空間所擁有的角色的相關資訊。空間中的成員有三個角色：

- 空間管理員 — 此角色具有中的所有權限 CodeCatalyst，包括建立專案。僅將此角色指派給需要管理空間各個層面 (例如存取空間中所有專案) 的使用者。

  您之後必須先移除使用者，才能變更此角色。如需詳細資訊，請參閱 空間管理員角色。
- 進階使用者 — 此角色是 Amazon CodeCatalyst 空間中第二強大的角色，但無法存取空間中的專案。它是專為需要能夠在空間中建立專案並協助管理空間使用者和資源的使用者所設計。如需詳細資訊，請參閱 進階使用者角色。
- 有限的存取權 — 預設情況下，透過接受空間中專案的邀請，為加入空間的使用者指派此角色。專案成員會被指派專案中的角色。如需有關管理專案成員的資訊，請參閱管理專案成員。

S pace 管理員表格會顯示具有 S pace 管理員角色的使用者。這些使用者不會顯示在 S pace 成員中，因為這些使用者會自動 (隱含地) 指派給空間中的所有專案，而且在專案中沒有角色。

空間成員表顯示空間中具有專案角色但沒有 S pace 管理員角色的所有成員。

根據使用者是否具有在中的 S pace 管理員角色，會顯示使用者， CodeCatalyst 如下所示：

- 具有 S pace 管理員角色的使用者若稍後接受專案邀請和角色，則不會顯示在空間下的 S pace 成員表格中，或在專案底下的「專案成員」表中。它們將繼續顯示在這兩個位置的 S pace 管理員表格中。在每個專案中，所有具有 S pace 管理員角色的使用者都會顯示在該專案的專案 S pace 管理員表格中。
- 接受專案邀請以專案角色加入的使用者會新增至具有受限存取角色的空間。如果使用者的角色稍後變更為 S pace 管理員角色，但也會從「空間成員」表移至「空間管理員」表格。在專案下，使用者將從「專案成員」表移至「空間管理員」表格。

若要檢視您空間中的使用者和角色

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 請選擇 [設定]，然後選擇 [成員]。

   屬於空間成員的使用者會顯示在空間成員表格中。

> **Tip**
>
> 如果您具有 S pace 管理員角色，則可以檢視您直接受邀請加入的專案。導覽至專案的 [專案設定]，然後選擇 [我的專案]。

   在「狀態」欄中，下列為有效值：

   - 「已邀請」— 已 CodeCatalyst 發送邀請，但用戶尚未接受或拒絕。
   - 「成員」— 用戶接受了邀請。

直接邀請使用者到空間

您可以直接邀請使用者到您的 CodeCatalyst 空間。當您想要邀請該使用者協助您透過將 Space 系統管理員或 Power 使用者角色指派給他們來管理空間時，此功能非常有用。將其中一個角色指派給其他使用者可協助您將管理空間的責任分配給更多人員，而不必邀請這些使用者加入任何專案。

> **Note**
>
> 您必須具有 S pace 管理員或超級使用者角色才能邀請成員。

S pace 管理員表格會顯示具有 S pace 管理員角色的使用者。這些使用者不會顯示在 S pace 成員表中，因為這些使用者會自動 (隱含地) 指派給空間中的所有專案，而且在專案中沒有角色。

依預設，接受專案邀請的成員會新增至空間。「專案成員」(Project Members) 表格會顯示空間中具有專案角色的所有成員。

如需如何接受邀請並首次登入的詳細資訊，請參閱正在設定 CodeCatalyst。

邀請使用者加入您的空間

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。
3. 請選擇 [設定]，然後選擇 [成員]。
4. 選擇 Invite (邀請)。
5. 輸入您想邀請加入您的空間的人員的電子郵件地址。在角色中，選擇您要在空間中指派該使用者的角色。
6. 選擇邀請

取消空間邀請

如果您想要取消邀請加入您最近傳送的空間，但該空間尚未被接受，您可以取消邀請。

若要管理空間邀請，您必須具有 S pace 管理員或超級使用者角色。

取消空間成員邀請

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 請選擇 [設定]，然後選擇 [成員]。
4. 確認成員的狀態為「已邀請」。

> **Note**
>
> 您只能取消尚未接受的邀請。

5. 選擇具有受邀成員的列旁邊的選項，然後選擇 [取消邀請]。
6. 確認視窗隨即顯示。選擇 [取消邀請] 以確認。

變更空間成員的角色

您可以變更空間成員的指派角色。您必須具有 S pace 管理員角色才能變更空間中使用者的角色。

S pace 管理員表格會顯示具有 S pace 管理員角色的使用者。這些使用者不會顯示在 S pace 成員表中，因為這些使用者會自動 (隱含地) 指派給空間中的所有專案。

若要變更空間中使用者的角色

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 請選擇 [設定]，然後選擇 [成員]。
4. 在「空間成員」表中，選擇您要變更其角色的使用者。選擇 [變更角色]。

移除空間成員

當空間成員不需要存取任何空間資源時，您可以移除空間成員。您必須具有 S pace 管理員角色才能從空間中移除成員。

S pace 管理員表格會顯示具有 S pace 管理員角色的使用者。這些使用者不會顯示在 S pace 成員表中，因為這些使用者會自動 (隱含地) 指派給空間中的所有專案，而且在專案中沒有角色。您只能直接移除此表格中的空間成員。

若要從「專案成員」表中移除使用者

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 請選擇 [設定]，然後選擇 [成員]。
4. 在「專案成員」表中選擇使用者。選擇移除。

> **Note**
>
> 從空間中移除成員將從空間中的所有專案中移除該使用者，以及與這些專案中資源相關聯的權限。

移除或變更具有 S pace 管理員角色之使用者的角色

您可以移除或變更具有空間管理員角色之使用者的角色。

您必須具有 S pace 管理員角色才能從空間中移除具有 S pace 管理員角色的使用者。變更具有 S pace 管理員角色之使用者的角色基本上會從 S pace 管理員表格中移除該使用者。如果該使用者在空間中的任何專案中沒有專案角色，則從使用者移除 S pace 管理員角色會從空間中移除該使用者。

> **Note**
>
> 身為具有 S pace 管理員角色的使用者，您無法移除自己。請連絡具有 S pace 管理員角色的其他使用者。

若要從 Space 成員表格中移除具有 S pace 管理員角色的使用者

> **Note**
>
> 對於尚未明確新增至專案的使用者，他們沒有任何專案角色 (專案管理員或參與者)。如果 S pace 管理員角色是使用者唯一的角色，則會將使用者從空間中完全移除。

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您要移除或變更具有 Space 管理員角色之使用者角色的空間。
3. 請選擇 [設定]，然後選擇 [成員]。
4. 檢視成員清單的邀請狀態，並確定清單中沒有未經授權的擱置邀請空間 (狀態為「已邀請」)。

> **Important**
>
> 移除具有 S pace 系統管理員角色的使用者之前，您必須確認尚未起始擱置的邀請。

5. 選擇「成員」頁標。在空間管理員表格中，選擇使用者，然後選擇移除。

   在「移除成員」對話方塊中，執行下列其中一項作業。

   - 選擇僅移除使用者的 S pace 管理員角色的選項。選擇移除。

     > **Important**
     >
     > 如果使用者未指派任何其他角色，則從 S pace 管理員變更角色會將使用者從空間中移除。

   - 選擇從空間及其所有專案中移除具有 S pace 管理員角色的使用者的選項。選擇移除。

6. 重新整理「成員」標籤。使用者會自動新增至使用者透過專案角色具有成員資格的任何專案中的專案成員清單。如果 S pace 管理員角色是使用者唯一的角色，則會將使用者從空間中完全移除。

管理團隊

建立空間後，您可以新增團隊。團隊可讓您將使用者分組，以便他們可以共用權限並管理中的專案、問題追蹤、角色和資源 CodeCatalyst。

您必須具有 S pace 管理員角色才能管理團隊。

主題

建立團隊

團隊可以在空間中擁有角色權限，例如超級使用者。專案團隊也可以擁有專案權限，例如專案管理員。專案團隊可以與每個專案具有不同角色的許多專案相關聯。您可以管理團隊，其中小組成員是 AWS Builder ID 空間的個別使用者，或是支援身分聯合的空間 SSO 群組。

在空間和專案使用者的成員頁面上，使用者可以有多個角色。具有多個角色的使用者會在具有多個角色時顯示一個指示器，並且會先顯示具有最多權限的角色。

> **Note**
>
> 如果您的空間支援身分聯合，您必須已在 IAM 身分中心設定 SSO 使用者或 SSO 群組。

管理團隊成員的方式取決於新增和移除使用者的方式。管理專案團隊成員有兩個選項：

- 直接新增使用者 — 您可以個別新增或移除使用者。例如，您可以選擇已在 IAM 身分中心設定的 AWS Builder ID 使用者或 SSO 使用者，將使用者新增至團隊。當您選擇直接新增 AWS Builder ID 使用者或 SSO 使用者來管理小組成員時，將無法再使用 SSO 群組的選項。
- 使用 SSO 群組 — 您可以透過已在 IAM 身分中心設定的 SSO 群組來管理團隊成員。當您選擇使用 SSO 群組來管理小組成員時，將無法再使用直接新增使用者的選項。

您必須具有 S pace 管理員角色才能管理團隊。

若要建立團隊

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。
3. 選擇 [建立團隊]。
4. 在小組名稱中，輸入團隊的描述性名稱。

   > **Note**
   >
   > 團隊名稱在您的空間中必須是唯一的。

   (選擇性) 在小組說明中，輸入團隊的說明。

5. 在「空間角色」下，從您要指派給小組的 CodeCatalyst 可用空間角色清單中選擇角色。該角色將由專案團隊的所有成員繼承。

   - 空間管理員-如需詳細資訊，請參閱空間管理員角色。
   - 限制存取-如需詳細資訊，請參閱有限的存取角色。
   - 進階使用者-如需詳細資訊，請參閱進階使用者角色。

6. 在專案團隊成員資格中，選擇下列其中一項，選擇將成員新增至專案團隊的方法。

   - 選擇「直接新增成員」以個別管理使用者。這包括為空間新增 AWS 產生器 ID 使用者，或為支援身分識別聯合的空間新增 SSO 使用者。
   - 選擇 [使用 SSO 群組] 以選擇您已在 IAM 身分中心設定的 SSO 群組。

     在 SSO 群組中，選擇您要新增之群組旁邊的核取方塊。您最多可以新增五個 SSO 群組。

   > **Note**
   >
   > 您之後無法變更此設定。當您選擇直接新增 AWS Builder ID 使用者或 SSO 使用者來管理小組成員時，將無法再使用 SSO 群組的選項。當您選擇使用 SSO 群組來管理小組成員時，將無法再使用直接新增使用者的選項。

7. 選擇建立。

   > **Note**
   >
   > 當您選擇使用 SSO 群組時，請注意，SSO 群組中的使用者不會在建立小組時提取。使用者必須先登入， CodeCatalyst 才會顯示在清單中。

檢視專案團隊

在中 CodeCatalyst，您可以檢視專案團隊的專案和角色。在成員頁面上，您可以檢視專案角色和使用者清單。對於 SSO 群組類型團隊，您也可以看到與該小組相關聯的 SSO 群組清單。

若要檢視團隊

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。

3. 在 S pace 角色中，檢視為此空間指派給小組的角色。
4. 在 [專案角色] 索引標籤上，針對已新增小組為成員的空間中的每個 CodeCatalyst 專案，檢視指派給小組的專案和專案角色 (僅適用於 AWS Builder ID 空間)。
5. 在 [成員] 索引標籤上，檢視指派給專案團隊的成員清單。
6. 在 [SSO 群組] 索引標籤上，檢視指派給群組的 SSO 群組清單 (僅適用於支援身分識別聯合的空間)。

管理團隊的空間角色

團隊可以在空間中擁有角色權限，例如超級使用者。您可以變更專案團隊的空間角色，但請注意，專案團隊的所有成員都將繼承這些權限。

您必須具有 S pace 管理員角色才能管理團隊。

變更團隊的空間角色

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。
3. 在動作中，選擇變更空間角色。您可以將空間角色變更為下列其中一項。這會變更專案團隊所有成員的角色。
   - 空間管理員-如需詳細資訊，請參閱空間管理員角色。
   - 限制存取-如需詳細資訊，請參閱有限的存取角色。
   - 進階使用者-如需詳細資訊，請參閱進階使用者角色。
4. 選擇儲存。

管理專案團隊的專案角色

中 CodeCatalyst 的專案團隊與使用者類似，因為專案團隊成員可以擁有專案中的角色權限，例如「專案管理員」。角色變更將套用至專案團隊，而專案團隊的所有成員都將繼承這些權限。您可以為每個專案選擇一個會自動授與專案團隊的角色。

您必須具有 S pace 管理員角色才能管理團隊。

若要新增或變更專案角色

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。

2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。
3. 選擇 [專案角色] 索引標籤。
4. 若要變更角色，請選擇此清單中專案旁邊的選取器，然後選擇 [變更角色]。若要新增角色，請選擇 [新增專案角色]。在專案中，選擇您要新增的專案，然後在角色中選擇角色。選擇其中一個可用的專案角色：
   - 專案管理員-如需詳細資訊，請參閱專案管理員角色。
   - 貢獻者-如需詳細資訊，請參閱貢獻者角色。
   - 複查者-如需詳細資訊，請參閱「審核者」角色。
   - 唯讀-如需詳細資訊，請參閱唯讀角色。
5. 選擇儲存。

若要移除專案角色

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。
3. 選擇 [專案角色] 索引標籤。
4. 選擇您要移除的角色。

> **Important**
> 從小組移除角色會移除專案團隊中所有使用者的相關權限。

5. 選擇儲存。

直接將使用者新增至團隊

您可以將團隊成員新增至您的團隊。當您新增使用者時，新使用者將繼承專案團隊中所有現有角色的權限。

無論您的空間是為 AWS Builder ID 使用者支援還是身分聯盟設定，您都可以設定您的空間以直接新增使用者。

> **Note**
>
> 當您的空間設定為使用 SSO 群組來管理團隊成員時，無法使用直接新增使用者的選項。若要使用 SSO 群組，請參閱將 SSO 群組新增至群組。

您必須具有 S pace 管理員角色才能管理團隊。

直接新增使用者

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。
3. 選擇「成員」頁標。
4. 選擇 [新增成員]。

   > **Note**
   >
   > 新增至團隊的使用者必須已經是空間的成員。您無法新增或邀請非 Space 成員的團隊成員。

5. 在下拉式欄位中選擇使用者，然後選擇 [儲存]。選擇已在 IAM 身分中心設定的 AWS 產生器 ID 使用者或 SSO 使用者。

直接從團隊中移除使用者

您可以從團隊中移除團隊成員。使用者將不再繼承所有權限。您可以稍後將使用者新增回團隊。

> **Note**
>
> 當您移除專案團隊成員時，會從空間中的所有專案和資源中移除該使用者的關聯權限。

您必須具有 S pace 管理員角色才能管理團隊。

移除團隊成員

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。

3. 選擇「成員」頁標。
4. 選擇您要移除的使用者旁邊的選取器，然後選擇 [移除]。
5. 在輸入欄位中輸入 remove，然後選擇 [移除]。

將 SSO 群組新增至群組

如果您的空間設定為具有 IAM 身分中心管理 SSO 使用者和群組的空間，您可以新增 SSO 群組，以個別的團隊形式加入該空間。

> **Note**
>
> 當您選擇直接新增 AWS 產生器 ID 使用者或 SSO 使用者來管理小組成員時，無法使用 SSO 群組的選項。若要直接新增使用者，請參閱直接將使用者新增至團隊。

您必須具有 S pace 管理員角色才能管理團隊。

若要將 SSO 群組新增為群組

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在您空間的頁面上，選擇團隊。選擇 SSO 群組索引標籤。
3. 選擇您要新增的 SSO 群組。您最多可以新增五個 SSO 群組。

刪除群組

您可以刪除不再需要的團隊。

> **Note**
>
> 當您刪除專案團隊時，會移除空間中所有專案和資源中所有專案團隊成員的關聯權限。

您必須具有 S pace 管理員角色才能管理團隊。

刪除團隊

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間。請選擇 [設定]，然後選擇 [團隊]。

3. 在 [動作] 中選擇 [刪除小組]。這會變更整個專案團隊的角色。
4. 選擇 刪除。

管理機器資源

CodeCatalyst 透過 SSO 存取時，機器資源代表您授權資源的身分。機器資源用於將權限授與空間中的資源，例如藍圖和工作流程。您可以檢視空間中的機器資源，也可以選擇啟用或停用空間的機器資源。例如，您可能想要停用機器資源來管理存取權，然後稍後重新啟用。

在需要撤銷或停用機器資源的情況下，這些作業可用於機器資源。例如，如果您懷疑認證可能已遭到入侵，您可以停用機器資源。一般而言，這些作業不需要使用。

您必須具有 S pace 管理員角色才能檢視此頁面並管理空間層級的機器資源。

主題

檢視機器資源

您可以檢視空間中正在使用的機器資源清單。

您必須具有 S pace 管理員角色才能管理機器資源。

若要檢視機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間，然後選擇 [設定]。選擇 [機器資源]。
3. 在下拉式清單中，選擇「工作流程」動作以僅檢視工作流程的機器資源。選擇藍圖以僅檢視藍圖的機器資源。

   您也可以使用「篩選」欄位篩選名稱。

停用機器資源

您可以選擇停用空間中正在使用的機器資源。

> **Important**
>
> 停用機器資源將移除空間中所有關聯藍圖或工作流程的所有權限。

您必須具有 S pace 管理員角色才能管理機器資源。

若要停用機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間，然後選擇 [設定]。選擇 [機器資源]。
3. 選擇下列其中一項。

> **Important**
>
> 停用機器資源將移除空間中所有關聯藍圖或工作流程的所有權限。

- 若要個別停用，請選擇您要停用的一或多個機器資源旁邊的選取器。選擇 [停用]，然後選擇 [此資源]。
- 若要停用所有資源，請選擇 [停用]，然後選擇 [所有資源]。
- 若要停用所有的工作流程動作，請選擇 [停用]，然後選擇 [所有工作流程
- 若要停用所有藍圖，請選擇 [停用]，然後選擇 [所有藍圖]。

啟用機器資源

您可以選擇啟用空間中正在使用且已停用的機器資源。

您必須具有 S pace 管理員角色才能管理機器資源。

啟用機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的空間，然後選擇 [設定]。選擇 [機器資源]。
3. 選擇下列其中一項。

- 若要個別啟用，請選擇您要啟用的一或多個機器資源旁邊的選取器。選擇 [啟用]，然後選擇 [此資源]。
- 若要啟用所有資源，請選擇 [啟用]，然後選擇 [所有資源]。
- 欲啟用所有工作流程動作，請選擇 [啟用]，然後選擇 [所有工作流程動作
- 若要啟用所有藍圖，請選擇 [啟用]，然後選擇 [所有藍圖]。

管理空間的開發環境

所有開發環境都會建立為空間內專案的一部分。Space 成員可以在源存儲庫級別的項目中創建自己的開發環境。然後，空間管理員可以使用 Amazon CodeCatalyst 主控台代表空間成員檢視、編輯、刪除和停止開發環境。簡而言之，空間管理員會在空間層級維護開發環境。

管理開發環境的考量

- 您必須具有 S pace 管理員角色才能檢視 [設定] 下的 [開發環境] 頁面，並在空間層級管理開發環境。
- Space 成員透過其 CodeCatalyst 帳戶管理他們在專案中建立的開發環境。以空間管理員身分管理開發環境時，您代表空間成員維護這些資源。
- 開發環境預設為特定的計算和儲存設定。如需升級組態的帳單和費率的相關資訊，請參閱 Amazon CodeCatalyst 定價頁面。

有關開發環境的其他考量，包括停止運行的實例，默認計算配置，升級計算，產生成本以及配置超時，請參閱。開發環境 CodeCatalyst

主題

- 檢視您空間的開發環境
- 編輯空間的開發環境
- 停止空間的開發環境
- 刪除空間的開發環境

檢視您空間的開發環境

您可以檢視空間中所有開發環境的類型、狀態和詳細資料。如需建立和執行開發環境的詳細資訊，請參閱建立開發環境。

您必須具有 S pace 管理員角色才能檢視此頁面並在空間層級管理開發環境。

在您的空間中檢視開發環境

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 選擇 [設定]，然後選擇 [開發環境]。

   此頁面會列出您空間中的所有開發環境。您可以檢視資源名稱、資源別名 (如果適用)、IDE 類型、預設或已設定的計算和儲存體，以及每個開發環境的設定逾時。

編輯空間的開發環境

您可以編輯開發環境的組態，例如設定的逾時時間長度 (如果有的話)，讓閒置的開發環境停止執行。如需編輯開發環境的詳細資訊，請參閱編輯開發環境。

您必須具有 S pace 管理員角色才能檢視此頁面並在空間層級管理開發環境。

在您的空間中編輯開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 選擇 [設定]，然後選擇 [開發環境]。
4. 選擇您要管理的開發環境旁邊的選擇器。選擇編輯。
5. 對開發環境的計算或閒置逾時進行您想要的變更。
6. 選擇 Save (儲存)。

停止空間的開發環境

如果開發環境設定為逾時，您可以在執行中的開發環境閒置之前將其停止。否則，已經過時間的開發環境將會停止。如需停止開發環境的詳細資訊，請參閱停止開發環境。

您必須具有 S pace 管理員角色才能檢視此頁面並在空間層級管理開發環境。

在您的空間中停止開發環境

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 選擇 [設定]，然後選擇 [開發環境]。
4. 選擇您要管理的開發環境旁邊的選擇器。選擇停止。

刪除空間的開發環境

您可以刪除不再需要或不再擁有擁有者的開發環境。如需刪除開發環境之考量的詳細資訊，請參閱刪除開發環境。

您必須具有 S pace 管理員角色才能檢視此頁面並在空間層級管理開發環境。

若要刪除空間中的開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 選擇 [設定]，然後選擇 [開發環境]。
4. 選擇您要管理的開發環境旁邊的選擇器。選擇刪除。若要確認，請輸入`delete`，然後選擇 [刪除]。

中空格的配額 CodeCatalyst

下表說明 Amazon 中空間的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

每位使用者每個使用中的空間數 AWS 區域	五
每個使用者每月每個區域建立的空間數	五
空間描述	空間描述是可選的。如果指定，它們的長度必須介於 0 到 200 個字元之間。它們可以包含字母、數字、空格、句點、底線、逗號、破折號和下列特殊字元的任意組合： `? & $ % + = / \ ; : \n \t \r`
空間名稱	空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。 空格名稱的長度必須介於 3 到 63 個字元之間。它們還必須以字母數字字符開頭。空格名稱可以包含字母、數字、句號、底線和破折號的任意組合。它們不能包含下列任何字元： `! ? @ # $ % ^ & * ( ) + = { } [ ] | /\ > < ~ ` ' " ; :`

中的專案 CodeCatalyst

您可以使用 Amazon 中的專案 CodeCatalyst 來建立協作空間，讓開發團隊可以透過共用持續整合/持續交付 (CI/CD) 工作流程和存放庫來執行開發任務。建立專案時，您可以新增、更新或移除資源。您也可以監控團隊工作的進度。您可以在一個空間內有多個專案。

中的空間 CodeCatalyst 由專案組成。您可以查看空間中的每個專案，但只能使用您身為成員的專案。當您建立專案時，會產生專案的預設角色，您可以將這些角色指派給邀請加入專案的使用者。

- 指派給具有專案角色之專案的任何人 (例如「參與者」角色) 都可以存取專案資源，例如來源存放庫。
- 具有 S pace 管理員專案管理員或角色的任何人都可以傳送加入專案的邀請。
- 具有專案管理員角色的使用者可以跨共用資源追蹤活動、狀況和其他設定。
- 具有受限存取角色的使用者可以管理功能、程式碼修正和測試的專案指派，做為 CI/CD 工作流程的一部分。

  工作流程可用來建置、測試、發行或更新應用程式，做為 CI/CD 管線。您可以透過新增可傳輸和處理來源人工因素的動作來組合工作流程。當您執行動作時，您的專案雲端資源會用來為工作流程動作提供隨選運算能力。您可以根據要設定的活動和輸出來設定更多 CI/CD 工作流程。例如，您可以建立僅用於建置和測試動作的工作流程，您可以在其中檢視測試結果並在修正錯誤的情況下完成工作流程，而不需要部署。然後，您可以建立另一個工作流程，以建置應用程式並將其部署到測試環境。

建立專案時，您可以使用藍圖建立包含範例程式碼並建立資源的專案，也可以從空白專案開始。如果您使用藍圖建立專案，則您選擇的藍圖會決定要將哪些資源新增至專案，以及 CodeCatalyst 建立或設定的工具，以便您追蹤和使用專案資源。您可以在建立專案後手動新增或移除資源。下列資源可由以下人員建立或設定 CodeCatalyst：

- 問題 — 您可以在稱為問題的不同記錄中跟踪與項目相關的工時。您可以針對項目的功能，任務，錯誤和任何其他工作發生問題。
- 通知 — 您可以透過選擇要監視的資源、要監視的事件以及要接收通知的目的地用戶端或電子郵件來配置通知。
- 搜尋 — 您可以在專案中搜尋程式碼、問題、使用者、提取要求和套件。您可以在單一專案中進行搜尋，也可以在所有專案中進行搜尋。
- 原始碼儲存庫 — 您可以在專案的儲存庫中使用原始程式碼。當您提交原始程式碼變更或合併指定分支中的提取要求時，會 CodeCatalyst 更新您的來源。

每個專案會依使用者將專案活動追蹤為事件清單，例如建立專案或修改資源時。專案活動會在空間層級進行監控和彙總。如需使用活動資料的更多資訊，請參閱檢視所有專案。

如果您的專案使用 AWS 資源，您可以將您的 CodeCatalyst 帳戶連線到具有管理權限的 AWS 帳戶，以整合專案的資源。

您可以在建立專案之後，將來源儲存庫、問題和其他資源新增至專案。您必須具有 S pace 管理員角色才能建立專案。

在 Amazon 創建一個項目 CodeCatalyst

透過 CodeCatalyst 專案，您可以使用共用的持續整合/持續交付 (CI/CD) 工作流程和存放庫來執行開發工作、管理資源、追蹤問題並新增使用者。

建立專案之前，您必須具有 S pace 系統管理員或超級使用者角色。

主題

- 使用藍圖建立專案
- 在 Amazon 中創建一個空項目 CodeCatalyst
- 建立具有連結 GitHub儲存庫的專案

使用藍圖建立專案

您可以使用專案藍圖佈建所有專案資源和範例程式碼。如需藍圖的相關資訊，請參閱的藍圖參考。專案藍圖參考

使用藍圖建立專案

1. 在主 CodeCatalyst 控台中，導覽至您要建立專案的空間。
2. 在空間儀表板上，選擇建立專案。
3. 選擇從藍圖開始。
4. 選擇藍圖，然後選擇 [下一步]。
5. 在「為您的專案命名」下，輸入您要指派給專案的名稱及其相關聯的資源名稱。名稱在您的空間中必須是唯一的。
6. 在 [專案資源] 下，設定一般專案參數。

7. (選擇性) 若要根據您選取的專案參數來檢視含有更新的定義檔，請從「產生專案預覽」中選擇「檢視程式碼」或「檢視工作流程」。
8. (選擇性) 從藍圖的卡片中選擇檢視詳細資料以檢視有關藍圖的特定詳細資料，例如藍圖架構概觀、所需的連線和權限，以及藍圖建立的資源類型。
9. 選擇建立專案。

   如需專案藍圖的詳細資訊，請參閱專案藍圖參考。

在 Amazon 中創建一個空項目 CodeCatalyst

您可以建立沒有資源的空白專案，並在稍後手動新增您想要的資源。

建立專案之前，您必須具有 S pace 系統管理員或超級使用者角色。

建立空專案的步驟

1. 導覽至您要在其中建立專案的空間。
2. 在空間儀表板上，選擇建立專案。
3. 選擇「從頭開始」。
4. 在「為您的專案命名」下，輸入您要指派給專案的名稱。名稱在您的空間中必須是唯一的。
5. 選擇建立專案。

建立具有連結 GitHub儲存庫的專案

您可以建立連結至 GitHub 來源儲存庫的新 CodeCatalyst 專案。然後，您可以在項目中使用鏈接的 GitHub 源代碼存儲 CodeCatalyst 庫。

建立 CodeCatalyst 專案之前，您必須具有 S pace 系統管理員或超級使用者角色。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間 及 直接邀請使用者到空間 。

您必須已經有一個 GitHub 帳戶，並且您需要已經創建了要鏈接到的存儲庫。

若要在連結到您 GitHub 帳戶中 CodeCatalyst 的來源儲存庫中建立專案，您需要完成以下三項工作：

1. 安裝GitHub 儲存庫擴充功能。
2. 將您的 GitHub 帳戶 Connect 到 CodeCatalyst.

3. 建立連結至您 GitHub 帳戶的 CodeCatalyst 專案。

若要安裝GitHub 儲存庫擴充功能

1. 導覽至您要在其中建立專案的空間。
2. 在空間儀表板上，選擇建立專案。
3. 選擇 [使用自己的程式碼]。

   如果尚未安裝GitHub 儲存庫延伸功能，則會顯示安裝提示。
4. 選擇 Install (安裝)。檢閱擴充功能所需的權限，如果您要繼續，請再次選擇 [安裝]。

安裝GitHub 儲存庫擴充功能後，下一步就是將您的 GitHub 帳戶連線到您的 CodeCatalyst 空間。

> **Note**
>
> 您無法在 CodeCatalyst 專案中使用空白或封存的 GitHub 儲存庫。GitHub 儲存庫擴充功能與GitHub企業伺服器儲存庫不相容。

將您的 GitHub 帳戶連接到 CodeCatalyst

1. 在中的 [建立專案] 頁面上 CodeCatalyst，如果沒有 GitHub 帳戶連線，則會顯示提示。選擇 [Connect GitHub 帳戶] 以前往的外部網站 GitHub。
2. 使用您 GitHub 的 GitHub 憑據登錄到您的帳戶，然後選擇要安裝 Amazon 的帳戶 CodeCatalyst。

   > **Tip**
   >
   > 如果您先前已將 GitHub 帳戶連線到空間，則不會提示您重新授權。如果您是多個空間的成員或協作者，則會看到一個對話方塊，詢問您要在何處安裝擴充功能；如果您只屬於一個 GitHub 空間，則會看到 Amazon CodeCatalyst 應用程式的設定頁面。 GitHub 針對您要允許的存放庫存取設定應用程式，然後選擇 [儲存]。如果 [儲存] 按鈕未啟用，請變更組態，然後再試一次。
3. 選擇是否允許存 CodeCatalyst 取所有目前和 future 的儲存庫，或選擇要在其中使用的特定 GitHub 儲存庫 CodeCatalyst。預設選項是包含 GitHub 帳戶中的所有 GitHub 儲存庫，包括將 future 存取的儲存庫 CodeCatalyst。
4. 檢閱授予的權限 CodeCatalyst，然後選擇 [安裝]。

將您的 GitHub 帳戶連接到後 CodeCatalyst，您將能夠將該帳戶中的 GitHub 存儲庫鏈接到您的 CodeCatalyst 項目。

若要建立專案

1. 在 [建立專案] 頁面上，從GitHub帳戶下拉式功能表中，執行下列其中一個動作：
   - 選擇您已連接的 GitHub 帳戶 CodeCatalyst。
   - (選擇性) 如果您沒有看到要使用的 GitHub 帳戶，請選擇 [Connect GitHub 帳戶] 以前往中的擴充功能頁面 CodeCatalyst。如需詳細資訊，請參閱 在中使用 GitHub 儲存庫 CodeCatalyst。
2. 在GitHub 存放庫下拉式功能表中，您連線 GitHub 帳戶的 GitHub 儲存庫會顯示在下拉式清單中。選擇您要鏈接到項目的 GitHub 存儲庫。
3. 在 [命名您的專案文字輸入] 欄位中，輸入您要指派給專案的名稱。名稱在您的空間中必須是唯一的。
4. 選擇建立專案。

專案準備就緒後，您可以新增資源和任務。

- 若要瞭解使用專案建立的 CI/CD 工作流程，請參閱。開始使用中的工作流程 CodeCatalyst
- 若要使用與將建置成品部署到 Amazon S3 儲存貯體的新專案類似的建置動作，請參閱使用工作流程建置 CodeCatalyst和教學課程：將成品上傳到 Amazon S3。
- 若要從空白專案開始，並使用 AWS CloudFormation 堆疊部署來部署類似的無伺服器應用程式，請參閱教學課程：使用部署無伺服器應用程式 AWS CloudFormation。
- 若要新增「問題」規劃委員會，請參閱中的問題 CodeCatalyst。
- 若要檢視專案概觀、專案狀況、最近的專案團隊活動和指派的工作，請參閱檢視專案。
- 若要檢視原始程式碼或建立提取要求，請參閱來源儲存庫 CodeCatalyst。
- 若要設定傳送工作流程執行成功或失敗狀態警示的通知，請參閱在 Amazon 中管理通知 CodeCatalyst。
- 若要邀請成員加入您的專案，請參閱管理專案成員。
- 若要設定開發環境，請參閱開發環境 CodeCatalyst。

檢視專案

在您的 CodeCatalyst 空間中，您可以檢視具有專案權限的每個專案的詳細資訊。

若要檢視專案，您必須是專案的成員或具有該空間的 Space 管理員角色。

如果尚未建立專案，請參閱在 Amazon 創建一個項目 CodeCatalyst。您必須具有要在其中建立專案的空間的 Space 管理員角色。

- 在專案概觀中，您可以檢視專案成員、原始碼儲存庫、工作流程執行、開放式提取要求、專案開發環境和問題。
- 在項目設置下，您可以查看和管理項目詳細信息，刪除項目，邀請新成員加入項目，管理項目成員以及配置通知。

檢視專案工作和開發環境

若要檢視專案工作摘要，例如指派給您或您建立的未決問題和提取要求，以及專案關聯的開發環境，請使用主控台。

若要檢視專案，您必須是專案的成員或具有該空間的 Space 管理員角色。

若要檢視您的來源儲存庫、工作流程執行、問題、提取要求、開發環境和問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇概觀。
4. 檢視指派給您並由您建立的專案任務。

    - 查看成員 + 查看所有列表以查看項目成員列表。
    - 檢視「存放庫」卡片，以檢視與專案相關聯的來源儲存庫。
    - 檢視「工作流程」執行卡片，以檢視與專案相關聯的工作流程。
    - 檢視 Open 提取要求卡，以檢視程式碼儲存庫狀態摘要，以及指派給您並由您建立的提取要求。
    - 檢視 [我的開發環境] 卡片，以檢視與專案相關聯之開發環境的摘要。
    - 檢視「問題」卡片，以檢視指派的工作或您建立的工作摘要。

檢視所有專案

在您空間的專案清單中，您可以檢視您有權限的所有專案。

若要檢視專案工作摘要，例如指派給您或您建立的未決問題和提取要求，以及專案關聯的開發環境，請使用主控台。

若要檢視專案，您必須是專案的成員或具有該空間的 Space 管理員角色。

若要檢視您的來源儲存庫、工作流程執行、問題、提取要求、開發環境和問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇 [專案設定]。
4. 檢視專案名稱、路徑、專案 ID 和說明。

檢視專案設定

在專案設定中，您可以檢視專案成員、原始碼儲存庫、工作流程執行、開啟提取要求、專案開發環境和問題。

若要檢視專案工作摘要，例如指派給您或您建立的未決問題和提取要求，以及專案關聯的開發環境，請使用主控台。

若要檢視您的來源儲存庫、工作流程執行、問題、提取要求、開發環境和問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇 [專案設定]。
4. 檢視專案名稱、路徑、專案 ID 和說明。

變更為中的其他專案 CodeCatalyst

若要變更為其他專案，請使用主控台從您有權存取的專案清單中進行選擇。

若要變更為其他專案

1. 在 CodeCatalyst 控制台中，選擇頂部的項目選擇器。
2. 展開下拉菜單，然後選擇要導航到的項目。

刪除 Amazon 中的項目 CodeCatalyst

您可以刪除專案以移除專案資源的所有存取權。您必須具有 Space 管理員或專案管理員角色才能刪除專案。刪除專案後，專案成員將無法存取專案資源，而第三方來源儲存庫提示的任何工作流程都會停止。

若要刪除您的專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇 [專案設定]。
4. 選擇刪除專案。
5. 輸入**delete**以確認刪除。
6. 選擇刪除專案。

管理專案成員

您可以使用 Amazon CodeCatalyst 主控台管理專案中的成員。您可以新增或移除使用者、管理目前成員的角色、傳送邀請以加入專案，以及取消尚未接受的邀請。

在空間和專案使用者的成員頁面上，使用者可以有多個角色。具有多個角色的使用者會在具有多個角色時顯示一個指示器，並且會先顯示具有最多權限的角色。

檢視專案中的成員

將使用者新增至專案時，您會指派授與專案權限的角色，如下所示：

- 專案管理員角色具有專案中的所有權限。僅將此角色指派給需要管理專案各個層面的使用者，包括編輯專案設定、管理專案權限和刪除專案。如需詳細資訊，請參閱 專案管理員角色。
- 參與者角色具有在專案中工作所需的權限。將此角色指派給需要在專案中處理程式碼、工作流程、問題和動作的使用者。如需詳細資訊，請參閱 貢獻者角色。
- 「審核者」角色具有審核權限。如需詳細資訊，請參閱 「審核者」角色。
- 唯讀角色具有讀取權限。如需詳細資訊，請參閱 唯讀角色。

您不需要邀請具有 S pace 管理員角色的使用者加入您的專案，因為他們已經具有對空間中所有專案的隱含存取權。

當您邀請使用者加入您的專案時 (未指派 S pace 系統管理員角色)，使用者將顯示在專案下的 [專案成員] 表格中，以及顯示在 [專案成員] 表格中的空間下。

若要檢視您空間中的使用者和角色

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇 [專案設定]。
4. 選擇「成員」頁標。

   「專案成員」(Project Me mbers) 表格會顯示在專案中具有角色的所有成員。

   > **Tip**
   >
   > 如果您具有 S pace 管理員角色，則可以檢視您直接受邀請加入的專案。導覽至專案的 [專案設定]，然後選擇 [我的專案]。

   S pace 管理員表格會顯示具有 S pace 管理員角色的使用者。這些使用者會自動 (隱含) 指派給空間中的所有專案，且在專案中沒有角色。

   在「狀態」欄中，下列為有效值：

   - 「已邀請」— 已 CodeCatalyst 發送邀請，但用戶尚未接受或拒絕。
   - 「成員」— 用戶接受了邀請。

主題

邀請使用者加入您的專案

您可以使用主控台邀請使用者加入您的專案。您可以邀請您的空間成員，或從空間外部新增名稱。

若要邀請使用者加入您的專案，您必須以專案管理員或 Space 管理員角色登入。

您不需要邀請具有 S pace 管理員角色的使用者加入您的專案，因為他們已經具有對空間中所有專案的隱含存取權。

當您邀請使用者加入您的專案時 (未指派 S pace 系統管理員角色)，使用者將顯示在專案下的 [專案成員] 表格中，以及顯示在 [專案成員] 表格中的空間下。

若要從 [專案設定] 索引標籤邀請成員加入您的專案

1. 導航到您的項目。

   > **Tip**
   >
   > 您可以選擇要在頂部導航欄中查看的項目。

2. 在導覽窗格中，選擇 [專案設定]。
3. 選擇「成員」頁標。
4. 在 [專案成員] 中，選擇 [邀請新成員]。
5. 輸入新成員的電子郵件地址，選擇此成員的角色，然後選擇 [邀請]。如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

從項目概述頁面邀請成員加入您的項目

1. 導航到您的項目。

   > **Tip**
   >
   > 您可以選擇要在頂部導航欄中查看的項目。

2. 選擇「成員 +」按鈕。
3. 輸入新成員的電子郵件地址，選擇此成員的角色，然後選擇 [邀請]。如需角色的詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

取消邀請

如果您最近寄出邀請，只要邀請尚未被接受，您就可以取消邀請。

若要管理專案邀請，您必須具有專案管理員或 S pace 管理員角色。

若要取消專案成員邀請

1. 導覽至您已傳送要取消邀請的專案。
2. 在導覽窗格中，選擇 [專案設定]。
3. 檢視 [成員] 索引標籤，並確認該成員的狀態為 [已邀請]。

> **Note**
>
> 您只能取消尚未接受的邀請。

4. 選擇具有受邀成員的列旁邊的選項，然後選擇 [取消邀請]。
5. 確認視窗隨即顯示。選擇 [取消邀請] 以確認。

從專案中移除使用者

您可以使用主控台從專案中移除使用者。

若要從專案中移除使用者，您必須以專案管理員或 Space 管理員角色登入。

> **Note**
>
> 從空間內的所有專案中移除使用者會自動將使用者從該空間中移除。

從專案中移除使用者的步驟

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至包含您要檢視之專案的空間。在「專案」下，選擇您的專案。
3. 在導覽窗格中，選擇 [專案設定]。
4. 選擇「成員」頁標。
5. 選擇您要移除的設定檔旁邊的選取器，然後選擇 [移除]。
6. 確認您要移除使用者，然後選擇 [移除]。

接受或拒絕專案的邀請

您可能會收到加入 Amazon CodeCatalyst 專案的電子郵件邀請函。您可以接受或拒絕邀請。

接受或拒絕邀請

1. 開啟邀請電子郵件。
2. 選擇電子郵件中的專案連結。
3. 選擇接受或拒絕。

   如果您選擇「拒絕」，系統會向專案管理帳戶傳送電子郵件，通知他們您已拒絕邀請。

管理專案團隊

建立專案後，您可以新增團隊。團隊可讓您將使用者分組，以便他們可以共用權限，並以專案和空間成員的 CodeCatalyst 身分管理專案、問題追蹤、角色和資源。

您必須具有「專案管理員」角色，才能管理專案的專案團隊。

主題

將團隊新增至專案

您可以管理專案團隊成員可以存取專案中資源的專案團隊。

在空間和專案使用者的成員頁面上，使用者可以有多個角色。具有多個角色的使用者會在具有多個角色時顯示一個指示器，並且會先顯示具有最多權限的角色。

若要新增團隊

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目。選擇 [專案設定]，然後選擇 [團隊]。
3. 選擇 [新增團隊]。
4. 在團隊中，從可用的團隊清單中選擇一個團隊。
5. 在「專案角色」下，從中的可用專案角色清單中選擇角色 CodeCatalyst。

   - 專案管理員 — 如需詳細資訊，請參閱專案管理員角色。

- 貢獻者 — 如需詳細資訊，請參閱貢獻者角色。
- 審閱者 — 如需詳細資訊，請參閱「審核者」角色。
- 唯讀 — 如需詳細資訊，請參閱唯讀角色。

6. 選擇 [新增團隊]。

管理專案團隊的專案角色

團隊可以在空間中擁有角色權限，例如超級使用者。您可以變更專案團隊的空間角色，但請注意，專案團隊的所有成員都將繼承這些權限。

若要新增或變更專案角色

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。選擇 [專案設定]，然後選擇 [團隊]。
3. 若要變更角色，請選擇此清單中專案團隊旁邊的選取器，然後選擇 [變更角色]。若要新增角色，請選擇 [新增專案角色]。在專案中，選擇您要新增的專案，然後在角色中選擇角色。選擇其中一個可用的專案角色：

   - 專案管理員-如需詳細資訊，請參閱專案管理員角色。
   - 貢獻者-如需詳細資訊，請參閱貢獻者角色。
   - 複查者-如需詳細資訊，請參閱「審核者」角色。
   - 唯讀-如需詳細資訊，請參閱唯讀角色。

4. 選擇儲存。

移除專案團隊的專案角色

在中 CodeCatalyst，您可以檢視專案團隊的專案角色。您也可以檢視專案團隊中的成員。您可以移除專案團隊的專案角色。

若要移除專案角色

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的空間。選擇 [專案設定]，然後選擇 [團隊]。
3. 選擇 [專案角色] 索引標籤。
4. 選擇您要移除的角色。

> **Important**
>
> 從小組移除角色會移除專案團隊中所有使用者的相關權限。

5. 選擇 Save (儲存)。

管理機器資源

CodeCatalyst 透過 SSO 存取時，機器資源代表您授權資源的身分。機器資源用於將權限授與專案中的資源，例如藍圖和工作流程。您可以檢視專案中的機器資源，也可以選擇啟用或停用專案的機器資源。例如，您可能想要停用機器資源來管理存取權，然後稍後重新啟用。

在需要撤銷或停用機器資源的情況下，這些作業可用於機器資源。例如，如果您懷疑認證可能已遭到入侵，您可以停用機器資源。一般而言，這些作業不需要使用。

您必須具有 S pace 管理員角色或專案管理員角色，才能檢視此頁面並在專案層級管理機器資源。

主題

檢視機器資源

您可以檢視專案中正在使用的機器資源清單。

您必須具有 S pace 管理員角色或專案管理員角色。

若要檢視機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的專案，然後選擇 [專案設定]。選擇 [機器資源]。
3. 在下拉式清單中，選擇「工作流程」動作以僅檢視工作流程的機器資源。選擇藍圖以僅檢視藍圖的機器資源。

   您也可以使用「篩選」欄位篩選名稱。

停用機器資源

您可以選擇停用專案中正在使用的機器資源。

> **Important**
>
> 停用機器資源將移除空間中所有關聯藍圖或工作流程的所有權限。

您必須具有 S pace 管理員角色或專案管理員角色。

若要停用機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的專案，然後選擇 [專案設定]。選擇 [機器資源]。
3. 選擇下列其中一項。

   > **Important**
   >
   > 停用機器資源將移除空間中所有關聯藍圖或工作流程的所有權限。

   - 若要個別停用，請選擇您要停用的一或多個機器資源旁邊的選取器。選擇 [停用]，然後選擇 [此資源]。
   - 若要停用所有資源，請選擇 [停用]，然後選擇 [所有資源]。
   - 若要停用所有的工作流程動作，請選擇 [停用]，然後選擇 [所有工作流程
   - 若要停用所有藍圖，請選擇 [停用]，然後選擇 [所有藍圖]。

啟用機器資源

您可以選擇啟用專案中正在使用且已停用的機器資源。

您必須具有 S pace 管理員角色或專案管理員角色。

啟用機器資源

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的專案，然後選擇 [專案設定]。選擇 [機器資源]。

3. 選擇下列其中一項。

   - 若要個別啟用，請選擇您要啟用的一或多個機器資源旁的選取器。選擇 [啟用]，然後選擇 [此資源]。
   - 若要啟用所有資源，請選擇 [啟用]，然後選擇 [所有資源]。
   - 欲啟用所有工作流程動作，請選擇 [啟用]，然後選擇 [所有工作流程動作
   - 若要啟用所有藍圖，請選擇 [啟用]，然後選擇 [所有藍圖]。

以下項目的配額 CodeCatalyst

下表說明 Amazon 中專案的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

每個空間的最大項目數	100
使用者可以屬於的專案數目上限	1,000
可以屬於專案的成員數目上限。	10,000
專案名稱	專案名稱在空間中必須是唯一的。名稱必須介於 3 到 63 個字元之間。名稱區分大小寫。專案名稱必須以英數字元開頭。有效字元：A-Z、a-z、0-9、空格和.、_ (底線)-(連字號) 專案名稱不能包含下列任何字元：！ ？ @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
專案說明	專案描述最多可包含 200 個字元。有效字元：A-Z、a-z、0-9、空格和.、_ (底線)-(連字號)。專案說明為選擇性。

使用中的通知 CodeCatalyst

您可以設定通知來監視中的專案和資源 CodeCatalyst。使用者可以在任何專案中選擇他們想要接收電子郵件的專案事件 (他們是其成員)。您也可以選擇在團隊訊息應用程式 (例如 Slack) 中設定傳送給整個團隊的通知，方法是設定 CodeCatalyst 空間與 Slack 工作區之間的存取權限，然後設定要傳送至該

Slack 工作區中一或多個通道的專案通知。一旦您設定了 CodeCatalyst 空間和 Slack 工作區之間的存取權限，專案成員也可以選擇新增自己的 Slack 成員 ID，以便他們可以直接收到有關連線 Slack 工作區和頻道中 CodeCatalyst 事件的通知。

> **Note**
>
> 可傳送給 Slack 的專案事件集與使用者可以選擇在電子郵件中收到通知的事件集不同。

主題

通知如何運作？

您可以將專案設定為向團隊訊息應用程式 (例如 Slack) 提供通知。

通知需要哪些權限？

任何專案成員都可以設定、檢視、更新或刪除中頻道的通知設定 CodeCatalyst。但是，只有具有 Space 管理員角色的使用者才能新增或刪除 Slack 工作區。所有使用者都可以針對他們所屬的專案設定他們想要接收電子郵件的專案事件 CodeCatalyst。

我可以設定哪些 CodeCatalyst 事件的通知？

您可以設定 CodeCatalyst 為向一或多個 Slack 通道傳遞有關工作流程事件的通知。在 CodeCatalyst 專案和 Slack 之間設定通知後，專案使用者可以選擇新增自己的 Slack 成員 ID，以便在 Slack 頻道中接收有關事件的直接訊息。 CodeCatalyst 新增 Slack 成員 ID 的使用者將會在為其專案設定的 Slack 頻道中收到對其 ID 的直接提及資訊，有助於提高他們關心的事件的知名度。

您也可以選擇要接收電子郵件的活動。這些電子郵件會傳送至為您的 AWS Builder ID 設定的電子郵件地址。

通知如何浮出水面？

您可以設定 CodeCatalyst 為向一或多個 Slack 通道傳送通知。您需要授權才 CodeCatalyst 能授予存取 Slack 工作區的權限。提供授權後，即 CodeCatalyst 可將通知傳送至您設定的 Slack 通道。如果專

案成員選擇新增其 Slack 成員 ID，他們可以在針對該專案配置的 Slack 頻道中收到有關 CodeCatalyst 事件的提及。

如何設定通知？

電子郵件通知設定為的一部分 CodeCatalyst。專案使用者可以在「我的設定」 頁面中選擇他們想要接收電子郵件的事件。

若要為專案資源設定 Slack 通知，您必須完成下列高階工作。

若要設定通知 (高階工作)

1. 在中 CodeCatalyst，您可以設定 CodeCatalyst 和訊息用戶端之間的連線，例如 Slack。連接 Slack 工作區後，該工作區將可供空間中的所有專案使用。

   > **Note**
   >
   > 只有具有 Space 管理員角色的使用者才能新增或刪除 Slack 工作區。

2. 在您的專案中 CodeCatalyst，新增您希望團隊接收通知的頻道。
3. 在中 CodeCatalyst，您可以開啟各種事件的通知，例如工作流程執行失敗，並指定要將其傳送到哪個通道。

如需詳細步驟，請參閱開始使用 Slack 通知。

一旦在 CodeCatalyst 空間和 Slack 之間配置了通知，用戶可以選擇添加自己的 Slack 成員 ID，以接收有關為其項目配置的 Slack 頻道中的 CodeCatalyst 事件的直接消息，

開始使用 Slack 通知

建立專案後，您可以設定 Slack 通知，協助您的團隊監控專案資源。

這些步驟會引導您首次設定 Slack 通知。 CodeCatalyst如果您已設定通知，請參閱在 Amazon 中管理通知 CodeCatalyst。

> **Note**
>
> 可傳送至通知通道的專案事件集與使用者可以選擇在電子郵件中收到通知的事件集不同。如需更多詳細資訊，請參閱 在 Amazon 中管理通知 CodeCatalyst。

主題

先決條件

開始之前，您必須準備好以下事項：

- 一個 CodeCatalyst 空間。如需有關建立 CodeCatalyst 空間和首次登入的資訊，請參閱正在設定 CodeCatalyst。
- 一個 CodeCatalyst 項目。如需詳細資訊，請參閱在 Amazon 創建一個項目 CodeCatalyst。
- 具有專案管理員或 Space 管理員角色的 CodeCatalyst 帳戶。如需詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。
- 可由存取的 Slack 帳戶和 Slack 工作區。 CodeCatalyst
- Slack 頻道 CodeCatalyst 會在其中傳送通知。該頻道可以是公開的或私人的。

第 1 步：Connect CodeCatalyst 到您的 Slack 工作區

只有具有 S pace 管理員角色的使用者才能新增或刪除 Slack 工作區。新增或刪除 Slack 工作區會影響空間中的所有專案。若要建立 CodeCatalyst 和 Slack 之間的連線，請與 Slack 工作區 CodeCatalyst 執行安全的 OAuth 驗證交握。

請使用下列指示連線 CodeCatalyst 至 Slack 工作區。

> **Note**
>
> 這只需要為每個 Slack 工作區完成一次。然後，您可以通過 Slack 頻道設置通知。

若要連線 CodeCatalyst 至您的 Slack 工作區

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 導航到您的項目。
3. 在導覽窗格中，選擇 [專案設定]。
4. 選擇「通知」標籤。
5. 選擇 [設定通知]。
6. 選擇 Connect 到 Slack 工作區。
7. 閱讀對話方塊內容，然後選擇「Connect 至 Slack」工作區。
8. 在 AWSChatbot 消息中：

   a. 在右上角，選擇包含頻道的 Slack 工作區。

   b. 選擇 Allow (允許)。

   您將返回 CodeCatalyst 主控台。
9. 繼續進行步驟 2：將您的 Slack 頻道添加到 CodeCatalyst。

步驟 2：將您的 Slack 頻道添加到 CodeCatalyst

您需要 Slack 頻道 ID 才能新增頻道。 CodeCatalyst

取得您的 Slack 頻道 ID

1. 登入至鬆弛。如需詳細資訊，請參閱登入 Slack。
2. 前往 Slack 工作區，其中包含您希望通知傳送的頻道。如需詳細資訊，請參閱在 Slack 工作區之間切換或登入其他 Slack 工作區。
3. 在功能窗格中，開啟您要傳送通知的頻道的內容 (按一下滑鼠右鍵) 選單，然後選擇 [開啟頻道詳細資料]。

   通道 ID 會顯示在對話方塊的底部。
4. 複製通道識別碼值。下一個步驟將需要此值。

使用剛剛複製的頻道 ID，您現在可以將 Slack 頻道連接到 CodeCatalyst。

若要將您的 Slack 頻道新增至 CodeCatalyst

1. 在開始之前，如果您的 Slack 頻道為私人頻道，請將 AWS Chatbot 應用程序添加到頻道，如下所示：

a. 在 Slack 頻道的訊息方塊中，輸入**@aws**並從對話方塊中選擇 aws 應用程式。

b. 按 Enter。

會出現 Slackbot 訊息，表示 AWS Chatbot 不在私人頻道中。

c. 選擇邀請他們邀請 AWS Chatbot 加入頻道。

2. 在 CodeCatalyst 主控台中，選擇 [下一步]。
3. 在「通道 ID」中，貼上您先前取得的 Slack 通道 ID。
4. 在頻道名稱中，輸入名稱。我們建議您使用 Slack 頻道名稱。
5. 選擇下一步。
6. 在選取通知事件中，選擇您要接收通知的事件類型。
7. 選擇 Finish (完成)。

步驟 3：測試從 Slack CodeCatalyst 到 Slack 的通知

將專案設定為傳送工作流程狀態通知後，您可以在 Slack 中檢視通知。

在 Slack 中檢視您的通知

1. 在 CodeCatalyst 專案中，手動啟動工作流程，以便完成工作流程執行，並在執行完成時收到狀態通知。
2. 在 Slack 中，檢視您設定的通知頻道。您的通知會顯示每次工作流程執行的最新狀態，以及失敗或成功。

步驟 4：後續步驟

為您的 CodeCatalyst 空間配置 Slack 工作區後，您可以添加其他 Slack 渠道現有 CodeCatalyst 項目，並在創建新項目後將其添加到新項目中。您也可以讓專案使用者知道他們可以為其 Slack 成員 ID 設定個人 Slack 通知，並設定他們將接收電子郵件的事件。如需詳細資訊，請參閱 在 Amazon 中管理通知 CodeCatalyst。

在 Amazon 中管理通知 CodeCatalyst

您可以設定 CodeCatalyst 為傳送專案中事件的相關通知。您可以傳送通知給訊息用戶端，例如 Slack 頻道。項目用戶可以選擇通過發送到為其配置文件配置的電子郵件地址的電子郵件來通知他們的項目事件。

> **Note**
>
> 可傳送至通知通道的專案事件集與使用者可以選擇在電子郵件中收到通知的事件集不同。

主題

管理直接傳送給您的通知

您可以選擇向您傳送有關您身為成員之任何專案中事件的電子郵件通知。這些電子郵件將發送到您的 AWS Builder ID 中配置的電子郵件地址。默認情況下，您將收到有關可以發送電子郵件的所有項目事件的電子郵件。

如果專案已設定為傳送通知至 Slack 頻道，您可以新增 Slack 會員 ID，以接收有關該 Slack 頻道中 CodeCatalyst 事件的直接提及資訊。這有助於提高您對在您擔任角色的項目中發生的事件的認識。

若要設定專案事件的電子郵件通知

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。 CodeCatalyst 我的設定」 頁面隨即開啟。

   > **Tip**
   >
   > 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

3. 在 [電子郵件通知] 中，在清單中尋找您要設定電子郵件通知的專案，然後選擇 [編輯]。
4. 選取您要接收電子郵件的事件，然後選擇 [儲存]。

若要設定個人 Slack 通知

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。 CodeCatalyst 我的設定」 頁面隨即開啟。

> **Tip**
>
> 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

3. 在「個人鬆弛」通知中，選擇「Connect Slack ID」，然後選擇「Connect 至 Slack 工作區」。將打開一個單獨的窗口。

> **Tip**
>
> 除非具有 S pace 管理員角色的使用者已為您 CodeCatalyst的空間新增 Slack 工作區，否則無法設定此選項。如需詳細資訊，請參閱 開始使用 Slack 通知 及 管理傳送至通道的通知。

4. 在權限要求視窗中，確定工作區的名稱與為 CodeCatalyst 空間配置的 Slack 工作區相符。選擇允許以允許AWS Chatbot存取工作區。視窗隨即關閉，而 Slack 工作區會將連結狀態顯示為「已連線」。

> **Tip**
>
> 如果連線狀態未變更，請檢查連線 Slack 工作區是否發生錯誤。您可能必須向上捲動才能看到錯誤。

5. 若要停止接收個人 Slack 通知，請選擇已連接的 Slack 工作區，然後選擇 [中斷連線 S lack ID]。

管理傳送至通道的通知

您可以選擇在 CodeCatalyst 傳送至小組資源 (例如 Team Slack 頻道) 中新增和管理有關專案事件的通知。如此一來，您可以協助確保整個團隊都知道重要事件，例如工作流程執行失敗時。

> **Note**
>
> 專案的任何成員都可以管理傳送至該專案通道的通知。但是，只有具有 S pace 管理員角色的使用者才能新增或刪除 Slack 工作區。

新增專案的通知通道

您可以新增要接收通知的頻道，例如團隊的 Slack 頻道。

新增通知的 Slack 頻道

1. 如果您要新增第一個 Slack 頻道，請改開始使用 Slack 通知為參閱。

   設定第一個頻道後，請返回此程序以設定其他頻道。
2. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
3. 導航到您的項目。
4. 在導覽窗格中，選擇 [專案設定]。
5. 選擇「通知」標籤。
6. 選擇 Add channel (新增頻道)。
7. 選擇 [選擇工作區]，然後選取包含您要傳送通知的頻道的 Slack 工作區。

   如果您的 Slack 工作區不在清單中，您可以依照中開始使用 Slack 通知的指示加入它。
8. 在輸入頻道 ID 之前，如果您要新增的 Slack 頻道是私人的，請完成以下步驟：

   a. 在 Slack 頻道的訊息方塊中，輸入**@aws**並從彈出視窗中選擇 aws 應用程式。

   b. 按 Enter。

      會出現 Slackbot 訊息，表示 AWS Chatbot 不在私人頻道中。

   c. 選擇邀請他們邀請 AWS Chatbot 加入頻道。
9. 在 CodeCatalyst的「通道識別碼」欄位中，輸入 Slack 通道識別碼。若要尋找 ID，請移至 Slack，然後在導覽窗格中，以滑鼠右鍵按一下頻道，然後選擇 [開啟頻道詳細資料]。

   通道 ID 會顯示在對話方塊的底部。
10. 在頻道名稱中，輸入名稱。我們建議使用 Slack 頻道名稱。
11. 在選取通知事件中，選擇您要接收通知的事件類型。
12. 選擇 Add (新增)。

編輯通知頻道的通知

您可以變更前往哪些頻道的通知，也可以完全關閉特定通知。

編輯通知

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導航到您的項目。
3. 在導覽窗格中，選擇 [專案設定]。
4. 選擇「通知」標籤。
5. 選擇 [編輯通知]。
6. 執行下列任意一項：
   - 若要傳送通知至特定頻道，請從下拉式清單中選擇頻道。
   - 若要全域關閉通知，請選擇通知旁的切換開關。
   - 若要停止向特定頻道傳送通知，請選擇頻道上的 X。
7. 選擇 儲存 。

移除頻道

若要移除頻道

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導航到您的項目。在導覽窗格中，選擇 [專案設定]。
3. 在 [專案設定] 頁面上，選擇 [通知] 索引標籤。
4. 選擇您要移除的頻道旁邊的指示器，然後選擇 [移除頻道]。出現提示時，在確認視窗中選擇「確定」。

中的藍圖 CodeCatalyst

藍圖是代表 CodeCatalyst 專案架構元件的任意程式碼產生器。該組件可以包含從單個文件中的工作流程到整個項目的完整示例代碼的任何內容。藍圖採用任意一組選項，並使用這些選項來生成一組任意輸出代碼，該代碼被轉發到項目中。隨著藍圖更新為最新的最佳做法或新選項，它可以在包含該藍圖的專案中重新產生程式碼庫的相關部分。

您可以使用 Amazon CodeCatalyst 藍圖建立包含來源儲存庫、範例原始程式碼、CI/CD 工作流程、建立和測試報告，以及整合式問題追蹤工具的完整專案。 CodeCatalyst 藍圖會根據組態參數集產生資源和原始程式碼。使用 CodeCatalyst-managed 藍圖時，您選擇的藍圖會決定要新增至專案的資源，以及 CodeCatalyst 建立或設定的工具，以便您追蹤和使用專案資源。身為藍圖使用者，您可以使用藍圖建立專案或將其套用至現有 CodeCatalyst 專案。您可以在專案中套用多個藍圖，而且每個藍圖都可以做為獨立元件套用。例如，您可以擁有使用 Web 應用程式藍圖建立的專案，然後稍後再套用安全性藍圖。當其中一個藍圖更新時，您可以透過生命週期管理將變更或修正納入專案中。如需詳細資訊，請參閱 專案藍圖參考 及 以藍圖使用者身分使用生命週期管理。

身為藍圖作者，您還可以建立和發佈自訂藍圖，讓 S CodeCatalyst pace 成員使用您的專案資源。您可以開發自訂藍圖，以滿足您空間專案的特定需求。將自訂藍圖新增至空間的藍圖目錄後，您可以管理藍圖並繼續進行更新，以便您空間的專案以最新的最佳做法保持最新狀態。如需詳細資訊，請參閱 使用中的自訂藍圖 CodeCatalyst。若要檢視藍圖 SDK 和範例藍圖，請參閱開放原始碼存放 GitHub 庫。

主題

- 使用藍圖建立專案
- 在專案中套用和取消關聯藍圖
- 更新專案中的藍圖
- 編輯專案中藍圖的描述
- 以藍圖使用者身分使用生命週期管理
- 專案藍圖參考
- 使用中的自訂藍圖 CodeCatalyst
- 中藍圖的配額 CodeCatalyst

使用藍圖建立專案

您可以使用 Amazon CodeCatalyst 目錄中的藍圖或包含自訂藍圖的團隊空間目錄快速建立專案。根據藍圖，您的專案是使用特定資源建立的。如需詳細資訊，請參閱 使用藍圖建立專案 及 專案藍圖參考。

建立專案後，您可以使用自訂藍圖，從 CodeCatalyst 目 CodeCatalyst 錄或空間目錄將其他藍圖套用至專案。藍圖代表架構元件，因此可以在專案中同時使用多個藍圖，以整合團隊的最佳實務。這也使您能夠確保您的項目是最新的與不斷發展的組件的最新更改。若要深入瞭解如何在專案中使用藍圖，請參閱以藍圖使用者身分使用生命週期管理。

在專案中套用和取消關聯藍圖

您可以在專案中套用多個藍圖，以合併功能元件、資源和控管。您的專案可以支援在不同藍圖中獨立管理的各種元素。將藍圖套用至專案可減少手動建立資源並使軟體元件正常運作的需求。您的專案也可以隨著需求的發展保持最新狀態。如果您的專案不再需要藍圖中的資源，您可以取消藍圖與專案的關聯。若要進一步瞭解如何在專案中套用藍圖，請參閱以藍圖使用者身分使用生命週期管理。

主題

- 將藍圖套用至專案
- 取消藍圖與專案的關聯

將藍圖套用至專案

若要將藍圖套用至您的專案

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在 CodeCatalyst 主控台中，導覽至空間，然後選擇要套用藍圖的專案。
3. 在導覽窗格中，選擇 [藍圖]，然後選擇 [套用藍圖]。
4. 從藍圖索引標籤選擇CodeCatalyst 藍圖或從空間藍圖索引標籤中選擇自訂藍圖，然後選擇下一步。
5. 在藍圖詳細資料下，從「目標版本」下拉式功能表中選擇藍圖版本。系統會自動選取最新版本。
6. 在設定藍圖下，設定藍圖參數。
7. 檢閱目前藍圖版本與更新版本之間的差異。提取請求中顯示的差異顯示了當前版本和最新版本之間的更改，這是提取請求創建時所需的版本。如果未顯示任何變更，表示版本可能相同，或者您可能已為目前版本和所需版本選擇相同的版本。
8. 如果您滿意提取要求包含您要檢閱的程式碼和變更，請選擇 [套用藍圖]。建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整個提取請求。您可以使用@符號後跟檔案名稱，將資源連結新增為檔案。

> **Note**
>
> 在核准並合併提取要求之前，不會套用藍圖。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

藍圖作者也可以將自訂藍圖套用至指定空間中沒有藍圖可用來建立新專案或套用至現有專案的專案。如需詳細資訊，請參閱 在指定的空間和專案中發佈和套用自訂藍圖。

取消藍圖與專案的關聯

如果您不想要來自藍圖的新更新，可以取消藍圖與專案的關聯。從藍圖新增至專案的資源和功能性軟體元件將在您的專案中主要。

取消藍圖與專案的關聯

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在 CodeCatalyst 主控台中，導覽至空間，然後選擇要取消藍圖關聯的專案。
3. 在導覽窗格中，選擇 Blueprints (藍圖)。
4. 選擇具有您要取消關聯之資源的藍圖，選擇 [動作] 下拉式功能表，然後選擇 [取消關聯藍圖]。
5. 輸入`confirm`以確認解除關聯。
6. 選擇確認。

更新專案中的藍圖

如果您使用藍圖建立專案或將藍圖套用至現有專案，則會收到有關藍圖新版本的通知。在透過核准的提取要求更新藍圖版本之前，您可以檢視程式碼變更和受影響的環境。生命週期管理可讓您更新專案中的一或多個套用藍圖，以便更新每個藍圖，而不會影響專案的其他區域。您也可以覆寫藍圖更新。如需詳細資訊，請參閱 以藍圖使用者身分使用生命週期管理。

將藍圖更新為最新版本

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在主 CodeCatalyst 控台中，導覽至要更新藍圖版本的空間。
3. 在空間儀表板上，選擇具有您要更新之藍圖的專案。

4. 在導覽窗格中，選擇 [藍圖]，然後選擇要更新之藍圖的圓形按鈕。
5. 選擇 [作業] 下拉式功能表，然後選擇 [更新版本]。
6. 從「目標版本」下拉式選單中，選擇您要用於更新的版本。系統會自動選取最新版本。
7. 在設定藍圖下，設定藍圖參數。
8. 檢閱目前藍圖版本與更新版本之間的差異。提取請求中顯示的差異在於當前版本和最新版本之間的更改，這是創建提取請求時所需的版本。如果沒有顯示任何變更，表示版本可能相同，或者您可能已經為目前版本和所需版本選擇了相同的版本。
9. 如果您滿意提取要求包含您要檢閱的程式碼和變更，請選擇 [套用更新]。建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整個提取請求。您可以使用@符號 (後面接著檔案名稱) 來新增資源的連結，例如檔案。

> **Note**
>
> 在核准並合併提取要求之前，藍圖不會更新。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

> **Note**
>
> 如果您已開啟現有的提取要求以更新藍圖版本，請先關閉先前的提取要求，然後再建立新的提取要求。當您選擇 [更新版本] 時，系統會將您導向至藍圖的擱置提取要求清單。您也可以從專案 [設定] 和 [專案摘要] 頁面的 [藍圖] 索引標籤中檢視擱置的提取要求。如需更多詳細資訊，請參閱 檢視提取要求。

編輯專案中藍圖的描述

您可以編輯用於建立專案或在建立專案後套用的藍圖描述。一個藍圖可以在專案中使用多次。若要區分專案中藍圖的用途，您可以使用這些藍圖的描述。描述也可用於識別您要從特定藍圖套用的元件。

若要在專案中編輯藍圖的描述

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在 CodeCatalyst 主控台中，導覽至您的空間，然後選擇具有您要更新之藍圖設定的專案。
3. 在導覽窗格中，選擇 Blueprints (藍圖)。
4. 選擇具有您要更新之描述的藍圖，選擇 [動作] 下拉式功能表，然後選擇 [編輯說明]。

5. 在藍圖描述文字輸入欄位中，輸入說明以識別專案中的藍圖。
6. 選擇 Save (儲存)。

以藍圖使用者身分使用生命週期管理

生命週期管理是能夠從更新的藍圖選項或版本重新產生程式碼庫。這可讓藍圖作者集中管理包含特定藍圖之每個專案的軟體開發生命週期。例如，將安全性修正程式推送至 Web 應用程式藍圖，將允許包含 Web 應用程式藍圖或從 Web 應用程式藍圖建立的每個專案自動取得該修正程式。此相同的管理架構也可讓您身為藍圖使用者，在選取藍圖選項後變更藍圖選項。

主題

對現有專案使用生命週期管理

您可以針對從藍圖建立的專案或未與任何藍圖相關聯的現有專案使用生命週期管理。例如，您可以將標準安全性做法藍圖新增至從未從藍圖建立的 five-year-old Java 應用程式。藍圖會產生安全性掃描工作流程和其他相關程式碼。現在，任何時候對藍圖進行變更，Java 應用程式中的程式碼庫部分現在都會自動與團隊的最佳做法保持在最新狀態。

在專案中的多個藍圖上使用生命週期管理

由於藍圖代表架構元件，因此通常可以在同一個專案中一起使用多個藍圖。例如，專案可以由公司平台工程師建置的中央 Web API 藍圖組成，以及應用程式安全性團隊建置的版本檢查藍圖。這些藍圖中的每一個都可以獨立更新，並記住過去套用到它們的合併解決方案。

> **Note**
>
> 作為任意架構元件，並非所有藍圖都可以在一起合併或邏輯上一起工作，即使它們仍然會嘗試相互合併。

處理生命週期提取請求中的衝突

有時，生命週期提取請求可能會產生合併衝突。這些問題可以手動解決。在後續的藍圖更新中會記住解決方案。

選擇退出生命週期管理變更

使用者可以從專案中移除藍圖，以取消與藍圖的所有參考資料的關聯，並選擇退出生命週期更新。基於安全理由，這不會移除或影響專案的任何程式碼或資源，包括從藍圖中新增的內容。如需詳細資訊，請參閱 取消藍圖與專案的關聯。

覆寫專案中藍圖的生命週期管理

如果您想要覆寫專案中特定檔案的藍圖更新，可以在存放庫中包含擁有權檔案。GitLab的程式碼擁有者規格是建議的準則。藍圖始終尊重代碼所有者文件而不是其他所有內容，並且可以生成如下所示的示例：

```
new BlueprintOwnershipFile(sourceRepo, {
      resynthesis: {
        strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied accross all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: MergeStrategies.neverUpdate,
            globs: [
              '**/src/**',
              '**/css/**',
            ],
          },
        ],
      },
    });
```

這將生成一個.ownership-file包含以下內容：

```
[dont-override-sample-code] @amazon-codecatalyst/blueprints.import-from-git
# This strategy is applied accross all sample code. The blueprint will create sample
 code, but skip attempting to update it.
# Internal merge strategy: neverUpdate
**/src/**
```

```
**/css/**
```

專案藍圖參考

使用藍圖建立專案時， CodeCatalyst 會建立包含來源儲存庫、範例原始程式碼、CI/CD 工作流程、建置和測試報告，以及整合式問題追蹤工具的完整專案。專案藍圖會使用程式碼為不同類型的應用程式和架構佈建雲端基礎結構、資源和範例來源成品。

如需詳細資訊，請參閱 在 Amazon 創建一個項目 CodeCatalyst。您必須是 Space 管理員才能建立專案。

主題

- 可用藍圖
- 尋找專案藍圖資訊

可用藍圖

藍圖名稱	藍圖描述
核心網頁 API	此藍圖會建立 .NET 6 核心網頁 API 應用程式。藍圖使用 .NET 的 AWS 部署工具， AWS App Runner並提供設定 Amazon 彈性容器服務或 AWS Elastic Beanstalk 做為部署目標的選項。
AWS Glue	此藍圖使用 AWS CDK、 AWS Glue、 AWS Lambda 和 Amazon Athena 建立範例擷取轉換載入 (ETL) 參考實作，將逗號分隔值 (CSV) 轉換為 Apache 實作地板。
DevOps 部署管線	此藍圖使用部署管線參考架構建立 AWS 部署管線，該架構可 AWS 跨多個階段部署參考應用程式。
與 Java 的 API AWS Fargate	此藍圖會建立容器化 Web 服務專案。該專案使用AWS 副駕駛員 CLI 在 Amazon ECS 上建立和部署由 Amazon DynamoDB 支援的容器化春季啟動 Java Web 服務。專案會將容器化應用

藍圖名稱	藍圖描述
	程式部署到無伺服器運算上的 Amazon ECS 叢集。 AWS Fargate 此應用程式會將資料儲存在 DynamoDB 表格中。工作流程成功執行之後，範例 Web 服務即可透過應用程式負載平衡器公開取得。
現代化的三層 Web 應用程式	該藍圖為應用程序層和 Vue 前端框架以 Python 生成代碼，以構建和部署架構良好的 3 層現代 Web 應用程序。
NET 無伺服器應用程式	此藍圖會使用 .NET CLI Lambda 工具建立 AWS Lambda 函數。藍圖提供 AWS Lambda 函式的選項，包括 C# 或 F# 的選項。
Node.js 應用程式介面 AWS Fargate	此藍圖會建立容器化 Web 服務專案。該項目使用AWS 副駕駛 CLI 在 Amazon 彈性容器服務上構建和部署容器化 Express/Node.js Web 服務。專案會將容器化應用程式部署到無伺服器運算上的 Amazon ECS 叢集。 AWS Fargate 工作流程成功執行之後，範例 Web 服務即可透過應用程式負載平衡器公開取得。
無伺服器應用程式模型 (SAM)	此藍圖會建立使用無伺服器應用程式模型 (SAM) 來建立和部署 API 的專案。您可以選擇 SDK for Java TypeScript，或者為 Python 開發套件作為程式設計語言。
無伺服器影像處理常	此藍圖可建立高速影像處理的應用程式，而不會降低影像品質。
無伺服器靜態微服務	此藍圖會建立使 AWS Lambda 用「待辦事項」服務參考的 REST API。 Amazon API Gateway 您可以選擇 SDK for Java TypeScript，或者為 Python 開發套件作為程式設計語言。

藍圖名稱	藍圖描述
單頁應用	此藍圖創建一個使用反應，Vue 和角框架的單頁應用程序（SPA）。對於託管，請選擇 AWS Amplify 託 Amazon CloudFront 管或 Amazon S3。
靜態網站	該藍圖使用 Hugo 或 Jekyll 靜態站點生成器創建一個靜態網站。靜態網站產生器會使用文字輸入檔案 (例如 Markdown) 來產生靜態網頁。它們非常適合罕見變化，信息豐富的內容，例如產品頁面，文檔和博客。藍圖會使用 AWS CDK 將靜態網頁部署到 AWS Amplify 或 Amazon S3 + CloudFront。
要做網絡應用程序	此藍圖會建立具有前端和後端元件的無伺服器 Web 應用程式。您可以選擇 SDK for Java TypeScript，或者為 Python 開發套件作為程式設計語言。
V ideo-on-demand 網絡服務	此藍圖會建立一項 video-on-demand 服務，以提供接收、轉碼和交付內容的能力。藍圖使用 AWS Lambda Amazon CloudWatch、Amazon S3 和 AWS Elemental MediaConvert.
訂閱外部藍圖	此藍圖會為每個匯入的封裝建立工作流程。這些工作流程每天執行一次，以檢查 NPM 是否有新版本的套件。如果存在新版本，工作流程會嘗試將其作為自訂藍圖新增至您的 CodeCatalyst 空間。如果找不到封裝或不是藍圖，則動作將失敗。目標套件必須位於 NPM 上，且套件必須是藍圖。空間必須在支援自訂藍圖的層訂閱。

藍圖名稱	藍圖描述
基岩 Genai 聊天機器人	該藍圖與 Amazon 基岩和人性公司的克勞德建立了生成式 AI 聊天機器人。使用此藍圖，您可以構建和部署自己的安全，受登錄保護的 LLM 遊樂場，該遊樂場可根據您的數據進行自定義。有關更多信息，請參閱基岩 GenAI Chatbot 文檔。
AWS 專案開發套件 (AWS PDK) 藍圖	這些 PDK 藍圖可以組合在一起，以建立包含 React 網站、史密斯鋪 API 和支援 CDK 基礎設施的應用程式，以便將其部署到 AWS。AWS PDK 提供常見模式的建置區塊，以及管理和建置專案的開發工具。如需詳細資訊，請參閱 AWS PDK GitHub 來源儲存庫和教學課程：建立包含可組合式 PDK 藍圖的完整堆疊應用程式.

尋找專案藍圖資訊

中 CodeCatalyst有數個專案藍圖可供使用。對於每個藍圖，都有隨附的摘要和 README 檔案。摘要描述藍圖所安裝的資源，而 README 檔案會詳細說明藍圖，並提供如何使用藍圖的指示。

使用中的自訂藍圖 CodeCatalyst

您可以使用自訂藍圖標準化 CodeCatalyst 空間專案的開發和最佳實務。自訂藍圖可用來定義 CodeCatalyst 專案的各個層面，例如工作流程定義和應用程式程式碼。使用自訂藍圖建立新專案或套用至現有專案後，對藍圖的任何變更都可以作為提取申請更新提供給這些專案。身為藍圖作者，您可以檢視有關哪些專案在整個空間中使用藍圖的詳細資料，以便瞭解如何跨專案套用標準。藍圖的生命週期管理可讓您集中管理每個專案的軟體開發生命週期，讓您能夠確保空間中的專案繼續遵循最新變更或修正的最佳實務。如需詳細資訊，請參閱 以藍圖作者身分使用生命週期管理。

自訂藍圖提供透過重新同步對先前專案更新藍圖版本的功能。Resynthesis 是使用更新版本重新運行藍圖合成的過程，或者將修復程序和更改納入現有項目的能力。如需詳細資訊，請參閱 自訂藍圖概念。

若要檢視藍圖 SDK 和範例藍圖，請參閱開放原始碼存放 GitHub 庫。

主題

自訂藍圖概念

以下是在中 CodeCatalyst使用自訂藍圖時應瞭解的一些概念和術語。

主題

藍圖專案

藍圖專案可讓您開發藍圖並將其發佈到您的空間。在專案建立程序期間會建立來源儲存庫，而儲存庫的名稱是您在輸入 Project 資源詳細資訊時所選擇的名稱。在藍圖建立程序期間，如果您選擇產生工作流

程版本，則會使用藍圖產生器藍圖在您的藍圖中建立發佈工作流程。工作流程會自動發佈您的最新版本。

空間, 藍圖

當您導覽至空間的藍圖區段時，您可以從 S pace 藍圖表格檢視和管理所有藍圖。將藍圖發佈到您的空間後，它們將作為空間藍圖提供，以便從空間的藍圖目錄中新增和移除。您也可以在空間的 [藍圖] 區段中管理發佈權限和刪除藍圖。如需詳細資訊，請參閱 檢視自訂藍圖的詳細資料、版本和專案。

空間藍圖目錄

您可以從空間的藍圖目錄檢視所有新增的自訂藍圖。在這裡，空間成員可以選擇您的自訂藍圖來建立新專案。此目錄與目錄不同， CodeCatalyst 目錄已為所有空間成員提供可用藍圖。如需詳細資訊，請參閱 專案藍圖參考。

合成

合成是生成一個 CodeCatalyst 項目包，表示源代碼，配置和資源在一個項目的過程。然後 CodeCatalyst 部署 API 作業會使用套件來部署到專案中。此程序可在本機執行，同時開發您的自訂藍圖，以模擬專案建立，而不必在中 CodeCatalyst建立專案。以下命令可用於執行合成：

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

藍圖通過調用合併該選項的主`blueprint.ts`類來啟動`defaults.json`。`synth/synth.`*`[options-name]`*`/proposed-bundle/`資料夾下會產生一個新的專案組合包。輸出包括自訂藍圖產生的專案服務包 (提供您設定的選項)，包括您可能已設定的部分選項。

樹脂同步

Resynthesis 是使用不同藍圖選項或現有專案的藍圖版本重新產生藍圖的程序。身為藍圖作者，您可以在自訂藍圖程式碼中定義自訂合併策略。您也可以在中定義擁有權界限，`.ownership-file`以指定允許更新藍圖程式碼基底的哪些部分。雖然自訂藍圖可以向提議更新`.ownership-file`，但使用自訂藍圖的專案開發人員可以決定其專案的擁有權界限。您可以在本機執行重新同步，並在發佈自訂藍圖之前進行測試和更新。使用下列指令來執行重新同步：

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

藍圖通過調用合併該選項的主blueprint.ts類來啟動defaults.json。synth/resynth.*[options-name]*/資料夾下會產生一個新的專案組合包。輸出包括自訂藍圖產生的專案服務包 (提供您設定的選項)，包括您可能已設定的部分選項。

以下內容是在合成和重新同步過程之後創建的：

- 建議捆綁- 當合成與目標藍圖版本的新選項運行時的輸出。
- 現有的捆綁- 你現有的項目的模擬。如果此文件夾中沒有任何內容，則會以與他相同的輸出生成proposed-bundle。
- 祖先捆綁- 模擬使用以前版本，先前的選項或組合運行時藍圖將生成的內容。如果此資料夾中沒有任何內容，則會以與proposed-bundle.
- 已解析-束一律重新產生，且預設為、與. proposed-bundle existing-bundle ancestor-bundle 此套件提供了重新同步將在本機輸出的模擬。

若要深入瞭解藍圖輸出服務包，請參閱使用重新同步產生檔案。

部分選項

您可以添加選項變體src/wizard-configuration/，而不必枚舉整個Options界面，並且選項將合併到文件的頂部。defaults.json這使您可以跨特定選項定制測試用例。

範例：

Options接口：

```
{
  language: "Python" | "Java" | "Typescript",
  repositoryName: string
  ...
}
```

defaults.json 檔案：

```
{
  language: "Python",
  repositoryName: "Myrepo"
  ...
}
```

其他配置測試：

- ```
  #wizard-config-typescript-test.json
  {
    language: "Typescript",
  }
  ```
- ```
  #wizard-config-java-test.json
  {
    language: "Java",
  }
  ```

普羅真

Projen 是一種開放原始碼工具，可自訂藍圖用來保持自己的最新狀態和一致性。藍圖是 Projen 套件，因為這個架構提供您建置、捆綁和發佈專案的能力，而且您可以使用介面來管理專案的組態和設定。

您可以使用 Projen 大規模更新藍圖，即使在建立藍圖之後也是如此。Projen 工具是產生專案套件的藍圖合成背後的基礎技術。Projen 擁有專案的組態，不應該影響您身為藍圖作者。您可以執行`yarn projen`以在加入相依性之後重新產生專案的組態，也可以變更`projenrc.ts`檔案中的選項。Projen 也是用於合成項目的自定義藍圖的基礎生成工具。如需詳細資訊，請參閱 Projen GitHub 頁面。要了解有關使用 Projen 的更多信息，請參閱 Projen 文檔以及如何使用 Projen 簡化項目設置。

開始使用自訂藍圖

在建立藍圖的過程中，您可以設定藍圖並產生專案資源的預覽。每個自訂藍圖均由一個 CodeCatalyst 專案管理，該專案預設包含用於發佈至空間藍圖目錄的工作流程。

主題

必要條件

建立自訂藍圖之前，請考慮下列需求：

- 您的 CodeCatalyst 空間必須是企業層。如需詳細資訊，請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。
- 您必須具有 S pace 系統管理員或 Power 使用者角色，才能建立自訂藍圖。如需詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

步驟 1：在中建立自訂藍圖 CodeCatalyst

當您從空間的設定建立自訂藍圖時，會為您建立存放庫。存放庫包含將藍圖發佈到空間藍圖目錄之前，必須具備的所有必要資源。

建立自訂藍圖

- a. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
  b. 在主 CodeCatalyst 控台中，導覽至您要建立自訂藍圖的空間。
  c. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
  d. 選擇 [建立藍圖]。
  e. 在為您的藍圖命名下，輸入要指派給專案的名稱及其關聯的資源名稱。名稱在您的空間中必須是唯一的。
  f. 在藍圖詳細資料底下，執行下列操作：
    i. 在藍圖顯示名稱文字輸入欄位中，輸入將出現在空間藍圖目錄中的名稱。
    ii. 在描述文字輸入欄位中，輸入自訂藍圖的描述。
    iii. 在「作者名稱」文字輸入欄位中，輸入自訂藍圖的作者名稱。
    iv. (選擇性) 選擇「進階」設定。
      A. 選擇 [+ 新增] 以新增新增至`package.json`檔案的標籤。
      B. 選擇 [授權] 下拉式功能表，然後為您的自訂藍圖選擇授權。
      C. 在藍圖封裝名稱文字輸入欄位中，輸入用於識別藍圖套件的名稱。
      D. 依預設，會使用名為「藍圖產生器」的專案中的發佈藍圖產生發行工作流程。當您推送變更時，工作流程會將最新的藍圖版本發佈到您的空間，因為發行工作流程已啟用發佈權限。欲關閉工作流程產生，請取消核取「發行工作流程」核取方塊
  g. (選擇性) 藍圖專案隨附預先定義的程式碼，以支援將藍圖發佈至空間目錄。若要根據您選取的專案參數檢視含有更新的定義檔，請從 [產生藍圖預覽] 中選擇 [檢視程式碼] 或 [檢視工作流程]。
  h. 選擇 [建立藍圖]。

如果您沒有關閉自訂藍圖的工作流程產生，則工作流程會在建立藍圖時自動開始執行。工作流程執行完成後，依預設，您的自訂藍圖即可新增至空間的藍圖目錄。如果您不希望將最新的藍圖版本自動發佈到您的空間，則可以關閉發佈權限。如需詳細資訊，請參閱 在專案中套用和取消關聯藍圖。

由於呼叫的發佈工作流程`blueprint-release`是使用藍圖建立的，因此可以在專案中找到該藍圖做為套用的藍圖。如需詳細資訊，請參閱 在專案中套用和取消關聯藍圖 及 使用工作流程。

步驟 2：使用元件開發自訂藍圖

當您建立自訂藍圖時會產生藍圖精靈，並且可在開發自訂藍圖時使用元件對其進行修改。您可以更新`src/blueprints.js`和`src/defaults.json`檔案以修改精靈。

> **Important**
>
> 如果您想要使用來自外部來源的藍圖套件，請考慮這些套件可能帶來的風險。您必須負責新增至空間的自訂藍圖及其產生的程式碼。

在設定藍圖程式碼之前，使用支援的整合式開發環境 (IDE) 在 CodeCatalyst 專案中建立開發環境。開發環境是使用必要的工具和套件所需的必要條件。

若要建立開發環境

1. 在導覽窗格中，執行下列其中一項作業：
   a. 選擇 [概觀]，然後瀏覽至 [我的開發環境] 區段。
   b. 選擇 [程式碼]，然後選擇 [開發環境]。
   c. 選擇 [程式碼]，選擇 [來源存放庫]，然後選擇您在建立藍圖時建立的存放庫。
2. 選擇 [建立開發環境]。
3. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱開發環境支援的整合式開發環境。
4. 選擇「在現有分支中工作」，然後從「現有分支」下拉式功能表中選擇您建立的功能分支。
5. (選擇性) 在別名-選擇性文字輸入欄位中，輸入別名以識別開發環境。
6. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [啟動]，且狀態欄會在建立開發環境時顯示 [執行中]。

如需詳細資訊，請參閱 開發環境 CodeCatalyst。

若要開發您的自訂藍圖

1. 在工作終端中，使用以下yarn命令來安裝依賴關係：

   ```
   yarn
   ```

   所需的工具和套件可透過 CodeCatalyst 開發環境取得，包括 Yarn。如果您正在使用沒有開發環境的自訂藍圖，請先將 Yarn 安裝到您的系統。如需詳細資訊，請參閱 Yarn 的安裝文件。
2. 開發您的自訂藍圖，以便根據您的喜好進行設定。您可以透過新增元件來修改藍圖的精靈。如需詳細資訊，請參閱 開發自訂藍圖、使用精靈 及 發佈自訂藍圖。

步驟 3：預覽自訂藍圖

設定和開發自訂藍圖後，您可以預覽藍圖的預覽版本，並將其發佈到您的空間。預覽版本可讓您在藍圖用於建立新專案或套用至現有專案之前，先檢查藍圖是否是您想要的。

預覽自訂藍圖

1. 在工作終端中，使用以下yarn命令：

   ```
   yarn blueprint:preview
   ```
2. 導覽至提供的`See this blueprint at:`連結以預覽您的自訂藍圖。
3. 根據您的組態，檢查 UI (包括文字) 是否如預期般顯示。如果您想要變更自訂藍圖，可以編輯`blueprint.ts`檔案、重新同步藍圖，然後再次發佈預覽版本。如需詳細資訊，請參閱 樹脂同步。

(選用) 步驟 4：發佈自訂藍圖預覽版本

如果要將自訂藍圖新增至空間的藍圖目錄，您可以將自訂藍圖的預覽版本發佈到您的空間。這可讓您在將非預覽版本新增至目錄之前，以使用者身分檢視藍圖。預覽版可讓您在不佔用實際版本的情況下發佈。例如，如果您使用某個`0.0.1`版本，則可以發佈並新增預覽版本，以便可以將第二個版本的新更新發行並新增為`0.0.2`。

發佈自訂藍圖的預覽版本

導覽至提供的`Enable version` *`[version number]`* `at:`連結以啟用您的自訂藍圖。在中執行`yarn`指令時會提供此連結步驟 3：預覽自訂藍圖。

建立、開發、預覽和發佈自訂藍圖後，您可以將最終藍圖版本發佈並新增至空間的藍圖目錄。如需更多詳細資訊，請參閱 在空間中新增和移除自訂藍圖。

教學課程：建立和更新 React 應用程式

身為藍圖作者，您可以開發自訂藍圖，並將其新增至空間的藍圖目錄。然後，空間成員可以使用這些藍圖來建立新專案或將其套用至現有專案。您可以繼續對藍圖進行變更，然後透過提取要求提供為更新。

本教學課程從藍圖作者的角度和藍圖使用者的觀點提供逐步解說。本教學課程說明如何建立 React 單頁 Web 應用程式藍圖。然後會使用藍圖來建立新專案。使用變更更新藍圖時，從藍圖建立的專案會透過提取要求合併這些變更。

主題

必要條件

若要建立和更新自訂藍圖，您必須已完成中的工作，正在設定 CodeCatalyst如下所示：

- 有一個用於登錄的 AWS 生成器 ID CodeCatalyst。
- 屬於空間，並在該空間中為您指派 Space 管理員或超級使用者角色。如需詳細資訊，請參閱建立支援 AWS 產生器 ID 使用者的空間、管理空間使用者及空間管理員角色。

步驟 1：建立自訂藍圖

當您建立自訂藍圖時，會建立包含藍圖原始程式碼以及開發工具和資源的 CodeCatalyst 專案。您的專案是您開發、測試和發佈藍圖的地方。

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在主 CodeCatalyst 控台中，導覽至要建立藍圖的空間。
3. 選擇設定以導覽至空間設定。
4. 在空間設定索引標籤中，選擇藍圖，然後選擇建立藍圖。
5. 使用下列值更新藍圖建立精靈中的欄位：

   - 在藍圖名稱中，輸入`react-app-blueprint`。
   - 在藍圖顯示名稱中，輸入`react-app-blueprint`。
6. 或者，選擇 [檢視程式碼] 以預覽藍圖的藍圖原始程式碼。同樣地，選擇 [檢視工作流程] 以預覽將在建置和發佈藍圖的專案中建立的工作流程。
7. 選擇 [建立藍圖]。
8. 建立藍圖後，您將進入藍圖的專案。此專案包含藍圖原始程式碼，以及開發、測試和發佈藍圖所需的工具和資源。已產生發行工作流程，並自動將您的藍圖發佈至空間。
9. 現在您的藍圖和藍圖專案已建立，下一個步驟是透過更新原始程式碼來進行設定。您可以使用開發環境直接在瀏覽器中打開和編輯源代碼存儲庫。

   在瀏覽窗格中，選擇 [程式碼]，然後選擇 [開發環境]。
10. 選擇創建開發環境，然後選擇 AWS Cloud9 （在瀏覽器中）。
11. 保留預設設定，然後選擇 [建立]。
12. 在 AWS Cloud9 終端機中，透過執行下列命令導覽至您的藍圖專案目錄：

    ```
    cd react-app-blueprint
    ```
13. 建立藍圖時，會自動建立`static-assets`資料夾並填入資料夾。在本教程中，您將刪除默認文件夾並為反應應用程序藍圖生成一個新文件夾。

    通過運行以下命令刪除靜態資產文件夾：

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

    AWS Cloud9 是建立在一個基於 Linux 的平台上。如果您使用的是 Windows 作業系統，則可以改用下列命令：

    ```
    rmdir /s /q static-assets
    ```
14. 現在已刪除預設資料夾，請執行下列命令來建立 React-app 藍圖的`static-assets`資料夾：

```
npx create-react-app static-assets
```

如果出現提示，請輸y入以繼續。

在包含必要套件的static-assets資料夾中建立了一個新的反應應用程式。這些更改需要推送到您的遠程 CodeCatalyst 源存儲庫。

15. 確定您擁有最新的變更，然後執行下列命令，將變更認可並推送至藍圖的 CodeCatalyst 來源存放庫：

```
git pull
```

```
git add .
```

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

```
git push
```

將變更推送至藍圖的來源存放庫時，會自動啟動發行工作流程。此工作流程會增加藍圖版本、建置藍圖，然後將其發佈到您的空間。在下一步中，您將導航到發行工作流程運行以查看其運行情況。

步驟 2：檢視發行工作流程

1. 在 CodeCatalyst 主控台的功能窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇藍圖釋放工作流程。
3. 您可以看到工作流程具有建立和發佈藍圖的動作。
4. 在「最新執行」下，選擇工作流程執行連結，以從您所做的程式碼變更中檢視執行。
5. 執行完成後，即會發佈新藍圖版本。已發佈的藍圖版本可以在您的空間設定中看到，但在將其新增至空間的藍圖目錄之前，無法在專案中使用。在下一個步驟中，您會將藍圖新增至目錄。

步驟 3：將藍圖新增至目錄

將藍圖新增至空間的藍圖目錄可讓藍圖用於空間中的所有專案。Space 成員可以使用藍圖建立新專案或將其套用至現有專案。

1. 在主 CodeCatalyst 控台中，導覽回空間。
2. 選擇 [設定]，然後選擇 [藍圖]。
3. 選擇 react-app-blueprint，然後選擇 [新增至目錄]。
4. 選擇儲存。

步驟 4：使用藍圖建立專案

現在已將藍圖新增至目錄，即可在專案中使用該藍圖。在此步驟中，您將使用剛才建立的藍圖建立專案。在稍後的步驟中，您將透過更新和發佈新版本的藍圖來更新此專案。

1. 選擇「項目」選項卡，然後選擇「創建項目」。
2. 選擇 [空間藍圖]，然後選擇react-app-blueprint。

> **Note**
>
> 選擇藍圖後，您可以看到藍圖`README.md`檔案的內容。

3. 選擇下一步。
4.

> **Note**
>
> 此專案建立精靈的內容可在藍圖中設定。

以藍圖使用者身分輸入專案名稱。針對本教學，輸入 `react-app-project`。如需詳細資訊，請參閱 開發自訂藍圖。

接下來，您將對藍圖進行更新，並將新版本新增至您將用於更新此專案的目錄。

步驟 5：更新藍圖

使用藍圖建立新專案或套用至現有專案後，您可以繼續以藍圖作者的身分進行更新。在此步驟中，您將對藍圖進行變更，並自動將新版本發佈到空間。然後可以將新版本新增為目錄版本。

1. 導覽至在中建立的react-app-blueprint專案教學課程：建立和更新 React 應用程式。
2. 開啟在中建立的開發環境教學課程：建立和更新 React 應用程式。
   a. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [開發環境]。

b. 從表格中找到開發環境，然後選擇 [在 AWS Cloud9 瀏覽器中開啟]。

3. 執行藍圖發行工作流程時，它會透過更新`package.json`檔案來增加藍圖版本。通過在 AWS Cloud9 終端中運行以下命令來提取該更改：

```
git pull
```

4. 執行下列命令以導覽至`static-assets`資料夾：

```
cd /projects/react-app-blueprint/static-assets
```

5. 通過運行以下命令在`hello-world.txt`文件`static-assets`夾中創建文件：

```
touch hello-world.txt
```

AWS Cloud9 是建立在一個基於 Linux 的平台上。如果您使用的是 Windows 作業系統，則可以改用下列命令：

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

6. 在左側導覽列中，按兩下`hello-world.txt`檔案以在編輯器中開啟檔案，然後新增下列內容：

```
Hello, world!
```

儲存檔案。

7. 確定您擁有最新的變更，然後執行下列命令，將變更認可並推送至藍圖的 CodeCatalyst 來源存放庫：

```
git pull
```

```
git add .
```

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

```
git push
```

推送變更會啟動發行工作流程，該工作流程會自動將新版本的藍圖發佈到空間。

步驟 6：將藍圖的已發佈目錄版本更新為新版本

使用藍圖建立新專案或套用至現有專案後，您仍可以藍圖作者身分更新藍圖。在此步驟中，您將對藍圖進行變更並變更藍圖的目錄版本。

1. 在主 CodeCatalyst 控台中，導覽回空間。
2. 選擇 [設定]，然後選擇 [藍圖]。
3. 選擇 react-app-blueprint，然後選擇 [管理目錄版本]。
4. 選擇新版本，然後選擇 [儲存]。

步驟 7：使用新藍圖版本更新專案

空間的藍圖目錄中現在提供新版本。身為藍圖使用者，您可以更新在中建立之專案的版本步驟 4：使用藍圖建立專案。如此可確保您擁有符合最佳實務所需的最新變更和修正。

1. 在 CodeCatalyst 主控台中，瀏覽至中建立的react-app-project專案步驟 4：使用藍圖建立專案。
2. 在導覽窗格中，選擇 Blueprints (藍圖)。
3. 在資訊方塊中選擇 [更新藍圖]。
4. 在右側的「程式碼變更」面板中，您可以看到`hello-world.txt`和`package.json`更新。
5. 選擇 [套用更新]。

選擇 [套用更新] 會在專案中建立一個提取要求，其中包含來自更新藍圖版本的變更。若要對專案進行更新，您必須合併提取要求。如需詳細資訊，請參閱 檢閱提取要求 及 合併提取請求。

1. 在藍圖表格中，尋找藍圖。在「狀態」欄中，選擇「擱置中」提取請求，然後選擇連至未結提取請求的連結。
2. 複查提取請求，然後選擇「合併」。
3. 選擇快進合併以保留預設值，然後選擇「合併」。

步驟 8：檢視專案中的變更

藍圖的變更現在可在之後在您的專案中使用步驟 7：使用新藍圖版本更新專案。身為藍圖使用者，您可以檢視來源存放庫中的變更。

1. 在瀏覽窗格中，選擇 [來源儲存庫]，然後選擇建立專案時所建立之來源儲存庫的名稱。
2. 在「檔案」下，您可以檢視在中建立的`hello-world.txt`檔案步驟 5：更新藍圖。

3. 選擇hello-world.txt以檢視檔案的內容。

生命週期管理讓藍圖作者能夠集中管理包含特定藍圖的每個專案的軟體開發生命週期。如本教學課程所見，您可以將更新推送至藍圖，然後再由使用藍圖建立新專案的專案合併，或將其套用至現有專案的專案。如需更多詳細資訊，請參閱 以藍圖作者身分使用生命週期管理。

以藍圖作者身分使用生命週期管理

生命週期管理可讓您從單一通用的最佳實務來源保持大量專案同步。這樣可以在整個軟體開發生命週期中擴展修正程式的傳播，以及任意數量的專案維護。生命週期管理可簡化內部宣傳活動、安全性修正、稽核、執行階段升級、最佳實務變更以及其他維護作法，因為這些標準是在單一位置定義，並在發佈新標準時自動保持集中更新。

發佈藍圖的新版本時，會提示包含該藍圖的所有專案更新為最新版本。身為藍圖作者，您還可以查看每個專案出於合規目的而包含的特定藍圖版本。當現有來源儲存庫發生衝突時，生命週期管理會建立提取要求。對於所有其他資源，例如開發環境，所有生命週期管理更新都會嚴格建立新資源。用戶可以自由地合併或不合併這些提取請求。合併擱置的提取要求時，會更新專案中使用的藍圖版本 (包括選項)。若要瞭解如何以藍圖使用者身分使用生命週期管理，請參閱對現有專案使用生命週期管理和在專案中的多個藍圖上使用生命週期管理。

主題

- 測試週期管理
- 使用合併策略
- 使用前後關聯物件

測試週期管理

您可以在本機測試藍圖的生命週期管理並合併衝突解決方案。系統會產生一系列代表生命週期更新各階段的synth/目錄下的套裝軟體。若要測試生命週期管理，您可以在藍圖上執行下列 yarn 命令：yarn blueprint: resynth若要進一步瞭解重新同步和套裝軟體，請參閱樹脂同步和。使用重新同步產生檔案

使用合併策略

主題

- 使用重新同步產生檔案
- 使用合併策略

- 指定生命週期管理更新的檔案
- 撰寫合併策略

使用重新同步產生檔案

Resynthesis 可以將藍圖產生的原始程式碼與先前由相同藍圖產生的原始程式碼合併，從而允許對藍圖的變更傳播到現有專案。合併是從跨藍圖輸出服務包的`resynth()`函數執行。Resynthesis 首先會產生三個束，代表藍圖和專案狀態的不同層面。它可以使用`yarn blueprint:resynth`命令在本地手動運行，如果它們不存在，它們將創建捆綁包。手動使用包將允許您在本地模擬和測試重新同步行為。依預設，藍圖只會在下的儲存庫中執行重新同步，`src/*`因為只有套裝軟體的該部分位於原始檔控制之下。

- `existing-bundle`-此捆綁是現有項目狀態的表示。這是由合成計算人為構建的，以提供有關它正在部署到的項目中的內容（如果有的話）的藍圖上下文。如果在本地運行 resynthesis 時在此位置已經存在某些東西，它將被重置並尊重為模擬。否則，它將被設置為的內容`ancestor-bundle`。
- `ancestor-bundle`-如果與某些先前的選項和/或版本合成藍圖輸出，則表示藍圖輸出的捆綁包。如果這是第一次將此藍圖新增至專案，則祖系不存在，因此會將其設定`existing-bundle`為與. 在本地，如果這個包已經存在於這個位置，它將被視為一個模擬。
- `proposed-bundle`-如果使用一些新選項和/或版本合成藍圖，這是嘲笑藍圖的捆綁包。這是將由`synth()`函數產生的同一個包。在本機上，此套件一律會遭到覆寫。

每個套裝軟體都會在重新同步階段建立，該階段可從下的藍圖類別存取。`this.context.resynthesisPhase`

- `resolved-bundle`-這是最後一個包，這是什麼被打包和部署到 CodeCatalyst 項目的表示。您可以檢視傳送至部署機制的檔案和差異。這是解析其他三個包之間合併的`resynth()`函數的輸出。

採用和之間的差異`ancestor-bundle`proposed-bundle並將其應用於以產生三`existing-bundle`向合併。`resolved-bundle`所有合併策略都會將檔案解析為`resolved-bundle`. Resynthesis 會在期間使用藍圖的合併策略解決這些套裝軟體的覆蓋範圍，`resynth()`並從結果中產生已解析的套裝軟體。

使用合併策略

您可以使用藍圖庫提供的合併策略。這些策略提供了解決使用重新同步產生檔案本節中所述檔案的檔案輸出和衝突的方法。

- alwaysUpdate-一律解析為建議檔案的策略。
- neverUpdate-一律解析為現有檔案的策略。
- onlyAdd-當現有檔案不存在時，可解析為建議檔案的策略。否則，會解析為現有檔案。
- threeWayMerge-一種在現有，建議和共同祖先文件之間執行三向合併的策略。如果文件無法乾淨地合併，則解析的文件可能包含衝突標記。提供的檔案內容必須以 UTF-8 編碼，以便策略產生有意義的輸出。該策略嘗試檢測輸入文件是否為二進製文件。如果策略偵測到二進位檔案中的合併衝突，它總是會傳回建議的檔案。
- preferProposed-一種在現有，建議和共同祖先文件之間執行三向合併的策略。此策略透過選取每個衝突的建議檔案側來解決衝突。
- preferExisting-一種在現有，建議和共同祖先文件之間執行三向合併的策略。此策略透過選取每個衝突的現有檔案側來解決衝突。

若要檢視合併策略的原始碼，請參閱開放原始碼 GitHub 儲存庫。

指定生命週期管理更新的檔案

重新同步期間，藍圖會控制變更合併到現有來源儲存庫的方式。但是，您可能不希望將更新推送到藍圖中的每個檔案。例如，CSS 樣式表之類的示例代碼旨在特定於項目。如果您未指定其他策略，則預設選項為三向合併策略。藍圖可以通過在存儲庫構造本身上指定合併策略來指定它們擁有的文件以及它們不擁有的文件。藍圖可以更新其合併策略，並且可以在重新同步期間使用最新策略。

```
const sourceRepo = new SourceRepository(this, {
      title: 'my-repo',
    });
    sourceRepo.setResynthStrategies([
      {
        identifier: 'dont-override-sample-code',
        description: 'This strategy is applied accross all sample code. The blueprint
 will create sample code, but skip attempting to update it.',
        strategy: MergeStrategies.neverUpdate,
        globs: [
          '**/src/**',
          '**/css/**',
        ],
      },
    ]);
```

可以指定多個合併策略，最後一個策略優先。未覆蓋的文件默認為 three-way-merge 類似於 Git。通過MergeStrategies構造提供了幾種合併策略，但是您可以編寫自己的合併策略。提供的策略堅持 git 合併策略驅動程序。

撰寫合併策略

除了使用提供的其中一種構建合併策略之外，您還可以編寫自己的策略。策略必須堅持標準的策略界面。您必須撰寫策略函數，該函數會從、和取得檔案的版本 existing-bundle proposed-bundleancestor-bundle，並將它們合併到單一已解析檔案中。例如：

```
type StrategyFunction = (
   /**
   * file from the ancestor bundle (if it exists)
   */
    commonAncestorFile: ContextFile | undefined,
   /**
   * file from the existing bundle (if it exists)
   */
    existingFile: ContextFile | undefined,
   /**
   * file from the proposed bundle (if it exists)
   */
    proposedFile: ContextFile | undefined,
    options?: {})
    /**
    * Return: file you'd like in the resolved bundle
    * passing undefined will delete the file from the resolved bundle
    */
=> ContextFile | undefined;
```

如果文件不存在（未定義），則該文件路徑不存在於該特定位置包中。

範例：

```
strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied across all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: (ancestor, existing, proposed) => {
                const resolvedfile = ...
                ...
```

```
            // do something
            ...
            return resolvedfile
          },
          globs: [
            '**/src/**',
            '**/css/**',
          ],
        },
      ],
```

使用前後關聯物件

身為藍圖作者，您可以在合成期間從藍圖的專案存取內容，以取得空間和專案名稱等資訊，或是專案來源存放庫中的現有檔案。您也可以取得詳細資料，例如藍圖正在產生的重新同步階段。例如，您可以訪問上下文以了解是否要重新同步以生成祖系包還是提議的包。然後可以使用現有的代碼上下文來轉換存儲庫中的代碼。例如，您可以撰寫自己的重新同步策略來設定特定的程式碼標準。您可以將策略新增至小型藍圖的`blueprint.ts`檔案中，或者您可以為策略建立單獨的檔案。

下列範例說明如何在專案的前後關聯中尋找檔案、設定工作流程建置器，以及如何為特定檔案設定藍圖散佈的重新同步策略：

```
const contextFiles = this.context.project.src.findAll({
      fileGlobs: ['**/package.json'],
    });

    // const workflows = this.context.project.src.findAll({
    //   fileGlobs: ['**/.codecatalyst/**.yaml'],
    // });

    const security = new WorkflowBuilder(this, {
      Name: 'security-workflow',
    });
    new Workflow(this, repo, security.getDefinition());
    repo.setResynthStrategies([
      {
        identifier: 'force-security',
        globs: ['**/.codecatalyst/security-workflow.yaml'],
        strategy: MergeStrategies.alwaysUpdate,
      },
    ]);
```

```
    for (const contextFile of contextFiles) {
      const packageObject = JSON.parse(contextFile.buffer.toString());
      new SourceFile(internalRepo, contextFile.path, JSON.stringify({
        ...packageObject,
      }, null, 2));
    }
  }
```

開發自訂藍圖

在發佈自訂藍圖之前，您可以開發藍圖以符合特定需求。您可以透過在預覽時建立專案來開發自訂藍圖並測試藍圖。您可以開發自訂藍圖以包含專案元件，例如特定原始程式碼、帳戶連線、工作流程、問題或可在中建立的任何其他元件 CodeCatalyst。

> **Important**
>
> 如果您想要使用來自外部來源的藍圖套件，請考慮這些套件可能帶來的風險。您必須負責新增至空間的自訂藍圖及其產生的程式碼。

開發或更新自訂藍圖

1. 恢復您的開發環境。如需詳細資訊，請參閱 恢復開發環境。

   如果您沒有開發環境，則必須先創建一個開發環境。如需詳細資訊，請參閱 建立開發環境。
2. 在您的開發環境中打開一個工作終端。
3. 如果您在建立藍圖時選擇加入發行工作流程，則會自動發佈最新的藍圖版本。提取更改以確保`package.json`文件具有遞增的版本。使用下列命令：

   ```
   git pull
   ```
4. 在`src/blueprint.ts`檔案中，編輯自訂藍圖的選項。 CodeCatalyst 精靈會動態解譯`Options`介面，以產生選取使用者介面 (UI)。您可以透過新增元件和支援的標籤來開發自訂藍圖。如需詳細資訊，請參閱 使用精靈、環境元件、區域元件、儲存庫和原始程式碼元件、工作流元件、開發環境元件。

   您也可以在開發自訂藍圖時檢視藍圖 SDK 和範例藍圖，以取得其他支援。如需詳細資訊，請參閱開放原始碼 GitHub 存放庫。

使用精靈

上的藍圖選取 CodeCatalyst 精靈會由`blueprint.ts`檔案中的`Options`介面自動產生。前端嚮導支持`Options`使用 JSDOC 樣式註釋和標籤藍圖的修改和功能。您可以使用 JSDOC 風格的註釋和標籤來執行任務。例如，您可以選取選項上方顯示的文字、啟用輸入驗證等功能，或使選項可摺疊。該嚮導通過解釋從`Options`接口 TypeScript 類型生成的抽象語法樹（AST）來工作。精靈會盡可能自動將自己設定為描述的類型。並非所有類型都受到支援。其他支援的類型包括區域選取器和環境選擇器。

以下是使用帶有藍圖的 JSDOC 註解和標籤的精靈範例：`Options`

```
export interface Options {
  /**
   * What do you want to call your new blueprint?
   * @validationRegex /^[a-zA-Z0-9_]+$/
   * @validationMessage Must contain only upper and lowercase letters, numbers and
 underscores
   */
  blueprintName: string;

  /**
   * Add a description for your new blueprint.
   */
   description?: string;

   /**
    * Tags for your Blueprint:
    * @collapsed true
    */
  tags?: string[];
}
```

`camelCase`依預設，`Options`介面的每個選項的顯示名稱都會顯示在中。JSDOC 樣式註解中的純文字會顯示為精靈中選項上方的文字。

主題

- 支援的標籤
- 支援的 TypeScript 類型
- 在合成過程中與用戶溝通

支援的標籤

前端精靈0ptions中的自訂藍圖支援下列 JSDOC 標籤。

@inlinePolicy. /路徑/到/政策/文件json

- 需要-選項是一種類型Role。
- 用法-可讓您溝通角色所需的內嵌原則。路policy.json徑預計在源代碼之下。當您需要角色的自訂原則時，請使用此標記。
- 依賴關係-blueprint-cli 0.1.12 及以上
- 範例-@inlinePolicy ./deployment-policy.json

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

@trustPolicy. /路徑/到/政策/文件json

- 需要-選項是一種類型Role。
- 用法-可讓您傳達角色所需的信任原則。路policy.json徑預計在源代碼之下。當您需要角色的自訂原則時，請使用此標記。
- 依賴關係-blueprint-cli 0.1.12 及以上
- 範例-@trustPolicy ./trust-policy.json

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

@validationRegex 正則表達式

- 要求-選項是一個字符串。
- 用法-通過使用給定的正則表達式和顯示對選項執行輸入驗證@validationMessage。
- 範例-@validationRegex /^[a-zA-Z0-9_]+$/
- 建議-搭配使用@validationMessage。驗證訊息預設為空白。

@validationMessage 字串

- 需要-@validationRegex 或其他錯誤以查看使用情況。
- 用法-在@validation*失敗時顯示驗證訊息。
- 示例-@validationMessage Must contain only upper and lowercase letters, numbers, and underscores。
- 建議-搭配使用@validationMessage。驗證訊息預設為空白。

@collapsed 布林值 (選用)

- 需求-不適用
- 用法-允許子選項可折疊的布爾值。如果存在摺疊註解，則其預設值為 true。將該值設置為@collapsed false創建一個最初打開的可折疊部分。
- 範例-@collapsed true

@displayName 字串

- 需求-不適用
- 用法-變更選項顯示名稱。允許使用 camelCase 以外的格式作為顯示名稱。
- 範例-@displayName Blueprint Name

@displayName 字串

- 需求-不適用
- 用法-變更選項顯示名稱。允許使用 camelCase 以外的格式作為顯示名稱。
- 範例-@displayName Blueprint Name

@defaultEntropy 號碼

- 要求-選項是一個字符串。
- 用法-將指定長度的隨機字母數字字串附加至選項。
- 範例-`@defaultEntropy 5`

@placeholder 字符串（可選）

- 需求-不適用
- 用法-變更預設文字欄位預留位置。
- 範例-`@placeholder type project name here`

@textArea 號碼 (選填)

- 需求-不適用
- 用法-將字符串輸入轉換為文本區域組件，用於較大的文本部分。新增數字會定義列數。預設值為五列。
- 範例-`@textArea 10`

@hidden 布林值 (選用)

- 需求-不適用
- 用法-除非驗證檢查失敗，否則對使用者隱藏檔案。默認值為真。
- 範例-`@hidden`

@button 布林值 (選用)

- 需求-不適用
- 用法-註釋必須位於布林屬性上。添加一個按鈕，該按鈕在選擇時合成為 true。不是一個切換。
- 範例-`buttonExample: boolean;`

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

@showName 布林值 (選用)

- 需求-不適用
- 用法-只能在帳戶連線類型上使用。顯示隱藏的名稱輸入。預設為 default_environment。
- 範例-@showName true

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

@ showEnvironmentType 布爾（ 可選 ）

- 需求-不適用
- 用法-只能在帳戶連線類型上使用。顯示隱藏的環境類型下拉菜單。所有連線預設為production。選項為「非生產」或「生產」。
- 範例-@showEnvironmentType true

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

@forceDefault 布林值 (選用)

- 需求-不適用
- 用法-使用藍圖作者提供的預設值，而非使用者先前使用的值。
- 範例-forceDeafultExample: any;

```
/**
  * @forceDefault
  */
forceDeafultExample: any;
```

@requires 藍圖名稱

- 需要-註解介面 Options
- 使用情況-警告使用者將指定套用blueprintName至專案作為目前藍圖的需求。
- 範例-@requires '@amazon-codecatalyst/blueprints.blueprint-builder'

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

支援的 TypeScript 類型

前端精靈Options中的自訂藍圖支援下列 TypeScript 類型。

Number

- 需要-選項是一種類型number。
- 用法-生成一個數字輸入字段。
- 範例-age: number

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

字串

- 需要-選項是一種類型string。
- 用法-生成一個字符串輸入字段。
- 範例-name: string

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

```
}
```

字串清單

- 需要-選項是一種類型boolean。
- 用法-產生核取方塊。
- 範例-isProduction: boolean

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

收音机

- 要求-選項是由三個或更少的字符串組成的聯合。
- 用法-產生選取的無線電。

> **Note**
>
> 當有四個或更多的項目，這種類型呈現為一個下拉列表。

- 範例-color: 'red' | 'blue' | 'green'

```
{
  color: 'red' | 'blue' | 'green'
  ...
}
```

下拉式清單

- 要求-選項是由四個或更多字符串組成的聯合。
- 用法-生成一個下拉菜單。
- 範例-runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'

```
{
```

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

可擴展部分

- 要求-選項是一個對象。
- 用法-產生可展開的區段。物件中的選項將嵌套在精靈中的可展開區段中。
- 範例-

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

元組

- 需要-選項為類型Tuple。
- 用法-生成一個鍵值支付輸入。
- 範例-tuple: Tuple[string, string]>

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

元組列表

- 要求-選項是一個類型的數組Tuple。
- 用法-生成一個元組列表輸入。
- 範例-tupleList: Tuple[string, string]>[]

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

```
  ...
}
```

選擇器

- 需要-選項為類型Selector。
- 使用-生成應用於項目的源存儲庫或藍圖的下拉列表。
- 範例-sourceRepo: Selector<SourceRepository>

```
{
    sourceRepo: Selector<SourceRepository>;
    sourceRepoOrAdd: Selector<SourceRepository | string>;
    blueprintInstantiation: Selector<BlueprintInstantiation>;
  ...
}
```

多重選取

- 需要-選項為類型Selector。
- 用法-產生多重選取輸入。
- 範例-multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>

```
{
  multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>;
  ...
}
```

在合成過程中與用戶溝通

身為藍圖作者，除了驗證訊息之外，您還可以與使用者溝通。例如，空間成員可能會檢視產生不清楚藍圖的選項組合。自訂藍圖支援透過叫用合成將錯誤訊息傳回給使用者的能力。基礎藍圖實作的throwSynthesisError(...)函數需要明確的錯誤訊息。您可以使用下列命令叫用訊息：

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

```
})
```

環境元件

自訂藍圖精靈是從透過精靈公開的Options介面動態產生的。藍圖支援從公開類型產生使用者介面 (UI) 元件。

若要匯入 Amazon CodeCatalyst 藍圖環境元件

在您的blueprint.ts檔案中，新增下列內容：

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

主題

- 建立開發環境
- 模擬界面示例

建立開發環境

下列範例顯示如何將應用程式部署到雲端：

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

介面會產生 UI 元件，要求使用單一帳戶連線 (thisIsMyFirstAccountConnection. myNewEnvironment 帳號連線上的角色 (thisIsARole) 也會以最低必要的['lambda', 's3', 'dynamo']角色權能產生。並非所有使用者都有帳戶連線，因此您應該檢查使用者未連線帳戶或未與角色連線帳戶的情況。角色也可以用@inlinePolicies註釋。如需詳細資訊，請參閱 @inlinePolicy. /路徑/到/政策/文件json。

環境元件需要name和environmentType。下面的代碼是所需的最小默認形狀：

```
{
```

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

然後 UI 組件會提示您輸入各種欄位。當您填寫欄位時，藍圖會取得完全展開的形狀。為了測試和開發目的，在defaults.json文件中包含完整的模擬可能會有所幫助。

模擬界面示例

簡單的模擬界面

```
{
    ...
    "thisIsMyEnvironment": {
        "name": "myProductionEnvironment",
        "environmentType": "PRODUCTION",
        "thisIsMySecondAccountConnection": {
            "id": "12345678910",
            "name": "my-account-connection-name",
            "secondAdminRole": {
                "arn": "arn:aws:iam::12345678910:role/ConnectedQuokkaRole",
                "name": "ConnectedQuokkaRole",
                "capabilities": [
                    "lambda",
                    "s3",
                    "dynamo"
                ]
            }
        }
    }
}
```

複雜的模擬界面

```
export interface Options extends ParentOptions {
  /**
   * The name of an environment
   * @displayName This is a Environment Name
   * @collapsed
   */
```

```
  thisIsMyEnvironment: EnvironmentDefinition{
    /**
     * comments about the account that is being deployed into
     * @displayName This account connection has an overriden name
     * @collapsed
     */
    thisIsMyFirstAccountConnection: AccountConnection{
      /**
       * Blah blah some information about the role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
       * @displayName This role has an overriden name
       */
      adminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
       * Blah blah some information about the second role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
       */
      lambdaRole: Role['lambda', 's3'];
    };
    /**
     * comments about the account that is being deployed into
     */
    thisIsMySecondAccountConnection: AccountConnection{
      /**
         * Blah blah some information about the role that I expect
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondAdminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
         * Blah blah some information about the second role that I expect
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondLambdaRole: Role['lambda', 's3'];
    };
  };
}
```

完整的模擬界面

```
{
  ...
  "thisIsMyEnvironment": {
    "name": "my-production-environment",
```

```
    "environmentType": "PRODUCTION",
    "thisIsMySecondAccountConnection": {
      "id": "12345678910",
      "name": "my-connected-account",
      "secondAdminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "secondLambdaRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
    },
    "thisIsMyFirstAccountConnection": {
      "id": "12345678910",
      "name": "my-connected-account",
      "adminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "lambdaRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
```

```
    }
  },
}
```

秘密組件

密碼可用於儲存 CodeCatalyst 可在工作流程中參照的敏感資料。您可以將密碼新增至自訂藍圖，並在工作流程中參考它。如需詳細資訊，請參閱 與秘密一起工作。

若要匯入 Amazon CodeCatalyst 藍圖區域類型

在您的`blueprint.ts`檔案中，新增下列內容：

```
import { Secret, SecretDefinition } from '@amazon-codecatalyst/blueprint-
component.secrets'
```

主題

- 建立密碼
- 參考工作流程中的密碼

建立密碼

下列範例會建立 UI 元件，提示使用者輸入密碼值和選用說明：

```
export interface Options extends ParentOptions {
    ...
    mySecret: SecretDefinition;
}


export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    new Secret(this, options.secret);
}
```

秘密元件需要`name`. 下面的代碼是所需的最小默認形狀：

```
{
    ...
```

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

}
```

參考工作流程中的密碼

下列範例藍圖會建立機密和參考密碼值的工作流程。如需詳細資訊，請參閱 參考工作流程中的密碼。

```
export interface Options extends ParentOptions {
    ...
/**
*
* @validationRegex /^\w+$/
*/
  username: string;


  password: SecretDefinition;
}


export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    const password = new Secret(this, options_.password);

    const workflowBuilder = new WorkflowBuilder(this, {
      Name: 'my_workflow',
    });


    workflowBuilder.addBuildAction({
      actionName: 'download_files',
      input: {
        Sources: ['WorkflowSource'],
      },
      output: {
        Artifacts: [{ Name: 'download', Files: ['file1'] }],
      },
      steps: [
        `curl -u ${options_.username}:${password.reference} https://example.com`,
      ],
    });
```

```
    new Workflow(
      this,
      repo,
      workflowBuilder.getDefinition(),
    );

}
```

若要進一步瞭解如何在中使用密碼 CodeCatalyst，請參閱與秘密一起工作。

區域元件

您可以將區域類型新增至自訂藍圖的Options界面，以便在藍圖精靈中產生元件，您可以輸入一或多個 AWS Gions。Gion 類型可以從blueprint.ts檔案中的基本藍圖匯入。如需詳細資訊，請參閱 AWS 區域。

若要匯入 Amazon CodeCatalyst 藍圖區域類型

在您的blueprint.ts檔案中，新增下列內容：

```
import { Region } from '@amazon-codecatalyst/blueprints.blueprint'
```

區域類型參數是一組 AWS 區域代碼可供選擇，或者您可以用*來包含所有支援的 AWS 區域。

主題

- 註釋
- 區域元件範例

註釋

JSDoc 標籤可以新增至Options介面中的每個欄位，以自訂欄位在精靈中的顯示和行為方式。對於區域類型，支援下列標籤：

- 註@displayName釋可用於在精靈中變更欄位的標籤。

  範例：@displayName AWS Region
- @placeholder註解可用於變更選取/多重選取元件的預留位置。

  範例：@placeholder Choose AWS Region

區域元件範例

從指定的列表中選擇一個區域

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>;
}
```

從指定清單中選擇一或多個地區

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Regions
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

選擇一個 AWS 通路

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['*']>;
}
```

從指定清單中選擇一或多個地區

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Regions
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

儲存庫和原始程式碼元件

Amazon 使用存儲庫 CodeCatalyst 來存儲代碼。存儲庫需要一個名稱作為輸入。大多數組件都存儲在存儲庫中，例如源代碼文件，工作流程以及其他組件，例如託管開發環境（MDE）。來源儲存庫元件也會匯出用於管理檔案和靜態資產的元件。儲存庫有名稱限制。如需詳細資訊，請參閱 來源儲存庫 CodeCatalyst。

```
const repository = new SourceRepository(this, {
  title: 'my-new-repository-title',
});
```

若要匯入 Amazon CodeCatalyst 藍圖儲存庫和原始程式碼元件

在您的`blueprint.ts`檔案中，新增下列內容：

```
import {...} from '@caws-blueprint-component/caws-source-repositories'
```

主題

- 新增檔案
- 新增一般檔案
- 複製檔案
- 定位多個檔案
- 創建一個新的存儲庫和添加文件

新增檔案

您可以使用`SourceFile`構造將文本文件寫入存儲庫。此作業是最常見的使用案例之一，會取得儲存庫、檔案路徑和文字內容。如果儲存庫中不存在檔案路徑，則元件會建立所有必要的資料夾。

```
new SourceFile(repository, `path/to/my/file/in/repo/file.txt`, 'my file contents');
```

> **Note**
>
> 如果您將兩個檔案寫入同一儲存庫中的相同位置，則最新的實作會覆寫前一個檔案。您可以使用此功能來分層產生的程式碼，而且對於延伸自訂藍圖可能產生的程式碼而言，它特別有用。

新增一般檔案

您可以將任意位元寫入儲存庫。您可以從緩衝區讀取並使用File建構。

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

複製檔案

您可以通過複製和粘貼初學者代碼開始使用生成的代碼，然後在該基礎上生成更多代碼。將代碼放在static-assets目錄中，然後使用StaticAsset構造定位該代碼。在這種情況下，路徑始終從目錄的根目static-assets錄開始。

```
const starterCode = new StaticAsset('path/to/file/file.txt')
const starterCodeText = new StaticAsset('path/to/file/file.txt').toString()
const starterCodeRawContent = new StaticAsset('path/to/image/hello.png').content()

const starterCodePath = new StaticAsset('path/to/image/hello.png').path()
// starterCodePath is equal to 'path/to/image/hello.png'
```

的子類別StaticAsset是SubstitutionAsset。子類的功能完全相同，但是您可以在文件上運行鬍子替換。它對於執行 copy-and-replace 樣式生成很有用。

靜態資產替換使用鬍子模板引擎來呈現植入生成源存儲庫的靜態文件。鬍子範本化規則會在轉譯期間套用，這表示所有值預設都是 HTML 編碼。要呈現未轉義的 HTML，請使用三重鬍子語法。{{{name}}}如需詳細資訊，請參閱鬍子範本規則。

> **Note**
>
> 對不可文字解譯的檔案執行替代可能會產生錯誤。

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

定位多個檔案

靜態資產通過 on 的靜態函數StaticAsset和調用的子類來支持 glob 定位findAll(...)，該函數返回預先加載了路徑，內容等的靜態資產列表。您可以使用File建構來鏈結清單，以便在static-assets目錄中複製和貼上內容。

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

創建一個新的存儲庫和添加文件

您可以使用儲存庫元件，在產生的專案中建立新的存放庫。然後，您可以將檔案或工作流程新增至建立的存放庫。

```
import { SourceRepository } from '@amazon-codecatalyst/codecatalyst-source-
repositories';
...
const repository = new SourceRepository(this, { title: 'myRepo' });
```

下列範例顯示如何將檔案和工作流程新增至現有存放庫：

```
import { SourceFile } from '@amazon-codecatalyst/codecatalyst-source-repositories';
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows';
...
new SourceFile(repository, 'README.md', 'This is the content of my readme');
new Workflow(this, repository, {/**...workflowDefinition...**/});
```

結合這兩段程式碼會產生一個名為myRepo來源檔案的單一儲存庫，README.md並在根目錄中命名為CodeCatalyst工作流程。

工作流元件

Amazon CodeCatalyst 專案會使用工作流程根據觸發器執行動作。您可以使用工作流程元件來建置和組合工作流程 YAML 檔案。如需詳細資訊，請參閱 工作流定義參考。

若要匯入 Amazon CodeCatalyst 藍圖工作流程元件

在您的blueprint.ts檔案中，新增下列內容：

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

主題

- 工作流程元件範
- 連線至環境

工作流程元件範

WorkflowBuilder 元件

您可以使用類別來建立工作流程定義。可將定義指定給工作流程元件，以便在存放庫中轉譯。

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

const workflowBuilder = new WorkflowBuilder({} as Blueprint, {
  Name: 'my_workflow',
});

// trigger the workflow on pushes to branch 'main'
workflowBuilder.addBranchTrigger(['main']);

// add a build action
workflowBuilder.addBuildAction({
  // give the action a name
  actionName: 'build_and_do_some_other_stuff',

  // the action pulls from source code
  input: {
    Sources: ['WorkflowSource'],
  },

  // the output attempts to autodiscover test reports, but not in the node modules
  output: {
    AutoDiscoverReports: {
      Enabled: true,
      ReportNamePrefix: AutoDiscovered,
      IncludePaths: ['**/*'],
      ExcludePaths: ['*/node_modules/**/*'],
    },
  },
  // execute some arbitrary steps
```

```
  steps: [
    'npm install',
    'npm run myscript',
    'echo hello-world',
  ],
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

工作流程程序元件

下列範例顯示如何使用 Projen 元件將工作流程 YAML 寫入儲存庫：

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

...

const repo = new SourceRepository
const blueprint = this;
const workflowDef = workflowBuilder.getDefinition()

// creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

連線至環境

許多工作流程都需要在 AWS 帳戶連線中執行。工作流程透過允許動作連接至具有帳戶和角色名稱規格的環境來處理此問題。

```
import { convertToWorkflowEnvironment } from '@amazon-codecatalyst/codecatalyst-
workflows'


const myEnv = new Environment(...);

// can be passed into a workflow constructor
const workflowEnvironment = convertToWorkflowEnvironment(myEnv);
```

```
// add a build action
workflowBuilder.addBuildAction({
  ...
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

開發環境元件

受管理的開發環境 (MDE) 可用來在中建立和站立 MDE 工作區。 CodeCatalyst元件會產生一個`devfile.yaml`檔案。如需詳細資訊，請參閱 Devfile 和移動開發環境的存儲庫 devfile.

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

若要匯入 Amazon CodeCatalyst 藍圖工作區元件

在您的`blueprint.ts`檔案中，新增下列內容：

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

問題元件

在中 CodeCatalyst，您可以監視專案中涉及的功能、工作、錯誤以及任何其他工作。每件作品都保存在一個稱為問題的不同記錄中。每個問題都可以有描述、工作負責人、狀態和其他屬性，您可以搜尋、分組和篩選。您可以使用預設檢視來檢視您的問題，也可以使用自訂篩選、排序或群組來建立自己的檢視。如需有關「問題」概念的詳細資訊，請參閱問題, 概念和中的問題配額 CodeCatalyst。

問題元件會產生問題的 JSON 表示法。該組件接受一個 ID 字段和問題定義作為輸入。

要導入 Amazon CodeCatalyst 藍圖問題組件

在您的`blueprint.ts`檔案中，新增下列內容：

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

主題

- 問題元件範例

問題元件範例

建立問題

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

建立高優先順序問題

```
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
...
const repo = new SourceRepository
const blueprint = this;
const workflowDef = workflowBuilder.getDefinition()

// Creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// Can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

使用標籤建立低優先順序問題

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myThirdIssue', {
  title: 'myThirdIssue',
  content: 'This is an example of a low priority issue with a label.',
  priority: 'LOW',
  labels: ['exampleLabel'],
});
```

使用藍圖工具和 CLI

藍圖 CLI 提供用於管理和使用自訂藍圖的工具。

主題

- 使用藍圖工具
- 圖片上傳工具

使用藍圖工具

若要使用藍圖工具

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 恢復您的開發環境。如需詳細資訊，請參閱 恢復開發環境。

   如果您沒有開發環境，則必須先創建一個開發環境。如需詳細資訊，請參閱 建立開發環境。
3. 在工作中的終端機中，執行下列命令以安裝藍圖 CLI：

   ```
   npm install -g @amazon-codecatalyst/blueprint-util.cli
   ```
4. 在`blueprint.ts`檔案中，以下列格式匯入您要使用的工具：

   ```
   import { <tooling-function-name> } from '@amazon-codecatalyst/blueprint-util.cli/
   lib/<tooling-folder-name>/<tooling-file-name>;
   ```

   > **Tip**
   >
   > 您可以CodeCatalyst blueprints GitHub repository到以尋找您要使用之工具的名稱。

   如果要使用圖像上傳工具，請將以下內容添加到腳本中：

   ```
   import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
   image-upload-tool/upload-image-to-aws';
   ```

   範例

   - 如果要使用發佈函數，請將以下內容新增至指令碼：

     ```
     import { publish } from '@amazon-codecatalyst/blueprint-util.cli/lib/publish/
     publish';
     ```
   - 如果要使用圖像上傳工具，請將以下內容添加到腳本中：

```
import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
image-upload-tool/upload-image-to-aws';
```

5. 呼叫函數。

   範例：

   - 如果要使用發佈函數，請將以下內容新增至指令碼：

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

   - 如果要使用圖像上傳工具，請將以下內容添加到腳本中：

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

圖片上傳工具

圖像上傳工具可讓您將自己的映像上傳到 AWS 帳戶中的 S3 儲存貯體，然後將該映像公開分發到後面 CloudFront。該工具將本地存儲中的圖像路徑（以及可選存儲桶名稱）作為輸入，並將 URL 返回到可公開使用的圖像。如需詳細資訊，請參閱什麼是 Amazon CloudFront？ 什麼是 Amazon S3？

使用影像上傳工具

1. 複製可 GitHub 存取藍圖 SDK 和範例藍圖的開放原始碼藍圖存放庫。在工作終端中，運行以下命令：

   ```
   git clone https://github.com/aws/codecatalyst-blueprints.git
   ```

2. 執行下列命令以導覽至藍圖 GitHub 儲存庫：

   ```
   cd codecatalyst-blueprints
   ```

3. 執行下列命令以安裝相依性：

   ```
   yarn && yarn build
   ```

4. 執行下列命令以確保已安裝最新的藍圖 CLI 版本：

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

5. 使用您要將映像上傳到的 S3 儲存貯體登入 AWS 帳戶。如需詳細資訊，請參閱設定 AWS CLI 和透過 AWS 命令列界面登入。
6. 從 CodeCatalyst 存放庫的根目錄執行下列命令，以使用藍圖 CLI 導覽至目錄：

```
cd packages/utils/blueprint-cli
```

7. 執行下列命令，將映像上傳至 S3 儲存貯體：

```
yarn blueprint upload-image-public <./path/to/your/image>
      <optional:optional-bucket-name>
```

您的圖片的 URL 隨即產生。該 URL 將無法立即使用，因為它需要一些時間才能部署 CloudFront 分發。檢查發佈狀態以取得最新的部署狀態。如需詳細資訊，請參閱使用發行版。

快照測試

支援跨藍圖的多個組態產生的快照測試。

藍圖支援針對您身為藍圖作者所提供的組態進行快照測試。組態是合併在藍圖根目錄之預設 .json 檔案之上的部分覆寫。啟用並配置快照測試後，構建和測試過程將合成給定的配置，並驗證合成輸出是否未從參考快照更改。若要檢視快照測試程式碼，請參閱CodeCatalyst 藍圖 GitHub 儲存庫。

啟用快照測試

1. 在.projenrc.ts檔案中， ProjenBlueprint 使用您要快照的檔案更新輸入物件。例如：

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

2. 重新同步藍圖以在藍圖專 TypeScript 案中建立檔案。不要編輯源文件，因為它們是由 Projen 維護和重新生成的。使用下列命令：

```
yarn projen
```

3. 導覽至目src/snapshot-configurations錄以檢視含有空物件的default-config.json檔案。使用您自己的一個或多個測試配置更新或替換文件。然後，每個測試配置與項目的defaults.json文件合併，合成，並在測試時與快照進行比較。使用以下命令進行測試：

```
yarn test
```

第一次使用 test 指令時，會顯示下列訊息：Snapshot Summary › NN snapshots written from 1 test suite。隨後的測試運行驗證合成輸出沒有從快照更改，並顯示以下消息：Snapshots: NN passed, NN total。

如果您故意變更藍圖以產生不同的輸出，請執行下列命令來更新參考快照：

```
yarn test:update
```

快照期望合成輸出在每次運行之間保持不變。如果您的藍圖產生不同的檔案，您必須從快照測試中排除這些檔案。更新ProjenBluerpint輸入blueprintSnapshotConfiguration物件的物件以新增snapshotGlobs屬性。此snapshotGlobs屬性是 globs 陣列，可決定哪些檔案要包含在快照集之外或排除。

> **Note**
>
> 有一個默認的全局列表。如果您指定自己的清單，您可能需要明確恢復預設項目。

發佈自訂藍圖

自訂藍圖提供成功合成的結果預覽服務包。該項目包代表項目中的源代碼，配置和資源，並且它被 CodeCatalyst 部署 API 操作部署到項目中使用。如果您要繼續開發自訂藍圖，請重新執行藍圖合成程序。如需詳細資訊，請參閱 自訂藍圖概念。

> **Important**
>
> 如果您想要使用來自外部來源的藍圖套件，請考慮這些套件可能帶來的風險。您必須負責新增至空間的自訂藍圖及其產生的程式碼。

主題

- 檢視和發佈自訂藍圖的預覽版本
- 檢視和發佈自訂藍圖的一般版本
- 在指定的空間和專案中發佈和套用自訂藍圖

檢視和發佈自訂藍圖的預覽版本

如果要將自訂藍圖新增至空間的藍圖目錄，可以將自訂藍圖的預覽版本發佈到您的空間。這可讓您在將非預覽版本新增至目錄之前，以使用者身分檢視藍圖。預覽版可讓您在不佔用實際版本的情況下發佈。例如，如果您使用某個`0.0.1`版本，則可以發佈並新增預覽版本，以便可以將第二個版本的新更新發行並新增為`0.0.2`。

進行變更後，透過執行檔案來重建自訂藍圖的套`package.json`件，並預覽您的變更。

若要檢視和發佈自訂藍圖的預覽版本

1. 恢復您的開發環境。如需更多資訊，請參閱 恢復開發環境
2. 在您的開發環境中打開一個工作終端。
3. （可選）在工作終端中，如果尚未安裝項目，請為項目安裝必要的依賴關係。使用下列命令：

   ```
   yarn
   ```

4. (選擇性) 如果您對`.projenrc.ts`檔案進行了變更，請在建置和預覽藍圖之前重新產生專案的組態。使用下列命令：

   ```
   yarn projen
   ```

5. 使用以下命令重建和預覽您的自訂藍圖。使用以下命令：

   ```
   yarn blueprint:preview
   ```

   導覽至提供的`See this blueprint at:`連結以預覽您的自訂藍圖。根據您的組態，檢查 UI (包括文字) 是否如預期般顯示。如果您想要變更自訂藍圖，可以編輯`blueprint.ts`檔案、重新同步藍圖，然後再次發佈預覽版本。如需詳細資訊，請參閱 樹脂同步。
6. (選擇性) 您可以發佈自訂藍圖的預覽版本，然後將其新增至空間的藍圖目錄。導覽至`Enable version` *`[preview version number]`* `at:`連結以將預覽版本發佈至您的空間。

您可以模擬專案建立，而不必在中 CodeCatalyst建立專案。要合成您的項目，請使用以下命令：

```
yarn blueprint:synth
```

藍圖會在synth/synth.*[options-name]*/proposed-bundle/資料夾中產生。如需詳細資訊，請參閱 合成。

如果您要更新自訂藍圖，請改為使用下列命令重新同步專案：

```
yarn blueprint:resynth
```

藍圖會在synth/synth.*[options-name]*/proposed-bundle/資料夾中產生。如需詳細資訊，請參閱 樹脂同步。

發佈預覽版本後，您可以新增藍圖，以便空間成員可以使用它來建立新專案或在現有專案中套用。如需詳細資訊，請參閱 將自訂藍圖新增至空間目錄。

檢視和發佈自訂藍圖的一般版本

完成開發和預覽自訂藍圖之後，您可以檢視並發佈要新增至空間藍圖目錄的新版本。建立專案時產生的發行工作流程會自動發佈已推送的變更。如果您在建立藍圖時關閉工作流程產生，則您的藍圖不會自動新增至空間的藍圖目錄。執行yarn命令後，您仍然可以將自訂藍圖發佈到您的空間。

若要檢視和發佈自訂藍圖

1. 恢復您的開發環境。如需更多資訊，請參閱 恢復開發環境
2. 在您的開發環境中打開一個工作終端。
3.
   - 如果您在建立藍圖時選擇退出版本工作流程產生，請使用下列命令：

     ```
     yarn blueprint:release
     ```

     您仍然可以導覽至提供的`See this blueprint at:`連結以檢視您的自訂藍圖。

     發佈自訂藍圖的更新版本，然後可將其新增至空間的藍圖目錄。導覽至`Enable version [release version number] at:`連結，將最新版本發佈至您的空間。
   - 如果您在建立藍圖時選擇加入發行工作流程，則推送變更時會自動發佈最新的藍圖版本。使用下列命令：

     ```
     git add .
     ```

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

```
git push
```

發佈正常版本後，您可以新增藍圖，以便空間成員可以使用它來建立新專案或在現有專案中套用藍圖。如需詳細資訊，請參閱 將自訂藍圖新增至空間目錄。

在指定的空間和專案中發佈和套用自訂藍圖

依預設，`blueprint:preview`和命`blueprint:release`令會發佈到您在其中建立藍圖的 CodeCatalyst 空間中。如果您有多個企業空間，您也可以在這些空間中預覽和發佈相同的藍圖。您也可以將藍圖套用至另一個空間的現有專案。

在指定空間中發佈或套用自訂藍圖

1. 恢復您的開發環境。如需詳細資訊，請參閱 恢復開發環境。
2. 在您的開發環境中打開一個工作終端。
3. （可選）如果尚未安裝項目，請為項目安裝必要的依賴項。使用下列命令：

```
yarn
```

4. 使用標`--space`籤將預覽或一般版本發佈到指定的空間。例如：

- ```
  yarn blueprint:preview --space my-awesome-space # publishes under a "preview"
   version tag to 'my-awesome-space'
  ```

  輸出範例：

  ```
  Enable version 0.0.1-preview.0 at: https://codecatalyst.aws/spaces/my-awesome-
  space/blueprints
  Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
  blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
  a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.0/projects/create
  ```

- ```
  yarn blueprint:release --space my-awesome-space # publishes normal version to
   'my-awesome-space'
  ```

輸出範例：

```
Enable version 0.0.1 at: https://codecatalyst.aws/spaces/my-awesome-space/
blueprints
Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1/projects/create
```

使用`--project`將自訂藍圖的預覽版本套用至指定空間中的現有專案。例如：

```
yarn blueprint:preview --space my-awesome-space --project my-project # previews
 blueprint application to an existing project
```

輸出範例：

```
Enable version 0.0.1-preview.1 at: https://codecatalyst.aws/spaces/my-awesome-
space/blueprints
Blueprint applied to [my-project]: https://codecatalyst.aws/spaces/my-awesome-
space/projects/my-project/blueprints/%40amazon-codecatalyst%2FmySpace.my-blueprint/
publishers/1524817d-a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.1/add
```

檢視自訂藍圖的詳細資料、版本和專案

您可以檢視空間的已發佈自訂藍圖，包括藍圖的詳細資料、版本以及使用藍圖建立或套用藍圖的專案。

主題

- 檢視空間的自訂藍圖
- 檢視使用或套用自訂藍圖建立的專案

檢視空間的自訂藍圖

檢視空間的自訂藍圖

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在主 CodeCatalyst 控台中，導覽至您要檢視自訂藍圖的空間。

3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖] 以檢視空間藍圖。下列詳細資訊會顯示在表格中：

   - 名稱-自訂藍圖的名稱。
   - 目錄狀態-是否將自訂藍圖發佈至空間的藍圖目錄。
   - 最新版本-自訂藍圖的最新版本。
   - 最新修改日期-上次更新空間藍圖的日期。

檢視使用或套用自訂藍圖建立的專案

檢視使用自訂藍圖建立或套用自訂藍圖的專案

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在主 CodeCatalyst 控台中，導覽至您要檢視自訂藍圖的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 從 S pace 藍圖表格中，選擇自訂藍圖的名稱以檢視使用藍圖的專案和不使用藍圖表格的專案。

在空間中新增和移除自訂藍圖

將自訂藍圖發佈到空間後，即可將其新增至空間的藍圖目錄。如果您不再希望自訂藍圖用於建立新專案或套用至現有專案，也可以從空間的藍圖目錄中移除該藍圖。

主題

- 將自訂藍圖新增至空間目錄
- 從空間目錄移除自訂藍圖

將自訂藍圖新增至空間目錄

如果您將自訂藍圖新增至 CodeCatalyst 空間的藍圖目錄，則該藍圖可供所有空間成員在建立專案或將其套用至現有專案時使用。將自訂藍圖新增至空間的藍圖目錄之前，必須先啟用藍圖的發佈權限。如果您選擇產生工作流程發行版本，則依預設會啟用發佈權限。如需詳細資訊，請參閱 管理自訂藍圖的發佈權限 及 發佈自訂藍圖。

將藍圖新增至空間的藍圖目錄

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 只能從來源存放庫的預設分支新增藍圖。如果您在功能分支上開發藍圖，請將功能分支與預設分支的變更合併。建立提取要求，將任何變更合併至預設分支。如需詳細資訊，請參閱在 Amazon 中使用拉取請求 CodeCatalyst。
3. 在 CodeCatalyst 主控台中，導覽至具有自訂藍圖的空間儀表板。
4. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
5. 選擇您要新增的藍圖名稱，然後選擇 [新增至目錄]。如果您有多個版本，請從「目錄版本」下拉式選單中選擇一個版本
6. 選擇 儲存 。

從空間目錄移除自訂藍圖

> **Note**
>
> 如果您從空間目錄移除自訂藍圖，則不會影響從藍圖建立的專案或套用藍圖的專案。藍圖的資源不會從專案中移除。

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在 CodeCatalyst 主控台中，使用您的自訂藍圖導覽至空間儀表板
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 選擇您要移除的藍圖名稱，然後選擇從目錄移除藍圖。

管理自訂藍圖的發佈權限

依預設，如果在專案建立期間產生工作流程版本，則會啟用自訂藍圖的權限。啟用發佈權限時，可將藍圖發佈至空間。您可以停用權限，以便無法發佈藍圖。停用權限時，不會執行在藍圖建立期間產生的發行工作流程。除非啟用藍圖的權限，否則無法發佈對藍圖的新變更。

> **Important**
>
> 若要啟用或停用藍圖專案的發佈權限，您必須具有 S pace 管理員角色。

管理藍圖專案的發佈權限

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。

2. 在 CodeCatalyst 主控台中，導覽至您要管理自訂藍圖發佈權限的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 選擇專案發佈權限索引標籤，以檢視所有空間藍圖的發佈權限。
5. 選擇您要管理的藍圖，然後選擇 [啟用] 或 [停用] 以變更發佈權限。如果您要啟用權限，請檢閱權限變更詳細資料，然後選擇 [啟用藍圖發佈] 以確認變更。

管理自訂藍圖的版本

身為藍圖作者，您可以管理要發佈到空間藍圖目錄的版本。變更藍圖的目錄版本不會影響使用不同藍圖版本的專案。

管理自訂藍圖版本

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在 CodeCatalyst 主控台中，導覽至要變更自訂藍圖版本的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 在空間藍圖表格上，為您要管理的自訂藍圖選擇圓鈕。
5. 選擇 [建立目錄版本]，然後從 [目錄版本] 下拉式功能表中選擇版本。
6. 選擇 Save (儲存)。

刪除已發佈的自訂藍圖或版本

當您從 Amazon CodeCatalyst 空間刪除自訂藍圖的版本或藍圖本身時，您對藍圖專案或藍圖版本資源的所有存取權都會移除。刪除藍圖版本或藍圖後，專案成員將無法存取專案資源，並且會停止第三方來源存放庫提示的任何工作流程。

> **Note**
>
> 如果刪除藍圖，則不會影響已套用藍圖的專案。藍圖的資源不會從專案中移除。

如果藍圖版本已發佈至空間的藍圖目錄，請先為目錄選擇新版本，然後再刪除已發佈的版本。

刪除自訂藍圖的目錄版本

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 在 CodeCatalyst 主控台中，導覽至要刪除自訂藍圖目錄版本的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 選擇具有您要刪除之目錄版本的藍圖名稱。
5. 選擇您要刪除之目錄版本的圓鈕，然後選擇 [刪除版本]。
6. 檢閱詳細資料，然後從 [選擇新的藍圖目錄版本] 下拉式功能表中選擇另一個藍圖版本。
7. 輸入`delete`以確認藍圖目錄版本的選擇。
8. 選擇刪除。

如果藍圖版本不在空間的藍圖目錄中，您可以刪除該版本而無需選擇新版本。

刪除自訂藍圖版本

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在 CodeCatalyst 主控台中，導覽至要刪除自訂藍圖版本的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 選擇具有您要刪除之版本的藍圖名稱。
5. 選擇您要刪除之版本的圓鈕，然後選擇 [刪除版本]。
6. 輸入`delete`以確認刪除藍圖版本。
7. 選擇刪除。

從空間的藍圖目錄刪除藍圖會刪除藍圖的所有版本。使用藍圖的空間專案不會受到刪除的影響。

刪除自訂藍圖版本

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在主 CodeCatalyst 控台中，導覽至要刪除自訂藍圖的空間。
3. 在空間儀表板上，選擇 [設定] 索引標籤，然後選擇 [藍圖]。
4. 在空間藍圖表格上，選擇要刪除之自訂藍圖的圓鈕，然後選擇 [刪除藍圖]。
5. 輸入`delete`以確認刪除自訂藍圖。
6. 選擇刪除。

使用相依性和工具

主題

- 新增相依性
- 處理依賴類型不匹配
- 使用紗線和 npm
- 升級模具和元件

新增相依性

身為藍圖作者，您可能需要將套件新增至藍圖，例如`@amazon-codecatalyst/blueprint-component.environments`. 您需要使用該軟`projen.ts`件包更新文件，然後使用 Projen 重新生成項目的配置。Projen 充當每個藍圖程式碼庫的專案模型，透過變更模型呈現組態檔的方式，提供向後推送相容工具更新的功能。該`package.json`文件是由 Projen 模型部分擁有的文件。Projen 承認 package.json 文件中包含的依賴版本，但其他選項需要來自模型。

若要新增相依性和更新**`projenrc.ts`**檔案

1. 在`projen.ts`檔案中，導覽至 deps 區段。
2. 新增您要在藍圖中使用的相依性。
3. 使用以下命令重新生成項目的配置：

```
yarn projen && yarn
```

處理依賴類型不匹配

在 Yarn 更新之後，您可能會收到有關儲存庫參數的下列錯誤：

```
Type 'SourceRepository' is missing the following properties from type
 'SourceRepository': synthesisSteps, addSynthesisStep
```

該錯誤是由於某個組件依賴於另一個組件的較新版本，但依賴組件固定到舊版本時，會發生相依性不匹配。可以通過使所有組件依賴相同的版本來修復錯誤，以便它們之間的版本同步。除非您確定如何處理這些版本，否則最好將 al blueprint-vended 軟件包保留在相同的最新版本（`0.0.x`）下。下列範例顯示如何設定`package.json`檔案，讓所有相依性都依賴相同的版本：

```
...
"@caws-blueprint-component/caws-environments": "^0.1.12345",
"@caws-blueprint-component/caws-source-repositories": "^0.1.12345",
```

```
"@caws-blueprint-component/caws-workflows": "^0.1.12345",
"@caws-blueprint-component/caws-workspaces": "^0.1.12345",
"@caws-blueprint-util/blueprint-utils": "^0.1.12345",
...
"@caws-blueprint/blueprints.blueprint": "*",
```

設定所有相依性的版本之後，請使用下列指令：

```
yarn install
```

使用紗線和 npm

藍圖使用紗線進行工具。使用 npm 和 Yarn 會導致工具問題，因為依賴樹的解決方式是不同的。為了避免此類問題，最好只使用紗線。

如果您不小心使用 npm 安裝了依賴關係，則可以刪除生成的`package-lock.json`文件，並確保您的`.projenrc.ts`文件已使用所需的依賴項進行更新。您可以使用 Projen 重新產生專案的組態。

使用下列項目從模型中再生：

```
yarn projen
```

確定您的 .projenrc.ts 檔案已更新為必要的相依性之後，請使用下列命令：

```
yarn
```

升級模具和元件

有時候，您可能會想要升級工具和元件，以引入可用的新功能。除非您確定如何處理版本，否則建議您將所有組件保留在相同的版本上。版本在組件之間進行同步，因此所有組件的相同版本可確保它們之間的適當依賴關係。

使用紗線工作區

使用下列命令從自訂藍圖的存放庫的根目錄升級 utils 和元件：

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

如果您沒有使用 monorepo，請使用以下命令：

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

您可以用來升級工具和元件的其他選項：

- 使用 npm 視圖`@caws-blueprint-component/<some-component>`獲取最新版本。
- 通過在 package.json 文件中設置版本並使用以下命令來手動增加到最新版本：。`yarn`所有組件和應用程序都應具有相同的版本。

貢獻

藍圖軟體開發套件 (SDK) 是您可以貢獻的開放原始碼程式庫。作為貢獻者，請考慮貢獻指南，反饋和缺陷。如需詳細資訊，請參閱藍圖 GitHub 存放庫。

中藍圖的配額 CodeCatalyst

下表說明 Amazon CodeCatalyst 中藍圖的配額和限制。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

每 CodeCatalyst 個專案套用的藍圖數目上限	100

來源儲存庫 CodeCatalyst

CodeCatalyst 源存儲庫是在 Amazon 託管的 Git 存儲庫 CodeCatalyst。您可以在中使用來源儲存庫 CodeCatalyst 來安全地儲存、編排版本和管理專案的資產。

CodeCatalyst 儲存庫中的資產可以包括：

- Documents
- 源代碼
- 二進制文件

CodeCatalyst 也會使用專案的來源儲存庫來儲存專案的組態資訊，例如工作流程設定檔。

您可以在一個 CodeCatalyst 項目中有多個源存儲庫。例如，您可能想要為前端原始程式碼、後端原始程式碼、公用程式和文件建立個別的原始碼儲存庫。

以下是一個可能的工作流程，用於在源存儲庫，提取請求和開發環境中的代碼 CodeCatalyst：

Mary Major 使用藍圖建立 Web 應 CodeCatalyst 用程式專案，該藍圖會建立含範例程式碼的來源儲存庫。她邀請她的朋友李娟, 薩恩維·薩卡爾, 和豪爾赫·索薩與她一起工作的項目. 李娟查看源代碼庫中的示例代碼，並決定進行一些快速更改以將測試添加到代碼中。Li 創建一個開發環境，選擇 AWS Cloud9 作為 IDE，並指定一個新的分支，####。開發環境隨即開啟。Li 快速添加代碼，然後提交並將分支與中 CodeCatalyst的源存儲庫的更改推送到. Li 然後創建一個拉請求。作為創建拉請求的一部分, 李增加了豪爾赫·索薩和薩恩維·薩卡爾作為審查者，以確保代碼被審查.

在審查代碼時，豪爾赫·索薩（Jorge Souza）記得他有自己的項目存儲庫，其中包 GitHub 含他們正在使用的應用程序的原型。他要求 Mary Major 安裝和配置擴展程序，這將允許他將 GitHub 存儲庫鏈接到項目作為附加源存儲庫。Mary 審查存儲庫， GitHub 並與 Jorge 一起配置 GitHub 擴展，以便他可以將 GitHub 存儲庫鏈接為項目的附加源存儲庫。

CodeCatalyst 來源儲存庫支援 Git 的標準功能，並可搭配現有的 GIT 工具使用。在從 Git 用戶端或整合式開發環境 (IDE) 複製和使用來源儲存庫時，您可以建立個人存取權杖 (PAT) 並將其用作應用程式特定的密碼。這些 PAT 與您的 CodeCatalyst使用者身分相關聯。如需詳細資訊，請參閱 在 Amazon 中管理個人訪問令牌 CodeCatalyst。

CodeCatalyst 來源儲存庫支援提取要求。這是您和其他項目成員在將代碼更改從一個分支合併到另一個分支之前查看和註釋代碼更改的簡單方法。您可以在 CodeCatalyst 控制台中查看更改並對代碼行進行註釋。

推送至 CodeCatalyst 來源儲存庫中的分支可以在工作流程中自動開始執行，在工作流程中可以建置、測試和部署變更。如果您的來源儲存庫是使用專案樣板建立為專案的一部分，則會為您配置一或多個工作流程，做為專案的一部分。您可以隨時為儲存庫新增其他工作流程。專案中工作流程的 YAML 設定檔會儲存在針對這些工作流程的來源動作中設定的來源存放庫中。如需更多詳細資訊，請參閱 開始使用中的工作流程 CodeCatalyst。

主題

- 來源儲存庫概念
- 設定使用來源儲存庫
- 開始使用 CodeCatalyst 來源儲存庫和單頁應用程式藍圖
- 使用中的來源儲存庫 CodeCatalyst
- 與 Amazon 的分支機構合作 CodeCatalyst
- 在 Amazon 中使用文件 CodeCatalyst
- 在 Amazon 中使用拉取請求 CodeCatalyst
- 在 Amazon 中處理提交 CodeCatalyst
- 來源儲存庫的配額 CodeCatalyst

來源儲存庫概念

以下是使用 CodeCatalyst 原始碼儲存庫時需要瞭解的一些概念。

主題

- 專案
- 來源儲存庫
- 開發環境
- 個人存取權杖 (PAT)
- 分支
- 預設分支
- 遞交
- 提取請求
- 修訂

- 工作流程

專案

專案代表中 CodeCatalyst 支援開發團隊和任務的協同合作工作。建立專案後，您可以新增、更新或移除使用者和資源、自訂專案儀表板，以及監控團隊工作進度。您可以在一個空間中擁有多個專案。

來源儲存庫特定於您在空間中建立或連結它們的專案。您無法在專案之間共用儲存庫，也無法將儲存庫連結至空間中的多個專案。在專案中具有 Contributor 或專案管理員角色的使用者，可以根據授予這些角色的權限，與該專案相關聯的來源儲存庫互動。如需詳細資訊，請參閱 在 Amazon 中使用角色 CodeCatalyst。

來源儲存庫

源存儲庫是您安全地存儲項目的代碼和文件的地方。它還存儲文件的版本歷史記錄。默認情況下，源存儲庫與 CodeCatalyst 項目中的其他用戶共享。您可以為一個項目擁有多個源存儲庫。您可以為中的專案建立來源儲存庫 CodeCatalyst，或者如果已安裝的擴充功能支援該服務，也可以選擇連結由其他服務託管的現有來源儲存庫。例如，您可以在安裝儲 GitHub 存庫擴充功能之後，將存放GitHub 庫連結至專案。如需詳細資訊，請參閱 使用中的來源儲存庫 CodeCatalyst 及 快速入門：在中使用 GitHub 儲存庫 CodeCatalyst。

開發環境

開發環境是一種雲端式開發環境，您可以使用它 CodeCatalyst 來快速處理儲存在專案原始碼儲存庫中的程式碼。開發環境中包含的項目工具和應用程序庫由項目源存儲庫中的 devfile 定義。如果您的源存儲庫中沒有 devfile，則會自動應用默認的 devfile。默認的 devfile 包括最常用的編程語言和框架的工具。根據預設，開發環境設定為具有 2 核心處理器、4 GB 的 RAM 和 16 GiB 的持續性儲存裝置。

您可以選擇將來源存放庫的現有分支複製到您的開發環境中，或者您可以選擇建立新分支作為建立開發環境的一部分。

個人存取權杖 (PAT)

個人存取權杖 (PAT) 類似於密碼。它與您的使用者身分相關聯，以便在中的所有空間和專案中使用 CodeCatalyst。您可以使用 PAT 來存取包含整合式開發環境 (IDE) 和 Git 型來 CodeCatalyst 源存放庫的資源。PAT 代表您，您可 CodeCatalyst 以在使用者設定中管理它們。使用者可以擁有多個 PAT。個人存取權杖只會顯示一次。最佳做法是，請務必將它們安全地儲存在本機電腦上。依預設，PAT 會在一年後過期。

使用整合式開發環境 (IDE) 時，PAT 相當於 Git 密碼。在設定 IDE 以使用 Git 儲存庫時，在系統要求輸入密碼時提供 PAT。有關如何將 IDE 與基於 Git 的存儲庫連接的詳細信息，請參閱 IDE 的文檔。

分支

分支是指向 Git 和中提交的指針或引用 CodeCatalyst。您可以使用分支來組織您的工作。例如，您可以使用分支來處理新版或不同版本的檔案，而不會影響其他分支中的檔案。您可以使用分支來開發新功能、儲存專案的特定版本等等。源存儲庫可以有一個或多個分支。當您使用範本建立專案時，為專案建立的來源儲存庫會包含名為 main 的分支中的範例檔案。主要分支是儲存庫的預設分支。

預設分支

中的源存儲庫 CodeCatalyst 具有默認分支，無論您如何創建它們。如果您選擇使用範本建立專案，除了範例程式碼、工作流程定義和其他資源之外，針對該專案建立的來源存放庫還包含一個 README.md 檔案。如果您在未使用範本的情況下建立來源儲存庫，則會在第一次提交時為您新增一個 README.md 檔案，並在建立儲存庫時為您建立預設分支。這個默認分支被命名為 main。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。您可以變更使用哪個分支做為預設分支。如需詳細資訊，請參閱 檢視和變更儲存庫的預設分支。

您無法刪除來源儲存庫的預設分支。搜尋結果僅包含來自預設分支的結果。

遞交

提交是對一個文件或一組文件的更改。在 Amazon 主 CodeCatalyst控台中，提交會儲存您的變更，並將其推送至來源儲存庫。提交包含有關變更的資訊，包括進行變更的使用者身分、變更的時間和日期、提交標題，以及任何包含有關變更的訊息。如需詳細資訊，請參閱 在 Amazon 中處理提交 CodeCatalyst。

在中的來源儲存庫環境中 CodeCatalyst，認可是儲存庫內容的快照以及對儲存庫內容所做的變更。您還可以將 Git 標籤添加到提交中，以識別特定的提交。

提取請求

提取要求是您和其他使用者在來源儲存庫中檢閱、註解和合併程式碼變更的主要方式。您可以使用提取請求協同檢閱程式碼變更，以瞭解次要變更或修正、主要新增功能或發行軟體的新版本。在提取請求中，您可以檢視來源與目標分支之間的變更，或是這些分支的修訂之間的差異。您可以在各行程式碼變更中新增註解，以及整體提取要求的註解。

> **Tip**
>
> 建立提取要求時，顯示的差異在於來源分支尖端與目標分支尖端之間的差異。一旦建立了提取請求，顯示的差異將在您選擇的提取請求的修訂版本與建立提取請求時作為目的地分支提示的確認之間。有關 Git 中差異和合併基礎的更多信息，請參閱 Git 文檔git-merge-base中的。

修訂

修訂版是提取請求的更新版本。每次推送至提取要求的 source 分支，都會建立一個修訂版，其中包含該推送中包含的認可中所做的變更。除了來源和目的地分支之間的差異之外，您還可以檢視提取請求修訂之間的差異。如需詳細資訊，請參閱 在 Amazon 中使用拉取請求 CodeCatalyst。

工作流程

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要採取的一系列步驟或動作。工作流程也會定義導致工作流程啟動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

來源儲存庫也可以儲存專案的工作流程、通知、問題及其他組態資訊的組態檔和其他資訊。當您建立需要組態檔的資源，或將存放庫指定為工作流程的來源動作時，會建立組態檔並儲存在來源儲存庫中。如果您從藍圖建立專案，則組態檔已經儲存在為您建立的來源存放庫中，做為專案的一部分。此組態資訊儲存在儲存庫預設分支.codecatalyst中名為的資料夾中。每當您建立預設分支的分支時，除了該分支中的所有其他檔案和資料夾之外，都會建立此資料夾及其組態的副本。

設定使用來源儲存庫

當您在本機電腦 CodeCatalyst 上使用 Amazon 中的原始碼儲存庫時，您可以單獨使用 Git 或在受支援的整合式開發環境 (IDE) 中使用 Git 來變更程式碼並推送和提取程式碼。根據最佳實務，我們建議您使用 Git 和其他軟體的最新版本。

> **Note**
>
> 如果您使用開發環境，則不需要安裝 Git。您的開發環境中包含了最新版本的 Git。

的版本相容性資訊 CodeCatalyst

元件	版本
Git	最新

安裝 Git

若要在不使用 IDE 的情況下從 Git 用戶端處理來源儲存庫中的檔案、提交、分支和其他資訊，請在本機電腦上安裝 Git。

若要安裝 Git，我們建議使用 Git 下載等網站。

建立個人存取權杖

若要將來源儲存庫複製到本機電腦或偏好的 IDE，您必須建立個人存取權杖 (PAT)。

若要建立個人存取權杖 (PAT)

1. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。

   > **Tip**
   >
   > 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

2. 在 PAT 名稱中，輸入 PAT 的描述性名稱。
3. 在到期日中，保留預設日期，或選擇行事曆圖示以選取自訂日期。到期日預設為從目前日期算起一年。
4. 選擇建立。

   當您為來源儲存庫選擇複製儲存庫時，也可以建立此權杖。
5. 將 PAT 秘密保存在安全的位置。

> **Important**
>
> PAT 秘密僅顯示一次。關閉視窗之後，您無法擷取它。

開始使用 CodeCatalyst 來源儲存庫和單頁應用程式藍圖

請遵循本教學中的步驟，了解如何使用 Amazon 中的來源儲存庫 CodeCatalyst。

開始在 Amazon 中使用來源儲存庫的最快方法 CodeCatalyst 是使用範本建立專案。當您使用範本建立專案時，會為您建立資源，包括包含範例程式碼的來源儲存庫。您可以使用此存儲庫和代碼示例來學習如何：

- 檢視專案的來源儲存庫並瀏覽其內容
- 使用新的分支創建一個開發環境，您可以在其中處理代碼
- 更改文件，並提交並推送更改
- 建立提取要求，並與其他專案成員一起檢閱您的程式碼變更
- 查看項目的工作流程自動構建並測試拉取請求的源分支中的更改
- 將來源分支中的變更合併到目標分支，並關閉提取要求
- 查看自動構建和部署的合併更改

為了充分利用本教程，請邀請其他人加入您的項目，以便您可以一起處理提取請求。您也可以探索中的其他功能 CodeCatalyst，例如建立問題並將問題與提取要求產生關聯，或是設定通知並在相關工作流程執行時接收警示。如需完整的探索 CodeCatalyst，請參閱入門教學課程。

使用藍圖建立專案

創建一個項目是能夠一起工作的第一步。您可以使用藍圖來建立專案，這也會建立包含範例程式碼和工作流程的來源儲存庫，以便在您變更程式碼時自動建置和部署程式碼。在本教學課程中，我們將引導您完成使用單頁應用程式藍圖建立的專案，但是您可以針對具有來源存放庫的任何專案遵循程序。如果在建立專案時沒有 IAM 角色，請務必選擇 IAM 角色或新增 IAM 角色。建議您針對此專案使用CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色。

如果您已經有一個項目，則可以跳到檢視專案的儲存庫。

> **Note**
>
> 只有具有 Space 管理員或超級使用者角色的使用者才能在中建立專案 CodeCatalyst。如果您沒有這個角色，而且您需要一個專案來進行本教學課程，請要求具有這些角色之一的人為您建立專案，並將您加入至已建立的專案。如需詳細資訊，請參閱 在 Amazon 中使用角色 CodeCatalyst。

使用藍圖建立專案

1. 在主 CodeCatalyst 控台中，導覽至您要建立專案的空間。
2. 在空間儀表板上，選擇建立專案。
3. 選擇從藍圖開始。
4. 選擇藍圖，然後選擇 [下一步]。
5. 在「為您的專案命名」下，輸入您要指派給專案的名稱及其相關聯的資源名稱。名稱在您的空間中必須是唯一的。
6. 在 [專案資源] 下，設定一般專案參數。
7. (選擇性) 若要根據您選取的專案參數來檢視含有更新的定義檔，請從「產生專案預覽」中選擇「檢視程式碼」或「檢視工作流程」。
8. (選擇性) 從藍圖的卡片中選擇檢視詳細資料以檢視有關藍圖的特定詳細資料，例如藍圖架構概觀、所需的連線和權限，以及藍圖建立的資源類型。
9. 選擇建立專案。

   如需專案藍圖的詳細資訊，請參閱專案藍圖參考。

只要您建立專案或接受專案邀請並完成登入程序，專案概觀頁面就會開啟。新專案的專案概觀頁面不包含未決的問題或提取請求。您可以選擇性地選擇建立問題，並將其指定給您自己。您也可以選擇邀請其他人加入您的專案。如需詳細資訊，請參閱 在中建立問題 CodeCatalyst 及 邀請使用者加入您的專案。

檢視專案的儲存庫

身為專案的成員，您可以檢視專案的來源儲存庫。您也可以選擇建立其他儲存庫。如果具有 S pace 管理員角色的使用者已安裝並設定GitHub 儲存庫擴充功能，您也可以在為擴充功能設定的 GitHub 帳戶中新增 GitHub存放庫連結。如需詳細資訊，請參閱 建立來源儲存庫 及 快速入門：在中使用 GitHub 儲存庫 CodeCatalyst。

> **Note**
>
> 對於使用單頁應用程式藍圖建立的專案，包含範例程式碼的來源存放庫的預設名稱為 ***sp*** a-app。

若要導覽至專案的來源儲存庫

1. 導覽至您的專案，然後執行下列其中一項作業：
   - 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。
   - 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。在來源儲存庫中，從清單中選擇存放庫的名稱。您可以在篩選列中輸入部分存放庫名稱來篩選儲存庫清單。
2. 在儲存區域的首頁上，檢視儲存區域的內容以及相關資源的相關資訊，例如提取要求數目和工作流程。依預設，會顯示預設分支的內容。您可以從下拉式清單中選擇不同的分支來變更檢視。

儲存區域的簡介頁面包含針對此儲存區域及其檔案分支所設定之工作流程和提取要求的相關資訊。如果您剛剛建立專案，建置、測試和部署程式碼的初始工作流程仍會執行，因為這些工作流程需要幾分鐘的時間才能完成。您可以選擇「相關」工作流程下的數字來查看相關的工作流程及其狀態，但這樣會在 CI/ CD 中打開「工作流程」頁。在本教學課程中，請停留在概觀頁面，並探索存放庫中的程式碼。檔案內容會呈現在儲存庫`README.md`檔案下方的此頁面上。在檔案中，會顯示預設分支的內容。您可以更改文件視圖以顯示另一個分支的內容（如果有）。該文件`.codecatalyst`夾包含用於項目的其他部分，如工作流 YAML 文件的代碼。

若要檢視資料夾的內容，請選擇資料夾名稱旁邊的箭頭將其展開。例如，選擇旁邊的箭頭`src`來檢視該資料夾中包含的單頁 Web 應用程式的檔案。若要檢視檔案的內容，從清單中選擇檔案。這將打開查看文件，您可以在其中瀏覽多個文件的內容。您也可以在主控台中編輯單一檔案，但若要編輯多個檔案，您必須建立開發環境。

建立開發環境

您可以在 Amazon CodeCatalyst 主控台的來源儲存庫中新增和變更檔案。不過，若要有效地處理多個檔案和分支，我們建議您使用開發環境或將存放庫複製到您的本機電腦。在本教程中，我們將創建一個名為的分支 AWS Cloud9 開發環境**`develop`**。您可以選擇不同的分支名稱，但是藉由命名分支**`develop`**，當您稍後在本教學課程中建立提取要求時，會自動執行工作流程來建置和測試您的程式碼。

> **Tip**
>
> 如果您決定在本機複製儲存庫，而非使用開發環境，或是除了使用開發環境之外，請確定您的本機電腦上有 Git，或者 IDE 包含 Git。如需詳細資訊，請參閱 設定使用來源儲存庫。

使用新分支建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要創建開發環境的項目。
3. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]、選擇 [原始碼儲存庫]，然後選擇您要建立開發環境的存放庫。
4. 在儲存庫首頁上，選擇建立開發環境。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 選擇要複製的存放庫，選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。
7. 選擇性地新增開發環境的別名。
8. 您也可以選擇 [開發環境] 設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
9. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。將在您選擇的 IDE 中打開一個新選項卡，並顯示您的開發環境。您可以編輯代碼並提交和推送更改。

建立開發環境後，您可以編輯檔案、提交變更，並將變更推送至**test**分支。在本教學課程中，編輯`src`資料夾中`App.tsx`檔案中<p>標籤之間的內容，以變更網頁上顯示的文字。提交並推送您的更改，然後返回到選 CodeCatalyst 項卡。

從 AWS Cloud9 開發環境中進行和推送更改

1. 在中 AWS Cloud9，展開側邊導覽功能表以瀏覽檔案。展`src`開並開啟`App.tsx`。
2. 變更<p>標籤內的文字。
3. 儲存檔案，然後使用 Git 選單提交並推送您的變更。或者，在終端機視窗中，使用和git push指令提交git commit並推送變更。

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

> **Tip**
>
> 在成功運行 Git 命令之前，您可能需要將終端中的目錄更改為 Git 存儲庫目錄。

建立提取請求

您可以使用提取請求協同檢閱程式碼變更，以瞭解次要變更或修正、主要新增功能或發行軟體的新版本。在本教程中，您將創建一個提取請求來查看與主分支相比，對##分支所做的更改。在使用範本建立的專案中建立提取要求，也會啟動其關聯工作流程的執行 (如果有的話)。

若要建立提取請求

1. 導航到您的項目。
2. 執行以下任意一項：
   - 在瀏覽窗格中，選擇 [程式碼]，選擇 [提取要求]，然後選擇 [建立提取要求]。
   - 在儲存區域首頁上，選擇 [其他]，然後選擇 [建立提取要求]。
   - 在專案頁面上，選擇 [建立提取要求]。
3. 在源存儲庫中，確保指定的源存儲庫是包含提交代碼的存儲庫。僅當您未從存放庫的主頁面建立提取要求時，此選項才會顯示。
4. 在「目的地」分支中，選擇要在檢閱程式碼之後將其合併到的分支。
5. 在原始碼分支中，選擇包含已提交程式碼的分支。
6. 在提取請求標題中，輸入一個標題，以幫助其他用戶了解需要審查的內容以及原因。
7. (選擇性) 在提取要求說明中，提供問題連結或變更說明等資訊。

> **Tip**
>
> 您可以選擇 [為我撰寫說明]， CodeCatalyst 自動產生提取要求中所包含變更的描述。您可以在將自動產生的描述新增至提取請求後，對其進行變更。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

8. (選擇性) 在問題中，選擇「連結問題」，然後從清單中選擇問題或輸入其 ID。若要取消連結問題，請選擇「取消連結」圖示。
9. (選擇性) 在 [必要的複查者] 中，選擇 [新增必要的審核者]。從專案成員清單中選擇要新增的專案成員。必要的審核者必須先核准變更，才能將提取請求合併到目的地分支。

> **Note**
>
> 您無法同時將審核者新增為必要審核者和選擇性複查者。您無法將自己新增為審核者。

10. (選擇性) 在 [選擇性複查者] 中，選擇 [新增選用複查者]。從專案成員清單中選擇要新增的專案成員。選用審核者不需要核准變更作為需求，才能將提取請求合併到目的地分支中。
11. 查看分支之間的差異。提取請求中顯示的差異在於源分支中的修訂版本和合併基礎之間的更改，這是在提取請求創建時目的地分支的頭部提交。如果沒有顯示任何更改，則分支可能相同，或者您可能已經為源和目標選擇了相同的分支。
12. 當您滿意提取要求包含您要檢閱的程式碼和變更時，請選擇 [建立]。

> **Note**
>
> 建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整個提取請求。您可以使用 @符號後面接到檔案名稱來新增資源連結，例如檔案。

您可以選擇概要，然後複查「工作流程執行」下「提取請求明細」區域中的資訊，來檢視建立此提取請求所開始之相關聯工作流程的相關資訊。若要檢視工作流程執行，請選擇執行。

> **Tip**
>
> 如果您將分支命名為其他名稱**develop**，則工作流程將不會自動運行以構建和測試您的更改。如果您想要進行設定，請編輯工onPullRequestBuildAndTest作流程的 YAML 檔案。如需詳細資訊，請參閱 建立、編輯和刪除工作流程。

您可以對此提取請求發表評論，並要求其他項目成員對其進行評論。您也可以選擇新增或變更選用或必要的審核者。您可以選擇對儲存庫的來源分支進行更多變更，並查看這些已確認的變更如何為提取要求建立修訂版本。如需詳細資訊檢閱提取要求，請參閱更新提取請求在 Amazon 中使用拉取請求 CodeCatalyst、、和檢視工作流程執行狀態與詳細。

合併提取請求

一旦提取請求經過審核並收到必要審核者的核准，您就可以將其來源分支合併到 CodeCatalyst主控台中的目的地分支。合併提取要求也會透過與目標分支相關聯的任何工作流程開始執行變更。在本教程中，您將合併測試分支到 main，這將啟動onPushToMainDeployPipeline工作流的運行。

若要合併提取要求 (主控台)

1. 在提取請求中，選擇您在上一步中建立的提取請求。在提取請求中，選擇「合併」。
2. 從提取請求的可用合併策略中選擇。選擇性地選取或取消選取要在合併提取請求之後刪除來源分支的選項，然後選擇「合併」。合併完成後，提取請求的狀態會變更為「已合併」，且不會再出現在提取請求的預設檢視中。預設檢視會顯示狀態為「開啟」的提取要求。您仍然可以檢視合併的提取請求，但無法核准該請求或變更其狀態。

> **Note**
>
> 如果「合併」按鈕未處於作用中狀態，或者您看到「不可合併」標籤，表示所需的複查者尚未核准提取請求，或者無法在主控台中合併提取請求。 CodeCatalyst 未核准提取請求的審核者會在 [提取請求詳細資料] 區域的 [概觀] 中以時鐘圖示表示。如果所有必要的審核者都已核准提取請求，但「合併」按鈕仍未處於作用中狀態，表示您可能發生合併衝突，或者您已超過該空間的儲存配額。您可以在開發環境中解決目標分支的合併衝突，推送更改，然後合併提取請求，或者您可以解決衝突並在本地合併，然後將包含合併的提交推送到 CodeCatalyst。如需詳細資訊，請參閱合併一個提取請求（Git）和您的 Git 文件。

檢視部署的程式碼

現在是時候查看默認分支中最初部署的代碼，並在自動構建，測試和部署後合併的更改。若要這麼做，您可以返回儲存區域的總覽頁面，並選擇相關工作流程圖示旁邊的數字，或在導覽窗格中選擇 CI/CD，然後選擇「工作流程」。

若要檢視已部署的程式碼

1. 在「工作流程」中`onPushToMainDeployPipeline`，展開「最近的執行」。

> **Note**
>
> 這是使用單頁應用程式藍圖建立之專案的工作流程預設名稱。

2. 最近一次執行是由合併的提取要求提交至`main`分支所啟動的執行，並且可能會顯示「進行中」狀態。從清單中選擇已成功完成的執行，以開啟該執行的詳細資訊。
3. 選擇「變數」。複製應用程式 URL 的值。這是部署的單頁 Web 應用程式的 URL。開啟新的瀏覽器索引標籤並貼上值，以檢視建置和部署的程式碼。保持標籤處於開啟狀態。

4. 返回工作流程執行的清單，並等待最近的執行完成。當它發生時，返回到您打開的選項卡以查看 Web 應用程序並刷新瀏覽器。您應該會看到您在合併的提取請求中所做的變更。

清除資源

一旦您探索了使用源存儲庫和提取請求，您可能需要刪除任何不需要的資源。您無法刪除提取請求，但可以關閉它們。您可以刪除您建立的任何分支。

如果您不再需要來源儲存庫或專案，也可以刪除這些資源。如需更多詳細資訊，請參閱 刪除來源儲存庫 及 刪除 Amazon 中的項目 CodeCatalyst。

使用中的來源儲存庫 CodeCatalyst

源存儲庫是您安全地存儲項目的代碼和文件的地方。它還存儲您的源歷史記錄，從第一次提交到最新更改。如果您選擇的藍圖包含來源存放庫，則該存放庫也會包含專案工作流程和通知的組態檔和其他資訊。此配置信息存儲在名為 .code 催化劑的文件夾中。

您可以在中建立來源存放庫， CodeCatalyst 方法是建立具有藍圖的專案，以便在建立專案時建立來源存放庫，或在現有專案中建立來源存放庫。項目用戶將自動看到並能夠使用您為項目創建的存儲庫。您也可以選擇將託管的 Git 儲存庫連結 GitHub 到您的專案。當您這樣做時，您的專案使用者可以在專案的儲存庫清單中檢視和存取該連結的儲存庫。

> **Note**
>
> 連結存放庫之前，您必須先安裝裝載它的服務的擴充功能。您無法連結已封存的存放庫。雖然您可以鏈接一個空存儲庫，但是在使用創建默認分支的初始提交初始化它 CodeCatalyst 之前，您不能使用它。如需詳細資訊，請參閱 安裝擴充功能。

預設情況下，來源儲存庫會與 Amazon CodeCatalyst 專案的其他成員共用。您可以為專案建立其他來源儲存庫，或將儲存庫連結至專案。專案的所有成員都可以檢視、新增、編輯和刪除專案來源儲存庫中的檔案和資料夾。

若要快速處理來源儲存庫中的程式碼，您可以建立一個開發環境，以複製指定的儲存庫並分支到其中，您可以在為開發環境選擇的整合式開發環境 (IDE) 中處理程式碼。您可以克隆本地計算機上的源存儲庫，並在本地存儲庫和中的遠程存儲庫之間提取和推送更改 CodeCatalyst。只要 IDE 支援認證管理，您也可以在偏好的 IDE 中設定來源儲存庫的存取權來使用來源儲存庫。

存放庫名稱在 CodeCatalyst 專案中必須是唯一的。

主題

建立來源儲存庫

當您使用 Amazon 中的藍圖建 CodeCatalyst 立專案時 CodeCatalyst，請為您建立來源儲存庫。除了為您建立的工作流程和其他資源的組態資訊外，該來源儲存庫還包含範例程式碼。這是在中開始使用儲存庫的建議方式 CodeCatalyst。您可以選擇為專案建立儲存庫。這些存儲庫將包含一個文件，一個 README.md 文件，您可以隨時編輯或刪除該文件。根據您在建立來源存放庫時的選擇，儲存庫可能也會包含`.gitignore`檔案。

若要建立來源儲存庫

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
5. 在存放庫名稱中，提供存放庫的名稱。在本指南中，我們使用 *`codecatalyst-source-repository`*，但您可以選擇不同的名稱。存放庫名稱在專案中必須是唯一的。如需有關存放庫名稱需求的詳細資訊，請參閱來源儲存庫的配額 CodeCatalyst。
6. (選擇性) 在說明中，新增存放庫的說明，以協助專案中的其他使用者瞭解存放庫的用途。
7. (選擇性) 針對您計劃推送的程式碼類型新增`.gitignore`檔案。
8. 選擇建立。

> **Note**
>
> CodeCatalyst 創建`README.md`文件時將文件添加到存儲庫中。 CodeCatalyst還會在名為 main 的默認分支中為存儲庫創建一個初始提交。您可以編輯或刪除 README.md 檔案，但無法變更或刪除預設分支。

連結來源儲存庫

將來源儲存庫連結至專案時，如果為您的空間安裝了該 CodeCatalyst 擴充功能，您可以加入具有託管儲存庫之服務副檔名的儲存庫。只有具有 Space 管理員角色的使用者才能安裝擴充功能 安裝擴充功能後，您可以連結至設定以供該擴充功能存取的儲存庫。如需詳細資訊，請參閱安裝和解除安裝擴充功能 CodeCatalyst或遵循快速入門：在中使用 GitHub 儲存庫 CodeCatalyst。

> **Note**
>
> 您只能將儲存庫連結至空間中的一個專案。您無法連結已封存的存放庫。雖然您可以鏈接一個空存儲庫，但是在使用創建默認分支的初始提交初始化它 CodeCatalyst 之前，您不能使用它。

若要連結來源儲存庫

1. 導覽至您要連結存放庫的專案。

   > **Note**
   >
   > 具有 Space 管理員角色的使用者必須先為託管存放庫的提供者安裝擴充功能，才能連結存放庫。如需詳細資訊，請參閱 安裝擴充功能。

2. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇新增儲存庫，然後選擇連結儲存庫。
4. 在存放庫提供者中，選擇提供者的名稱。在GitHub 帳戶中，選擇要用於在儲存庫中工作的連結帳戶，然後在存放庫中選擇存放庫的名稱。

   > **Note**
   >
   > 您只能連結使用中的存放庫。您無法連結已封存的存放庫。

5. 檢閱資訊，然後選擇 [連結]。

   > **Note**
   >
   > 您只能將儲存庫連結至空間中的一個專案。

若要取消儲存庫與專案的連結，您必須具有 S pace 管理員角色。如需更多詳細資訊，請參閱 管理 GitHub 儲存庫 CodeCatalyst。

檢視來源儲存庫

您可以在 Amazon 中檢視與專案關聯的來源儲存庫 CodeCatalyst。對於中的來源儲存庫 CodeCatalyst，儲存庫的概觀頁面會提供該儲存區域中資訊和活動的快速概觀，包括：

- 存放庫的描述 (如果有的話)
- 儲存庫中的分支數
- 儲存庫的開啟提取要求數目
- 儲存庫的相關工作流程數
- 預設分支或您選擇的分支中的檔案和資料夾
- 上次提交到顯示分支的標題，作者和日期
- 在降價中呈現的 README.md 文件的內容，如果包含任何 README.md 文件

此頁面也提供儲存區域認可、分支和提取要求的連結，以及開啟、檢視和編輯個別檔案的快速方法。

> **Note**
>
> 您無法在主控台中檢視有關連結儲存庫的此資 CodeCatalyst 訊。若要檢視連結儲存區域的相關資訊，請選擇儲存區域清單中的連結，以便在代管該儲存區域的服務中開啟該儲存區域。

若要導覽至專案的來源儲存庫

1. 導覽至您的專案，然後執行下列其中一項作業：
   - 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。
   - 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。在來源儲存庫中，從清單中選擇存放庫的名稱。您可以在篩選列中輸入部分存放庫名稱來篩選儲存庫清單。
2. 在儲存區域的首頁上，檢視儲存區域的內容以及相關資源的相關資訊，例如提取要求數目和工作流程。依預設，會顯示預設分支的內容。您可以從下拉式清單中選擇不同的分支來變更檢視。

> **Tip**
>
> 您也可以從專案摘要頁面中選擇 [查看專案程式碼]，快速瀏覽至專案的儲存庫。

編輯來源儲存庫的設定

您可以管理儲存庫的設定，包括編輯存放庫的說明、選擇預設分支、建立和管理分支規則，以及建立和管理中提取要求的核准規則 CodeCatalyst。這有助於專案成員瞭解存放庫的用途，並協助您執行團隊使用的最佳作法和程序。

> **Note**
>
> 您無法編輯來源儲存庫的名稱。
> 您無法在中編輯連結儲存庫的名稱、說明或其他資訊 CodeCatalyst。若要修改連結儲存庫的相關資訊，您必須在代管連結存放庫的提供者中編輯該連結儲存庫。如需詳細資訊，請參閱裝載連結存放庫之服務的說明文件。

若要編輯儲存庫的設定

1. 在 CodeCatalyst 主控台中，導覽至包含要編輯其設定之來源儲存庫的專案。
2. 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。從專案的來源儲存庫清單中選擇儲存庫的名稱。
3. 在存放庫的概觀頁面上，選擇 [其他]，然後選擇 [管理設定]。
4. 執行下列其中一項或多項：
   - 編輯存放庫的說明，然後選擇 [儲存]。
   - 若要變更存放庫的預設分支，請在「預設」分支中選擇 「編輯」。如需詳細資訊，請參閱 檢視和變更儲存庫的預設分支。
   - 若要新增、移除或變更哪些專案角色有權在分支中執行某些動作的規則，請在「分支規則」中選擇 「編輯」。如需詳細資訊，請參閱 使用分支規則管理分支允許的操作。
   - 若要新增、移除或變更將提取路由合併至分支的核准規則，請在 [核准規則] 中選擇 [編輯]。如需更多詳細資訊，請參閱 管理合併提取請求與核准規則的需求。

複製來源儲存庫

若要有效地處理來源儲存庫中的多個檔案、分支和認可，請將來源儲存庫複製到您的本機電腦，並使用 Git 用戶端或整合式開發環境 (IDE) 進行變更。將變更提交並推送至來源儲存庫，以便處理問題和提取要求等 CodeCatalyst 功能。您也可以選擇建立開發環境來處理程式碼。建立開發環境會自動將您指定的存放庫和分支複製到開發環境中。

> **Note**
>
> 您無法在 CodeCatalyst 控制台中克隆鏈接的存儲庫或為其創建開發環境。若要在本機複製連結的儲存區域，請選擇儲存區域清單中的連結，以在裝載該儲存區域的服務中開啟該儲存區域，然後複製它。如需詳細資訊，請參閱裝載連結存放庫之服務的說明文件。

若要從來源儲存庫建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇您要在其中處理代碼的源存儲庫。
4. 選擇 [建立開發環境]。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 執行以下任意一項：
   - 選擇「在現有分支中工作」，然後從「現有分支」下拉式功能表中選擇分支。
   - 選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。
7. 選擇性地新增開發環境的名稱或編輯其組態。
8. 選擇建立。

複製來源儲存庫

1. 導航到您的項目。
2. 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。從專案的來源儲存庫清單中選擇儲存庫的名稱。您可以在篩選列中輸入部分存放庫名稱來篩選儲存庫清單。
3.

4. 選擇複製儲存庫。複製存放庫的複製 URL。

> **Note**
>
> 如果您沒有個人存取權杖 (PAT)，請選擇 [建立權杖]。複製令牌並將其保存在安全位置。當 Git 用戶端或整合式開發環境 (IDE) 提示輸入密碼時，您將使用此 PAT。

5. 執行以下任意一項：

   - 若要將存放庫複製到本機電腦，請開啟終端機或命令列，然後在git clone命令後使用 clone URL 執行命令。例如：

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

     當系統提示輸入密碼時，請貼上您先前儲存的 PAT。

     > **Note**
     >
     > 如果您的作業系統提供認證管理，或者您已安裝認證管理系統，則只需提供 PAT 一次。如果沒有，您可能必須為每個 Git 操作提供 PAT。最佳做法是確保您的憑證管理系統安全地儲存您的 PAT。請勿將 PAT 包含為複製 URL 字串的一部分。

   - 若要使用 IDE 複製儲存庫，請遵循 IDE 的說明文件。選擇複製 Git 儲存庫並提供 URL 的選項。當系統提示您輸入密碼時，請提供 PAT。

刪除來源儲存庫

如果不再需要 Amazon CodeCatalyst 專案的來源儲存庫，您可以將其刪除。刪除來源儲存庫也會刪除儲存在儲存庫中的所有專案資訊。如果有任何工作流程依賴於來源儲存庫，則會在刪除儲存庫後從專案工作流程清單中刪除這些工作流程。參照來源儲存庫的問題將不會遭到刪除或變更，但是刪除儲存庫後，任何加入至問題的來源儲存庫連結都會失敗。

> **Important**
>
> 刪除來源儲存庫無法復原。刪除來源儲存庫之後，您將無法再複製它、從中提取資料或將資料推送至該儲存庫。刪除來源儲存庫並不會刪除該儲存庫的任何本機副本 (本機存放庫)。若要刪除本機存放庫，請使用您本機電腦的目錄和檔案管理工具。

> **Note**
>
> 您無法刪除 CodeCatalyst 主控台中的連結存放庫。若要刪除連結的儲存區域，請選擇儲存區域清單中的連結，在託管該儲存區域的服務中開啟該儲存區域，然後將其刪除。如需詳細資訊，請參閱裝載連結存放庫之服務的說明文件。
> 若要從專案中移除連結的儲存庫，請參閱管理 GitHub 儲存庫 CodeCatalyst。

若要刪除來源儲存庫

1. 導覽至包含您要刪除之來源儲存庫的專案。
2. 在專案的摘要頁面上，從清單中選擇所需的存放庫，然後選擇 [檢視存放庫]。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。從專案的來源儲存庫清單中選擇儲存庫的名稱。
3. 在儲存區域的首頁上，選擇 [其他]，然後選擇 [刪除儲存區域]。
4. 檢閱分支、提取要求及相關工作流程資訊，以協助確保您不會刪除仍在使用中或尚未完成工作的存放庫。如果您要繼續，請輸入 [刪除]，然後選擇 [刪除]。

與 Amazon 的分支機構合作 CodeCatalyst

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、儲存專案的特定版本等。您可以設定來源儲存庫中分支的規則，將分支上的某些動作限制為該專案中的特定角色。

無論您如何建立 CodeCatalyst ，Amazon 中的來源儲存庫都有內容和預設分支。連結的儲存庫可能沒有預設分支或內容，但在您初始化它們並建立預設分支 CodeCatalyst 之前無法使用。使用藍圖建立專案時， CodeCatalyst 會為該專案建立包含 README.md 檔案、範例程式碼、工作流程定義和其他資源的來源存放庫。當您在不使用藍圖的情況下建立來源存放庫時，會為您新增一個 README.md 檔案作為第一次認可，並為您建立預設分支。這個默認分支被命名為 main。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。

> **Note**
>
> 您無法刪除預設分支。為來源儲存庫建立的第一個分支是該儲存庫的預設分支。此外，搜尋只會顯示預設分支的結果。您無法在其他分支中搜尋程式碼。

在中建立儲存庫 CodeCatalyst 也會建立第一次提交，該提交會建立包含在其中的 README.md 檔案的預設分支。該默認分支的名稱是 main。這是本指南範例中使用的預設分支名稱。

主題

創建和刪除分支

您可以使用 CodeCatalyst 控制台在 CodeCatalyst 存儲庫中創建和刪除分支。其他使用者下次從儲存庫中提取變更時，就可以看到您建立的分支。當您刪除分支時，該分支的副本會保留在本機電腦上存放庫的複製中，直到使用者提取並同步這些變更為止。

> **Tip**
>
> 您還可以創建分支作為創建開發環境來處理代碼的一部分。如需詳細資訊，請參閱 建立開發環境。

您也可以使用 Git 來創建和刪除分支。如需詳細資訊，請參閱 用於分支的常用 Git 命令。

要創建一個分支（控制台）

1. 在主 CodeCatalyst 控台中，導覽至來源儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇要在其中創建分支的存儲庫。
4. 在儲存庫的概觀頁面上，選擇 [其他]，然後選擇 [建立分支]。
5. 輸入分支的名稱。
6. 選擇要從中建立分支的分支，然後選擇 [建立]。

刪除分支（控制台）

1. 導覽至儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要刪除分支的存放庫。
3. 在存放庫的概觀頁面上，選擇分支名稱旁邊的下拉式選取器，然後選擇 [檢視全部]。
4. 選擇您要刪除的分支，然後選擇 [刪除分支]。

   > **Note**
   >
   > 您無法刪除存放庫的預設分支。
5. 出現確認對話方塊。它會顯示儲存庫、開啟的提取要求數目，以及與分支相關聯的工作流程數目。
6. 若要確認刪除分支，請在文字方塊中鍵入 delete，然後選擇 [刪除]。

檢視和變更儲存庫的預設分支

您可以在 Amazon 的來源儲存庫中指定要使用哪個分支做為預設分支 CodeCatalyst。中的所有源存儲庫都 CodeCatalyst 具有內容和默認分支，無論您如何創建它們。如果您使用藍圖建立專案，則為該專案建立的來源存放庫中的預設分支會命名為 main。預設分支的內容會自動顯示在該存放庫的概觀頁面上。

與源存儲庫中的所有其他分支相同，默認分支的處理方式略有不同。它的名稱旁邊有一個特殊的標籤，默認。預設分支是當使用者使用 Git 用戶端將存放庫複製到本機電腦時，在本機儲存庫 (Repos) 中用作基底或預設分支的分支。它也是建立用於儲存工作流程 YAML 檔案的工作流程，以及儲存問題資訊時所使用的預設值。在中使用搜尋時 CodeCatalyst，只會搜尋存放庫的預設分支。由於預設分支是專案許多層面的基礎，因此如果將分支指定為預設分支，您就無法刪除該分支。但是，您可以選擇使用不同的分支作為默認分支。如果這樣做，任何套用至先前預設分支的分支規則都會自動套用至您指定為預設分支的分支。

> **Note**
>
> 您必須具有「專案管理員」角色，才能變更 CodeCatalyst 專案中來源儲存庫的預設分支。這不適用於連結的儲存庫。

若要檢視和變更存放庫的預設分支

1. 導覽至儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視設定的存放庫，包括預設分支。
3. 在存放庫的概觀頁面上，選擇 [其他]，然後選擇 [管理設定]。
4. 在「預設」(Default) 分支中，會顯示指定為預設分支的分支名稱，並在名稱旁邊顯示一個名為「預設」的標籤。此相同的標籤會出現在「分支」清單中的分支名稱旁邊。
5. 若要變更預設分支，請選擇 「編輯」。

   > **Note**
   >
   > 您必須在專案中具有專案管理員角色，才能變更預設分支。

6. 從下拉式清單中選擇要建立預設分支的分支名稱，然後選擇 [儲存]。

使用分支規則管理分支允許的操作

當您建立分支時，該分支會根據該角色的權限允許某些動作。您可以透過配置分支規則來變更特定分支允許的動作。分支規則是根據使用者在專案中擁有的角色而定。您可以選擇將某些預先定義的動作限制為專案中具有特定角色的使用者，例如將提交推送至分支。這可協助您保護專案中的特定分支，方法是限制允許哪些角色執行特定動作。例如，如果您將分支規則設定為只允許具有專案管理員角色的使用者合併或推送至該分支，則專案中具有其他角色的使用者將無法變更該分支中的程式碼。

您應該仔細考慮為分支創建規則的所有含義。例如，如果您選擇將推送到分支限制為具有 Project 管理員角色的使用者，具有 Contributor 角色的使用者將無法在該分支中建立或編輯工作流程，因為工作流程 YAML 儲存在該分支中，而且這些使用者無法認可和推送變更至 YAML。最佳做法是在建立任何分支規則之後進行測試，以確保它們不會產生任何您不想要的影響。您也可以將分支規則與提取請求的核准規則搭配使用。如需詳細資訊，請參閱 管理合併提取請求與核准規則的需求。

> **Note**
>
> 您必須具有專案管理員角色，才能管理 CodeCatalyst 專案中來源儲存庫的分支規則。您無法為連結的儲存庫建立分支規則。

您只能建立比角色預設權限更嚴格的分支規則。您無法建立比使用者在專案中所允許的角色更寬鬆的分支規則。例如，您無法建立允許具有「審核者」角色的使用者推送至分支的分支規則。

套用至來源儲存庫預設分支的分支規則會與套用至其他分支的分支規則稍有不同。套用至預設分支的任何規則都會自動套用至您指定為預設分支的任何分支。先前設定為預設分支的分支仍會保留套用規則，除了它將不再具有防止刪除的保護。該保護僅適用於當前默認分支。

分支規則有兩種狀態：標準和自訂。「標準」表示分支上允許的動作是與使用者針對分支動作所擁有的角色權限相符 CodeCatalyst 的動作。若要深入瞭解哪些角色具有哪些權限，請參閱在 Amazon 中使用角色 CodeCatalyst。「自訂」(Custom) 表示一或多個分支動作具有允許執行該動作的特定角色清單，這些動作與專案中使用者 Roe 授予的預設權限不同。

Note

如果您建立分支規則來限制分支的一或多個動作，則「刪除分支」(Delete the branch) 動作會自動設定為只允許具有「專案管理員」角色的使用者刪除該分支。

下表列出了允許在分支上執行這些動作的角色的動作和預設設定。

分支動作和角色

分支動作	未套用分支規則時允許執行此動作的角色
合併到分支（這包括將拉取請求合併到分支）	項目管理員，貢獻者
推送到分支機構	項目管理員，貢獻者
刪除分支	項目管理員，貢獻者
刪除分支（默認分支）	不允許

您無法刪除分支規則，但您可以更新它們，以允許所有角色的動作在分支上執行此動作，從而有效地移除規則。

> **Note**
>
> 您必須具有專案管理員角色，才能設定 CodeCatalyst 專案中來源儲存庫的分支規則。這不適用於連結的儲存庫。連結的儲存庫不支援中的分支規則 CodeCatalyst。

若要檢視和編輯存放庫的分支規則

1. 導覽至儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視分支規則的存放庫。
3. 在存放庫的概觀頁面上，選擇「分支」。
4. 在「分支規則」欄中，檢視存放庫每個分支的規則狀態。「標準」表示分支動作的規則是在來源存放庫中建立的任何分支的預設規則，並符合授與專案中這些角色的權限。「自訂」表示一或多個分支動作具有將該分支允許的一或多個動作限制為不同角色集的規則。

   若要檢視分支規則的細節，請選擇您要檢閱的分支旁邊的「標準」或「自訂」一詞。
5. 若要建立或變更分支規則，請選擇 [管理設定]。在來源儲存區域的設定頁面上，選擇 「分支規則」中的「編輯」。
6. 在「分支」中，從下拉式清單中選擇要為其配置規則的分支名稱。針對每個允許的動作類型，從下拉式清單中選擇您要允許執行該動作的角色，然後選擇 [儲存]。

適用於分支的 Git 命令

您可以使用 Git 在您的計算機（本地存儲庫）或開發環境中創建，管理和刪除源存儲庫的克隆中的分支，然後將更改提交並推送到 CodeCatalyst 源存儲庫（遠程存儲庫）。例如：

用於分支的常用 Git 命令

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`
將遠程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分支。在您提交並推送更改之前，此分支不會出現在遠程存儲庫中。	`git branch branch-name`
使用指定的分支名稱在本地回購中創建一個分支，然後切換到它。	`git checkout -b 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`
使用本地存儲庫為遠程存儲庫指定的暱稱和指定的分支名稱將分支從本地回購推送到遠程存儲庫。還為本地回購中的分支設置上游跟踪信息。	`git push -u remote-name 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`
使用本地存儲庫對遠程存儲庫的指定暱稱和指定的分支名稱刪除遠程存儲庫中的分支。(注意冒號 (:) 的使用方式。) 或者，指定`--delete`為指令的一部分。	`git push remote-name :branch-name` `git push remote-name --delete branch-name`

如需詳細資訊，請參閱您的 Git 文件。

查看分支和詳細信息

您可以在 Amazon CodeCatalyst 主控台中檢視有關 Amazon 遠端分支的資訊 CodeCatalyst，包括特定分支的檔案、資料夾和最近提交的詳細資訊。您也可以使用 Git 命令和本機作業系統來檢視遠端和本機分支的這項資訊。

若要檢視分支 (主控台)

1. 在 CodeCatalyst 控制台中，導航到包含要查看分支的源存儲庫的項目。選擇「程式碼」，選擇「來源儲存庫」，然後選擇來源儲存庫。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視分支的儲存庫。
3. 此時會顯示儲存庫的預設分支。您可以看到分支中的文件和文件夾的列表，有關最近提交的信息以及 README.md 文件的內容（如果它存在於分支中）。若要檢視不同分支的資訊，請從存放庫分支的下拉式清單中選擇該分支。
4. 若要檢視某個儲存庫的所有分支，請選擇檢視全部。「分支」頁面會顯示每個分支的名稱、最近的認可和規則的相關資訊。

如需如何使用 Git 和作業系統來檢視分支和詳細資訊的詳細資訊，請參閱分支的一般 Git 命令、Git 文件和作業系統說明文件。

在 Amazon 中使用文件 CodeCatalyst

在 Amazon 中 CodeCatalyst，檔案是一種版本控制且獨立的資訊，可供您和儲存檔案的來源儲存庫和分支的其他使用者使用。您可以使用目錄結構來組織儲存庫檔案。 CodeCatalyst自動跟踪文件的每個提交更改。您可以將不同版本的檔案儲存在不同的儲存庫分支中。

若要在來源儲存庫中新增或編輯多個檔案，您可以使用 Git 用戶端、開發環境或整合式開發環境 (IDE)。若要新增或編輯單一檔案，您可以使用主 CodeCatalyst 控台。

主題

- 建立或新增檔案
- 檢視檔案
- 編輯檔案
- 重新命名或刪除檔案

建立或新增檔案

若要建立檔案並將其新增至來源儲存庫，您可以使用 Amazon CodeCatalyst 主控台、開發環境、連線的整合式開發環境 (IDE) 或 Git 用戶端。 CodeCatalyst 控制台包括用於創建文件的代碼編輯器。此編

輯器是在儲存庫分支中建立或編輯簡單檔案 (例如 Readme.md 檔案) 的便捷方式。處理多個檔案時，請考慮建立開發環境。

若要從來源儲存庫建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇您要在其中處理代碼的源存儲庫。
4. 選擇 [建立開發環境]。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 執行以下任意一項：
   - 選擇「在現有分支中工作」，然後從「現有分支」下拉式功能表中選擇分支。
   - 選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。
7. 選擇性地新增開發環境的名稱或編輯其組態。
8. 選擇建立。

在 CodeCatalyst 主控台中建立檔案

1. 導覽至您要在其中建立檔案的專案。如需如何導覽至存放庫的詳細資訊，請參閱檢視來源儲存庫。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要在其中建立檔案的存放庫。
3. (選擇性) 如果您要在與預設分支不同的分支中建立檔案，請選擇您要在其中建立檔案的分支。
4. 選擇 [建立檔案]。
5. 在檔案名稱中輸入檔案的名稱。在編輯器中添加文件的內容。

   > **Tip**
   >
   > 如果您要在分支根目錄的子資料夾或子目錄中建立檔案，請將該結構納入檔案名稱的一部分。

   如果您滿意所做的變更，請選擇「提交」。

6. 在 [檔案名稱] 中，檢閱檔案名稱，並對其進行任何您可能想要的變更。選擇性地從「分支」中的可用分支清單中選擇要在其中建立檔案的分支。在「提交」訊息中，選擇性地輸入您進行此變更的原因簡短但資訊詳盡的說明。這將顯示為提交的基本提交信息，該提交將文件添加到源存儲庫中。
7. 選擇「提交」以確認並將檔案推送至來源儲存庫。

您也可以將檔案複製到本機電腦，並使用 Git 用戶端或連線的整合式開發環境 (IDE) 來推送檔案和變更，將檔案新增至來源儲存庫。

> **Note**
>
> 如果您想要新增 Git 子模組，您必須使用 Git 用戶端或開發環境並執行命git submodule add令。您無法在 CodeCatalyst 主控台中新增或檢視 Git 子模組，或檢視提取要求中 Git 子模組的差異。如需 Git 子模組的詳細資訊，請參閱 Git 文件。

若要使用 Git 用戶端或連線的整合式開發環境 (IDE) 新增檔案

1. 將來源儲存庫複製到本機電腦。如需詳細資訊，請參閱 複製來源儲存庫。
2. 在本地存儲庫中創建文件或將文件複製到本地存儲庫中。
3. 執行下列其中一項動作來建立並推送提交：
   - 如果您使用的是 Git 用戶端，請在終端機或命令列執行git add命令，並指定要新增的檔案名稱。或者，若要新增所有新增或變更的檔案，請執行指git add令，然後執行一個或兩個句點，以指示您是要包含目前目錄層級的所有變更 (單一句點)，還是要包含目前目錄和所有子目錄中的所有變更 (兩個句點)。若要提交變更，請執行git commit -m命令並提供提交訊息。若要將變更推送至中的來源儲存庫 CodeCatalyst，請執行git push。如需 Git 命令的詳細資訊，請參閱您的 Git 文件和適用於分支的 Git 命令.
   - 如果您使用的是開發環境或 IDE，請在 IDE 中建立檔案並新增檔案，然後提交並推送變更。如需詳細資訊，請參閱開發環境 CodeCatalyst或參閱 IDE 文件。

檢視檔案

您可以在 Amazon CodeCatalyst 主控台中檢視來源儲存庫中的檔案。您可以在默認分支和任何其他分支中查看文件。檔案內容可能會因您選擇檢視的分支而有所不同。

在 CodeCatalyst 主控台中檢視檔案

1. 導覽至您要檢視檔案的專案。如需詳細資訊，請參閱 檢視來源儲存庫。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視檔案的儲存庫。
3. 會顯示預設分支的檔案和資料夾清單。檔案會以 paper 張圖示表示，而資料夾則以資料夾圖示表示。
4. 執行下列任何一項：
   - 要查看不同分支中的文件和文件夾，請從分支列表中選擇它。
   - 若要展開資料夾，請從清單中選擇該資料夾。
5. 若要檢視特定檔案的內容，請從清單中選擇該檔案。該文件的內容將顯示在分支中。若要檢視不同分支中的檔案內容，請從分支選取器中選擇您想要的分支。

> **Tip**
>
> 檢視檔案內容時，您可以從 [檢視檔案] 中選擇要檢視的其他檔案。若要編輯檔案，請選擇 [編輯]。

您可以在主控台中檢視多個檔案。您也可以使用 Git 用戶端或整合式開發環境 (IDE) 來檢視已複製到本機電腦的檔案。如需詳細資訊，請參閱 Git 用戶端或 IDE 的文件。

> **Note**
>
> 您無法在 CodeCatalyst 主控台中檢視 Git 子模組。如需 Git 子模組的詳細資訊，請參閱 Git 文件。

編輯檔案

您可以在 Amazon CodeCatalyst 主控台中編輯個別檔案。若要一次編輯多個檔案，請建立開發環境或複製儲存庫，然後使用 Git 用戶端或整合式開發環境 (IDE) 進行變更。如需詳細資訊，請參閱 開發環境 CodeCatalyst 或 複製來源儲存庫 。

在 CodeCatalyst 主控台中編輯檔案

1. 導覽至您要編輯檔案的專案。如需如何導覽至存放庫的詳細資訊，請參閱檢視來源儲存庫。
2. 選擇您要編輯檔案的存放庫。選擇 [檢視分支]，然後選擇您要使用的分支。從該分支中的文件和文件夾列表中選擇文件。

   將顯示檔案的內容。
3. 選擇編輯。
4. 在編輯器中，編輯檔案內容，然後選擇 [提交]。或者，在提交更改中，在提交消息中添加有關更改的更多信息。如果您滿意所做的變更，請選擇「提交」。

重新命名或刪除檔案

您可以在開發環境、電腦本機或整合式開發環境 (IDE) 中重新命名或刪除檔案。重命名或刪除文件後，提交並將這些更改推送到源存儲庫。您無法在 Amazon CodeCatalyst 主控台中重新命名或刪除檔案。

在 Amazon 中使用拉取請求 CodeCatalyst

提取請求是您和其他專案成員可以檢閱、註解，以及將程式碼變更從一個分支合併到另一個分支的主要方式。您可以使用提取請求協同檢閱程式碼變更，以瞭解次要變更或修正、主要新增功能或發行軟體的新版本。如果您使用問題來追蹤專案的工時，您可以將特定問題連結至提取請求，以協助您追蹤提取要求中的程式碼變更所解決的問題。當您建立、更新、註解、合併或關閉提取要求時，系統會自動傳送電子郵件給提取要求的作者，以及提取要求的任何必要或選用審核者。

> **Tip**
>
> 您可以配置哪些拉請求事件，您將收到有關電子郵件作為您的個人資料的一部分。如需詳細資訊，請參閱 在 Amazon 中管理通知 CodeCatalyst。

提取要求在來源儲存庫中需要兩個分支：包含您要檢閱之程式碼的來源分支，以及您要合併檢閱程式碼的目標分支。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。

> **Note**
>
> 建立提取要求時，顯示的差異在於來源分支尖端與目標分支尖端之間的差異。建立提取請求之後，顯示的差異會在您選擇的提取請求修訂版本與建立提取請求時作為目的地分支提示的確認之間。有關 Git 中差異和合併基礎的更多信息，請參閱 Git 文檔git-merge-base中的。

當針對特定來源儲存庫和分支建立提取要求時，您可以建立、檢視、檢視和關閉這些要求，做為處理專案的一部分。您不需要檢視來源儲存庫即可檢視和處理提取要求。當您建立提取要求狀態時，會將其設定為「開啟」。提取請求會保持開啟狀態，直到您在 CodeCatalyst 控制台中將其合併為止，該請求會將狀態更改為「已合併」，或將其關閉，從而將狀態更改為「已關閉」。

當您的程式碼經過審核之後，您可以透過下列其中一種方式變更提取要求狀態：

- 在 CodeCatalyst 控制台中合併提取請求。拉取請求的源分支中的代碼將被合併到目標分支中。提取請求狀態將變更為「已合併」。無法將其變更回「開啟」。
- 在本地合併分支並推送更改，然後在 CodeCatalyst 控制台中關閉提取請求。
- 使用 CodeCatalyst 控制台關閉提取請求而不合併。這會將狀態更改為「已關閉」，並且不會將源分支中的代碼合併到目標分支中。

在您建立提取請求之前：

- 提交並將您要查看的代碼更改推送到分支（源分支）。
- 為您的專案設定通知，以便在您建立提取請求時所執行的任何工作流程，通知其他使用者。(此步驟為選擇性步驟，但建議使用。)

主題

建立提取請求

Amazon 的生成式 AI 功能 CodeCatalyst 正在預覽版中，可能會有所變更。它們僅在美國西部 (奧勒岡) 區域提供。生成式 AI 功能的使用方式因層級而異。如需詳細資訊，請參閱 定價。

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。確認並將變更推送至儲存庫之後，您可以建立一個提取要求，將來源分支的內容與目的地分支的內容進行比較。

您可以從特定分支、提取請求頁面或專案概觀在 Amazon CodeCatalyst 主控台中建立提取請求。從特定分支建立提取要求會自動在提取要求建立頁面上提供儲存庫名稱和來源分支。當您創建一個提取請求時，您將自動收到有關提取請求的任何更新以及合併或關閉拉取請求的電子郵件。

> **Note**
>
> 建立提取要求時，顯示的差異在於來源分支尖端與目標分支尖端之間的差異。一旦建立了提取請求，顯示的差異將在您選擇的提取請求的修訂版本與建立提取請求時作為目的地分支提示的確認之間。有關 Git 中差異和合併基礎的更多信息，請參閱 Git 文檔git-merge-base中的。

建立提取請求時，您可以使用「為我寫入描述」功能，讓 Amazon Q 自動建立提取請求中所包含變更的說明。選擇此選項時，Amazon Q 會分析包含程式碼變更的來源分支與您要合併這些變更的目的地分支之間的差異。然後，它會建立這些變更內容的摘要，以及它對這些變更的意圖和影響的最佳解釋。

> **Note**
>
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於「為我撰寫說明」和「建立內容摘要」功能建置在 Amazon Bedrock 上，因此使用者可以充分利用 Amazon Bedrock 中實作的控制項來強制執行人工智慧 (AI) 的安全性、安全性和負責任的使用。

若要建立提取請求

1. 導航到您的項目。
2. 執行以下任意一項：
   - 在瀏覽窗格中，選擇 [程式碼]，選擇 [提取要求]，然後選擇 [建立提取要求]。

- 在儲存區域首頁上，選擇其他，然後選擇建立提取要求。
- 在專案頁面上，選擇 [建立提取要求]。

3. 在源存儲庫中，確保指定的源存儲庫是包含提交代碼的存儲庫。僅當您未從存放庫的主頁面建立提取要求時，此選項才會顯示。
4. 在「目標」分支中，選擇要在檢閱程式碼之後將其合併到的分支。
5. 在原始碼分支中，選擇包含已提交程式碼的分支。
6. 在提取請求標題中，輸入一個標題，以幫助其他用戶了解需要審查的內容以及原因。
7. (選擇性) 在提取要求說明中，提供問題連結或變更說明等資訊。

> **Tip**
>
> 您可以選擇 [為我撰寫說明]， CodeCatalyst 自動產生提取要求中所包含變更的描述。您可以在將自動產生的描述新增至提取請求後，對其進行變更。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

8. (選擇性) 在問題中，選擇連結問題，然後從清單中選擇問題或輸入其 ID。若要取消連結問題，請選擇「取消連結」圖示。
9. (選擇性) 在 [必要的複查者] 中，選擇 [新增必要的審核者]。從專案成員清單中選擇要新增的專案成員。必要的審核者必須先核准變更，才能將提取請求合併到目的地分支。

> **Note**
>
> 您無法同時將審核者新增為必要審核者和選擇性複查者。您無法將自己新增為審核者。

10. (選擇性) 在 [選擇性複查者] 中，選擇 [新增選用複查者]。從專案成員清單中選擇要新增的專案成員。選用審核者不必核准變更作為需求，才能將提取請求合併到目的地分支。
11. 查看分支之間的差異。提取請求中顯示的差異在於源分支中的修訂版本和合併基礎之間的更改，這是在提取請求創建時目的地分支的頭部提交。如果沒有顯示任何更改，則分支可能相同，或者您可能已經為源和目標選擇了相同的分支。
12. 當您滿意提取要求包含您要檢閱的程式碼和變更時，請選擇 [建立]。

> **Note**
>
> 建立提取請求之後，您可以新增註解。註解可以新增至提取請求或檔案中的個別行，以及整個提取請求。您可以使用 @符號後面接到檔案名稱來新增資源連結，例如檔案。

若要從分支建立提取要求

1. 瀏覽至您要在其中建立提取請求的專案。
2. 在瀏覽窗格中，選擇 [來源存放庫]，然後選擇包含要檢閱程式碼變更之分支的儲存庫。
3. 選擇預設分支名稱旁邊的下拉箭頭，然後從清單中選擇所需的分支。若要檢視某個儲存庫的所有分支，請選擇檢視全部。
4. 選擇「其他」，然後選擇「建立提取請求」。
5. 系統會為您預先選取存放庫和來源分支。在目標分支中，選擇一旦審核代碼，您將在其中合併代碼的分支。在提取請求標題中，輸入一個標題，以幫助其他項目用戶了解必須審查的內容以及原因。選擇性地在提取請求說明中提供更多資訊，例如貼上中相關問題的連結 CodeCatalyst，或新增您所做變更的描述。

   > **Note**
   >
   > 如果提取要求的目的地分支符合工作流程中指定的其中一個分支，則設定為針對提取要求建立事件執行的工作流程將會在建立提取要求之後執行。

6. 查看分支之間的差異。如果未顯示任何變更，則分支可能相同，或者您可能已經為來源和目標選擇了相同的分支。
7. (選擇性) 在問題中，選擇連結問題，然後從清單中選擇問題或輸入其 ID。若要取消連結問題，請選擇「取消連結」圖示。
8. (選擇性) 在 [必要的複查者] 中，選擇 [新增必要的審核者]。從專案成員清單中選擇要新增的專案成員。必要的審核者必須先核准變更，才能將提取請求合併到目的地分支。

   > **Note**
   >
   > 您無法同時新增必要和選用的檢閱者。您無法將自己新增為審核者。

9. (選擇性) 在 [選擇性複查者] 中，選擇 [新增選用複查者]。從專案成員清單中選擇要新增的專案成員。在提取請求合併到目的地分支之前，選用審核者不必核准變更。
10. 如果您滿意提取請求包含您要複查的變更，並包含必要的複查者，請選擇「建立」。

如果您將任何工作流程設定為在與提取要求中的目的地分支相符的位置執行，則會在建立提取要求之後，在「提取要求詳細資訊」區域的「概觀」中看到這些工作流程執行的相關資訊。如需更多詳細資訊，請參閱 新增推送、拉取或排程觸發器。

檢視提取要求

Amazon 的生成式 AI 功能 CodeCatalyst 正在預覽版中，可能會有所變更。它們僅在美國西部 (奧勒岡) 區域提供。生成式 AI 功能的使用方式因層級而異。如需詳細資訊，請參閱 定價。

您可以在 Amazon CodeCatalyst 主控台中檢視專案的提取請求。專案摘要頁面會顯示專案的所有開啟提取要求。若要檢視所有提取要求 (不論狀態為何)，請瀏覽至專案的提取要求頁面。檢視提取請求時，您可以選擇對為您建立的提取要求變更留下的所有註解摘要。

> **Note**
>
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於「為我撰寫說明」和「建立內容摘要」功能建置在 Amazon Bedrock 上，因此使用者可以充分利用 Amazon Bedrock 中實作的控制項來強制執行人工智慧 (AI) 的安全性、安全性和負責任的使用。

若要檢視開啟的提取要求

1. 導覽至您要檢視提取請求的專案。
2. 在專案頁面上，會顯示開啟的提取要求，包括建立提取要求的人員、包含提取要求分支的儲存庫，以及提取要求建立日期的相關資訊。您可以按源存儲庫過濾打開的提取請求視圖。
3. 若要檢視所有提取要求，請選擇檢視全部。您可以使用選擇器在選項之間進行選擇。例如，若要檢視所有提取請求，請選擇 [任何狀態] 和 [任何作者]。

   或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [提取要求]，然後使用選取器來精簡您的檢視。
4. 在提取請求頁面上，您可以按 ID、標題、狀態等來排序提取請求。若要自訂提取要求頁面上顯示的資訊和資訊量，請選擇齒輪圖示。
5. 若要檢視特定的提取請求，請從清單中選擇它。
6. 若要檢視與此提取請求相關聯的工作流程執行狀態，請選擇「概要」，然後複查「工作流程」執行下拉取請求之提取請求的「提取請求明細」區域中的資訊。

   如果工作流程配置了提取請求建立或修訂事件，且工作流程中的目的地分支需求符合提取請求中指定的目的地分支，則會執行工作流程。如需詳細資訊，請參閱 新增推送、拉取或排程觸發器。
7. 若要檢視連結的問題 (若有的話)，請選擇概觀，然後複查「問題」下「提取請求詳細資料」中的資訊 如果您要檢視連結的問題，請從清單中選擇其 ID。

8. (選擇性) 若要建立提取要求修訂版本中程式碼變更所留下的註解摘要，請選擇 [建立內容摘要]。摘要不會包含對整體提取要求留下的任何註解。

> **Note**
>
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

9. 若要檢視提取要求中的程式碼變更，請選擇 [變更]。您可以在「檔案變更」中快速檢視提取要求中有多少檔案有變更，以及提取請求中的哪些檔案有其註解。資料夾旁邊顯示的註解數目表示該資料夾中檔案的註解數目。展開資料夾以檢視資料夾中每個檔案的注釋數量。您也可以檢視在特定程式碼行上留下的任何註解。

> **Note**
>
> 並非提取要求中的所有變更都可以顯示在主控台中。例如，您無法在主控台中檢視 Git 子模組，因此無法檢視提取要求中子模組的差異。某些差異可能太大而無法顯示。如需詳細資訊，請參閱 來源儲存庫的配額 CodeCatalyst 及 檢視檔案。

10. 若要檢視此提取請求的品質報表，請選擇「報表」。

> **Note**
>
> 工作流程必須設定為產生報表，以便它們顯示在您的提取請求中。如需更多詳細資訊，請參閱 使用工作流程測試 CodeCatalyst。

管理合併提取請求與核准規則的需求

當您建立提取請求時，您可以選擇將必要或選用的審核者新增至該個別提取請求。不過，您也可以建立合併至特定目的地分支時，所有提取請求都必須符合的需求。這些需求稱為核准規則。核准規則是針對存放庫中的分支配置的。當您建立目的地分支已針對其設定核准規則的提取請求時，除了從任何必要複查者核准之外，還必須符合該規則的需求，才能將提取請求合併至該分支。建立核准規則可協助您維護合併至分支 (例如預設分支) 的品質標準。

套用至來源儲存庫預設分支的核准規則會與套用至其他分支的核准規則稍有不同。套用至預設分支的任何規則都會自動套用至您指定為預設分支的任何分支。先前設定為預設分支的分支仍會保留套用規則。

建立核准規則時，您應該考慮目前和 future 專案使用者將如何符合該規則。例如，如果您的專案中有六位使用者，而您建立的核准規則需要五次核准才能合併至目的地分支，則您已經有效地建立了規則，要求除了建立提取請求的人員以外的所有人都必須核准該提取請求，才能合併該提取請求。

> **Note**
>
> 您必須具有專案管理員角色，才能在 CodeCatalyst 專案中建立及管理核准規則。您無法為連結的儲存庫建立核准規則。

您無法刪除核准規則，但可以將其更新為不需要核准，這樣可有效地移除規則。

若要檢視及編輯提取要求之目的地分支的核准規則

1. 導覽至儲存庫所在的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視核准規則的儲存區域。
3. 在存放庫的概觀頁面上，選擇「分支」。
4. 在「核准規則」欄中，選擇「檢視」以查看儲存庫每個分支的任何規則狀態。

   在核准次數下限中，該數字對應於將提取請求合併到該分支之前所需的核准數目。
5. 若要建立或變更核准規則，請選擇 [管理設定]。在來源儲存區域的設定頁面上，選擇 [核准規則] 中的 [編輯]。

   > **Note**
   >
   > 您必須具有專案管理員角色才能編輯核准規則。
6. 在「分支」中，從下拉式清單中選擇要為其配置核准規則的分支名稱。在核准次數下限中，輸入數字，然後選擇 [儲存]。

檢閱提取要求

Amazon 的生成式 AI 功能 CodeCatalyst 正在預覽版中，可能會有所變更。它們僅在美國西部 (奧勒岡) 區域提供。生成式 AI 功能的使用方式因層級而異。如需詳細資訊，請參閱 定價。

您可以使用 Amazon CodeCatalyst 主控台協同審查提取請求中包含的變更並加上註解。您可以在來源分支和目的地分支之間的差異，或者提取請求修訂之間的差異，將註解新增至個別程式碼行。您可以選擇建立提取要求中程式碼變更留下的註解摘要，以協助您快速瞭解其他使用者留下的意見反應。您也可以選擇建立開發環境來處理程式碼。

> **Note**
>
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於「為我撰寫說明」和「建立內容摘要」功能建置在 Amazon Bedrock 上，因此使用者可以充分利用 Amazon Bedrock 中實作的控制項來強制執行人工智慧 (AI) 的安全性、安全性和負責任的使用。

> **Tip**
>
> 您可以配置哪些拉請求事件，您將收到有關電子郵件作為您的個人資料的一部分。如需詳細資訊，請參閱 在 Amazon 中管理通知 CodeCatalyst。

提取要求會顯示提取要求的修訂版本與您建立提取要求時目的地分支提示的提交之間的差異。這就是所謂的合併基礎。有關 Git 中差異和合併基礎的更多信息，請參閱 Git 文檔git-merge-base中的。

> **Tip**
>
> 在控制台中工作時，特別是如果您已經打開了一段時間的提取請求，請考慮重新整理瀏覽器，以確保您在開始審核提取請求之前擁有最新版本可用於提取請求。

在 CodeCatalyst 主控台中檢閱提取要求

1. 導航到您的項目。
2. 執行下列其中一項作業，以瀏覽至提取要求：
   - 如果提取請求列在項目頁面上，請從列表中選擇它。
   - 如果提取請求未列在專案頁面上，請選擇 [檢視全部]。使用篩選器並排序來尋找提取請求，然後從清單中選擇。
   - 在功能窗格中，選擇 [程式碼]，然後選擇 [提取要求]。
3. 從清單中選擇您要檢閱的提取請求。您可以在篩選列中輸入提取要求的一部分名稱來篩選提取要求清單。

4. 在概觀中，您可以查看提取請求的名稱和標題。您可以創建和查看提取請求本身留下的註釋。您也可以檢視提取請求的詳細資訊，包括工作流程執行、連結問題、檢閱者、提取請求的作者，以及可行合併策略的相關資訊。

> **Note**
>
> 在特定程式碼行上留下的註解會出現在變更中。

5. (選擇性) 若要新增套用至整個提取要求的註解，請展開提取要求的註解，然後選擇 [建立註解]。
6. (選擇性) 若要檢視此提取請求修訂版本變更所留下的所有註解摘要，請選擇「建立註解摘要」。

> **Note**
>
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

7. 在變更中，您可以看到目的地分支與提取請求的最新修訂版之間的差異。如果有多個修訂版，您可以變更要比較哪些修訂版本，以及它們之間的差異。如需修訂的詳細資訊，請參閱修訂。

> **Tip**
>
> 您可以在「檔案變更」中快速檢視提取要求中有多少檔案有變更，以及提取請求中的哪些檔案有其註解。資料夾旁邊顯示的註解數目表示該資料夾中檔案的註解數目。展開資料夾以檢視資料夾中每個檔案的注釋數量。

8. 若要變更差異的顯示方式，請選擇「統一」和「分割」。
9. 若要將註解新增至提取要求的行，請前往您要加上註解的行。選擇該行顯示的註解圖示，輸入註解，然後選擇 [儲存]。
10. 若要檢視提取請求中修訂之間的變更，或在其來源與目的地分支之間檢視變更，請從比較中的可用選項中選擇。修訂中的行的註解會保留在這些修訂中。
11. 如果您已將工作流程設定為針對提取要求觸發程式產生程式碼涵蓋範圍報告，則可以在相關提取要求中檢視明細行與分支涵蓋範圍的發現項目。若要隱藏程式碼涵蓋範圍發現項目，請選擇隱藏程式如需詳細資訊，請參閱 代碼覆蓋率報告。
12. 如果您想要變更提取要求的程式碼，您可以從提取要求建立開發環境。選擇 [建立開發環境]。選擇性地新增開發環境的名稱或編輯其組態，然後選擇 [建立]。
13. 在「報表」中，您可以查看此提取請求中的質量報表。如果有多個修訂版，您可以變更要比較哪些修訂版本，以及它們之間的差異。您可以按名稱、狀態、工作流程、操作和類型來過濾報表。

> **Note**
>
> 工作流程必須設定為產生報表，以便它們顯示在您的提取請求中。如需詳細資訊，請參閱配置報告。

14. 若要檢視特定報告，請從清單中選擇該報告。如需詳細資訊，請參閱 使用工作流程測試 CodeCatalyst。
15. 如果您被列為此提取請求的複查者，並且想要核准變更，請確定您正在檢視最新的版次，然後選擇「核准」。

> **Note**
>
> 所有必要的審核者都必須先核准提取請求，才能合併提取請求。

更新提取請求

您可以更新提取請求，讓其他專案成員更容易檢閱程式碼。您可以更新提取請求以變更其審核者、問題連結、提取請求的標題或描述。例如，您可能想要變更提取請求的必要審核者，以移除休假的人員，並新增其他人。您也可以透過將提交推送至開啟提取要求的來源分支，藉此更新提取要求並進一步變更程式碼。每次推送至來源儲存庫中提取要求的 CodeCatalyst 來源分支，都會建立一個修訂。專案成員可以檢視提取請求中修訂之間的差異。

若要更新提取要求的複查者

1. 導覽至您要更新提取請求複查者的專案。
2. 在專案頁面的「開啟提取請求」下，選擇您要更新審核者的提取請求。或者，在功能窗格中，選擇 [程式碼]、選擇 [提取要求]，然後選擇您要更新的提取要求。
3. (選擇性) 在「概觀」的「提取請求詳細資料」區域中，選擇加號以新增必要或選用的審核者。選擇複查者旁邊的 X，將其移除為選擇性或必要的複查者。
4. (選擇性) 在「概觀」的「提取請求詳細資料」區域中，選擇「連結問題」，將問題連結至提取請求，然後從清單中選擇問題或輸入其 ID。若要取消連結問題，請在您要取消連結的問題旁選擇 [取消連結] 圖示。

若要更新提取要求的來源分支中的檔案和程式碼

1. 要更新多個文件，請創建一個開發環境，或克隆存儲庫及其源分支，並使用 Git 客戶端或集成開發環境（IDE）對源分支中的文件進行更改。將變更提交並推送至來源儲存庫中的 CodeCatalyst 來源分支，以使用變更自動更新提取要求。如需詳細資訊，請參閱 複製來源儲存庫 及 在 Amazon 中處理提交 CodeCatalyst。
2. 要更新源分支中的單個文件，您可以像對多個文件一樣使用 Git 客戶端或 IDE。您也可以直接在CodeCatalyst主控台中編輯它。如需詳細資訊，請參閱 編輯檔案。

更新提取請求的標題和描述

1. 導覽至您要更新提取請求標題或說明的專案。
2. 專案頁面會顯示開啟的提取要求，包括建立提取要求的人員、包含提取要求分支的儲存庫，以及何時建立提取要求的資訊。您可以按源存儲庫過濾打開的提取請求視圖。從清單中選擇您要變更的提取要求。
3. 若要檢視所有提取要求，請選擇檢視全部。或者，在功能窗格中，選擇 [程式碼]，然後選擇 [提取要求]。使用篩選方塊或排序函數來尋找您要變更的提取要求，然後加以選擇。
4. 在「概觀」中選擇 「編輯」。
5. 變更標題或說明，然後選擇 [儲存]。

合併提取請求

在您的程式碼經過審核並且所有必要的審核者都已核准之後，您可以使用支援的合併策略 (例如快轉) 在 CodeCatalyst 主控台中合併提取請求。並非 CodeCatalyst 控制台中支援的所有合併策略都可作為所有提取請求的選擇使用。 CodeCatalyst 評估合併，只允許您在控制台中可用並能夠將源分支合併到目標分支的合併策略之間進行選擇。您也可以在本機電腦或開發環境中執行git merge命令，將來源分支合併到目的地分支，將提取要求與您選擇的 Git 合併策略合併。然後，您可以將目標分支中的這些更改推送到中的源存儲庫 CodeCatalyst。

> **Note**
>
> 合併分支並在 Git 中推送更改不會自動關閉提取請求。

如果您具有「專案管理員」角色，您也可以選擇合併尚未符合所有核准與核准規則需求的提取請求。

合併提取請求（控制台）

如果來源和目的地分支之間沒有合併衝突，且所有必要的審核者都已核准提取請求，您可以在 CodeCatalyst 主控台中合併提取請求。如果發生衝突或無法完成合併，則合併按鈕處於非作用中狀態，並顯示「不可合併」標籤。在這種情況下，您必須取得任何必要核准人的核准，必要時在本機解決衝突，並在合併之前推送這些變更。合併提取請求會自動傳送電子郵件給提取請求的建立者以及任何必要或選用的審核者。它不會自動關閉或更改與提取請求鏈接的任何問題的狀態。

> **Tip**
>
> 您可以配置哪些拉請求事件，您將收到有關電子郵件作為您的個人資料的一部分。如需詳細資訊，請參閱 在 Amazon 中管理通知 CodeCatalyst。

若要合併提取請求

1. 瀏覽至您要合併提取請求的專案。
2. 在專案頁面的「開啟提取請求」下，選擇您要合併的提取請求。如果您沒有看到提取請求，請選擇「檢視所有提取請求」，然後從清單中選擇。或者，在瀏覽窗格中，選擇 [程式碼]、選擇 [提取要求]，然後選擇您要合併的提取要求。選擇 Merge (合併)。
3. 從提取請求的可用合併策略中選擇。選擇性地選取或取消選取要在合併提取請求之後刪除來源分支的選項，然後選擇「合併」。

> **Note**
>
> 如果「合併」按鈕處於非作用中狀態，或者您看到「不可合併」標籤，表示所需的複查者尚未核准提取請求，或者無法在主控台中合併提取請求。 CodeCatalyst 尚未核准提取請求的審核者會在「概述」的「提取請求詳細資料」區域中以時鐘圖示表示。如果所有必要的複查者都已核准提取請求，但「合併」按鈕仍處於非作用中狀態，則您可能會發生合併衝突。選擇加底線的 [不可合併] 標籤，即可查看有關為何無法合併提取要求的更多詳細資訊。您可以在開發環境或 CodeCatalyst 控制台中解決目標分支的合併衝突，然後合併提取請求，或者您可以解決衝突並在本地合併，然後將包含合併的提交推送到源分支 CodeCatalyst。如需詳細資訊，請參閱合併一個提取請求（Git）和您的 Git 文件。

覆寫合併需求

如果您具有「專案管理員」角色，則可以選擇合併尚未符合所需核准與核准規則之所有需求的提取請求。這被稱為覆蓋拉取請求的要求。如果無法使用必要的複查者，或者迫切需要將特定的提取請求合併至具有無法快速符合的核准規則的分支，您可以選擇這樣做。

若要合併提取請求

1. 在您要覆寫需求並合併的提取請求中，選擇「合併」按鈕旁邊的下拉式箭頭。選擇「覆寫核准需求」。
2. 在 「覆寫原因」 中，提供合併此提取請求的原因，而不符合核准規則與必要複查者需求的詳細資訊。雖然這是可選的，但強烈建議您這樣做。
3. 選擇性地選擇合併策略，或接受預設策略。您也可以選擇使用更多詳細信息更新自動生成的提交消息。
4. 選取或取消選取此選項，以在合併時刪除來源分支。我們建議您在覆寫合併提取請求的需求時保留來源分支，直到您有機會與其他專案團隊成員檢閱決定為止。
5. 選擇 Merge (合併)。

合併一個提取請求（Git）

Git 支持許多用於合併和管理分支的選項。下列指令是您可以使用的一些選項。如需詳細資訊，請參閱 Git 網站上的可用文件。合併並推送變更後，請手動關閉提取要求。如需詳細資訊，請參閱 關閉提取請求。

用於合併分支的常用 Git 命令

將本地回購中源分支的更改合併到本地回購中的目標分支。	`git checkout destination-branch-name`
	
	`git merge source-branch-name`

將源分支合併到目標分支中，指定快進合併。這將合併分支並將目標分支指針移動到源分支的尖端。	`git checkout destination-branch-name`
	
	`git merge --ff-only source-branch-name`

將源分支合併到目標分支中，指定壁球合併。這將來自源分支的所有提交合併到目標分支中的單個合併提交中。	`git checkout destination-branch-name`
	
	`git merge --squash source-branch-name`

將源分支合併到目標分支中，指定三向合併。這將創建一個合併提交，並將源分支中的單個提交添加到目標分支。	`git checkout destination-branch-name`
	
	`git merge --no-ff source-branch-name`

刪除本地回購中的源分支。這對於在合併到目標分支並將更改推送到源存儲庫之後對本地存儲庫進行清理非常有用。	`git branch -d source-branch-name`

使用本機存放庫指定的遠端存放庫暱稱，刪除遠端儲存庫中的來源分支 (中的來源儲存庫 CodeCatalyst)。(注意冒號 (:) 的使用方式。) 或者，指定`--delete`為指令的一部分。	`git push remote-name :source-branch-name`
	
	`git push remote-name --delete source-branch-name`

關閉提取請求

您可以將提取請求標記為「已關閉」。這不會合併提取請求，但它可以幫助您確定哪些提取請求需要採取動作，以及哪些提取請求不再相關。我們建議您在合併後關閉提取請求。關閉拉取要求會自動傳送電子郵件給提取要求的建立者，以及任何必要或選用的審核者。它不會自動更改與提取請求鏈接的任何問題的狀態。

> **Note**
>
> 關閉提取請求後，您無法重新開啟該請求。

若要關閉提取請求

1. 瀏覽至您要關閉提取請求的專案。
2. 在專案頁面上，會顯示開啟的提取要求。選擇您要關閉的提取請求。

3. 選擇關閉。
4. 複查資訊，然後選擇「關閉提取要求」。

在 Amazon 中處理提交 CodeCatalyst

遞交是儲存庫的內容和內容變更的快照。每次用戶提交並將更改推送到分支時，都會保存該信息。Git 提交信息包括提交作者，提交更改的人，日期和時間以及所做的更改。當您在 Amazon CodeCatalyst 主控台中建立或編輯檔案時，系統會自動包含類似資訊，但作者名稱是您的 CodeCatalyst 使用者名稱。您還可以將 Git 標籤添加到提交中，以幫助您識別特定的提交。

在 Amazon CodeCatalyst，您可以：

- 檢視分支的提交清單。
- 檢視個別提交，包括在提交中所做的變更，與其父項或父項相比。

您也可以檢視檔案和資料夾。如需詳細資訊，請參閱 在 Amazon 中使用文件 CodeCatalyst。

主題

- 查看對分支的提交
- 更改提交的顯示方式（CodeCatalyst控制台）

查看對分支的提交

您可以通過在控制台中查看分支的提交來查看對分支所做的更改歷史記 CodeCatalyst 錄。這有助於您了解誰對分支進行了更改以及何時進行了更改。您還可以查看在特定提交中所做的更改。

您也可以使用 Git 用戶端來檢視提交。如需詳細資訊，請參閱您的 Git 文件。

若要檢視提交 (主控台)

1. 導覽至包含要檢視提交之來源儲存庫的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要檢視分支提交的儲存庫。

3. 此時會顯示儲存庫的預設分支，包括最近提交至分支的相關資訊。選擇提交。或者，選擇 [更多]，然後選擇 [檢視認可]。
4. 若要檢視不同分支的認可，請選擇分支選取器，然後選擇分支的名稱。
5. 要查看有關特定提交的詳細信息，請從提交標題中選擇其標題。此時會顯示提交的詳細資訊，包括其父項認可的相關資訊，以及透過比較父項認可與指定之認可對程式碼所做的變更。

> **Tip**
>
> 如果確認有一個以上的父項，您可以選擇父項確認 ID 旁邊的下拉式圖示，選擇要檢視資訊和顯示變更的父項確認。

更改提交的顯示方式（CodeCatalyst控制台）

您可以變更「提交」檢視中顯示的資訊。您可以選擇隱藏或顯示作者和提交 ID 等欄。

要更改提交的顯示方式（控制台）

1. 導覽至包含要檢視提交之來源儲存庫的專案。
2. 從專案的來源儲存庫清單中選擇儲存庫的名稱。或者，在導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

   選擇您要變更檢視認可方式的儲存庫。
3. 此時會顯示儲存庫的預設分支，包括最近提交至分支的相關資訊。選擇提交。
4. 選擇齒輪圖示。
5. 在「偏好設定」 中，選擇要顯示的提交數目，並選擇是否要顯示有關提交作者、確認日期和提交 ID 的資訊。

> **Note**
>
> 您無法在顯示資訊時隱藏提交標題。

6. 完成變更後，請選擇 [儲存] 以儲存變更，或選擇 [取消] 捨棄變更。

來源儲存庫的配額 CodeCatalyst

下表說明 Amazon 中來源儲存庫的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

資源	資訊
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，且在儲存庫中必須是唯一的。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.
提取要求的註解	一個提取請求上限為 1,000 個。
提交訊息	最多可輸入 1024 個字元。
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： /./ /../

資源	資訊
	// • 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 CodeCatalyst主控台時，任何個別檔案最多可容納 6 MB。
可在 CodeCatalyst 主控台中檢視的檔案大小	使用 CodeCatalyst主控台時，任何個別檔案最多可容納 6 MB。
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。但是，作為最佳實踐，請考慮進行多個較小的提交，而不是一次大型提交。
提交的元數據	提交的合併中繼資料最多 6 MB (例如，作者資訊、日期、父項提交清單和提交訊息的組合)。 **Note** 在單一遞交中，所有檔案的數目或大小總計沒有限制，只要資料不超過 20 MB、個別檔案不超過 6 MB 且單一 Blob 不超過 2 GB 就沒問題。

資源	資訊
可連結至提取要求的 CodeCatalyst 問題數	50
可連結至提取要求的 Jira 問題數目	50
空間中開啟的提取要求數目	最多 1,000 一個 Amazon CodeCatalyst 空間。
空間中的總提取要求數	Amazon CodeCatalyst 空間最多 10,000。
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。
空間中的儲存庫數目	Amazon CodeCatalyst 空間最多 5,000。
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	存放庫名稱在專案中必須是唯一的。它們可以包含長度介於 1 到 100 個字元之間的字母、數字、句點、底線和破折號的任意組合。名稱不區分大小寫。存放庫名稱不能以 .git 結尾，不能包含空格，也不能包含下列任何字元：！ ？ @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
儲存庫大小	儲存庫大小會受到空間整體儲存限制的影響。如需詳細資訊，請參閱 定價和 疑難排解來源儲存庫問題。
提取請求的審核者	一個提取請求最多 100 位審核者總計 (選擇性或必要)。
提取要求的書面摘要	提取要求的書面摘要數目上限取決於您空間的計費層級。如需詳細資訊，請參閱 定價。

開發環境 CodeCatalyst

開發環境是基於雲的開發環境。在 Amazon 中 CodeCatalyst，您可以使用開發環境來處理存放在專案原始碼儲存庫中的程式碼。當創建一個開發環境，你有幾個選擇：

- 在中建立專案特定的開發環境， CodeCatalyst 以使用受支援的整合式開發環境 (IDE) 處理程式碼。
- 創建一個空的 Dev 環境，將代碼從源存儲庫克隆到其中，並使用支持的 IDE 處理該代碼。
- 在 IDE 中建立開發環境，並將來源儲存庫複製到開發環境

devfile 是一個開放的標準 YAML 檔案，可將您的開發環境標準化。換句話說，這個文件編寫了開發環境所需的開發工具。因此，您可以快速設定開發環境、在專案之間切換，以及跨團隊成員複寫開發環境設定。開發環境可將您建立和維護本機開發環境所花費的時間降到最低，因為它們使用 devfile 來設定指定專案的程式碼、測試和偵錯所需的所有工具。

開發環境中包含的項目工具和應用程序庫由項目源存儲庫中的 devfile 定義。如果您的源存儲庫中沒有 devfile，則 CodeCatalyst 會自動應用默認的開發文件。這個默認的 devfile 包括最常用的編程語言和框架的工具。如果您的專案是使用藍圖建立的，則會自動由 CodeCatalyst建立 devfile。如需有關開發檔案的詳細資訊，請參閱 https://devfile.io。

建立開發環境之後，只有您可以存取它。在您的開發環境中，您可以在受支援的 IDE 中檢視和處理原始碼儲存庫的程式碼。

根據預設，開發環境設定為具有 2 核心處理器、4 GB 的 RAM 和 16 GB 的持續性儲存空間。如果您具有 Space 管理員權限，則可以變更空間的計費層，以使用不同的開發環境設定選項，以及管理運算和儲存限制。以下是開發環境的一個可能的工作流程：

阿庫阿曼薩是在實施例公司 Agua 一個新的開發人員加入了她的新團隊只是球隊的產品的新版本發布之前。Agua 的團隊需要她立即為該產品的即將到來的版本開發新功能。為了避免廣泛的設置過程，Agua 接受加入團隊 CodeCatalyst 項目的邀請。然後，她為自己指派相關問題，並從小組原始碼儲存庫的現有分支建立開發環境。Agua 的開發環境會在她選擇的 IDE 中開啟原始碼儲存庫的程式碼。打開的 IDE 實例連接到她的開發環境，其中包含一個已自動識別和應用的 devfile。devfile 會指定所有她需要開始使用的工具。Agua 撰寫新產品功能的程式碼、提交程式碼變更、將變更推送至現有分支，然後在工作完成時刪除她的開發環境。Agua 已為新團隊的原始程式碼儲存庫貢獻程式碼，而不需要冗長的設定程序。

建立開發環境

您可以透過多種方式建立開發環境：

- CodeCatalyst 透過概觀、開發環境或 CodeCatalyst 來源存放庫頁面的來源儲存庫或連結來源儲存庫，在中建立開發環境
- 在其中 CodeCatalyst 建立未從開發環境頁面連接到來源儲存庫的空白 Dev 環境
- 在您選擇的 IDE 中建立開發環境，並將任何來源儲存庫複製到開發環境

您可以為存放庫的每個分支建立一個開發環境。一個專案可以有多個儲存庫。您建立的開發環境只能使用您的 CodeCatalyst 帳戶進行管理，但是您可以開啟開發環境並在其中使用任何支援的 IDE。您必須 AWS 工具組 安裝，才能在 IDE 中使用開發環境。如需詳細資訊，請參閱 支援開發環境的整合式開發環境。根據預設，開發環境是使用 2 核心處理器、4 GB RAM 和 16 GB 的持續性儲存裝置來建立。

> **Note**
>
> 如果您建立了與來源儲存庫相關聯的開發環境，則 [資源] 資料行一律會顯示您在建立此開發環境時指定的分支。即使您創建另一個分支，切換到開發環境中的另一個分支或克隆其他存儲庫，這也適用。如果您建立了空白的開發環境，[資源] 資料行將會是空白的。

支援開發環境的整合式開發環境

您可以在下列支援的整合式開發環境 (IDE) 中使用開發環境：

- AWS Cloud9
- JetBrains IDE
  - IntelliJ 理念終極
  - GoLand
  - PyCharm專業
- Visual Studio 程式碼

在中建立開發環境 CodeCatalyst

若要開始在中使用開發環境 CodeCatalyst，請使用您的AWS 產生器 ID 或 SSO 驗證並登入。

從分支建立開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要創建開發環境的項目。
3. 在導覽窗格中，執行下列其中一項作業：
   - 選擇 [概觀]，然後瀏覽至 [我的開發環境] 區段。
   - 選擇 [程式碼]，然後選擇 [開發環境]。
   - 選擇 [程式碼]，選擇 [原始碼儲存庫]，然後選擇您要建立開發環境的儲存庫。
4. 選擇 [建立開發環境]。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 選擇複製儲存庫。
7. 執行以下任意一項：
   a. 選擇要複製的存放庫，選擇在現有分支中工作，然後從現有分支下拉式功能表中選擇分支。

      > **Note**
      > 如果您選擇第三方存放庫，則必須在現有分支中工作。

   b. 選擇要複製的存放庫，選擇在新分支中工作，在「分支名稱」欄位中輸入分支名稱，然後從「建立分支來源」下拉式功能表中選擇要從中建立新分支的分支。

      > **Note**
      > 如果您從 [來源儲存庫] 頁面或特定來源儲存庫建立開發環境，則不需要選擇存放庫。開發環境將從您從「來源儲存庫」頁面選擇的來源儲存庫建立。

8. (選擇性) 在別名-選擇性中，輸入開發環境的別名。
9. (選擇性) 選擇開發環境設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
10. （可選）在 Amazon Virtual Private Cloud (Amazon VPC) 中-可選，從下拉式功能表中選取您要與開發環境建立關聯的 VPC 連線。

    如果為您的空間設置了默認 VPC，則您的開發環境將運行連接到該 VPC。您可以透過關聯不同的 VPC 連線來覆寫此項目。此外，請注意，連接 VPC 的開發環境不支援。 AWS 工具組

> **Note**
>
> 當您使用 VPC 連線建立開發環境時，VPC 內會建立新的網路介面。 CodeCatalyst 使用關聯的 VPC 角色與此介面互動。此外，請確定您的 IPv4 CIDR 區塊未設定為 `172.16.0.0/12` IP 位址範圍。

11. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。

若要建立空白的開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要創建開發環境的項目。
3. 在導覽窗格中，執行下列其中一項作業：
   - 選擇 [概觀]，然後瀏覽至 [我的開發環境] 區段。
   - 選擇 [程式碼]，然後選擇 [開發環境]。
4. 選擇 [建立開發環境]。
5. 從下拉式功能表中選擇支援的 IDE。如需詳細資訊，請參閱支援開發環境的整合式開發環境。
6. 選擇 [建立空白的開發環境]。
7. (選擇性) 在別名-選擇性中，輸入開發環境的別名。
8. (選擇性) 選擇開發環境設定編輯按鈕，以編輯開發環境的運算、儲存或逾時設定。
9. （可選）在 Amazon Virtual Private Cloud (Amazon VPC) 中-可選，從下拉式功能表中選取您要與開發環境建立關聯的 VPC 連線。

   如果為您的空間設置了默認 VPC，則您的開發環境將運行連接到該 VPC。您可以透過關聯不同的 VPC 連線來覆寫此項目。此外，請注意，連接 VPC 的開發環境不支援。 AWS 工具組

> **Note**
>
> 當您使用 VPC 連線建立開發環境時，VPC 內會建立新的網路介面。 CodeCatalyst 使用關聯的 VPC 角色與此介面互動。此外，請確定您的 IPv4 CIDR 區塊未設定為 `172.16.0.0/12` IP 位址範圍。

10. 選擇建立。建立您的開發環境時，開發環境狀態欄會顯示 [開始]，而且建立開發環境後，狀態欄會顯示 [執行中]。

> **Note**
>
> 第一次建立和開啟開發環境可能需要一到兩分鐘的時間。

> **Note**
>
> 在 IDE 中開啟開發環境之後，您可能需要先將目錄變更至來源儲存庫，然後再提交並推送變更至程式碼。

在 IDE 中建立開發環境

您可以使用支援的整合式開發環境 (IDE) 來處理儲存在專案原始碼儲存庫中的程式碼。

- 與 Amazon CodeCatalyst 合作 AWS Cloud9
- VS 代碼 CodeCatalyst 中的 Amazon

  有關身份驗證的說明，請參閱 AWS Toolkit for Visual Studio Code 用戶指南中的 AWS 從開發環境進行身份驗證並連接到以及適用 CodeCatalyst於 Amazon 的身份驗證。
- Amazon CodeCatalyst 在 JetBrains

  如需驗證的指示，請參閱 AWS Toolkit for JetBrains 使用指南 CodeCatalyst中的「驗證和連接 JetBrains 閘道」。

停止開發環境

開發環境的`/projects`目錄會儲存從來源儲存庫中提取的檔案，以及用來設定開發環境的 devfile。該`/home`目錄在創建開發環境時為空，存儲您在使用開發環境時創建的文件。開發環境`/projects`和`/home`目錄中的所有內容都會持續存儲，因此如果您需要切換到另一個開發環境，存儲庫或項目，則可以停止在開發環境中工作。

> **Warning**
>
> 如果任何實例（包括 Web 瀏覽器，遠程 shell 和 IDE）保持連接狀態，則開發環境不會超時。因此，請務必關閉所有連線的執行個體，以避免產生額外費用。

如果開發環境在建立開發環境期間在 [逾時] 欄位中選取的時間長度內閒置，則開發環境將自動停止。您可以在開發環境閒置之前停止它。如果您在建立開發環境時選擇 [無逾時]，則開發環境不會自動停止。相反，它將連續運行。

> **Warning**
>
> 如果您停止與已刪除 VPC 連線相關聯的開發環境，則無法恢復該環境。

從開發環境頁面停止開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到要停止開發環境的項目。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇開發環境。
5. 選擇您要停止的開發環境的選項按鈕。
6. 從「動作」功能表中選擇「停止」。

> **Note**
>
> 運算使用量只會在開發環境執行時計費，但是儲存體使用量會按開發環境存在的整個時間計費。在開發環境不使用時停止計算計費，請停止該環境。

恢復開發環境

開發環境的`/projects`目錄會儲存從來源儲存庫中提取的檔案，以及用來設定開發環境的 devfile。該`/home`目錄在創建開發環境時為空，存儲您在使用開發環境時創建的文件。開發環境`/projects`和`/home`目錄中的所有內容都會持續存儲，因此如果您需要切換到另一個開發環境，存儲庫或項目並在以後在開發環境中繼續工作，則可以停止在開發環境中工作。

如果開發環境在建立開發環境期間在 [逾時] 欄位中選取的時間長度內閒置，則開發環境將自動停止。您必須關閉 AWS Cloud9 瀏覽器標籤，開發環境才會處於閒置狀態。

> **Note**
>
> 即使您刪除了用來建立開發環境的分支，開發環境仍然可以使用並執行。如果您想在刪除分支的開發環境中繼續工作，請創建一個新分支並將更改推送到該分支。

從概觀頁面恢復開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至您要繼續開發環境的專案，然後瀏覽至 [我的開發環境] 區段。
3. 選擇在 (IDE) 中繼續。
   - 針對 JetBrains IDE，請在系統提示 [選擇應用程式以開啟 JetBrains -閘道] 連結時選擇閘道-EAP。 JetBrains選擇開啟連結以在出現提示時進行確認
   - 對於 VS Code IDE，請在系統提示 [選擇應用程式以開啟 VS Code] 連結時選擇 [VS 程式碼]。選擇「開啟連結」以確認。

從來源儲存庫恢復開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您要恢復開發環境的項目。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇來源儲存庫。
5. 選擇包含您要繼續的開發環境的來源存放庫。
6. 選擇分支名稱以查看分支的下拉菜單，然後選擇您的分支。
7. 選擇 [繼續開發環境]。
   - 如果是 JetBrains IDE，請選擇「開啟連結」以在系統提示「允許此網站開啟 JetBrains閘道嗎？」 JetBrains 。
   - 對於 VS 代碼 IDE，選擇打開鏈接以在提示時確認允許此站點使用 Visual Studio 代碼打開 VS 代碼鏈接？ 。

從開發環境頁面恢復開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您要恢復開發環境的項目。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇開發環境。
5. 從 IDE 資料行中，針對開發環境選擇 [在 (IDE) 中繼續執行]。
   - 如果是 JetBrains IDE，請選擇「開啟連結」以在系統提示「允許此網站開啟 JetBrains閘道嗎？」 JetBrains 。
   - 對於 VS 代碼 IDE，選擇打開鏈接以在提示時確認允許此站點使用 Visual Studio 代碼打開 VS 代碼鏈接？ 。

> **Note**
>
> 系統可能需要幾分鐘來恢復開發環境。

編輯開發環境

當您的 IDE 正在執行時，您可以編輯開發環境。如果您編輯計算或閒置逾時，您的開發環境將在您儲存變更後重新啟動。

若要編輯開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至您要編輯開發環境的專案。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇開發環境。
5. 選擇您要編輯的開發環境。
6. 選擇編輯。
7. 對計算或閒置逾時進行您想要的變更。
8. 選擇 Save (儲存)。

刪除開發環境

當您完成處理儲存在開發環境中的內容後，您可以刪除開發環境。建立新的開發環境以處理新內容。如果您刪除您的開發環境，持續性的內容將被永久刪除。在刪除開發環境之前，請確保您提交並將代碼更改推送到開發環境的原始源存儲庫。刪除開發環境後，開發環境的計算和儲存體計費將停止。

刪除開發環境後，可能需要幾分鐘的時間才能更新儲存配額。如果您已達到儲存配額，您將無法在此期間建立新的開發環境。

> **Important**
>
> 刪除開發環境無法復原。刪除開發環境之後，您將無法再復原它。

若要刪除開發環境

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導航到要刪除開發環境的項目。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇開發環境。
5. 選擇您要刪除的開發環境。
6. 選擇刪除。
7. 輸入**delete**以確認刪除開發環境。
8. 選擇刪除。

> **Note**
>
> 刪除空間中的 VPC 連線之前，請務必移除與該 VPC 相關聯的開發環境。
> 即使刪除開發環境，也可能不會刪除 VPC 中的網路介面。請務必視需要清理您的資源。如果刪除 VPC 連線的開發環境時發生錯誤，您必須卸離過時的連線，並在確認未使用過時的連線後將其刪除。

透過 SSH 連線至開發環境

您可以透過 SSH 連線至您的開發環境，不受限制地執行動作，例如連接埠轉送、上傳和下載檔案，以及使用其他 IDE。

> **Note**
>
> 如果您想要在關閉 IDE 索引標籤或視窗之後長時間繼續使用 SSH，請務必為您的開發環境設定較高的逾時，這樣它就不會因為 IDE 中的閒置而停止。

必要條件

- 您需要下列其中一種作業系統：
  - 視窗 10 或更新版本且已啟用開啟 SSH
  - macOS 和 Bash 版本 3 或更高版本
  - Linux`yum`，`dpkg`或`rpm`包管理器和 Bash 版本 3 或更高版本
- 您還需要 2.9.4 或更高 AWS CLI 版本。

若要透過 SSH 連線至開發環境

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導航到要通過 SSH 連接到開發環境的項目。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇開發環境。
5. 選擇您要透過 SSH 連線的執行中開發環境。
6. 選擇 [透過 SSH Connect]，選擇您想要的作業系統，然後執行下列動作：
   - 如果您尚未這樣做，請在指定的終端中粘貼並執行第一個命令。此命令會下載指令碼，並在本機環境中執行下列修改，以便您可以透過 SSH 連線至開發環境：
     - 安裝工作階段管理員外掛程式 AWS CLI
     - 修改您的本機 AWS Config 並新增 CodeCatalyst 設定檔，以便您能夠執行 SSO 登入。如需詳細資訊，請參閱 設定以使用AWS CLI與 CodeCatalyst。
     - 修改您的本機 SSH 設定，並新增透過 SSH 連線至開發環境的必要組態。

  - 在 SSH 用戶端用來連線至您的開發環境的`~/.aws/codecatalyst-dev-env`目錄中新增指令碼。此指令碼會呼叫 CodeCatalyst StartDevEnvironmentSession API，並使用 AWS Systems Manager Session Manager 外掛程式與您的開發環境建立 AWS Systems Manager 工作階段，本機 SSH 用戶端會使用此工作階段來安全地連線至遠端開發環境。
- CodeCatalyst 使用第二個命令使用 AWS SSO 登錄到 Amazon。此命令要求並擷取認證，以便目`~/.aws/codecatalyst-dev-env`錄中的指令碼可以呼叫 CodeCatalyst StartDevEnvironmentSession API。每當您的認證過期時，都應執行此命令。當您在模態 (ssh<destination>) 中執行最後一個命令時，如果您的認證已過期，或者您尚未按照此步驟中的指示執行 SSO 登入，就會收到錯誤訊息。
- 使用第三個命令，透過 SSH Connect 到指定的開發環境。該命令具有以下結構：

  ```
  ssh codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
  ```

  您也可以使用這個命令來執行 SSH 用戶端允許的其他動作，例如連接埠轉送或上傳和下載檔案：

  - 端口轉發：

    ```
    ssh -L <local-port>:127.0.0.1:<remote-port> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
    ```

  - 將文件上傳到開發環境中的主目錄：

    ```
    scp -O </path-to-local-file> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>:</path-to-remote-file-or-directory>
    ```

設定您的開發環境

devfile 是一種開放式標準，可協助您在整個團隊中自訂開發開發環境。devfile 是編寫您所需開發工具的 YAML 檔案。透過設定 devfile，您可以預先確定所需的專案工具和應用程式庫，然後 Amazon 將它們 CodeCatalyst 安裝到您的開發環境中。devfile 是特定於為其建立的儲存庫，您可以為每個儲存庫建立個別的 devfile。您的開發環境支援命令和事件，並提供預設的通用 devfile 映像檔。

如果您使用空藍圖建立專案，則可以手動建立 devfile。如果您使用不同的藍圖建 CodeCatalyst 立專案，則會自動建立 devfile。開發環境的`/projects`目錄存儲從源存儲庫和 devfile 中提取的文件。當您第一次建立開發環境時，`/home`目錄是空的，會儲存您在使用開發環境時建立的檔案。Dev 環境`/projects`和`/home`目錄中的所有內容都會持續存儲。

> **Note**
>
> 只有在您變更開發檔案或 devfile 元件名稱的名稱時，/home資料夾才會變更。如果您變更 devfile 或 devfile 元件名稱，則會取代/home目錄的內容，而且您先前的/home目錄資料無法復原。

如果您建立一個開發環境，其來源儲存庫不包含其根目錄中的 devfile，或者如果您建立了沒有來源存放庫的開發環境，則預設的通用 devfile 會自動套用至來源儲存庫。相同的默認通用開發文件映像用於所有 IDE。 CodeCatalyst 目前支持開發文件版本 2.0.0。有關開發文件的更多信息，請參閱開發文件模式-版本 2.0.0。

> **Note**
>
> 您只能在您的 devfile 中包含公用容器影像。

請注意，連接 VPC 的開發環境僅支援下列開發檔案映像檔：

- 通用影像
- 私有 Amazon ECR 映像檔 (如果儲存庫與虛擬私人 VPC 位於相同的區域)

主題

編輯開發環境的儲存庫開發檔 CodeCatalyst

若要編輯儲存庫開發檔

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至包含您要編輯 devfile 之來源儲存庫的專案。
3. 在功能窗格中，選擇 [程式碼]。
4. 選擇來源儲存庫。
5. 選擇包含您要編輯之 devfile 的來源儲存庫。
6. 從檔案清單中選擇檔`devfile.yaml`案。
7. 選擇編輯。
8. 編輯開發文件。
9. 選擇「確認」，或建立提取請求，讓團隊成員可以檢閱並核准變更。

> **Note**
>
> 如果您編輯您的 devfile，您必須重新啟動 devfile，變更才會生效。這可以通過運行來完成/`aws/mde/mde start --location devfile.yaml`。如果啟動您的 devfile 時出現問題，它將進入恢復模式。但是，如果您編輯與 VPC 連接的開發環境相關聯的開發文件，則必須重新啟動開發環境，而不是使更改生效。

您可以通過運/`aws/mde/mde status`行查看正在使用哪個開發文件。位置字段具有相對於環境文件/`projects`夾的 devfile 的路徑。

```
{
    "status": "STABLE",
    "location": "devfile.yaml"
}
```

在 IDE 中編輯開發環境的存放庫開發檔

要更改開發環境的配置，您必須編輯 devfile。我們建議您在受支援的 IDE 中編輯 devfile，然後更新您的開發環境，但您也可以從中的來源儲存庫根目錄編輯 devfile。 CodeCatalyst如果您在支援的 IDE 中編輯 devfile，您必須認可並將變更推送至來源儲存庫，或建立提取要求，以便團隊成員可以檢閱並核准 devfile 編輯。

- 編輯開發環境的儲存庫 devfile AWS Cloud9
- 在 VS 代碼中編輯開發環境的存儲庫開發文件
- 編輯開發環境的儲存庫 devfile JetBrains

移動開發環境的存儲庫 devfile

您可以將默認的 devfile 移動`/projects/devfile.yaml`到源代碼存儲庫中。要更新開發文件的位置，請使用以下命令：`/aws/mde/mde start --location` *`repository-name`*`/devfile.yaml`。

恢復模式

如果啟動 devfile 時出現問題，它將進入恢復模式，以便您仍然可以連接到您的環境並修復您的 devfile。在恢復模式下，運行`/aws/mde/mde status`不會包含您的 devfile 的位置。

```
{
    "status": "STABLE"
}
```

您可以檢查日誌中的錯誤`/aws/mde/logs`，修復 devfile，然`/aws/mde/mde start`後再次嘗試運行。

支援的開發檔功能 CodeCatalyst

CodeCatalyst 在 2.0.0 版本上支援下列開發檔案功能。有關開發文件的更多信息，請參閱開發文件模式-版本 2.0.0。

功能	Type
`exec`	Command
`postStart`	事件
`container`	元件
`args`	零組件屬性
`env`	零組件屬性

功能	Type
mountSources	零組件屬性
volumeMounts	零組件屬性

範例：為您的開發環境設定開發檔

下面是一個簡單的開發文件的一個例子。

```
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

在/aws/mde/logs中擷取並儲存 Devfile 啟動、命令和事件記錄檔。要調試 devfile 行為，請使用工作的開發文件啟動您的開發環境並訪問日誌。

開發檔命令

目前， CodeCatalyst 僅支持您的 devfile 中的exec命令。如需詳細資訊，請參閱 DevFile.io 文件中的新增命令。

下列範例說明如何在 devfile 中指定exec命令。

```
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
```

連線到開發環境之後，您可以透過終端機執行已定義的命令。

```
/aws/mde/mde command <command-id>
/aws/mde/mde command executescript
```

對於長時間運行的命令，您可以使用 Streaming 標誌-s來實時輸出命令的執行。

```
/aws/mde/mde -s command <command-id>
```

> **Note**
>
> command-id必須是小寫。

支援的執行參數 CodeCatalyst

CodeCatalyst 在開發文件版本 2.0.0 上支持以下`exec`參數。

- `commandLine`
- `component`
- `id`
- `workingDir`

開發檔案事件

目前， CodeCatalyst 僅支持您的 devfile 中的`postStart`事件。如需詳細資訊，請參閱 DevFile.io 說明文件postStartObject中的。

下面的例子顯示了如何在您的 devfile 中添加`postStart`事件綁定。

```
commands:
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - updateyum
    - executescript
```

啟動後，您的開發環境將按照定義的順序執行指定的`postStart`命令。如果命令失敗，開發環境將繼續運行，並將執行輸出存儲在下的日誌中`/aws/mde/logs`。

開發文件組件

目前，CodeCatalyst 僅支持您的開發文`container`件中的組件。如需詳細資訊，請參閱 DevFile.io 說明文件中的新增元件。

下列範例說明如何將啟動命令加入您的 devfile 中的容器。

```
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      command: ['sleep', 'infinity']
```

> **Note**
>
> 當容器具有短暫的條目命令時，您必須包含`command: ['sleep', 'infinity']`以保持容器運行。

CodeCatalyst 也支援容器元件中的下列屬性：`args`、`env`、`mountSources`、和`volumeMounts`。

通用開發文件圖像

預設的通用映像檔包括最常用的程式設計語言和可用於 IDE 的相關工具。如果未指定影像，請 CodeCatalyst 提供此影像，並包含由維護的工具 CodeCatalyst。若要在新映像發行時保持通知，請參閱透過 SNS 進行通用影像通知。

> **Note**
>
> 該`public.ecr.aws/aws-mde/universal-image:latest`圖像獲取`public.ecr.aws/aws-mde/universal-image:3.0`圖像。

Amazon CodeCatalyst 支持以下開發文件圖像。

影像版本	映像識別符
`Universal image 1.0`	`public.ecr.aws/aws-mde/universal-image:1.0`

影像版本	映像識別符
Universal image 2.0	public.ecr.aws/aws-mde/universal-image:2.0
Universal image 3.0	public.ecr.aws/aws-mde/universal-image:3.0

> **Note**
>
> 如果您使用的是 AWS Cloud9，升級到universal-image:3.0後，自動完成功能不適用於 PHP，Ruby 和 CSS。

主題

透過 SNS 進行通用影像通知

CodeCatalyst 提供通用影像通知服務。您可以使用它來訂閱 Amazon Simple Notification Service (SNS) 主題，該主題會在 CodeCatalyst 通用映像更新發佈時通知您。如需 SNS 主題的詳細資訊，請參閱什麼是 Amazon 簡單通知服務？ 。

每當發行新的通用影像時，我們都會傳送通知給訂閱者；本節說明如何訂閱 CodeCatalyst 通用影像更新。

範例訊息

```
{
    "Type": "Notification",
    "MessageId": "123456789",
    "TopicArn": "arn:aws:sns:us-east-1:1234657890:universal-image-updates",
    "Subject": "New Universal Image Release",
    "Message": {
        "v1": {
```

```
            "Message": "A new version of the Universal Image has been released. You are
 now able to launch new DevEnvironments using this image.",
            "image ": {
                "release_type": "MAJOR VERSION",
                "image_name": "universal-image",
                "image_version": "2.0",
                "image_uri": "public.ecr.aws/amazonlinux/universal-image:2.0"
            }
        }
    },
    "Timestamp": "2021-09-03T19:05:57.882Z",
    "UnsubscribeURL": "example url"
}
```

使用 Amazon SNS 主控台訂閱 CodeCatalyst 通用映像更新

1. 開啟 Amazon SNS 主控台到儀表板。
2. 在導覽列中，選擇您的 AWS 區域.
3. 在導覽窗格中選擇 Subscriptions (訂閱)，然後選擇 Create subscription (建立訂閱)。
4. 在主題 ARN 中，輸入`arn:aws:sns:us-east-1:089793673375:universal-image-updates`。
5. 在 Protocol (通訊協定) 中，選擇 Email (電子郵件)。
6. 在端點中，提供電子郵件地址。此電子郵件地址將用於接收通知。
7. 選擇建立訂閱。
8. 您將收到一封主旨為「AWS 通知-訂閱確認」的確認電子郵件。開啟電子郵件並選擇 [確認訂閱]。

使用 Amazon SNS 主控台取消訂閱 CodeCatalyst 通用映像更新

1. 開啟 Amazon SNS 主控台到儀表板。
2. 在導覽列中，選擇您的 AWS 區域.
3. 在功能窗格中，選擇 [訂閱]，然後選取您要取消訂閱的訂閱。
4. 選擇 [動作]，然後選擇 [刪除訂閱]。
5. 選擇刪除。

通用圖像 1.0 運行時版本

下表列出的可用執行階段。`universal-image:1.0`

universal-image:1.0運行時版本

執行時間名稱	版本	特定主要和最新次要版本
AWS CLI	2.11	`aws-cli: 2.x`
docker 組成	2.16	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.19	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	科雷特托 17	`java: corretto17.x`
nodejs	14.20	`nodejs: 14.x`
	16.19	`nodejs: 16.x`
OpenSSL	1.0	`openssl: 1.x`
	1.1	
php	7.2	`php: 7.x`
python	3.9	`python: 3.x`
	3.10	
ruby	3.1	`ruby: 3.x`
地形	1.4	`terraform: 1.x`

通用圖像 2.0 運行時版本

下表列出的可用執行階段。`universal-image:2.0`

universal-image:2.0運行時版本

執行時間名稱	版本	特定主要和最新次要版本
AWS CLI	2.11	aws-cli: 2.x
docker 組成	2.17	docker-compose: 2.x
dotnet	6.0	dotnet: 6.x
	7.0	dotnet: 7.x
golang	1.20	golang: 1.x
java	corretto11	java: corretto11.x
	科雷特托 17	java: corretto17.x
nodejs	16.19	nodejs: 16.x
OpenSSL	1.0	openssl: 1.x
	1.1	
php	7.2	php: 7.x
python	3.9	python: 3.x
	3.10	
ruby	3.2	ruby: 3.x
地形	1.4	terraform: 1.x

通用圖像 3.0 運行時版本

下表列出的可用執行階段。universal-image:3.0

universal-image:3.0運行時版本

執行時間名稱	版本	特定主要和最新次要版本
AWS CLI	2.11	`aws-cli: 2.x`
docker 組成	2.17	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.21	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	科雷特托 17	`java: corretto17.x`
nodejs	18.17	`nodejs: 18.x`
	20.6	`nodejs: 20.x`
OpenSSL	3.0	`openssl: 3.x`
php	8.2	`php: 8.x`
python	3.9	`python: 3.x`
	3.11	
ruby	3.2	`ruby: 3.x`
地形	1.5	`terraform: 1.x`

透過 VPC 連線使用開發環境

VPC 連線是一種 CodeCatalyst 資源，其中包含存取 VPC 工作流程所需的所有組態。空間管理員可以代表空間成員在 Amazon CodeCatalyst 主控台中新增自己的 VPC 連線。透過新增虛擬私人雲端連線，Space 成員可以執行工作流程動作，並建立遵守網路規則並可存取相關 VPC 中的資源的開發環境。

您只能在建立開發環境時將開發環境與 VPC 連線相關聯。建立開發環境之後，您無法變更與開發環境相關聯的 VPC 連線。如果您想要使用不同的 VPC 連線，您必須刪除目前的開發環境並建立新的開發環境。

> **Important**
>
> 具有 VPC 連線的開發環境不支援連結至 CodeCatalyst的第三方來源儲存庫。

請注意，開發環境在創建時會使用多種 AWS 資源和服務。這表示開發環境會連線至下列 AWS 服務：

- Amazon CodeCatalyst
- AWS 超音波
- AWS KMS
- Amazon ECR
- Amazon CloudWatch
- Amazon ECS

> **Note**
>
> AWS 工具組 不支援使用關聯 VPC 連線建立開發環境。另請注意，如果您使用以外的 IDE AWS Cloud9，您可能會遇到大約五分鐘的載入時間。

您必須具有 S pace 管理員角色或超級使用者角色，才能在空間層級管理 VPC 連線。如需 VPC 的詳細資訊，請參閱《管理CodeCatalyst 員指南》 CodeCatalyst中的〈管理 Amazon VPC〉。

搭配 IDE 使用開發環境

您可以使用開發環境來快速處理儲存在專案原始碼儲存庫中的程式碼。開發環境可以提高您的開發速度，因為您可以在具有支援的整合式開發環境 (IDE) 的專案特定、功能完整的雲端開發環境中立即開始撰寫程式碼。

如需有關 CodeCatalyst 從 IDE 使用的資訊，請參閱下列文件。

- Amazon CodeCatalyst 適用於 JetBrains IDE
- Amazon CodeCatalyst VS 代碼

- Amazon CodeCatalyst AWS Cloud9

開發環境的配額 CodeCatalyst

下表說明 Amazon 中開發環境的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

每月開發環境小時數	開發環境時數會受到空間整體儲存限制的影響。如需詳細資訊，請參閱 定價和 疑難排解開發環境的問題。
每個空間的開發環境儲存量	開發環境存儲我們受到您空間的整體存儲限制的影響。如需詳細資訊，請參閱 定價和 疑難排解開發環境的問題。
開發環境運算量	開發環境運算會受到空間整體儲存限制的影響。如需詳細資訊，請參閱 定價和 疑難排解開發環境的問題。

中的套件 CodeCatalyst

Amazon CodeCatalyst 包含全受管套件存放庫服務，可讓您的開發團隊輕鬆安全地存放和共用用於應用程式開發的軟體套件。這些套件儲存在套件儲存庫中，這些套件儲存庫是在中的專案中建立和組織的 CodeCatalyst。

CodeCatalyst 支援下列套件格式：

- NPM

您可以探索套裝程式儲存區域中的套裝程式，並在包含儲存區域的專案成員之間共用。

若要將套件新增至存放庫，請將套件管理員設定為使用存放庫端點 (URL)。然後，您可以使用套件管理員將套件發佈至存放庫。

您可以設定 CodeCatalyst 工作流程，將套 CodeCatalyst 裝軟體發佈至套裝軟體庫，以及使用套裝軟體庫。如需在工作流程中使用封裝的詳細資訊，請參閱使用套件。

您可以將某個套件儲存庫中的套件新增為上游儲存庫，讓相同專案中的另一個儲存庫可用。上游存放庫可用的所有套件版本也可供下游存放庫使用。

您可以透過將公用的外部儲存庫連線至儲 CodeCatalyst 存庫，讓開放原始碼套件可供儲存庫使用。如需有關上游儲存庫和連線至外部儲存庫的詳細資訊，包括支援的儲存庫清單，請參閱使用上游儲存庫。

主題

數據包概念

以下是在中管理、發佈或使用套件時應瞭解的一些概念和術語 CodeCatalyst。

套件

套件是同時包含軟體與中繼資料的套裝軟體，以及安裝軟體並解決任何相依性所需的中繼資料。CodeCatalyst 支持 npm 包格式。

一個軟件包包括：

- 名稱 (例如，webpack是常用 npm 套件的名稱)
- 選用的命名空間 (例如，@types在中@types/node)
- 一組版本 (例如1.0.0、1.0.1、1.0.2)
- 套件層級中繼資料 (例如 npm dist 標籤)

Package 命名空間

某些套件格式支援階層式套件名稱，可將套件組織成邏輯群組，並協助避免名稱衝突。具有相同名稱的套件可以儲存在不同的命名空間中。例如，npm 支援範圍，而 npm 套件@types/node的範圍@types和名稱為node. @types範圍中還有許多其他軟件包名稱。在中 CodeCatalyst，範圍（「類型」）被稱為包命名空間，並且名稱（「節點」）被稱為包名稱。如果您沒有辦法對軟件包名稱進行分組，則避免名稱衝突可能會更加困難。

Package 版本

套件版本可識別套件的特定版本，例如@types/node@12.6.9。不同套件格式的版本號碼格式和語意會有所不同。例如，npm 套件版本必須符合語意版本控制規格。在中 CodeCatalyst，套件版本包含版本識別碼、中 package-version-level 繼資料和一組資產。

資產

資產是儲存在 CodeCatalyst 與封裝版本 (例如 npm 檔案) 相關聯的個別.tgz檔案。

Package 儲存庫

套件儲存庫 CodeCatalyst 包含一組套件，其中包含套件版本，每個套件版本都會對應至一組資產。每個套裝程式存放庫都會提供端點，以便使用 Node.js CLI (npm) 等工具擷取和發佈套件。您最多可以在每個空間建立 1,000 個套裝程式儲存區域。

您可以使用上游儲存庫將套裝程式儲存區域連結至另一個儲存庫。當您將套裝程式儲存區域連結為上游儲存庫時，您可以透過設定的存放庫使用連結儲存庫中的套裝程式。如需詳細資訊，請參閱 上游儲存庫。

Gateway 軟體庫是一種特殊類型的套件儲存庫，可從官方的外部套件授權單位提取和儲存套件。如需詳細資訊，請參閱 閘道儲存庫。

閘道儲存庫

閘道儲存庫是一種特殊類型的套件儲存庫，連接到受支援的外部官方套件授權單位。當您將閘道儲存庫新增為上游存放庫時，您可以使用來自對應官方套件授權單位的套件。您的下游存放庫不會與公用存放庫通訊，而是由閘道儲存庫中介所有項目。以這種方式使用的套件會儲存在收到原始要求的閘道儲存庫和下游存放庫中。

閘道儲存庫已預先定義，但必須在每個專案中建立才能使用。下列清單包含可在其中建立的每個閘道儲存庫，以 CodeCatalyst 及它們所連線的套件授權單位。

- npm-public-registry-gateway提供來自 NPM 的套件。

上游儲存庫

您可以使用 CodeCatalyst 在兩個套裝程式儲存庫之間建立上游關係。當套裝程式儲存區域所包含的套裝程式版本可以從下游存放庫的套裝程式儲存區域端點存取時，套裝程式儲存區域就是另一個 有了上游關係，兩個套裝程式儲存庫的內容會從用戶端的角度來有效地合併。

例如，如果套件管理員要求的套件版本不存在於儲存庫中，則 CodeCatalyst 會搜尋已設定的上游儲存庫以尋找套件版本。上游儲存庫會依照設定的順序進行搜尋，一旦找到套件，就 CodeCatalyst 會停止搜尋。

使用套件儲存庫

在中 CodeCatalyst，套件會儲存並管理在套件儲存庫中。若要將套裝程式發佈到 CodeCatalyst (CodeCatalyst 或任何支援的公用套裝程式儲存區域) 或使用套裝程式，您必須建立套裝程式儲存區域，然後將套裝程式管理員連線至套裝程式

主題

- 建立套裝程式儲存庫
- 連線至套裝程式儲存區域
- 編輯套裝程式儲存庫
- 刪除套裝程式儲存庫

建立套裝程式儲存庫

執行下列步驟，在中建立套裝程式儲存區域 CodeCatalyst。

建立套裝程式儲存區域

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至您要在其中建立套裝程式儲存區域的專案。
3. 在導覽窗格中，選擇 [封裝]。
4. 在「Package 程式儲存區域」頁面上，選擇建立套裝程式
5. 在「Package 式儲存區域詳細資訊」段落中，新增下列

   a. 存放庫名稱。請考慮使用具有詳細資訊的描述性名稱，例如專案或專案團隊名稱，或儲存庫的使用方式。

   b. (選擇性) 儲存庫說明。當您在專案中跨多個團隊擁有多個儲存庫時，存放庫描述特別有用。
6. 在 「編輯上游儲存區域」段落中，新增要透過套裝程式儲存區域存取的任何 CodeCatalyst 套裝程式儲存區域。您可以新增 Gateway 儲存區域以連線至外部套裝程式儲存區域或其他 CodeCatalyst 套裝程式

   - 從套裝程式儲存區域要求套裝軟體時，上游儲存庫會依照它們在此清單中顯示的順序搜尋。一旦找到包裹， CodeCatalyst 將停止搜索。若要變更上游存放庫的順序，您可以將存放庫拖放到清單中，或使用重新排序按鈕。
7. 選擇建立以建立套裝程式儲存區域。

連線至套裝程式儲存區域

若要將套件發佈至 CodeCatalyst 或從中使用套件 CodeCatalyst，您必須使用套件存放庫端點資訊和 CodeCatalyst 認證來設定套件管理員。如果您尚未建立儲存庫，可以依照中的指示執行建立套裝程式儲存庫。

有關如何將 npm 軟件包管理器連接到軟件 CodeCatalyst 包存儲庫的說明，請參閱配置和使用 npm。

編輯套裝程式儲存庫

執行下列步驟來編輯套裝程式儲存區域的描述和上游儲存庫。

編輯套裝程式儲存區域

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至包含您要編輯之套裝程式儲存區域的專案。
3. 在導覽窗格中，選擇 [封裝]。
4. 在「P ackage 程式儲存區域」頁面上，選擇要刪除的儲存區域。
5. 選擇「動作」 下拉式清單，然後選擇 「
6. 編輯存放庫描述和上游儲存庫。如需有關上游儲存庫的詳細資訊，請參閱使用上游儲存庫。
7. 選擇儲存。

刪除套裝程式儲存庫

執行下列步驟來刪除中的套裝程式儲存區域 CodeCatalyst。

刪除套裝程式儲存區域

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至包含您要刪除之套裝程式儲存區域的專案。
3. 在導覽窗格中，選擇 [封裝]。
4. 在「P ackage 程式儲存區域」頁面上，選擇要刪除的儲存區域。
5. 選擇「作業」 下拉式清單並選擇「刪除
6. 複查所提供有關刪除套裝程式儲存區域之影響的資訊。
7. 進入輸`delete`入字段，然後選擇刪除。

使用上游儲存庫

您可以將閘道儲存區域和其他套裝程式儲存區域做為上行串流連線至套裝程式儲存區域。這可讓套裝程式管理員用戶端使用單一套裝程式儲存區域端點，存取包含在多個套裝程式儲存區域中的套裝程式。以下是使用上游軟件庫的主要好處：

- 您只需要將套件管理員設定為單一存放庫端點，即可從多個來源提取。
- 從上游存放庫使用的套件會儲存在下游存放庫中，以確保即使上游存放庫發生非預期的中斷，您的套件也可以使用。

您可以在建立套件存放庫時新增上游儲存庫。您也可以從 CodeCatalyst 主控台中的現有套裝程式儲存區域新增或移除上游儲存庫。

當您將閘道儲存庫新增為上游儲存庫時，套裝程式儲存區域會連線至閘道儲存區域的對應公用套件儲存區域。如需支援之公用套件儲存區域的清單，請參閱支援的外部套件儲存庫及其閘道儲存庫。

您可以將多個存放庫作為上游存放庫連結在一起 例如，假設您的團隊創建了一個名為的存儲庫，`project-repo`並且已經使用另一個名為的存儲庫，`team-repo`該存儲庫已npm-public-registry-gateway添加為上游存儲庫，該存儲庫已連接到公共 npm 存儲庫`npmjs.com`。您可以將上游`team-repo`存放庫新增至`project-repo`。在這種情況下，您只需要將套件管理員設定`project-repo`為用來從`project-repo`、`team-repo`npm-public-registry-gateway、和中提取套件`npmjs.com`。

主題

- 新增上游存放庫
- 編輯上游儲存庫的搜尋順序
- 請求具有上游存儲庫的軟件包版本
- 移除上游存放庫

新增上游存放庫

將公用套裝程式存放庫或另一個 CodeCatalyst 套裝程式儲存庫作為上游存放庫新增至您的下游儲存庫，可讓連線至下游存放庫的套裝程式管理員使用。

若要新增上游存放庫

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面上，選擇要新增上游儲存區域的套裝程式儲存區域。
3. 選擇「動作」下拉式選單，然後選擇 「編輯
4. 在上游存放庫區段中，選擇新增上游存放庫。
5. 選擇搜尋列以顯示並搜尋可用儲存庫的清單。您可以將支援的公用套件儲存庫或其他 CodeCatalyst 儲存庫新增為上游儲存庫。找到要新增的儲存庫時，請從清單中選擇它。

> **Note**
>
> 在網關存儲庫中，有一個npm-public-registry-gateway存儲庫。若要連線至公用外部套件授權單位 (例如 npmjs.com)，請 CodeCatalyst 使用閘道儲存庫做為中介儲存庫，以搜尋

並儲存從外部儲存庫提取的套件。這樣可以節省時間和資料傳輸，因為專案中的所有套件儲存庫都會使用閘道儲存庫中的套件。

6. 選取要新增為上游儲存庫的所有儲存庫後，請選擇 [新增]。
7. 如需有關變更上游存放庫搜尋順序的詳細資訊，請參閱編輯上游儲存庫的搜尋順序。

新增上游儲存庫後，您可以使用連接到本機儲存庫的套件管理員，從上游儲存庫擷取套件。您不需要更新套件管理員設定。如需有關從上游存放庫要求套件版本的詳細資訊，請參閱請求具有上游存儲庫的軟件包版本。

編輯上游儲存庫的搜尋順序

CodeCatalyst 按照其配置的搜索順序搜索上游存儲庫。找到套件時， CodeCatalyst 會停止搜尋。您可以變更搜尋上游儲存庫中套裝軟體的順序。

若要編輯上游存放庫的搜尋順序

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「儲存區域」頁面上，選擇您要編輯其上游儲存區域搜尋順序的套裝程式儲存區域。
3. 選擇「動作」 下拉式清單，然後選擇 「
4. 在「上游存放庫」區段中，您可以檢視上游存放庫及其搜尋順序。若要變更搜尋順序，請將儲存庫拖放到清單中，或使用重新排序按鈕。
5. 編輯完上游儲存庫的搜尋順序後，請選擇 [儲存]。

請求具有上游存儲庫的軟件包版本

下列範例顯示套件管理員從具有上游儲存庫的套裝程式儲存區域要求 CodeCatalyst 套件時的可能案例。

在此範例中，套件管理員 (例如) 會從名downstream為npm具有多個上游儲存庫的套裝程式儲存庫要求套件版本。當要求套件時，可能會發生下列情況：

- 如果downstream包含請求的軟件包版本，則將其返回給客戶端。
- 如果downstream不包含要求的套件版本，請依其設定downstream的搜 CodeCatalyst 尋順序在上游儲存庫中搜尋該套件版本。如果找到套件版本，則會將其參照複製到downstream，並將套件版本傳回給用戶端。

- 如果downstream或其上游存放庫都不包含套件版本，則會將 HTTP 404 Not Found 回應傳回給用戶端。

一個儲存庫允許的直接上游儲存庫數目上限為 10。當要求套件版本時，儲存庫 CodeCatalyst 搜尋的數目上限為 25。

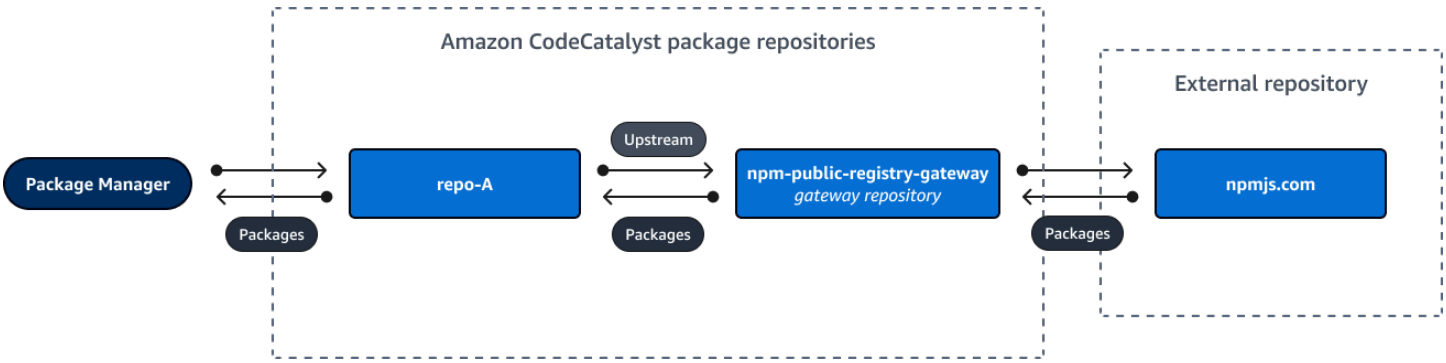
從上游儲存庫保留 Package

如果在上游存放庫中找到要求的套件版本，則會保留對該套件的參考，並且永遠可在要求該套件的存放庫中使用。如此可確保在上游儲存庫發生意外中斷時，您可以存取套件。保留的套件版本不受下列任何一項影響：

- 刪除上游存放庫。
- 中斷上游存放庫與下游存放庫的連線。
- 從上游存放庫刪除套件版本。
- 編輯上游存放庫中的套件版本 (例如，向其新增資產)。

透過上游關係擷取套件

CodeCatalyst 可以通過稱為上游存儲庫的多個鏈接存儲庫獲取包。如果套 CodeCatalyst 裝程式儲存庫具有與閘道儲存庫之上游連線的另一個 CodeCatalyst 套裝程式儲存庫的上游連線，則會從外部存放庫複製不在上游儲存庫中的套裝軟體的要求。例如，請考慮下列組態：名為的存放庫repo-A具有與閘道存放庫的上游連線npm-public-registry-gateway。 npm-public-registry-gateway與公共軟件包存儲庫有一個上游連接，https://npmjs.com。



如果設定npm為使用repo-A儲存庫，執行中npm install會啟動從 https://npmjs.com 將套件複製到npm-public-registry-gateway。安裝的版本也會被拉入repo-A。下列範例會安裝lodash。

```
$ npm config get registry
```

```
https://packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/
$ npm install lodash
+ lodash@4.17.20
added 1 package from 2 contributors in 6.933s
```

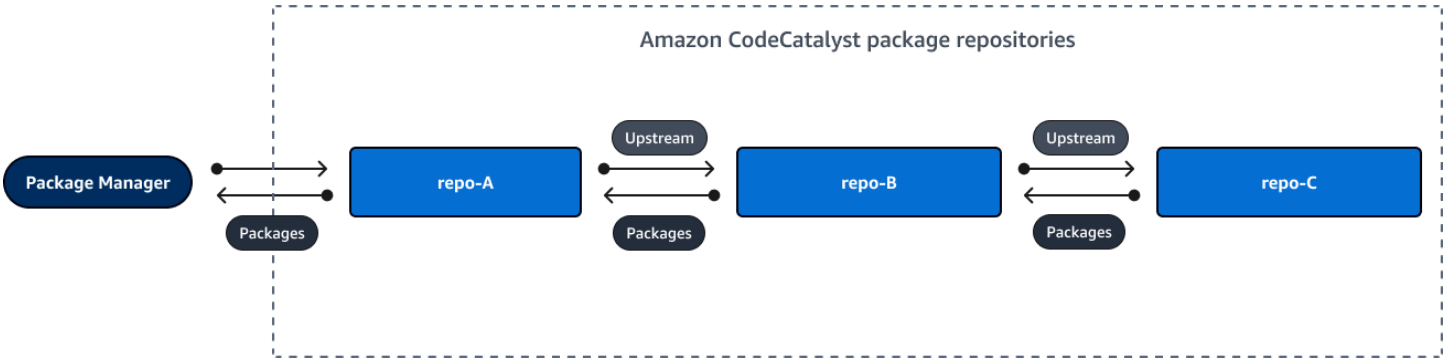
運行後npm install，僅repo-A包含最新版本（lodash 4.17.20），因為這是npm從中repo-A獲取的版本。

由於npm-public-registry-gateway有一個外部上游連接到 https://npmjs.com，所以從 https://npmjs.com 導入的所有軟件包版本都存儲在npm-public-registry-gateway. 這些套件版本可能已由任何具有上游連線的下游存放庫擷取。npm-public-registry-gateway

的內容npm-public-registry-gateway提供了一種方法，讓您可以查看從 https://npmjs.com 匯入的所有套件和套件版本。

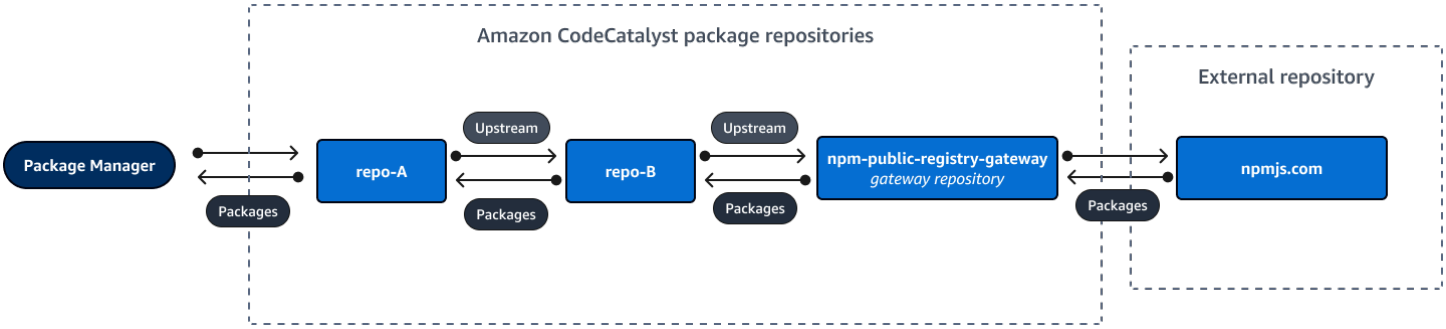
中繼儲存庫中的 Package 保留

CodeCatalyst 允許您鏈接上游存儲庫。例如，repo-A可以具有repo-B上游存儲庫，並且repo-B可以repo-C作為上游存儲庫。此配置使軟件包版本在中repo-B並從中repo-C提供repo-A。



當套裝程式管理員連線至儲存庫repo-A並從儲存庫擷取套裝程式版本時repo-C，套件版本不會保留在儲存庫repo-B中。封裝版本只會保留在最遠的下游存放庫中，在此範例中為。repo-A它不會保留在任何中間存儲庫中。對於較長的鏈結也是如此；例如，如果有四個存儲庫：repo-A、、和 repo-B repo-Crepo-D，以及連接到從中repo-A獲取軟件包版本的軟件包管理器repo-D，則該軟件包版本將保留在或中，repo-A但不會保留在repo-B或repo-C中。

從公用 Package 存放庫提取套件版本時，套件保留行為與公用套件存放庫有直接上游連線的閘道儲存庫中會一直保留套件版本。例如，repo-A具有repo-B作為上游存儲庫。 repo-Bnpm-public-registry-gateway作為一個上游存儲庫，它與公共存儲庫 npmjs.com 具有上游連接; 請參閱下圖。



如果連接到的軟件包管理器**repo-A**請求特定的軟件包版本（例如 lodash 4.17.20），並且該軟件包版本不存在於三個存儲庫中的任何一個，則將從 npmjs.com 提取該軟件包版本。當 lodash 4.17.20 被提取時，它會被保留在中，**repo-A**因為它是最遠的下游存儲庫，並且它具有與**npm-public-registry-gateway**公共外部存儲庫 npmjs.com 的上游連接。 lodash 4.17.20 不會保留在中，repo-B因為這是一個中繼儲存庫。

移除上游存放庫

如果您不想再存取上游儲存庫中的套件，您可以從套裝程式儲存庫中移除上游存放庫。

> **Warning**
>
> 當您刪除上游存儲庫時，您可能會破壞上游關係鏈，這可能會破壞您的項目或構建。

若要移除上游存放庫

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面上，選擇要從中移除上游儲存區域的套裝程式儲存區域。
3. 選擇「動作」 下拉式清單，然後選擇 「
4. 在「上游存放庫」區段中，找到您要移除的上游存放庫，然後選擇「移除」。
5. 完成移除上游儲存庫後，請選擇 [儲存]。

連接到公共外部存儲庫

您可以將對應的閘道儲存庫新增為上游儲存庫，將 CodeCatalyst 套裝程式儲存庫連接至受支援的公用外部儲存庫。閘道儲存庫是搜尋和儲存從外部儲存庫提取的套件的中介儲存庫。這樣可以節省時間和資料傳輸，因為專案中的所有套件儲存庫都使用閘道儲存庫中的套件。

使用閘道儲存庫連線至公用存放庫

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面的「閘道儲存區域」中，您可以檢視支援的閘道儲存區域清單及其說明。若要使用閘道儲存庫，首先必須建立它。如果已建立閘道存放庫，則會顯示其建立日期和時間。如果尚未建立，請選擇 [建立] 以建立它。
3. 選擇要連線至公用儲存區域的套裝程式儲存區域。
4. 選擇「動作」下拉式選單，然後選擇 「編輯
5. 若要連線至公用存放庫，請新增與您要做為上游存放庫連線的公用存放庫對應的閘道存放庫。

   在 「編輯上游儲存庫」段落中，選擇新增 CodeCatalyst 存放庫。
6. 「閘道儲存庫」區段會列出所有可用的閘道儲存庫。當您找到與您要連線的公用外部儲存庫對應的閘道儲存庫時，請從清單中選擇該儲存庫，然後選擇 [新增]。
7. 從存放庫要求套裝軟體時，會依其在 「編輯上游存放庫」清單中顯示的順序 CodeCatalyst 搜尋上游儲存庫。找到套件時， CodeCatalyst 會停止搜尋。若要變更上游存放庫的順序，請將存放庫拖放到清單中，或使用重新排序箭頭。
8. 當您完成新增和排序上游儲存庫時，請選擇 [儲存]。

當您將閘道儲存庫新增為上游儲存庫時，您可以使用連線到您本機存放庫的套件管理員，從對應的公開外部套件儲存庫中擷取套件。您不需要更新套件管理員設定。以這種方式使用的套件會同時儲存在閘道儲存庫和您的本機套件儲存庫中。如需有關從上游存放庫要求套件版本的詳細資訊，請參閱請求具有上游存儲庫的軟件包版本。

支援的外部套件儲存庫及其閘道儲存庫

CodeCatalyst 支援將上游連線新增至下列具有閘道儲存庫的官方套件授權單位。

儲存庫套件類型	描述	閘道儲存庫名稱
NPM	npm 公共註冊表	`npm-public-registry-gateway`

使用套件

中的套件 CodeCatalyst 是解決相依性和安裝軟體所需的軟體套件和中繼資料。 CodeCatalyst 支持 npm 軟件包格式。本節提供有關發佈、檢視和刪除封裝，以及更新封裝版本狀態的資訊。

主題

Package 發佈

您可以使用套件管理員工具，將任何受支援 CodeCatalyst 套件類型的版本發佈至套件存放庫。

如需將套件管理員連接至套件存放庫以發佈套件的相關資訊，請參閱連線至套裝程式儲存區域。 CodeCatalyst

內容

發佈和上游儲存庫

在中 CodeCatalyst，您無法發佈存在於可存取的上游存放庫或公用存放庫中的套件版本。例如，假設您想要將 npm 套件發佈至套件存放庫 `lodash@1.0`myrepo，並myrepo透過設定為上游存放庫的閘道儲存庫連線至 npmjs.com。如果lodash@1.0存在於上游儲存庫或 npmjs.com 中，則會透過發出 409 衝突錯誤來 CodeCatalyst 拒絕任何嘗試發佈至其中myrepo的嘗試。這有助於防止您意外發佈與上游儲存庫中套件名稱和版本相同的套件，這可能會導致非預期的行為。

您仍然可以發佈存在於上游存放庫中的不同版本的套件名稱。例如，如果存在lodash@1.0於上游存放庫中，但不存lodash@1.1在，則您可以發佈lodash@1.1至下游存放庫。

私有套件和公開儲存庫

CodeCatalyst 不會將儲存在儲存 CodeCatalyst 庫中的套件發佈到公用儲存庫，例如 npmjs.com。CodeCatalyst 將套件從公用儲 CodeCatalyst 存庫匯入至儲存庫，但不會以相反方向移動套件。您發佈至 CodeCatalyst 儲存庫的套件會保持非公開狀態，且僅供存放庫所屬的 CodeCatalyst 專案使用。

覆寫套件資產

您無法重新發佈已存在且其中包含不同內容的封裝資產。由於 npm 僅支持每個軟件包版本的單個資產，因此要修改已發布的軟件包版本，必須首先將其刪除。

檢視封裝版本詳細資

您可以使用 CodeCatalyst 主控台來檢視有關特定套件版本的詳細資料。

檢視套件版本詳細資訊

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面上，選擇包含您要檢視其詳細資訊之套裝程式版本的儲存區域。
3. 在「封裝」表格中搜尋套件版本。您可以使用搜尋列按套件名稱篩選套件。從清單中選擇套件。
4. 在「Package 件詳細資料」頁面中，選擇「版本」，然後選擇您要檢視的版本。

刪除套件版本

您可以從 CodeCatalyst 主控台的「Package」版本詳細資料頁面刪除套件版本。

刪除套件版本

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面上，選擇包含要刪除之套裝程式版本的儲存區域。
3. 搜尋並從表格中選擇套件。
4. 在「Package 件詳細資料」頁面上，選擇「版本」，然後選擇要刪除的版本。
5. 在 [P ackage 版本詳細資料] 頁面上，選擇 [版本動作]，然後選擇 [刪除
6. 在文字欄位中輸入刪除，然後選擇 [刪除]。

更新套件版本的狀態

中的每個套件版本都 CodeCatalyst 有描述套件版本目前狀態和可用性的狀態。您可以在 CodeCatalyst 主控台中變更套件版本狀態。如需有關套件版本可能狀態值及其含義的詳細資訊，請參閱Package 版本狀態。

更新套件版本的狀態

1. 在導覽窗格中，選擇 Packages (套件)。
2. 在「P ackage 程式儲存區域」頁面上，選擇包含要更新其狀態之套裝程式版本的儲存區域。
3. 搜尋並從表格中選擇套件。
4. 在 [Package 件詳細資料] 頁面上，選擇 [版本]，然後選擇您要檢視的版本。
5. 在「Package 件版本詳細資料」頁面上，選擇「動作」，然後選擇「取消刊登」、「封存」或「處理 如需有關每個套件版本狀態的資訊，請參閱Package 版本狀態。
6. 在文字欄位中輸入確認文字，然後根據您要更新的目標狀態，選擇「取消刊登」、「封存」或「處理」。

Package 版本狀態

以下是套件版本狀態的可能值。您可以在主控台中變更套件版本狀態。如需詳細資訊，請參閱 更新套件版本的狀態。

- 已發佈：套件版本已成功發佈，套件管理員可以要求。套件版本將包含在傳回給套件管理員的套件版本清單中；例如，在的輸出中`npm view <package-name> versions`。套件版本的所有資產都可從儲存庫取得。
- 未列出：套件版本資產可從儲存庫下載，但套件版本不包含在傳回至套件管理員的版本清單中。例如，對於 npm 套件，的輸出`npm view <package-name> versions`不包含套件版本。這表示 npm 相依性解析邏輯不會選取套件版本，因為該版本不會出現在可用版本清單中。但是，如果檔案中已參考「未列出」套`npm package-lock.json`件版本，則仍然可以下載並安裝該套件版本，例如執行`npm ci`時。
- 已封存：無法下載套件版本資產。套件版本不會包含在傳回至封裝管理員的版本清單中。由於資產無法使用，因此會封鎖用戶端使用套件版本。如果您的應用程式建置取決於已更新為封存的版本，除非套件版本已在本機快取，否則組建將會失敗。您無法使用套件管理員或建置工具來重新發佈封存的套件版本，因為它仍然存在於存放庫中。不過，您可以在主控台中將套件版本狀態變更回「未公開」或「已發佈」。

- 已處置：套件版本不會出現在清單中，且無法從儲存庫下載資產。「已處置」和「已封存」之間的主要區別在於，如果狀態為「已處置」，套件版本的資產會被永久刪除。 CodeCatalyst因此，您無法將封裝版本從「已處置」移至「已封存」、「未公開」或「已發佈」。無法使用封裝版本，因為資產已刪除。當套件版本標示為「已處置」時，系統不會向您收取封裝資產的儲存費用。

除了上述清單中的狀態之外，還可以刪除封裝版本。刪除套件之後，套件版本不在儲存庫中，您可以使用套件管理員或建置工具自由地重新發佈該套件版本。

編輯套件原點控制項

在 Amazon 中 CodeCatalyst，套件版本可以直接發佈、從上游儲存庫下拉，或從外部公用存放庫擷取套件版本，將其新增至套件儲存庫。如果您允許透過直接發佈和從公用存放庫擷取來新增套件版本，則容易遭受相依性替換攻擊。如需詳細資訊，請參閱 依賴替換攻擊。若要保護自己免受相依性替代攻擊，請在儲存庫中的套件上設定套件來源控制項，以限制如何將該套件的版本新增至儲存庫。

您應該考慮設定套件來源控制項，讓不同套件的新版本同時來自內部來源，例如直接發佈和外部來源，例如公開儲存庫。根據預設，套件原始控制項是根據第一個套件版本新增至儲存庫的方式來設定。

Package 原點控制設定

使用套件來源控制項，您可以設定如何將套件版本新增至儲存庫。下列清單包括可用的套件原點控制設定和值。

發布

此設定可設定是否可以使用套裝程式管理員或類似工具將套件版本直接發佈至儲存庫。

- 允許：Package 版本可以直接發布。
- BLOCK：Package 版本不能直接發布。

上游

此設定可設定套件版本是否可以從外部公用儲存庫擷取，或是在套件管理員要求時從上游儲存庫保留套件版本。

- 允許：任何套件版本都可以從設定為上游 CodeCatalyst 儲存庫的其他儲存庫中保留，或從具有外部連線的公用來源擷取的套件版本。
- BLOCK：無法從設定為上游儲存庫的其他 CodeCatalyst 儲存庫保留 Package 版本，或從具有外部連線的公用來源擷取套件版本。

預設套件原始碼控制設定

套裝程式的預設套裝程式來源控制項將以該套裝程式的第一個版本新增至套裝程式儲存區域的方式為基礎。

- 如果第一個軟件包版本由軟件包管理器直接發布，則設置將是「發布：允許」和「上游：塊」。
- 如果第一個套件版本是從公開來源擷取，則設定會是「發佈:區塊」和「上游:允許」。

常見的套件存取控制案例

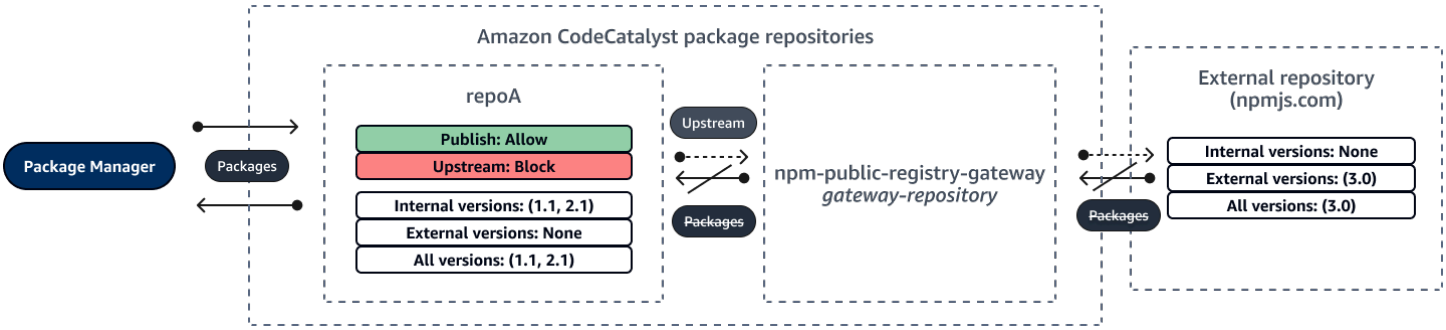
本節說明何時將套裝程式版本新增至套裝程式儲存區域的一 CodeCatalyst 些常見案例。根據第一個 Package 件版本的新增方式，會為新套件設定套件原始碼控制設定。

在下列案例中，內部套件會直接從套件管理員發行至您的存放庫，例如您維護的套件。外部套件是存在於公用存放庫中的套件，可透過外部連線擷取到儲存庫中。

針對現有內部套件發行外部套件版本

在這個案例中，請考慮一個內部套件，PackaGa。您的團隊會將 PackaGea 的第一個套件版本發佈至 CodeCatalyst 套件存放庫。因為這是該套件的第一個套件版本，所以套件原始控制項設定會自動設定為「發佈:允許」和「上游:區塊」。在您的儲存區域中發佈套裝程式之後，會將具有相同名稱的套裝程式發行至連線至 CodeCatalyst 套裝程式儲存區域的公用儲存區域。這可能是針對內部軟件包的嘗試依賴替換攻擊，也可能是巧合。無論如何，套件來源控制項都設定為封鎖新外部版本的擷取，以保護自己免受潛在攻擊。

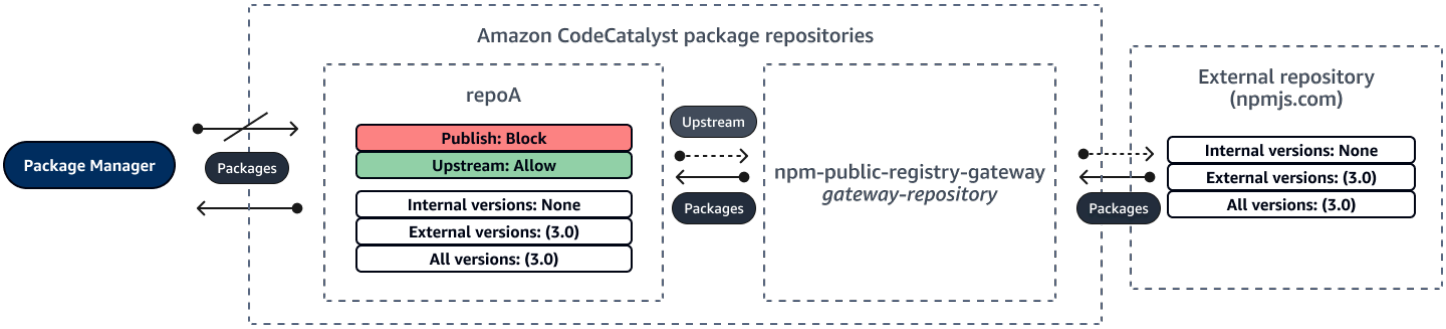
在下圖中，RePoA 是您的 CodeCatalyst 套件儲存庫，具有外部連線至公用存放庫。您的存儲庫包含的版本 1.1 和 2.1 的 PackaGa，但 3.0 版本已發佈到公共存儲庫。一般而言，RePoA 會在套件管理員要求套件之後擷取 3.0 版。由於套件擷取設定為「封鎖」，因此 3.0 版本不會擷取到您的 CodeCatalyst 套件儲存庫中，而且無法提供給連線到它的套件管理員。



針對現有外部套件發行內部套件版本

在這個案例中，套件 PackageB 存在於外部您已連線至存放庫的公用存放庫中。當連接到儲存庫的套件管理員要求 PackageB 時，套件版本會從公用存放庫擷取到您的存放庫中。因為這是第一個套件版本的 PackageB 新增至您的儲存庫，因此套件原始設定會設定為「發佈:區塊」和「上游:允許」。稍後，您嘗試將具有相同套件名稱的版本發佈到存放庫。您可能不知道公用套件，並嘗試以相同的名稱發佈不相關的套件，或者您可能嘗試發佈修補的版本，或者您可能會嘗試直接發佈外部已存在的套件版本。 CodeCatalyst 拒絕您嘗試發行的版本，但如有必要，您可以明確覆寫拒絕並發佈版本。

在下圖中，RePoA 是您的 CodeCatalyst 套件儲存庫，具有外部連線至公用存放庫。您的套件存放庫包含從公用存放庫擷取的 3.0 版。您想要將 1.2 版發佈至您的套件存放庫。通常，您可以將 1.2 版發佈至 RePoA，但由於發佈設定為「封鎖」，因此無法發佈 1.2 版。



發佈現有外部套件的修補套件版本

在這個案例中，套件 PackageB 存在於外部，您已連線至套裝程式儲存區域的公用存放庫中。當連接到儲存庫的套件管理員要求 PackageB 時，套件版本會從公用存放庫擷取到您的存放庫中。因為這是第一個套件版本的 PackageB 新增至您的儲存庫，因此套件原始設定會設定為「發佈:區塊」和「上游:允許」。您的團隊決定將此套件的已修補套件版本發佈至存放庫。為了能夠直接發佈套件版本，您的團隊將套件原始控制項設定變更為「發佈:允許」和「上游:BLOCK」。此套件的版本現在可以直接發佈至您的儲存庫，並從公用存放庫擷取。在您的團隊發佈已修補的套件版本之後，您的團隊會將套件來源設定還原為「發佈:BLOCK」和「上游:允許」。

編輯套件原點控制項

Package 程式原始控制項會根據套裝程式的第一個套裝程式版本新增至套裝程式儲存區域的方式，自動設定。如需詳細資訊，請參閱 預設套件原始碼控制設定。若要新增或編輯套裝程式儲存區域中套裝程式的 CodeCatalyst 套裝程式原始控制項，請執行下列程序中的步驟。

新增或編輯套件原始控制項

1. 在導覽窗格中，選擇 Packages (套件)。
2. 選擇包含要編輯之套裝程式的套裝程式儲存區域。
3. 在「封裝」表格中，搜尋並選擇您要編輯的封裝。

4. 在套件摘要頁面中，選擇編輯原始控制項。
5. 在 Origin 控制項中，選擇您要為此套件設定的套件原始控制項。必須同時設定套件原始控制項設定 (「發佈」和「上游」)。

   - 若要允許直接發佈封裝版本，請在 [發佈] 中選擇 [允許]。若要封鎖套件版本的發佈，請選擇 [封鎖]。
   - 若要允許從外部儲存庫擷取套裝程式並從上游儲存庫提取套裝程式，請在上游來源中選擇允許。若要封鎖所有從外部和上游儲存庫擷取套件版本，請選擇「封鎖」。
6. 選擇儲存。

發佈和上游儲存庫

在中 CodeCatalyst，您無法發佈存在於可存取的上游存放庫或公用存放庫中的套件版本。例如，假設您想要將 npm 套件發佈`lodash@1.0`至儲存庫`myrepo`，並且`myrepo`具有與 npmjs.com 外部連線的上游儲存庫。請考慮下列案例。

1. 上的套件原始控制項設定`lodash`為「發佈：允許」和「上游：允許」。如果`lodash@1.0`存在於上游儲存庫或 npmjs.com 中，則會透過發出 409 衝突錯誤來 CodeCatalyst 拒絕任何嘗試發佈至其中`myrepo`的嘗試。您仍然可以發佈不同的版本，例如`lodash@1.1`.
2. 上的套件原始控制項設定`lodash`為「發佈:允許」和「上游:區塊」。您可以將任何版本的發佈`lodash`到尚未存在的存放庫，因為無法存取套件版本。
3. 上的套件原始控制項設定`lodash`為「發佈:區塊」和「上游:允許」。您無法將任何套件版本直接發佈到存放庫。

依賴替換攻擊

Package 管理器簡化了打包和共享可重複使用代碼的過程。這些套件可能是由組織開發用於其應用程式的私有套件，也可能是公開的，通常是開放原始碼套件，這些套件是在組織外部開發並由公開套件儲存庫散佈的。請求軟件包時，開發人員依賴他們的軟件包管理器來獲取其依賴項的新版本。相依性替換攻擊 (也稱為相依性混淆攻擊) 會利用套件管理員通常無法區分合法版本的套件與惡意版本的事實。

依賴性替換攻擊屬於被稱為軟件供應鏈攻擊的攻擊子集。軟件供應鏈攻擊是一種利用軟件供應鏈中任何地方漏洞的攻擊。

依賴性替換攻擊可以針對使用內部開發的軟件包和從公共存儲庫獲取的軟件包的任何人。攻擊者識別內部軟件包名稱，然後策略性地將具有相同名稱的惡意代碼放置在公共軟件包存儲庫中。一般而言，惡意程式碼會以較高版本號碼的套件發佈。Package 件管理員會從這些公開摘要擷取惡意程式碼，因

為他們認為惡意套件是套件的最新版本。這會導致所需的軟件包和惡意軟件包之間出現「混淆」或「替代」，從而導致代碼受到攻擊。

為了防止相依性替換攻擊，Amazon CodeCatalyst 提供套件來源控制。Package 件原始控制項是控制套件新增至儲存庫的方式的設定。當新套件的第一個套件版本新增至 CodeCatalyst 儲存庫時，控制項會自動設定控制項。控制項可確保套件版本不能同時直接發佈至您的儲存庫，並從公開來源擷取，以保護您免受相依性替代攻擊。如需套件來源控制項以及如何變更它們的詳細資訊，請參閱編輯套件原點控制項。

使用 NPM

這些主題說明如何使用 npm Node.js 套件管理員與 CodeCatalyst.

> **Note**
>
> CodeCatalyst 支持以`node v4.9.1`及更高版本`npm v5.0.0`和更高版本。

主題

- 配置和使用 npm
- npm 標籤處理

配置和使用 npm

若要npm搭配使用 CodeCatalyst，您必須連線npm至套件存放庫，並提供個人存取權杖 (PAT) 以進行驗證。您可以在 CodeCatalyst 主控台中檢視連線npm至套裝程式儲存區域的指示。

內容

- 使用 npm 配置 CodeCatalyst
- 從套件儲存庫安裝 npm 套件 CodeCatalyst
- 通過以下方式從 npmjs 安裝 npm 軟件包 CodeCatalyst
- 將 npm 套件發佈到您的 CodeCatalyst 套件存放庫
- npm 命令支持
  - 支援與套件儲存庫互動的指令
  - 支援的用戶端命

- 不支援的命

使用 npm 配置 CodeCatalyst

下列指示說明如何驗證並連線npm至 CodeCatalyst 套件儲存庫。有關 npm 的更多信息，請參閱官方的 npm 文檔。

連線**npm**至您的 CodeCatalyst 套裝程式儲存區域

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目。
3. 在導覽窗格中，選擇 Packages (套件)。
4. 從清單中選擇您的套裝程式儲存區域。
5. 選擇「Connect 至儲存庫」。
6. 在配置詳細信息的 P ackage 管理器客戶端中，選擇 npm 客戶端。
7. 選擇您的作業系統以檢視對應的組態步驟。
8. 需要個人訪問令牌（PAT）來驗證 npm CodeCatalyst。如果您已經有令牌，則可以使用它。如果沒有，您可以使用以下步驟創建一個。

   a. (選擇性)：更新 P AT 名稱和到期日。

   b. 選擇 [建立權杖]。

   c. 將 PAT 複製並存儲在安全的位置。

   > ⚠ Warning
   >
   > 關閉對話方塊後，您將無法再次查看或複製 PAT。認證應該是短暫的，以盡量減少攻擊者在盜用憑證後可以使用憑證的時間長度。

9. 從項目的根目錄運行以下命令，以使用您的包存儲庫配置 npm。這些命令將執行以下操作：

   - 如果您的專案沒有專`.npmrc`案層級檔案，請建立專案層級檔案。
   - 將套件存放庫端點資訊新增至您的專案層級`.npmrc`檔案。
   - 將您的認證 (PAT) 新增至使用者層級`.npmrc`檔案。

   取代下列值。

> **Note**
>
> 如果您是從主控台指示複製，則會為您更新下列指令中的值，而且不需要變更。

- 用您的用戶*#*替換 CodeCatalyst 用戶名。
- 以您的 *PAT* 取代 CodeCatalyst PAT。
- 將*####*取代為您的 CodeCatalyst 空間名稱。
- 用您 CodeCatalyst 的項目名稱替換 *proj_name*。
- 以您的 CodeCatalyst 套件儲存區*##### repo_name*。

```
npm set registry=https://packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/ --location project
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/:_authToken=username:PAT
```

對於 npm 6 或更低版本：要使 npm 始終將身份驗證令牌傳遞給甚至對於GET請求，請按如下方式設置始終身份驗證配置變量。 CodeCatalyst npm config set

```
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/:always-auth=true --location project
```

從套件儲存庫安裝 npm 套件 CodeCatalyst

按照中的步驟將 npm 連接到儲存庫之後使用 npm 配置 CodeCatalyst，您就可以在存放庫上執行npm命令。

您可以使用npm install指令來安裝 CodeCatalyst 套件儲存庫或其上游儲存庫中的 npm 套件。

```
npm install lodash
```

通過以下方式從 npmjs 安裝 npm 軟件包 CodeCatalyst

您可以透過儲存庫使用連線至 npmjs.com 的閘道 CodeCatalyst 儲存庫的上游連線來設定儲存庫，從 npmjs.com 安裝 npm 套件。npm-public-registry-gateway從 npmjs 安裝的套件會擷取並儲存在閘道儲存庫和最遠的下游套件儲存區域中。

若要從 npmjs 安裝套件

1. 如果您尚未這麼做，請依照中的步驟npm使用 CodeCatalyst套件儲存庫進行設定使用 npm 配置 CodeCatalyst。
2. 檢查您的存放庫是否已將閘道儲存庫新增為上游連線。npm-public-registry-gateway您可以按照中的指示新增上游存放庫並選擇npm-public-registry-gateway存放庫，來檢查要新增或新增哪些上游來源npm-public-registry-gateway作為上游來源。
3. 使用`npm install`指令安裝套件。

```
npm install package_name
```

如需有關從上游存放庫要求套件的詳細資訊，請參閱請求具有上游存儲庫的軟件包版本。

將 npm 套件發佈到您的 CodeCatalyst 套件存放庫

完成後使用 npm 配置 CodeCatalyst，您可以執行npm命令。

您可以使用`npm publish`指令將 npm 套件發佈至 CodeCatalyst 套件儲存庫。

```
npm publish
```

如需如何建立 npm 套件的詳細資訊，請參閱在 npm 文件上建立 Node.js 模組。

npm 命令支持

下列段落摘要說明 CodeCatalyst套裝程式儲存區域支援的命令，並列出不支援的特定命令。npm

主題

- 支援與套件儲存庫互動的指令
- 支援的用戶端命
- 不支援的命

支援與套件儲存庫互動的指令

本節列出npm用npm戶端對其設定的登錄發出一或多個要求的命令 (例如，`npm config set registry`)。這些指令已經過驗證，在針對 CodeCatalyst 套件儲存區域叫用時可以正確運作。

Command	描述
蟲子	猜測軟件包錯誤跟踪器 URL 的位置，然後嘗試打開它。
CI	使用乾淨的石板安裝專案。
棄用	棄用套件的版本。
死亡標籤	修改封裝發佈標籤。
文件	猜測軟件包的文檔 URL 的位置，然後嘗試使用 `--browser` config 參數打開它。
醫生	執行一組檢查，以驗證您的 npm 安裝是否可以管理您的 JavaScript 套件。
安裝	安裝套件。
install-ci-test	使用乾淨的平板安裝項目並運行測試。別名：`npm cit`。此命令運行`npm ci`，然後立即運行`npm test`.
安裝測試	安裝軟件包並運行測試。運行`npm install`，然後立即執行`npm test`.
過時	檢查設定的登錄，以判斷是否有任何已安裝的套件已過期。
平	ping 配置或給定的 npm 註冊表並驗證身份驗證。
發佈	將套件版本發佈至登錄。
update	猜測套件存放庫 URL 的位置，然後嘗試使用 `--browser` config 參數來開啟它。
檢視	顯示套件中繼資料。也可用於列印中繼資料屬性。

支援的用戶端命

這些命令不需要與套件儲存庫進行任何直接互動，因此 CodeCatalyst不需要任何東西來支援它們。

Command	描述
垃圾桶 (舊版)	顯示 NPM bin 目錄。
建立	構建一個包。
快取	操作套件快取。
完成	在所有 npm 命令中啟用選項卡完成。
配置	更新使用者和全域npmrc檔案的內容。
刪除	搜尋本機套件樹狀結構，並嘗試將相依性往上移動樹狀結構，以便讓多個相依套件更有效地共用相依性，藉此簡化結構。
編輯	編輯已安裝的套件。選取目前工作目錄中的相依性，並在預設編輯器中開啟套件目錄。
探索	瀏覽已安裝的套件。在指定的已安裝套件的目錄中產生一個子 shell。如果指定了一個命令，那麼它在子 shell 中運行，然後立即關閉。
help	獲取關於 npm 的幫助。
幫助搜索	搜索 npm 幫助文檔。
初始化	建立package.json 檔案。
鏈接	符號鏈接一個包目錄。
ls	列出已安裝的套件。
包	從封裝建立壓縮包。
prefix	顯示首碼。除非另外指定，否則這-g是包含package.json 文件的最接近父目錄。

Command	描述
修剪	移除父套件相依性清單中未列出的套件。
重建	在相符的資料夾上執行npm build命令。
重啟	執行套件的停止、重新啟動和啟動指令碼，以及相關的前置指令碼和後置指令碼。
根	將有效目node_modules 錄打印為標準輸出。
運行腳本	執行任意套件指令碼。
shrinkwrap	鎖定要發佈的相依性版本。
卸載	解除安裝套件。

不支援的命

CodeCatalyst 套件儲存區域不支援這些npm指令。

Command	描述	備註
存取	設定已發佈封裝的存取層級。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的權限模型。
添加用戶	新增登錄使用者帳戶	CodeCatalyst 使用與公共 npmjs 存儲庫不同的用戶模型。
審計	執行安全稽核。	CodeCatalyst 目前沒有出現安全漏洞數據。
掛鉤	管理 npm 掛鉤，包括添加，刪除，列出和更新。	CodeCatalyst 目前不支援任何變更通知機制。
登入	驗證使用者。這是 npm adduser 的別名。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的身份驗證

Command	描述	備註
		模型。如需相關資訊，請參閱使用 npm 配置 CodeCatalyst。
登出	登出登錄。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的身份驗證模型。無法從 CodeCatalyst 存儲庫註銷，但是身份驗證令牌在其可配置的到期時間後過期。默認令牌持續時間為 12 小時。
所有者	管理套件擁有者。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的權限模型。
profile	變更登錄設定檔的設定。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的用戶模型。
search	在登錄中搜尋符合搜尋字詞的套件。	CodeCatalyst 不支援該`search`指令。
明星	標記您最喜歡的軟件包。	CodeCatalyst 目前不支持任何收藏夾機制。
星星	檢視標記為我的最愛的套件。	CodeCatalyst 目前不支持任何收藏夾機制。
團隊	管理團隊和團隊成員資格。	CodeCatalyst 使用與公共 npmjs 存放庫不同的使用者和群組成員資格模型。
token	管理您的身份驗證令牌。	CodeCatalyst 使用不同的模型來獲取身份驗證令牌。如需相關資訊，請參閱使用 npm 配置 CodeCatalyst。

Command	描述	備註
取消發佈	從登錄中移除套件。	CodeCatalyst 不支持使用 npm 客戶端從存儲庫中刪除軟件包版本。您可以在主控台中刪除套件。
哇美	顯示 npm 使用者名稱。	CodeCatalyst 使用與公共 npmjs 存儲庫不同的用戶模型。

npm 標籤處理

npm 註冊表支持標籤，這是包版本的字符串別名。您可以使用標籤來提供別名，而不是使用版本號碼。例如，您有一個具有多個開發流的項目，並且為每個流使用不同的標籤（例如 `stablebeta`，`dev`，，`canary`）。有關更多信息，請參閱 npm 文檔上的 dist-tag。

依預設，npm 會使用`latest`標籤來識別套件的目前版本。 `npm install` *`pkg`*（沒有`@`*`version`*或`@`*`tag`*說明符）安裝最新的標籤。一般而言，專案只會針對穩定版本使用最新標籤。其他標籤用於不穩定版或預發布版本。

使用 npm 用戶端編輯標籤

這三個`npm dist-tag`命令 (`add`、`rm`、和`ls`) 在 CodeCatalyst 套件儲存庫中的運作方式與它們在預設 npm 登錄中的運作方式相同。

npm 標籤和上游存儲庫

當`npm`要求套件的標籤以及該套件的版本也存在於上游存放庫中時，會先 CodeCatalyst 合併標籤，然後再將它們傳回給用戶端。例如，名為的存放庫`R`有一個名為的上游存放庫`U`。下表顯示兩個儲存庫中存在`web-helper`的套件的標籤。

儲存庫	套件名稱	Package 標籤
R	`web-helper`	最新（別名版本 1.0.0）
U	`web-helper`	阿爾法（版本 1.0.1 的別名）

在這種情況下，當 npm 客戶端從存儲庫獲取web-helper包的標籤時R，它會同時接收最新和 alpha 標籤。標籤指向的版本不會更改。

當上游和本機存放庫中的相同套件上存在相同的標籤時， CodeCatalyst 會使用上次更新的標籤。例如，假設 webhelper 上的標籤已被修改為如下所示。

儲存庫	套件名稱	Package 標籤	上次更新
R	web-helper	最新（別名版本 1.0.0）	2023年1月1日
U	web-helper	最新的（別名版本 1.0.1）	2023 年 6 月 1 日

在這種情況下，當 npm 客戶端從存儲庫獲取包 webhelper 的標籤時R，最新的標籤將別名版本 1.0.1，因為它是最後更新的。這可讓您輕鬆使用上游儲存庫中尚未存在於本機儲存庫中的新套件版本npm update。

套件配額

下表說明 Amazon 中套件的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

資源	預設配額
Package 儲存庫	每個空間最多可容納 1000 個。
直接上游存儲庫	每個套裝程式儲存區域最多 10 個。
搜尋上游套件儲存庫	每個要求的套件版本最多搜尋 25 個上游儲存庫。
Package 資產檔案大小	每個套件資產最多可容納 5GB。
Package 資產	每個軟件包版本最多 150 個。

使用中的工作流程來建置、測試和部署 CodeCatalyst

在CodeCatalyst開發環境中撰寫應用程式程式碼並將其推送到CodeCatalyst 原始碼儲存庫之後，您就可以進行部署。自動執行此操作的方法是通過工作流程。

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要採取的一系列步驟或動作。工作流程也會定義導致工作流程啟動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

關於工作流程定義檔

工作流程定義檔案是描述您工作流程的 YAML 檔案。檔案儲存在來源儲存庫根目錄的`~/.codecatalyst/workflows/`資料夾中。檔案的副檔名可以是 .yml 或 .yaml。

以下是簡單工作流程定義檔案的範例。我們在下面的表中解釋這個例子的每一行。

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest .
```

折線圖	描述
`Name: MyWorkflow`	指定工作流程的名稱。若要取得有關Name性質的更多資訊，請參閱頂層屬性。
`SchemaVersion: 1.0`	指定工作流程結構描述版本。若要取得有關SchemaVersion 性質的更多資訊，請參閱頂層屬性。
`RunMode: QUEUED`	指示 CodeCatalyst 處理多個梯段的方式。如需執行模式的詳細資訊，請參閱設定佇列、已取代和 parallel 執行。
`Triggers:`	指定將導致工作流程執行開始的邏輯。關於觸發條件的詳細資訊，請參閱 使用觸發程序。

`- Type: PUSH`
`  Branches:`
`  - main`
	指出每當您將程式碼推送至預設來源儲存庫的main分支時，工作流程都必須啟動。如需有關工作流程來源的詳細資訊，請參閱使用來源。

`Actions:`	定義工作流程執行期間要執行的工作。在此範例中，Actions區段定義了名為的單一動作Build。如需動作的詳細資訊，請參閱使用動作。
`Build:`	定義動Build作的屬性。如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
`Identifier: aws/build@v1`	指定建置動作的唯一硬式編碼識別元。

`Inputs:`
`  Sources:`
`  - WorkflowSource`
	指出建置動作應該在來WorkflowSource 源儲存庫中尋找完成處理所需的檔案。如需詳細資訊，請參閱 使用來源。

`Configuration:`	包含建置動作特定的組態屬性。

折線圖	描述

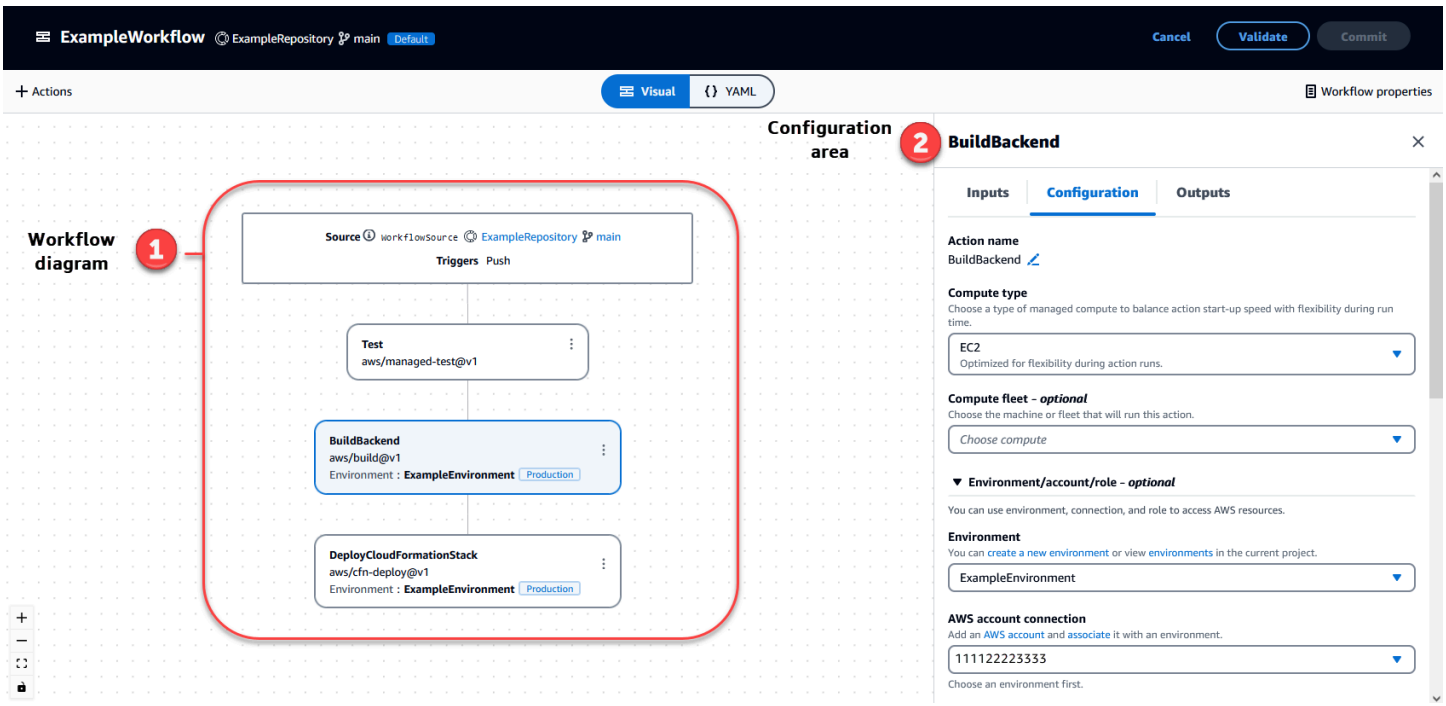
`Steps:`
`  - Run: docker build -t MyApp:lat`
`est .`
	告訴構建操作構建一個名為 Docker 映像MyApp並標記它。`latest`

如需工作流程定義檔案中所有可用性質的完整清單，請參閱工作流定義參考。

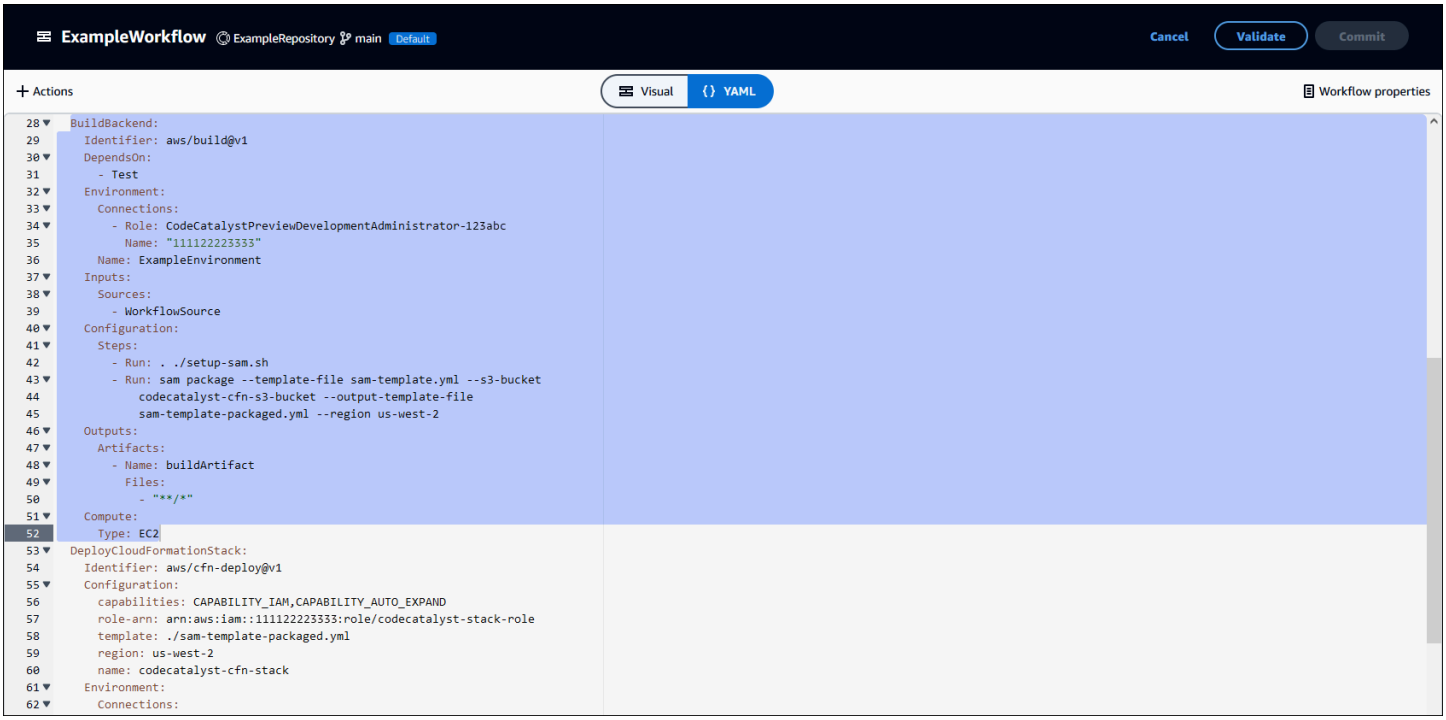
使用主 CodeCatalyst 控台的視覺化和 YAML 編輯器

若要建立和編輯工作流程定義檔案，您可以使用偏好的編輯器，但我們建議您使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器。這些編輯器提供有用的檔案驗證，以協助確保 YAML 屬性名稱、值、巢狀、間距、大小寫等正確無誤。

下圖顯示了可視化編輯器中的工作流程。視覺化編輯器為您提供完整的使用者介面，透過該介面建立和設定工作流程定義檔案。可視化編輯器包括工作流程圖 (1) 顯示工作流程的主要元件，以及配置區域 (2)。



或者，您可以使用 YAML 編輯器，如下圖所示。使用 YAML 編輯器貼上大型程式碼區塊 (例如，從教學課程)，或新增視覺化編輯器未提供的進階屬性。

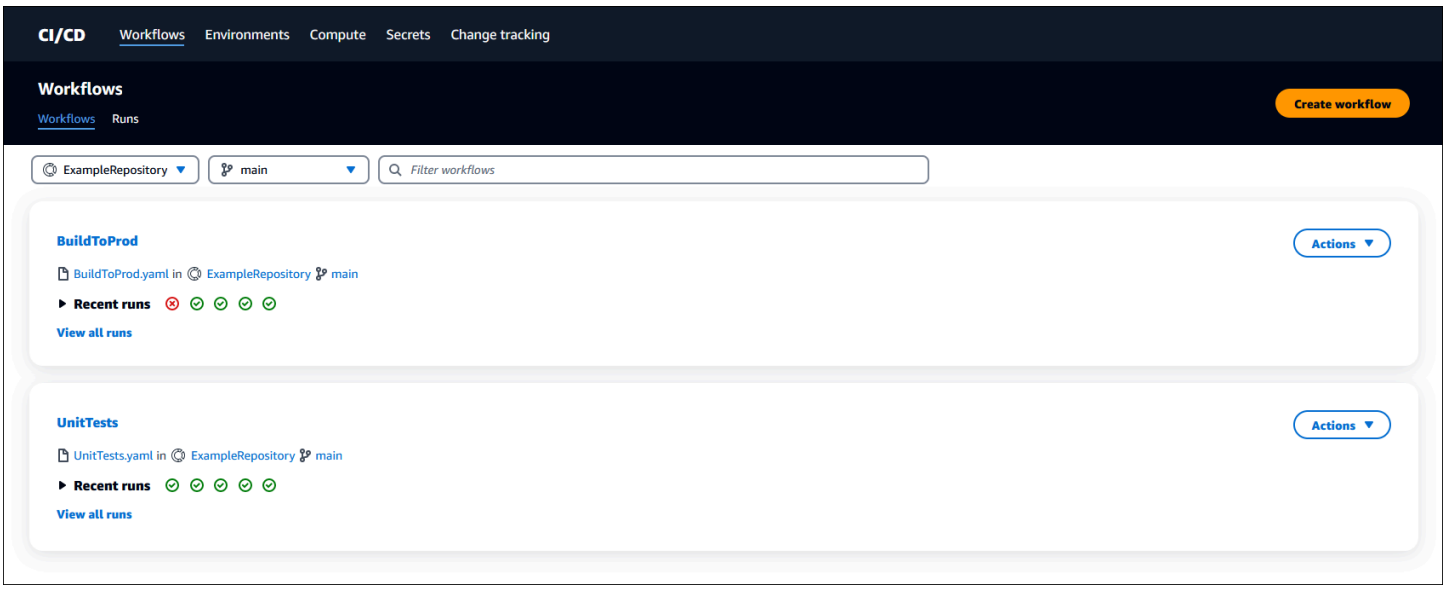


您可以從視覺化編輯器切換至 YAML 編輯器，以查看您的設定對基礎 YAML 程式碼的效果。

探查工作流

您可以在 [工作流程摘要] 頁面上檢視工作流程，以及您在同一專案中設定的其他工作流程。

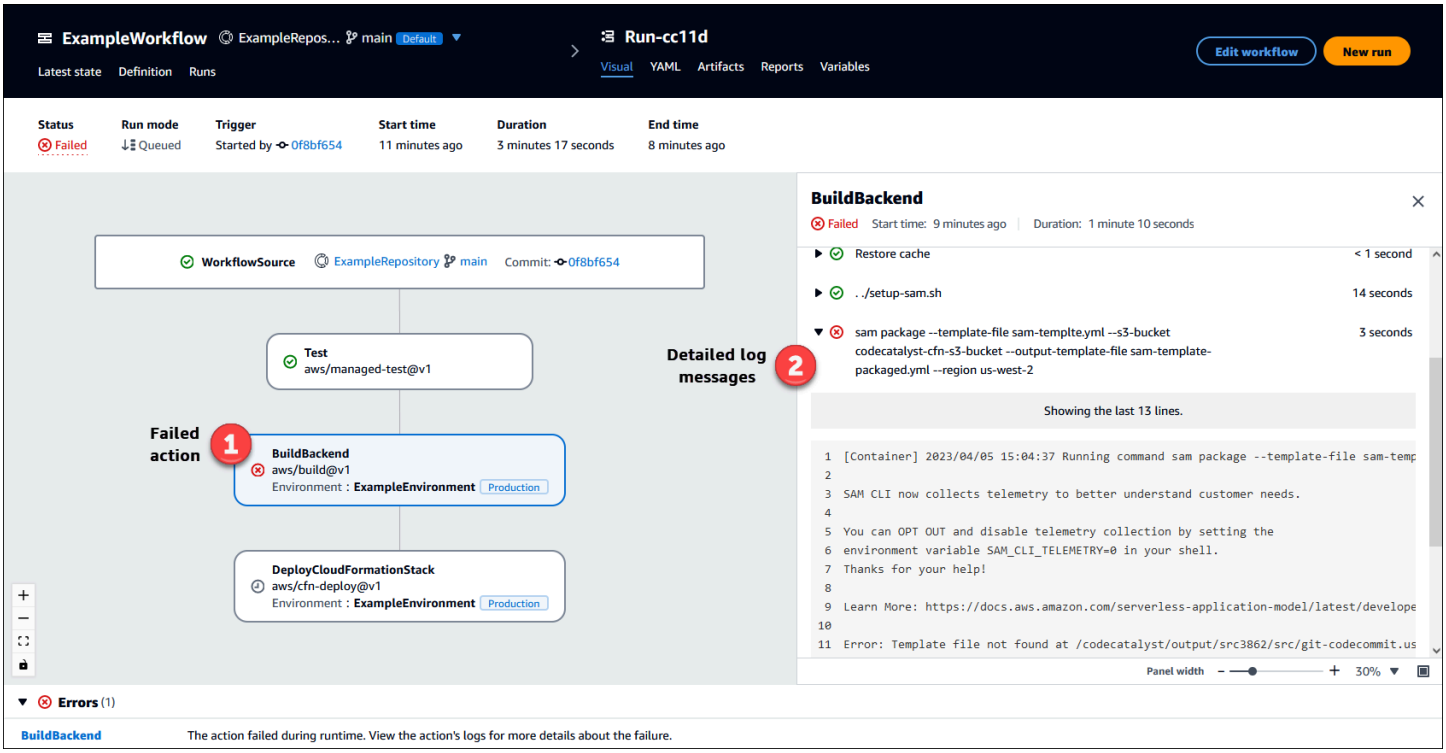
下圖顯示 [工作流程摘要] 頁面。它會填入兩個工作流程：BuildToProd和UnitTests。您可以看到兩者都已經運行了幾次。您可以選擇 [最近執行] 以快速查看執行歷程記錄，或選擇工作流程名稱以查看工作流程的 YAML 程式碼和其他詳細資訊。



檢視工作流程執行詳

您可以在 [工作流程摘要] 頁面中選擇執行，以檢視工作流程執行的詳細資訊。

下圖顯示名為 Run-CC11d 的工作流程執行的詳細資料，該工作流程執行是在提交至來源時自動啟動的。工作流程圖表示動作失敗 (1)。您可以瀏覽至記錄 (2) 以檢視詳細的記錄訊息並疑難排解問題。如需工作流程執行的更多資訊，請參閱使用梯段。



後續步驟

若要進一步瞭解工作流程概念，請參閱工作流概念。

若要建立第一個工作流程，請參閱開始使用中的工作流程 CodeCatalyst。

工作流概念

以下是在中使用工作流程建置、測試或部署程式碼時應瞭解的一些概念和術語 CodeCatalyst。

工作流程

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要採取的一系列步驟或動作。工作流程也會定義導致工作流程啟

動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

如需工作流程的相關詳細資訊，請參閱 使用工作流程。

工作流程定義檔

工作流程定義檔案是描述您工作流程的 YAML 檔案。檔案儲存在來源儲存庫根目錄的`~/.codecatalyst/workflows/`資料夾中。檔案的副檔名可以是 .yml 或 .yaml。

若要取得有關工作流程定義檔的更多資訊，請參閱工作流定義參考。

動作

動作是工作流程的主要建置區塊，可定義工作流程執行期間要執行的邏輯工作單元。一般而言，工作流程包含多個依序執行或 parallel 執行的動作，具體取決於您設定它們的方式。

如需動作的詳細資訊，請參閱使用動作。

動作群組

動作群組包含一或多個動作。將動作分組到動作群組中，可協助您保持工作流程井然有序，也可讓您配置不同動作群組之間的相依性。

如需有關動作群組的詳細資訊，請參閱將動作分組到動作群組中。

成品

人工因素是工作流程動作的輸出，通常由資料夾或檔案的封存組成。人工因素很重要，因為它們可讓您在動作之間共用檔案和資訊。

如需成品的詳細資訊，請參閱 使用人工因素。

運算

計算是指由 CodeCatalyst 管理和維護以執行工作流程動作的計算引擎 (CPU、記憶體和作業系統)。

如需有關計算的詳細資訊，請參閱使用計算。

環境

不要與開發環境混淆的環境是代碼部署到的地方。它通常包含一個正在運行的應用程序的實例及其相關的基礎設施。您可以為環境命名，例如開發、測試、測試或生產環境。對某個環境產生的 CodeCatalyst 任何部署都會顯示在「環境」頁面上。若要設定環境，請為其指定一個名稱，例如`my-production-environment`，然後將其與您的 AWS 帳戶.

除了顯示部署資訊之外，環境還可作為將 AWS IAM 角色指派給工作流程動作的機制。

如需環境的更多資訊，請參閱使用環境。

報告

報告包含工作流程執行期間所發生之測試的詳細資訊。您可以建立報告，例如測試報告、程式碼覆蓋率報告、軟體組成分析報告和靜態分析報告。您可以使用報告來協助疑難排解工作流程期間的問題。如果您有來自多個工作流程的許多報告，則可以使用報告來檢視趨勢和失敗率，以協助您最佳化應用程式和部署組態。

如需報告的詳細資訊，請參閱測試報告類型。

執行

執行是工作流程的單一版序。在執行期間， CodeCatalyst執行工作流程組態檔案中定義的動作，並輸出相關聯的記錄、人工因素和變數。

如需執行的更多資訊，請參閱使用梯段。

來源

來源 (也稱為輸入來源) 是工作流程動作需要存取才能執行其工作的來源儲存庫。例如，工作流程動作可能需要存取來源，才能取得單元測試，並針對應用程式來源檔案執行這些測試。

如需來源的詳細資訊，請參閱 使用來源。

Variables

變數是包含您可以在 CodeCatalyst 工作流程中參照的資訊的索引鍵值配對。

如需變數的更多資訊，請參閱使用變數。

工作流觸發

工作流程觸發器 (或只是觸發器) 可讓您在某些事件發生時 (例如程式碼推送) 自動啟動工作流程執行。您可能想要設定觸發程序，讓軟體開發人員不必透過 CodeCatalyst 主控台手動啟動工作流程執行。

您可以使用三種類型的觸發器：

- 推送 — 程式碼推送觸發程序會在推送提交時啟動工作流程執行。
- 提取請求 — 提取請求觸發程序會在建立、修訂或關閉提取請求時啟動工作流程執行。
- 「時間表」— 計划觸發器會使工作流程按照您定義的時間表開始運行。請考慮使用排程觸發程式來執行軟體的每晚組建，以便軟體開發人員可以在第二天早上進行工作。

您可以單獨或在相同的工作流程中組合使用推送、提取請求和排程觸發器。

關於觸發條件的詳細資訊，請參閱 使用觸發程序。

開始使用中的工作流程 CodeCatalyst

在本教程中，您將學習如何創建和配置您的第一個工作流程。

> **Tip**
>
> 偏好從預先設定的工作流程開始嗎？ 請參閱使用藍圖建立專案，其中包括使用正常運作的工作流程、範例應用程式和其他資源來設定專案的說明。

主題

- 必要條件
- 步驟 1：建立並設定您的工作流程
- 步驟 2：使用提交保存工作流程
- 步驟 3：檢視執行結果
- (選用) 步驟 4：清除

必要條件

開始之前：

- 你需要一個 CodeCatalyst 空間 如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。
- 在您的 CodeCatalyst 空間中，您需要一個空的，從頭開始的 CodeCatalyst 項目，名為：

```
codecatalyst-project
```

  如需詳細資訊，請參閱 在 Amazon 中創建一個空項目 CodeCatalyst。
- 在你的項目中，你需要一 CodeCatalyst 個名為：

```
codecatalyst-source-repository
```

  如需詳細資訊，請參閱 建立來源儲存庫。

> **Note**
>
> 如果您有現有的專案和來源儲存庫，您可以使用它們；不過，建立新的儲存庫可讓您在本自學課程結束時更容易進行清理。

步驟 1：建立並設定您的工作流程

在此步驟中，您會建立並設定工作流程，以便在進行變更時自動建置和測試原始程式碼。

若要建立工作流程

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇建立工作流程。

   工作流程定義檔案會出現在 CodeCatalyst 主控台的 YAML 編輯器中。

若要設定工作流程

您可以在視覺化編輯器或 YAML 編輯器中設定工作流程。讓我們從 YAML 編輯器開始，然後切換到可視化編輯器。

1. 選擇 [+ 動作] 以查看可新增至工作流程的工作流程動作清單。
2. 在「建置」動作中，選擇 +，將動作的 YAML 新增至工作流程定義檔案。您的工作流程現在看起來類似於以下內容。

```
Name: Workflow_fe47
SchemaVersion: "1.0"

# Optional - Set automatic triggers.
Triggers:
  - Type: Push
    Branches:
      - main

# Required - Define action configurations.
Actions:
  Build_f0:
    Identifier: aws/build@v1

    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this workflow as
 a source

    Outputs:
      AutoDiscoverReports:
        Enabled: true
        # Use as prefix for the report files
        ReportNamePrefix: rpt

    Configuration:
      Steps:
        - Run: echo "Hello, World!"
        - Run: echo "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" >> report.xml
        - Run: echo "<testsuite tests=\"1\" name=\"TestAgentJunit\" >" >>
 report.xml
        - Run: echo "<testcase classname=\"TestAgentJunit\" name=\"Dummy
            Test\"/></testsuite>" >> report.xml
```

工作流程會將WorkflowSource來源存放庫中的檔案複製到執行Build_f0動作的計算機器、列印Hello, World!至記錄檔、探索計算機器上的測試報告，然後將其輸出至 CodeCatalyst 主控台的 [報告] 頁面。

3. 選擇 [視覺] 以在視覺化編輯器中檢視工作流程定義檔案。視覺化編輯器中的欄位可讓您設定 YAML 編輯器中顯示的 YAML 屬性。

步驟 2：使用提交保存工作流程

在此步驟中，您會儲存變更。因為工作流程會以.yaml檔案形式儲存在儲存庫中，因此您可以透過認可儲存變更。

若要提交工作流程變更

1. (選擇性) 選擇 [驗證] 以確定工作流程的 YAML 程式碼有效。
2. 選擇 Commit (遞交)。
3. 在工作流程檔案名稱中，輸入工作流程組態檔案的名稱，例如**my-first-workflow**。
4. 在提交訊息中，輸入訊息來識別您的提交，例如**create my-first-workflow.yaml**。
5. 在存放庫中，選擇您要在 (codecatalyst-repository) 中儲存工作流程的存放庫。
6. 在「分支名稱」中，選擇要在 (main) 中儲存工作流程的分支。
7. 選擇 Commit (遞交)。

您的新工作流程會顯示在工作流程清單中。出現可能需要幾分鐘時間。

由於工作流程與認可一起儲存，而且由於工作流程已設定程式碼推送觸發程序，因此儲存工作流程會啟動自動執行工作流程。

步驟 3：檢視執行結果

在此步驟中，您會瀏覽至從提交開始的執行，並檢視結果。

若要檢視執行結果

1. 選擇工作流程的名稱，例如，Workflow_fe47。

   顯示源存儲庫（WorkflowSource）和構建操作（例如 build_ f0）的標籤的工作流程圖。
2. 在工作流程執行圖中，選擇建置動作 (例如 Build_F 0)。
3. 檢閱 [記錄檔]、[報告]、[組態] 和 [變數] 索引標籤的內容。這些索引標籤會顯示建置動作的結果。

   如需詳細資訊，請參閱 檢視建構動作的結果。

(選用) 步驟 4：清除

在此步驟中，您將清理在此自學課程中建立的資源。

若要刪除資源

1. 如果您為此自學課程建立了新專案，請將其刪除。如需說明，請參閱刪除 Amazon 中的項目 CodeCatalyst。刪除專案也會刪除來源儲存庫和工作流程。
2. 刪除尚未刪除的工作流程。如需說明，請參閱若要刪除工作流程。
3. 如果尚未刪除存放庫，請將其刪除。如需相關指示，請參閱刪除來源儲存庫。

透過提交檢視程式碼品質和部署狀態 CodeCatalyst

在開發生命週期的任何時候，請務必瞭解特定提交的部署狀態，例如錯誤修正、新功能或其他有影響力的變更。請考慮下列情況下，部署狀態追蹤功能對開發團隊很有幫助：

- 身為開發人員，您已修正錯誤，並且想要報告其在團隊部署環境中的部署狀態。
- 身為發行管理員，您想要檢視已部署的認可清單，以追蹤和報告其部署狀態。

CodeCatalyst 提供一個檢視，讓您一目了然地判斷個別認可或變更的部署位置，以及哪個環境。此檢視包括：

- 提交列表。
- 包含認可的部署狀態。
- 成功部署認可的環境。
- 任何測試的狀態都會針對 CI/CD 工作流程中的認可執行。

下列程序詳細說明如何瀏覽至此檢視並使用此檢視來追蹤專案中的變更。

> **Note**
>
> 只有CodeCatalyst 儲存庫才支援透過提交追蹤部署狀態。您無法將此功能與GitHub 存放庫搭配使用。

若要透過提交追蹤部署狀態

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在功能窗格中，選擇 CI/CD，然後選擇 [變更追蹤]。

4. 在主窗格頂端的兩個下拉式清單中，選擇包含您要檢視其核發狀態之確認的來源儲存庫與分支。
5. 選擇 [檢視變更]。

   提交清單隨即顯示。

   對於每次提交，您可以查看以下內容：

   - 提交資訊，例如 ID、作者、訊息，以及提交的時間。如需詳細資訊，請參閱 來源儲存庫 CodeCatalyst。
   - 每個環境的部署狀態。如需詳細資訊，請參閱 使用環境。
   - 測試和代碼覆蓋結果。如需詳細資訊，請參閱 使用工作流程測試 CodeCatalyst。

   > **Note**
   >
   > 不會顯示軟體組成分析 (SCA) 結果。

6. (選擇性) 若要檢視與特定認可相關變更的詳細資訊，包括最新部署、詳細的程式碼涵蓋範圍和單元測試資訊，請選擇檢視該認可的詳細資訊。

使用工作流程建置 CodeCatalyst

使用CodeCatalyst 工作流程，您可以建置應用程式和其他資源。

主題

- 如何建立應用程式？
- 建置動作的優點
- 建置動作的替代方案
- 添加構建操作
- 檢視建構動作的結果
- 教學課程：將成品上傳到 Amazon S3

如何建立應用程式？

若要在中建置應用程式或資源 CodeCatalyst，請先建立工作流程，然後在其中指定建置動作。

建置動作是工作流程建置區塊，可編譯您的原始程式碼、執行單元測試，以及產生準備好部署的成品。

您可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器，將建置動作新增至工作流程。

建置應用程式或資源的高階步驟如下。

若要建置應用程式 (高階工作)

1. 在中 CodeCatalyst，您可以為要建置的應用程式新增原始程式碼。如需詳細資訊，請參閱 使用中的來源儲存庫 CodeCatalyst。
2. 在中 CodeCatalyst，您可以建立工作流程。您可以在工作流程中定義如何建置、測試和部署應用程式。如需詳細資訊，請參閱 開始使用中的工作流程 CodeCatalyst。
3. (選擇性) 在工作流程中，您可以新增觸發器，指出將導致工作流程自動啟動的事件。如需更多資訊，請參閱使用觸發程序
4. 在工作流程中，您可以加入編譯和封裝應用程式或資源原始程式碼的建置動作。或者，如果您不想為這些目的使用測試或部署動作，也可以讓構建操作運行單元測試，生成報告並部署應用程序。如需測試和部署動作的詳細資訊，請參閱添加構建操作。
5. (選擇性) 在工作流程中，您可以新增測試動作和部署動作，以測試和部署應用程式或資源。您可以從數個預先設定的動作中進行選擇，將應用程式部署到不同的目標，例如 Amazon ECS。如需詳細資訊，請參閱 使用工作流程測試 CodeCatalyst 和 使用工作流程部署 CodeCatalyst。
6. 您可以手動或透過觸發器自動啟動工作流程。工作流程會依序執行建置、測試和部署動作，以建置、測試應用程式和資源並將其部署到目標。如需詳細資訊，請參閱 啟動工作流程執行。

建置動作的優點

在工作流程中使用建構動作具有下列優點：

- 完全受管 — 建置動作不需要設定、修補、更新和管理您自己的組建伺服器。
- 隨需 — 建置動作會隨需調整，以符合您的建置需求。您只需針對實際使用的組建分鐘數付費。如需詳細資訊，請參閱 使用計算。
- 開箱即用 — CodeCatalyst 包含預先封裝的執行階段環境 Docker 影像，這些影像可用來執行所有工作流程動作，包括建置動作。這些映像檔已預先設定好有用的工具來建置應用程式，例如 AWS CLI 和 Node.js。您可以設定 CodeCatalyst 為使用從公用或私人登錄提供的組建映像。如需詳細資訊，請參閱 使用執行階段環境 Docker 影像。

建置動作的替代方案

如果您使用建置動作來部署應用程式，請考慮改用 CodeCatalyst部署動作。如果您使用的是建 behind-the-scenes置動作，部署動作會執行設定，否則您必須手動寫入這些設定。如需可用部署動作的詳細資訊，請參閱部署動作清單。

您也可以使用 AWS CodeBuild 來建置您的應用程式。如需詳細資訊，請參閱什麼是 CodeBuild？ 。

主題

- 添加構建操作
- 檢視建構動作的結果
- 教學課程：將成品上傳到 Amazon S3

添加構建操作

請使用下列步驟將建構動作新增至您的 CodeCatalyst 工作流程。

Visual

若要使用視覺化編輯器新增建置動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
4. 選擇編輯。
5. 選擇 [視覺]。
6. 選擇動作。
7. 在 [動作] 中選擇 [建置]。
8. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱建立和測試動作參考。此參考提供每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增建置動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
4. 選擇編輯。
5. 選擇 YAML。
6. 選擇動作。
7. 在 [動作] 中選擇 [建置]。
8. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明建立和測試動作參考。
9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

建置動作定義

建置動作會定義為工作流程定義檔案中的一組 YAML 屬性。如需有關這些性質的資訊，請參閱建立和測試動作參考中的工作流定義參考。

檢視建構動作的結果

使用下列指示來檢視組建動作的結果，包括產生的記錄檔、報告和變數。

若要檢視建置動作的結果

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 在工作流程圖中，選擇建置動作的名稱，例如「建置」。
4. 若要檢視建置執行的記錄，請選擇 [記錄檔]。顯示各種構建階段的日誌。您可以視需要展開記錄檔。
5. 若要檢視建置動作所產生的測試報告，請選擇 [報告]，或在導覽窗格中選擇 [報告]。如需詳細資訊，請參閱 測試報告類型。

6. 若要檢視用於建置動作的組態，請選擇「組態」。如需詳細資訊，請參閱 添加構建操作。
7. 若要檢視建置動作所使用的變數，請選擇「變數」。如需更多詳細資訊，請參閱 使用變數。

教學課程：將成品上傳到 Amazon S3

在本教學中，您將學習如何使用包含幾個建置動作的 CodeCatalyst 工作流程將成品上傳到 Amazon S3 儲存貯體。當工作流程啟動時，這些動作會連續執行。第一個建置動作會產生兩個檔案`Goodbye.txt`，`Hello.txt`然後將它們捆綁到組建成品中。第二個建置動作會將成品上傳到 Amazon S3。您將配置工作流程，以便在每次將提交推送到源存儲庫時運行。

主題

必要條件

開始之前，您必須準備好以下事項：

- 您需要具有已連線 AWS 帳戶的 CodeCatalyst 空間。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。
- 在您的空間中，您需要一個空的，從頭開始的 CodeCatalyst項目，名為：

  ```
  codecatalyst-artifact-project
  ```

  如需詳細資訊，請參閱 在 Amazon 中創建一個空項目 CodeCatalyst。
- 在你的項目中，你需要一 CodeCatalyst 個名為：

  ```
  codecatalyst-artifact-environment
  ```

  設定此環境的方式如下：

- 選擇任何類型，例如開發。
- 將您的 AWS 帳戶 Connect 到它。

如需詳細資訊，請參閱 使用環境。

步驟 1：建立 AWS 角色

在此步驟中，您會建立 AWS IAM 角色，稍後將該角色指派給工作流程中的建置動作。此角色授予 CodeCatalyst 建立動作權限，以存取您的 AWS 帳戶並寫入存放您的成品的 Amazon S3。此角色稱為「建置」角色。

> **Note**
>
> 如果您已經有為其他教學課程建立的建置角色，也可以在本教學課程中使用它。只要確保它具有以下過程中顯示的權限和信任策略即可。

如需 IAM 角色的詳細資訊，請參閱AWS AWS Identity and Access Management 使用者指南中的 IAM 角色。

若要建立建置角色

1. 建立角色的策略，執行方式如下：
   a. 登入 AWS。
   b. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
   c. 在導覽窗格中，選擇政策。
   d. 選擇 Create policy (建立政策)。
   e. 請選擇 JSON 標籤。
   f. 刪除現有的程式碼。
   g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-s3-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋codecatalyst-s3-build-policy並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-s3-build-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst build role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

步驟 2：創建一個 Amazon S3 存儲桶

在此步驟中，您會建立 Amazon S3 儲存貯體，Hello.txt並在其中上傳和Goodbye.txt成品。

建立 Amazon S3 儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在主窗格中，選擇 [建立值區]。
3. 若為「值區名稱」，請輸入：

```
codecatalyst-artifact-bucket
```

4. 對於 AWS 區域，選擇一個區域。本教學課程假設您選擇美國西部 (奧勒岡) us-we st-2。如需 Amazon S3 支援區域的相關資訊，請參閱中的 Amazon 簡單儲存服務端點和配額AWS 一般參考。
5. 在頁面底部，選擇 [建立值區]。
6. 複製您剛建立的值區名稱，例如：

```
codecatalyst-artifact-bucket
```

您現在已經建立了一個名為美國西部 (奧勒岡) us-west-2 區**codecatalyst-artifact-bucket**域的值區。

步驟 3：建立來源儲存庫

在此步驟中，您可以在中建立來源儲存庫 CodeCatalyst。此儲存庫用於儲存自學課程的工作流程定義檔案。

如需來源儲存庫的詳細資訊，請參閱建立來源儲存庫。

若要建立來源儲存庫

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目，`codecatalyst-artifact-project`。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
5. 在存放庫名稱中，輸入：

```
codecatalyst-artifact-source-repository
```

6. 選擇建立。

現在，您已經創建了一個名為`codecatalyst-artifact-source-repository`.

步驟 4：建立工作流程

在此步驟中，您會建立由下列依序執行的建置區塊組成的工作流程：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。如需觸發器的詳細資訊，請參閱使用觸發程序。

- 名為 GenerateFiles — 觸發器的建置GenerateFiles動作會建立兩個檔案Goodbye.txt，Hello.txt然後將它們封裝到名為的輸出成品中codecatalystArtifact。
- 另一個名為的建置動作 Upload — 動作完成後，GenerateFiles動Upload作會執行 AWS CLI 命令，aws s3 sync將來源儲存庫中codecatalystArtifact和來源儲存庫中的檔案上傳到 Amazon S3 儲存貯體。 CodeCatalyst計算平台已預先安裝並預先設定，因此您不需要安裝或設定它。 AWS CLI

  如需有關 CodeCatalyst 運算平台上預先封裝軟體的詳細資訊，請參閱使用執行階段環境 Docker 影像。若要取得有關aws s3 sync指令 AWS CLI的更多資訊，請參閱《AWS CLI 指令參考》中的 sync。

如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。

若要建立工作流程

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇建立工作流程。
3. 刪除 YAML 範例程式碼。
4. 新增下列 YAML 程式碼：

```
Name: codecatalyst-artifact-workflow
SchemaVersion: 1.0

Triggers:
  - Type: Push
    Branches:
      - main
Actions:
  GenerateFiles:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        # Create the output files.
        - Run: echo "Hello, World!" > "Hello.txt"
        - Run: echo "Goodbye!" > "Goodbye.txt"
    Outputs:
      Artifacts:
        - Name: codecatalystArtifact
          Files:
            - "**/*"
```

```
  Upload:
    Identifier: aws/build@v1
    DependsOn:
      - GenerateFiles
    Environment:
      Name: codecatalyst-artifact-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-s3-build-role
    Inputs:
      Artifacts:
        - codecatalystArtifact
    Configuration:
      Steps:
        # Upload the output artifact to the S3 bucket.
        - Run: aws s3 sync . s3://codecatalyst-artifact-bucket
```

在上面的代碼中，替換：

- *codecatalyst-artifact-environment*使用您在中建立的環境名稱必要條件。
- *codecatalyst-account-connection*使用您在中創建的帳戶連接的名稱必要條件。
- *####-s3 ##################*的名稱。步驟 1：建立 AWS 角色
- *codecatalyst-artifact-bucket*使用您在其中創建的 Amazon S3 的名稱步驟 2：創建一個 Amazon S3 存儲桶。

如需有關此檔案中性質的資訊，請參閱建立和測試動作參考。

5. (選擇性) 選擇 [驗證]，確定 YAML 程式碼在認可之前是有效的。
6. 選擇 Commit (遞交)。
7. 在「提交工作流程」對話方塊中，輸入以下內容：

   a. 對於「工作流程」檔案名稱，請保留預設值codecatalyst-artifact-workflow。

   b. 對於提交訊息，請輸入：

   ```
   add initial workflow file
   ```

   c. 針對「儲存庫」，選擇codecatalyst-artifact-source-repository。

   d. 選擇「主要」做為「分支名稱」。

   e. 選擇 Commit (遞交)。

您現在已建立工作流程。工作流程執行會自動啟動，因為工作流程頂端定義了觸發器。具體而言，當您認可 (並推送) codecatalyst-artifact-workflow.yaml 檔案至來源儲存庫時，觸發程序會啟動工作流程執行。

若要檢視進行中的工作流程執行

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇您剛建立的工作流程：codecatalyst-artifact-workflow。
3. 選擇GenerateFiles查看第一個建置動作進度。
4. 選擇 [上傳] 以查看第二個建置動作進度。
5. 當「上載」動作完成時，請執行下列動作：
   - 如果工作流程執行成功，請移至下一個程序。
   - 如果工作流程執行失敗，請選擇 [記錄檔] 以疑難排解問題。

步驟 5：驗證結果

工作流程執行後，前往 Amazon S3 服務並查看您的儲存*codecatalyst-artifact-bucket*貯體。它現在應該包括以下文件和文件夾：

```
.
|— .aws/
|— .git/
|Goodbye.txt
|Hello.txt
|REAME.md
```

Goodbye.txt和Hello.txt檔案已上傳，因為它們是codecatalystArtifact成品的一部分。.aws/.git/、和README.md檔案已上傳，因為它們位於您的來源儲存庫中。

清除

清理 CodeCatalyst 並 AWS 避免被收取這些服務費用。

若要在中進行清理 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 刪除來codecatalyst-artifact-source-repository源儲存庫。
3. 刪除codecatalyst-artifact-workflow工作流程。

若要在中進行清理 AWS

1. 在 Amazon S3 中進行清理，如下所示：
   a. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
   b. 刪除codecatalyst-artifact-bucket值區中的檔案。
   c. 刪除codecatalyst-artifact-bucket值區。
2. 在 IAM 中進行清理，如下所示：
   a. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
   b. 刪除codecatalyst-s3-build-policy。
   c. 刪除codecatalyst-s3-build-role。

使用工作流程測試 CodeCatalyst

在中 CodeCatalyst，您可以執行測試作為不同工作流程動作 (例如建置和測試) 的一部分。這些工作流程動作都可以產生報告。測試動作是產生測試、程式碼涵蓋範圍、軟體組成分析和靜態分析報告的工作流程動作。這些報告會顯示在主 CodeCatalyst 控台中。

主題

測試報告類型

Amazon CodeCatalyst 測試動作支援以下類型的報告。如需如何在 YAML 中格式化這些報表的範例，請參閱範例：設定報表。

主題

- 測試報告
- 代碼覆蓋率報告
- 軟件成分分析報告
- 靜態分析報告

測試報告

在中 CodeCatalyst，您可以設定在建置期間執行的單元測試、整合測試和系統測試。然後 CodeCatalyst 可以創建包含測試結果的報告。

您可以使用測試報告來協助疑難排解測試問題。如果您有許多來自多個組建的測試報告，則可以使用測試報告來檢視失敗率，以協助您最佳化組建。

您可以使用下列測試報告檔案格式：

- 黃瓜
- 六月 XML（的 .xml）
- 單位 XML（.xml）
- nUnit3 XML (.xml)
- 。TestNG。。XML。
- 視覺工作室 TRX（.trx，.xml）

代碼覆蓋率報告

在中 CodeCatalyst，您可以為測試產生程式碼涵蓋率報告。 CodeCatalyst 提供下列程式碼涵蓋率指標：

線路覆蓋

衡量您的測試涵蓋了多少陳述。聲明是一個單一的指令，不包括註釋。

```
line coverage = (total lines covered)/(total number of lines)
```

分公司覆蓋

測量您的測試涵蓋了控制結構（例如`if`或`case`語句）的每個可能分支的分支的數量。

```
branch coverage = (total branches covered)/(total number of branches)
```

支援下列程式碼涵蓋範圍報表檔案格式：

- JaCoCo XML
- SimpleCov JSON（由簡單生成的，而不是簡單的 JSON，.json）
- 三葉草 XML（版本 3，.xml）
- XML 編輯器（的 .xml）
- 甲型冠狀病毒 (. 資訊)

軟件成分分析報告

在中 CodeCatalyst，您可以使用軟體組成分析 (SCA) 工具來分析應用程式的元件，並檢查是否有已知的安全性弱點。您可以發現和解析 SARIF 報告，其中詳細說明具有不同嚴重性的漏洞以及修復方法。有效嚴重性值 (從最大到最不嚴重) 為：`CRITICAL`、`HIGH`、、`MEDIUM`、`LOW`、、`INFORMATIONAL`。

支援下列 SCA 報告檔案格式：

- 紗麗夫 (. 沙里夫,. JSON)

靜態分析報告

您可以使用靜態分析 (SA) 報告來識別來源層級的程式碼瑕疵。在中 CodeCatalyst，您可以產生 SA 報告，以協助解決程式碼中的問題，然後再部署它。這些問題包括錯誤，安全漏洞，質量問題和其他漏洞。有效嚴重性值 (從最大到最不嚴重) 為：`CRITICALHIGH`、`MEDIUM`、`LOW`、、和`INFORMATIONAL`。

CodeCatalyst 提供下列 SA 測量結果：

臭蟲

識別源代碼中發現的一些可能的錯誤。這些錯誤可能包括有關內存安全性的問題。以下是一個錯誤的例子。

```
// The while loop will inadvertently index into array x out-of-bounds
int x[64];
while (int n = 0; n <= 64; n++) {
  x[n] = 0;
}
```

安全漏洞

識別在原始程式碼中發現的數個可能的安全性弱點。這些安全漏洞可能包括諸如以純文本形式存儲密令牌之類的問題。

品質問題

識別原始程式碼中可能發現的許多品質問題。這些品質問題可能包括有關樣式慣例的問題。以下是品質問題的範例。

```
// The function name doesn't adhere to the style convention of camelCase
int SUBTRACT(int x, int y) {
  return x-y
}
```

其他漏洞

識別原始程式碼中可能發現的許多其他弱點。

CodeCatalyst 支援下列 SA 報告檔案格式：

- PyLint (.py)
- 埃斯林特（.js，.jsx，.ts，.tsx）
- 紗麗夫 (. 沙里夫,. JSON)

新增測試動作

使用下列程序將測試動作新增至您的工作流程。

若要新增測試動作

- 請遵循中的說明進行添加構建操作 添加構建操作和測試操作的說明完全相同。

測試動作定義

測試動作會定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱建立和測試動作參考中的工作流定義參考。

配置報告

本節介紹如何配置質量報告。

主題

- 設定自動探索和手動報告
- 設定報告的成功準則
- 範例：設定報表
- SARIF 支持軟件組成分析和靜態分析報告

設定自動探索和手動報告

啟用自動探索後，會 CodeCatalyst 搜尋傳入動作的所有輸入，以及動作本身產生的所有檔案，尋找測試、程式碼涵蓋範圍、軟體組成分析 (SCA) 和靜態分析 (SA) 報告。您可以在中檢視和操作這些報告 CodeCatalyst。

您也可以手動設定要產生哪些報告。您可以指定要產生的報告類型以及檔案格式。如需詳細資訊，請參閱 測試報告類型。

設定報告的成功準則

您可以設定決定測試、程式碼涵蓋範圍、軟體組成分析 (SCA) 或靜態分析 (SA) 報告的成功準則的值。

成功準則是決定報告是否通過或失敗的臨界值。 CodeCatalyst 首先生成您的報告，該報告可以是測試，代碼覆蓋率，SCA 或 SA 報告，然後將成功標準應用於生成的報告。然後，它會顯示是否符合成功準則，以及達到何種程度。如果有任何報告不符合指定的成功條件，則指定成功準則的 CodeCatalyst 動作會失敗。

例如，當您設定 SCA 報告的成功準則時，有效弱點值範圍從最大到最不嚴重程度為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。如果您將條件設定為掃瞄一個HIGH嚴重性弱點，則報告將會失敗，如果至少有一個嚴重性弱點或沒有HIGH嚴重性弱點，但至少有一個較高HIGH嚴重性等級的弱點，例如一個嚴重性弱點，則報告將會失敗。CRITICAL

如果您未指定成功準則，則：

- 根據原始 CodeCatalyst 報表產生的報告將不會顯示成功條件。
- 成功條件不會用來決定相關聯的工作流程動作是否通過或失敗。

Visual

若要設定成功準則

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇包含產生報告之動作的工作流程。這是您要套用成功條件的報表。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 選擇編輯。
4. 選擇 [視覺]。
5. 在工作流程圖表中，選擇您已設定為產生 CodeCatalyst 報告的動作。
6. 選擇 Output (輸出) 索引標籤。
7. 在「自動探索報告」下或「手動設定報告」下，選擇「成功準則」

   成功標準隨即出現。視您之前的選擇而定，您可能會看到下列任一或所有選項：

   合格率

   指定測試報告中必須通過的測試百分比，這些測試百分比才能標記為通過的關聯 CodeCatalyst 報告。有效值包括十進位數字。例如：50、60.5。合格率標準僅適用於測試報告。如需測試報告的詳細資訊，請參閱測試報告。

   線路覆蓋

   指定程式碼涵蓋範圍報告中必須涵蓋的行百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。明細行涵蓋範圍條件僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

   分公司覆蓋

   指定程式碼涵蓋範圍報告中必須涵蓋的分支百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。分支涵蓋範圍標準僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

   漏洞 (SCA)

針對要標記為通過的相關 CodeCatalyst 報告，指定 SCA 報告中允許的弱點數目上限和嚴重性。若要指定弱點，您必須指定：

- 您要包含在計數中的弱點的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL漏洞將被統計。
- 您要允許之指定嚴重性的弱點數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

弱點準則僅適用於 SCA 報告。如需有關 SCA 報告的詳細資訊，請參閱軟件成分分析報告。

臭蟲

針對要標記為通過的相關 CodeCatalyst 報告，指定 SA 報告中允許的最大錯誤數目和嚴重性。若要指定錯誤，您必須指定：

- 您要包含在計數中的錯誤的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL錯誤將被計算。
- 您要允許的指定嚴重性的最大錯誤數目。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

錯誤條件僅適用於 PyLint 和 eSlint SA 報告。如需 SA 報告的詳細資訊，請參閱靜態分析報告。

安全漏洞

指定 SA 報告中允許的相關 CodeCatalyst 報告標示為通過的安全性弱點數目上限和嚴重性。若要指定安全性弱點，您必須指定：

- 您要包含在計數中的安全性弱點的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL安全漏洞將被結算。
- 您要允許之指定嚴重性的最大安全性弱點數目。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

安全性弱點條件僅適用於 PyLint 和 eSlint SA 報告。如需 SA 報告的詳細資訊，請參閱靜態分析報告。

品質問題

指定 SA 報告中允許將相關 CodeCatalyst 報告標示為通過的品質問題的最大數目和嚴重性。要指定質量問題，您必須指定：

- 您要包含在計數中的品質問題的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL質量問題將被結算。
- 您要允許之指定嚴重性的品質問題數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

「質量問題」條件僅適用于「eSlint SA」報表。 PyLint 如需 SA 報告的詳細資訊，請參閱靜態分析報告。

8. 選擇 Commit (遞交)。
9. 執行工作流程，將成功條件 CodeCatalyst 套用至您的原始報表，並重新產生包含成功條件資訊的相關聯 CodeCatalyst 報表。如需詳細資訊，請參閱 啟動工作流程執行。

YAML

若要設定成功準則

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇包含產生報告之動作的工作流程。這是您要套用成功條件的報表。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 選擇編輯。
4. 選擇 YAML。
5. 在工作流程圖表中，選擇您已設定為產生 CodeCatalyst 報告的動作。
6. 在詳細資料窗格中，選擇 [輸出] 索引標籤。
7. 在動作、AutoDiscoverReports區段或區Reports段中，加入SuccessCriteria屬性，以及PassRate、LineCoverage、BranchCoverage、Vulnerabilities、StaticAnalysisBug和StaticAnalysisQuality屬性。

如需每個屬性的說明，請參閱建立和測試動作參考。

8. 選擇 Commit (遞交)。
9. 執行工作流程，將成功條件 CodeCatalyst 套用至原始報表，並重新產生含有成功條件資訊的相關聯 CodeCatalyst 報表。如需啟動工作流程的詳細資訊，請參閱啟動工作流程執行。

範例：設定報表

下列範例顯示如何手動設定四個報告：測試報告、程式碼涵蓋範圍報告、軟體組成分析報告，以及靜態分析報告。

```
Reports:
  MyTestReport:
    Format: JUNITXML
    IncludePaths:
      - "*.xml"
    ExcludePaths:
      - report1.xml
      SuccessCriteria:
        PassRate: 90
  MyCoverageReport:
    Format: CLOVERXML
    IncludePaths:
      - output/coverage/jest/clover.xml
      SuccessCriteria:
        LineCoverage: 75
        BranchCoverage: 75
  MySCAReport:
    Format: SARIFSCA
    IncludePaths:
      - output/sca/reports.xml
      SuccessCriteria:
        Vulnerabilities:
          Number: 5
          Severity: HIGH
  MySAReport:
    Format: ESLINTJSON
    IncludePaths:
      - output/static/eslint.xml
      SuccessCriteria:
        StaticAnalysisBug:
          Number: 10
```

```
          Severity: MEDIUM
        StaticAnalysisSecurity:
          Number: 5
          Severity: CRITICAL
        StaticAnalysisQuality:
          Number: 0
          Severity: INFORMATIONAL
```

SARIF 支持軟件組成分析和靜態分析報告

SARIF (靜態分析結果交換格式) 是一種輸出檔案格式，可在中的軟體組成分析和靜態分析報告中 CodeCatalyst使用。下列範例顯示如何在靜態分析報表中手動設定 SARIF：

```
Reports:
  MySAReport:
    Format: SARIFSA
    IncludePaths:
      - output/sa_report.json
    SuccessCriteria:
      StaticAnalysisFinding:
        Number: 25
        Severity: HIGH
```

CodeCatalyst 支援下列 SARIF 屬性，這些屬性可用於最佳化分析結果在報表中的顯示方式。

主題

- sarifLog 物件
- run 物件
- toolComponent 物件
- reportingDescriptor 物件
- result 物件
- location 物件
- physicalLocation 物件
- logicalLocation 物件
- fix 物件

`sarifLog` 物件

名稱	必要	描述
`$schema`	是	版本 2. 1.0 的 SARIF JSON 結構描述的 URI。
`version`	是	CodeCatalyst 僅支持 SARIF 版本 2.1.0。
`runs[]`	是	SARIF 檔案包含一個或多個執行的陣列，每個執行都代表分析工具的單一執行。

`run` 物件

名稱	必要	描述
`tool.driver`	是	描述分析工具的`toolComponent` 物件。
`tool.name`	否	指示用於執行分析之工具名稱的性質。
`results[]`	是	顯示在上的分析工具的結果 CodeCatalyst。

`toolComponent` 物件

名稱	必要	描述
`name`	是	分析工具的名稱。
`properties.artifactScanned`	否	工具分析的成品總數。

名稱	必要	描述
rules[]	是	代表規則的reporting Descriptor 物件陣列。根據這些規則，分析工具會在分析的程式碼中尋找問題。

reportingDescriptor 物件

名稱	必要	描述
id	是	用來參照發現項目之規則的唯一識別碼。 長度上限：1 個字元
name	否	規則的顯示名稱。 長度上限：1 個字元
shortDescription.text	否	規則的簡短描述。 長度上限：3,000 個字元
fullDescription.text	否	規則的完整描述。 長度上限：3,000 個字元
helpUri	否	可當地語系化以包含規則主要文件之絕對 URI 的字串。 長度上限：3,000 個字元
properties.unscore	否	指出掃描發現項目是否已評分的旗標。
properties.score.severity	否	指定發現項目嚴重性層級的固定字串集。

名稱	必要	描述
		長度上限：1 個字元
properties.cvssv3_baseSeverity	否	常見弱點評分系統 v 3.1 的定性嚴重性等級。
properties.cvssv3_baseScore	否	基本分數介於 0.0 至 10. 0 之間。
properties.cvssv2_severity	否	如果無法使用 CVSS v3 值，則會 CodeCatalyst 搜尋 CVSS V2 值。
properties.cvssv2_score	否	一個 CVSS V2 的基本分數範圍為 0.0-10. 0。
properties.severity	否	指定發現項目嚴重性層級的固定字串集。 長度上限：1 個字元
defaultConfiguration.level	否	規則的預設嚴重性。

result 物件

名稱	必要	描述
ruleId	是	用來參照發現項目之規則的唯一識別碼。 長度上限：1 個字元
ruleIndex	是	工具元件中關聯規則的索引rules[]。
message.text	是	描述結果並顯示每個發現項目之訊息的訊息。

名稱	必要	描述
		長度上限：3,000 個字元
rank	否	介於 0.0 到 100.0 之間的值，代表結果的優先順序或重要性。此刻度值 0.0 是最低優先級，100.0 是最高優先級。
level	否	結果的嚴重性。 長度上限：1 個字元
properties.unscore	否	指出掃描發現項目是否已評分的旗標。
properties.score.severity	否	指定發現項目嚴重性層級的固定字串集。 長度上限：1 個字元
properties.cvssv3_baseSeverity	否	常見弱點評分系統 v 3.1 的定性嚴重性等級。
properties.cvssv3_baseScore	否	基本分數介於 0.0 至 10. 0 之間。
properties.cvssv2_severity	否	如果無法使用 CVSS v3 值，則會 CodeCatalyst 搜尋 CVSS V2 值。
properties.cvssv2_score	否	一個 CVSS V2 的基本分數範圍為 0.0-10. 0。
properties.severity	否	指定發現項目嚴重性層級的固定字串集。 長度上限：1 個字元

名稱	必要	描述
locations[]	是	偵測到結果的位置集。除非只能透過在每個指定位置進行變更來修正問題，否則只能包含一個位置。 CodeCatalyst 使用位置陣列中的第一個值來註解結果。 最大對location象數：10
relatedLocations[]	否	尋找項目中其他位置參照的清單。 最大location對象數量：50
fixes[]	否	fix物件陣列，代表掃描工具提供的建議。 CodeCatalyst 使用fixes陣列中的第一個建議。

location 物件

名稱	必要	描述
physicalLocation	是	識別人工因素和區域。
logicalLocations[]	否	依名稱描述而不參照人工因素的位置集。

physicalLocation 物件

名稱	必要	描述
artifactLocation.uri	是	指示加工品位置的 URI，通常是存放庫中或在組建期間產生的檔案。

名稱	必要	描述
fileLocation.uri	否	指示檔案位置的後退 URI。如果artifactLocation.uri 返回空，則使用此選項。
region.startLine	是	區域中第一個字元的行號。
region.startColumn	是	區域中第一個字元的欄號。
region.endLine	是	區域中最後一個字元的行號。
region.endColumn	是	區域中最後一個字元的欄號。

logicalLocation 物件

名稱	必要	描述
fullyQualifiedName	否	描述結果位置的其他資訊。 長度上限：1 個字元

fix 物件

名稱	必要	描述
description.text	否	顯示每個發現項目之建議的訊息。 長度上限：3,000 個字元
artifactChanges.[0].artifactLocation.uri	否	URI 指出需要更新的加工品位置。

使用 universal-test-runner

測試動作與開放原始碼命令列工具整合universal-test-runner。該工具提供高級測試功能，例如從測試報告中重試一個或多個測試用例。 universal-test-runner使用測試執行協議在給定框架中為任何語言運行測試。 universal-test-runner支持以下框架：

- 搖籃
- 開玩笑
- Maven
- 最炎熱的
- .NET

universal-test-runner僅安裝在用於測試動作的策劃圖像上。如果您將測試動作設定為使用自訂 Docker Hub 或 Amazon ECR，則必須手動安裝universal-test-runner才能啟用進階測試功能。若要這麼做，請在映像檔上安裝 Node.js (14 或更高版本)，然後npm使用殼層指令universal-test-runner進行安裝- Run: npm install -g @aws/universal-test-runner。如需有關透過 shell 命令在容器中安裝 Node.js 的詳細資訊，請參閱安裝及更新節點版本管理員。

如需有關的詳細資訊universal-test-runner，請參閱什麼是 universal-test-runner？

Visual

若要 universal-test-runner 在視覺化編輯器中使用

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。
4. 選擇編輯。
5. 選擇 [視覺]。
6. 選擇動作。
7. 在 [動作] 中選擇 [測試]。
8. 在 [組態] 索引標籤上，使用您選擇的支援架構更新範例程式碼，以完成 [命令介面命令] 欄位。例如，若要使用支援的架構，您可以使用類似下列的Run命令。

```
- Run: run-tests <framework>
```

如果不支援您想要的架構，請考慮提供自訂轉接器或執行器。如需「命令介面」指令欄位的說明，請參閱Steps。

9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要 universal-test-runner 在 YAML 編輯器中使用

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。
4. 選擇編輯。
5. 選擇 YAML。
6. 選擇動作。
7. 在 [動作] 中選擇 [測試]。
8. 根據您的需要修改 YAML 代碼。例如，若要使用支援的架構，您可以使用類似下列的Run命令。

```
Configuration:
  Steps:
    - Run: run-tests <framework>
```

如果不支援您想要的架構，請考慮提供自訂轉接器或執行器。如需「步驟」性質的描述，請參閱Steps。

9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

最佳實務

使用提供的測試功能時 CodeCatalyst，建議您遵循這些最佳做法。

主題

- 自動探索

- 成功條件
- 包含/排除路徑

自動探索

在中設定動作時 CodeCatalyst，自動探索可讓您自動探索各種工具的輸出，例如 JUnit 測試報告，並從中產生相關 CodeCatalyst報告。自動探索有助於確保即使探索到的輸出的名稱或路徑有所變更，仍可繼續產生報告。加入新檔案時， CodeCatalyst會自動探索這些檔案並產生相關報告。但是，如果您使用自動探索，請務必考慮此功能的下列某些方面：

- 當您在動作中啟用自動探索時，所有自動探索相同類型的報告都會共用相同的成功準則。例如，共用準則 (例如最低通過率) 會套用至所有自動探索的測試報告。如果相同類型的報告需要不同的準則，則必須明確配置每個報告。
- 自動探索也可以尋找相依性所產生的報告，如果已設定成功準則，可能會在這些報告上執行動作失敗。更新排除路徑組態可以解決此問題。
- 不保證自動探索每次都會產生相同的報告清單，因為它會在執行階段掃描動作。如果您希望始終產生特定報告，則應明確配置報告。例如，如果測試停止作為構建的一部分運行，則測試框架將不會產生任何輸出，因此不會生成測試報告，並且操作可能會成功。如果您希望動作的成功取決於該特定測試，則必須明確配置該報告。

> **Tip**
>
> 開始使用新專案或現有專案時，請對整個專案目錄 (包括**/*) 使用自動探索。這會呼叫專案中所有檔案的報表產生，包括子目錄中的檔案。

如需詳細資訊，請參閱 配置報告。

成功條件

您可以設定成功準則，在報表上強制執行品質閾值。例如，如果自動探索兩份程式碼涵蓋範圍報告，其中一個行覆蓋率為 80%，另一個行覆蓋率為 60%，則您可以選擇下列選項：

- 將線路涵蓋範圍的自動探索成功準則設定為 80%。這會導致第一份報告通過，而第二個報告失敗，這將導致整體動作失敗。要解除阻止工作流程，請向項目添加新測試，直到第二個報告的線路覆蓋率超過 80%。

- 將線路涵蓋範圍的自動探索成功準則設定為 60%。這將導致兩個報告通過，這將導致動作成功。然後，您可以在第二個報告中增加代碼覆蓋率。但是，使用這種方法，您無法保證第一份報告中的涵蓋範圍不會降至 80% 以下。
- 使用視覺化編輯器或為每個報表新增明確的 YAML 區段和路徑，明確設定一個或兩個報表。這將允許您為每個報告配置單獨的成功條件和自定義名稱。但是，使用這種方法，如果報告路徑變更，動作可能會失敗。

如需詳細資訊，請參閱 設定報告的成功準則。

包含/排除路徑

檢閱動作結果時，您可以透過配置`IncludePaths`和來調整所 CodeCatalyst 產生的報告清單`ExcludePaths`。

- 用`IncludePaths`於指定搜尋報告時 CodeCatalyst 要包括的檔案和檔案路徑。例如，如果您指定`"/test/report/*"`，會 CodeCatalyst 搜尋尋找`/test/report/`目錄的動作所使用的整個組建映像。當它找到該目錄時， CodeCatalyst 會在該目錄中尋找報表。

  > **Note**
  >
  > 對於手動設定的報告，`IncludePaths`必須是符合單一檔案的 glob 模式。

- 用`ExcludePaths`於指定搜尋報告時 CodeCatalyst 要排除的檔案和檔案路徑。例如，如果您指定`"/test/reports/**/*"`， CodeCatalyst將不會搜尋`/test/reports/`目錄中的檔案。要忽略目錄中的所有文件，請使用 `**/*` glob 模式。

以下是可能的全域模式的示例。

模式	描述
`*.*`	匹配當前目錄中包含點的所有對象名稱
`*.xml`	符合目前目錄中以結尾的所有物件名稱 `.xml`
`*.{xml,txt}`	符合目前目錄中以`.xml`或結尾的所有物件名稱 `.txt`
`**/*.xml`	在所有以下結尾的目錄中匹配對象名稱 `.xml`

模式	描述
testFolder	匹配一個名為的對象testFolder ，將其視為一個文件
testFolder/*	符合子資料夾中某個層級中的物件testFolder ，例如 testFolder/file.xml
testFolder/*/*	符合子資料夾中兩個層級的物件testFolder ，例如 testFolder/reportsFolder/file.xml
testFolder/**	符合資料夾 testFolder 以及 testFolder 下的檔案，例如 testFolder/file.xml 和 testFolder/otherFolder/file.xml

CodeCatalyst 解釋 glob 模式如下：

- 斜線 (/) 字元可分隔檔案路徑中的目錄。
- 星號 (*) 字元符合不超過資料夾邊界之名稱元件的零個或多個字元。
- 雙星號 (**) 會在所有目錄中比對名稱元件的零個或多個字元。

> **Note**
>
> ExcludePaths優先順序高於IncludePaths。如果同時IncludePathsExcludePaths包含相同的資料夾，則不會掃描該資料夾中的報告。

使用測試

本節說明如何檢視和設定測試動作。

主題

- 檢視測試動作的結果
- 跳過失敗的測試
- 重試測試用例

檢視測試動作的結果

使用下列指示來檢視測試動作的結果，包括產生的記錄檔、報告和變數。

若要檢視測試動作的結果

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 在工作流程圖中，選擇測試動作的名稱（例如「測試」）。
4. 如果要檢視處理行動產生的記錄檔，請選擇「記錄檔」。隨即顯示各種動作階段的記錄檔。您可以視需要展開或收合記錄檔。
5. 若要檢視測試動作所產生的測試報告，請選擇 [報告]。如需詳細資訊，請參閱 測試報告類型。
6. 若要檢視用於測試動作的組態，請選擇組態。如需詳細資訊，請參閱 新增測試動作。
7. 若要檢視測試動作所使用的變數，請選擇「變數」。如需更多詳細資訊，請參閱 使用變數。

跳過失敗的測試

如果您的動作有多個測試命令，則即使先前的命令失敗，您也可能希望允許動作中的後續測試命令運行。例如，在以下命令中，即使`test1`失敗，您也可能希望`test2`始終運行。

```
Steps:
  - Run: npm install
  - Run: npm run test1
  - Run: npm run test2
```

通常，當步驟返回錯誤時，Amazon CodeCatalyst 會停止工作流程操作並將其標記為失敗。您可以將錯誤輸出重新導向至，以允許動作步驟繼續執行。`null`您可以透過新增`2>/dev/null`至指令來執行此操作。透過此修改，上述範例如下所示。

```
Steps:
  - Run: npm install
  - Run: npm run test1 2>/dev/null
  - Run: npm run test2
```

在第二個代碼片段中，`npm install`命令的狀態將被遵守，但該命`npm run test1`令返回的任何錯誤都將被忽略。因此，該`npm run test2`命令被運行。通過這樣做，無論是否發生錯誤，您都可以一次查看這兩個報告。

重試測試用例

如果您的報告因為多個測試用例而失敗，則只能重試這些單獨的測試。這使您可以快速檢查測試用例的質量，並確定解決問題的後續步驟，例如執行損壞的依賴關係或啟動工作流程重新運行。您的測試操作包含僅重universal-test-runner試選定的測試用例，而不是整個動作。您一次只能重試一組選定的測試用例，每個動作只能重試五次。如需詳細資訊，請參閱 使用 universal-test-runner。

> **Note**
>
> 如果您在報表上重試測試案例，則不會影響產生原始報告的工作流程狀態。

請使用下列指示重試報表中的測試案例。

若要重試報告的測試案例

1. 在導覽窗格中，選擇 Reports (報告)。
2. 選擇報告的名稱。您可以依名稱、狀態、存放庫、分支或報表類型進行篩選。
3. 在報告名稱下，選擇「結果」。
4. 選取您要重試的測試案例，選擇 [重新執行]，然後選擇 [選取的測試案例]。
5. 重試完成後，請選擇橫幅上的 [重新整理]，然後檢視更新的結果。

使用工作流程部署 CodeCatalyst

您可以使用CodeCatalyst 工作流程將應用程式和其他資源部署到各種目標 AWS Lambda，例如 Amazon ECS 等。

主題

- 如何部署應用程式？
- 部署動作清單
- 部署動作的好處
- 部署動作的替代方案
- 教學課程：使用部署無伺服器應用程式 AWS CloudFormation
- 教學課程：將應用程式部署到 Amazon ECS
- 教學課程：將應用程式部署到 Amazon EKS

- 新增「部署 AWS CloudFormation 堆疊」動作
- 新增「部署到 Amazon ECS」動作
- 新增「部署至 Kubernetes 叢集」動作
- 新增「AWS CDK 部署」動作
- 使用部署

如何部署應用程式？

若要透過部署應用程式或資源 CodeCatalyst，請先建立工作流程，然後在其中指定部署動作。部署動作是工作流程建置區塊，可定義您要部署的項目、部署位置以及部署方式 (例如，使用藍色/綠色配置)。您可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器，將部署動作新增至工作流程。

部署應用程式或資源的高階步驟如下。

若要部署應用程式 (高階工作)

1. 在 CodeCatalyst 專案中，您可以為要部署的應用程式加入原始程式碼。如需詳細資訊，請參閱 使用中的來源儲存庫 CodeCatalyst。
2. 在 CodeCatalyst 專案中，您可以建立工作流程。您可以在工作流程中定義如何建置、測試和部署應用程式。如需詳細資訊，請參閱 開始使用中的工作流程 CodeCatalyst。
3. 在工作流程中，您可以新增觸發程序、建置動作，以及選擇性地加入測試動作。如需詳細資訊，請參閱使用觸發程序、添加構建操作及新增測試動作。
4. 在工作流程中，您可以新增部署動作。您可以選擇多種針對應用程式 CodeCatalyst提供的部署動作到不同的目標，例如 Amazon ECS。您也可以使用建置動作或動 GitHub 作來部署應用程式。如需有關建構動 GitHub 作與動作的詳細資訊，請參閱部署動作的替代方案。)
5. 您可以手動或透過觸發器自動啟動工作流程。工作流程會依序執行建置、測試和部署動作，以將應用程式和資源部署到目標。如需詳細資訊，請參閱 啟動工作流程執行。

部署動作清單

下列是可用的部署動作：

- 部署 AWS CloudFormation 堆疊 — 此動作會 AWS 根據您提供的AWS CloudFormation 範本或AWS Serverless Application Model 範本在中建立 CloudFormation 堆疊。如需詳細資訊，請參閱 新增「部署 AWS CloudFormation 堆疊」動作。

- 部署到 Amazon ECS — 此動作會註冊您提供的任務定義檔案。如需詳細資訊，請參閱 新增「部署到 Amazon ECS」動作。
- 部署到 Kubernetes 叢集 — 此動作會將應用程式部署到 Amazon Elastic Kubernetes Service 叢集。如需詳細資訊，請參閱 新增「部署至 Kubernetes 叢集」動作。
- AWS CDK 部署 — 此動作會將AWS CDK 應用程式部署至 AWS。如需詳細資訊，請參閱 新增「AWS CDK 部署」動作。

> **Note**
>
> 還有其他 CodeCatalyst 動作可以部署資源；不過，這些動作不會被視為部署動作，因為它們的部署資訊不會顯示在 [環境] 頁面上。若要深入了解「環境」頁面和檢視建置，請參閱使用環境和使用部署。

部署動作的好處

在工作流程中使用部署動作具有下列優點：

- 部署歷史記錄 — 檢視部署歷史記錄，以協助管理和溝通已部署軟體中的變更。
- 可追蹤性 — 透過 CodeCatalyst 主控台追蹤部署狀態，並查看每個應用程式修訂版的部署時間和位置。
- 復原 — 如果發生錯誤，則會自動復原部署。您也可以設定警示以啟動部署復原。
- 監控 — 觀察部署在工作流程的各個階段進行時。
- 與其他 CodeCatalyst 功能集成 — 存儲源代碼，然後構建，測試和部署它，所有這些都從一個應用程序。

部署動作的替代方案

您不必使用部署動作，雖然建議您使用這些動作，因為這些動作提供了前一節所述的優點。相反地，您可以使用下列CodeCatalyst 動作：

- 建置動作。

  一般而言，如果您想要部署至不存在對應部署動作的目標，或是想要對建置程序進行更多控制，則可以使用建置動作。如需使用建置動作部署資源的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。

- 一個GitHub 動作。

  您可以在工GitHub 作 CodeCatalyst 流程中使用動作來部署應用程式和資源 (而非 CodeCatalyst動作)。如需有關如何在工作 CodeCatalyst 流程中使用 GitHub 「動作」的資訊，請參閱 新增 GitHub 動作

如果您不想使用工作流程來部署應用程式，也可以使用下列 AWS 服務來部署應用 CodeCatalyst 程式：

- AWS CodeDeploy — 請參閱什麼是 CodeDeploy?
- AWS CodeBuild 和 AWS CodePipeline — 請參閱什麼是 AWS CodeBuild? 什麼是 AWS CodePipeline？
- AWS CloudFormation — 請參閱什麼是 AWS CloudFormation?

針對複雜的企業部署使 CodeDeploy用、、和 CloudFormation 服務。 CodeBuild CodePipeline

教學課程：使用部署無伺服器應用程式 AWS CloudFormation

在本教學課程中，您將學習如何使用工作流程和 AWS CloudFormation.

本教程中的應用程序是一個簡單的 Web 應用程序，輸出「Hello World」消息。它由一個 AWS Lambda 函數和一個 Amazon API Gateway 組成，您可以使用 AWS Serverless Application Model (AWS SAM) 來建立它，這是 AWS CloudFormation.

主題

- 必要條件
- 步驟 1：建立來源儲存庫
- 步驟 2：建立 AWS 角色
- 步驟 3：將 AWS 角色新增至 CodeCatalyst
- 步驟 4：創建一個 Amazon S3 存儲桶
- 步驟 5：添加源文件
- 步驟 6：建立並執行工作流程
- 步驟 7：進行變更
- 清除

必要條件

開始之前：

- 您需要具有已連線 AWS 帳戶的 CodeCatalyst 空間。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。
- 在您的空間中，您需要一個空的，從頭開始的 CodeCatalyst項目，名為：

  ```
  codecatalyst-cfn-project
  ```

  如需詳細資訊，請參閱 在 Amazon 中創建一個空項目 CodeCatalyst。
- 在你的項目中，你需要一 CodeCatalyst 個名為：

  ```
  codecatalyst-cfn-environment
  ```

  設定此環境的方式如下：
  - 選擇任何類型，例如非生產。
  - 將您的 AWS 帳戶 Connect 到它。

  如需詳細資訊，請參閱 使用環境。

步驟 1：建立來源儲存庫

在此步驟中，您可以在中建立來源儲存庫 CodeCatalyst。此儲存庫用於儲存教學課程的來源檔案，例如 Lambda 函數檔案。

如需來源儲存庫的詳細資訊，請參閱建立來源儲存庫。

若要建立來源儲存庫

1. 在功能窗格中 CodeCatalyst，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
2. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
3. 在存放庫名稱中，輸入：

   codecatalyst-cfn-source-repository
4. 選擇建立。

現在，您已經創建了一個名為codecatalyst-cfn-source-repository.

步驟 2：建立 AWS 角色

在此步驟中，您會建立下列 AWS IAM 角色：

- 部署角色 — 授與部 CodeCatalyst 署 AWS CloudFormation 堆疊動作權限，以存取您將在其中部署無伺 CloudFormation 服器應用程式的 AWS 帳戶和服務。「部署 AWS CloudFormation 堆疊」動作是工作流程的一部分。
- 建立角色 — 授與 CodeCatalyst 建立動作權限，以存取您的 AWS 帳戶並寫入存放無伺服器應用程式套件的 Amazon S3。建置動作是工作流程的一部分。
- 堆疊角色 — CloudFormation 授與讀取和修改您稍後將提供之 AWS SAM 範本中指定之資源的權限。此外，也會授予權限 CloudWatch。

如需 IAM 角色的詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的 IAM 角色。

> **Note**
>
> 為了節省時間，您可以建立一個稱為角色的單一CodeCatalystWorkflowDevelopmentRole-*spaceName*角色，而不是先前列出的三個角色。如需詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。本教學課程假設您正在建立先前列出的三個角色。

> **Note**
>
> Lambda 執行角色也是必要的，但您現在不需要建立它，因為當您在步驟 5 中執行工作流程時，sam-template.yml檔案會為您建立角色。

若要建立部署角色

1. 建立角色的策略，執行方式如下：

a. 登入到 AWS。

b. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

c. 在導覽窗格中，選擇政策。

d. 選擇 Create policy (建立政策)。

e. 請選擇 JSON 標籤。

f. 刪除現有的程式碼。

g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-deploy-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立部署角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋`codecatalyst-deploy-policy`並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-deploy-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst deploy role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的部署角色。

3. 取得部署角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入剛建立的角色名稱 (codecatalyst-deploy-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的部署角色，並取得其 ARN。

若要建立建置角色

1. 建立角色的策略，執行方式如下：

a. 登入到 AWS。

b. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

c. 在導覽窗格中，選擇政策。

d. 選擇 Create policy (建立政策)。

e. 請選擇 JSON 標籤。

f. 刪除現有的程式碼。

g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
```

```
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
```

```
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋codecatalyst-build-policy並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-build-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst build role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

3. 取得建置角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入剛建立的角色名稱 (codecatalyst-build-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的組建角色，並取得其 ARN。

若要建立堆疊角色

1. AWS 使用您要部署堆疊的帳戶登入。
2. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
3. 建立堆疊角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)。

b. 選擇 Create Role (建立角色)。

c. 選擇 AWS 服務。

d. 在 [使用案例] 區段中，CloudFormation從下拉式清單中選擇。

e. 選取選項按CloudFormation鈕。

f. 選擇底部的 [下一步]。

g. 使用搜尋方塊尋找下列權限原則，然後選取各自的核取方塊。

> **Note**
>
> 如果您搜尋的策略並未顯示，請務必選擇 [清除篩選器]，然後再試一次。

- CloudWatchFullAccess
- AWS CloudFormationFullAccess
- IAM FullAccess
- AWS羔羊 FullAccess
- 亞馬遜 API GatewayAdministrator
- 亞馬遜 FullAccess
- 亚马逊 ContainerRegistryFullAccess

第一個原則允許存取， CloudWatch 以便在發生警示時啟用堆疊復原。

其餘的原則 AWS SAM 允許存取將在本教學課程中部署的堆疊中的服務和資源。如需詳細資訊，請參閱AWS Serverless Application Model 開發人員指南中的權限。

h. 選擇下一步。

i. 在「角色名稱」中，輸入：

```
codecatalyst-stack-role
```

j. 選擇建立角色。

4. 取得堆疊角色的 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入剛建立的角色名稱 (codecatalyst-stack-role)。

c. 從清單中選擇角色。

d. 在「摘要」段落中，複製 ARN 值。供稍後使用。

您現在已建立具有適當權限的堆疊角色，而且已取得其 ARN。

步驟 3：將 AWS 角色新增至 CodeCatalyst

在此步驟中，您將組建角色 (codecatalyst-build-role) 和部署 role (codecatalyst-deploy-role) 新增至空間中的 CodeCatalyst 帳戶連線。

> **Note**
>
> 你不需要堆棧角色（ codecatalyst-stack-role ）添加到連接。這是因為堆疊角色是由 CloudFormation(不 CodeCatalyst) 使用，在 CodeCatalyst 與 AWS 使用部署角色之間建立連接之後。由於堆疊角色不會被用 CodeCatalyst 來取得存取權 AWS，因此不需要與帳戶連線產生關聯。

將建置和部署角色新增至您的帳戶連線

1. 在中 CodeCatalyst，導覽至您的空間。
2. 選擇AWS 帳戶。此時會顯示帳戶連線清單。
3. 選擇代表您建立組建和部署角色之 AWS 帳戶的帳戶連線。
4. 從管理主控台選擇 [AWS 管理角色]。

   將會顯示「將 IAM 角色新增至 Amazon CodeCatalyst 空間」頁面。您可能需要登入才能存取此頁面。
5. 選取 [新增您在 IAM 中建立的現有角色]。

   這時系統顯示下拉列表。此清單會顯示具有信任政策的所有 IAM 角色，其中包括codecatalyst-runner.amazonaws.com和codecatalyst.amazonaws.com服務主體。
6. 在下拉式清單中，選擇codecatalyst-build-role，然後選擇 [新增角色]。
7. 選擇 [新增 IAM 角色]，選擇 [新增您在 IAM 中建立的現有角色]，然後在下拉式清單中選擇codecatalyst-deploy-role。選擇 Add role (新增角色)。

您現在已將組建和部署角色新增至您的空間。

8. 複製 Amazon CodeCatalyst 顯示名稱的值。建立工作流程時，您將需要此值。

步驟 4：創建一個 Amazon S3 存儲桶

在此步驟中，您會建立 Amazon S3 儲存貯體，用於存放無伺服器應用程式的部署套件 .zip 檔案。

建立 Amazon S3 儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在主窗格中，選擇 [建立值區]。
3. 若為「值區名稱」，請輸入：

```
codecatalyst-cfn-s3-bucket
```

4. 對於 AWS 區域，選擇一個區域。本教學課程假設您選擇美國西部 (奧勒岡) us-we st-2。如需 Amazon S3 支援區域的相關資訊，請參閱中的 Amazon 簡單儲存服務端點和配額AWS 一般參考。
5. 在頁面底部，選擇 [建立值區]。

您現在已經建立了一個名為美國西部 (奧勒岡) us-west-2 區**codecatalyst-cfn-s3-bucket**域的值區。

步驟 5：添加源文件

在此步驟中，您可以將數個應用程式來源檔案新增至 CodeCatalyst 來源儲存庫。資`hello-world`料夾包含您將部署的應用程式檔案。該`tests`文件夾包含單元測試。資料夾結構如下：

```
.
|— hello-world
|   |— tests
|      |— unit
|         |— test-handler.js
|   |— app.js
|— .npmignore
|— package.json
```

```
|— sam-template.yml
|— setup-sam.sh
```

.npmignore 文件

該文.npmignore件指示 npm 應從應用程序包中排除哪些文件和文件夾。在本教程中，npm 會排除該tests文件夾，因為它不是應用程序的一部分。

若要加入 .npmignore 檔案

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的項目，codecatalyst-cfn-project
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 從來源儲存庫清單中，選擇您的儲存庫codecatalyst-cfn-source-repository。
5. 在檔案中，選擇建立檔案。
6. 在「檔案名稱」中，輸入：

   ```
   .npmignore
   ```

7. 在文字方塊中，輸入下列程式碼：

   ```
   tests/*
   ```

8. 選擇「確認」，然後再次選擇「確認」。

   現在，您已經在存儲庫的根目錄.npmignore中創建了一個名為的文件。

打包. JSON 文件

該package.json文件包含有關 Node 項目的重要元數據，例如項目名稱，版本號，描述，依賴項以及描述如何與應用程序交互和運行應用程序的其他詳細信息。

本教學課程package.json中包含相依性清單和指test令碼。測試腳本執行以下操作：

- 使用摩卡，測試腳本運行中指定的單元測試，hello-world/tests/unit/並使用 xunit 報告器將結果寫入junit.xml文件。
- 使用伊斯坦布爾（nyc），測試腳本使用三葉草記者生成代碼覆蓋率報告（clover.xml）。如需詳細資訊，請參閱伊斯坦布爾文件中的使用替代記者。

若要新增封裝 .json 檔案

1. 在存放庫的檔案中，選擇建立檔案。
2. 在「檔案名稱」中，輸入：

   ```
   package.json
   ```

3. 在文字方塊中，輸入下列程式碼：

   ```
   {
     "name": "hello_world",
     "version": "1.0.0",
     "description": "hello world sample for NodeJS",
     "main": "app.js",
     "repository": "https://github.com/awslabs/aws-sam-cli/tree/develop/samcli/local/
   init/templates/cookiecutter-aws-sam-hello-nodejs",
     "author": "SAM CLI",
     "license": "MIT",
     "dependencies": {
       "axios": "^0.21.1",
       "nyc": "^15.1.0"
     },
     "scripts": {
       "test": "nyc --reporter=clover mocha hello-world/tests/unit/ --reporter xunit
    --reporter-option output=junit.xml"
     },
     "devDependencies": {
       "aws-sdk": "^2.815.0",
       "chai": "^4.2.0",
       "mocha": "^8.2.1"
     }
   }
   ```

4. 選擇「確認」，然後再次選擇「確認」。

   現在，您已經添加了一個名`package.json`為存儲庫的根目錄的文件。

薩姆模板 .yml 文件

該`sam-template.yml`檔案包含部署 Lambda 函數和 API Gateway 以及一起設定它們的說明。它遵循AWS Serverless Application Model 模板規範，擴展了 AWS CloudFormation 模板規範。

您在本教學課程中使用 AWS SAM 範本而不是一般 AWS CloudFormation 範本，因為 AWS SAM 提供有用的:: 無伺服器AWS:: Function 資源類型。這種類型執行許多 behind-the-scenes 配置，您通常必須寫出來才能使用基本 CloudFormation 語法。例如，AWS::Serverless::Function會建立啟動函數的 Lambda 函數、Lambda 執行角色和事件來源對應。如果你想使用基本編寫它，你必須編寫所有這些 CloudFormation。

雖然本教學課程使用預先撰寫的範本，但您可以使用建置動作來產生作為工作流程的一部分。如需詳細資訊，請參閱 新增「部署 AWS CloudFormation 堆疊」動作。

若要新增範例範本 .yml 檔案

1. 在存放庫的檔案中，選擇建立檔案。
2. 在「檔案名稱」中，輸入：

```
sam-template.yml
```

3. 在文字方塊中，輸入下列程式碼：

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  serverless-api

  Sample SAM Template for serverless-api

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # For details on this resource type,
 see https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello-world/
      Handler: app.lambdaHandler
      Runtime: nodejs12.x
      Events:
        HelloWorld:
```

```
          Type: Api # For details on this event source type, see
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

Outputs:
  # ServerlessRestApi is an implicit API created out of the events key under
 Serverless::Function
  # Find out about other implicit resources you can reference within AWS SAM at
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/
internals/generated_resources.rst#api
  HelloWorldApi:
    Description: "API Gateway endpoint URL for the Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.
${AWS::Region}.amazonaws.com/Prod/hello/"
  HelloWorldFunction:
    Description: "Hello World Lambda function ARN"
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: "Implicit Lambda execution role created for the Hello World
 function"
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

4. 選擇「確認」，然後再次選擇「確認」。

   現在，您已經在存儲庫的根文件夾`sam-template.yml`下添加了一個名為的文件。

setup-sam.sh 檔案

此`setup-sam.sh`檔案包含下載和安裝 AWS SAM CLI 公用程式的指示。工作流程會使用此公用程式來封裝`hello-world`來源。

若要新增 setup-sam.sh 檔案

1. 在存放庫的檔案中，選擇建立檔案。
2. 在「檔案名稱」中，輸入：

   ```
   setup-sam.sh
   ```

3. 在文字方塊中，輸入下列程式碼：

```
#!/usr/bin/env bash
echo "Setting up sam"

yum install unzip -y

curl -LO https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
linux-x86_64.zip
unzip -qq aws-sam-cli-linux-x86_64.zip -d sam-installation-directory

./sam-installation-directory/install; export AWS_DEFAULT_REGION=us-west-2
```

在前面的代碼中，將 *us-west-2* 替換為您的區域。 AWS

4. 選擇「確認」，然後再次選擇「確認」。

現在，您已經添加了一個名`setup-sam.sh`為存儲庫的根目錄的文件。

app.js 檔案

`app.js`包含 Lambda 函數程式碼。在本教程中，代碼返回文本`hello world`。

若要新增 app.js 檔案

1. 在存放庫的檔案中，選擇建立檔案。
2. 在「檔案名稱」中，輸入：

```
hello-world/app.js
```

3. 在文字方塊中，輸入下列程式碼：

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;

/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
```

```
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
    try {
        // const ret = await axios(url);
        response = {
            'statusCode': 200,
            'body': JSON.stringify({
                message: 'hello world',
                // location: ret.data.trim()
            })
        }
    } catch (err) {
        console.log(err);
        return err;
    }

    return response
};
```

4. 選擇「確認」，然後再次選擇「確認」。

   您現在已經創建了一個名為的文件夾hello-world和一個名為的文件app.js。

test-handler.js 檔案

該test-handler.js文件包含 Lambda 函數的單元測試。

若要新增 test-handler.js 檔案

1. 在存放庫的檔案中，選擇建立檔案。
2. 在「檔案名稱」中，輸入：

   ```
   hello-world/tests/unit/test-handler.js
   ```

3. 在文字方塊中，輸入下列程式碼：

```
'use strict';

const app = require('../../app.js');
const chai = require('chai');
const expect = chai.expect;
var event, context;

describe('Tests index', function () {
    it('verifies successful response', async () => {
        const result = await app.lambdaHandler(event, context)

        expect(result).to.be.an('object');
        expect(result.statusCode).to.equal(200);
        expect(result.body).to.be.an('string');

        let response = JSON.parse(result.body);

        expect(response).to.be.an('object');
        expect(response.message).to.be.equal("hello world");
        // expect(response.location).to.be.an("string");
    });
});
```

4. 選擇「確認」，然後再次選擇「確認」。

   現在，您已經在文件hello-world/tests/unit夾test-handler.js下添加了一個名為的文件。

您現在已經添加了所有源文件。

花點時間仔細檢查您的工作，並確保將所有文件放在正確的文件夾中。資料夾結構如下：

```
.
|— hello-world
|  |— tests
|     |— unit
|        |— test-handler.js
|  |— app.js
|— .npmignore
|— README.md
|— package.json
|— sam-template.yml
```

```
|— setup-sam.sh
```

步驟 6：建立並執行工作流程

在此步驟中，您會建立封裝 Lambda 原始程式碼並進行部署的工作流程。工作流程由下列依序執行的建置區塊組成：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 測試操作（Test）-在觸發器上，此操作安裝節點包管理器（NPM），然後運行該`npm run test`命令。這個命令告訴 npm 運行`package.json`文件中定義的`test`腳本。`test`腳本反過來運行單元測試並生成兩個報告：測試報告（`junit.xml`）和代碼覆蓋報告（`clover.xml`）。如需詳細資訊，請參閱 打包. JSON 文件。

  接下來，測試動作將 XML 報告轉換為 CodeCatalyst 報告，並在 CodeCatalyst 控制台中，測試動作的「報告」選項卡下顯示它們。

  如需測試動作的詳細資訊，請參閱使用工作流程測試 CodeCatalyst。
- 建置動作 (`BuildBackend`) — 完成測試動作後，建置動作會下載並安裝 AWS SAM CLI、封裝`hello-world`來源，然後將套件複製到 Lambda 服務預期的 Amazon S3 儲存貯體。此動作也會輸出名為的新 AWS SAM 範本檔案，`sam-template-packaged.yml`並將其置於名為的輸出加工品中`buildArtifact`。

  如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
- 部署動作 (`DeployCloudFormationStack`) — 建置動作完成後，部署動作會尋找建置動作 (`buildArtifact`) 所產生的輸出成品，尋找 AWS SAM 範本內部，然後執行範本。 AWS SAM 範本會建立部署無伺服器應用程式的堆疊。

若要建立工作流程

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇建立工作流程。
3. 針對來源儲存庫，選擇`codecatalyst-cfn-source-repository`。
4. 對於「分支」，請選擇`main`。
5. 選擇建立。
6. 刪除 YAML 範例程式碼。

7. 新增下列 YAML 程式碼：

```
Name: codecatalyst-cfn-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Test:
    Identifier: aws/managed-test@v1
    Inputs:
      Sources:
        - WorkflowSource
    Outputs:
      Reports:
        CoverageReport:
          Format: CLOVERXML
          IncludePaths:
            - "coverage/*"
        TestReport:
          Format: JUNITXML
          IncludePaths:
            - junit.xml
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
  BuildBackend:
    Identifier: aws/build@v1
    DependsOn:
      - Test
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-build-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
```

```
        - Run: . ./setup-sam.sh
        - Run: sam package --template-file sam-template.yml --s3-
bucket codecatalyst-cfn-s3-bucket --output-template-file sam-template-packaged.yml
 --region us-west-2
    Outputs:
      Artifacts:
        - Name: buildArtifact
          Files:
            - "**/*"
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    DependsOn:
      - BuildBackend
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-deploy-role
    Inputs:
      Artifacts:
        - buildArtifact
      Sources: []
    Configuration:
      name: codecatalyst-cfn-stack
      region: us-west-2
      role-arn: arn:aws:iam::111122223333:role/StackRole
      template: ./sam-template-packaged.yml
      capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

在前面的代碼中，替換：

- 兩個執行個體都*codecatalyst-cfn-environment*具有您的環境名稱。
- *codecatalyst-account-connection*與您的帳戶連接的顯示名稱的兩個實例。顯示名稱可能是數字。如需詳細資訊，請參閱 步驟 3：將 AWS 角色新增至 CodeCatalyst。
- *codecatalyst-build-role*具有您在中建立之建置角色的名稱步驟 2：建立 AWS 角色。
- codecatalyst-cfn-s具有您在步驟 4：創建一個 Amazon S3 存儲桶其中創建的 Amazon S3 存儲桶的名稱的 *3* 個存儲桶。
- *us-west-2* 的兩個執行個體與 Amazon S3 儲存貯體所在區域 (第一個執行個體) 以及堆疊的部署位置 (第二個執行個體)。這些區域可以是不同的。本自學課程假設兩個「區域」都設定為us-

west-2。如需 Amazon S3 支援區域的詳細資訊 AWS CloudFormation，請參閱 AWS 一般參考.

- *codecatalyst-deploy-role*使用您在中建立的部署角色名稱步驟 2：建立 AWS 角色。
- *codecatalyst-cfn-environment*使用您在中建立的環境名稱必要條件。
- *arn: aw:iam:: 1111223333: ##/StackRole############*的 Amazon 資源名稱 (ARN)。步驟 2：建立 AWS 角色

> **Note**
>
> 如果您決定不建立建置、部署和堆疊角色，請以角色的名稱或 ARN 取代*codecatalyst-build-rolecodecatalyst-deploy-role*、和 *arn: aw: iam:: 111122223 StackRole* 333:Role/。CodeCatalystWorkflowDevelopmentRole-*spaceName*如需有關此角色的詳細資訊，請參閱 步驟 2：建立 AWS 角色。

如需先前所示程式碼中屬性的相關資訊，請參閱「部署AWS CloudFormation堆疊」動作參考。

8. (選擇性) 選擇 [驗證]，確定 YAML 程式碼在認可之前是有效的。
9. 選擇 Commit (遞交)。
10. 在「提交工作流程」對話方塊中，輸入以下內容：

    a. 對於「工作流程」檔案名稱，請保留預設值，codecatalyst-cfn-workflow。

    b. 對於提交訊息，請輸入：

    ```
    add initial workflow file
    ```

    c. 針對「儲存庫」，選擇codecatalyst-cfn-source-repository。

    d. 選擇「主要」做為「分支名稱」。

    e. 選擇 Commit (遞交)。

您現在已建立工作流程。工作流程執行會自動啟動，因為工作流程頂端定義了觸發器。具體而言，當您認可 (並推送) codecatalyst-cfn-workflow.yaml 檔案至來源儲存庫時，觸發程序會啟動工作流程執行。

若要檢視進行中的工作流程執行

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇您剛建立的工作流程：`codecatalyst-cfn-workflow`。
3. 選擇「執行」頁標。
4. 在「執行 ID」欄中，選擇執行 ID。
5. 選擇測試以查看測試進度。
6. 選擇BuildBackend查看構建進度。
7. 選擇DeployCloudFormationStack以查看部署進度。

   如需檢視執行詳細資訊的詳細資訊，請參閱檢視工作流程執行狀態與詳細。
8. DeployCloudFormationStack動作完成後，請執行下列動作：
   - 如果工作流程執行成功，請移至下一個程序。
   - 如果測試或BuildBackend動作上的工作流程執行失敗，請選擇 [記錄檔] 以疑難排解問題。
   - 如果DeployCloudFormationStack動作上的工作流程執行失敗，請選擇部署動作，然後選擇 [摘要] 索引標籤。捲動至「CloudFormation 事件」區段以檢視詳細的錯誤訊息。如果發生復原，請先透過中的 AWS CloudFormation 主控台刪除`codecatalyst-cfn-stack`堆疊， AWS 然後再重新執行工作流程。

驗證部署的步驟

1. 成功部署後，從靠近頂部的水平功能表列中選擇變數 (7)。請勿在右側窗格中選擇「變數」。)
2. 在旁邊 HelloWorldApi，將 `https://` URL 貼到瀏覽器中。

   隨即顯示來自 Lambda 函數的全球 JSON 訊息，指出工作流程已成功部署和設定 Lambda 函數和 API Gateway。

> **Tip**
>
> 您可以在工作流程圖中 CodeCatalyst 顯示此 URL，並使用一些小型設定。如需詳細資訊，請參閱 顯示已部署應用程式的 URL。

驗證單元測試結果和代碼覆蓋率

1. 在工作流程圖表中，選擇 [測試]，然後選擇 [報告]。
2. 選擇TestReport檢視單元測試結果，或選擇檢視CoverageReport要測試之檔案的程式碼涵蓋範圍詳細資訊 (在本例中為app.js和) test-handler.js。

驗證已部署的資源

1. 登入 AWS Management Console 並開啟 API Gateway 主控台，網址為 https://console.aws.amazon.com/apigateway/。
2. 觀察該 AWS SAM 模板創建的 codecatalyst-cfn-stackAPI。API 名稱來自工作流程定義檔案 (codecatalyst-cfn-workflow.yaml) 中的Configuration/name值。
3. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
4. 在導覽視窗中，選擇函數。
5. 選擇您的 Lambda 函數、codecatalyst-cfn-stack-HelloWorldFunction-*string*.
6. 您可以查看 API Gateway 如何成為函數的觸發器。此整合已依 AWS SAM AWS::Serverless::Function資源類型自動設定。

步驟 7：進行變更

在此步驟中，您可以變更 Lambda 原始程式碼並進行認可。此提交會啟動新的工作流程執行。此執行會以藍綠色配置部署新的 Lambda 函數，該配置使用 Lambda 主控台中指定的預設流量轉移組態。

若要變更您的 Lambda 來源

1. 在中 CodeCatalyst，導覽至您的專案。
2. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 選擇您的來源儲存庫codecatalyst-cfn-source-repository。
4. 更改應用程序文件：
    a. 選擇 hello-world 資料夾。
    b. 選擇app.js檔案。
    c. 選擇編輯。
    d. 在第 23 行上，變更hello world為**Tutorial complete!**。
    e. 選擇「確認」，然後再次選擇「確認」。

提交會導致工作流程執行開始。此運行將失敗，因為您尚未更新單元測試以反映名稱更改。

5. 更新單元測試：

   a. 選擇 `hello-world\tests\unit\test-handler.js`。

   b. 選擇編輯。

   c. 在第 19 行上，`hello world`將變更為**`Tutorial complete!`**。

   d. 選擇「確認」，然後再次選擇「確認」。

   提交會導致另一個工作流程執行開始。此次執行將會成功。

6. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
7. 選擇`codecatalyst-cfn-workflow`，然後選擇 [執行]。
8. 選擇最近一次執行的執行 ID。它應該仍在進行中。
9. 選擇測試、BuildBackend，並DeployCloudFormationStack查看工作流程執行進度。
10. 當工作流程完成時，選擇頂部附近的變數 (7)。
11. 在旁邊 HelloWorldApi，將 `https://` URL 貼到瀏覽器中。

    瀏覽器中會出現一則`Tutorial complete!`訊息，指出您的新應用程式已成功部署。

清除

清理本教程中使用的文件和服務，以避免被收取費用。

若要在 CodeCatalyst 主控台中進行清理

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 刪除`codecatalyst-cfn-workflow`。
3. 刪除`codecatalyst-cfn-environment`。
4. 刪除`codecatalyst-cfn-source-repository`。
5. 刪除`codecatalyst-cfn-project`。

若要在中進行清理 AWS Management Console

1. 在中進行清理 CloudFormation，如下所示：

a. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation

b. 刪除codecatalyst-cfn-stack。

刪除堆疊會移除 API Gateway 和 Lambda 服務中的所有教學課程資源。

2. 在 Amazon S3 中進行清理，如下所示：

a. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。

b. 選擇 codecatalyst-cfn-s3-bucket。

c. 刪除值區內容。

d. 刪除儲存貯體。

3. 在 IAM 中進行清理，如下所示：

a. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

b. 刪除codecatalyst-deploy-policy。

c. 刪除codecatalyst-build-policy。

d. 刪除codecatalyst-stack-policy。

e. 刪除codecatalyst-deploy-role。

f. 刪除codecatalyst-build-role。

g. 刪除codecatalyst-stack-role。

在本教學課程中，您學習了如何使用 CodeCatalyst 工作流程和 Deploy CloudFormation 堆疊動作將無伺服器應用程式部署為 AWS CloudFormation 堆疊。

教學課程：將應用程式部署到 Amazon ECS

在本教學中，您將學習如何使用工作流程、Amazon ECS 和其他一些服務將無伺服器應用程式部署到 Amazon 彈性容器服務 (Amazon ECS)。 AWS 部署的應用程序是一個簡單的你好世界網站，建立在 Apache 的 Web 服務器碼頭圖像。此教學課程會引導您完成必要的準備工作，例如設定叢集，然後說明如何建立工作流程以建置和部署應用程式。

> **Tip**
>
> 您可以使用藍圖來為您完成完整的 Amazon ECS 設定，而不是逐步完成本教學課程。您將需要使用任一 Node.js API 與 AWS Fargate或 Java API 與 AWS Fargate藍圖。如需詳細資訊，請參閱 使用藍圖建立專案。

主題

必要條件

開始之前：

- 您需要具有已連線 AWS 帳戶的 CodeCatalyst 空間。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。
- 在您的空間中，您需要一個空的，從頭開始的 CodeCatalyst項目，名為：

  ```
  codecatalyst-ecs-project
  ```

  如需詳細資訊，請參閱 在 Amazon 中創建一個空項目 CodeCatalyst。
- 在你的項目中，你需要一 CodeCatalyst 個名為：

  ```
  codecatalyst-ecs-environment
  ```

設定此環境的方式如下：

- 選擇任何類型，例如非生產。
- 將您的 AWS 帳戶 Connect 到它。

如需詳細資訊，請參閱 使用環境。

步驟 1：設定 AWS 使用者和 AWS CloudShell

本教學課程的第一個步驟是在中建立使用者 AWS IAM Identity Center，然後以此使用者身分啟動 AWS CloudShell 執行個體。在本教程的持續時間， CloudShell 是您的開發計算機，是您配置 AWS 資源和服務的地方。完成教學課程後刪除此使用者。

> **Note**
>
> 請勿在本教學課程中使用 root 使用者。您必須建立個別的使用者，否則稍後在 AWS Command Line Interface (CLI) 中執行動作時可能會遇到問題。

如需 IAM 身分中心使用者的詳細資訊 CloudShell，請參閱AWS IAM Identity Center 使用者指南和AWS CloudShell 使用者指南。

建立 IAM 身分中心使用者

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

   > **Note**
   >
   > 確保您使用連接到您的 CodeCatalyst空間的登錄。 AWS 帳戶 您可以瀏覽至您的空間並選擇 AWS 帳戶索引標籤，以確認已連接哪個帳戶。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。

2. 在導覽窗格中，選擇 使用者，然後選擇 新增使用者。
3. 在使用者名稱中，輸入：

   ```
   CodeCatalystECSUser
   ```

4. 在 [密碼] 下方，選擇 [產生一次性密碼，您可以與此使用者共用]。

5. 在 [電子郵件地址] 和 [確認電子郵件地址] 中，輸入 IAM 身分中心尚未存在的電子郵件地址。
6. 在名字和姓氏中，輸入：

```
CodeCatalystECSUser
```

7. 在 [顯示名稱] 中，保留自動產生的名稱：

```
CodeCatalystECSUser CodeCatalystECSUser
```

8. 選擇下一步。
9. 在 [新增使用者至群組] 頁面上，選擇 [下一步]。
10. 在 [檢閱並新增使用者] 頁面上，檢閱資訊並選擇 [新增使用者]。

    這時系統顯示一次性密碼對話框。
11. 選擇 [複製]，然後貼上登入資訊，包括 AWS 存取入口網站 URL 和一次性密碼。
12. 選擇關閉。

建立許可集合

您會將此權限集指派給CodeCatalystECSUser稍後。

1. 在瀏覽窗格中，選擇 [權限集]，然後選擇 [建立權限集]。
2. 選擇「預先定義」權限集，然後選取AdministratorAccess。此原則為所有人提供完整權限 AWS 服務。
3. 選擇下一步。
4. 在權限集名稱中，輸入：

```
CodeCatalystECSPermissionSet
```

5. 選擇下一步。
6. 在 [檢閱並建立] 頁面上，檢閱資訊並選擇 [建立]。

若要將權限集指派給 CodeCatalyst ECSUSER

1. 在功 AWS 帳戶 能窗格中，選擇 AWS 帳戶，然後選取您目前登入的對象旁邊的核取方塊。
2. 選擇 [指派使用者或群組]。

3. 選擇 Users (使用者) 索引標籤。
4. 選取旁邊的核取方塊CodeCatalystECSUser。
5. 選擇下一步。
6. 選取旁邊的核取方塊CodeCatalystECSPermissionSet。
7. 選擇下一步。
8. 複查資訊，然後選擇「提交」。

   您現在已經分配CodeCatalystECSUser並CodeCatalystECSPermissionSet將 AWS 帳戶它們綁定在一起。

若要以 CodeCatalyst ECSUSER 的身分登出並重新登入

1. 在您登出之前，請確定您擁有的 AWS 存取入口網站 URL 以及使用者名稱和一次性密碼CodeCatalystECSUser。您應該先前將此資訊複製到文字編輯器中。

   > **Note**
   >
   > 如果您沒有這些信息，請轉到 IAM 身份中心的CodeCatalystECSUser詳細信息頁面，選擇重置密碼，生成一次性密碼 [...] ，然後再次重設密碼以在螢幕上顯示資訊。

2. 登出 AWS。
3. 將 AWS 訪問門戶網址粘貼到瀏覽器的地址欄中。
4. 使用的使用者名稱和一次性密碼登入CodeCatalystECSUser。
5. 在 [新密碼] 中，輸入密碼，然後選擇 [設定新密碼]。

   螢幕上會出現一個AWS 帳戶方塊。

6. 選擇 AWS 帳戶，然後選擇您 AWS 帳戶 為其指派CodeCatalystECSUser使用者和權限集的名稱。
7. 在旁邊CodeCatalystECSPermissionSet，選擇 [管理主控台]。

   AWS Management Console 隨即出現。您現在已使用適當CodeCatalystECSUser的權限登入。

若要啟動 AWS CloudShell 執行個體

1. 作為CodeCatalystECSUser，在頂部導航欄中，選擇 AWS 圖標
   （ ）。

   的主頁面 AWS Management Console 隨即出現。
2. 在上方導覽列中，選擇 AWS CloudShell 圖示
   ( )。

   CloudShell 打開。等待 CloudShell 環境建立完成。

   > **Note**
   >
   > 如果沒有看到 CloudShell 圖示，請確定您所在的地區受支援 CloudShell。本教學課程假設您位於美國西部 (奧勒岡) 區域。

若要確認 AWS CLI 已安裝

1. 在 CloudShell 終端機中，輸入：

   ```
   aws --version
   ```
2. 檢查是否顯示版本。

   AWS CLI 已針對目前使用者設定CodeCatalystECSUser，因此不需要設定 AWS CLI 金鑰和認證，就像通常情況一樣。

步驟 2：將預留位置應用程式部署到 Amazon ECS

在本節中，您要手動將預留位置應用程式部署到 Amazon ECS。此預留位置應用程式將由您的工作流程部署的 Hello World 應用程式取代。預留位置應用程式是 Apache 網頁伺服器。

如需 Amazon ECS 的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南。

完成下列一系列程序來部署預留位置應用程式。

若要建立作業執行角色

此角色授予 Amazon ECS 和代表您進行 API 呼叫的 AWS Fargate (Fargate) 權限。

1. 建立信任原則：

   a. 在中 AWS CloudShell，輸入下列命令：

   ```
   cat > codecatalyst-ecs-trust-policy.json
   ```

   CloudShell 終端機中會出現閃爍的提示。

   b. 在提示符下輸入以下代碼：

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": "ecs-tasks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   c. 將游標置於最後一個大括號 (}) 之後。

   d. 按，**Enter**然後**Ctrl+d**保存文件並退出貓。

2. 建立工作執行角色：

   ```
   aws iam create-role \
         --role-name codecatalyst-ecs-task-execution-role \
         --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
   ```

3. 將 AWS 受管理的AmazonECSTaskExecutionRolePolicy策略附加到角色：

   ```
   aws iam attach-role-policy \
         --role-name codecatalyst-ecs-task-execution-role \
         --policy-arn arn:aws:iam::aws:policy/service-role/
   AmazonECSTaskExecutionRolePolicy
   ```

4. 顯示角色的詳細資訊：

   ```
   aws iam get-role \
   ```

```
    --role-name codecatalyst-ecs-task-execution-role
```

5. 請記下角色的"Arn":值，例如，arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role。您稍後將需要此 Amazon 資源名稱（ARN）。

建立 Amazon ECS 叢集

此叢集將包含 Apache 預留位置應用程式，以及之後的 Hello World 應用程式。

1. 如同 CodeCatalystECSUser AWS CloudShell，在中建立空叢集：

```
aws ecs create-cluster --cluster-name codecatalyst-ecs-cluster
```

2. (選擇性) 確認叢集是否已成功建立：

```
aws ecs list-clusters
```

codecatalyst-ecs-cluster叢集的 ARN 應該會出現在清單中，表示建立成功。

建立工作定義檔案的步驟

任務定義文件指示運行從 DockerHub中提取的 Apache 2.4 Web 服務器碼頭圖像（httpd:2.4）。

1. 如同 CodeCatalystECSUser AWS CloudShell，在中建立工作定義檔案：

```
cat > taskdef.json
```

2. 在提示符下粘貼以下代碼：

```
{
    "executionRoleArn": "arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            "image": "httpd:2.4",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
```

```
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "family": "codecatalyst-ecs-task-def",
    "memory": "512",
    "networkMode": "awsvpc"
}
```

在前面的代碼中，替換成組*#AWN#IAM:: 11112222333###/## codecatalyst-ecs-task-execution*

使用您在中若要建立作業執行角色記錄的任務執行角色的 ARN。

3. 將游標置於最後一個大括號 (}) 之後。
4. 按，**Enter**然後**Ctrl+d**保存文件並退出貓。

若要向 Amazon ECS 註冊任務定義檔案

1. 如同 CodeCatalystECSUser AWS CloudShell，在中註冊任務定義：

```
aws ecs register-task-definition \
    --cli-input-json file://taskdef.json
```

2. (選擇性) 確認工作定義是否已註冊：

```
aws ecs list-task-definitions
```

codecatalyst-ecs-task-def任務定義應該會出現在清單中。

要創建 Amazon ECS 服務

Amazon ECS 服務會執行 Apache 預留位置應用程式的任務 (以及相關聯的碼頭容器)，以及之後的 Hello World 應用程式。

1. 作為CodeCatalystECSUser，切換到 Amazon 彈性容器服務主控台（如果尚未這樣做）。

2. 選擇您先前建立的叢集codecatalyst-ecs-cluster。
3. 在「服務」標籤中，選擇「建立」。
4. 在「建立」頁面中，執行下列動作：

   a. 保留所有預設設定，但接下來列出的設定除外。

   b. 針對 Launch type (啟動類型)，選擇 FARGATE。

   c. 在「工作定義」下的「系列」下拉式清單中，選擇：

      codecatalyst-ecs-task-def

   d. 在「服務名稱」中，輸入：

      ```
      codecatalyst-ecs-service
      ```

   e. 針對 [想要的工作]，輸入：

      ```
      3
      ```

      在本教學課程中，每項工作都會啟動單一 Docker 容器。

   f. 展開 [網路] 區段。

   g. 對於 VPC，請選擇任何 VPC。

   h. 對於子網路，請選擇任何子網路。

      > **Note**
      >
      > 僅指定一個子網路。這就是本教程所需要的一切。

      > **Note**
      >
      > 如果您沒有 VPC 和子網路，請建立它們。請參閱 Amazon VPC 使用者指南中的建立 VPC，以及在您的 VPC 中建立子網路。

   i. 針對 [安全性群組]，選擇 [建立新的安全性群組]，然後執行下列動作：

      i. 在「安全性群組名稱」中，輸入：

         ```
         codecatalyst-ecs-security-group
         ```

ii. 對於安全性群組說明，請輸入：

```
CodeCatalyst ECS security group
```

iii. 選擇新增規則。在「類型」中選擇「HTTP」，並針對「來源」選擇「任意位置」

j. 選擇底部的 [建立]。

k. 等待服務建立完成。這可能需要幾分鐘的時間。

5. 選擇 [作業] 索引標籤，然後選擇 [重新整理] 按鈕。確認所有三個工作的 [上次狀態] 欄都設定為 [執行中]。

(選擇性) 若要確認您的 Apache 預留位置應用程式正在執行

1. 在 [工作] 索引標籤中，選擇三個工作中的任何一個。
2. 在 [公用 IP] 欄位中，選擇 [開啟位址]。

   隨即顯示`It Works!`頁面。這表示 Amazon ECS 服務成功啟動了使用 Apache 映像啟動碼頭容器的任務。

   在本教學中，您已手動部署 Amazon ECS 叢集、服務和任務定義，以及 Apache 預留位置應用程式。有了這些項目，您現在就可以建立工作流程，以教學課程的 Hello World 應用程式取代 Apache 預留位置應用程式。

步驟 3：建立 Amazon ECR 映像儲存庫

在本節中，您將在 Amazon Elastic Container Registry (Amazon ECR) 中創建一個私有映像存儲庫。此儲存庫會儲存教學課程的 Docker 映像檔，以取代您先前部署的 Apache 預留位置影像。

如需 Amazon ECR 的詳細資訊，請參閱 Amazon 彈性容器登錄使用者指南。

若要在 Amazon ECR 中建立映像儲存庫

1. 在中 `CodeCatalystECSUser` AWS CloudShell，在 Amazon ECR 中創建一個空存儲庫：

```
aws ecr create-repository --repository-name codecatalyst-ecs-image-repo
```

2. 顯示 Amazon ECR 儲存庫的詳細資訊：

```
aws ecr describe-repositories \
```

```
    --repository-names codecatalyst-ecs-image-repo
```

3. 請注意“repositoryUri”:值，例如，111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo。

   稍後在將存放庫新增至工作流程時需要它。

步驟 4：建立 AWS 角色

在本節中，您會建立 CodeCatalyst 工作流程才能運作所需的 AWS IAM 角色。這些角色包括：

- 建置角色 — 授予 CodeCatalyst 建置動作 (在工作流程中) 存取 AWS 帳戶並寫入 Amazon ECR 和 Amazon EC2 的權限。
- 部署角色 — 授予「 CodeCatalyst 部署到 ECS」動作 (在工作流程中) 存取 AWS 帳戶、Amazon ECS 和其他 AWS 一些服務的權限。

如需 IAM 角色的詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的 IAM 角色。

> **Note**
>
> 為了節省時間，您可以建立一個稱為角色的單一CodeCatalystWorkflowDevelopmentRole-*spaceName*角色，而不是先前列出的兩個角色。如需詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。本教學課程假設您正在建立先前列出的兩個角色。

若要建立組建和部署角色，您可以使用 AWS Management Console 或 AWS CLI.

AWS Management Console

若要建立建置和部署角色，請完成下列一系列程序。

若要建立建置角色

1. 建立角色的策略，如下所示：

a. 登入到 AWS。

b. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

c. 在導覽窗格中，選擇政策。

d. 選擇 Create policy (建立政策)。

e. 請選擇 JSON 標籤。

f. 刪除現有的程式碼。

g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-ecs-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

   a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

   b. 選擇 [自訂信任原則]。

   c. 刪除現有的自訂信任原則。

   d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   e. 選擇下一步。

   f. 在 [權限] 原則中，搜尋`codecatalyst-ecs-build-policy`，選取其核取方塊。

   g. 選擇下一步。

   h. 在「角色名稱」中，輸入：

```
codecatalyst-ecs-build-role
```

   i. 對於「角色」描述，請輸入：

```
CodeCatalyst ECS build role
```

   j. 選擇建立角色。

您現在已建立具有權限原則和信任原則的組建角色。

3. 取得建置角色 ARN，如下所示：

   a. 在導覽窗格中，選擇角色。

   b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-ecs-build-role)。

   c. 從清單中選擇角色。

      此時會顯示角色的 [摘要] 頁面。

   d. 在頂端複製 ARN 值。供稍後使用。

若要建立部署角色

1. 建立角色的策略，如下所示：

   a. 登入到 AWS。

   b. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

   c. 在導覽窗格中，選擇政策。

   d. 選擇建立政策。

   e. 請選擇 JSON 標籤。

   f. 刪除現有的程式碼。

   g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
```

```
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
```

```
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元。然後，您可以在策略可用之後使用資源名稱來縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-ecs-deploy-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立部署角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋codecatalyst-ecs-deploy-policy，選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-ecs-deploy-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst ECS deploy role
```

j. 選擇建立角色。

您現在已建立具有信任原則的部署角色。

3. 取得部署角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-ecs-deploy-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。供稍後使用。

AWS CLI

若要建立建置和部署角色，請完成下列一系列程序。

建立兩個角色的信任原則

如同 CodeCatalystECSUser AWS CloudShell，在中建立信任原則檔案：

1. 建立檔案：

```
cat > codecatalyst-ecs-trust-policy.json
```

2. 在終端提示符下，粘貼以下代碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

3. 將游標置於最後一個大括號 (}) 之後。
4. 按，**Enter**然後**Ctrl+d**保存文件並退出貓。

若要建立建置原則和建置角色

1. 建立建置原則：

   a. 如同 CodeCatalystECSUser AWS CloudShell，在中建立組建原則檔案：

   ```
   cat > codecatalyst-ecs-build-policy.json
   ```

   b. 在提示符下，輸入以下代碼：

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "ecr:*",
                   "ec2:*"
               ],
               "Resource": "*"
   ```

```
        }
    ]
}
```

c. 將游標置於最後一個大括號 (}) 之後。

d. 按，**Enter**然後**Ctrl+d**保存文件並退出貓。

2. 將構建策略添加到 AWS：

```
aws iam create-policy \
    --policy-name codecatalyst-ecs-build-policy \
    --policy-document file://codecatalyst-ecs-build-policy.json
```

3. 在命令輸出中，記下"arn":值，例如，arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy。你稍後需要這個 ARN。

4. 建立建置角色並將信任原則附加至該角色：

```
aws iam create-role \
      --role-name codecatalyst-ecs-build-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 將組建原則附加至組建角色：

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy
```

其中 *ARN: aw: IAM:: 111122333: ##/############codecatalyst-ecs-build-policy*的 ARN 所取代。

6. 顯示建置角色的詳細資料：

```
aws iam get-role \
      --role-name codecatalyst-ecs-build-role
```

7. 請記下角色的"Arn":值，例如，arn:aws:iam::111122223333:role/codecatalyst-ecs-build-role。你稍後需要這個 ARN。

若要建立部署原則和部署角色

1. 建立部署原則：

   a. 在 AWS CloudShell中建立部署原則檔案：

   ```
   cat > codecatalyst-ecs-deploy-policy.json
   ```

   b. 在提示符下，輸入以下代碼：

   ```
   {
       "Version": "2012-10-17",
       "Statement": [{
       "Action":[
         "ecs:DescribeServices",
         "ecs:CreateTaskSet",
         "ecs:DeleteTaskSet",
         "ecs:ListClusters",
         "ecs:RegisterTaskDefinition",
         "ecs:UpdateServicePrimaryTaskSet",
         "ecs:UpdateService",
         "elasticloadbalancing:DescribeTargetGroups",
         "elasticloadbalancing:DescribeListeners",
         "elasticloadbalancing:ModifyListener",
         "elasticloadbalancing:DescribeRules",
         "elasticloadbalancing:ModifyRule",
         "lambda:InvokeFunction",
         "lambda:ListFunctions",
         "cloudwatch:DescribeAlarms",
         "sns:Publish",
         "sns:ListTopics",
         "s3:GetObject",
         "s3:GetObjectVersion",
         "codedeploy:CreateApplication",
         "codedeploy:CreateDeployment",
         "codedeploy:CreateDeploymentGroup",
         "codedeploy:GetApplication",
         "codedeploy:GetDeployment",
         "codedeploy:GetDeploymentGroup",
         "codedeploy:ListApplications",
         "codedeploy:ListDeploymentGroups",
         "codedeploy:ListDeployments",
         "codedeploy:StopDeployment",
   ```

```
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

c. 將游標置於最後一個大括號 (}) 之後。

d. 按，**Enter**然後**Ctrl+d**保存文件並退出貓。

2. 將部署原則新增至 AWS：

```
aws iam create-policy \
```

```
    --policy-name codecatalyst-ecs-deploy-policy \
    --policy-document file://codecatalyst-ecs-deploy-policy.json
```

3. 在命令輸出中，記下部署原則的"arn":值，例如arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy。你稍後需要這個 ARN。
4. 建立部署角色並將信任原則附加至該角色：

```
aws iam create-role \
      --role-name codecatalyst-ecs-deploy-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 將部署政策附加到部署角色，其中 *arn: aw: iam:: 11112222333: ##/############ ##codecatalyst-ecs-deploy-policy*的 ARN。

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-
policy
```

6. 顯示部署角色的詳細資料：

```
aws iam get-role \
      --role-name codecatalyst-ecs-deploy-role
```

7. 請記下角色的"Arn":值，例如，arn:aws:iam::111122223333:role/codecatalyst-ecs-deploy-role。你稍後需要這個 ARN。

步驟 5：將 AWS 角色新增至 CodeCatalyst

在此步驟中，您將組建角色 (codecatalyst-ecs-build-role) 和部署 role (codecatalyst-ecs-deploy-role) 新增至空間中的 CodeCatalyst 帳戶連線。

將建置和部署角色新增至您的帳戶連線

1. 在中 CodeCatalyst，導覽至您的空間。
2. 選擇AWS 帳戶。此時會顯示帳戶連線清單。
3. 選擇代表您建立組建和部署角色之 AWS 帳戶的帳戶連線。
4. 從管理主控台選擇 [AWS 管理角色]。

將會顯示「將 IAM 角色新增至 Amazon CodeCatalyst 空間」頁面。您可能需要登入才能存取此頁面。

5. 選取 [新增您在 IAM 中建立的現有角色]。

   這時系統顯示下拉列表。此清單會顯示具有信任政策的所有 IAM 角色，其中包括`codecatalyst-runner.amazonaws.com`和`codecatalyst.amazonaws.com`服務主體。

6. 在下拉式清單中，選擇`codecatalyst-ecs-build-role`，然後選擇 [新增角色]。

   > **Note**
   >
   > 如果您看到`The security token included in the request is invalid`，可能是因為您沒有正確的權限。若要修正此問題，請使 AWS 用您建立 CodeCatalyst空間時使用的 AWS 帳戶登出以重新登入。

7. 選擇 [新增 IAM 角色]，選擇 [新增您在 IAM 中建立的現有角色]，然後在下拉式清單中選擇`codecatalyst-ecs-deploy-role`。選擇 Add role (新增角色)。

   您現在已將組建和部署角色新增至您的空間。

8. 複製 Amazon CodeCatalyst 顯示名稱的值。在建立工作流程時，您將需要此值。

步驟 6：建立來源儲存庫

在此步驟中，您可以在中建立來源儲存庫 CodeCatalyst。此儲存庫儲存自學課程的來源檔案，例如工作定義檔案。

如需來源儲存庫的詳細資訊，請參閱建立來源儲存庫。

若要建立來源儲存庫

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目，`codecatalyst-ecs-project`。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
5. 在存放庫名稱中，輸入：

   ```
   codecatalyst-ecs-source-repository
   ```

6. 選擇建立。

步驟 7：添加源文件

在本節中，您將 Hello World 源文件添加到您的 CodeCatalyst 存儲庫中，`codecatalyst-ecs-source-repository`. 它們包括：

- `index.html`檔案 — 在瀏覽器中顯示 Hello World 訊息。
- Docker 檔案 — 描述要用於 Docker 影像的基本影像，以及要套用到它的 Docker 指令。
- `taskdef.json`檔案 — 定義將工作啟動到叢集時要使用的 Docker 映像檔。

資料夾結構如下：

```
.
|— public-html
|   |— index.html
|— Dockerfile
|— taskdef.json
```

> **Note**
>
> 下面的說明告訴你如何使用 CodeCatalyst 控制台添加文件，但如果你願意，你可以使用 Git。如需詳細資訊，請參閱 複製來源儲存庫。

主題

- index.html
- Dockerfile
- 任務定義

index.html

該`index.html`文件在瀏覽器中顯示 Hello World 消息。

若要新增 index.html 檔案

1. 在主 CodeCatalyst 控台中，移至您的來源儲存庫`codecatalyst-ecs-source-repository`。

2. 在 [檔案] 中選擇 [建立檔案]。
3. 在「檔案名稱」中，輸入：

```
public-html/index.html
```

> **Important**
>
> 請務必包含public-html/前置詞，以建立相同名稱的資料夾。預期index.html會位於此資料夾中。

4. 在文字方塊中，輸入下列程式碼：

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

5. 選擇「確認」，然後再次選擇「確認」。

   會index.html新增至資public-html料夾中的儲存庫。

Dockerfile

Docker 文件描述了要使用的基本碼頭圖像以及要應用於它的碼頭命令。如需 Docker 檔案的詳細資訊，請參閱 Doc ker 檔案參考。

此處指定的碼頭文件表示使用 Apache 2.4 基本圖像（ ）`httpd`。它還包括將名為的源文件複製`index.html`到服務網頁的 Apache 服務器上的文件夾的說明。碼頭文件中的`EXPOSE`指令告訴碼頭集裝箱正在接聽端口 80。

添加碼頭文件

1. 在來源儲存庫中，選擇 [建立檔案]。
2. 在「檔案名稱」中，輸入：

   ```
   Dockerfile
   ```

   請勿包含副檔名。

   > **Important**
   >
   > Docker 文件必須駐留在存儲庫的根文件夾中。工作流程的`Docker build`命令期望它在那裡。

3. 在文字方塊中，輸入下列程式碼：

   ```
   FROM httpd:2.4
   COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
   EXPOSE 80
   ```

4. 選擇「確認」，然後再次選擇「確認」。

   碼頭文件被添加到您的存儲庫中。

任務定義

您在此步驟中新增的`taskdef.json`檔案步驟 2：將預留位置應用程式部署到 Amazon ECS與您在中指定的檔案相同，但差異如下：

這裡的任務定義不是在`image:`字段（`httpd:2.4`）中指定硬編碼的 Docker 映像名稱，而是使用幾個變量來表示圖像：和。`$REPOSITORY_URI` `$IMAGE_TAG`當您在稍後的步驟中執行工作流程時，這些變數會取代為工作流程建置動作所產生的實際值。

如需任務定義參數的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的任務定義參數。

若要新增工作定義 .json 檔案

1. 在來源儲存庫中，選擇 [建立檔案]。
2. 在「檔案名稱」中，輸入：

   taskdef.json

3. 在文字方塊中，輸入下列程式碼：

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            # The $REPOSITORY_URI and $IMAGE_TAG variables will be replaced
            # by the workflow at build time (see the build action in the
            # workflow)
            "image": $REPOSITORY_URI:$IMAGE_TAG,
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "codecatalyst-ecs-task-def"
}
```

在前面的代碼中，替換

ARN: AW: IAM:: ## ID: ##/## codecatalyst-ecs-task-execution

使用您在中若要建立作業執行角色記錄的任務執行角色的 ARN。

4. 選擇「確認」，然後再次選擇「確認」。

   taskdef.json檔案即會新增至您的儲存庫。

步驟 8：建立並執行工作流程

在此步驟中，您會建立一個工作流程，將來源檔案建置到 Docker 映像中，然後將映像部署到 Amazon ECS 叢集。此部署取代現有的 Apache 預留位置應用程式。

工作流程由下列依序執行的建置區塊組成：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 建置動作 (BuildBackend) — 在觸發器上，動作會使用 Docker 檔案建立 Docker 映像，並將映像推送至 Amazon ECR。建置動作也會taskdef.json使用正確的image欄位值更新，然後建立此檔案的輸出成品。此成品會做為下一個部署動作的輸入。

  如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
- 部署動作 (DeployToECS) — 建置動作完成後，部署動作會尋找由建置動作 (TaskDefArtifact) 產生的輸出成品，尋找其taskdef.json內部，然後將其註冊至 Amazon ECS 服務。然後，服務會遵循taskdef.json檔案中的指示，在您的 Amazon ECS 叢集內執行三個 Amazon ECS 任務 (以及相關聯的 Hello World 泊塢視窗容器)。

若要建立工作流程

1. 在 CodeCatalyst 主控台的功能窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇建立工作流程。
3. 針對來源儲存庫，選擇codecatalyst-ecs-source-repository。
4. 對於「分支」，請選擇main。
5. 選擇建立。
6. 刪除 YAML 範例程式碼。
7. 新增下列 YAML 程式碼：

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0
```

```
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in taskdef.json
        - Run: find taskdef.json -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find taskdef.json -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|
g"
```

```
        - Run: cat taskdef.json
        # The output artifact will be a zip file that contains a task definition
file.
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - taskdef.json
 DeployToECS:
    DependsOn:
      - BuildBackend
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-deploy-role
    Inputs:
      Sources: []
      Artifacts:
        - TaskDefArtifact
    Configuration:
      region: us-west-2
      cluster: codecatalyst-ecs-cluster
      service: codecatalyst-ecs-service
      task-definition: taskdef.json
```

在前面的代碼中，替換：

- 兩個例證都*codecatalyst-ecs-environment*具有您在中建立的環境名稱必要條件。
- *codecatalyst-account-connection*與您的帳戶連接的顯示名稱的兩個實例。顯示名稱可能是數字。如需詳細資訊，請參閱 步驟 5：將 AWS 角色新增至 CodeCatalyst。
- *codecatalyst-ecs-build-role*使用您在中建立之建置角色的名稱步驟 4：建立 AWS 角色。
- *############## Amazon ECR #### URI #####codecatalyst-ecs-image-repo* Value: 步驟 3：建立 Amazon ECR 映像儲存庫
- 加*# Amazon ECR #### URI####### () ##-### ECR #### URI ()*。Run: aws ecr /codecatalyst-ecs-image-repo
- *codecatalyst-ecs-deploy-role*使用您在中建立的部署角色名稱步驟 4：建立 AWS 角色。

- 使用您 AWS 的區域代*## us-west-2* 的兩個實例。如需區域代碼的清單，請參閱 AWS 一般參考.

> **Note**
>
> 如果您決定不建立組建和部署角色，請取代CodeCatalystWorkflowDevelopmentRole-*spaceName*角色名稱*codecatalyst-ecs-build-role*和*codecatalyst-ecs-deploy-role*。如需有關此角色的詳細資訊，請參閱 步驟 4：建立 AWS 角色。

> **Tip**
>
> 您可以使用 Render Amazon ECS 任務定義動作，而不是使用先前工作流程程式碼中顯示的和sed命令來更新儲存庫和映像名稱。find如需詳細資訊，請參閱 新增「渲染 Amazon ECS 任務定義」動作。

8. (選擇性) 選擇「驗證」，確定 YAML 程式碼在認可之前是有效的。
9. 選擇 Commit (遞交)。
10. 在「提交工作流程」對話方塊中，輸入以下內容：

    a. 對於提交訊息，請移除文字並輸入：

    ```
    Add first workflow
    ```

    b. 針對「儲存庫」，選擇codecatalyst-ecs-source-repository。

    c. 選擇「主要」做為「分支名稱」。

    d. 選擇 Commit (遞交)。

您現在已建立工作流程。工作流程執行會自動啟動，因為工作流程頂端定義了觸發器。具體而言，當您認可 (並推送) workflow.yaml 檔案至來源儲存庫時，觸發程序會啟動工作流程執行。

若要檢視工作流程執行進度

1. 在 CodeCatalyst 主控台的瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇您剛建立的工作流程codecatalyst-ecs-workflow。

3. 選擇BuildBackend查看構建進度。
4. 選擇 DeployToECS 以查看部署進度。

   如需檢視執行詳細資訊的詳細資訊，請參閱檢視工作流程執行狀態與詳細。

驗證部署的步驟

1. 開啟 Amazon ECS 傳統主控台，網址為 https://console.aws.amazon.com/ecs/。
2. 選擇您的叢集、codecatalyst-ecs-cluster.
3. 選擇 Tasks (任務) 索引標籤。
4. 選擇三個任務中的任何一個。
5. 在 [公用 IP] 欄位中，選擇 [開啟位址]。

   瀏覽器中會出現「Hello World」頁面，表示 Amazon ECS 服務已成功部署您的應用程式。

步驟 9：對源文件進行更改

在本節中，您會變更來源儲存庫中的index.html檔案。此變更會導致工作流程建立新的 Docker 映像、使用提交 ID 標記它、將其推送到 Amazon ECR，然後將其部署到 Amazon ECS。

若要變更 index.html 的步驟

1. 在 CodeCatalyst 主控台的導覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]，然後選擇您的存放庫codecatalyst-ecs-source-repository。
2. 選擇 public-html (下一步)，然後選擇 index.html (完成)。

   index.html會顯示的內容。
3. 選擇編輯。
4. 在第 14 行中，將Hello World文字變更為Tutorial complete!。
5. 選擇「確認」，然後再次選擇「確認」。

   認可會啟動新的工作流程執行。
6. (選擇性) 移至來源儲存庫的主頁面，選擇 [檢視認可]，然後記下index.html變更的提交 ID。
7. 觀看部署進度：

   a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。

b. 選擇codecatalyst-ecs-workflow檢視最新執行。

c. 選擇BuildBackend和 DeployToECS 以查看工作流程執行進度。

8. 確認您的應用程式已更新，如下所示：

a. 開啟 Amazon ECS 傳統主控台，網址為 https://console.aws.amazon.com/ecs/。

b. 選擇您的叢集、codecatalyst-ecs-cluster.

c. 選擇 Tasks (任務) 索引標籤。

d. 選擇三個任務中的任何一個。

e. 在 [公用 IP] 欄位中，選擇 [開啟位址]。

隨即顯示Tutorial complete!頁面。

9. (選擇性) 在中 AWS，切換至 Amazon ECR 主控台，並確認新的 Docker 映像已使用步驟 6 中的提交 ID 標記。

清除

清理本教程中使用的文件和服務，以避免被收取費用。

在中 AWS Management Console，按照以下順序進行清理：

1. 在 Amazon ECS 中，執行以下操作：
   a. 刪除codecatalyst-ecs-service。
   b. 刪除codecatalyst-ecs-cluster。
   c. 取消註冊codecatalyst-ecs-task-definition。
2. 在 Amazon ECR 中，刪除codecatalyst-ecs-image-repo。
3. 在 Amazon EC2 中，刪除codecatalyst-ecs-security-group。
4. 在 IAM 身分中心中，刪除：
   a. CodeCatalystECSUser
   b. CodeCatalystECSPermissionSet

在主 CodeCatalyst 控台中，清理如下：

1. 刪除codecatalyst-ecs-workflow。
2. 刪除codecatalyst-ecs-environment。

3. 刪除codecatalyst-ecs-source-repository。
4. 刪除codecatalyst-ecs-project。

在本教學中，您學會了如何使用 CodeCatalyst 工作流程和部署到 Amazon ECS 動作將應用程式部署到 Amazon ECS 服務。

教學課程：將應用程式部署到 Amazon EKS

在本教學中，您將學習如何使用 Amazon CodeCatalyst 工作流程、Amazon EKS 和其他一些服務，將容器化應用程式部署到 Amazon 彈性 Kubernetes 服務。 AWS 部署的應用程序是一個簡單的「你好，世界！」 建立在 Apache Web 服務器碼頭圖像上的網站。本教學將引導您完成必要的準備工作，例如設定開發機器和 Amazon EKS 叢集，然後說明如何建立工作流程以建置應用程式並將其部署到叢集。

完成初始部署之後，自學課程會指示您變更應用程式來源。此變更會建立新的 Docker 映像檔，並以新的修訂版資訊推送至您的 Docker 映像儲存庫。然後，碼頭映像的新修訂版本會部署到 Amazon EKS 中。

> **Tip**
>
> 您可以使用為您完成完整 Amazon EKS 設定的藍圖，而不是逐步完成本教學課程。您將需要使用 EKS 應用程式部署藍圖。如需詳細資訊，請參閱 使用藍圖建立專案。

主題

必要條件

在您開始本自學課程之前：

- 您需要一個具有連接 AWS 帳戶的 Amazon CodeCatalyst 空間。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。
- 在您的空間中，您需要一個空的，從頭開始的 CodeCatalyst項目，名為：

  ```
  codecatalyst-eks-project
  ```

  如需詳細資訊，請參閱 在 Amazon 中創建一個空項目 CodeCatalyst。
- 在你的項目中，你需要 CodeCatalyst 一個名為：

  ```
  codecatalyst-eks-source-repository
  ```

  如需詳細資訊，請參閱 來源儲存庫 CodeCatalyst。
- 在你的項目中，你需要一個 CodeCatalyst CI/CD 環境（不是開發環境），名為：

  ```
  codecatalyst-eks-environment
  ```

  設定此環境的方式如下：
  - 選擇任何類型，例如非生產。
  - 將您的 AWS 帳戶 Connect 到它。

  如需詳細資訊，請參閱 使用環境。

步驟 1：設定您的開發機器

本教學課程的第一個步驟是使用一些您將在本教學課程中使用的工具來設定開發機器。這些工具是：

- `eksctl`公用程式 — 用於建立叢集
- `kubectl`實用程序-的先決條件 `eksctl`
- 的 AWS CLI -也是一個先決條件 `eksctl`

您可以在現有的開發電腦上安裝這些工具 (如果有的話)，或者您可以使用基於雲的 CodeCatalyst 開發環境。 CodeCatalyst 開發環境的好處在於，它很容易啟動和拆卸，並與其他 CodeCatalyst 服務整合，讓您以更少的步驟完成本教學課程。

本教學課程假設您將使用 CodeCatalyst 開發環境。

以下說明描述了啟動 CodeCatalyst 開發環境並使用所需工具進行配置的快速方法，但是如果您需要詳細說明，請參閱：

- 本指南中的 建立開發環境。
- 在 Amazon E KS 用戶指南中安裝 kubectl。
- 在 Amazon EKS 用戶指南中安裝或升級 eksctl。
- 安裝或更新《AWS Command Line Interface 使用指南》 AWS CLI中的最新版本。

若要啟動開發環境

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目，`codecatalyst-eks-project`。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇來源儲存庫的名稱`codecatalyst-eks-source-repository`。
5. 在頂部附近選擇創建開發環境，然後選擇 AWS Cloud9 （在瀏覽器中）。
6. 確定已選取 [在現有分支和主要工作]，然後選擇 [建立]。

   您的開發環境會在新的瀏覽器索引標籤中啟動，而您的存放庫 (`codecatalyst-eks-source-repository`) 則會複製到其中。

若要安裝和設定 Kubectl

1. 在開發環境終端中，輸入：

   ```
   curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
   ```

2. 輸入：

   ```
   chmod +x ./kubectl
   ```

3. 輸入：

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
```

4. 輸入：

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

5. 輸入：

```
kubectl version --short --client
```

6. 檢查是否顯示版本。

   您現在已經安裝了kubectl。

若要安裝和設定 Eksctl

> **Note**
>
> eksctl不是嚴格要求的，因為你可以使kubectl用。但是，具eksctl有自動化大部分叢集配置的好處，因此是本教學課程建議使用的工具。

1. 在開發環境終端中，輸入：

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. 輸入：

```
sudo cp /tmp/eksctl /usr/bin
```

3. 輸入：

```
eksctl version
```

4. 檢查是否顯示版本。

   您現在已經安裝了eksctl。

若要確認是否 AWS CLI 已安裝

1. 在開發環境終端中，輸入：

```
aws --version
```

2. 檢查是否顯示版本以確認 AWS CLI 已安裝。

   完成剩餘的程序，以配置 AWS CLI 具有必要的存取權限 AWS。

若要設定 AWS CLI

您必須配置使 AWS CLI 用存取金鑰和工作階段權杖，才能授予其對 AWS 服務的存取權。下列指示提供了設定金鑰和權杖的快速方法，但如果您需要詳細指示，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈設定〉。

1. 建立 IAM 身分中心使用者，如下所示：

   a. 請登入 AWS Management Console 並開啟 AWS IAM Identity Center 主控台，網址為 https://console.aws.amazon.com/singlesignon/。

      如果您之前從未登入 IAM 身分中心，您可能需要選擇「啟用」。)

      > **Note**
      >
      > 確保您使用連接到您的 CodeCatalyst空間的登錄。 AWS 帳戶 您可以瀏覽至您的空間並選擇 AWS 帳戶索引標籤，以確認已連接哪個帳戶。如需詳細資訊，請參閱 建立支援 AWS 產生器 ID 使用者的空間。

   b. 在導覽窗格中，選擇 使用者，然後選擇 新增使用者。

   c. 在使用者名稱中，輸入：

```
codecatalyst-eks-user
```

   d. 在 [密碼] 下方，選擇 [產生一次性密碼，您可以與此使用者共用]。

   e. 在 [電子郵件地址] 和 [確認電子郵件地址] 中，輸入 IAM 身分中心尚未存在的電子郵件地址。

   f. 在名字中，輸入：

```
codecatalyst-eks-user
```

g. 在姓氏中，輸入：

```
codecatalyst-eks-user
```

h. 在 [顯示名稱] 中，保留：

```
codecatalyst-eks-user codecatalyst-eks-user
```

i. 選擇下一步。

j. 在 [新增使用者至群組] 頁面上，選擇 [下一步]。

k. 在 [檢閱並新增使用者] 頁面上，檢閱資訊並選擇 [新增使用者]。

這時系統顯示一次性密碼對話框。

l. 選擇「複製」，然後將登錄信息粘貼到文本文件中。登入資訊包含 AWS 存取入口網站 URL、使用者名稱和一次性密碼。

m. 選擇關閉。

2. 建立權限集，執行方式如下：

a. 在瀏覽窗格中，選擇 [權限集]，然後選擇 [建立權限集]。

b. 選擇預先定義的權限集，然後選取AdministratorAccess。此原則為所有人提供完整權限 AWS 服務。

c. 選擇下一步。

d. 在 [權限集名稱] 中，移除AdministratorAccess並輸入：

```
codecatalyst-eks-permission-set
```

e. 選擇下一步。

f. 在 [檢閱並建立] 頁面上，檢閱資訊並選擇 [建立]。

3. 將權限集指派給codecatalyst-eks-user，執行方式如下：

a. 在功 AWS 帳戶 能窗格中，選擇 AWS 帳戶，然後選取您目前登入的對象旁邊的核取方塊。

b. 選擇 [指派使用者或群組]。

c. 選擇 Users (使用者) 索引標籤。

d. 選取旁邊的核取方塊codecatalyst-eks-user。

e. 選擇下一步。

f. 選取旁邊的核取方塊codecatalyst-eks-permission-set。

g. 選擇下一步。

h. 複查資訊，然後選擇「提交」。

您現在已經分配codecatalyst-eks-user並codecatalyst-eks-permission-set將 AWS 帳戶它們綁定在一起。

4. 獲取codecatalyst-eks-user的訪問密鑰和會話令牌，如下所示：

a. 請確定您擁有的 AWS 存取入口網站 URL、使用者名稱和一次性密碼codecatalyst-eks-user。您應該先前將此資訊複製到文字編輯器中。

> **Note**
>
> 如果您沒有此信息，請轉到 IAM 身份中心的codecatalyst-eks-user詳細信息頁面，選擇「重置密碼」，「生成一次性密碼 [...] ，然後再次重設密碼以在螢幕上顯示資訊。

b. 登出 AWS。

c. 將 AWS 訪問門戶網址粘貼到瀏覽器的地址欄中。

d. 使用以下方式登入：

- 用戶名：

```
codecatalyst-eks-user
```

- 密碼：

one-time-password

e. 在 [設定新密碼] 中，輸入新密碼並選擇 [設定新密碼]。

螢幕上會出現一個AWS 帳戶方塊。

f. 選擇 AWS 帳戶，然後選擇您 AWS 帳戶 為其指派codecatalyst-eks-user使用者和權限集的名稱。

g. 在旁邊codecatalyst-eks-permission-set，選擇 [命令列] 或 [程式設計存取]。

h. 複製頁面中間的命令。它們看起來類似於以下內容：

```
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
```

```
export AWS_SESSION_TOKEN="session-token"
```

... 其中####是一個長隨機字符串。

5. 將存取金鑰和工作階段權杖新增至 AWS CLI，如下所示：

   a. 返回您的 CodeCatalyst 開發環境。

   b. 在終端機提示下，貼上您複製的指令。按 Enter。

   您現在已經配置了 AWS CLI 與訪問密鑰和會話令牌。您現在可以使用 AWS CLI 來完成本教學課程所需的工作。

> **Important**
>
> 如果您在本教學課程中隨時看到類似以下內容的訊息：
> `Unable to locate credentials. You can configure credentials by running "aws configure".`
> 或者：
> `ExpiredToken: The security token included in the request is expired`
> ... 這是因為您的 AWS CLI 工作階段已過期。在此情況下，請勿執行`aws configure`命令。請改用此程序步驟 4 中的指示`Obtain codecatalyst-eks-user's access key and session token`來重新整理工作階段。

步驟 2：創建一個 Amazon EKS 集群

在本節中，您將在 Amazon EKS 中建立叢集。下列指示說明使用建立叢集的快速方法`eksctl`，但如果您需要詳細說明，請參閱：

- 在 Amazon E KS 用戶指南中開始使用 eksctl

  或

- 開始使用主控台和 AWS CLI Amazon EKS 使用者指南 (本主題提供建立叢集的指`kubectl`示)

> **Note**
>
> 與 Amazon EKS CodeCatalyst 整合不支援私有叢集。

開始之前

請確定您已在開發電腦上完成下列工作：

- 安裝了該eksctl實用程序。
- 安裝了該kubectl實用程序。
- 安裝 AWS CLI 並使用訪問密鑰和會話令牌對其進行配置。

如需如何完成這些工作的資訊，請參閱步驟 1：設定您的開發機器。

建立叢集

> **Important**
>
> 請勿使用 Amazon EKS 服務的使用者界面建立叢集，因為無法正確設定叢集。使用eksctl公用程式，如下列步驟所述。

1. 移至您的開發環境。
2. 建立叢集和節點：

   ```
   eksctl create cluster --name codecatalyst-eks-cluster --region us-west-2
   ```

   其中：

   - *codecatalyst-eks-cluster*會取代為您要提供叢集的名稱。
   - *us-west-2* 已被您的地區取代。

   10-20 分鐘後，會出現類似下列內容的訊息：

   ```
   EKS cluster "codecatalyst-eks-cluster" in "us-west-2" region is ready
   ```

   > **Note**
   >
   > AWS 建立叢集時，您會看到多則waiting for CloudFormation stack訊息。這是預期的行為。

3. 確認您的叢集已成功建立：

```
kubectl cluster-info
```

您會看到類似下列內容的訊息，指出成功建立叢集：

```
Kubernetes master is running at https://long-string.gr7.us-west-2.eks.amazonaws.com
CoreDNS is running at https://long-string.gr7.us-west-2.eks.amazonaws.com/api/v1/
namespaces/kube-system/services/kube-dns:dns/proxy
```

步驟 3：建立 Amazon ECR 映像儲存庫

在本節中，您將在 Amazon Elastic Container Registry (Amazon ECR) 中創建一個私有映像存儲庫。此儲存庫會儲存教學課程的 Docker 映像檔。

如需 Amazon ECR 的詳細資訊，請參閱 Amazon 彈性容器登錄使用者指南。

若要在 Amazon ECR 中建立映像儲存庫

1. 移至您的開發環境。
2. 在 Amazon ECR 中創建一個空存儲庫：

   ```
   aws ecr create-repository --repository-name codecatalyst-eks-image-repo
   ```

   取代*codecatalyst-eks-image-repo*為您要提供給 Amazon ECR 儲存庫的名稱。

   本教學課程假設您為儲存庫命名codecatalyst-eks-image-repo。
3. 顯示 Amazon ECR 儲存庫的詳細資訊：

   ```
   aws ecr describe-repositories \
         --repository-names codecatalyst-eks-image-repo
   ```
4. 請注意“repositoryUri”:值，例如，111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo。

   稍後在將存放庫新增至工作流程時需要它。

步驟 4：添加源文件

在本節中，您會將應用程式來源檔案新增至來源儲存庫 (codecatalyst-eks-source-repository)。它們包括：

- 一個index.html文件-顯示一個「你好，世界！」 在瀏覽器中的消息。
- Docker 檔案 — 描述要用於 Docker 影像的基本影像，以及要套用到該影像的 Docker 指令。
- deployment.yaml檔案 — 定義 Kubernetes 服務和部署的 Kubernetes 資訊清單。

資料夾結構如下：

```
|— codecatalyst-eks-source-repository
   |— Kubernetes
      |— deployment.yaml
   |— public-html
   |  |— index.html
   |— Dockerfile
```

主題

- index.html
- Dockerfile
- 部署. 羊

index.html

該index.html文件顯示一個「你好，世界！」 在瀏覽器中的消息。

若要新增 index.html 檔案的步驟

1. 移至您的開發環境。
2. 在中codecatalyst-eks-source-repository，建立名為的資料夾public-html。
3. 在中/public-html，建立一個名為index.html的檔案，其內容如下：

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
```

```
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

4. 在終端提示下，輸入：

```
cd /projects/codecatalyst-eks-source-repository
```

5. 添加，提交和推送：

```
git add .
git commit -m "add public-html/index.html"
git push
```

會index.html新增至資public-html料夾中的儲存庫。

Dockerfile

Docker 文件描述了要使用的基本碼頭圖像以及要應用於它的碼頭命令。如需 Docker 檔案的詳細資訊，請參閱 Doc ker 檔案參考。

此處指定的碼頭文件表示使用 Apache 2.4 基本圖像（ ）httpd。它還包括將名為的源文件複製index.html到服務網頁的 Apache 服務器上的文件夾的說明。碼頭文件中的EXPOSE指令告訴碼頭集裝箱正在接聽端口 80。

添加碼頭文件

1. 在中codecatalyst-eks-source-repository，建立一個名為Dockerfile的檔案，其內容如下：

```
FROM httpd:2.4
COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
EXPOSE 80
```

請勿包含副檔名。

> **Important**
>
> Docker 文件必須駐留在存儲庫的根文件夾中。工作流程的`Docker build`命令期望它在那裡。

2. 添加，提交和推送：

```
git add .
git commit -m "add Dockerfile"
git push
```

碼頭文件被添加到您的存儲庫中。

部署. 羊

在本節中，您將`deployment.yaml`文件添加到存儲庫中。此`deployment.yaml`檔案是 Kubernetes 資訊清單，可定義要執行的兩種 Kubernetes 資源類型或種類：「服務」和「部署」。

- 「服務」會將負載平衡器部署到 Amazon EC2 中。負載平衡器為您提供面向網際網路的公用 URL 和標準連接埠 (連接埠 80)，您可以使用這些連接埠來瀏覽「Hello，World！」 應用程式。
- 「部署」會部署三個網繭，而且每個網繭都會包含具有「Hello，World！」的 Docker 容器 應用程式。這三個網繭會部署到您建立叢集時建立的節點上。

本教學課程中的資訊清單很簡短；不過，資訊清單可以包含任意數量的 Kubernetes 資源類型，例如網繭、工作、入口和網路原則。此外，如果部署很複雜，您可以使用多個資訊清單檔案。

若要新增部署的 .yaml 檔案

1. 在中`codecatalyst-eks-source-repository`，建立名為的資料夾`Kubernetes`。
2. 在中`/Kubernetes`，建立一個名為`deployment.yaml`的檔案，其內容如下：

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
```

```
    app: my-app
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: codecatalyst-eks-container
        # The $REPOSITORY_URI and $IMAGE_TAG placeholders will be replaced by
 actual values supplied by the build action in your workflow
        image: $REPOSITORY_URI:$IMAGE_TAG
        ports:
        - containerPort: 80
```

3. 添加，提交和推送：

```
git add .
git commit -m "add Kubernetes/deployment.yaml"
git push
```

該deployment.yaml文件被添加到您的存儲庫中的文件夾名為Kubernetes。

您現在已經添加了所有源文件。

花點時間仔細檢查您的工作，並確保將所有文件放在正確的文件夾中。資料夾結構如下：

```
|— codecatalyst-eks-source-repository
   |— Kubernetes
      |— deployment.yaml
   |— public-html
   |  |— index.html
   |— Dockerfile
```

步驟 5：建立 AWS 角色

在本節中，您會建立 CodeCatalyst 工作流程才能運作所需的 AWS IAM 角色。這些角色包括：

- 建置角色 — 授予 CodeCatalyst 建置動作 (在工作流程中) 存取 AWS 帳戶並寫入 Amazon ECR 和 Amazon EC2 的權限。
- 部署角色 — 授與「 CodeCatalyst 部署到 Kubernetes」叢集動作 (在工作流程中) 存取 AWS 帳戶和 Amazon EKS 的權限。

如需 IAM 角色的詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的 IAM 角色。

> **Note**
>
> 為了節省時間，您可以建立一個稱為角色的單一CodeCatalystWorkflowDevelopmentRole-*spaceName*角色，而不是先前列出的兩個角色。如需詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。本教學課程假設您正在建立先前列出的兩個角色。

若要建立建置和部署角色，請完成下列一系列程序。

1. 若要為這兩個角色建立信任原則

1. 移至您的開發環境。

2. 在Cloud9-*long-string*目錄中，建立名為的檔案codecatalyst-eks-trust-policy.json，其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 建立組建角色的組建原則

- 在Cloud9-*long-string*目錄中，建立名為的檔案codecatalyst-eks-build-policy.json，其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

3. 建立部署角色的部署原則

- 在Cloud9-*long-string*目錄中，建立名為的檔案codecatalyst-eks-deploy-policy.json，其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

您現在已將三個原則文件新增至您的開發環境。您的目錄結構現在看起來像這樣：

```
|— Cloud9-long-string
```

```
|– .c9
|– codecatalyst-eks-source-repository
   |– Kubernetes
   |– public-html
   |– Dockerfile
codecatalyst-eks-build-policy.json
codecatalyst-eks-deploy-policy.json
codecatalyst-eks-trust-policy.json
```

4. 若要將組建原則新增至 AWS

1. 在開發環境終端中，輸入：

```
cd /projects
```

2. 輸入：

```
aws iam create-policy \
    --policy-name codecatalyst-eks-build-policy \
    --policy-document file://codecatalyst-eks-build-policy.json
```

3. 按 Enter。
4. 在命令輸出中，記下"arn":值，例如，arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy。你稍後需要這個 ARN。

5. 若要將部署原則新增至 AWS

1. 輸入：

```
aws iam create-policy \
    --policy-name codecatalyst-eks-deploy-policy \
    --policy-document file://codecatalyst-eks-deploy-policy.json
```

2. 按 Enter。
3. 在命令輸出中，記下部署原則的"arn":值，例如arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy。你稍後需要這個 ARN。

6. 若要建立建置角色

1. 輸入：

```
aws iam create-role \
      --role-name codecatalyst-eks-build-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. 按 Enter。
3. 輸入：

```
aws iam attach-role-policy \
      --role-name codecatalyst-eks-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy
```

其中 *ARN: aw: IAM:: 111122333: ##/############codecatalyst-eks-build-policy*的 ARN 所取代。

4. 按 Enter。
5. 在終端提示下，輸入：

```
aws iam get-role \
      --role-name codecatalyst-eks-build-role
```

6. 按 Enter。
7. 請記下角色的"Arn":值，例如，arn:aws:iam::111122223333:role/codecatalyst-eks-build-role。你稍後需要這個 ARN。

7. 若要建立部署角色

1. 輸入：

```
aws iam create-role \
      --role-name codecatalyst-eks-deploy-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. 按 Enter。
3. 輸入：

```
aws iam attach-role-policy \
      --role-name codecatalyst-eks-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy
```

其中 *ARN: aw: IAM:: 111122333: ##/############codecatalyst-eks-deploy-policy*的 ARN 取代。

4. 按 Enter。
5. 輸入：

```
aws iam get-role \
      --role-name codecatalyst-eks-deploy-role
```

6. 按 Enter。
7. 請記下角色的"Arn":值，例如，arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role。你稍後需要這個 ARN。

您現在已建立建置和部署角色，並註明其 ARN。

步驟 6：將 AWS 角色新增至 CodeCatalyst

在此步驟中，您將組建角色 (codecatalyst-eks-build-role) 和部署 role (codecatalyst-eks-deploy-role) 新增至您連線至空間的角色。 AWS 帳戶 這樣就可以在您的工作流程中使用這些角色。

若要將建置和部署角色新增至 AWS 帳戶

1. 在 CodeCatalyst 主控台中，導覽至您的空間。
2. 選擇畫面頂端的 [設定]。
3. 在功能窗格中，選擇 [AWS 帳戶]。此時會顯示帳號清單。
4. 在 Amazon CodeCatalyst 顯示名稱欄中，複製建立和部署角色所 AWS 帳戶 在位置的顯示名稱。（這可能是一個數字。）在建立工作流程時，您將需要此值。
5. 選擇顯示名稱。
6. 從管理主控台選擇 [AWS 管理角色]。

   將會顯示「將 IAM 角色新增至 Amazon CodeCatalyst 空間」頁面。您可能需要登入才能存取此頁面。
7. 選取 [新增您在 IAM 中建立的現有角色]。

這時系統顯示下拉列表。此清單會顯示建置和部署角色，以及具有信任政策 (包括codecatalyst-runner.amazonaws.com和codecatalyst.amazonaws.com服務主體) 的任何其他 IAM 角色。

8. 從下拉式清單中，新增：

   - codecatalyst-eks-build-role
   - codecatalyst-eks-deploy-role

> **Note**
>
> 如果您看到The security token included in the request is invalid，可能是因為您沒有正確的權限。若要修正此問題，請使 AWS 用您建立 CodeCatalyst空間時使用的 AWS 帳戶登出以重新登入。

9. 返回 CodeCatalyst 主控台並重新整理頁面。

   建置和部署角色現在應顯示在 IAM 角色下。

   這些角色現在可用於 CodeCatalyst 工作流程。

步驟 7：更新 ConfigMap

您必須將在中建立的部署角色新增步驟 5：建立 AWS 角色至 Kubernetes ConfigMap 檔案，讓「部署至 Kubernetes」叢集動作 (在您的工作流程中) 能夠存取叢集並與叢集互動。您可以使用eksctl或kubectl執行此工作。

若要使用 eksctl 來設定庫伯內特檔 ConfigMap

- 在開發環境終端中，輸入：

```
eksctl create iamidentitymapping --cluster codecatalyst-eks-cluster --
arn arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role --group
 system:masters --username codecatalyst-eks-deploy-role --region us-west-2
```

其中：

- *codecatalyst-eks-cluster*會取代為 Amazon EKS 叢集的叢集名稱。

- *arn: aw: IAM:: 1111223333: ##/############codecatalyst-eks-deploy-role*的 ARN 所取代。步驟 5：建立 AWS 角色
- *codecatalyst-eks-deploy-role*(旁邊的--username) 會取代為您在中建立之部署角色的名稱步驟 5：建立 AWS 角色。

> **Note**
>
> 如果您決定不建立部署角色，請*codecatalyst-eks-deploy-role*以角CodeCatalystWorkflowDevelopmentRole-*spaceName*色的名稱取代。如需有關此角色的詳細資訊，請參閱 步驟 5：建立 AWS 角色。

- *us-west-2* 已被您的地區取代。

如需有關此命令的詳細資訊，請參閱管理 IAM 使用者和角色。

會出現類似下列內容的訊息：

```
2023-06-09 00:58:29 [#]  checking arn arn:aws:iam::111122223333:role/codecatalyst-
eks-deploy-role against entries in the auth ConfigMap
2023-06-09 00:58:29 [#]  adding identity "arn:aws:iam::111122223333:role/
codecatalyst-eks-deploy-role" to auth ConfigMap
```

使用 Kubectl 規劃庫伯內特斯 ConfigMap 檔案的步驟

1. 在開發環境終端中，輸入：

```
kubectl edit configmap -n kube-system aws-auth
```

該 ConfigMap 文件出現在屏幕上。

2. 添加紅色斜體文本：

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
```

```
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/eksctl-codecatalyst-eks-cluster-n-
NodeInstanceRole-16BC456ME6YR5
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - system:masters
      rolearn: arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role
      username: codecatalyst-eks-deploy-role
  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2023-06-08T19:04:39Z"
  managedFields:
  ...
```

其中：

- *arn: aw: IAM:: 1111223333: ##/############codecatalyst-eks-deploy-role*的 ARN 所取代。步驟 5：建立 AWS 角色
- *codecatalyst-eks-deploy-role*(旁邊的username:) 會取代為您在中建立之部署角色的名稱步驟 5：建立 AWS 角色。

> **Note**
>
> 如果您決定不建立部署角色，請*codecatalyst-eks-deploy-role*以角CodeCatalystWorkflowDevelopmentRole-*spaceName*色的名稱取代。如需有關此角色的詳細資訊，請參閱 步驟 5：建立 AWS 角色。

如需詳細資訊，請參閱 Amazon EKS 使用者指南中的啟用對叢集的 IAM 主體存取權限。

您現在已授予部署角色，並透過擴充「部署到 Amazon EKS」動作，即可獲得 Kubernetes 叢集的system:masters許可。

步驟 8：建立並執行工作流程

在此步驟中，您會建立一個工作流程，將來源檔案建置到 Docker 映像中，然後將映像部署到 Amazon EKS 叢集中的樹狀網繭中。

工作流程由下列依序執行的建置區塊組成：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 建置動作 (BuildBackend) — 在觸發器上，動作會使用 Docker 檔案建立 Docker 映像，並將映像推送至 Amazon ECR。建置動作也會以正確的值更新`deployment.yaml`檔案中的`$REPOSITORY_URI`和`$IMAGE_TAG`變數，然後建立此檔案和`Kubernetes`資料夾中任何其他檔案的輸出成品。在本自學課程中，`Kubernetes`資料夾中唯一的檔案是，`deployment.yaml`但您可以包含更多檔案。成品會用作下一個部署動作的輸入。

  如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
- 部署動作 (DeployToEKS) — 建置動作完成後，部署動作會尋找建置動作 (`Manifests`) 所產生的輸出成品，並尋找其中的`deployment.yaml`檔案。然後，該操作按照`deployment.yaml`文件中的說明運行三個 Pods-每個包含一個「你好，世界！」 碼頭容器 — 在您的 Amazon EKS 集群中。

若要建立工作流程

1. 前往主 CodeCatalyst 控台。
2. 導航到您的項目（`codecatalyst-eks-project`）。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇建立工作流程。
5. 針對來源儲存庫，選擇`codecatalyst-eks-source-repository`。
6. 對於「分支」，請選擇`main`。
7. 選擇建立。
8. 刪除 YAML 範例程式碼。
9. 新增下列 YAML 程式碼以建立新的工作流程定義檔案：

> **Note**
>
> 若要取得有關工作流程定義檔的更多資訊，請參閱工作流定義參考。

```
Name: codecatalyst-eks-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in deployment.yaml
```

```
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|g"
        - Run: cat Kubernetes/*
        # The output artifact will be a zip file that contains Kubernetes manifest
 files.
    Outputs:
      Artifacts:
        - Name: Manifests
          Files:
            - "Kubernetes/*"
  DeployToEKS:
    DependsOn:
      - BuildBackend
    Identifier: aws/kubernetes-deploy@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-deploy-role
    Inputs:
      Artifacts:
        - Manifests
    Configuration:
      Namespace: default
      Region: us-west-2
      Cluster: codecatalyst-eks-cluster
      Manifests: Kubernetes/
```

在前面的代碼中，替換：

- 兩個例證都*codecatalyst-eks-environment*具有您在中建立的環境名稱必要條件。
- *codecatalyst-account-connection*與您的帳戶連接的顯示名稱的兩個實例。顯示名稱可能是數字。如需詳細資訊，請參閱 步驟 6：將 AWS 角色新增至 CodeCatalyst。
- *codecatalyst-eks-build-role*使用您在中建立之建置角色的名稱步驟 5：建立 AWS 角色。
- *############## Amazon ECR #### URI #####codecatalyst-eks-image-repo* Value: 步驟 3：建立 Amazon ECR 映像儲存庫
- 加*# Amazon ECR #### URI####### () ##-### ECR #### URI (*)。Run: aws ecr /codecatalyst-eks-image-repo

- *codecatalyst-eks-deploy-role*使用您在中建立的部署角色名稱步驟 5：建立 AWS 角色。
- 使用您 AWS 的區域代*## us-west-2* 的兩個實例。如需區域代碼的清單，請參閱 AWS 一般參考.

> **Note**
>
> 如果您決定不建立組建和部署角色，請取代CodeCatalystWorkflowDevelopmentRole-*spaceName*角色名稱*codecatalyst-eks-build-role*和*codecatalyst-eks-deploy-role*。如需有關此角色的詳細資訊，請參閱 步驟 5：建立 AWS 角色。

10. (選擇性) 選擇「驗證」，確定 YAML 程式碼在認可之前是有效的。
11. 選擇 Commit (遞交)。
12. 在「提交工作流程」對話方塊中，輸入以下內容：

    a. 對於提交訊息，請移除文字並輸入：

    ```
    Add first workflow
    ```

    b. 針對「儲存庫」，選擇codecatalyst-eks-source-repository。

    c. 選擇「主要」做為「分支名稱」。

    d. 選擇 Commit (遞交)。

    您現在已建立工作流程。工作流程執行會自動啟動，因為工作流程頂端定義了觸發器。具體而言，當您認可 (並推送) workflow.yaml 檔案至來源儲存庫時，觸發程序會啟動工作流程執行。

若要檢視工作流程執行進度

1. 在 CodeCatalyst 主控台的瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇您剛建立的工作流程codecatalyst-eks-workflow。
3. 選擇BuildBackend查看構建進度。
4. 選擇 DeployToEKS 以查看部署進度。

   如需檢視執行詳細資訊的詳細資訊，請參閱檢視工作流程執行狀態與詳細。

驗證部署的步驟

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在左側的底部附近，選擇負載平衡器。
3. 選取在 Kubernetes 部署中建立的負載平衡器。如果您不確定要選擇哪個負載平衡器，請在「標籤」索引標籤下尋找下列標籤：
   - `kubernetes.io/service-name`
   - `kubernetes.io/cluster/ekstutorialcluster`
4. 選取正確的負載平衡器後，選擇 [說明] 索引標籤。
5. 將 DNS 名稱值複製並貼到瀏覽器的網址列中。

   在 '你好, 世界!' 網頁會出現在瀏覽器中，表示您已成功部署應用程式。

步驟 9：對源文件進行更改

在本節中，您會變更來源儲存庫中的`index.html`檔案。此變更會導致工作流程建立新的 Docker 映像、使用提交 ID 標記它、將其推送到 Amazon ECR，然後將其部署到 Amazon ECS。

若要變更 index.html 的步驟

1. 移至您的開發環境。
2. 在終端機提示符下，切換到源存儲庫：

```
cd /projects/codecatalyst-eks-source-repository
```

3. 提取最新的工作流程變更：

```
git pull
```

4. 打開 `codecatalyst-eks-source-repository/public-html/index.html`.
5. 在第 14 行中，將`Hello, World!`文字變更為`Tutorial complete!`。
6. 添加，提交和推送：

```
git add .
git commit -m "update index.html title"
git push
```

工作流程執行會自動啟動。

7. (選擇性) 輸入：

```
git show HEAD
```

請記下index.html變更的提交 ID。此提交 ID 將標記為 Docker 映像檔，該映像檔將由您剛開始的工作流程執行部署。

8. 觀看部署進度：

   a. 在 CodeCatalyst 主控台的功能窗格中，選擇 CI/CD，然後選擇 [工作流程]。

   b. 選擇codecatalyst-eks-workflow檢視最新執行。

   c. 選擇BuildBackend和 DeployToEKS 以查看工作流程執行進度。

9. 確認您的應用程式已更新，如下所示：

   a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

   b. 在左側的底部附近，選擇負載平衡器。

   c. 選取在 Kubernetes 部署中建立的負載平衡器。

   d. 將 DNS 名稱值複製並貼到瀏覽器的網址列中。

   「教程完成！」 網頁會出現在瀏覽器中，表示您已成功部署應用程式的新修訂版本。

10. (選擇性) 在中 AWS，切換至 Amazon ECR 主控台，並確認新的 Docker 映像已使用本程序步驟 7 的提交 ID 標記。

清除

您應該清理環境，這樣您就不會針對本教學所使用的儲存體和運算資源不必要的費用。

清理方式

1. 刪除您的叢集：

   - 在開發環境終端中，輸入：

```
eksctl delete cluster --region=us-west-2 --name=codecatalyst-eks-cluster
```

其中：

- *us-west-2* 已被您的地區取代。
- *codecatalyst-eks-cluster*會以您建立的叢集名稱取代。

5-10 分鐘後，會刪除叢集和相關資源，包括但不限於 AWS CloudFormation 堆疊、節點群組 (在 Amazon EC2 中) 和負載平衡器。

> **Important**
>
> 如果命`eksctl delete cluster`令不起作用，您可能需要重新整理 AWS 認證或`kubectl`認證。如果您不確定要重新整理哪些認證，請先重新整理 AWS 認證。若要重新整理 AWS 認證，請參閱如何修復「找不到憑據」和「ExpiredToken」錯誤？。若要重新整理`kubectl`認證，請參閱如何修復「無法連接到服務器」錯誤？。

2. 在 AWS 主控台中，進行清理，如下所示：

   1. 在 Amazon ECR 中，刪除`codecatalyst-eks-image-repo`。
   2. 在 IAM 身分中心中，刪除：
      a. `codecatalyst-eks-user`
      b. `codecatalyst-eks-permission-set`
   3. 在 IAM 中，刪除：
      - `codecatalyst-eks-build-role`
      - `codecatalyst-eks-deploy-role`
      - `codecatalyst-eks-build-policy`
      - `codecatalyst-eks-deploy-policy`

3. 在 CodeCatalyst 主控台中，進行清理，如下所示：

   1. 刪除`codecatalyst-eks-workflow`。
   2. 刪除`codecatalyst-eks-environment`。
   3. 刪除`codecatalyst-eks-source-repository`。
   4. 刪除您的開發環境。
   5. 刪除`codecatalyst-eks-project`。

在本教學中，您學習了如何使用 CodeCatalyst 工作流程和「部署到 Kubernetes」叢集動作將應用程式部署到 Amazon EKS 服務。

新增「部署 AWS CloudFormation 堆疊」動作

> **Tip**
>
> 如需示範如何使用「部署 AWS CloudFormation 堆疊」動作的教學課程，請參閱教學課程：使用部署無伺服器應用程式 AWS CloudFormation。

本節說明如何將「部署 AWS CloudFormation 堆疊」動作新增至工作流程。此動作會 AWS 根據您提供的範本在中建立 CloudFormation 堆疊。該模板可以是：

- AWS CloudFormation 範本 — 如需詳細資訊，請參閱使用 AWS CloudFormation 範本。
- AWS SAM 範本-如需詳細資訊，請參閱 AWS Serverless Application Model (AWS SAM) 規格。

> **Note**
>
> 若要使用 AWS SAM 範本，您必須先使用`sam package`作業來封裝 AWS SAM 應用程式。如需說明如何在 Amazon CodeCatalyst 工作流程中自動執行此封裝的教學課程，請參閱教學課程：使用部署無伺服器應用程式 AWS CloudFormation。

如果堆疊已存在，動作會執行 CloudFormation `CreateChangeSet`作業，然後執行`ExecuteChangeSet`作業。然後，動作會等待部署變更，並根據結果將自己標示為失敗成功。

如果您已經擁有包含要部署之資源的 AWS CloudFormation 或 AWS SAM 範本，或者您計劃使用 AWS SAM 和之類的工具自動產生一個或範本，作為工作流程建置動作的一部分，請使用「部署 AWS CloudFormation 堆疊」動作AWS Cloud Development Kit (AWS CDK)。

您可以使用的範本沒有任何限制，無論您可以在中編寫 CloudFormation 或 AWS SAM 搭配「部署 AWS CloudFormation 堆疊」動作使用的範本。

Visual

若要使用視覺化編輯器新增「部署 AWS CloudFormation 堆疊」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署 AWS CloudFormation 堆疊」動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇「部署 AWS CloudFormation 堆疊」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「部署 AWS CloudFormation堆疊」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增「部署 AWS CloudFormation 堆疊」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。

7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署 AWS CloudFormation 堆疊」動作，然後執行下列其中一項作業：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或
   - 選擇「部署 AWS CloudFormation 堆疊」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「部署AWS CloudFormation堆疊」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

「部署 AWS CloudFormation 堆疊」動作所產生的變數

當「部署 AWS CloudFormation 堆疊」動作執行時，它會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「部署AWS CloudFormation堆疊」動作變數中的預先定義的變數清單。

「部署 AWS CloudFormation 堆疊」動作定義

部署 AWS CloudFormation 堆疊動作會定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「部署AWS CloudFormation堆疊」動作參考中的工作流定義參考。

新增「部署到 Amazon ECS」動作

> **Tip**
>
> 如需說明如何使用「部署到 Amazon ECS」動作的教學課程，請參閱教學課程：將應用程式部署到 Amazon ECS。

> **Tip**
>
> 如需「部署到 Amazon ECS」動作的工作範例，請使用 Node.js API AWS Fargate或含 AWS Fargate藍圖的 Java API 建立專案。如需詳細資訊，請參閱 使用藍圖建立專案。

本節說明如何將「部署到 Amazon ECS」動作新增至您的工作流程。此動作會註冊您提供的任務定義檔案。註冊後，任務定義會由 Amazon ECS 叢集中執行的 Amazon ECS 服務實例化。「實例化任務定義」等同於將應用程式部署到 Amazon ECS。

若要使用此動作，您必須準備好現有的 Amazon ECS 叢集、服務和任務定義檔案。

如需 Amazon ECS 的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南。

Visual

若要使用視覺化編輯器新增「部署到 Amazon ECS」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署到 Amazon ECS」動作，然後執行下列其中一個動作：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇部署到 Amazon ECS。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「部署到 Amazon ECS」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

使用 YAML 編輯器新增「部署到 Amazon ECS」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署到 Amazon ECS」動作，然後執行下列其中一個動作：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇部署到 Amazon ECS。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「部署到 Amazon ECS」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

「部署到 Amazon ECS」動作所產生的變數

當「部署到 Amazon ECS」動作執行時，它會產生可用於後續工作流程動作的變數。如需詳細資訊，請參閱「部署到 Amazon ECS」動作變數中的預先定義的變數清單。

「部署到 Amazon ECS」動作定義

「部署到 Amazon ECS」動作定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「部署到 Amazon ECS」動作參考中的工作流定義參考。

新增「部署至 Kubernetes 叢集」動作

> **Tip**
>
> 如需說明如何使用「部署至 Kubernetes」叢集動作的教學課程，請參閱。教學課程：將應用程式部署到 Amazon EKS

本節說明如何將「部署至 Kubernetes」叢集動作新增至您的工作流程。此動作會使用您提供的一或多個 Kubernetes 資訊清單檔案，將您的應用程式部署到您在 Amazon Elastic Kubernetes Service (EKS) 中設定的 Kubernetes 叢集。如需資訊清單範例，請參閱部署. 羊中的教學課程：將應用程式部署到 Amazon EKS。

如需有關 Kubernetes 的詳細資訊，請參閱 Kubernetes 文件。

有關 Amazon EKS 的更多信息，請參閱什麼是 Amazon E KS？ 在 Amazon EKS 用戶指南。

運作方式

「部署到 Kubernetes」叢集的運作方式如下：

1. 在執行階段，此動作會將 Kubernetes `kubectl` 公用程式安裝至執行動作的 CodeCatalyst計算機器。此動作會設定`kubectl`為指向您在設定動作時提供的 Amazon EKS 叢集。接下來，該`kubectl`實用程序是運行`kubectl apply`命令所必需的。
2. 此動作會執行`kubectl apply -f` *`my-manifest.yaml`*命令，該命令會執行 *`my-manifest.yaml`* 中的指示，將您的應用程式部署為一組容器和網繭到已設定的叢集中。如需有關此命令的詳細資訊，請參閱 Kubernetes 參考文件中的 kubectl 套用主題。

必要條件

若要使用此動作，您必須準備好下列項目：

> **Tip**
>
> 若要快速設定這些先決條件，請遵循中的指示教學課程：將應用程式部署到 Amazon EKS。

- Amazon EKS 中的一個庫伯尼特群集。如需叢集的相關資訊，請參閱 Amazon EKS 使用者指南中的 Amazon EKS 叢集。
- 至少一個 Docker 文件描述如何將您的應用程序組裝到 Docker 映像中。有關碼頭文件的更多信息，請參閱碼頭文件參考。
- 至少有一個 Kubernetes 資訊清單檔案，在 Kubernetes 文件中稱為組態檔案或組態。如需詳細資訊，請參閱 Kubernetes 文件中的管理資源。
- 一種 IAM 角色，可讓「部署到 Kubernetes」叢集動作存取您的 Amazon EKS 叢集並與之互動。如需詳細資訊，請參閱 「部署至 Kubernetes 叢集」動作參考 中的「Role」主題。

  建立此角色之後，您必須將其新增至：

  - 您的庫伯內特斯 ConfigMap 檔案。若要了解如何將角色新增至 ConfigMap 檔案，請參閱 Amazon EKS 使用者指南中的啟用 IAM 主體存取叢集。
  - CodeCatalyst。要了解如何將 IAM 角色添加到 CodeCatalyst，請參閱將 IAM 角色新增至帳戶連線。
- CodeCatalyst 空間、專案和環境。空間和環境都必須連線到您要部署應用程式的 AWS 帳戶。如需詳細資訊，請參閱建立支援 AWS 產生器 ID 使用者的空間、在 Amazon 中創建一個空項目 CodeCatalyst及使用環境。
- 支援的來源儲存庫 CodeCatalyst。儲存庫會儲存您的應用程式來源檔案、Docker 檔案和 Kubernetes 資訊清單。如需詳細資訊，請參閱 來源儲存庫 CodeCatalyst。

新增「部署至叢集」動作

使用下列指示將「部署至叢集」動作新增至您的工作流程。

Visual

若要使用視覺化編輯器新增「部署至 Kubernetes 叢集」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署到 Kubernetes」叢集動作，然後執行下列其中一項作業：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇部署至叢集。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「部署至 Kubernetes 叢集」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增「部署至 Kubernetes 叢集」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。

4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋「部署到 Kubernetes」叢集動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇部署至叢集。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「部署至 Kubernetes 叢集」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

「部署至 Kubernetes 叢集」動作所產生的變數

當「部署到 Kubernetes」叢集動作執行時，它會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「部署至 Kubernetes 叢集」動作變數中的預先定義的變數清單。

「部署至 Kubernetes 叢集」動作定義

部署至 Kubernetes 叢集動作會定義為工作流程定義檔案中的一組 YAML 內容。若要取得有關這些性質的資訊，請參閱「部署至 Kubernetes 叢集」動作參考中的工作流定義參考。

新增「AWS CDK 部署」動作

本節說明如何將AWS CDK 部署動作新增至工作流程。部AWS CDK 署動作會將您的 AWS Cloud Development Kit (AWS CDK) 應用程式合成並部署到中。 AWS如果您的應用程式已存在於中 AWS，動作會在必要時更新該應用程式。

如需有關使用撰寫應用程式的一般資訊 AWS CDK，請參閱什麼是 AWS CDK? 在AWS Cloud Development Kit (AWS CDK) 開發人員指南中。

主題

何時使用此動作

如果您已使用開發應用程式 AWS CDK，且現在想要將其自動部署為自動持續整合與傳遞 (CI/CD) 工作流程的一部分，請使用此動作。例如，每當有人合併與您的 AWS CDK 應用程序源相關的提取請求時，您可能希望自動部署您的 AWS CDK 應用程序。

運作方式

部AWS CDK 署的工作方式如下：

1. 在執行階段，如果您指定的動作版本 1.0.12 或更早版本，動作會將最新的 CDK CLI (也稱為 AWS CDK Tookit) 下載至組建映像檔。 CodeCatalyst

   如果您指定版本 1.0.13 或更新版本，則動作隨附於特定版本的 CDK CLI，因此不會進行下載。

2. 此動作會使用 CDK CLI 來執行命`cdk deploy`令。此命令可合成您的 AWS CDK 應用程序並將其部署到. AWS有關此命令的更多信息，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南中的AWS CDK 工具包（cdk 命令）主題。

「AWS CDK 部署」動作所使用的 CDK CLI 版本

下表顯示不同版本的AWS CDK 部署動作預設會使用哪個版本的 CDK CLI。

> **Note**
>
> 您可能可以覆寫預設值。如需詳細資訊，請參閱「AWS CDK 部署」動作參考 中的 CdkCliVersion。

「AWS CDK 部署」動作版本	AWS CDK CLI 版本
1.0.0 —	最新
1.0.13 或更高版本	2.99.1

動作可以部署多少堆疊？

部AWS CDK 署只能部署單一堆疊。如果您的 AWS CDK 應用程式包含多個堆疊，則必須使用巢狀堆疊建立父系堆疊，並使用此動作部署父項。

必要條件

在您可以使用AWS CDK 部署動作之前，請先完成下列工作：

1. 準備好 AWS CDK 應用程式。您可以使 AWS CDK 用 AWS CDK v1 或 v2 撰寫您的應用程式，使用 AWS CDK. 確保您的 AWS CDK 應用程序文件在以下位置可用：
   - 來 CodeCatalyst 源儲存庫，或
   - 由另一個工作流程動作產生的 CodeCatalyst 輸出人工
2. 引導您的 AWS 環境。要引導，您可以：
   - 使用AWS Cloud Development Kit (AWS CDK) 開發人員指南中如何引導中描述的其中一種方法。
   - 使用AWS CDK 啟動程序動作。您可以在與AWS CDK 部署相同的工作流程中新增此動作，或在不同的工作流程中新增此動作。只需確保引導操作至少運行一次之前運行AWS CDK 部署操作，以便必要的資源就位。如需AWS CDK 啟動程序動作的詳細資訊，請參閱添加「AWS CDK 引導」操作。

     如需有關啟動載入的詳細資訊，請參閱開發人員指南中的啟動載入。AWS Cloud Development Kit (AWS CDK)

範例工作流程

下列範例工作流程包含AWS CDK 部署動作以及啟動程AWS CDK 序動作。工作流程由下列依序執行的建置區塊組成：

> **Note**
>
> 下列工作流程範例僅供說明用途，如果沒有其他組態，將無法運作。它旨在為您提供一個示例，說明使用AWS CDK 部署動作配置工作流程時可能會是什麼樣子。

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。此存儲庫包含您的 AWS CDK 應用程序。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 啟動程AWS CDK 序動作 (CDKBootstrap) — 在觸發器上，動作會將啟動程CDKToolkit序堆疊部署到 AWS。如果CDKToolkit堆疊已存在於環境中，則必要時將其升級；否則，不會執行任何動作，且動作會標示為成功。
- AWS CDK 部署動作 (AWS CDK Deploy) — 完成AWS CDK 啟動程序動作後，AWS CDK 部署動作會將您的 AWS CDK 應用程式程式碼合成到 AWS CloudFormation 範本中，並將範本中定義的堆疊部署到。 AWS

```
Name: codecatalyst-cdk-deploy-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  CDKBootstrap:
    Identifier: aws/cdk-bootstrap@v1
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-bootstrap-role
    Configuration:
```

```
      Region: us-west-2

  CDKDeploy:
    Identifier: aws/cdk-deploy@v1
    DependsOn:
      - CDKBootstrap
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-deploy-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: my-app-stack
      Region: us-west-2
```

新增「AWS CDK 部署」動作

使用下列指示將AWS CDK 部署動作新增至您的工作流程。

必要條件

在開始之前，請確定您已完成中所述的工作必要條件。

Visual

若要使用視覺化編輯器新增「AWS CDK 部署」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。

9. 搜尋AWS CDK 部署動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇「AWS CDK 部署」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「AWS CDK 部署」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

> **Note**
>
> 如果您的AWS CDK 部署動作失敗並出`npm install`現錯誤，請參我如何解決「npm 安裝」錯誤？閱以取得有關如何修正錯誤的資訊。

YAML

若要使用 YAML 編輯器新增「AWS CDK 部署」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋AWS CDK 部署動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

或

- 選擇「AWS CDK 部署」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
  - (選擇性) 選擇「下載」以檢視動作的原始程式碼。
  - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「AWS CDK 部署」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

> **Note**
>
> 如果您的AWS CDK 部署動作失敗並出`npm install`現錯誤，請參我如何解決「npm 安裝」錯誤？閱以取得有關如何修正錯誤的資訊。

「AWS CDK 部署」動作產生的變數

AWS CDK 部署動作執行時，它會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「AWS CDK部署」動作變數中的預先定義的變數清單。

「AWS CDK 部署」動作定義

部AWS CDK 署動作會定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「AWS CDK 部署」動作參考中的工作流定義參考。

使用部署

本節說明如何在 Amazon 中管理、監控和設定部署 CodeCatalyst。

主題

- 部署應用程式或資源
- 重新執行部署
- 檢視部署狀態、認可和提取要求
- 檢視部署記錄
- 設定復原
- 顯示已部署應用程式的 URL

- 移除部署目標

部署應用程式或資源

若要使用 Amazon 部署應用程式或資源 CodeCatalyst，您必須使用 CodeCatalyst 工作流程。此工作流程必須包含至少一個可以部署某些項目的動作。例如，如果您想要部署 AWS CloudFormation 堆疊，您可以建立包含「部署 AWS CloudFormation 堆疊」動作的工作流程。

如需工作流程的相關詳細資訊，請參閱 使用中的工作流程來建置、測試和部署 CodeCatalyst。如需動作的詳細資訊，請參閱使用動作。

重新執行部署

若要在 Amazon 中重新執行部署 CodeCatalyst，請重新執行其相關聯的工作流程。如需說明，請參閱啟動工作流程執行。

檢視部署狀態、認可和提取要求

您可以檢視有關 Amazon 中部署的下列資訊 CodeCatalyst：

- 部署活動，包括部署狀態、開始時間、結束時間、歷史記錄和事件持續時間。
- 堆疊名稱 AWS 區域、上次更新時間和關聯的工作流程。
- 提交和提取請求。
- 動作特定資訊，例如 CloudFormation 事件和輸出。

您可以從工作流程、環境或工作流程動作開始檢視部署資訊。

從工作流程開始檢視部署資訊的步驟

- 移至部署應用程式的工作流程執行。如需說明，請參閱檢視工作流程執行狀態與詳細。

若要檢視從環境開始的部署資訊

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 選擇部署堆疊的環境，例如`Production`。

5. 選擇「部署活動」以檢視堆疊的部署歷史記錄、部署狀態 (例如「成功」或「失敗」)，以及其他部署相關資訊。
6. 選擇部署目標以檢視部署至環境中之堆疊、叢集或其他目標的相關資訊。您可以檢視堆疊名稱、地區、提供者和識別碼等資訊。

若要檢視從動作開始的部署資訊

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 在工作流程圖中，選擇部署應用程式的工作流程動作。例如，您可以選擇DeployCloudFormationStack。
6. 檢閱右窗格中的內容，以取得動作特定的部署資訊。

檢視部署記錄

您可以檢視與特定部署動作相關的日誌，以便疑難排解 Amazon 中的問題 CodeCatalyst。

您可以從工作流程或環境開始檢視記錄。

若要檢視從工作流程開始部署動作的記錄

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇「執行」。
6. 選擇部署應用程式的工作流程執行。
7. 在工作流程圖中，選擇您要檢視其記錄的動作。
8. 選擇記錄檔索引標籤，然後展開區段以顯示記錄訊息。
9. 若要檢視更多記錄檔，請選擇 [摘要] 索引標籤，然後選擇 [檢視位置 CloudFormation (如果可用的話)] 以檢視其他記錄。您可能需要登入 AWS。

若要檢視從環境開始部署動作的記錄

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 選擇部署應用程式的環境。
5. 在「部署」活動中，找到「工作流程執行 ID」欄，然後選擇部署堆疊的工作流程執行。
6. 在工作流程圖中，選擇您要檢視其記錄的動作。
7. 選擇記錄檔索引標籤，然後展開區段以顯示記錄訊息。
8. 若要檢視更多記錄檔，請選擇 [摘要] 索引標籤，然後選擇 [檢視位置 CloudFormation (如果可用的話)] 以檢視其他記錄。您可能需要登入 AWS。

設定復原

根據預設，如果「部署 AWS CloudFormation 堆疊」動作失敗，就會造成堆疊回復 AWS CloudFormation 到上次已知的穩定狀態。您可以變更行為，以便不僅在動作失敗時進行復原，還可以在發生指定的 Amazon CloudWatch 警示時進行復原。如需有關 CloudWatch 警示的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 Amazon CloudWatch 警示。

您也可以變更預設行為，以便在動作失敗時 CloudFormation 不會復原堆疊。

請使用下列指示來設定復原。

> **Note**
>
> 您無法手動啟動復原。

Visual

開始之前

1. 請確定您的工作流程包含正常運作的「部署 AWS CloudFormation 堆疊」動作。如需詳細資訊，請參閱 新增「部署 AWS CloudFormation 堆疊」動作。
2. 在 [部署 AWS CloudFormation 堆疊] 動作的 [堆疊角色-選用] 欄位中指定的角色中，請確定包含CloudWatchFullAccess權限。如需有關使用適當權限建立此角色的資訊，請參閱步驟 2：建立 AWS 角色。

設定「部署 AWS CloudFormation 堆疊」動作的復原警示

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇包含「部署 AWS CloudFormation 堆疊」動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇您的部署 AWS CloudFormation 堆疊動作。
8. 在詳細資料窗格中，選擇「組態」。
9. 展開底部的 [進階]。
10. 在 [監控警示 ARN] 下方，選擇 [新增警示]。
11. 在下列欄位中輸入資訊。

    - ARN 報警

      指定要用作回滾觸發器的 Amazon CloudWatch 警報的 Amazon 資源名稱 (ARN)。例如 `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`。您最多可以有五個回復觸發器。

      > **Note**
      > 如果您指定 CloudWatch 警報 ARN，您還需要配置其他權限以啟用該動作才能訪問 CloudWatch。如需詳細資訊，請參閱 設定復原。

    - 監測時間

      指定從 0 到 180 分鐘的時間長度，在此期間 CloudFormation 監視指定的警報。部署所有堆疊資源之後，就會開始監視。如果警示在指定的監視時間內發生，則部署會失敗，並 CloudFormation 復原整個堆疊作業。

      預設值：0。 CloudFormation 只會在部署堆疊資源時監控警示，而不會在之後監視警示。

YAML

設定「部署 AWS CloudFormation 堆疊」動作的復原觸發程序

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇包含「部署 AWS CloudFormation 堆疊」動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在 YAML 程式碼中新增`monitor-alarm-arns`和`monitor-timeout-in-minutes`屬性，以新增回復觸發程序。如需每個性質的說明，請參閱「部署AWS CloudFormation堆疊」動作參考。
8. 在部署 AWS CloudFormation 堆疊動作`role-arn`屬性中指定的角色中，請確定包含CloudWatchFullAccess權限。如需有關使用適當權限建立此角色的資訊，請參閱步驟 2：建立 AWS 角色。

Visual

關閉「部署 AWS CloudFormation 堆疊」動作的復原

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇包含「部署 AWS CloudFormation 堆疊」動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇您的部署 AWS CloudFormation 堆疊動作。
8. 在詳細資料窗格中，選擇「組態」。
9. 展開底部的 [進階]。
10. 開啟 [停用復原]。

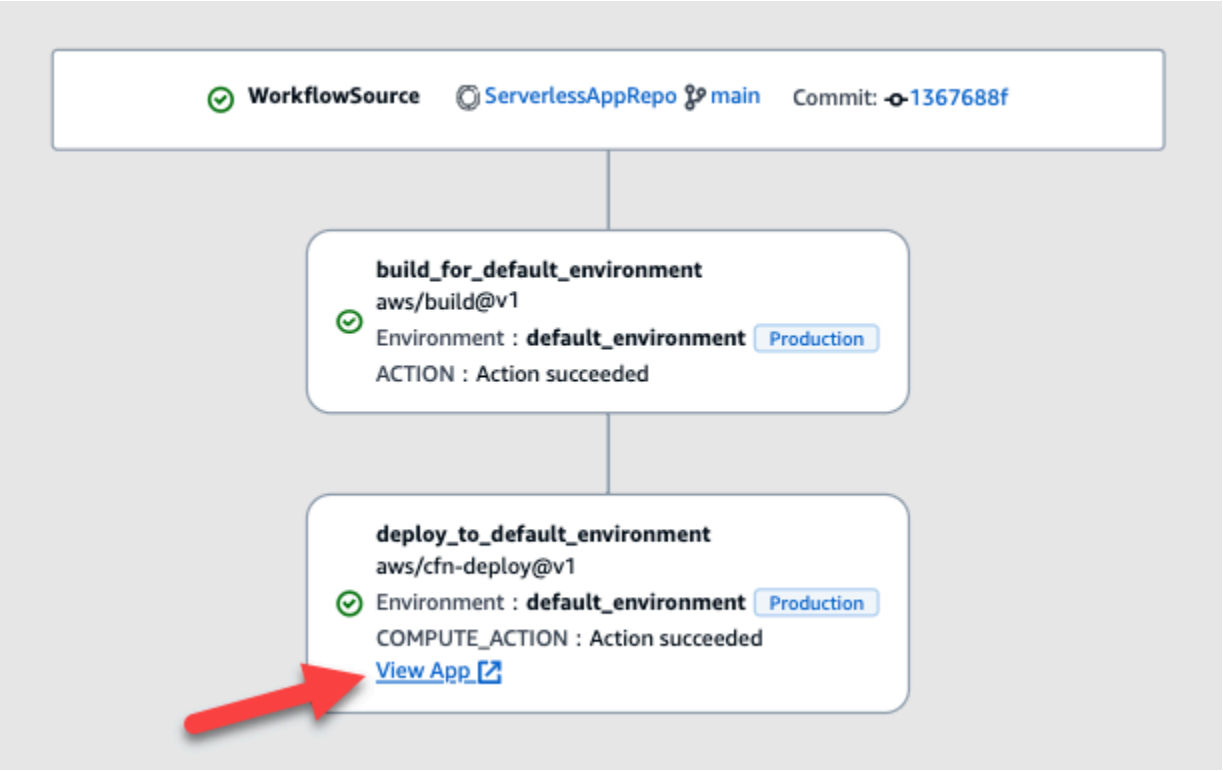
YAML

關閉「部署 AWS CloudFormation 堆疊」動作的復原

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇包含「部署 AWS CloudFormation 堆疊」動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在 YAML 程式碼中新增`disable-rollback: 1`屬性以停止復原。如需此性質的說明，請參閱〈〉「部署AWS CloudFormation堆疊」動作參考。

顯示已部署應用程式的 URL

如果您的工作流程部署了應用程式，您可以設定 Amazon CodeCatalyst 將應用程式的 URL 顯示為可點選的連結。此連結會顯示在部署該連結的動作內的 CodeCatalyst 主控台中。下列工作流程圖顯示動作底部出現的檢視應用程式 URL。



只要在 CodeCatalyst 主控台中按一下此 URL，您就可以快速驗證應用程式部署。

> **Note**
>
> 「部署到 Amazon ECS」動作不支援應用程式 URL。

若要啟用此功能，請使用包含appurl或的名稱將輸出變數新增至動作endpointurl。您可以使用帶有或不帶連接破折號 (-)、底線 (_) 或空格 ( ) 的名稱。字串不區分大小寫。將變數的值設定為已部署應用程式的http或 https URL。

> **Note**
>
> 如果您要更新現有的輸出變數以包含app url、或endpoint url字串，請更新此變數的所有參考，以使用新的變數名稱。

如需詳細步驟，請參閱下列其中一個程序：

- 在「AWS CDK 部署」動作中顯示應用程式 URL
- 若要在「部署 AWS CloudFormation 堆疊」動作中顯示應用程式 URL
- 在所有其他動作中顯示應用程式 URL

完成設定 URL 後，請依照下列指示確認其如預期般顯示：

- 驗證是否已新增應用程式 URL

在「AWS CDK 部署」動作中顯示應用程式 URL

1. 如果您正在使用部AWS CDK 署操作，請在應用 AWS CDK 程序代碼中添加一個Cfn0utput構造（這是一個鍵值對）：

   - 索引鍵名稱必須包含或 appurlendpointurl，不論是否有連接破折號 (-)、底線 (_) 或空格 ( )。字串不區分大小寫。
   - 此值必須是已部署應用程式的http或 https URL。

   例如，您的 AWS CDK 程式碼可能如下所示：

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'my-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'APP-URL', {
      value: https://mycompany.myapp.com,
      description: 'The URL of the deployed application',
      exportName: 'myApp',
    });
    ...
  }
}
```

如需有關CfnOutput建構的詳細資訊，請參閱 AWS Cloud Development Kit (AWS CDK) API 參考 CfnOutputProps中的介面。

2. 儲存並提交您的程式碼。
3. 繼續執行「驗證是否已新增應用程式 URL」。

若要在「部署 AWS CloudFormation 堆疊」動作中顯示應用程式 URL

1. 如果您使用的是「部署 AWS CloudFormation 堆疊」動作，請將輸出新增至 CloudFormation 範本或 AWS SAM 範本中具有下列特性的Outputs區段：

   - 鍵（也稱為邏輯 ID）必須包含或 appurlendpointurl，帶有或不帶連接破折號（-），底線（_）或空格（ ）。字串不區分大小寫。
   - 此值必須是已部署應用程式的http或 https URL。

   例如，您的 CloudFormation 範本可能看起來像這樣：

```
"Outputs" : {
  "APP-URL" : {
```

```
    "Description" : "The URL of the deployed app",
    "Value" : "https://mycompany.myapp.com",
    "Export" : {
      "Name" : "My App"
    }
  }
}
```

若要取得有關 CloudFormation 輸出的更多資訊，請參閱AWS CloudFormation 使用指南中的輸出。

2. 儲存並提交您的程式碼。
3. 繼續執行「驗證是否已新增應用程式 URL」。

在所有其他動作中顯示應用程式 URL

如果您正在使用其他動作來部署應用程式 (例如建置動GitHub 作或動作)，請執行下列動作以顯示應用程式 URL。

1. 在工作流程定義檔案中動作的Inputs或Steps區段中定義環境變數。變數必須具有下列特性：
   - name必須包含或 appurlendpointurl，帶或連接破折號 (-)、底線 (_) 或空格 ( )。字串不區分大小寫。
   - 此值必須是已部署應用程式的http或 https URL。

   例如，建置動作可能如下所示：

```
Build-action:
  Identifier: aws/build@v1
  Inputs:
    Variables:
      - Name: APP-URL
        Value: https://mycompany.myapp.com
```

   ... 或這個：

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
```

```
    Steps:
      - Run: APP-URL=https://mycompany.myapp.com
```

如需定義環境變數的詳細資訊，請參閱〈〉定義一個變量。

2. 匯出變數。

   例如，您的構建操作可能如下所示：

```
Build-action:
  ...
  Outputs:
    Variables:
      - APP-URL
```

   如需匯出變數的資訊，請參閱匯出變數，以便其他動作可以使用它。

3. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
4. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。
5. 繼續執行「驗證是否已新增應用程式 URL」。

驗證是否已新增應用程式 URL

- 啟動工作流程執行 (如果尚未自動啟動)。新的運行應該在其工作流程圖中將應用程序 URL 顯示為可點擊的鏈接。如需開始執行的詳細資訊，請參閱啟動工作流程執行。

移除部署目標

您可以從 CodeCatalyst 主控台的「環境」頁面移除部署目標，例如 Amazon ECS 叢集或 AWS CloudFormation 堆疊。

> **Important**
>
> 當您移除部署目標時，它會從主 CodeCatalyst 控台移除，但仍可在裝載該目標的 AWS 服務中使用 (如果它仍然存在)。

如果目標已過時，請考慮移除部署目標。 CodeCatalyst目標可能會變得過時，如果：

- 您已刪除部署至目標的工作流程。

- 您已變更要部署到的堆疊或叢集。
- 您已從 AWS 主控台中的 CloudFormation 或 Amazon ECS 服務刪除堆疊或叢集。

若要移除部署目標

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 選擇包含您要移除之建置目標的環境名稱。如需有關環境的資訊，請參閱使用環境。
5. 選擇「部署目標」索引標籤。
6. 選擇您要移除的部署目標旁邊的圓鈕。
7. 選擇移除。

   目標即會從頁面中移除。

使用工作流程

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要採取的一系列步驟或動作。工作流程也會定義導致工作流程啟動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

主題

- 建立、編輯和刪除工作流程
- 檢視工作流程狀態
- 使用動作
- 使用人工因素
- 使用運算和執行階段環境 Docker 影像

- 使用環境
- 使用檔案快取
- 使用套件
- 使用梯段
- 與秘密一起工作
- 使用來源
- 使用觸發程序
- 使用變數

建立、編輯和刪除工作流程

請使用下列程序來建立、編輯和刪除工作流程。

Visual

使用視覺化編輯器建立工作流程

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇建立工作流程。

   這時系統顯示創建工作流程對話框
5. 在「來源存放庫」欄位中，選擇工作流程定義檔案所在的來源儲存庫。檔案將儲存在所選儲存庫的`~/.codecatalyst/workflows/`資料夾中。如果沒有來源儲存庫，請建立一個。
6. 在「分支」(Branch) 欄位中，選擇工作流程定義檔案所在的分支。
7. 選擇建立。

   Amazon 會 CodeCatalyst 將存放庫和分支資訊儲存在記憶體中，但工作流程尚未提交。
8. 選擇 [視覺]。
9. 建立工作流程：

   a. (選擇性) 在工作流程圖表中，選擇「來源與觸發程式」方塊。觸發程式窗格隨即出現。選擇 [新增觸發器] 以新增觸發器。如需詳細資訊，請參閱新增推送、拉取或排程觸發器。

   b. 選擇 [+ 動作] (左上角)。「動作」目錄隨即出現。

c. 選擇動作內的加號 (+)，以將其新增至工作流程。使用右側的窗格設定動作。如需詳細資訊，請參閱新增動作。

d. （可選）選擇「工作流程屬性」（右上角）。工作流程屬性窗格隨即出現。設定工作流程名稱執行模式和計算。如需詳細資訊，請參閱 設定佇列、已取代和 parallel 執行 及 使用運算和執行階段環境 Docker 影像。

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。

11. 選擇「確認」，然後在「確認工作流程」對話方塊中執行下列操作：

a. 對於「工作流程」檔案名稱，保留預設名稱或輸入您自己的名稱。

b. 在「提交」訊息中，保留預設訊息或輸入您自己的訊息。

c. 對於「存放庫和分支」，請選擇工作流程定義檔案的來源儲存庫和分支。這些欄位應設定為您先前在 [建立工作流程] 對話方塊中指定的存放庫和分支。如果您願意，您可以立即更改存儲庫和分支。

> **Note**
>
> 提交您的工作流程定義檔案後，它無法與其他儲存庫或分支建立關聯，因此請務必謹慎選擇它們。

d. 選擇「確認」以確認工作流程定義檔案。

YAML

若要使用 YAML 編輯器建立工作流程

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇建立工作流程。

   這時系統顯示創建工作流程對話框

5. 在「來源存放庫」欄位中，選擇工作流程定義檔案所在的來源儲存庫。檔案將儲存在所選儲存庫的`~/.codecatalyst/workflows/`資料夾中。如果沒有來源儲存庫，請建立一個。
6. 在「分支」(Branch) 欄位中，選擇工作流程定義檔案所在的分支。
7. 選擇建立。

Amazon 會 CodeCatalyst 將存放庫和分支資訊儲存在記憶體中，但工作流程尚未提交。

8. 選擇 YAML。
9. 建立工作流程：

   a. (選擇性) 將觸發程式新增至 YAML 程式碼。如需詳細資訊，請參閱新增推送、拉取或排程觸發器。

   b. 選擇 [+ 動作] (左上角)。「動作」目錄隨即出現。

   c. 選擇動作內的加號 (+)，以將其新增至工作流程。使用右側的窗格設定動作。如需詳細資訊，請參閱新增動作。

   d. （可選）選擇「工作流程屬性」（右上角）。工作流程屬性窗格隨即出現。設定工作流程名稱、執行模式和計算。如需詳細資訊，請參閱 設定佇列、已取代和 parallel 執行 及 使用運算和執行階段環境 Docker 影像。

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，然後在「確認工作流程」對話方塊中執行下列操作：

    a. 對於「工作流程」檔案名稱，保留預設名稱或輸入您自己的名稱。

    b. 在「提交」訊息中，保留預設訊息或輸入您自己的訊息。

    c. 對於「存放庫和分支」，請選擇工作流程定義檔案的來源儲存庫和分支。這些欄位應設定為您先前在 [建立工作流程] 對話方塊中指定的存放庫和分支。如果您願意，您可以立即更改存儲庫和分支。

    > **Note**
    >
    > 提交您的工作流程定義檔案後，它無法與其他儲存庫或分支建立關聯，因此請務必謹慎選擇它們。

    d. 選擇「確認」以確認工作流程定義檔案。

若要編輯工作流程

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇您要編輯的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。

5. 選擇 編輯 。
6. 編輯工作流程。選擇 [視覺] 以使用視覺化編輯器，或選擇 YAML 使用 YAML 編輯器。您可以：
   a. 編輯觸發器。如需詳細資訊，請參閱使用觸發程序。
   b. 編輯動作。如需詳細資訊，請參閱使用動作。
7. 編輯其他工作流程屬性。選擇「工作流程屬性」（右上角）以變更工作流程名稱、執行模式或計算。如需詳細資訊，請參閱 設定佇列、已取代和 parallel 執行 及 使用運算和執行階段環境 Docker 影像。

若要刪除工作流程

刪除工作流程會刪除工作流程定義檔案和關聯的執行。從刪除的工作流程部署的任何AWS服務都必須手動移除，否則這些服務將繼續產生費用。

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇您要刪除的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇刪除。
6. 選擇確認。

檢視工作流程狀態

您可能想要檢視工作流程的狀態，以查看是否有任何工作流程設定問題需要解決，或疑難排解無法啟動的執行問題。 CodeCatalyst每次建立或更新工作流程的基礎工作流程定義檔案時，都會評估工作流程狀態。

> **Note**
>
> 您也可以檢視工作流程的執行狀態，這與工作流程狀態不同。如需詳細資訊，請參閱 檢視工作流程執行狀態與詳細。

工作流程可以包含下列其中一種狀態：

- 有效 — 工作流程可執行，且可由觸發程序啟動。

  若要將工作流程標示為有效，下列兩個條件都必須成立：

  - 工作流程定義檔案必須有效。
  - 工作流程必須沒有觸發器、沒有推送觸發程序，或是使用目前分支上的檔案執行的推播觸發程序。如需詳細資訊，請參閱 分支時的觸發考量。

- 「無效」— 工作流程的定義文件無效。工作流程無法手動執行，也無法透過觸發程序自動執行。無效的工作流程與工作流程定義一起顯示在 CodeCatalyst 控制台中有 ***n*** 個錯誤消息（或類似的）。

  若要將工作流程標記為無效，下列條件必須為真：

  - 工作流程定義檔案必須設定錯誤。

    若要修正設定錯誤的工作流程定義檔案，請參閱如何修復「工作流定義有 *n* 個錯誤」錯誤？。

- 「非活動」— 工作流程定義有效，但不能手動運行，也無法通過觸發器自動運行。

  若要將工作流程標示為非使用中，下列兩個條件都必須成立：

  - 工作流程定義檔案必須有效。
  - 工作流程定義檔案必須包含指定與工作流程定義檔案目前所在分支不同的推入觸發器。如需詳細資訊，請參閱 分支時的觸發考量。

    要將工作流程從「非活動」切換到「活動」，請參閱 如何修正「工作流程處於非作用中狀態」訊息？

> **Note**
>
> 如果工作流程指定了稍後移除的資源 (例如，套件存放庫)，將 CodeCatalyst 不會偵測到此變更，並會繼續將工作流程標示為有效。工作流程執行時，會發現這些類型的問題。

若要檢視工作流程的狀態

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 查找您要查看其狀態的工作流程。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。

這時系統顯示該狀態，并在列表中顯示工作流程。

5. (選擇性) 選擇工作流程的名稱，然後尋找「工作流程定義」欄位。它顯示工作流程的狀態。

使用動作

動作是工作流程的主要建置區塊，可定義工作流程執行期間要執行的邏輯工作單元。一般而言，工作流程包含多個依序執行或 parallel 執行的動作，具體取決於您設定它們的方式。

主題

動作類型

在 Amazon CodeCatalyst 工作流程中，您可以使用以下類型的動作。

動作類型

- CodeCatalyst 動作
- CodeCatalyst 實驗室動作
- GitHub 動作
- 第三方動作

CodeCatalyst 動作

CodeCatalyst 動作是由 CodeCatalyst 開發團隊撰寫、維護並完全支援的動作。

建置、測試和部署應用程式，以及執行其他工作，例如叫用 AWS Lambda 函數的 CodeCatalyst 動作。

可用的 CodeCatalyst 動作如下：

- 建置

  此動作會建置您的成品，並在 Docker 容器中執行單元測試。如需詳細資訊，請參閱 添加構建操作。

- 測試

  此動作會針對您的應用程式或成品執行整合和系統測試。如需詳細資訊，請參閱 新增測試動作。

- Amazon S3 發布

  此動作會將您的應用程式成品複製到 Amazon S3 儲存貯體。如需詳細資訊，請參閱 添加「Amazon S3 發布」操作。

- AWS CDK 引導

  此動作會佈建部署 CDK 應用程式所 AWS CDK 需的資源。如需詳細資訊，請參閱 添加「AWS CDK 引導」操作。

- AWS CDK 部署

  此動作會合成並部署應用 AWS Cloud Development Kit (AWS CDK) 程式。如需詳細資訊，請參閱 新增「AWS CDK 部署」動作。

- AWS Lambda 調用

  此操作調用一個 AWS Lambda 函數。如需詳細資訊，請參閱 添加「AWS Lambda調用」操作。

- 部署 AWS CloudFormation 堆疊

  此動作會部署 AWS CloudFormation 堆疊。如需詳細資訊，請參閱 新增「部署 AWS CloudFormation 堆疊」動作。

- 部署到 Amazon ECS

  此動作會註冊 Amazon ECS 任務定義，並將其部署到 Amazon ECS 服務。如需詳細資訊，請參閱 新增「部署到 Amazon ECS」動作。

- 部署至 Kubernetes 叢集

  此動作會將應用程式部署到 Kubernetes 叢集。如需詳細資訊，請參閱 新增「部署至 Kubernetes 叢集」動作。

- 渲染 Amazon ECS 任務定義

  此動作會將容器映像 URI 插入 Amazon ECS 任務定義 JSON 檔案，並建立新的任務定義檔案。如需詳細資訊，請參閱 新增「渲染 Amazon ECS 任務定義」動作。

本指南和每個動 CodeCatalyst 作的 Readme 中提供了動作的文件。

如需有關可用動 CodeCatalyst 作以及如何將動作新增至工作流程的資訊，請參閱新增動作。

CodeCatalyst 實驗室動作

實驗CodeCatalyst 室動作是 Amazon CodeCatalyst Labs 的一部分動作，這是實驗應用程式的試驗場所。 CodeCatalyst 已開發實驗室動作，以展示與 AWS 服務的整合。

下列是可用的「 CodeCatalyst 實驗室」動作：

- 部署到 AWS Amplify 主機

  此動作會部署應用程式以 Amplify 主機。
- 部署到 AWS App Runner

  此動作會將來源映像檔儲存庫中的最新映像檔部署至應用程式執行器。
- 部署到 Amazon CloudFront 和 Amazon S3

  此動作會將應用程式部署到 CloudFront 和 Amazon S3。
- 使用部署 AWS SAM

  此動作會使用 AWS Serverless Application Model (AWS SAM) 部署您的無伺服器應用程式。
- 使 Amazon CloudFront 緩存無效

  此動作會使一組指定路徑的 CloudFront 快取無效。
- 傳出網路勾點動作

  此動作可讓使用者使用 HTTPS 要求，將工作流程內的訊息傳送至任意 Web 伺服器。
- 發佈至 AWS CodeArtifact

  此動作會將封裝發行至存 CodeArtifact 放庫。
- 發佈到 Amazon SNS

  此動作可讓使用者透過建立主題、發佈至主題或訂閱主題來與 Amazon SNS 整合。
- 推送到 Amazon ECR

  此動作會建立 Docker 映像檔，並將其發佈至 Amazon Elastic Container Registry (Amazon ECR) 儲存庫。
- 使用 Amazon CodeGuru 安全掃描

此動作會建立已設定程式碼路徑的 zip 歸檔，並使用 [CodeGuru 安全性] 執行程式碼掃描。

- 地形社區版

  此操作運行地形社區版`plan`和`apply`操作。

每個動作的讀我檔案都提供 CodeCatalyst 實驗室動作的說明文件。

如需有關將 CodeCatalyst Labs 動作新增至工作流程及檢視其讀我檔案的資訊，請參閱新增動作。

GitHub 動作

GitHub Action 很像一個動作，不同之CodeCatalyst 處在於它是為與工作 GitHub 流程搭配使用而開發的。如需有關 GitHub 動作的詳細資訊，請參閱GitHub 動作文件。

您可以將「 GitHub 動作」與工作 CodeCatalyst 流程中的原生 CodeCatalyst 動作搭配使用。

為了您的方便， CodeCatalyst 控制台可以訪問多個流 GitHub行的操作。您也可以使用 GitHub Marketplace 中列出的任何 GitHub 動作 (受到一些限制)。

動 GitHub 作的文件可在每個動作的讀我檔案中找到。

如需詳細資訊，請參閱 新增 GitHub 動作。

第三方動作

第三方動作是由第三方廠商編寫且可在 CodeCatalyst 主控台中使用的動作。第三方動作的其中一個範例是由 Mender 撰寫的「修補 SCA」動作。

每個動作的讀我檔案都提供協力廠商動作的說明文件。協力廠商可能也會提供其他文件。

如需將協力廠商動作新增至工作流程及檢視其 Readme 的資訊，請參閱新增動作。

新增、設定和移除動作

選擇下列其中一個主題，以瞭解如何將動作新增至工作流程，以及如何將其移除。

主題

- 新增動作
- 添加「Amazon S3 發布」操作
- 添加「AWS CDK 引導」操作
- 添加「AWS Lambda調用」操作

- 新增「渲染 Amazon ECS 任務定義」動作
- 新增 GitHub 動作
- 移除動作

新增動作

使用下列指示將動作新增至工作流程，然後對其進行配置。

若要新增和設定動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇左上角的 [+ 動作]，[動作] 目錄就會顯示。
7. 在下拉式清單中，執行下列任一項作業：
   - 選擇 Amazon CodeCatalyst 來檢視CodeCatalyst、CodeCatalyst 實驗室或第三方動作。
     - CodeCatalyst 動作具有按 AWS標籤。
     - CodeCatalyst 實驗室動作具有「按 CodeCatalyst 實驗室」標籤。
     - 第三方動作具有依**##**標籤，其中##是第三方廠商的名稱。
   - 選擇GitHub此選項可檢視精選的 GitHub 動作清單。
8. 在動作目錄中，搜尋動作，然後執行下列其中一項作業：
   - 選擇加號 (+) 以將動作新增至您的工作流程。
   - 選擇動作的名稱以檢視其讀我檔案。
9. 設定處理行動。選擇 [視覺] 以使用視覺化編輯器，或選擇 YAML 使用 YAML 編輯器。如需詳細指示，請參閱下列連結。

   如需新增動CodeCatalyst作的指示，請參閱：

   - 添加構建操作
   - 新增測試動作

- 添加「Amazon S3 發布」操作
- 添加「AWS CDK 引導」操作
- 新增「AWS CDK 部署」動作
- 添加「AWS Lambda調用」操作
- 新增「部署 AWS CloudFormation 堆疊」動作
- 新增「部署到 Amazon ECS」動作
- 新增「部署至 Kubernetes 叢集」動作
- 新增「渲染 Amazon ECS 任務定義」動作

如需新增CodeCatalyst 實驗室動作的指示，請參閱：

- 動作的讀我檔案。您可以在動作目錄中選擇動作名稱來尋找讀我檔案。

如需新增動GitHub 作的指示，請參閱：

- 新增 GitHub 動作

如需新增第三方動作的指示，請參閱：

- 動作的讀我檔案。您可以在動作目錄中選擇動作名稱來尋找讀我檔案。

10. (選擇性) 選擇 [驗證] 以確定 YAML 代碼有效。
11. 選擇「確認」以確認變更。

添加「Amazon S3 發布」操作

本節說明如何將 Amazon S3 發佈動作新增至您的工作流程。Amazon S3 發佈動作會將檔案從來源目錄複製到 Amazon S3 儲存貯體。來源目錄可以位於：

- 來源儲存庫，或
- 由另一個工作流程動作產生的輸出人工

主題

- 何時使用此動作
- 範例工作流

- 添加「Amazon S3 發布」操作
- 由「Amazon S3 發佈」動作產生的變數
- 「Amazon S3 發布」動作定義

何時使用此動作

在下列情況下使用此動作

- 您有一個工作流程，可產生您想要存放在 Amazon S3 中的檔案。

  例如，您可能有一個工作流程，建立您想要在 Amazon S3 中託管的靜態網站。在這種情況下，您的工作流程將包括用於建立網站 HTML 和支援檔案的建置動作，以及用於將檔案複製到 Amazon S3 的 Amazon S3 發佈動作。
- 您有一個來源儲存庫，其中包含您要存放在 Amazon S3 中的檔案。

  例如，您可能有一個包含應用程式來源檔案的來源儲存庫，您希望每晚將這些檔案存檔到 Amazon S3。

範例工作流

下列範例工作流程包括 Amazon S3 發佈動作以及建置動作。工作流程會建立靜態文件網站，然後將其發佈到託管該文件的 Amazon S3。工作流程由下列依序執行的建置區塊組成：

> **Note**
>
> 下列工作流程範例僅供說明用途，僅適用於其他組態。

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 建置動作 (`BuildDocs`) — 在觸發器上，動作會建立靜態文件網站 (`mkdocs build`)，並將關聯的 HTML 檔案和支援中繼資料新增至名為的成品`MyDocsSite`。如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
- Amazon S3 發佈動作 (`PublishToS3`) — 建置動作完成後，此動作會將成`MyDocsSite`品中的網站複製到 Amazon S3 以進行託管。

```
Name: codecatalyst-s3-publish-workflow
```

```
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildDocs:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: echo BuildDocs started on `date`
        - Run: pip install --upgrade pip
        - Run: pip install mkdocs
        - Run: mkdocs build
        - Run: echo BuildDocs completed on `date`
    Outputs:
      Artifacts:
      - Name: MyDocsSite
        Files:
          - "site/**/*"

  PublishToS3:
    Identifier: aws/s3-publish@v1
    Environment:
      Name: codecatalyst-s3-publish-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-s3-publish-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - MyDocsSite
    Configuration:
      DestinationBucketName: my-bucket
      SourcePath: /artifacts/PublishToS3/MyDocSite/site
      TargetPath: my/docs/site
```

添加「Amazon S3 發布」操作

使用下列指示將 Amazon S3 發佈動作新增至您的工作流程。

Visual

若要使用視覺化編輯器新增「Amazon S3 發佈」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋 Amazon S3 發佈動作，然後執行下列其中一個動作：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或
   - 選擇 Amazon S3 發布。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入]、[組態] 和 [輸出] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「Amazon S3 發布」動作參考。此參考提供每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增「Amazon S3 發佈」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋 Amazon S3 發佈動作，然後執行下列其中一個動作：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇 Amazon S3 發布。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「Amazon S3 發布」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

由「Amazon S3 發佈」動作產生的變數

Amazon S3 發佈動作執行時，會產生可用於後續工作流程動作的變數。如需詳細資訊，請參閱「Amazon S3 發布」動作變量中的預先定義的變數清單。

「Amazon S3 發布」動作定義

Amazon S3 發佈動作定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「Amazon S3 發布」動作參考中的工作流定義參考。

添加「AWS CDK 引導」操作

本節說明如何將啟動程AWS CDK 序動作新增至您的工作流程。引AWS CDK 導操作使用現代模板在您的 AWS 環境中佈建一個引導堆棧。如果啟動程序堆疊已存在，則動作會在必要時對其進行更新。中存在啟動程序堆疊 AWS 是部署 AWS CDK 應用程式的先決條件。

如需有關啟動載入的詳細資訊，請參閱開發人員指南中的啟動載入。AWS Cloud Development Kit (AWS CDK)

主題

何時使用此動作

如果您有部署應用程式的工作流 AWS CDK 程，並且想要同時部署 (並在需要時更新) 啟動程序堆疊，請使用此動作。在這種情況下，您可以將啟動程AWS CDK 序操作添加到與部署應用程序的工作流程相同的工作流 AWS CDK 程中。

如果符合下列任一條件，請勿使用此動作：

- 您已經使用其他機制部署了啟動程序堆疊，並且想要保持其完整 (無更新)。
- 您想要使用自訂啟動程序範本，啟動程AWS CDK 序動作不支援此範本。

運作方式

AWS CDK 引導程序的工作原理如下：

1. 在執行階段，如果您指定的動作版本 1.0.7 或更早版本，動作會將最新的 CDK CLI (也稱為 AWS CDK Tookit) 下載至組建映像檔。 CodeCatalyst

   如果您指定版本 1.0.8 或更新版本，則動作會隨附特定版本的 CDK CLI，因此不會進行下載。
2. 此動作會使用 CDK CLI 來執行命`cdk bootstrap`令。此命令執行開發人員指南中「啟動載入」主題中所述的啟動載入工作。AWS Cloud Development Kit (AWS CDK)

「AWS CDK 引導程序」操作使用的 CDK CLI 版本

下表顯示不同版本的AWS CDK 啟動程序動作預設會使用哪個版本的 CDK CLI。

> **Note**
>
> 您可能可以覆寫預設值。如需詳細資訊，請參閱 「AWS CDK引導」動作參考 中的 CdkCliVersion。

「AWS CDK 引導」動作版	AWS CDK CLI 版本
1.0.0 —	最新
1.0.8 或更新版本	2.99.1

必要條件

在您可以使用AWS CDK 引導操作之前，請確保您已準備好 AWS CDK 應用程序。引導操作將在引導之前合成 AWS CDK 應用程序。您可以使用支援的任何程式設計語言撰寫應用程式 AWS CDK。

確保您的 AWS CDK 應用程序文件在以下位置可用：

- 來 CodeCatalyst 源儲存庫，或
- 由另一個工作流程動作產生的 CodeCatalyst 輸出人工

範例工作流程

如需包含AWS CDK 啟動程序動作的新增「AWS CDK 部署」動作工作流程，請參閱範例工作流程中的〈〉。

添加「AWS CDK 引導」操作

使用下列指示將啟動程AWS CDK 序動作新增至您的工作流程。

必要條件

在開始之前，請確定您已完成中所述的工作必要條件。

Visual

使用可視化編輯器添加「AWS CDK 引導」操作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋啟動程AWS CDK 序動作，然後執行下列其中一項作業：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇AWS CDK 引導。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入]、[組態] 和 [輸出] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「AWS CDK引導」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

> **Note**
>
> 如果啟動程AWS CDK 序動作失敗並出`npm install`現錯誤，請參我如何解決「npm 安裝」錯誤？閱以取得有關如何修正錯誤的資訊。

YAML

若要使用 YAML 編輯器新增「AWS CDK 啟動程序」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋啟動程AWS CDK 序動作，然後選擇 + 將其新增至工作流程圖表，並開啟其設定窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「AWS CDK引導」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

> **Note**
>
> 如果啟動程AWS CDK 序動作失敗並出`npm install`現錯誤，請參我如何解決「npm 安裝」錯誤？閱以取得有關如何修正錯誤的資訊。

由「AWS CDK 引導」操作生成的變量

啟動程AWS CDK 序動作執行時，它會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「AWS CDK引導」動作變量中的預先定義的變數清單。

「AWS CDK 引導」動作定義

啟動程AWS CDK 序動作會定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「AWS CDK引導」動作參考中的工作流定義參考。

添加「AWS Lambda調用」操作

本節說明如何將 AWS Lambdainvoke 動作新增至工作流程。AWS Lambda叫用動作會叫用您指定的 Lambda 函數。

除了叫用函數之外，AWS Lambdainvoke 動作也會將從 Lambda 函數收到的回應裝載中的每個頂層索引鍵轉換為工作流程輸出變數。然後可以在後續的工作流程動作中參考這些變數。如果您不希望將所有頂層金鑰轉換為變數，您可以使用篩選器來指定確切的金鑰。若要取得更多資訊，請參閱〈〉中的ResponseFilters性質描述「AWS Lambda 調用」操作引用。

主題

何時使用此動作

如果您想要將功能新增至 Lambda 函數封裝並由 Lambda 函數執行的工作流程，請使用此動作。

例如，您可能希望工作流程在開始構建應用程序之前向 Slack 頻道發送`Build started`通知。在此情況下，您的工作流程會包含AWS Lambda叫用 Lambda 以傳送 Slack 通知的叫用動作，以及建置應用程式的建置動作。

另一個範例是，您可能希望工作流程在部署應用程式之前對應用程式執行弱點掃描。在此情況下，您可以使用建置動作來建置應用程式、AWS Lambda叫用 Lambda 掃描弱點的動作，以及部署動作來部署掃描應用程式。

範例工作流程

> **Note**
>
> 下列範例工作流程僅供說明用途，需要額外的設定工作才能正常運作。它旨在為您提供一個示例，說明使AWS Lambda用 invoke 操作配置工作流程時可能會是什麼樣子。

下列工作流程包括 AWS Lambdainvoke 動作以及部署動作。工作流程會傳送 Slack 通知，指出部署已開始，然後使用範本將應AWS用程式部署到中。AWS CloudFormation工作流程由下列依序執行的建置區塊組成：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- AWS Lambda叫用動作 (LambdaNotify) — 在觸發器上，此動作會叫用指定AWS帳戶和區域 (my-aws-account和us-west-2) 中的 Notify-Start Lambda 函數。在叫用時，Lambda 函數會傳送 Slack 通知，指出部署已開始。
- 部署AWS CloudFormation堆疊動作 (Deploy) — AWS Lambda呼叫動作完成時，「部署AWS CloudFormation堆疊」動作會執行範本 (cfn-template.yml) 以部署應用程式堆疊。如需「部署AWS CloudFormation堆疊」動作的詳細資訊，請參閱新增「部署 AWS CloudFormation 堆疊」動作。

```
Name: codecatalyst-lamda-invoke-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  LambdaNotify:
    Identifier: aws/lambda-invoke@v1
    Environment:
      Name: my-production-environment
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-lambda-invoke-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Function: Notify-Start
      AWSRegion: us-west-2

  Deploy:
    Identifier: aws/cfn-deploy@v1
    Environment:
      Name: my-production-environment
```

```
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-deploy-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      name: my-application-stack
      region: us-west-2
      role-arn: arn:aws:iam::111122223333:role/StackRole
      template: ./cfn-template.yml
      capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

添加「AWS Lambda調用」操作

使用下列指示將 AWS Lambdainvoke 動作新增至您的工作流程。

先決條件

在開始之前，請確定您的AWS Lambda函數和相關的 Lambda 執行角色已準備就緒且可在中使用 AWS。如需詳細資訊，請參閱開AWS Lambda發人員指南中的 Lambda 執行角色主題。

Visual

若要使用視覺化編輯器新增「AWS Lambda叫用」動作

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋AWS Lambda叫用動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

- 選擇AWS Lambda呼叫。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
  - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
  - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 在 [輸入]、[組態] 和 [輸出] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「AWS Lambda 調用」操作引用。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增「AWS Lambda叫用」動作

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋AWS Lambda叫用動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇AWS Lambda呼叫。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「AWS Lambda 調用」操作引用。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。

12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

「AWS Lambda調用」操作生成的變量

當 AWS Lambdainvoke 動作執行時，它會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「AWS Lambda調用」動作變量中的預先定義的變數清單。

「AWS Lambda調用」操作定義

AWS Lambda呼叫動作會定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「AWS Lambda 調用」操作引用中的工作流定義參考。

新增「渲染 Amazon ECS 任務定義」動作

本節說明如何將渲染 Amazon ECS 任務定義動作新增至您的工作流程。此動作將 Amazon 彈性容器服務 (Amazon ECS) 任務定義檔案中的映像欄位更新為 Docker 映像名稱，該名稱由您的工作流程在執行階段提供。

> **Note**
>
> 您也可以使用此動作，以環境變數更新任務定義的`environment`欄位。

主題

- 何時使用此動作
- 運作方式
- 範例工作流
- 新增「渲染 Amazon ECS 任務定義」動作
- 檢視更新的任務定義檔案
- 透過「渲染 Amazon ECS 任務定義」動作產生的變數
- 「渲染 Amazon ECS 任務定義」動作定義

何時使用此動作

如果您的工作流程使用動態內容（例如提交 ID 或時間戳記）構建和標記 Docker 映像，請使用此選項。

如果您的任務定義檔案包含始終保持不變的影像值，請勿使用此動作。在此情況下，您可以在工作定義檔案中手動輸入影像的名稱。

運作方式

您必須在工作流程中將 「渲染 Amazon ECS」任務定義動作與「建置和部署到 Amazon ECS」動作搭配使用。這些動作共同運作如下：

1. 構建操作會構建您的 Docker 映像，並使用名稱，提交 ID，時間戳或其他動態內容對其進行標記。例如，您的構建操作可能如下所示：

```
MyECSWorkflow
  Actions:
    BuildAction:
      Identifier: aws/build@v1
      ...
      Configuration:
        Steps:
        # Build, tag, and push the Docker image...
          - Run: docker build -t MyDockerImage:${WorkflowSource.CommitId} .
          ...
```

   在前面的代碼中，該`docker build -t`指令指示構建 Docker 映像並在操作運行時使用提交 ID 對其進行標記。生成的圖像名稱可能如下所示：

   `MyDockerImage:a37bd7e`

2. 渲染 Amazon ECS 任務定義動作會將動態產生的映像名稱新增到您的任務定義檔案中，如下所示：`MyDockerImage:a37bd7e`

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            "image":  MyDockerImage:a37bd7e,
            "essential": true,
            ...
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
```

```
                    "containerPort": 80
                }
            ]
        }
    ],
...
}
```

或者，您也可以讓渲染 Amazon ECS 任務定義動作將環境變數新增至任務定義，如下所示：

```
{
  "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-
role",
  "containerDefinitions": [
    {
      "name": "codecatalyst-ecs-container",
      "image":  MyDockerImage:a37bd7e,
      ...
      "environment": [
        {
          name": "ECS_LOGLEVEL",
          value": "info"
        }
      ]
    }
  ],
...
}
```

如需有關環境變數的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的指定環境變數。

3. 「部署到 Amazon ECS」動作會向 Amazon ECS 註冊更新的任務定義檔案。註冊更新的任務定義檔案會將新映像部署`MyDockerImage:a37bd7e`到 Amazon ECS。

範例工作流

以下是完整工作流程的範例，其中包括 Render Amazon ECS 任務定義動作以及建置和部署動作。工作流程的目的是建置 Docker 映像檔並將其部署到您的 Amazon ECS 叢集中。工作流程由下列依序執行的建置區塊組成：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。

- 建置動作 (BuildDocker) — 在觸發器上，動作會使用 Docker 檔案建立 Docker 映像，並使用提交 ID 標記它，然後將映像推送至 Amazon ECR。如需建置動作的詳細資訊，請參閱使用工作流程建置 CodeCatalyst。
- Render Amazon ECS 任務定義動作 (RenderTaskDef) — 建置動作完成後，此動作會使用包含正確提交 ID 的image欄taskdef.json位值更新位於來源儲存庫根目錄中的現有項目。它會以新的檔案名稱 (task-definition-*random-string*.json) 儲存更新的檔案，然後建立包含此檔案的輸出加工品。render 動作也會產生名為的變數，task-definition並將其設定為新工作定義檔案的名稱。工件和變量將用於部署操作，這是下一個。
- 「部署到 Amazon ECS」動作 (DeployToECS) — 完成渲染 Amazon ECS 任務定義動作後，「部署到 Amazon ECS」動作會尋找渲染動作 (TaskDefArtifact) 產生的輸出成品，尋找其中的task-definition-*random-string*.json檔案，然後將其註冊到 Amazon ECS 服務。然後，Amazon ECS 服務會遵循task-definition-*random-string*.json檔案中的指示，在您的 Amazon ECS 叢集中執行 Amazon ECS 任務及關聯的碼頭映像容器。

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildDocker:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
```

```
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-east-2 | docker login --username
 AWS --password-stdin 111122223333.dkr.ecr.us-east-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG

  RenderTaskDef:
    DependsOn:
      - BuildDocker
    Identifier: aws/ecs-render-task-definition@v1
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      task-definition: taskdef.json
      container-definition-name: codecatalyst-ecs-container
      image: $REPOSITORY_URI:$IMAGE_TAG
    # The output artifact contains the updated task definition file.
    # The new file is prefixed with 'task-definition'.
    # The output variable is set to the name of the updated task definition file.
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - "task-definition*"
      Variables:
        - task-definition

  DeployToECS:
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
```

```
    Connections:
      - Name: codecatalyst-account-connection
        Role: codecatalyst-ecs-deploy-role
  #Input artifact contains the updated task definition file.
  Inputs:
    Sources: []
    Artifacts:
      - TaskDefArtifact
  Configuration:
    region: us-east-2
    cluster: codecatalyst-ecs-cluster
    service: codecatalyst-ecs-service
    task-definition: ${RenderTaskDef.task-definition}
```

新增「渲染 Amazon ECS 任務定義」動作

使用下列指示將渲染 Amazon ECS 任務定義動作新增至您的工作流程。

先決條件

在開始之前，請確定您有包含可動態產生 Docker 映像的建置動作的工作流程。如需詳細資訊，請參閱上述範例工作流

Visual

若要使用視覺化編輯器新增「呈現 Amazon ECS 任務定義」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋渲染 Amazon ECS 任務定義動作，然後執行下列其中一個動作：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

或

- 選擇渲染 Amazon ECS 任務定義。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
  - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
  - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「渲染 Amazon ECS 任務定義」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

使用 YAML 編輯器新增「呈現 Amazon ECS 任務定義」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉列表中選擇 Amazon CodeCatalyst。
9. 搜尋渲染 Amazon ECS 任務定義動作，然後執行下列其中一個動作：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

   或

   - 選擇渲染 Amazon ECS 任務定義。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「渲染 Amazon ECS 任務定義」動作參考。

11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

後續步驟

新增渲染動作後，請按照中新增「部署到 Amazon ECS」動作的說明將「部署到 Amazon ECS」動作新增至您的工作流程。新增部署動作時，請執行下列動作：

1. 在部署動作的「輸入」索引標籤中，於「人工因素-選用」中，選取彩現動作所產生的人工因素。它包含更新的任務定義文件。

   如需成品的詳細資訊，請參閱 使用人工因素。
2. 在部署動作的「組態」索引標籤的「工作定義」欄位中，指定下列動作變數：`${action-name.task-definition}`其中`###`稱是彩現動作的名稱，例如。`RenderTaskDef`渲染操作將此變量設置為任務定義文件的新名稱。

   如需變數的更多資訊，請參閱使用變數。

   如需如何設定部署動作的詳細資訊，請參閱上述範例工作流程。

檢視更新的任務定義檔案

您可以檢視已更新任務定義檔案的名稱和內容。

若要檢視更新任務定義檔案的名稱，請在轉譯 Amazon ECS 任務定義動作處理完畢後。

1. 查找包含已完成渲染操作的運行：
   a. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
   b. 選擇您的專案。
   c. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
   d. 選擇包含彩現動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
   e. 選擇包含已完成彩現動作的執行。
2. 在工作流程圖中，選擇彩現動作。
3. 選擇「輸出」。
4. 選擇「變數」。

5. 隨即顯示工作定義檔案名稱。它看起來類似於`task-definition--259-0a2r7gxlTF5X-.json`.

檢視已更新任務定義檔案內容的步驟

1. 查找包含已完成渲染操作的運行：
   a. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
   b. 選擇您的專案。
   c. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
   d. 選擇包含彩現動作的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
   e. 選擇包含已完成彩現動作的執行。
2. 在工作流程執行中，在頂端的 [Visual] 和 [YAML] 旁邊，選擇 [工作流程輸出]。
3. 在「人工因素」區段中，選擇包含已更新任務定義檔案的人工因素旁邊的「下載」。此成品會將「產生者」欄位設定為您的渲染動作的名稱。
4. 開啟 .zip 檔案以檢視工作定義 .json 檔案。

透過「渲染 Amazon ECS 任務定義」動作產生的變數

執行渲染 Amazon ECS 任務定義動作時，它會產生可用於後續工作流程動作的變數。如需詳細資訊，請參閱「渲染 Amazon ECS 任務定義」動作變量中的預先定義的變數清單。

「渲染 Amazon ECS 任務定義」動作定義

渲染 Amazon ECS 任務定義動作是在工作流程定義檔案中定義為一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「渲染 Amazon ECS 任務定義」動作參考中的工作流定義參考。

新增 GitHub 動作

GitHub Action 是很像一個CodeCatalyst 動作，不同之處在於它是為與工作 GitHub 流程一起使用而開發的。如需有關 GitHub 動作的詳細資訊，請參閱GitHub 動作文件。

您可以將「 GitHub 動作」與工作 CodeCatalyst 流程中的原生 CodeCatalyst 動作搭配使用。

有兩種方法可將「 GitHub 動作」新增至 CodeCatalyst 工作流程：

- 您可以從 CodeCatalyst 主控台的精選清單中選取「 GitHub 動作」。有幾種流 GitHub 行的操作可用。如需詳細資訊，請參閱 新增策劃 GitHub 動作。

- 如果您要使用的「 GitHub 動作」在 CodeCatalyst 主控台中無法使用，您可以使用「動作」(GitHub Actions) 動作來新增該動作。

  GitHub 「動作」動作是包裝「CodeCatalyst 動作」並使其與工作 CodeCatalyst 流程相容的 GitHub 動作。

  以下是包裝超級林特 GitHub動GitHub 作的「動作」動作範例：

```
Actions:
  GitHubAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Lint Code Base
          uses: github/super-linter@v4
          env:
            VALIDATE_ALL_CODEBASE: "true"
            DEFAULT_BRANCH: main
```

  在先前的程式碼中，動 CodeCatalyst GitHub 作動作 (由識別`aws/github-actions-runner@v1`) 包裝超級林特動作 (由識別`github/super-linter@v4`)，使其在工作流程中運作。 CodeCatalyst

  如需詳細資訊，請參閱 新增「動GitHub 作」動作。

所有 GitHub 動作 (包括已組織和否) 都必須包裝在 GitHub 「動作」 (`aws/github-actions-runner@v1`) 中，如前一個範例所示。要使動作正常運作，需要包裝函式。

主題

- GitHub 動作與動作有何 CodeCatalyst 不同？
- GitHub 動作可以與工作流程中的其他 CodeCatalyst 動作互動嗎？
- 我可以使用哪些 GitHub 動作？
- 中 GitHub 動作的限制 CodeCatalyst
- 如何新增 GitHub 動作 (高階步驟)？
- GitHub 動作是否在執行中 GitHub？
- 我也可以使用 GitHub 工作流程嗎？
- 教學課程：使用動作的 Lint 程式 GitHub碼
- 新增「動GitHub 作」動作

- 新增策劃 GitHub 動作
- 使用 GitHub 動作輸出參數

GitHub 動作與動作有何 CodeCatalyst 不同？

GitHub 在工作 CodeCatalyst 流程中使用的動作沒有與動 CodeCatalyst 作相同的存取和整合層級 AWS 和 CodeCatalyst 功能 (例如環境和問題)。

GitHub 動作可以與工作流程中的其他 CodeCatalyst 動作互動嗎？

是。例如， GitHub 動作可以使用其他 CodeCatalyst 動作產生的變數作為輸入，也可以與 CodeCatalyst 動作共用輸出參數和成品。如需詳細資訊，請參閱 使用 GitHub 動作輸出參數。

我可以使用哪些 GitHub 動作？

您可以使用任何透過 CodeCatalyst 主控台提供的 GitHub 動作，以及 GitHubMarketplace 中可用的任何 GitHub 動作。如果您決定使用 Marketplace 中的 GitHub 動作，請記住以下限制。

中 GitHub 動作的限制 CodeCatalyst

- GitHub 動作無法與 CodeCatalyst Lambda 運算類型搭配使用。
- GitHub 操作在 2022 年 11 月運行時環境 Docker 映像上運行，其中包括較舊的工具。如需有關影像和工具的更多資訊，請參閱使用執行階段環境 Docker 影像。
- GitHub 內部依賴上下`github`文或引 GitHub用特定資源的操作將無法在中使用 CodeCatalyst。例如，下列動作不適用於 CodeCatalyst：
  - 嘗試新增、變更或更新 GitHub 資源的動作。範例包括更新提取請求或在中建立問題的動作 GitHub。
  - 幾乎所有在 https://github.com/actions 中列出的操作。
- GitHub Docker 容器動作的動作可以運作，但必須由預設的 Docker 使用者 (根) 執行這些動作。請勿以使用者 1001 的身分執行動作。（在撰寫本文時，用戶 1001 在中工作 GitHub，但不在中工作 CodeCatalyst。）如需詳細資訊，請參閱 Docker 檔案動作 GitHub 支援中的使用者主題。

如需可透過 CodeCatalyst 主控台 GitHub 執行的動作清單，請參閱新增策劃 GitHub 動作。

如何新增 GitHub 動作 (高階步驟)？

將「動作」新增至工 GitHub 作 CodeCatalyst 流程的高階步驟如下：

1. 在 CodeCatalyst 專案中，您可以建立工作流程。您可以在工作流程中定義如何建置、測試和部署應用程式。如需詳細資訊，請參閱 開始使用中的工作流程 CodeCatalyst。
2. 在工作流程中，您可以新增已策劃的「 GitHub 動作」，或新增「動 GitHub 作」動作。
3. 您可以執行下列其中一項作業：
   - 如果您選擇新增已策劃的動作，請對其進行設定。如需詳細資訊，請參閱 新增策劃 GitHub 動作。
   - 如果您選擇新增未組織的動作，請在 GitHub「動作」動作中貼上 GitHub 動作的 YAML 程式碼。您可以在 GitHubMarketplace 中選擇的 GitHub動作的詳細信息頁面上找到此代碼。您可能需要稍微修改代碼以使其正常工作 CodeCatalyst。如需詳細資訊，請參閱 新增「動GitHub 作」動作。
4. (選擇性) 在工作流程中，您可以新增其他動作，例如建置和測試動作。如需詳細資訊，請參閱 使用中的工作流程來建置、測試和部署 CodeCatalyst。
5. 您可以手動或透過觸發器自動啟動工作流程。工作流程會執行「 GitHub 動作」和工作流程中的任何其他動作。如需詳細資訊，請參閱 啟動工作流程執行。

如需詳細步驟，請參閱：

- 新增策劃 GitHub 動作.
- 新增「動GitHub 作」動作.

GitHub 動作是否在執行中 GitHub？

沒有 動 GitHub 作執行中 CodeCatalyst，使用 CodeCatalyst的建置機器。

我也可以使用 GitHub 工作流程嗎？

否.

教學課程：使用動作的 Lint 程式 GitHub碼

在本教程中，您將超級林特 GitHub 動作添加到 Amazon CodeCatalyst 工作流程。Super-Linter 動作會檢查程式碼、尋找程式碼有錯誤、格式化問題和可疑結構的區域，然後將結果輸出至主控台)。CodeCatalyst將棉絨加入至工作流程之後，您可以執行工作流程以連接範例 Node.js 應用程式 () `app.js`。然後，您可以修正報告的問題，並再次執行工作流程，以查看修正程式是否有效。

> **Tip**
>
> 請考慮使用超級林特來 LINT YAML 檔案，例如範本。AWS CloudFormation

主題

先決條件

在開始之前，您需要：

- 具有已連接的 CodeCatalyst 空間AWS 帳戶。如需詳細資訊，請參閱建立支援 AWS 產生器 ID 使用者的空間。
- 您空間中的 CodeCatalyst 空白專案稱為`codecatalyst-linter-project`. 選擇從頭開始選項以創建此項目。

  如需詳細資訊，請參閱在 Amazon 中創建一個空項目 CodeCatalyst。

步驟 1：建立來源儲存庫

在此步驟中，您可以在中建立來源儲存庫 CodeCatalyst。在本教學課程中，`app.js`您將使用此儲存庫來儲存範例應用程式來源檔案。

如需來源儲存庫的詳細資訊，請參閱建立來源儲存庫。

若要建立來源儲存庫

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的項目，`codecatalyst-linter-project`。

3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇 [新增儲存庫]，然後選擇 [建立儲存庫]
5. 在存放庫名稱中，輸入：

```
codecatalyst-linter-source-repository
```

6. 選擇建立。

步驟 2：添加一個 app.js 文件

在此步驟中，您將app.js檔案新增至來源儲存庫。包app.js含函數代碼有一些絨毛會發現的錯誤。

若要新增 app.js 檔案的步驟

1. 在 CodeCatalyst 主控台中，選擇您的專案codecatalyst-linter-project。
2. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
3. 從來源儲存庫清單中，選擇您的儲存庫codecatalyst-linter-source-repository。
4. 在 [檔案] 中選擇 [建立檔案]。
5. 在文字方塊中，輸入下列程式碼：

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;
/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
  try {
```

```
    // const ret = await axios(url);
    response = {
      statusCode: 200,
      'body': JSON.stringify({
        message: 'hello world'
        // location: ret.data.trim()
      })
    }
  } catch (err) {
    console.log(err)
    return err
  }

    return response
}
```

6. 對於「檔案名稱」，輸入app.js。保留其他預設選項。
7. 選擇 Commit (遞交)。

   現在，您已經創建了一個名為app.js.

步驟 3：建立執行超林特動作的工作流程

在此步驟中，您會建立工作流程，當您將程式碼推送至來源儲存庫時，執行 SuperLint 動作。工作流程包含下列您在 YAML 檔案中定義的建置區塊：

- 觸發器 — 當您將變更推送至來源儲存庫時，此觸發器會自動啟動工作流程執行。關於觸發條件的詳細資訊，請參閱 使用觸發程序。
- 「動GitHub 作」動作 — 在觸發器上，「動作」GitHub 動作會執行超級林特動作，進而檢查來源儲存庫中的所有檔案。如果短絨發現問題，工作流程動作將失敗。

若要建立執行超級林特動作的工作流程

1. 在 CodeCatalyst 主控台中，選擇您的專案codecatalyst-linter-project。
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇建立工作流程。
4. 針對來源儲存庫，選擇codecatalyst-linter-source-repository。
5. 對於「分支」，請選擇main。
6. 選擇建立。

7. 刪除 YAML 範例程式碼。
8. 新增下列的 YAML：

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        github-action-code
```

在前面的程式碼中，請*github-action-code*依照本程序下列步驟中的指示，以 SuperLint 動作程式碼取代。

9. 前往 Marketplace 中的超級林特頁面。 GitHub
10. 在 [小寫] 底下`steps:`，尋找程式碼並將其貼到 [`Steps:`(大寫)] 下的 CodeCatalyst工作流程中。

調整 GitHub 動作程式碼以符合 CodeCatalyst 標準，如下列程式碼所示。

您的 CodeCatalyst 工作流程現在看起來像這樣：

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Lint Code Base
          uses: github/super-linter@v4
          env:
            VALIDATE_ALL_CODEBASE: "true"
            DEFAULT_BRANCH: main
```

11. (選擇性) 選擇 [驗證]，確定 YAML 程式碼在認可之前是有效的。
12. 選擇提交，輸入提交信息，選擇您的`codecatalyst-linter-source-repository`存儲庫，然後再次選擇提交。

    您現在已建立工作流程。工作流程執行會自動啟動，因為工作流程頂端定義了觸發器。

若要檢視進行中的工作流程執行

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇您剛建立的工作流程：`codecatalyst-linter-workflow`。
3. 在工作流程圖中，選擇SuperLinterAction。
4. 等待動作失敗。這是預期的失敗，因為 linter 在代碼中發現了問題。
5. 讓 CodeCatalyst 主機保持開啟狀態，然後移至步驟 4：修復超級棉絨發現的問題。

步驟 4：修復超級棉絨發現的問題

超級林特應該在代碼中發現問題，以及源`app.js`代碼庫中包含的`README.md`文件。

解決毛絨發現的問題

1. 在 CodeCatalyst 主控台中，選擇 [記錄檔] 索引標籤，然後選擇 [Lint 程式碼基底]。

   系統會顯示「超級林特」動作所產生的記錄檔。
2. 在超級林特日誌中，向下滾動到 90 行左右，您可以在其中找到問題的開始。它們看起來類似於以下內容：

```
/github/workspace/hello-world/app.js:3:13: Extra semicolon.
/github/workspace/hello-world/app.js:9:92: Trailing spaces not allowed.
/github/workspace/hello-world/app.js:21:7: Unnecessarily quoted property 'body'
 found.
/github/workspace/hello-world/app.js:31:1: Expected indentation of 2 spaces but
 found 4.
/github/workspace/hello-world/app.js:32:2: Newline required at end of file but not
 found.
```

3. 修復`app.js`並`README.md`在源存儲庫中並提交更改。

> **Tip**
>
> 要修復README.md，markdown請添加到代碼塊，如下所示：
>
> ```
> ```markdown
> Setup examples:
> ...
> ```
> ```

變更會啟動另一個工作流程自動執行。等待工作流程完成。如果您修正了所有問題，工作流程應該會成功。

清除

清理 CodeCatalyst 以從您的環境中移除本自學課程的痕跡。

若要在中進行清理 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 刪除codecatalyst-linter-source-repository。
3. 刪除codecatalyst-linter-workflow。

在本教程中，您學會了如何將超級林特 GitHub 操作添加到工作 CodeCatalyst 流中，以便 lint 一些代碼。

新增「動GitHub 作」動作

GitHub 「動作」動作是包裝「CodeCatalyst 動作」並使其與工作 CodeCatalyst 流程相容的 GitHub 動作。

如需詳細資訊，請參閱新增 GitHub 動作。

欲將「動作 GitHub 」動作新增至工作流程，請遵循下列步驟。

> **Tip**
>
> 如需向您展示如何使用 GitHub 「動作」動作的自學課程，請參閱教學課程：使用動作的 Lint 程式 GitHub碼。

Visual

使用視覺化編輯器新增「GitHub 動作」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉式清單中選擇GitHub。
9. 搜尋「動作 GitHub 」動作，然後執行下列其中一項作業：
   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或
   - 選擇「GitHub 動作」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 在 [輸入] 和 [組態] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「動 GitHub 作」動作參考。此參考提供有關每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，因為它出現在 YAML 和視覺化編輯器中。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增「GitHub 動作」動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉式清單中選擇GitHub。
9. 搜尋「動作 GitHub 」動作，然後執行下列其中一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇「GitHub 動作」。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。
10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了每個可用屬性的說明「動GitHub 作」動作參考。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

「動GitHub 作」動作定義

GitHub 「動作」動作定義為工作流程定義檔案中的一組 YAML 屬性。若要取得有關這些性質的資訊，請參閱「動GitHub 作」動作參考中的工作流定義參考。

新增策劃 GitHub 動作

策劃的「 GitHub 動作」是可在 CodeCatalyst 主控台中使用的「 GitHub 動作」，可做為如何在工作CodeCatalyst流程中使用「 GitHub 動作」的範例。

「已編寫的動作」動 GitHub 作會包裝在由識別 CodeCatalyst元識別的「已編寫的GitHub 動作」動作中aws/github-actions-runner@v1。例如，以下是策劃的 GitHub 動作 TruffleHog OSS 的樣子：

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ' ' # Required; description: Repository path
            base: ' ' # Required; description: Start scanning from here (usually main
 branch).
            head: ' ' # Optional; description: Scan commits until here (usually dev
 branch).
            extra_args: ' ' # Optional; description: Extra args to be passed to the
 trufflehog cli.
```

在先前的程式碼中， CodeCatalyst GitHub 動作動作 (由識別aws/github-actions-runner@v1) 包裝 TruffleHog OSS 動作 (由識別trufflesecurity/trufflehog@v3.16.0)，使其在工作 CodeCatalyst 流程中運作。

若要設定此動作，您可以with:使用您自己的值取代下的空字串。例如：

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ./
```

```
          base: main # Required; description: Start scanning from here (usually main
branch).
          head: HEAD # Optional; description: Scan commits until here (usually dev
branch).
          extra_args: '--debug --only-verified' # Optional; description: Extra args
to be passed to the trufflehog cli.
```

欲將組織的 GitHub 動作新增至工作流程，請遵循下列步驟。有關在工作 CodeCatalyst 流程中使用 GitHub 「動作」的一般資訊，請參閱新增 GitHub 動作。

> **Note**
>
> 如果您在已策劃的 GitHub 動作清單中沒有看到您的「動作」，您仍然可以使用「動作」動作將其新增至工GitHub 作流程。如需詳細資訊，請參閱新增「動GitHub 作」動作。

Visual

若要使用視覺化編輯器新增策劃的 GitHub 動作

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉式清單中選擇GitHub。
9. 瀏覽或搜尋 GitHub 動作，然後執行下列任一項作業：

   - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

     或

   - 選擇「 GitHub 動作」的名稱。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
     - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
     - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 在 [輸入]、[組態] 和 [輸出] 索引標籤中，根據您的需求完成欄位。如需每個欄位的描述，請參閱「動GitHub 作」動作參考。此參考提供有關「動作」動GitHub作可用之每個欄位 (以及對應的 YAML 屬性值) 的詳細資訊，就像它同時出現在 YAML 和視覺化編輯器中時。

    如需策劃 GitHub動作可用之組態選項的相關資訊，請參閱其說明文件。

11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增策劃的 GitHub 動作

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 選擇左上角的 [+ 動作] 以開啟動作目錄。
8. 從下拉式清單中選擇GitHub。
9. 瀏覽或搜尋 GitHub 動作，然後執行下列任一項作業：

    - 選擇加號 (+) 以將動作新增至工作流程圖，並開啟其設定窗格。

      或

    - 選擇「 GitHub 動作」的名稱。動作詳細資訊對話方塊隨即出現。在此對話方塊中：
        - (選擇性) 選擇「檢視原始碼」以檢視動作的原始程式碼。
        - 選擇 [新增至工作流程] 以將動作新增至工作流程圖表，並開啟其組態窗格。

10. 根據您的需要修改 YAML 程式碼中的屬性。中提供了 GitHub 「動作」動作可用之每個屬性的說明「動GitHub 作」動作參考。

    如需策劃 GitHub動作可用之組態選項的相關資訊，請參閱其說明文件。

11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

使用 GitHub 動作輸出參數

您可以將GitHub 輸出參數整合到 CodeCatalyst 工作流程中。

> **Note**
>
> 輸出參數的另一個詞是可變的。因為在文件中 GitHub 使用「輸出參數」一詞，所以我們也會使用這個術語。

主題

- 匯 GitHub 出輸出參數，以便其他動作可以使用該參數
- 引用輸 GitHub出參數

匯 GitHub 出輸出參數，以便其他動作可以使用該參數

請使用下列指示匯 GitHub 出輸出參數，以便在其他 CodeCatalyst 工作流程動作中使用該參數。

匯出 GitHub 輸出參數的步驟

1. 開啟工作流程並選擇 「編輯」。如需相關指示，請參閱若要編輯工作流程。
2. 在產生您要匯出之輸出參數的 GitHub 「動作」動作中，新增具有下列基礎`Variables`屬性的`Outputs`區段：

```
Actions:
  MyGitHubAction:
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'step-id_output-name'
```

取代：

- *`## ID`* 與 GitHub 動作`steps`部分中的`id:`屬性值。
- *`######`*出參數的名稱。 GitHub

範例

下列範例說明如何匯出名為的 GitHub 輸出參數`SELECTEDCOLOR`。

```
Actions:
  MyGitHubAction:
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'random-color-generator_SELECTEDCOLOR'
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
```

引用輸 GitHub出參數

請使用下列指示來參照 GitHub 輸出參數。

參照 GitHub 輸出參數的步驟

1. 完成「匯 GitHub 出輸出參數，以便其他動作可以使用該參數」中的步驟。

   GitHub 輸出參數現在可用於其他動作。
2. 請注意輸出參數的`Variables`值。它包括一個下劃線 (_) 。
3. 請使用下列語法參照輸出參數：

   ```
   ${action-name.output-name}
   ```

   取代：

   - *####*以產生輸出參數的 CodeCatalystGitHub 動作名稱 (請勿使用 GitHub 動作的`name`或`id`)。
   - *########*提到的輸出參數的`Variables`值。

   範例

   ```
   BuildActionB:
     Identifier: aws/build@v1
     Configuration:
       Steps:
   ```

```
      - Run: echo ${MyGitHubAction.random-color-generator_SELECTEDCOLOR}
```

帶上下文的示例

下列範例說明如何在中設定SELECTEDCOLOR變數GitHubActionA、輸出變數，然後在中參照它BuildActionB。

```
Actions:
  GitHubActionA:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
    Outputs:
      Variables:
      - 'random-color-generator_SELECTEDCOLOR'

   BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: echo ${GitHubActionA.random-color-generator_SELECTEDCOLOR}
```

移除動作

使用下列指示從工作流程中移除動作。

Visual

使用視覺化編輯器移除動作的步驟

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。

6. 選擇 [視覺]。
7. 在工作流程圖表中您要移除的動作中，選擇垂直省略符號圖示，然後選擇 [移除]。
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器移除動作

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 尋找包含您要移除之動作的 YAML 區段。

   選擇部分，然後按鍵盤上的刪除鍵。
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

開發自訂動作

您可以使用「動作開發工具包」(ADK) 開發要在工 CodeCatalyst 作流程中使用的自訂動作。然後，您可以將動作發佈至動 CodeCatalyst 作目錄，以便其他使用 CodeCatalyst 者可以在其工作流程中檢視並使用該動作。

若要開發、測試和發行動作 (高階工作)

1. 安裝開發動作所需的必要工具和套件。
2. 創建一個 CodeCatalyst 存儲庫來存儲您的操作代碼。
3. 初始化動作。這會列出動作所需的來源檔案，包括您可以使用自己的程式碼更新的動作定義檔 (`action.yml`)。

4. 啟動動作程式碼，以取得必要的工具和程式庫，以建置、測試和發行動作專案。
5. 在本機電腦上建置動作，然後將變更推送至存 CodeCatalyst放庫。
6. 使用本機單元測試來測試動作，然後在中執行 ADK 產生的工作流程。 CodeCatalyst
7. 選擇主控台中的「發佈」按鈕，將動作發佈至動 CodeCatalyst 作目 CodeCatalyst 錄。

如需詳細步驟，請參閱 Amazon CodeCatalyst 動作開發套件開發人員指南。

將動作分組到動作群組中

動作群組包含一或多個動作。將動作分組到動作群組中，可協助您保持工作流程井然有序，也可讓您配置不同動作群組之間的相依性。

> **Note**
>
> 您無法將動作群組嵌套在其他動作或動作群組中。

使用下列指示來定義動作群組。

Visual

不可用。選擇 YAML 以檢視 YAML 指示。

YAML

定義動作群組

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在中`Actions`，新增類似下列內容的程式碼：

```
Actions:
```

```
  action-group-name:
    Actions:
      action-1:
        Identifier: aws/build@v1
        Configuration:
          ...
      action-2:
        Identifier: aws/ecs-deploy@v1
        Configuration:
          ...
```

如需其他範例，請參閱範例：定義兩個動作群組。如需詳細資訊，請參閱中動作的action-group-name屬性描述工作流定義參考。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

範例：定義兩個動作群組

下列範例顯示如何定義兩個動作群組：BuildAndTest和Deploy。BuildAndTest動作群組包括兩個動作 (Build和Test)，而Deploy動作群組也包括兩個動作 (DeployCloudFormationStack和DeployToECS)。

```
Actions:
  BuildAndTest: # Action group 1
    Actions:
      Build:
        Identifier: aws/build@v1
        Configuration:
          ...
      Test:
        Identifier: aws/managed-test@v1
        Configuration:
  Deploy: #Action group 2
    Actions:
      DeployCloudFormationStack:
        Identifier: aws/cfn-deploy@v1
        Configuration:
          ...
      DeployToECS:
        Identifier: aws/ecs-deploy@v1
        Configuration:
```

```
...
```

將動作配置為依賴其他動作

依預設，當您將動作新增至工作流程時，會在視覺化編輯器中並排新增動作。這表示當您啟動工作流程執行時，動作將 parallel 執行。如果您希望動作依序執行 (並在視覺化編輯器中垂直顯示)，您必須設定它們之間的相依性。例如，您可以將動作設定為依賴`Test`動`Build`作，以便在建置動作之後執行測試動作。

您可以設定動作之間、動作群組之間以及動作與動作群組之間的相依性。您還可以配置 one-to-many 依賴關係，以便一個動作依賴於其他多個動作才能啟動。請參閱設定相依性的準則以確保您的相依性設定符合工作流程的 YAML 語法。

主題

- 設定相依性
- 設定相依性的準則
- 範例

設定相依性

請使用下列指示來設定工作流程中動作之間的相依性。

Visual

若要使用視覺化編輯器設定相依性

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇相依於另一個動作的動作。
8. 選擇「輸入」頁標。
9. 在取決於-選用中，執行下列動作：

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器設定相依性

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在將依賴於另一個動作中，添加類似以下內容的代碼：

```
action-name:
  DependsOn:
    - action-1
```

如需更多範例，請參閱 範例。如需一般指導方針，請參閱設定相依性的準則。如需詳細資訊，請參閱「」中針對您的動作工作流定義參考的DependsOn屬性說明。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

設定相依性的準則

設定相依性時，請遵循下列準則：

- 如果動作位於動作群組內，則該動作只能依賴相同動作群組中的其他動作。
- 動作和動作群組可以依賴於 YAML 階層中相同層級的其他動作和動作群組，但不依賴於不同層級。

範例

下列範例顯示如何在工作流程定義檔案中配置動作與動作群組之間的相依性。

主題

- 範例：設定簡單的相依性
- 範例：將動作群組配置為依賴動作
- 範例：將動作群組配置為依賴另一個動作群組
- 範例：將動作群組配置為依賴多個動作

範例：設定簡單的相依性

下列範例顯示如何將Test動作設定為依賴使用該DependsOn內容的Build動作。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      ...
  Test:
    DependsOn:
      - Build
    Identifier: aws/managed-test@v1
     Configuration:
       ...
```

範例：將動作群組配置為依賴動作

下列範例顯示如何將DeployGroup動作群組設定為依賴FirstAction動作。請注意，動作和動作群組位於同一層級。

```
Actions:
  FirstAction: #An action outside an action group
    Identifier: aws/github-actions-runner@v1
    Configuration:
      ...
  DeployGroup: #An action group containing two actions
    DependsOn:
      - FirstAction
    Actions:
```

```
      DeployAction1:
      ...
      DeployAction2:
      ...
```

範例：將動作群組配置為依賴另一個動作群組

下列範例顯示如何將DeployGroup動作群組設定為依賴BuildAndTestGroup動作群組。請注意，動作群組位於同一層級。

```
Actions:
  BuildAndTestGroup: # Action group 1
    Actions:
      BuildAction:
      ...
      TestAction:
      ...
  DeployGroup: #Action group 2
    DependsOn:
      - BuildAndTestGroup
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

範例：將動作群組配置為依賴多個動作

下列範例顯示如何將DeployGroup動作群組配置為依賴FirstAction動SecondAction作、動作以及BuildAndTestGroup動作群組。請注意DeployGroup，與FirstAction、SecondAction和位於相同層級BuildAndTestGroup。

```
Actions:
  FirstAction: #An action outside an action group
    ...
  SecondAction: #Another action
    ...
  BuildAndTestGroup: #Action group 1
    Actions:
      Build:
      ...
      Test:
```

```
    ...
DeployGroup: #Action group 2
  DependsOn:
    - FirstAction
    - SecondAction
    - BuildAndTestGroup
  Actions:
    DeployAction1:
    ...
    DeployAction2:
    ...
```

檢視動作的原始程式碼

您可以檢視動作的原始程式碼，以確定其中不包含有風險的程式碼、安全性弱點或其他瑕疵。

使用下列指示來檢視CodeCatalyst、CodeCatalyst 實驗室或第三方動作的原始程式碼。

> **Note**
>
> 若要檢視「GitHub動作」的原始程式碼，請前往「GitHub Marketplace」中的動作頁面。此頁面包含動作存放庫的連結，您可以在其中找到動作的原始程式碼。

> **Note**
>
> 您無法檢視下列 CodeCatalyst 動作的原始程式碼：建置、測試、GitHub 動作。

> **Note**
>
> AWS不支持或保證 GitHub 操作或第三方操作的操作代碼。

若要檢視動作的原始程式碼

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 尋找您要檢視其程式碼的動作：

a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
b. 選擇任何工作流程的名稱，或建立一個工作流程。如需有關建立工作流程的資訊，請參閱建立、編輯和刪除工作流程。
c. 選擇 編輯 。
d. 選擇左上角的 [+ 動作] 以開啟動作目錄。
e. 在下拉式清單中，選擇 Amazon CodeCatalyst 以檢視 CodeCatalyst、 CodeCatalyst 實驗室和第三方動作。
f. 搜尋動作，然後選擇其名稱。請勿選擇加號 (+)。

   此時會顯示動作的詳細資訊。

4. 在「動作詳細資料」對話方塊中，選擇底部附近的「下載」。

   會出現一個頁面，顯示動作原始程式碼所在的 Amazon S3 儲存貯體。如需有關 Amazon S3 的資訊，請參閱什麼是 Amazon S3？ 在 Amazon 簡單存儲服務用戶指南。

5. 檢查程式碼，確保程式碼符合您對品質和安全性的期望。

使用動作版本

依預設，當您將動作新增至工作流程時，Amazon 會使用以下格式將完整版本新 CodeCatalyst 增至工作流程定義檔案：

v*major*.*minor*.*patch*

例如：

```
My-Build-Action:
  Identifier: aws/build@v1.0.0
```

您可以縮短Identifier屬性中的完整版本，以便工作流程始終使用動作的最新次要或修補程式版本。

例如，如果您指定：

```
My-CloudFormation-Action:
  Identifier: aws/cfn-deploy@v1.0
```

... 最新的補丁版本是1.0.4，然後該操作將使用1.0.4。如果發布了更高版本，比如說1.0.5，那麼操作將使用1.0.5。如果發布了次要版本，比如說1.1.0，那麼該動作將繼續使用1.0.5。

如需指定版本的詳細指示，請參閱下列其中一個主題。

主題

- 指定要使用的動作版本
- 確定哪些版本的動作可用

指定要使用的動作版本

請使用下列指示來指出您要工作流程使用的動作版本。您可以指定最新的主要或次要版本，或指定特定的修補程式版本。

我們建議您使用動作的最新次要或修補程式版本。

Visual

不可用。選擇 YAML 以檢視 YAML 指示。

YAML

將工作流程設定為使用動作的最新版本或特定修補程式版本

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 尋找您要編輯其版本的動作。
8. 尋找動作的`Identifier`屬性，並將版本設定為下列其中一項：
   - 動作識別碼 @v *major* — 使用此語法讓工作流程使用特定的主要版本，並允許自動選擇最新的次要版本和修補程式版本。
   - *##### @v ### minor* — 使用此語法讓工作流程使用特定的次要版本，並允許自動選擇最新的修補程式版本。
   - *##### @v ### ####。 patch* — 使用此語法讓工作流程使用特定的修補程式版本。

> **Note**
>
> 如果您不確定哪些版本可用，請參閱確定哪些版本的動作可用。

> **Note**
>
> 您不能省略主要版本。

9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

確定哪些版本的動作可用

請使用下列指示來決定您可以在工作流程中使用哪些動作版本。

Visual

若要判斷哪些動作版本可用

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 尋找您要檢視其版本的動作：
   a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
   b. 選擇任何工作流程的名稱，或建立一個工作流程。如需有關建立工作流程的資訊，請參閱建立、編輯和刪除工作流程。
   c. 選擇 編輯 。
   d. 選擇左上角的 [+ 動作] 以開啟動作目錄。
   e. 在下拉式清單中，選擇 Amazon CodeCatalyst 以檢視 CodeCatalyst、 CodeCatalyst實驗室和第三方動作，或選擇GitHub檢視策劃的 GitHub動作。
   f. 搜尋動作，然後選擇其名稱。請勿選擇加號 (+)。

      此時會顯示動作的詳細資訊。

4. 在「動作詳細資料」對話方塊的右上角附近，選擇「版本」下拉式清單，以查看動作的可用版本清單。

YAML

不可用。選擇「視覺」以查看可視化編輯器說明。

使用人工因素

人工因素是工作流程動作的輸出，通常由資料夾或檔案的封存組成。人工因素很重要，因為它們可讓您在動作之間共用檔案和資訊。

例如，您可能有一個生成sam-template.yml文件的構建操作，但您希望部署操作使用它。在這個案例中，您會使用成品來允許建置動作與部署動作共用sam-template.yml檔案。代碼可能看起來像這樣：

```
Actions:
  BuildAction:
    Identifier: aws/build@v1
    Steps:
      - Run: sam package --output-template-file sam-template.yml
    Outputs:
      Artifacts:
        - Name: MYARTIFACT
          Files:
            - sam-template.yml
  DeployAction:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Artifacts:
        - MYARTIFACT
    Configuration:
      template: sam-template.yml
```

在先前的程式碼中，建置動作 (BuildAction) 會產生sam-template.yml檔案，然後將其新增至名為的輸出成品MYARTIFACT。後續的部署動作 (DeployAction) 會指定MYARTIFACT為輸入，讓它能夠存取sam-template.yml檔案。

主題

- 我可以共享工件而不將它們指定為輸出和輸入嗎？

我可以共享工件而不將它們指定為輸出和輸入嗎？

是，您可以在動作之間共用成品，而不必在動作的 YAML 程式碼`Outputs`和`Inputs`區段中指定它們。若要這麼做，您必須開啟運算共用。如需有關計算共用以及如何在開啟時指定人工因素的詳細資訊，請參閱跨動作共用運算。

> **Note**
>
> 雖然運算共用功能可讓您省去`Outputs`和`Inputs`區段的需求來簡化工作流程的 YAML 程式碼，但是在開啟此功能之前，您應該注意這項功能的限制。如需有關這些限制的資訊，請參閱計算共用的考量。

可以在工作流程之間共用成品嗎？

否，您無法在不同工作流程之間共用人工因素；不過，您可以在相同工作流程中的動作之間共用人工因素。

定義輸出人工因素

使用下列指示來定義您要輸出動作的人工因素。然後，此成品可供其他動作使用。

> **Note**
>
> 並非所有動作都支援輸出加工品。若要判斷您的動作是否支援這些動作，請執行後續的視覺化編輯器指示，並查看動作是否包含「輸出」索引標籤上的「輸出成品」按鈕。如果是，則支援輸出加工品。

Visual

若要使用視覺化編輯器定義輸出成品

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇將產生加工品的動作。
8. 選擇 Output (輸出) 索引標籤。
9. 在人工因素下，選擇新增人工因素
10. 選擇新增人工因素，然後在欄位中輸入資訊，如下所示。

    建置成品名稱

    指定動作所產生的人工因素名稱。Artifact 名稱在工作流程中必須是唯一的，且僅限於英數字元 (a-z、A-Z、0-9) 和底線 (_)。不允許使用空格、連字號 (-) 和其他特殊字元。您無法使用引號來啟用輸出人工因素名稱中的空格、連字號和其他特殊字元。

    如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

    由構建生成的文件

    指定動作輸出的加工品中 CodeCatalyst 包含的檔案。這些檔案會在工作流程動作執行時由工作流程動作產生，也可在來源存放庫中使用。檔案路徑可以位於來源儲存庫或上一個動作的人工因素中，且相對於來源儲存庫或人工因素根目錄。您可以使用全域模式來指定路徑。範例：

    - 若要指定位於組建位置或來源存放庫位置根目錄中的單一檔案，請使用`my-file.jar`。
    - 若要在子目錄中指定單一檔案，請使用`directory/my-file.jar`或`directory/subdirectory/my-file.jar`。
    - 若要指定所有檔案，請使用`"**/*"`。**glob 模式指示匹配任意數量的子目錄。
    - 若要指定名為的目錄中的所有檔案和目錄`directory`，請使用`"directory/**/*"`。**glob 模式指示匹配任意數量的子目錄。

- 若要指定名為的目錄中的所有檔案`directory`，但不指定其任何子目錄中的檔案，請使用`"directory/*"`。

> **Note**
>
> 如果檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

> **Note**
>
> 您可能需要在檔案路徑中新增前置詞，以指出要在其中尋找的成品或來源。如需詳細資訊，請參閱 參考來源儲存庫中的檔案 及 參考人工因素中的檔案。

11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器定義輸出成品

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在工作流程動作中，新增類似下列內容的程式碼：

```
action-name:
  Outputs:
    Artifacts:
      - Name: artifact-name
```

```
        Files:
          - file-path-1
          - file-path-2
```

如需更多範例，請參閱範例。如需詳細資訊，請參工作流定義參考閱您的動作。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

定義輸入人工因素

如果您要使用由其他動作產生的人工因素，則必須將其指定為目前動作的輸入。您可以指定多個人工因素作為輸入，這取決於動作。如需詳細資訊，請參工作流定義參考閱您的動作。

> **Note**
>
> 您無法從其他工作流程參考人工因素。

使用下列指示，從另一個動作指定人工因素作為目前動作的輸入。

先決條件

在開始之前，請確定已從其他動作輸出成品。如需詳細資訊，請參閱 定義輸出人工因素。輸出成品可讓其他動作使用。

Visual

將人工因素指定為動作輸入的步驟 (視覺化編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要將成品指定為輸入的動作。

8. 選擇「輸入」。
9. 在人工因素-選擇性中，執行下列動作：

   指定您要提供作為此動作輸入的先前動作的人工因素。在先前動作中，必須將這些人工因素定義為輸出人工因素。

   如果您未指定任何輸入人工因素，則必須在下指定至少一個來源儲存庫*action-name*/Inputs/Sources。

   如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

   > **Note**
   >
   > 如果無法使用「成品-選用」下拉式清單 (視覺化編輯器)，或者在驗證 YAML (YAML 編輯器) 時發生錯誤，可能是因為動作僅支援一個輸入。在此情況下，請嘗試移除來源輸入。

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要指定成品做為動作的輸入 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在您要將成品指定為輸入的動作中，新增類似下列內容的程式碼：

```
action-name:
  Inputs:
    Artifacts:
      - artifact-name
```

如需更多範例，請參閱範例。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

參考人工因素中的檔案

如果您的檔案位於人工因素內，且需要在其中一個工作流程動作中參照此檔案，請完成下列程序。

> **Note**
>
> 另請參閱參考來源儲存庫中的檔案。

參照人工因素中的檔案

- 在您要參考檔案的動作中，新增類似下列內容的程式碼：

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - artifact-name
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_artifact-name/build-output && cat file.txt
```

在先前的程式碼中，動作會在人工因素`######build-output`目錄中尋找並顯示檔案。`file.txt`

如需更多範例，請參閱範例。

> **Note**
>
> 您可能可以省略`$CATALYST_SOURCE_DIR_artifact-name/`前綴，具體取決於您如何配置操作。如需詳細資訊，請參閱下列指引。

關於如何引用變量的指導：

- 如果您的動作在下只包含一個項目 Inputs (例如，它包含一個輸入成品而不包含來源)，則您可以省略前置詞，並僅指定相對於人工因素根目錄的檔案路徑。
- 如果檔案位於主要輸入中，您也可以省略前置詞。如果沒有，則主要輸入可以是WorkflowSource，或列出的第一個輸入成品WorkflowSource。
- 根據您使用的動作，前綴可能會有所不同。如需詳細資訊，請參閱下表。

動作類型	要使用的檔案路徑字首	範例
建置動作、測試動作	`$CATALYST_SOURCE_DIR_ artifact-name /`	`$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt`

所有其他動作	`$CATALYST_SOURCE_DIR_ artifact-name /`
	或
	`/artifacts/ current-action-name /artifact-name /########### $### _ # _ DIR _ ####/#`
		`$CATALYST_SOURCE_DIR_MyArtifact/folder1/file.txt`
		或
		`/artifacts/MyCurrentAction/MyArtifact/folder1/file.txt`

下載成品

您可以下載並檢查工作流程動作所產生的成品，以進行疑難排解。您可以下載兩種類型的神器：

- 來源人工因素 — 包含執行開始時存在的來源儲存庫內容快照的人工因素。
- 工作流程加工品 — 在工作流程組態檔案Outputs屬性中定義的人工因素。

若要依工作流程下載人工因素輸出

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。

3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. a 在工作流程名稱下，選擇 [執行]。
6. 在 [執行歷程記錄] 的 [執行 ID] 欄中，選擇執行。例如 Run-95a4d。
7. 在執行名稱下，選擇「成品」。
8. 選擇成品旁邊的 [下載]。存檔文件下載。它的文件名由七個隨機字符組成。
9. 使用您選擇的歸檔提取公用程序提取存檔。

範例

下列範例顯示如何在工作流程定義檔案中輸出及參考人工因素。

主題

- 範例：輸出人工因素
- 範例：輸入由其他動作產生的人工因素
- 範例：參考多個成品中的檔案
- 範例：參照單一人工因素中的檔案
- 範例：存在人工因素時參照檔案 WorkflowSource

範例：輸出人工因素

下列範例會示範如何輸出包含兩個 .jar 檔案的成品。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT1
          Files:
            - build-output/file1.jar
            - build-output/file2.jar
```

範例：輸入由其他動作產生的人工因素

下列範例說明如何輸出呼叫 ARTIFACT4 in 的加工品BuildActionA，並將其輸入BuildActionB。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT4
          Files:
            - build-output/file1.jar
            - build-output/file2.jar
  BuildActionB:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ARTIFACT4
    Configuration:
```

範例：參考多個成品中的檔案

下列範例說明如何輸出名為ART5和 ART6 in 的兩個成品BuildActionC，然後參考中 (下) 名為 `file5.txt` (在成品中ART5) 和 `file6.txt` (在成品中 ART6BuildActionD) 的兩個檔案。Steps

> **Note**
>
> 如需參考檔案的更多資訊，請參閱參考人工因素中的檔案。

> **Note**
>
> 雖然範例顯示正在使用的$CATALYST_SOURCE_DIR_ART5前置詞，但您可以省略它。這是因為ART5是主要輸入。若要深入瞭解主要輸入，請參閱參考人工因素中的檔案。

```
Actions:
  BuildActionC:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART5
          Files:
            - build-output/file5.txt
```

```
        - Name: ART6
          Files:
            - build-output/file6.txt
  BuildActionD:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART5
        - ART6
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART5/build-output && cat file5.txt
        - run: cd $CATALYST_SOURCE_DIR_ART6/build-output && cat file6.txt
```

範例：參照單一人工因素中的檔案

下面的例子說明如何輸出一個ART7在中命名的成品BuildActionE，然後在 file7.txt (下ART7) 中引用BuildActionF（在工件中Steps）。

請注意，參考如何不需要在build-output目錄前面的$CATALYST_SOURCE_DIR_######前綴，就像在中所做的那樣。範例：參考多個成品中的檔案這是因為在下只有一個項目指定Inputs。

> **Note**
>
> 如需參考檔案的更多資訊，請參閱參考人工因素中的檔案。

```
Actions:
  BuildActionE:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART7
          Files:
            - build-output/file7.txt
  BuildActionF:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART7
    Configuration:
      Steps:
```

```
        - run: cd build-output && cat file7.txt
```

範例：存在人工因素時參照檔案 WorkflowSource

下面的例子說明如何輸出一個ART8在中命名的成品BuildActionG，然後在 file8.txt (下ART8) 中引用BuildActionH（在工件中Steps）。

請注意參考如何需要$CATALYST_SOURCE_DIR_######前綴，就像在中一樣。範例：參考多個成品中的檔案這是因為在Inputs（源和成品）下指定了多個項目，因此您需要前綴來指示在何處查找文件。

> **Note**
>
> 如需參考檔案的更多資訊，請參閱參考人工因素中的檔案。

```
Actions:
  BuildActionG:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART8
          Files:
            - build-output/file8.txt
  BuildActionH:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - ART8
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART8/build-output && cat file8.txt
```

使用運算和執行階段環境 Docker 影像

在 CodeCatalyst 工作流程中，您可以指定 CodeCatalyst 用來執行工作流程動作的計算和執行階段環境影像。

計算是指由 CodeCatalyst 管理和維護以執行工作流程動作的計算引擎 (CPU、記憶體和作業系統)。

執行階段環境影像是 CodeCatalyst執行工作流程動作的 Docker 容器。Docker 容器會在您選擇的運算平台之上執行，並包含作業系統和工作流程動作可能需要的額外工具，例如 AWS CLI、Node.js 和 .tar。

主題

使用計算

計算是指由 CodeCatalyst 管理和維護以執行工作流程動作的計算引擎 (CPU、記憶體和作業系統)。

> **Note**
>
> 如果計算被定義為工作流程的屬性，則不能將其定義為該工作流程中任何動作的屬性。同樣地，如果計算被定義為任何動作的屬性，則無法在工作流程中定義它。

主題

關於運算類型

CodeCatalyst 提供下列運算類型：

- Amazon EC2
- AWS Lambda

Amazon EC2 在動作執行期間提供最佳化的彈性，Lambda 提供最佳化的動作啟動速度 由於啟動延遲較低，Lambda 支援更快的工作流程動作執行。Lambda 可讓您執行基本工作流程，以建置、測試和

部署具有一般執行階段的無伺服器應用程式。這些執行階段包括 Node.js、Python、Java、.NET 和圍棋。不過，有些使用案例不支援 Lambda，如果這些使用案例對您造成影響，請使用 Amazon EC2 運算類型：

- Lambda 不支援來自指定登錄的執行階段環境映像檔。
- Lambda 不支援需要根權限的工具。對於yum或之類的工具rpm，請使用 Amazon EC2 運算類型或其他不需要 root 許可的工具。
- Lambda 不支援泊塢視窗建置或執行。不支援以下使用 Docker 映像的動作：部署 AWS CloudFormation 堆疊、部署到 Amazon ECS、Amazon S3 發佈、 AWS CDK 引導、 AWS CDK 部署、 AWS Lambda 叫用和 GitHub 動作。Lambda 運算也不支援在動 CodeCatalyst GitHub 作動作中執行的碼頭式 GitHub 動作。您可以使用不需要 root 權限的替代方法，例如 Podman。
- Lambda 不支援寫入外部的檔案/tmp。設定工作流程動作時，您可以重新設定要安裝或寫入的工具。/tmp如果您有安裝的建置動作npm，請務必將其設定為安裝至/tmp。
- Lambda 不支援超過 15 分鐘的執行階段。

關於運算叢集

CodeCatalyst 提供下列運算叢集：

- 按需艦隊
- 佈建的艦隊

對於隨需叢集，當工作流程動作開始時，工作流程會佈建所需的資源。動作完成時，機器將被銷毀。您只需為執行動作的分鐘數付費。隨需叢集是全受管的，並包含自動擴充功能，可處理尖峰的需求。

CodeCatalyst 還提供佈建的叢集，其中包含由維護的 Amazon EC2 提供支援的機器。 CodeCatalyst透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。透過佈建的叢集，您的機器一律在執行中，只要佈建它們就會產生成本。

若要建立、更新或刪除叢集，您必須具有 S pace 管理員角色或專案管理員角色。

隨選叢集屬性

CodeCatalyst 提供下列隨選叢集：

名稱	作業系統	架構	vCPU	記憶體 (GiB)	磁碟空間	支援的運算類型
Linux.Arm64.Large	Amazon Linux 2	阿姆斯特	2	4	64 GB	Amazon EC2
					10 GB	Lambda
Linux.Arm64.XLarge	Amazon Linux 2	阿姆斯特	4	8	128 GB	Amazon EC2
					10 GB	Lambda
Linux.Arm64.2XLarge	Amazon Linux 2	阿姆斯特	8	16	128 GB	Amazon EC2
Linux.x86-64.Large	Amazon Linux 2	x86-64	2	4	64 GB	Amazon EC2
					10 GB	Lambda
Linux.x86-64.XLarge	Amazon Linux 2	x86-64	4	8	128 GB	Amazon EC2
					10 GB	Lambda
Linux.x86-64.2XLarge	Amazon Linux 2	x86-64	8	16	128 GB	Amazon EC2

如果未選取任何叢集，則 CodeCatalyst 使用Linux.x86-64.Large。

佈建的叢集屬性

已佈建的叢集包含下列內容：

作業系統

作業系統。以下是可用的作業系統：

- Amazon Linux 2
- Windows Server 2022

> **Note**
>
> Windows 叢集僅在建置動作中受支援。其他動作目前不支援視窗。

架構

處理器架構。以下是可用的架構：

- x86_64
- 阿姆斯特

機器類型

每個執行個體的機器類型。以下是可用的機器類型：

vCPU	記憶體 (GiB)	磁碟空間	作業系統
2	4	64 GB	Amazon Linux 2
4	8	128 GB	Amazon Linux 2
			Windows Server 2022
8	16	128 GB	Amazon Linux 2
			Windows Server 2022

容量

配置給叢集的初始機器數目，定義可以 parallel 執行的動作數目。

縮放模式

定義動作數目超過叢集容量時的行為。

根據需求提供額外容量

系統會依需求設定其他機器，這些機器會自動擴充以回應執行中的新動作，然後在動作完成時縮減至基本容量。這可能會產生額外的費用，因為您按分鐘支付每台運行的機器。

等到額外的叢集容量可用

動作執行會放置在佇列中，直到機器可用為止。這會限制額外的成本，因為沒有配置額外的機器。

建立、編輯和刪除已佈建的叢集

使用下列指示來建立、編輯和刪除已佈建的叢集。

> **Note**
>
> 佈建的艦隊將在閒置 2 週後停用。如果再次使用，它們將自動重新激活，但這種重新激活可能會導致延遲發生。

若要建立已佈建的叢集

1. 在功能窗格中，選擇 CI/CD，然後選擇 [計算]。
2. 選擇建立已佈建的叢集。
3. 在 [已佈建的叢集名稱] 文字欄位中，輸入叢集的名稱。
4. 從作業系統下拉式功能表中，選擇作業系統。
5. 從機器類型下拉式功能表中，選擇機器的機器類型。
6. 在 [容量] 文字欄位中，輸入叢集中機器的最大數目。
7. 從縮放模式下拉式功能表中，選擇所需的溢位行為。如需有關這些欄位的詳細資訊，請參閱 佈建的叢集屬性。
8. 選擇建立。

建立已佈建的叢集之後，您就可以準備將其指派給動作。如需詳細資訊，請參閱 將佈建的叢集或隨選運算指派給動作。

若要編輯已佈建的叢集

1. 在功能窗格中，選擇 CI/CD，然後選擇 [計算]。

2. 在 [已佈建的叢集] 清單中，選擇您要編輯的叢集。
3. 選擇編輯。
4. 在 [容量] 文字欄位中，輸入叢集中機器的最大數目。
5. 從縮放模式下拉式功能表中，選擇所需的溢位行為。如需有關這些欄位的詳細資訊，請參閱 佈建的叢集屬性。
6. 選擇儲存。

若要刪除已佈建的叢集

> ⚠ Warning
>
> 刪除佈建的叢集之前，請先從動作的 YAML 程式碼中刪除該`Fleet`內容，將其從所有動作中移除。刪除已佈建的叢集後繼續參照該叢集的任何動作都會在下次執行動作時失敗。

1. 在功能窗格中，選擇 CI/CD，然後選擇 [計算]。
2. 在 [已佈建的叢集] 清單中，選擇您要刪除的叢集。
3. 選擇刪除。
4. 輸入**`delete`**以確認刪除。
5. 選擇刪除。

將佈建的叢集或隨選運算指派給動作

依預設，工作流程動作會使用 Amazon EC2 運算類型的`Linux.x86-64.Large`隨需叢集。若要改用已佈建的叢集，或使用不同的隨選叢集，例如`Linux.x86-64.2XLarge`，請使用下列指示。

Visual

開始之前

- 如果您要指派已佈建的叢集，必須先建立已佈建的叢集。如需詳細資訊，請參閱 建立、編輯和刪除已佈建的叢集。

將已佈建的叢集或不同的叢集類型指派給動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要指派已佈建叢集或新叢集類型的動作。
8. 選擇 Configuration (組態) 索引標籤。
9. 在計算叢集中，執行下列動作：

   指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

   透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

   如`Fleet`果省略，預設值為`Linux.x86-64.Large`。
10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

開始之前

- 如果您要指派已佈建的叢集，必須先建立已佈建的叢集。如需詳細資訊，請參閱 建立、編輯和刪除已佈建的叢集。

將已佈建的叢集或不同的叢集類型指派給動作

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。

4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。 /
5. 選擇編輯。
6. 選擇 YAML。
7. 尋找您要指派已佈建叢集或新叢集類型的動作。
8. 在動作中，新增內Compute容並設定Fleet為叢集或隨選叢集類型的名稱。如需詳細資訊，請參閱「」中針對您的動作建立和測試動作參考的Fleet屬性說明。
9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

跨動作共用運算

主題

- 在共用運算上執行多個動作
- 計算共用的考量
- 開啟運算共用
- 範例

依預設，工作流程中的動作會在 eet 中的個別執行個體上執行。這種行為提供了對輸入狀態的隔離和可預測性的動作。預設行為需要明確設定，才能在動作之間共用上下文 (例如檔案和變數)。

計算共用是一項功能，可讓您在同一個執行個體上執行工作中的所有動作。使用計算共用可提供更快的工作執行時間，因為佈建執行個體所花費的時間較少。您也可以在動作之間共用檔案 (成品)，而不需要額外的工作流程限制。

使用計算共用執行工作流程時，預設或指定叢集中的執行個體會保留該工作流程中所有動作的持續時間。當工作流程執行完成時，會釋放執行個體保留項目。

在共用運算上執行多個動作

您可以在工作流程層級使用定義 YAML 中的Compute屬性來指定動作的叢集和計算共用內容。您也可以使用中的視覺化編輯器來設定計算屬性 CodeCatalyst。若要指定叢集，請設定現有叢集的名稱、將運算類型設定為 EC2，然後開啟運算共用。

> **Note**
>
> 只有在運算類型設定為 EC2 時，才支援運算共用，而且不支援 Windows 伺服器 2022 作業系統。如需有關運算叢集、運算類型和屬性的詳細資訊，請參閱使用計算。

> **Note**
>
> 如果您在免費方案上，並且在工作流程定義 YAML 中手動指定`Linux.x86-64.XLarge`或`Linux.x86-64.2XLarge`叢集，則動作仍會在預設叢集 (`Linux.x86-64.Large`) 上執行。如需有關運算可用性和定價的詳細資訊，請參閱各層選項的表格。

開啟計算共用時，包含工作流程來源的資料夾會自動跨動作複製。您不需要在整個工作流程定義 (YAML 檔案) 中設定輸出成品，並將其參考為輸入成品。身為工作流程作者，您需要使用輸入和輸出來連接環境變數，就像不使用計算共用一樣。如果您想要在工作流程來源之外的動作之間共用資料夾，請考慮檔案快取。如需詳細資訊，請參閱 使用人工因素 及 使用檔案快取。

工作流程定義檔案所在的來源儲存庫由標籤識別`WorkflowSource`。使用計算共用時，會在參考工作流程來源的第一個動作中下載工作流程來源，並自動讓工作流程執行中的後續動作可供使用。透過動作對包含工作流程來源的資料夾所做的任何變更 (例如新增、修改或移除檔案) 也會顯示在工作流程的後續動作中。您可以在任何工作流程動作中參考位於工作流程來源資料夾中的檔案，就像不使用計算共用一樣。如需詳細資訊，請參閱參考來源儲存庫中的檔案。

> **Note**
>
> 計算共用工作流程需要指定嚴格的動作順序，因此無法設定 parallel 動作。雖然可以在序列中的任何動作中設定輸出成品，但不支援輸入成品。

計算共用的考量

您可以透過運算共用來執行工作流程，以加速工作流程執行，並在使用相同執行個體的工作流程中的動作之間共用內容。請考慮下列事項，判斷使用計算共用是否適合您的案例：

	運算共用	不需共用運算
運算類型	Amazon EC2	Amazon EC2，AWS Lambda
實例佈建	動作在同一個實例上運行	動作在單獨的實例上運行
作業系統	Amazon Linux 2	Amazon Linux 2，視窗服務器 2022（僅適用於構建操作）
參考檔案	`$CATALYST_SOURCE_DIR_WorkflowSource` ，`/sources/WorkflowSource/`	`$CATALYST_SOURCE_DIR_WorkflowSource` ，`/sources/WorkflowSource/`
Workflow 結構	動作只能依序執行	動作可以並行執行
跨工作流程動作存取資料	存取快取的工作流程來源 (`WorkflowSource`)	存取共用成品的輸出 (需要其他組態)

開啟運算共用

使用下列指示開啟工作流程的計算共用。

Visual

使用視覺化編輯器開啟運算共用

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 選擇工作流程屬性。
8. 從 [運算類型] 下拉式功能表中，選擇 [EC2]。

9. (選擇性) 從 [計算叢集-選用] 下拉式功能表中，選擇要用來執行工作流程動作的叢集。您可以選擇隨需叢集，也可以建立並選擇已佈建的叢集。如需詳細資訊，請參閱 建立、編輯和刪除已佈建的叢集 和 將佈建的叢集或隨選運算指派給動作
10. 切換開關以開啟計算共用，並在同一叢集上執行工作流程中的動作。
11. (選擇性) 選擇工作流程的執行模式。如需詳細資訊，請參閱設定佇列、已取代和 parallel 執行。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

使用 YAML 編輯器開啟計算共用

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 開啟計算共用功能，將SharedInstance欄位設定Type為TRUE和為EC2。設定Fleet為您要用來執行工作流程動作的計算叢集。您可以選擇隨需叢集，也可以建立並選擇已佈建的叢集。如需詳細資訊，請參閱 建立、編輯和刪除已佈建的叢集 和 將佈建的叢集或隨選運算指派給動作

   在工作流程 YAML 中，新增類似下列內容的程式碼：

```
Name: MyWorkflow
SchemaVersion: "1.0"
Compute: # Define compute configuration.
  Type: EC2
  Fleet: MyFleet # Optionally, choose an on-demand or provisioned fleet.
  SharedInstance: true # Turn on compute sharing. Default is False.
Actions:
  BuildFirst:
    Identifier: aws/build@v1
    Inputs:
      Sources:
```

```
          - WorkflowSource
      Configuration:
        Steps:
          - Run: ...
          ...
```

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

範例

主題

- 範例：Amazon S3 發佈

範例：Amazon S3 發佈

下列工作流程範例顯示如何以兩種方式執行 Amazon S3 發佈動作：先使用輸入成品，然後使用運算共用。使用計算共用時，不需要輸入成品，因為您可以存取快取`WorkflowSource`。此外，不再需要「建置」動作中的輸出成品。S3 發佈動作設定為使用明確`DependsOn`屬性來維護連續動作；建置動作必須成功執行，S3 發佈動作才能執行。

- 如果沒有計算共用，您需要使用輸入成品，並與後續動作共用輸出：

```
Name: S3PublishUsingInputArtifact
SchemaVersion: "1.0"
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ArtifactToPublish
          Files: [output.zip]
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
```

```
    Inputs:
      Artifacts:
      - ArtifactToPublish
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
      DestinationBucketName: dev-bucket
```

- SharedInstance將設定為使用計算共用時TRUE，您可以在同一個執行個體上執行多個動作，並透過指定單一工作流程來源來共用人工因素。輸入加工品不是必需的，無法指定：

```
Name: S3PublishUsingComputeSharing
SchemaVersion: "1.0"
Compute:
  Type: EC2
  Fleet: dev-fleet
  SharedInstance: TRUE
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - Build
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
```

```
      DestinationBucketName: dev-bucket
```

使用執行階段環境 Docker 影像

執行階段環境影像是 CodeCatalyst執行工作流程動作的 Docker 容器。Docker 容器會在您選擇的運算平台之上執行，並包含作業系統和工作流程動作可能需要的額外工具，例如 AWS CLI、Node.js 和 .tar。

依預設，工作流程動作會在其中一個由提供和維護的作用中影像上執行 CodeCatalyst。只有構建和測試操作支持自定義映像。如需詳細資訊，請參閱 將自訂執行階段環境 Docker 影像指派給動作。

主題

- 作用中影像
- 如果使用中的影像不包含我需要的工具，該怎麼辦？
- 將自訂執行階段環境 Docker 影像指派給動作
- 範例

作用中影像

使用中映像檔是執行階段環境影像，受到預先安裝的工具完全支援 CodeCatalyst 並包含在內。目前有兩組活動圖像：一組於 2024 年 3 月發布，另一組於 2022 年 11 月發布。

動作是否使用 2024 年 3 月或 2022 年 11 月的圖像取決於動作：

- 在 2024 年 3 月 26 日或之後新增至工作流程的建置和測試動作將在其 YAML 定義中包含明確指定 2024 年 3 月映像的`Container`區段。您可以選擇性地移除`Container`區段，以回復到 2022 年 11 月的影像。
- 在 2024 年 3 月 26 日之前新增至工作流程的建置和測試動作將不會在其 YAML 定義中包含`Container`區段，因此將使用 2022 年 11 月映像。您可以保留 2022 年 11 月的圖像，也可以對其進行升級。若要升級映像檔，請在視覺化編輯器中開啟動作，選擇 [組態] 索引標籤，然後從 [執行階段環境 docker 視窗影像] 下拉式清單中選取 2024 年 3 月映像。此選取項目會在動作的 YAML 定義中新增`Container`區段，該區段會填入適當的 2024 年 3 月映像。
- 所有其他動作都將使用 2022 年 11 月的影像，無論這些影像何時新增至工作流程。目前無法升級這些動作以使用 2024 年 3 月的映像檔。

主題

二零二四年三月圖片

2024 年 3 月的圖像是提供的最新圖像。 CodeCatalyst每個運算類型/叢集組合都有一個 2024 年 3 月映像。

下表顯示每個 2024 年 3 月映像上安裝的工具。

三月圖像工具

工具	CodeCatalyst Amazon EC2-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst Lambda 的 **CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst Amazon EC2 **CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst Lambda 的 ARM64-**CodeCatalystLinuxLambda_Arm64:2024_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS 副駕駛 CLI	1.32.1	1.32.1	1.32.1	1.32.1
Docker	24.0.9	N/A	24.0.9	N/A
Docker Compose	2.23.3	N/A	2.23.3	N/A
Git	2.43.0	2.43.0	2.43.0	2.43.0
Go	1.21.5	1.21.5	1.21.5	1.21.5
Gradle	8.5	8.5	8.5	8.5
Java	科雷特托 17	科雷特托 17	科雷特托 17	科雷特托 1
Maven	3.9.6	3.9.6	3.9.6	3.9.6
Node.js	18.19.0	18.19.0	18.19.0	18.19.0
NPM	10.2.3	10.2.3	10.2.3	10.2.3

工具	CodeCatalyst Amazon EC2-**CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst Lambda 的 **CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst Amazon EC2 **CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst Lambda 的 ARM64-**CodeCatalystLinuxLambda_Arm64:2024_**
Python	3.9.18	3.9.18	3.9.18	3.9.18
Python3	3.11.6	3.11.6	3.11.6	3.11.6
pip	22.3.1	22.3.1	22.3.1	22.3.1
.NET	8.0.100	8.0.100	8.0.100	8.0.100

十一月圖片

每個運算類型/機隊組合都有一張 2022 年 11 月的映像檔。如果您已設定已佈建的叢集，建置動作也會提供 2022 年 11 月的 Windows 映像檔。

下表顯示 2022 年 11 月每個映像上安裝的工具。

十一月圖像工具

工具	CodeCatalyst Amazon EC2-**CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst Lambda 的 **CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst Amazon EC2 **CodeCatalystLinux_Arm64:2022_11**	CodeCatalyst Lambda 的 ARM64-**CodeCatalystLinuxLambda_Arm64:2022_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS 副駕駛 CLI	0.6.0	0.6.0	N/A	N/A
Docker	23.01	N/A	23.0.1	N/A
Docker Compose	2.16.0	N/A	2.16.0	N/A
Git	2.40.0	2.40.0	2.39.2	2.39.2

工具	CodeCatalyst Amazon EC2-**CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst Lambda 的 **CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst Amazon EC2 **CodeCatalystLinux_Arm64:2022_11**	CodeCatalyst Lambda 的 ARM64-**CodeCatalystLinuxLambda_Arm64:2022_**
Go	1.20.2	1.20.2	1.20.1	1.20.1
Gradle	8.0.2	8.0.2	8.0.1	8.0.1
Java	科雷特托 17	科雷特托 17	科雷特托 17	科雷特托 1
Maven	3.9.4	3.9.4	3.9.0	3.9.0
Node.js	16.20.2	16.20.2	16.19.1	16.14.2
NPM	8.19.4	8.19.4	8.19.3	8.5.0
Python	3.9.15	2.7.18	3.11.2	2.7.18
Python3	N/A	3.9.15	N/A	3.11.2
pip	22.2.2	22.2.2	23.0.1	23.0.1
.NET	6.0.407	6.0.407	6.0.406	6.0.406

如果使用中的影像不包含我需要的工具，該怎麼辦？

如果沒有任何由提供的活動圖像 CodeCatalyst 包括你需要的工具，你有幾個選項：

- 您可以提供包含必要工具的自訂執行階段環境 Docker 映像檔。如需詳細資訊，請參閱 將自訂執行階段環境 Docker 影像指派給動作。

  > **Note**
  >
  > 如果您想要提供自訂的執行階段環境 Docker 映像檔，請確定您的自訂映像檔已安裝 Git。

- 您可以讓工作流程的建置或測試動作安裝所需的工具。

例如，您可以在組建或測試動作的 YAML 程式碼的Steps區段中包含下列指示：

```
Configuration:
  Steps:
    - Run: ./setup-script
```

然後，####指令將運行以下腳本來安裝節點包管理器（npm）：

```
#!/usr/bin/env bash
echo "Setting up environment"

touch ~/.bashrc
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
source ~/.bashrc
nvm install v16.1.0
source ~/.bashrc
```

如需建置動作 YAML 的詳細資訊，請參閱建立和測試動作參考。

將自訂執行階段環境 Docker 影像指派給動作

如果您不想使用由提供的 Active 映像 CodeCatalyst，則可以提供自定義的運行時環境 Docker 映像。如果您想要提供自訂映像檔，請確定其中已安裝 Git。該映像可以位於碼頭集線器，Amazon 彈性容器註冊表或任何公共存儲庫中。

要了解如何創建自定義 Docker 映像，請參閱 Docker 文檔中的容器化應用程序。

請使用下列指示，將您的自訂執行階段環境 Docker 映像指派給動作。指定映像後，請在動作開始時將其 CodeCatalyst 部署到您的計算平台。

> **Note**
>
> 下列動作不支援自訂執行階段環境 Docker 映像檔：部署 AWS CloudFormation 堆疊、部署至 ECS 和GitHub 動作。自訂執行階段環境 Docker 映像檔也不支援 Lambda 運算類型。

Visual

若要使用視覺化編輯器指派自訂執行階段環境 Docker 影像

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
4. 選擇編輯。
5. 選擇 [視覺]。
6. 在工作流程圖中，選擇將使用您自訂執行階段環境 Docker 影像的動作。
7. 選擇 Configuration (組態) 索引標籤。
8. 在底部附近填寫以下欄位。

   運行時環境 Docker 映像-可選

   指定儲存映像檔的登錄。有效值包含：

   - `CODECATALYST`(YAML 編輯器)

     該映像存儲在 CodeCatalyst 註冊表中。
   - 碼頭集線器（可視化編輯器）或`DockerHub`（YAML 編輯器）

     該映像存儲在碼頭集線器映像註冊表中。
   - 其他註冊表（可視化編輯器）或`Other`（YAML 編輯器）

     影像儲存在自訂映像登錄中。任何公開可用的註冊表都可以使用。
   - Amazon 彈性容器註冊表（可視化編輯器）或`ECR`（YAML 編輯器）

     映像檔儲存在 Amazon 彈性容器登錄映像儲存庫中。若要在 Amazon ECR 儲存庫中使用映像檔，此動作需要存取 Amazon ECR。若要啟用此存取權，您必須建立包含下列許可和自訂信任政策的 IAM 角色。您可以視需要修改現有角色以包含權限和原則。)

     IAM 角色必須在其角色政策中包含以下許可：

     - `ecr:BatchCheckLayerAvailability`
     - `ecr:BatchGetImage`
     - `ecr:GetAuthorizationToken`

- ecr:GetDownloadUrlForLayer

IAM 角色必須包含下列自訂信任政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

如需建立 IAM 角色的詳細資訊，請參閱 IAM 使用者指南中的使用自訂信任政策 (主控台) 建立角色。

建立角色之後，您必須透過環境將其指派給動作。如需詳細資訊，請參閱 將環境、帳戶連線和 IAM 角色與工作流程動作建立關聯。

ECR 圖片網址、碼頭中心圖片或圖片網址

請指定下列其中一項：

- 如果您使用CODECATALYST登錄，請將映像設定為下列其中一個作用中的影像：
  - CodeCatalystLinux_x86_64:2024_03
  - CodeCatalystLinux_x86_64:2022_11
  - CodeCatalystLinux_Arm64:2024_03
  - CodeCatalystLinux_Arm64:2022_11
  - CodeCatalystLinuxLambda_x86_64:2024_03
  - CodeCatalystLinuxLambda_x86_64:2022_11

- `CodeCatalystLinuxLambda_Arm64:2024_03`
- `CodeCatalystLinuxLambda_Arm64:2022_11`
- `CodeCatalystWindows_x86_64:2022_11`

- 如果您使用的是 Docker Hub 註冊表，請將映像設置為 Docker Hub 映像名稱和可選標記。

  範例：`postgres:latest`
- 如果您使用的是 Amazon ECR 註冊表，請將映像設置為 Amazon ECR 註冊表 URI。

  範例：`111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo`
- 如果您使用自訂登錄，請將映像設定為自訂登錄所預期的值。

9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器指派自訂執行階段環境 Docker 影像

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 選擇編輯。
4. 選擇 YAML。
5. 尋找您要指派執行階段環境 Docker 影像的動作。
6. 在動作中，新增`Container`區段和基礎`Registry`和`Image`屬性。如需詳細資訊，請參閱中的`Container`、`Registry`和`Image`屬性的動作說明。
7. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
8. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

範例

下列範例顯示如何將自訂執行階段環境 Docker 影像指派給工作流程定義檔案中的動作。

主題

- 範例：使用自訂執行階段環境泊塢視窗映像檔，透過 Amazon ECR 新增對 Node.js 18 的支援

- 示例：使用自定義運行時環境泊塢窗映像添加對 Node.js 18 的支持與碼頭集線器

範例：使用自訂執行階段環境泊塢視窗映像檔，透過 Amazon ECR 新增對 Node.js 18 的支援

下列範例示範如何使用自訂執行階段環境泊塢視窗映像，透過 Amazon EC R 新增對 Node.js 18 的支援。

```
Configuration:
  Container:
    Registry: ECR
    Image: public.ecr.aws/amazonlinux/amazonlinux:2023
```

示例：使用自定義運行時環境泊塢窗映像添加對 Node.js 18 的支持與碼頭集線器

下面的例子演示了如何使用自定義運行時環境 Docker 映像添加對 Node.js 18 與碼頭集線器的支持。

```
Configuration:
  Container:
    Registry: DockerHub
    Image: node:18.18.2
```

使用環境

不要與開發環境混淆的環境是代碼部署到的地方。它通常包含一個正在運行的應用程序的實例及其相關的基礎設施。您可以為環境命名，例如開發、測試、測試或生產環境。對某個環境產生的 CodeCatalyst 任何部署都會顯示在「環境」頁面上。若要設定環境，請為其指定一個名稱，例如`my-production-environment`，然後將其與您的AWS 帳戶.

除了顯示部署資訊之外，環境還可作為將 AWS IAM 角色指派給工作流程動作的機制。

單一工作流程中是否可以存在多個環境？

是。如果工作流程包含多個動作，則每個動作都可以指派一個環境。例如，您可以擁有一個包含兩個部署動作的工作流程，其中一個動作會指派一個`my-staging-enviroment`環境，另一個則會指派`my-production-environment`環境。

哪些動作支援環境？

下列動作支援在「環境」頁面上顯示其部署資訊：

- 部署AWS CloudFormation堆疊 — 如需詳細資訊，請參閱 新增「部署 AWS CloudFormation 堆疊」動作
- 部署到 Amazon ECS — 如需詳細資訊，請參閱 新增「部署到 Amazon ECS」動作
- 部署至 Kubernetes 叢集 — 如需詳細資訊，請參閱 新增「部署至 Kubernetes 叢集」動作
- AWS CDK部署 — 如需詳細資訊，請參閱 新增「AWS CDK 部署」動作

> **Note**
>
> 如果您想要允許某個動作存取並執行AWS帳戶中的作業，則必須將其與環境建立關聯。許多動作都支援環境關聯，包括但不限於先前列出的動作。您可以判斷哪些動作支援環境關聯，因為這些動作會在視覺化編輯器的 [組態] 索引標籤上包含 [環境] 下拉式清單。

支援地區

「環境」頁面可以顯示任何AWS區域中的資源。

環境是強制性的嗎？

如果指派給該環境的工作流程動作將資源部署到AWS雲端，或基於其他原因 (例如監視和報告) 與AWS服務通訊，則該環境就是必要的。

建立環境

使用下列指示建立空的環境，以便稍後可與工作流程動作相關聯。

開始之前

您需要以下內容：

- 一個 CodeCatalyst 空間。如需詳細資訊，請參閱 正在設定 CodeCatalyst。
- 一個 CodeCatalyst 項目。如需詳細資訊，請參閱 使用藍圖建立專案。
- 包含工作流程動作將需要存取的 IAM 角色的AWS帳戶連線AWS。每個環境最多可以使用一個帳戶連線。如需詳細資訊，請參閱 管理 AWS 帳戶 空間。

> **Note**
>
> 您可以在沒有帳戶連線的情況下建立環境；不過，稍後您將需要返回並新增連線。

建立環境

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 在環境名稱中，輸入名稱，例如**Production**或**Staging**。
5. 在環境類型中，選取下列其中一項：

   - 非生產環境 — 在將應用程式移入生產環境之前，您可以測試應用程式以確保其正常運作。
   - 生產 — 一種「即時」環境，可公開使用並託管您的最終應用程式。

     如果您選擇「生產」，「生產」徽章會顯示在與環境相關聯的任何動作旁邊的 UI 中。徽章可協助您快速查看正在部署到生產環境的動作。除了徽章的外觀以外，生產環境和非生產環境之間沒有差異。
6. (選擇性) 在虛擬私人雲端連線中，選擇要與此環境建立關聯的 VPC 連線。如需有關建立此 VPC 連線的詳細資訊，請參閱《管理CodeCatalyst 員指南》中的「管理 Amazon 虛擬私有雲端」。
7. (選擇性) 在說明中，輸入說明，例如**Production environment for the hello-world app**。
8. 在連線-選用中，選擇您要與此環境產生關聯的AWS帳戶連線。確定帳戶連線包含您要與環境建立關聯的 IAM 角色。如需建立此連線的詳細資訊，請參閱管理 AWS 帳戶 空間。
9. 選擇 [建立環境]。 CodeCatalyst 創建一個空的環境。

後續步驟

- 現在，您已經建立了環境，就可以將其與工作流程動作相關聯。如需更多詳細資訊，請參閱 將環境、帳戶連線和 IAM 角色與工作流程動作建立關聯。

將環境、帳戶連線和 IAM 角色與工作流程動作建立關聯

當您將環境、帳戶連線和 IAM 角色與支援的工作流程動作建立關聯時，IAM 角色即可供動作使用。除了取得 IAM 角色的存取權之外，該動作還可能會將其部署資訊匯入「環境」頁面。如需詳細資訊，請參閱 哪些動作支援環境？。

使用下列指示將環境、帳戶連線和 IAM 角色與動作建立關聯。

步驟 1：將環境、帳戶連線和角色與工作流程動作相關聯

使用下列步驟將環境、帳戶連線和角色與工作流程動作相關聯。

Visual

使用可視化編輯器將環境、帳戶連線和角色與工作流程動作相關聯

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇環境支援的動作。如需詳細資訊，請參閱 哪些動作支援環境？。
8. 選擇「組態」頁籤，然後在欄位中指定資訊，如下所示。

   Environment (環境)

   指定要與動作搭配使用的 CodeCatalyst 環境。

   若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

   帳戶連接或連接-可選（取決於可用）

   指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

   如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

   Role

   指定此動作用於在 Amazon S3 和 Amazon ECR 等AWS服務中存取和操作的 IAM 角色名稱。確保此角色已添加到您的帳戶連接中。若要將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 如果角色具有足夠的權限，您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全性風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

如果您在清單中看不到該角色，這是因為您尚未將其與帳戶連線相關聯。如需詳細資訊，請參閱 將 IAM 角色新增至帳戶連線。

9. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
10. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

使用 YAML 編輯器將環境、帳戶連線和角色與工作流程動作產生關聯

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在您要與環境產生關聯的工作流程動作中，新增類似下列內容的程式碼：

```
action-name
  Environment:
    Name: environment-name
    Connections:
      - Name: account-connection-name
        Role: iam-role-name
```

如需詳細資訊，請參工作流定義參考閱您的動作。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

步驟 2：填入環境

將環境、帳戶連線和角色與工作流程動作產生關聯後，您可以在「環境」頁面中填入部署資訊。請使用下列指示來移入「環境」頁面。

> **Note**
>
> 「環境」頁面僅受工作流程動作的子集支援。如需詳細資訊，請參閱 哪些動作支援環境？。

移植環境的步驟

1. 如果在中提交變更時，工作流程執行未自動啟動步驟 1：將環境、帳戶連線和角色與工作流程動作相關聯，請依照下列步驟手動啟動執行：

   a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
   b. 選擇您要在其中開始執行的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
   c. 選擇執行。

   工作流程執行會啟動新的部署，導 CodeCatalyst 致在環境下新增您的應用程式資源資訊。
2. 確認您的應用程式資源出現在您的環境下：

   a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
   b. 選擇您的環境 (例如，`Production`)。
   c. 選擇「部署活動」索引標籤，並確認部署是否顯示「狀態」為「成功」。這表示工作流程執行已成功部署您的應用程式資源。
   d. 選擇 [部署目標] 索引標籤，並確認應用程式資源是否顯示。

管理環境

使用下列指示，透過將環境與 VPC 連線或建立關聯來管理環境。AWS 帳戶

建立 VPC 連線與環境的關聯

在具有 VPC 連線的環境中設定動作時，該動作將執行連線至 VPC，並遵循網路規則並存取相關聯 VPC 指定的資源。一或多個環境可以使用相同的 VPC 連線。

使用下列指示將 VPC 連線與環境建立關聯。

將 VPC 連線與環境建立關聯

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 選擇您的環境 (例如，`Production`)。
5. 選擇環境內容索引標籤。
6. 選擇 [管理 VPC 連線]，選擇您想要的 VPC 連線，然後選擇 [確認]。這會將您選取的 VPC 連線與此環境產生關聯。

如需詳細資訊，請參閱《管理CodeCatalyst 員指南》中的「管理 Amazon 虛擬私有雲端」。

建立AWS 帳戶與環境關聯

使用下列指示將環AWS 帳戶境關聯。

將環境AWS 帳戶與環境相關聯

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [環境]。
4. 選擇您的環境 (例如，`Production`)。
5. 選擇環境內容索引標籤。
6. 選擇「關聯」AWS 帳戶，選擇您想要的AWS 帳戶，然後選擇「關聯」。這會將您選取的項目 AWS 帳戶與此環境相關聯。

如需更多詳細資訊，請參閱 管理 AWS 帳戶 空間。

使用檔案快取

啟用檔案快取時，建置和測試動作會將磁碟上的檔案儲存至快取，並在後續工作流程執行時從該快取還原這些檔案。緩存減少了構建或下載運行之間沒有改變的依賴關係引起的延遲。 CodeCatalyst 也支援後援快取，可用來還原包含部分所需相依性的快取。這有助於減少快取遺漏的延遲影響。

> **Note**
>
> 檔案快取僅適用於 Amazon CodeCatalyst 建置和測試動作，且只有在設定為使用 EC2 運算類型時才能使用檔案快取。

主題

關於檔案快取

檔案快取可讓您將資料組織成多個快取，每個快取都會在`FileCaching`屬性下參照。每個緩存保存由給定路徑指定的目錄。指定的目錄將在 future 的工作流程執行中還原。以下是具有多個名為`cacheKey1`和`cacheKey2`的快取的快取的範例 YAML 程式碼片段。

```
Actions:
  BuildMyNpmApp:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
    Caching:
      FileCaching:
        cacheKey1:
          Path: file1.txt
          RestoreKeys:
             - restoreKey1
```

```
    cacheKey2:
      Path: /root/repository
      RestoreKeys:
         - restoreKey2
         - restoreKey3
```

> **Note**
>
> CodeCatalyst 使用多層緩存，其中包括一個本地緩存和遠程緩存。當佈建的叢集或隨選機器在本機快取上遇到快取未命中時，會從遠端快取還原相依性。因此，某些動作執行可能會因下載遠端快取而導致延遲。

CodeCatalyst 套用快取存取限制，以確保某個工作流程中的動作無法修改來自不同工作流程的快取。這樣可以保護每個工作流程，防止其他可能會推送影響組建或部署的不正確資料 使用緩存範圍強制執行限制，該緩存範圍將緩存隔離到每個工作流程和分支配對。例如，workflow-Ain branch feature-A 具有與同級分支feature-B不同workflow-A的文件緩存。

當工作流程尋找指定的檔案快取且無法找到它時，就會發生快取遺漏。這可能是由於多種原因而發生的，例如當創建新分支時或引用新緩存並且尚未創建緩存時。它也可能發生在緩存到期時，默認情況下發生在上次使用後 14 天。若要緩解快取遺漏並提高快取命中率，請 CodeCatalyst 支援後援快取。後援快取是替代快取，提供還原部分快取的機會，部分快取可以是較舊版本的快取。透過首先搜尋屬性名稱下的相符項目來還原快取，如果找不到，則會FileCaching進行評估RestoreKeys。如果屬性名稱和所有內容都有快取未命中RestoreKeys，工作流程將繼續執行，因為快取是最好的努力，而且不能保證。

建立快取

您可以使用下列指示將快取新增至工作流程。

Visual

若要使用視覺化編輯器新增快取

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。

5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇要新增快取的動作。
8. 選擇 Configuration (組態)。
9. 在 [檔案快取-選用] 底下，選擇 [新增快取]，然後在欄位中輸入資訊，如下所示：

   索引鍵

   指定主要快取屬性名稱的名稱。快取屬性名稱在工作流程中必須是唯一的。每個動作最多可以有五個項目`FileCaching`。

   路徑

   指定快取的相關路徑。

   還原金鑰-選用

   指定找不到主要快取屬性時要用作後援的還原金鑰。還原金鑰名稱在您的工作流程中必須是唯一的。每個快取中最多可有五個項目`RestoreKeys`。
10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器新增快取

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在工作流程動作中，新增類似下列內容的程式碼：

```
action-name:
  Configuration:
```

```
    Steps: ...
  Caching:
    FileCaching:
      key-name:
        Path: file-path
        # # Specify any additional fallback caches
        # RestoreKeys:
        #   - restore-key
```

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

Constraints

以下是性質名稱和的約束`RestoreKeys`：

- 名稱在工作流程中必須是唯一的。
- 名稱僅限於英數字元 (A-Z、a-z、0-9)、連字號 (-) 和底線 (_)。
- 名稱最多可包含 180 個字元。
- 每個動作最多可以有五個快取`FileCaching`。
- 每個快取中最多可有五個項目`RestoreKeys`。

以下是路徑的限制：

- 不允許使用星號 (*)。
- 路徑最多可包含 255 個字元。

使用套件

套件是同時包含軟體與中繼資料的套裝軟體，以及安裝軟體並解決任何相依性所需的中繼資料。CodeCatalyst 支持 npm 包格式。

一個軟件包包括：

- 名稱 (例如，`webpack`是常用 npm 套件的名稱)
- 選用的命名空間 (例如，`@types`在中`@types/node`)
- 一組版本 (例如`1.0.0`、`1.0.1`、`1.0.2`)

- 套件層級中繼資料 (例如 npm dist 標籤)

在中 CodeCatalyst，您可以將套 CodeCatalyst 裝軟體發佈到工作流程中的套裝程式，並使用套裝程式儲存庫 您可以使用 CodeCatalyst 套件存放庫來設定建置或測試動作，以自動設定動作的 npm Client，以便從指定的儲存庫推送和提取套件。

如需封裝的詳細資訊，請參閱中的套件 CodeCatalyst。

> **Note**
>
> 目前，建置和測試動作支援 CodeCatalyst 套件儲存庫。

主題

在工作流程中使用 CodeCatalyst 套件庫

在中 CodeCatalyst，您可以將 CodeCatalyst 套件存放庫新增至您的組建，並在工作流程中測試動作。您的套件存放庫必須設定套件格式，例如 npm。您也可以選擇包含所選套裝程式儲存區域的一系列範圍。

請使用下列指示來指定要與工作流程動作搭配使用的封裝組態。

Visual

若要指定動作將使用的套件組態 (視覺化編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖表中，選擇您要使用套裝程式儲存區域設定的動作。

8. 選擇「套件」。
9. 從 [新增設定] 下拉式功能表中，選擇您要與工作流程動作搭配使用的封裝組態。
10. 選擇新增套裝程式儲存庫
11. 在「P ackage 程式儲存區域」下拉式功能表中，指定您要動作使用的 CodeCatalyst 套裝程式儲存區域名稱。

    如需套裝程式儲存庫的詳細資訊，請參閱Package 儲存庫。
12. （可選）在範圍-可選中，指定要在套件登錄中定義的範圍序列。

    定義範圍時，指定的套裝程式儲存區域會設定為所有列出範圍的登錄。如果通過 npm 客戶端請求具有範圍的包，它將使用該存儲庫而不是默認的。每個範圍名稱必須加上前綴「@」。

    如果省Scopes略，則會將指定的套裝程式儲存區域設定為動作所使用之所有套裝程式的預設登錄。

    如需有關範圍的詳細資訊，請參閱Package 命名空間和範圍封裝。
13. 選擇 Add (新增)。
14. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
15. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要指定動作將使用的套件組態 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在動作中，新增類似下列內容的程式碼：

```
action-name:
 Configuration:
    Packages:
```

```
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: package-repository
              Scopes:
                - "@scope"
```

如需詳細資訊，請參閱中建立和測試動作參考針對您動作的Packages屬性說明。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

在工作流程中使用套件的範例

下列範例顯示如何參照工作流程定義檔案中的封裝。

主題

- 範例：定義套件 NpmConfiguration
- 範例：覆寫預設登錄
- 範例：覆寫套件登錄中的範圍

範例：定義套件 **NpmConfiguration**

下列範例顯示如何在工作流程定義檔案NpmConfiguration中定義封裝。

```
Actions:
  Build:
  Identifier: aws/build-beta@v1
  Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: main-repo
            - PackagesRepository: scoped-repo
              Scopes:
                - "@scope1"
```

這個例子配置 npm 客戶端如下：

```
default: main-repo
```

```
@scope1: scoped-repo
```

在此範例中，定義了兩個儲存庫。預設登錄的設定main-repo為沒有範圍的定義。範圍@scope1已在中設PackageRegistries定scoped-repo。

範例：覆寫預設登錄

下列範例說明如何覆寫預設登錄。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-repo-1
    - PackagesRepository: my-repo-2
    - PackagesRepository: my-repo-3
```

這個例子配置 npm 客戶端如下：

```
default: my-repo-3
```

如果您指定多個預設儲存庫，最後一個存放庫將優先處理。在此示例中，列出的最後一個存儲庫是my-repo-3，這意味著 npm 將連接到my-repo-3。這將覆蓋存儲庫my-repo-1和my-repo-2.

範例：覆寫套件登錄中的範圍

下列範例說明如何覆寫套件登錄中的範圍。

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-default-repo
    - PackagesRepository: my-repo-1
      Scopes:
        - "@scope1"
        - "@scope2"
    - PackagesRepository: my-repo-2
      Scopes:
        - "@scope2"
```

這個例子配置 npm 客戶端如下：

```
default: my-default-repo
@scope1: my-repo-1
```

```
@scope2: my-repo-2
```

如果您包含覆蓋範圍，則最後一個存儲庫將優先考慮。在此範例中，上次在中設定@scope2該範圍的時間PackageRegistries為my-repo-2。這會覆寫設@scope2定的範圍my-repo-1。

使用梯段

執行是工作流程的單一版序。在執行期間， CodeCatalyst執行工作流程組態檔案中定義的動作，並輸出相關聯的記錄、人工因素和變數。

主題

- 啟動工作流程執行
- 停止工作流程執行
- 檢視工作流程執行狀態與詳細
- 設定佇列、已取代和 parallel 執行

啟動工作流程執行

使用下列程序來手動啟動工作流程執行。

> **Note**
>
> 您也可以透過設定觸發器來自動啟動工作流程執行。

開始手動執行工作流程的步驟

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇您要執行的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇執行。

停止工作流程執行

使用下列程序停止正在進行的工作流程執行。如果它是意外啟動的，您可能想停止運行。

當您停止工作流程執行時，會 CodeCatalyst 等待進行中的動作完成，然後才會在主控台中將執行標示為「已停止」。 CodeCatalyst 任何沒有機會開始的動作將不會開始，而且會標示為「已放棄」。

> **Note**
>
> 如果執行已佇列 (也就是說，它沒有進行中的動作)，則會立即停止執行。

若要停止工作流程執行

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 在「工作流程」下，選擇「執行」，然後從清單中選擇進行中的執行。
5. 選擇 Stop (停止)。

檢視工作流程執行狀態與詳細

您可以檢視單一工作流程執行的狀態和詳細資訊，或同時檢視多個執行的狀態和詳細資訊。

> **Note**
>
> 此外，您還可以查看与工作流程運行狀態不同的工作流程狀態。如需詳細資訊，請參閱 檢視工作流程狀態。

主題

- 工作流程運行狀
- 檢視單次執行的狀態和詳細資訊
- 檢視專案中所有執行的狀態和詳細資訊
- 查看特定工作流程的所有運行狀態和詳細信息
- 檢視工作流程圖表中工作流程的執行

工作流程運行狀

工作流程運行可以包含下列其中一種狀態：

- 成功 — 已成功處理工作流程執行。
- 失敗 — 工作流程執行中的一或多個動作失敗。
- 進行中 — 目前正在處理工作流程執行。
- 已停止 — 一個人停止工作流程執行，而它正在進行中。
- [停止] — 目前正在停止工作流程執行。
- 「已取消」— 工作流程運行已被取消， CodeCatalyst 因為在運行進行中時刪除或更新了相關的工作流程。
- 已取代 — 只有在您已配置已取代的執行模式時才會發生。工作流程執行已被取消， CodeCatalyst 因為稍後的工作流程執行已取代它。

檢視單次執行的狀態和詳細資訊

您可以查看單個工作流程運行的狀態和詳細信息，以檢查它是否成功，查看完成時間，或者查看誰或者是什麼啟動了它。

若要檢視單一執行的狀態和詳細資訊

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 在工作流程名稱下，選擇「執行」。
6. 在 [執行歷程記錄] 的 [執行 ID] 欄中，選擇執行。例如 Run-95a4d。
7. 在執行名稱下，執行下列其中一項作業：

   - 以視覺方式查看工作流程圖表，顯示工作流程執行的動作及其狀態 (請參閱工作流程運行狀)。此檢視也會顯示執行期間使用的來源儲存庫和分支。

     在工作流程圖中，選擇動作以查看執行期間動作所產生的詳細資訊，例如記錄檔、報告和輸出。顯示的資訊取決於選取的動作類型。如需檢視組建或部署記錄的詳細資訊，請參閱檢視建構動作的結果或檢視部署記錄。
   - YAML 以查看用於執行的工作流程定義檔案。
   - 人工因素，以查看工作流程執行所產生的人工因素。如需成品的詳細資訊，請參閱 使用人工因素。

- 可查看測試報告和工作流程執行所產生的其他類型報告的報告。如需報告的詳細資訊，請參閱測試報告類型。
- 用於查看工作流程執行所產生的輸出變數的變數。如需變數的更多資訊，請參閱使用變數。

> **Note**
>
> 如果已刪除執行的父項工作流程，則會在執行詳細資訊頁面頂端顯示一則訊息，指出此事實。

檢視專案中所有執行的狀態和詳細資訊

您可以查看項目中所有工作流程運行的狀態和詳細信息，以了解項目中發生了多少工作流程活動，并了解工作流程的整體健康狀況。

若要檢視專案中所有執行的狀態和詳細資訊

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 在「工作流程」下選擇「執行

   會顯示所有工作流程、所有分支、專案中所有儲存庫的所有執行。

   此頁面包含下列資料欄：

   - 執行 ID — 執行的唯一識別碼。選擇執行 ID 連結以檢視執行的詳細資訊。
   - 「狀態」 — 工作流程運行的處理狀態。如需執行狀態的詳細資訊，請參閱工作流程運行狀。
   - 觸發程序 — 開始工作流程執行的人員、提交、提取請求 (PR) 或排程。如需詳細資訊，請參閱使用觸發程序。
   - 工作流程 — 啟動執行的工作流程名稱，以及工作流程定義檔案所在的來源存放庫和分支。您可能需要展開欄寬才能查看此資訊。

     > **Note**
     >
     > 如果此欄設定為 [無法使用]，通常是因為已刪除或移動相關聯的工作流程。

- 開始時間 — 工作流程執行開始的時間。
- 持續時間 — 工作流程執行需要多長時間才能處理。很長或很短的持續時間可能表明出現問題。
- 結束時間 — 工作流程執行結束的時間。

查看特定工作流程的所有運行狀態和詳細信息

您可能想要檢視與特定工作流程相關聯之所有執行的狀態和詳細資訊，以查看是否有任何執行在工作流程中建立瓶頸，或者查看目前正在進行中或已完成的執行。

若要檢視特定工作流程的所有執行狀態和詳細資訊

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 在工作流程名稱下，選擇「執行」。

   與所選工作流程相關聯的執行會出現。

   該頁面分為兩個部分：

   - 作用中的執行 — 顯示進行中的執行。這些執行會處於下列其中一種狀態：進行中。
   - 執行歷程記錄 — 顯示已完成 (也就是未進行中) 的執行。

     如需執行狀態的詳細資訊，請參閱工作流程運行狀。

檢視工作流程圖表中工作流程的執行

您可以檢視工作流程中所有執行工作流程的狀態。執行會顯示在工作流程圖表中 (而不是在清單檢視中)。這可讓您以視覺化方式呈現哪些執行正由哪些動作處理，以及佇列中正在等待哪些執行。

若要在工作流程中同時檢視多個執行的狀態

> **Note**
>
> 只有當您的工作流程使用佇列或已取代的執行模式時，才適用此程序。如需詳細資訊，請參閱設定佇列、已取代和 parallel 執行。

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇包含您要檢視之執行的工作流程名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。

   > **Note**
   >
   > 確保您正在查看工作流程頁面，而不是運行頁面。

5. 選擇左上角的 [最新狀態] 索引標籤。

   工作流程圖表隨即出現。
6. 檢閱工作流程圖表。此圖表顯示工作流程中目前正在進行的所有執行，以及已完成的最新執行。更具體地說：

   - 顯示在「來源」之前頂端的執行會排入佇列並等待開始。
   - 動作之間出現的執行會排入佇列，並等待下一個動作處理。
   - 動作目前正在處理，2. 已完成動作處理，或 3. 未由動作處理 (通常是因為先前的動作失敗)。

設定佇列、已取代和 parallel 執行

依預設，當同時執行多個工作流程時，會將它們排入 CodeCatalyst 佇列，並依照啟動的順序逐一處理它們。您可以透過指定執行模式來變更此預設行為。有幾種運行模式：

- (預設) 佇列執行模式 — CodeCatalyst 程序逐一執行
- 被取代的運行模式- CodeCatalyst 進程逐個運行，較新的運行超過舊的運行
- 並行運行模式- CodeCatalyst 進程並行運行

主題

關於佇列執行模式

在佇列執行模式中，執行會以串列方式發生，等待執行會形成佇列。

佇列會在動作和動作群組的進入點處形成，因此您可以在同一個工作流程中擁有多個佇列 (請參閱Figure 1)。當排入佇列的執行進入動作時，該動作將被鎖定，且其他執行都無法進入。當執行完成並結束動作時，動作會變成解除鎖定，並準備好進行下一次執行。

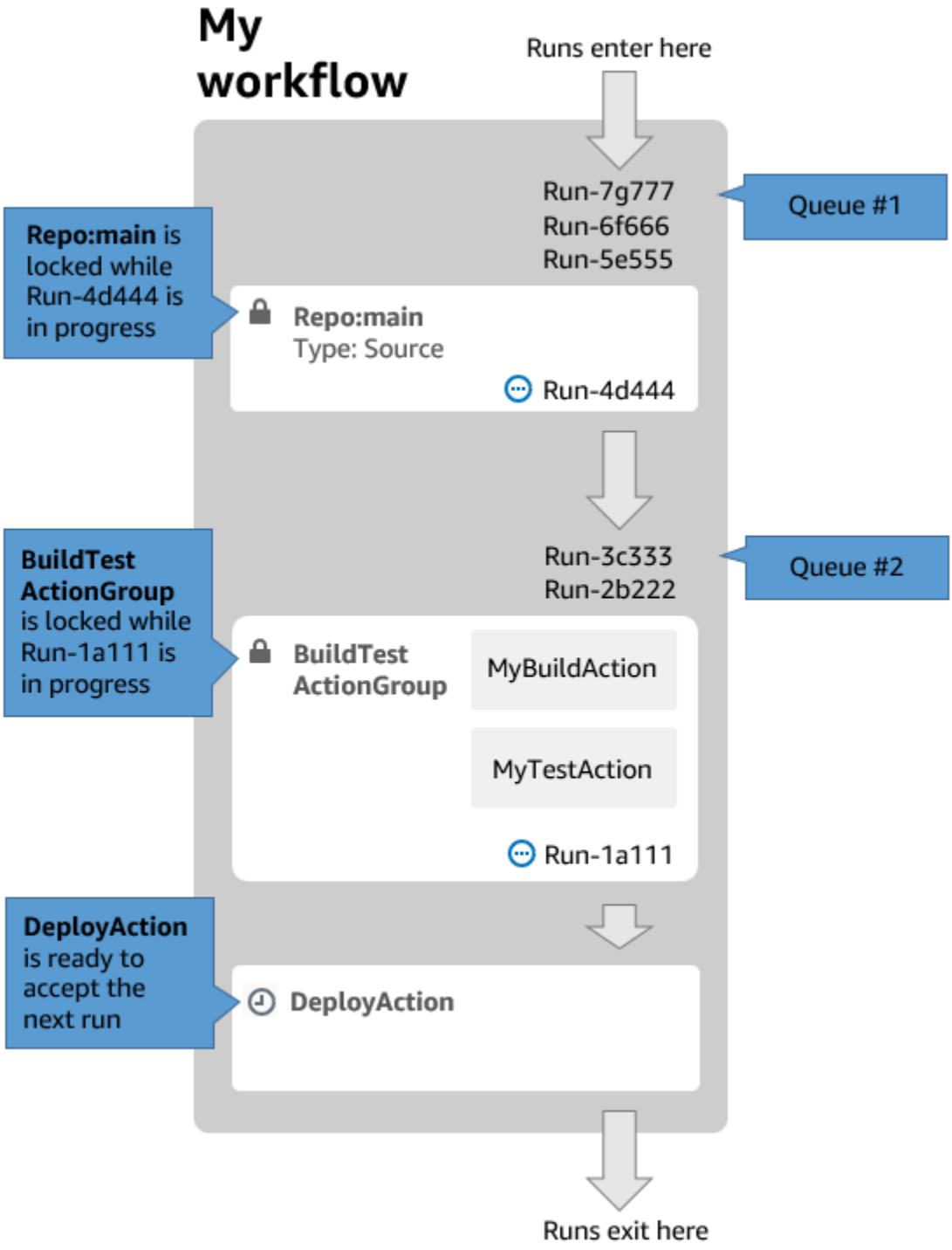
Figure 1說明了在佇列執行模式中配置的工作流程。它顯示：

- 七個運行通過工作流程他們的方式.
- 兩個佇列：一個位於輸入來源項目之外 (RePO: Main)，另一個佇列位於動作項目之外。BuildTestActionGroup
- 兩個鎖定的塊：輸入源（REPO：主）和. BuildTestActionGroup

以下是工作流程運行完成處理時，事情將如何發生：

- 當 Run-4D444 完成複製來源儲存庫時，它將結束輸入來源並加入 Run-3C333 後面的佇列。然後，運行 5E555 將進入輸入源。
- 當運行 1A111 完成構建和測試時，它將退出BuildTestActionGroup操作並進入操作。DeployAction然後，運行 -2B2222 將進入該動作。BuildTestActionGroup

圖 1：在「佇列執行模式」中設定的工作流程



在下列情況下使用佇列執行模式：

- 您想要在圖徵和管路之間保持 one-to-one 關係 — 使用取代模式時，這些特徵可能會被分組。例如，當您在提交 1 中合併圖徵 1 時，執行 1 開始，並在提交 2 中合併圖徵 2 時，執行 2 開始，依此類推。如果您要使用取代模式而不是排隊模式，則您的功能（和提交）將在運行中分組在一起，以取代其他功能。

- 您想要避免使用 parallel 模式時可能發生的競爭狀況和未預期的問題。例如，如果兩個軟體開發人員 Wang 和 Saanvi 在大致同一時間執行工作流程以部署到 Amazon ECS 叢集，則 Wang 的執行可能會在叢集上開始整合測試，而 Saanvi 的執行會將新的應用程式程式碼部署到叢集，導致 Wang 的測試失敗或測試錯誤的程式碼。作為另一個例子，您可能有一個沒有鎖定機制的目標，在這種情況下，這兩個運行可能會以意想不到的方式覆蓋彼此的更改。
- 您想要限制 CodeCatalyst 用來處理執行的運算資源的負載。例如，如果您的工作流程中有三個動作，您最多可以同時發生三個執行。對一次可能發生的執行次數施加限制，可讓執行輸送量更加可預測。
- 您想要限制工作流程對第三方服務提出的要求數量。例如，您的工作流程可能具有構建操作，其中包含從 Docker Hub 提取映像的說明。Docker Hub 將您可以發出的提取請求數量限制為每個帳戶每小時的特定數量，如果超出限制，您將被鎖定。使用佇列執行模式減慢執行輸送量，每小時對 Docker Hub 產生要求的影響會減少，進而限制鎖定以及造成建置和執行失敗的可能性。

最大佇列大小：50

佇列大小上限的注意事項：

- 佇列大小上限是指工作流程中所有佇列允許的最大執行次數。
- 如果佇列的執行時間超過 50 次，則會 CodeCatalyst 捨棄第 51 次及後續執行。

失敗行為：

如果執行在動作處理時變得沒有回應，則其後面的執行會保留在佇列中，直到動作逾時為止。動作會在一小時後逾時。

如果在動作中執行失敗，則允許在其後面排入佇列的第一個執行繼續。

關於已取代的執行模式

已取代的執行模式與排入佇列的執行模式相同，不同之處在於：

- 如果佇列的執行追蹤到佇列中的另一個執行，則稍後的執行會取代 (接管) 先前的執行，而先前的執行會被取消並標示為「已取代」。
- 作為第一個 bullet 中描述的行為的結果，當使用取代的運行模式時，隊列只能包含一次運行。

使用中的工作流程作Figure 1為指南，將已取代的執行模式套用至此工作流程會產生下列結果：

- Run-7g777 將取代其隊列中的其他兩個運行，並且將是隊列 #1 中唯一剩餘的運行。運行 6F666 和運行- 5E555 將被取消。
- 運行 3C333 將取代運行 2b222，並成為唯一剩餘在佇列 #2 中的運行。運行 -2b222 將被取消。

如果需要，請使用已取代的執行模式：

- 比排隊模式更好的吞吐量
- 與排隊模式相比，對第三方服務的請求甚至更少；如果第三方服務具有速率限制（例如 Docker Hub），則這很有利

關於 parallel 執行模式

在 parallel 執行模式中，執行彼此獨立，而且在開始之前不要等待其他執行完成。沒有佇列，執行輸送量僅受工作流程內動作完成的速度所限制。

在每個使用者都有自己的功能分支，並部署至其他使用者未共用的目標的開發環境中使用 parallel 執行模式。

> **Important**
>
> 如果您有多個使用者可以部署到的共用目標 (例如生產環境中的 Lambda 函數)，請勿使用 parallel 模式，因為可能會導致競爭情形。當 parallel 工作流程執行嘗試同時變更共用資源，導致無法預期的結果時，就會發生爭用情形。

parallel 執行的最大數目：每個 CodeCatalyst空間 1000

變更執行模式

您可以將執行模式設定為已排入佇列、已取代或 parallel。預設值為佇列。

當您將執行模式從佇列中變更或已取代為 parallel 時，會 CodeCatalyst 取消排入佇列的執行，並允許動作目前正在處理的執行完成，然後再取消它們。

當您將執行模式從 parallel 變更為佇列或已取代時，會 CodeCatalyst 讓所有目前執行的 parallel 執行完成。您在將執行模式變更為佇列或已取代之後啟動的任何執行都會使用新模式。

Visual

使用視覺化編輯器變更執行模式

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 在右上角，選擇「工作流程屬性」。
7. 展開「進階」，然後在「執行」模式下，選擇下列其中一項：
   a. 已排入佇列 — 請參閱 關於佇列執行模式
   b. 已取代 — 請參閱 關於已取代的執行模式
   c. 平行 — 請參閱 關於 parallel 執行模式
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要使用 YAML 編輯器變更執行模式

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 添加屬RunMode性，如下所示：

```
Name: Workflow_6d39
SchemaVersion: "1.0"
RunMode: QUEUED|SUPERSEDED|PARALLEL
```

如需詳細資訊，請參閱的一頂層屬性節中的RunMode屬性說明工作流定義參考。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

與秘密一起工作

有時候您可能需要在工作流程中使用敏感資料，例如驗證認證。應該避免將這些值以純文本形式存儲在存儲庫中的任何位置，因為任何可以訪問包含密碼的存儲庫的人都可以看到它們。同樣地，這些值不應該直接在任何工作流程定義中使用，因為它們在存放庫中會顯示為檔案。使用 CodeCatalyst，您可以在專案中加入密碼，然後參考工作流程定義檔案中的密碼來保護這些值。請注意，每個動作最多可以有五個密碼。

> **Note**
>
> 密碼只能用來取代工作流程定義檔案中的密碼和敏感資訊。

主題

建立密碼

使用下列程序來建立密碼。密碼包含您要從檢視中隱藏的敏感資訊。

> **Note**
>
> 密碼在動作中可見，並且在寫入檔案時不會遮罩。

若要建立機密

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 在功能窗格中，選擇 CI/CD，然後選擇 [密碼]。
3. 選擇 Create secret (建立機密)。
4. 輸入下列資訊：

   名稱

   輸入密碼的名稱。

   Value

   輸入密碼的值。這是您要從檢視中隱藏的敏感資訊。依預設，不會顯示該值。若要顯示值，請選擇「顯示值」。

   Description

   (選擇性) 輸入密碼的說明。
5. 選擇建立。

編輯密碼

請使用下列程序來編輯密碼。

若要編輯密碼

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在功能窗格中，選擇 CI/CD，然後選擇 [密碼]。
3. 在密碼清單中，選擇您要編輯的密碼。
4. 選擇編輯。
5. 編輯下列屬性：

   Value

   輸入密碼的值。這是您要從檢視中隱藏的值。依預設，不會顯示該值。

   Description

   (選擇性) 輸入密碼的說明。
6. 選擇儲存。

使用秘密

欲在工作流程動作中使用密碼，您必須取得密碼的參照識別元，並在工作流程動作中使用該識別元。

主題

獲取密碼的標識符

請使用下列程序來取得密碼的參照識別元。您會將此識別碼新增至您的工作流程。

若要取得密碼的參照識別碼

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在功能窗格中，選擇 CI/CD，然後選擇 [密碼]。
3. 在秘密清單中，找到您要使用的秘密。
4. 在「參照 ID」欄中，複製密碼的識別碼。以下是參考 ID 的語法：

```
${Secrets.<name>}
```

參考工作流程中的密碼

請使用下列程序來參照工作流程中的密碼。

若要參照密碼

1. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
2. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
3. 選擇編輯。
4. 選擇 YAML。
5. 修改 YAML 以使用密碼的識別碼。例如，若要透過`curl`指令使用儲存為密碼的使用者名稱和密碼，您可以使用類似下列的`Run`命令：

```
- Run: curl -u <username-secret-identifier>:<password-secret-identifier> https://
example.com
```

6. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
7. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

刪除密碼

使用下列程序來刪除密碼和秘密參照識別元。

> **Note**
>
> 刪除密碼之前，建議您從所有工作流程動作中移除密碼的參照識別元。如果您在未刪除參照識別元的情況下刪除密碼，則該動作在下次執行時會失敗。

若要從工作流程中刪除密碼的參照識別碼

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
3. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
4. 選擇編輯。
5. 選擇 YAML。
6. 在工作流程中搜尋下列字串：

   ```
   ${Secrets.
   ```

   這會尋找所有機密的所有參考識別碼。
7. 刪除所選密碼的參照識別碼，或以純文字值取代它。
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

刪除秘密

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在功能窗格中，選擇 CI/CD，然後選擇 [密碼]。
3. 在密碼清單中，選擇您要刪除的密碼。
4. 選擇刪除。
5. 輸入**delete**以確認刪除。
6. 選擇刪除。

使用來源

來源 (也稱為輸入來源) 是工作流程動作需要存取才能執行其工作的來源儲存庫。例如，工作流程動作可能需要存取來源，才能取得單元測試，並針對應用程式來源檔案執行這些測試。

CodeCatalyst 工作流程支援下列來源：

- CodeCatalyst 來源儲存庫 — 如需詳細資訊，請參閱來源儲存庫 CodeCatalyst。
- GitHub 來源儲存庫 — 如需詳細資訊，請參閱在中使用 GitHub 儲存庫 CodeCatalyst。

主題

- 指定將儲存工作流程定義檔案的來源
- 指定工作流程動作將使用的來源
- 參考來源儲存庫中的檔案
- 源所產生的變數

指定將儲存工作流程定義檔案的來源

請使用下列指示來指定您要儲存工作流程定義檔案的 CodeCatalyst 來源儲存庫。如果您想要指定來 GitHub 源儲存庫，請參閱在中使用 GitHub 儲存庫 CodeCatalyst。

您的工作流程定義檔案所在的來源儲存庫由標籤識別`WorkflowSource`。

> **Note**
>
> 您可以在第一次提交工作流程定義檔案時，指定工作流程定義檔所在的來源儲存庫。在此認可之後，儲存庫和工作流程定義檔案會永久連結在一起。在初始提交後變更存放庫的唯一方法是在不同的存放庫中重新建立工作流程。

若要指定將儲存工作流程定義檔案的來源儲存庫

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇 [建立工作流程] 並建立工作流程 如需詳細資訊，請參閱使用視覺化編輯器建立工作流程。

   在工作流程建立過程中，系統會要求您指定 CodeCatalyst儲存工作流程定義檔案的存放庫。

指定工作流程動作將使用的來源

請使用下列指示來指定要與工作流程動作搭配使用的來源存放庫。啟動時，動作會將已設定來源儲存庫中的檔案捆綁到成品中，將成品下載至執行動作所在的執行階段環境 Docker 映像，然後使用下載的檔案完成其處理。

> **Note**
>
> 目前，在工作流程動作中，您只能指定一個來源儲存庫，這是工作流程定義檔案所在的來源儲存庫 (位於`.codecatalyst/workflows/`目錄中)。此來源儲存庫由標籤表示`WorkflowSource`。

Visual

若要指定動作將使用的來源儲存庫 (視覺化編輯器)

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。

4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要指定來源的動作。
8. 選擇「輸入」。
9. 在來源中-選擇性執行下列動作：

   指定代表動作所需之來源儲存庫的標籤。目前唯一支援的標籤是`WorkflowSource`，它代表儲存工作流程定義檔案的來源儲存庫。

   如果您省略來源，則必須在下指定至少一個輸入成品*`action-name`*`/Inputs/Artifacts`。

   如需來源的詳細資訊，請參閱使用來源。
10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要指定動作將使用的來源儲存庫 (YAML 編輯器)

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在動作中，新增類似下列內容的程式碼：

```
action-name:
 Inputs:
   Sources:
     - WorkflowSource
```

如需詳細資訊，請參閱中工作流定義參考針對您動作的`Sources`屬性說明。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

參考來源儲存庫中的檔案

如果您有位於來源存放庫中的檔案，且您需要在其中一個工作流程動作中參考這些檔案，請完成下列程序。

> **Note**
>
> 另請參閱 參考人工因素中的檔案。

參照來源儲存庫中的檔案

- 在您要參考檔案的動作中，新增類似下列內容的程式碼：

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Configuration:
        Steps:
        - run: cd my-app && cat file1.jar
```

在先前的程式碼中，動作會在`WorkflowSource`來源儲存庫根`my-app`目錄中尋找並顯示`file1.jar`檔案。

源所產生的變數

當工作流程執行時，其來源會產生可在後續工作流程動作中使用的變數。如需詳細資訊，請參閱「BranchName" 和 "CommitId" 變數中的預先定義的變數清單。

使用觸發程序

工作流程觸發器 (或只是觸發器) 可讓您在某些事件發生時 (例如程式碼推送) 自動啟動工作流程執行。您可能想要設定觸發程序，讓軟體開發人員不必透過 CodeCatalyst 主控台手動啟動工作流程執行。

您可以使用三種類型的觸發器：

- 推送 — 程式碼推送觸發程序會在推送提交時啟動工作流程執行。
- 提取請求 — 提取請求觸發程序會在建立、修訂或關閉提取請求時啟動工作流程執行。
- 「時間表」— 計划觸發器會使工作流程按照您定義的時間表開始運行。請考慮使用排程觸發程式來執行軟體的每晚組建，以便軟體開發人員可以在第二天早上進行工作。

您可以單獨或在相同的工作流程中組合使用推送、提取請求和排程觸發器。

> **Tip**
>
> 若要查看觸發器的實際運作，請使用藍圖啟動專案。大多數藍圖都包含帶有觸發器的工作流程。在藍圖的工作流程定義檔案中尋找內容。`Trigger`如需有關藍圖的詳細資訊，請參閱 使用藍圖建立專案。

主題

一種常見的觸發配置

本節說明如何設定一般軟體發行版本和分支策略的觸發程式。

軟體發行與分支策略：

- 您在源存儲庫中有應用程序代碼。
- 您的`main`分支包含始終可以發布的最終代碼。
- 您的軟件開發人員在分支之外的功能分`main`支中進行更改。
- 您的軟件開發人員創建一個提取請求，要求將其功能分支合併到其功能準備就緒`main`時。

  您希望此提取要求能夠使用軟體開發人員功能分支上的檔案，自動啟動建置和測試 (但不部署) 應用程式的工作流程。

- 您的軟件開發人員檢查構建和測試，以確保一切看起來都不錯。然後，他們將提取請求合併到main分支中。

  您希望合併自動啟動建置和部署應用程式程式碼的工作流程。

建議的工作流程/觸發器配置：

根據先前概述的軟體分支策略，您可能需要使用兩個工作流程：

- 當建立或修訂提取要求時，工作流程 1 會建立並測試您的應用程式。
- 當合併提取請求時，工作流程 2 會建立和部署您的應用程式。

工作流程 1 看起來像這樣：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
Actions:
  BuildAction:
    instructions-for-building-the-app
  TestAction:
    instructions-for-test-the-app
```

每當軟體開發人員建立提取要求 (或修改) 要求將其功能分支合併到分支時，先前的觸發程式碼就會自動啟動工作流程執main行。 CodeCatalyst 使用來源分支 (也就是開發人員的功能分支) 中的程式碼來啟動工作流程執行。工作流程會建置和部署應用程式。

工作流程 2 看起來像這樣：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildAction:
    instructions-for-building-the-app
```

```
  DeployAction:
    instructions-for-deploying-the-app
```

在前面的觸發代碼中，當合併到發main生時，PUSH觸發器被激活。 CodeCatalyst 使用main分支中的代碼（現在包括提取請求的代碼）啟動工作流運行。工作流程會建置和部署應用程式。

如需將觸發器新增至工作流程定義檔案的說明，請參閱新增推送、拉取或排程觸發器。

如需觸發程序的更多範例和其他說明，請參閱觸發器的例子。

分支時的觸發考量

本節說明設定包含分支的觸發程序時的一些主要考量事項。

- 考量 1：對於推送和提取請求觸發程序，如果您要指定分支，則必須在觸發程序配置中指定目的地（或「to」）分支。永遠不要指定源（或「來自」）分支。

  在下列範例中，從任何分支推送以main啟動工作流程。

  ```
  Triggers:
    - Type: PUSH
      Branches:
        - main
  ```

  在下列範例中，從任何分支到的提取要求main會啟動工作流程。

  ```
  Triggers:
    - Type: PULLREQUEST
      Branches:
        - main
      Events:
        - OPEN
        - REVISION
  ```

- 考量 2：對於推送觸發程序，在啟動工作流程之後，工作流程將使用目標分支中的工作流程定義檔案和來源檔案來執行。
- 考量 3：對於提取請求觸發程序，在啟動工作流程之後，工作流程將使用來源分支中的工作流程定義檔案和來源檔案執行 (即使您在觸發程序組態中指定了目的地分支)。
- 考慮 4：一個分支中完全相同的觸發器可能無法在另一個分支中運行。

  請考慮下列推送觸發器：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

如果包含此觸發器的工作流程定義檔案已存在main且被複製到中test，則工作流程將永遠不會自動開始使用中的檔案 test (儘管您可以手動啟動工作流程以使用中的檔案test)。檢閱考量 1 和 2，以瞭解為什麼工作流程永遠不會使用中的檔案自動執行test。

另請考慮下列提取要求觸發程序：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

如果中存在包含此觸發器的工作流程定義檔案main，則工作流程將永遠不會使用中的檔案來執行main。(但是，如果您建立關閉的test分支main，工作流程將會使用中的檔案執行test。) 檢閱考量 1 和 3 以瞭解原因。

新增推送、拉取或排程觸發器

使用下列指示，將推送、提取或排程觸發器新增至工作流程。

Visual

若要新增觸發器 (視覺化編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。

6. 選擇 [視覺]。
7. 在工作流程圖表中，選擇「來源與觸發程式」方塊。
8. 在設定窗格中，選擇 [新增觸發器]。
9. 在「新增觸發器」對話方塊中，在欄位中提供資訊，如下所示。

   觸發類型

   指定觸發器的類型。您可以使用下列其中一個值：

   - 推送 (視覺化編輯器) 或 PUSH (YAML 編輯器)

     推送觸發程序會在將變更推送至來源儲存庫時啟動工作流程執行。工作流程執行將使用您要推送到的分支 (也就是目標分支) 中的檔案。

   - 提取請求（可視化編輯器）或PULLREQUEST（YAML 編輯器）

     在來源儲存庫中開啟、更新或關閉提取要求時，提取要求觸發程序會啟動工作流程執行。工作流程執行將使用您要從中提取的分支中的檔案 (也就是來源分支)。

   - 排程 (視覺化編輯器) 或 SCHEDULE (YAML 編輯器)

     排程觸發程序會根據您指定的 cron 運算式所定義的排程開始工作流程執行。將使用分支的檔案為來源儲存庫中的每個分支啟動個別的工作流程執行。若要限制觸發程序啟動的分支，請使用「分支」欄位 (視覺化編輯器) 或Branches屬性 (YAML 編輯器)。)

     設定排程觸發器時，請遵循下列準則：

     - 每個工作流程僅使用一個排程觸發器。
     - 如果您在 CodeCatalyst 空間中定義了多個工作流程，我們建議您排程不超過 10 個工作流程以同時啟動。
     - 確保在運行之間有足夠的時間配置觸發器的 cron 表達式。如需詳細資訊，請參閱 Expression。

   如需範例，請參閱 觸發器的例子。

   提取請求的事件

   只有在您選取提取要求觸發程式類型時，才會顯示此欄位。

   指定將啟動工作流程執行的提取要求事件類型。以下是有效值：

- 創建拉請求（可視化編輯器）或OPEN（YAML 編輯器）

  建立提取請求時，會啟動工作流程執行。

- 拉請求已關閉（可視化編輯器）或CLOSED（YAML 編輯器）

  當提取請求關閉時，會啟動工作流程執行。該CLOSED事件的行為很棘手，並且最好通過一個示例來理解。如需更多資訊，請參閱範例：具有拉動、分支和「CLOSE」事件的觸發器。

- 新的修訂做了拉請求（可視化編輯器）或REVISION（YAML 編輯器）

  工作流程執行會在建立提取請求的修訂時啟動。建立提取請求時會建立第一個修訂版本。之後，每當有人將新提交推送到提取請求中指定的源分支時，都會創建一個新的修訂版本。如果您將REVISION事件包含在提取要求觸發程序中，您可以省略該OPEN事件，因為REVISION這是的OPEN超集合。

您可以在同一個提取請求觸發程序中指定多個事件。

如需範例，請參閱 觸發器的例子。

排程

只有在您選取「排程」觸發類型時，才會顯示此欄位。

指定描述何時執行排程工作流程執行的 cron 運算式。

中的 Cron 運算式 CodeCatalyst 使用下列六個欄位語法，其中每個欄位都以空格分隔：

####days-of-month#days-of-week#

cron 表達式的例子

分鐘	小時	月份中的天數	月	星期中的天數	年	意義
0	0	?	*	MON-FRI	*	在每個星期一至星期五的午夜 (UTC +0) 執行工作流程。
0	2	*	*	?	*	每天在凌晨 2:00 (UTC+0) 執行工作流程。
15	22	*	*	?	*	每天在晚上 10:15 (UTC+0) 執行工作流程。
0/30	22-2	?	*	SAT-SUN	*	在開始日的晚上 10:00 至翌日凌晨 2:00 之間，每 30 分鐘執行一個工作流程 (UTC +0)。

分鐘	小時	月份中的天數	月	星期中的天數	年	意義
45	13	L	*	?	2023-2027	在 2023 年到 2027 年 (含 2027 年) 之間每月最後一天的下午 1:45 (UTC +0) 執行工作流程。

在中指定 cron 運算式時 CodeCatalyst，請確定遵循下列準則：

- 為每個SCHEDULE觸發器指定一個 cron 運算式。
- 在 YAML 編輯器中，以雙引號 ('') 括住 cron 運算式。
- 以國際標準時間 (UTC) 為單位指定時間。不支援其他時區。
- 設定執行間隔至少 30 分鐘。不支援較快的節奏。
- 指定*days-of-month*或*days-of-week*欄位，但不能同時指定兩者。如果您在其中一個欄位中指定值或星號 (*)，則必須在另一個欄位中使用問號 (?)。星號表示「全部」，問號表示「任何」。

如需 Cron 運算式的更多範例和萬用字元 (例如?、和L) 的相關資訊*，請參閱 Amazon EventBridge 使用者指南中的 Cron 運算式參考。Cron 表達式 EventBridge 和 CodeCatalyst 工作方式完全相同。

如需排程觸發器的範例，請參閱觸發器的例子。

分支和分支模式

(選用)

指定觸發程式監視的來源儲存庫中的分支，以便知道何時開始工作流程執行。您可以使用正則表達式模式來定義分支名稱。例如，使用`main.*`來比對以開頭的所有分支`main`。

根據觸發器類型，要指定的分支會有所不同：

- 對於推送觸發器，請指定要推送到的分支，也就是目標分支。每個匹配的分支將使用匹配分支中的文件啟動一個工作流程運行。

  範例: `main.*`, `mainline`
- 對於提取要求觸發程序，請指定您要推送到的分支，也就是目的地分支。每個匹配的分支將使用工作流程定義文件和源文件（而不是匹配的分支）啟動一個工作流程運行。

  範例：`main.*`,`mainline`, `v1\-.*` (符合開頭為的分支`v1-`)
- 對於排程觸發器，請指定包含您要排程執行使用之檔案的分支。每個匹配的分支將使用工作流程定義文件和匹配分支中的源文件啟動一個工作流程運行。

  範例: `main.*`, `version\-1\.0`

> **Note**
>
> 如果您未指定分支，則觸發程序會監視來源儲存庫中的所有分支，並使用下列項目中的工作流程定義檔案和來源檔案啟動工作流程執行：
>
> - 您正在推送的分支（用於推送觸發器）。如需詳細資訊，請參閱 範例：簡單的程式碼推送觸發器。
> - 您從中提取的分支（用於拉取請求觸發器）。如需詳細資訊，請參閱 範例：簡單的拉取要求觸發程序。
> - 所有分支 (用於排程觸發器)。來源儲存庫中的每個分支都會啟動一個工作流程執行。如需詳細資訊，請參閱 範例：一個簡單的排程觸發。

如需有關分支和觸發程序的更多資訊，請參閱分支時的觸發考量。

如需更多範例，請參閱 觸發器的例子。

檔案已變更

只有在選取「推送」或「提取」要求觸發程式類型時，才會顯示此欄位

指定觸發程式監視的來源儲存庫中的檔案或資料夾，以便知道何時開始執行工作流程。您可以使用規則運算式來比對檔案名稱或路徑。

如需範例，請參閱 觸發器的例子。

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要新增觸發程式 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 使用下列範例做為指南，新增`Triggers`區段和基礎屬性。如需詳細資訊，請參閱《Triggers》中的 工作流定義參考。

代碼推送觸發器可能如下所示：

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

拉取要求觸發程序可能如下所示：

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - OPEN
      - REVISION
```

```
      - CLOSED
```

排程觸發器可能如下所示：

```
Triggers:
  - Type: SCHEDULE
    Branches:
      - main.*
    # Run the workflow at 1:15 am (UTC+0) every Friday until the end of 2023
    Expression: "15 1 ? * FRI 2022-2023"
```

如需可在Expression屬性中使用的 cron 運算式的更多範例，請參閱Expression。

如需推送、提取要求和排程觸發程序的更多範例，請參閱觸發器的例子。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

觸發器的例子

下列範例顯示如何在工作流程定義檔案中新增不同類型的觸發器。

主題

- 範例：簡單的程式碼推送觸發器
- 示例：一個簡單的「推送到主」觸發器
- 範例：簡單的拉取要求觸發程序
- 範例：一個簡單的排程觸發
- 範例：具有排程和分支的觸發器
- 範例：具有排程、推送和分支的觸發器
- 範例：具有拉動和分支的觸發器
- 範例：具有拉動、分支和「CLOSE」事件的觸發器
- 範例：具有推送、分支和檔案的觸發器

範例：簡單的程式碼推送觸發器

下列範例顯示每當程式碼推送至來源儲存庫中的任何分支時，就會啟動工作流程執行的觸發程序。

啟動此觸發程序時， CodeCatalyst 會使用您要推送到的分支 (也就是目標分支) 中的檔案來啟動工作流程執行。

例如，如果您將提交推送至`main`，則會使用 workfow 定義檔案和其他來源檔案來 CodeCatalyst 啟動工作流程執行。`main`

另一個範例是，如果您將提交推送至`feature-branch-123`，則會使用 workfow 定義檔案和其他來源檔案 CodeCatalyst 啟動工作流程執行。`feature-branch-123`

```
Triggers:
  - Type: PUSH
```

> **Note**
>
> 如果您希望工作流程只在您推送至時才開始執行`main`，請參閱示例：一個簡單的「推送到主」觸發器。

示例：一個簡單的「推送到主」觸發器

下列範例顯示的觸發程序會在將程式碼推送至來源儲存庫中的分支 `main` (只有*`main`*分支) 時，啟動工作流程執行。

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

範例：簡單的拉取要求觸發程序

下列範例顯示每當在來源儲存庫中建立或修訂提取要求時，就會啟動工作流程執行的觸發程序。

啟動此觸發程序時， CodeCatalyst 會使用工作流程定義檔案和您要從中提取的分支 (也就是來源分支) 中的其他來源檔案來啟動工作流程執行。

例如，如果您使用名為的來源分支建立提取請求，`feature-123`而目的地分支名為`main`，則會使用 workfow 定義檔案和其他來源檔案來 CodeCatalyst 啟動工作流程執行。`feature-123`

```
Triggers:
```

```
  - Type: PULLREQUEST
    Events:
      - OPEN
      - REVISION
```

範例：一個簡單的排程觸發

下列範例顯示在每個星期一至星期五午夜 (UTC+0) 啟動工作流程執行的觸發程序。

啟動此觸發程序時， CodeCatalyst 會針對包含含有此觸發程序的工作流程定義檔案的來源儲存庫中的每個分支啟動單一工作流程執行。

例如，如果您的來源儲存庫、、、中有三個分支 `main` `release-v1` `feature-123`，並且每個分支都包含一個工作流程定義檔案，則會 CodeCatalyst 啟動三個工作流程執行：一個使用中的檔案`main`，另一個使用中的檔案`release-v1`，另一個使用中的檔案`feature-123`。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 ? * MON-FRI *"
```

如需可在`Expression`屬性中使用的 cron 運算式的更多範例，請參閱Expression。

範例：具有排程和分支的觸發器

下列範例顯示每天下午 6:15 (UTC+0) 啟動工作流程執行的觸發程序。

啟動此觸發器時，使用`main`分支中的檔案 CodeCatalyst 啟動工作流程執行，並為開頭的每個分支啟動其他執行`release-`。

例如，如果您的來源儲存庫`bugfix-2`中有名為`main` `release-v1` `bugfix-1`、、和的分支，則會 CodeCatalyst 啟動兩個工作流程執行：一個使用中的檔案`main`，另一個使用中的檔案`release-v1`。它不會啟動`bugfix-1`和分支的工作流程執`bugfix-1`行。

```
Triggers:
  - Type: SCHEDULE
    Expression: "15 18 * * ? *"
    Branches:
      - main
      - release\-.*
```

如需可在`Expression`屬性中使用的 cron 運算式的更多範例，請參閱Expression。

範例：具有排程、推送和分支的觸發器

下列範例顯示的觸發程序會在每天午夜 (UTC+0) 啟動工作流程執行，以及每當程式碼推送至分支時。main

在此範例中：

- 工作流程執行會在每天午夜開始執行。工作流程執行使用工作流程定義檔案和main分支中的其他來源檔案。
- 每當您將提交推送至main分支時，工作流程執行也會啟動。工作流程執行使用工作流程定義檔案和目標分支 (main) 中的其他來源檔案。

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 * * ? *"
    Branches:
      - main
  - Type: PUSH
    Branches:
      - main
```

如需可在Expression屬性中使用的 cron 運算式的更多範例，請參閱Expression。

範例：具有拉動和分支的觸發器

下列範例會顯示每當有人開啟或修改具有名為目標分支的提取要求時，就會啟動工作流程執行的觸發程序main。雖然Triggers設定中指定的分支是main，但工作流程執行會使用工作流程定義檔案，以及來源分支 (這是您要從中提取的分支) 中的其他來源檔案。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

範例：具有拉動、分支和「CLOSE」事件的觸發器

下列範例顯示每當在開頭為的分支上關閉提取要求時，就會啟動工作流程執行的觸發程序main。

在此範例中：

- 當您使用以開頭的目的地分支關閉提取請求時main，工作流程執行會自動開始使用工作流程定義檔案和 (現在已關閉) 來源分支中的其他來源檔案。
- 如果您已將源存儲庫配置為在拉取請求合併後自動刪除分支，則這些分支將永遠無法進入CLOSED狀態。這意味著合併的分支不會激活拉取請求CLOSED觸發器。在此案例中啟動CLOSED觸發程序的唯一方法是關閉提取要求而不合併它。

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - CLOSED
```

範例：具有推送、分支和檔案的觸發器

下列範例顯示了每當對main分支上的檔案或src目錄中的任何filename.txt檔案進行變更時，就會啟動工作流程執行的觸發程序。

啟動此觸發器時， CodeCatalyst 使用工作流程定義檔案和分支中的其他來源檔案啟動工作流程執main行。

```
Triggers:
  - Type: PUSH
    Branches:
      - main
    FilesChanged:
      - filename.txt
      - src\/.*
```

使用變數

變數是包含您可以在 CodeCatalyst 工作流程中參照的資訊的索引鍵值配對。

您可以在工作流程中使用兩種類型的變數：

- 使用者定義的變數 — 這些是您定義的索引鍵值配對。
- 預先定義的變數 — 這些是由工作流程自動發出的索引鍵值配對。您不需要定義它們。

> **Note**
>
> CodeCatalyst 也支援GitHub 輸出參數，其行為類似於變數，可在其他動作中參考。如需詳細資訊，請參閱 使用 GitHub 動作輸出參數。

主題

- 使用使用者定義
- 使用預定義變量

使用使用者定義

使用者定義的變數是您定義的索引鍵值配對。有兩種類型：

- 純文字變數，或單純變數 — 這些是您在工作流程定義檔案中以純文字定義的索引鍵值配對。
- 秘密 — 這些是您在 Amazon CodeCatalyst 主控台的個別「機密」頁面上定義的鍵值配對。金鑰 (name) 是公開標籤，值包含您要保持私密性的資訊。您只能在工作流程定義檔案中指定鍵。使用密碼代替工作流程定義檔案中的密碼和其他敏感資訊。

> **Note**
>
> 為了簡潔起見，本指南使用術語變量來表示純文本變量。

主題

- 定義一個變量
- 定義一個秘密
- 匯出變數，以便其他動作可以使用它
- 在定義它的動作中引用變量
- 透過另一個動作參考變數輸出
- 引用一個秘密
- 範例

定義一個變量

您可以透過兩種方式定義變數：

- 在工作流程動作Inputs區段中 — 請參閱在「輸入」一節中定義變數
- 在工作流程動作Steps區段中 — 請參閱在「步驟」一節中定義變數

> **Note**
>
> 該Steps方法僅適用於 CodeCatalyst 構建，測試和GitHub 操作操作，因為這些是包含Steps部分的唯一操作。

如需範例，請參閱範例。

Visual

若要在「輸入」區段中定義變數 (視覺化編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要設定變數的動作。
8. 選擇「輸入」。
9. 在變數-選用中，選擇新增變數，然後執行下列動作：

   指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

   如需有關變數的更多資訊 (包括範例)，請參閱使用變數。
10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要在「輸入」區段中定義變數 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在工作流程動作中，新增類似下列內容的程式碼：

```
action-name:
  Inputs:
    Variables:
      - Name: variable-name
        Value: variable-value
```

如需更多範例，請參閱 範例。如需詳細資訊，請參工作流定義參考閱您的動作。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

Visual

若要在「步驟」區段 (視覺化編輯器) 中定義變數

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要設定變數的動作。

8. 選擇 Configuration (組態)。
9. 在命令介面命令或GitHub動作 YAML 中，無論是可用的，都可以明確或隱含地定義動作中的變數。Steps

   - 要明確定義變量，請將其直接包含在 bash 命令Steps中。
   - 若要隱含定義變數，請在動作Steps區段中參照的檔案中指定變數。

   如需範例，請參閱範例。如需詳細資訊，請參工作流定義參考閱動作的。
10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要在「步驟」區段中定義變數 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在工作流程動作中，明確或隱含地在動作的Steps區段中定義變數。

   - 要明確定義變量，請將其直接包含在 bash 命令Steps中。
   - 若要隱含定義變數，請在動作Steps區段中參照的檔案中指定變數。

   如需範例，請參閱範例。如需詳細資訊，請參工作流定義參考閱動作的。
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

定義一個秘密

您可以在主控台的 [密碼] 頁面上定義密 CodeCatalyst碼。如需詳細資訊，請參閱與秘密一起工作。

例如，您可以定義如下所示的密碼：

- 名稱 (金鑰): **my-password**
- 值：**^*H3#!b9**

在定義密碼之後，您可以在工作流程定義檔案中指定密碼的金鑰 (**my-password**)。如需如何執行此作業的範例，請參閱 範例：參照密碼。

匯出變數，以便其他動作可以使用它

使用下列指示從動作匯出變數，以便您可以在其他動作中參考變數。

匯出變數之前，請注意下列事項：

- 如果您只需要在定義變數的動作中參考變數，則不需要匯出變數。
- 並非所有動作都支援匯出變數。若要判斷您的動作是否支援此功能，請執行後續的視覺化編輯器指示，並查看動作是否包含「輸出」索引標籤上的「變數」按鈕。如果是，則支援匯出變數。
- 若要從 GitHub 動作匯出變數，請參閱匯 GitHub 出輸出參數，以便其他動作可以使用該參數。

先決條件

請確定您已定義要匯出的變數。如需詳細資訊，請參閱定義一個變量。

Visual

匯出變數 (視覺化編輯器) 的步驟

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您要從中匯出變數的動作。
8. 選擇「輸出」。
9. 在變數-選用中，選擇新增變數，然後執行下列動作：

   指定您要匯出動作的變數名稱。此變數必須已在相同動作的`Inputs`或`Steps`區段中定義。

10. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
11. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

YAML

若要匯出變數 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在您要從中匯出變數的動作中，新增類似下列內容的程式碼：

```
action-name:
  Outputs:
    Variables:
      - Name: variable-name
```

如需更多範例，請參閱 範例。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

在定義它的動作中引用變量

使用下列指示，在定義變數的動作中參考變數。

> **Note**
>
> 若要參考由 GitHub 動作產生的變數，請參閱引用輸 GitHub出參數。

先決條件

請確定您已定義要參考的變數。如需詳細資訊，請參閱定義一個變量。

Visual

不可用。選擇 YAML 以檢視 YAML 指示。

YAML

若要在定義變數的動作中參照變數

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在定義您要參考之變數的 CodeCatalyst 動作中，請使用下列 bash 語法新增變數：

   ```
   $variable-name
   ```

   例如：

   ```
   MyAction:
       Configuration:
         Steps:
           - Run: $variable-name
   ```

   如需更多範例，請參閱 範例。如需詳細資訊，請參閱中的動作參考資訊工作流定義參考。
8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

透過另一個動作參考變數輸出

請使用下列指示來參考其他動作輸出的變數。

> **Note**
>
> 若要參考 GitHub 動作中的變數輸出，請參閱引用輸 GitHub出參數。

先決條件

請確定您已匯出要參照的變數。如需詳細資訊，請參閱匯出變數，以便其他動作可以使用它。

Visual

不可用。選擇 YAML 以檢視 YAML 指示。

YAML

透過另一個動作參考變數輸出的步驟 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在 CodeCatalyst 動作中，使用下列語法將參考加入至變數：

```
${action-group-name.action-name.variable-name}
```

取代：

- *action-group-name*具有包含輸出變數之動作的動作群組名稱。

> **Note**
>
> *action-group-name*如果沒有動作群組，或變數是由相同動作群組中的動作所產生，您可以省略此選項。

- *####*，其中包含輸出變數的動作名稱。
- *#####*量的名稱。

例如：

```
MySecondAction:
```

```
    Configuration:
      Steps:
        - Run: ${MyFirstAction.TIMESTAMP}
```

如需更多範例，請參閱 範例。如需詳細資訊，請參工作流定義參考閱您的動作。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

引用一個秘密

如需在工作流程定義檔案中參考密碼的指示，請參閱使用秘密。

如需範例，請參閱 範例：參照密碼。

範例

下列範例顯示如何定義和參照工作流程定義檔案中的變數。

範例

- 範例：使用輸入屬性定義變數
- 範例：使用步驟屬性定義變數
- 範例：使用輸出屬性匯出變數
- 範例：參考相同動作中定義的變數
- 範例：參考另一個動作中定義的變數
- 範例：參照密碼

範例：使用輸入屬性定義變數

下列範例說明如何在Inputs區段中定義兩個變數VAR2，以VAR1及如何定義。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
      - Name: VAR1
        Value: "My variable 1"
      - Name: VAR2
```

```
        Value: "My variable 2"
```

範例：使用步驟屬性定義變數

下列範例說明如何明確定義Steps區段中的DATE變數。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: DATE=$(date +%m-%d-%y)
```

範例：使用輸出屬性匯出變數

下列範例說明如何定義兩個變數，以REPOSITORY-URI及TIMESTAMP如何使用Outputs區段匯出這兩個變數。

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: REPOSITORY-URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
    Configuration:
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - REPOSITORY-URI
        - TIMESTAMP
```

範例：參考相同動作中定義的變數

下列範例說明如何在中指定VAR1變數MyBuildAction，然後在相同的動作中使用參考變數$VAR1。

```
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
```

```
      Variables:
        - Name: VAR1
          Value: my-value
    Configuration:
      Steps:
        - Run: $VAR1
```

範例：參考另一個動作中定義的變數

下列範例說明如何在中指定TIMESTAMP變數BuildActionA、使用Outputs屬性匯出變數，然後在BuildActionB使用中參考變數${BuildActionA.TIMESTAMP}。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - TIMESTAMP
  BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: docker build -t my-ecr-repo/image-repo:latest .
        - Run: docker tag my-ecr-repo/image-repo:${BuildActionA.TIMESTAMP}

        # Specifying just '$TIMESTAMP' here will not work
        # because TIMESTAMP is not a variable
        # in the BuildActionB action.
```

範例：參照密碼

下列範例說明如何參照my-password密碼。這my-password是秘密的關鍵。在工作流程定義檔中使用之前，必須先在 CodeCatalyst 主控台的 [Sec ret] 頁面上指定此機密的金鑰和對應的密碼值。如需詳細資訊，請參閱 與秘密一起工作。

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
```

```
    Steps:
      - Run: curl -u LiJuan:${Secrets.my-password} https://example.com
```

使用預定義變量

預先定義的變數是由工作流程自動發出的索引鍵值配對，並可供您在工作流程動作中使用。

您可在任何工作流程動作中使用預先定義的變數

主題

- 預先定義的變數清單
- 引用預定義的變量
- 決定工作流程發出哪些預先定義的變數
- 範例

預先定義的變數清單

請參閱下列各節，以檢視由自動產生的預先定義變數 CodeCatalyst。

> **Note**
>
> 此清單僅包含工作 CodeCatalyst 流程和CodeCatalyst 動作所發出的預先定義變數。如果您正在使用其他類型的動作 (例如「動 GitHub 作」或「 CodeCatalyst 實驗室」動作)，請改為參閱決定工作流程發出哪些預先定義的變數。

列出

- 「BranchName" 和 "CommitId" 變數
- 建立和測試動作變數
- 「Amazon S3 發布」動作變量
- 「AWS CDK引導」動作變量
- 「AWS CDK部署」動作變數
- 「AWS Lambda調用」動作變量
- 「部署AWS CloudFormation堆疊」動作變數
- 「部署到 Amazon ECS」動作變數

- 「部署至 Kubernetes 叢集」動作變數
- 「渲染 Amazon ECS 任務定義」動作變量
- 「動GitHub 作」動作變數

「BranchName" 和 "CommitId" 變數

CodeCatalyst 在工作流程執行時產生 BranchName CommitId "" 和 "" 變數。如需這些變數的相關資訊，請參閱下表。

索引鍵	值
CommitId	代表工作流程執行開始時存放庫狀態的提交 ID。 範例：`example3819261db00a3ab59468c8b` 另請參閱：範例：參考 "CommitId" 預先定義的變數
BranchName	工作流程執行開始所依據的分支名稱。 範例：`main`、`feature/branch` 、`test-LiJuan` 另請參閱：範例：參考 "BranchName" 預先定義的變數

建立和測試動作變數

建置和測試動作不會自動產生變數。

「Amazon S3 發布」動作變量

Amazon S3 發佈動作不會自動產生變數。

「AWS CDK引導」動作變量

AWS CDK啟動程序動作會產生下列預先定義的變數。

索引鍵	值
部署平台	部署平台的名稱。 硬編碼為AWS:CloudFormation .
region	在工作流程執行AWS 區域期間部署AWS CDK啟動程序堆疊的區域代碼。 範例：us-west-2
堆疊識別碼	已部署的啟動程AWS CDK序堆疊的 Amazon 資源名稱 (ARN)。 範例：arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-bootstrap-stack/6aad4380-100a-11ec-a10a-03b8a84d40df
略過部署	值true表示在工作流程執行期間略過AWS CDK啟動程序堆疊的部署。如果自上次部署之後堆疊中沒有變更，則會略過堆疊部署。 只有在其值為時才會產生此變數true。 硬編碼為true.

「AWS CDK部署」動作變數

部AWS CDK署動作會產生下列變數。

索引鍵	值
堆疊識別碼	在工作流程執行期間部署到的AWS CDK應用程式堆疊的 Amazon 資源名稱 (ARN)。 範例：arn:aws:cloudformation:us-west-2:111122223333:stack/co

索引鍵	值
	`decatalyst-cdk-app-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
部署平台	部署平台的名稱。 硬編碼為`AWS:CloudFormation` .
region	在工作流程執行期間部署的區域代碼。AWS 區域 範例：`us-west-2`
略過部署	值`true`表示在工作流程執行期間略過AWS CDK應用程式堆疊的部署。如果自上次部署之後堆疊中沒有變更，則會略過堆疊部署。 只有在其值為時才會產生此變數`true`。 硬編碼為`true`.
AWS CloudFormation變數	除了產生先前列出的變數之外，AWS CDK部署動作也會將CloudFormation輸出變數公開為工作流程變數，以便在後續的工作流程動作中使用。根據預設，動作只會公開找到的前四個 (或更少) CloudFormation 變數。若要判斷顯示哪些項目，請執行一次AWS CDK部署動作，然後查看執行詳細資訊頁面的「變數」索引標籤。如果 [變數] 索引標籤上列出的變數不是您想要的變數，您可以使用 `CfnOutputVariables` YAML 屬性來設定不同的變數。若要取得更多資訊，請參閱〈〉中的CfnOutputVariables性質描述「AWS CDK 部署」動作參考。

「AWS Lambda調用」動作變量

依預設，AWS Lambdainvoke 動作會在 Lambda 回應裝載中的每個頂層索引鍵產生一個變數。

例如，如果響應有效負載如下所示：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 然後動作將生成以下變量。

索引鍵	值
name	Saanvi
位置	西雅圖
department	{"公司」：「Amazon」，「團隊」："AWS"}

> **Note**
>
> 您可以變更使用 `ResponseFilters` YAML 屬性產生的變數。如需詳細資訊，請參閱《ResponseFilters》中的「AWS Lambda 調用」操作引用。

「部署AWS CloudFormation堆疊」動作變數

「部署AWS CloudFormation堆疊」動作會產生下列變數。

索引鍵	值
部署平台	部署平台的名稱。 硬編碼為`AWS:CloudFormation` .
region	在工作流程執行期間部署的區域代碼。AWS 區域

索引鍵	值
	範例：us-west-2
堆疊識別碼	已部署堆疊的 Amazon 資源名稱 (ARN)。 範例：arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cfn-stack/6aad4380-100a-11ec-a10a-03b8a84d40df

「部署到 Amazon ECS」動作變數

「部署到 Amazon ECS」動作會產生下列變數。

索引鍵	值
叢集	在工作流程執行期間部署到的 Amazon ECS 叢集名稱。 範例：codecatalyst-ecs-cluster
部署平台	部署平台的名稱。 硬編碼為AWS:ECS.
服務	在工作流程執行期間部署到的 Amazon ECS 服務名稱。 範例：codecatalyst-ecs-service
task-definition-arn	在工作流程執行期間註冊的任務定義的 Amazon 資源名稱 (ARN)。 範例：arn:aws:ecs:us-west-2:111122223333:task-definition/codecatalyst-task-def:8 上述範例:8中的指示已註冊的修訂版本。

索引鍵	值
部署網址	Amazon ECS 主控台「事件」索引標籤的連結，您可以在其中檢視與工作流程執行相關聯之 Amazon ECS 部署的詳細資訊。 範例：`https://console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/codecatalyst-ecs-cluster/services/codecatalyst-ecs-service/events`
region	在工作流程執行期間部署的區域代碼。AWS 區域 範例：`us-west-2`

「部署至 Kubernetes 叢集」動作變數

「部署至 Kubernetes」叢集動作會產生下列變數。

索引鍵	值
叢集	在工作流程執行期間部署到的 Amazon EKS 叢集的亞馬遜網站資源名稱 (ARN)。 範例：`arn:aws:eks:us-west-2:111122223333:cluster/codecatalyst-eks-cluster`
部署平台	部署平台的名稱。 硬編碼為`AWS:EKS`.
中繼資料	預訂. 與工作流程執行期間部署的叢集相關的 JSON 格式中繼資料。
命名空間	將叢集部署至其中的 Kubernetes 命名空間。

索引鍵	值
	範例：`default`
resources	預訂. 與工作流程執行期間部署的資源相關的 JSON 格式中繼資料。
server	您可以使用管理工具 (例如.) 與叢集通訊的 API 伺服器端點名稱`kubectl`。 如需 API 服務端點的詳細資訊，請參閱 Amazon EKS 叢集端點存取控制 (英文) 中的 Amazon EKS 使用者指南。 範例：`https://random-string .gr7.us-west-2.eks.amazonaws.com`

「渲染 Amazon ECS 任務定義」動作變量

彩現 Amazon ECS 任務定義動作會產生下列變數。

索引鍵	值
任務定義	由渲染 Amazon ECS 任務定義動作更新的任務定義檔案所指定的名稱。名稱會遵循格式 `task-definition- random-string .json`。 範例：`task-definition--259-0a2r7gxlTF5Xr.json`

「動GitHub 作」動作變數

「動GitHub 作」動作不會自動產生變數。

引用預定義的變量

請使用下列指示來參考預先定義的變數。

先決條件

決定您要參考的預先定義變數名稱，例如CommitId。如需詳細資訊，請參閱決定工作流程發出哪些預先定義的變數。

Visual

不可用。選擇 YAML 以檢視 YAML 指示。

YAML

若要參考預先定義的變數 (YAML 編輯器)

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇 編輯 。
6. 選擇 YAML。
7. 在 CodeCatalyst 動作中，使用下列語法新增預先定義的變數參照：

```
${action-group-name.action-name-or-WorkflowSource.variable-name}
```

取代：

- *action-group-name*與操作組的名稱。

> **Note**
>
> *action-group-name*如果沒有動作群組，或變數是由相同動作群組中的動作所產生，您可以省略。

- *action-name-or-WorkflowSource* 使用：

  輸出變數的動作名稱。

  或

  WorkflowSource，如果變數是BranchName或CommitId變數。

- *#####*量的名稱。

例如：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: echo ${MyFirstECSAction.cluster}
```

另一個範例是：

```
MySecondAction:
    Configuration:
      Steps:
        - Run: echo ${WorkflowSource.CommitId}
```

如需更多範例，請參閱 範例。如需詳細資訊，請參工作流定義參考閱您的動作。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。

決定工作流程發出哪些預先定義的變數

您可以透過兩種方式決定工作流程發出哪些預先定義的變數：

- 執行一次工作流程。執行完成後，工作流程所發出的變數會顯示在執行詳細資訊頁面的「變數」(Variables) 標籤上。如需詳細資訊，請參閱檢視工作流程執行狀態與詳細。
- 請諮詢預先定義的變數清單。此參考列出了每個預定義變量的變量名（key）和值。

> **Note**
>
> 中列出了工作流程變數的最大總大小中工作流程的配額 CodeCatalyst。如果總大小超過上限，達到最大值後發生的動作可能會失敗。

範例

下列範例顯示如何參照工作流程定義檔案中預先定義的變數。

範例

- 範例：參考 "CommitId" 預先定義的變數
- 範例：參考 "BranchName" 預先定義的變數

範例：參考 "CommitId" 預先定義的變數

下列範例說明如何在MyBuildAction動作中參照CommitId預先定義的變數。CommitId變數會由自動輸出 CodeCatalyst。

雖然範例顯示在建置動作中使用的變數，但您可以CommitId在任何動作中使用。

```
MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
      #Build Docker image and tag it with a commit ID
        - Run: docker build -t image-repo/my-docker-image:latest .
        - Run: docker tag image-repo/my-docker-image:${WorkflowSource.CommitId}
```

範例：參考 "BranchName" 預先定義的變數

下列範例說明如何在CDKDeploy動作中參照BranchName預先定義的變數。BranchName變數會由自動輸出 CodeCatalyst。

雖然範例顯示在AWS CDK部署動作中使用的變數，但您可以BranchName在任何動作中使用。

```
CDKDeploy:
    Identifier: aws/cdk-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: app-stack-${WorkflowSource.BranchName}
```

工作流定義參考

以下是工作流程定義檔案的參考文件。

工作流程定義檔案是描述您工作流程的 YAML 檔案。檔案儲存在來源儲存庫根目錄的~/.codecatalyst/workflows/資料夾中。檔案的副檔名可以是 .yml 或 .yaml。

若要建立和編輯工作流程定義檔案，您可以使用 vim 之類的編輯器，也可以使用 CodeCatalyst 主控台的視覺化編輯器或 YAML 編輯器。如需詳細資訊，請參閱 使用主 CodeCatalyst 控台的視覺化和 YAML 編輯器。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

主題

工作流程定義檔案的範例

以下是簡單工作流程定義檔案的範例。它包括一些頂層屬性、一個Triggers區段，以及包含兩個動作的Actions區段：Build和Test。如需詳細資訊，請參閱 關於工作流程定義檔。

```
Name: MyWorkflow
SchemaVersion: 1.0
```

```
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest .
  Test:
    Identifier: aws/managed-test@v1
    DependsOn:
      - Build
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
```

語法指南和慣例

本節說明工作流程定義檔案的語法規則，以及本參考文件中使用的命名慣例。

YAML 語法指南

工作流程定義檔案是以 YAML 撰寫，並遵循 YAML 1.1 規格，因此工作流程 YAML 也允許使用該規格中的任何內容。如果您是 YAML 的新手，以下是一些快速準則，以確保您提供了有效的 YAML 代碼。

- 區分大小寫：工作流程定義檔案區分大小寫，因此請務必使用本文件中顯示的大小寫。
- 特殊字元：我們建議在包含下列任何特殊字元的屬性值周圍使用引號或雙引號：{}[]、*、#、、?、|、-、、、、、< =、> !、%、、@:、`和，

  如果您不包含引號，先前列出的特殊字元可能會以非預期的方式解譯。
- 內容名稱：屬性名稱 (與屬性值相對) 僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您不能使用引號或雙引號來啟用屬性名稱中的特殊字元和空格。

不允許：

`'My#Build@action'`

`My#Build@action`

`My Build Action`

允許：

`My-Build-Action_1`

- 逸出代碼：如果您的屬性值包含逸出代碼 (例如\n或\t)，請遵循下列準則：
  - 使用單引號將轉義代碼作為字符串返回。例如`'my string \n my string'`，傳回字串`my string \n my string`。
  - 使用雙引號來解析轉義代碼。例如`"my string \n my new line"`，傳回：

```
my string
my new line
```

- 評論:與#前言評論.

  範例：

```
Name: MyWorkflow
# This is a comment.
SchemaVersion: 1.0
```

- 三重破折號 (**---**)：請勿`---`在 YAML 程式碼中使用。 CodeCatalyst 忽略之後的所有內容`---`。

命名慣例

在本指南中，我們使用術語屬性和部分來指工作流程定義文件中的主要項目。

- 屬性是包含冒號 (**:**) 的任何項目。例如，在下列程式碼片段中，下列所有項目都是屬性：`Name`、`SchemaVersion`、`RunMode`、`Triggers`、`Type`、、和`Branches`。
- 區段是具有子屬性的任何屬性。在下面的代碼片段中，有一個`Triggers`部分。

> **Note**
>
> 在本指南中，「部分」有時被稱為「屬性」，反之亦然，具體取決於上下文。

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
```

頂層屬性

以下是工作流程定義檔案中最上層屬性的參考文件。

```
# Name
Name: workflow-name

# Schema version
SchemaVersion: 1.0

# Run mode
RunMode: QUEUED|SUPERSEDED|PARALLEL

# Compute
Compute:
...

# Triggers
Triggers:
...

# Actions
Actions:
...
```

Name

(必要)

工作流程的名稱。工作流程名稱顯示在工作流程清單中，並在通知和記錄檔中提及。工作流程名稱和工作流程定義檔案名稱可以相符，或者您可以以不同方式命名它們。工作流程名稱不一定是唯一的。工作流程名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用工作流程名稱中的特殊字元和空格。

對應的 UI：視覺化編輯器/工作流程屬性/工作流程名稱

SchemaVersion

(必要)

工作流程定義的結構描述版本。目前唯一有效的值為：`1.0`。

對應的用戶界面：無

RunMode

(選用)

如何 CodeCatalyst 處理多次運行。您可以使用下列其中一個值：

- `QUEUED`— 將多個運行排入佇列並一個接一個地運行。您最多可以在佇列中執行 50 次。
- `SUPERSEDED`— 將多個運行排入佇列並一個接一個地運行。一個佇列只能有一個執行，所以如果兩個執行結束在同一個佇列中，則稍後的執行會取代 (接管) 先前的執行，並取消先前的執行。
- `PARALLEL`— 多次運行同時發生。

如果省略此屬性，預設值為`QUEUED`。

如需詳細資訊，請參閱 設定佇列、已取代和 parallel 執行。

對應的 UI：視覺化編輯器/工作流程屬性/進階/執行模式

Compute

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

如需有關計算的詳細資訊，請參閱使用運算和執行階段環境 Docker 影像。

對應的用戶界面：無

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Compute:
  Type: EC2 | Lambda
  Fleet: fleet-name
  SharedInstance: true | false
```

Type

(Compute/Type)

(如果設定Compute為必要)

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  針對動作執行期間的彈性進行優化
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱關於運算類型。

對應的 UI：視覺化編輯器/工作流程屬性/進階/運算類型

Fleet

(Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範

例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

如需運算叢集的詳細資訊，請參閱關於運算叢集。

對應的 UI：視覺化編輯器/工作流程屬性/進階/運算叢集

SharedInstance

(Compute/SharedInstance)

(選用)

為您的動作指定計算共用功能。透過運算共用，工作流程中的動作會在相同的執行個體上執行 (執行階段環境影像)。您可以使用下列其中一個值：

- TRUE表示執行階段環境影像會在工作流程動作之間共用。
- FALSE表示會針對工作流程中的每個動作啟動並使用個別的執行階段環境影像，因此如果沒有額外的設定，您就無法共用成品和變數等資源。

如需有關計算共用的詳細資訊，請參閱跨動作共用運算。

對應的用戶界面：無

Triggers

(選用)

此工作流程的一個或多個觸發器的序列。如果未指定觸發器，則您必須手動啟動工作流程。

關於觸發條件的詳細資訊，請參閱 使用觸發程序。

對應的 UI：視覺化編輯器/工作流程圖/觸發程序

```
Name: MyWorkflow
SchemaVersion: 1.0
```

```
...
Triggers:
  - Type: PUSH
    Branches:
      - branch-name
    FilesChanged:
      - folder1/file
      - folder2/

  - Type: PULLREQUEST
    Events:
      - OPEN
      - CLOSED
      - REVISION
    Branches:
      - branch-name
    FilesChanged:
      - file1.txt

  - Type: SCHEDULE
    # Run the workflow at 10:15 am (UTC+0) every Saturday
    Expression: "15 10 ? * 7 *"
    Branches:
      - branch-name
```

Type

(Triggers/Type)

(如果設定Triggers為必要)

指定觸發器的類型。您可以使用下列其中一個值：

- 推送 (視覺化編輯器) 或 PUSH (YAML 編輯器)

  推送觸發程序會在將變更推送至來源儲存庫時啟動工作流程執行。工作流程執行將使用您要推送到的分支 (也就是目標分支) 中的檔案。

- 提取請求（可視化編輯器）或PULLREQUEST（YAML 編輯器）

  在來源儲存庫中開啟、更新或關閉提取要求時，提取要求觸發程序會啟動工作流程執行。工作流程執行將使用您要從中提取的分支中的檔案 (也就是來源分支)。

- 排程 (視覺化編輯器) 或 SCHEDULE (YAML 編輯器)

排程觸發程序會根據您指定的 cron 運算式所定義的排程開始工作流程執行。將使用分支的檔案為來源儲存庫中的每個分支啟動個別的工作流程執行。若要限制觸發程序啟動的分支，請使用「分支」欄位 (視覺化編輯器) 或Branches屬性 (YAML 編輯器)。)

設定排程觸發器時，請遵循下列準則：

- 每個工作流程僅使用一個排程觸發器。
- 如果您已在 CodeCatalyst 空間中定義了多個工作流程，我們建議您排程不超過 10 個工作流程以同時啟動。
- 確保在運行之間有足夠的時間配置觸發器的 cron 表達式。如需詳細資訊，請參閱 Expression。

如需範例，請參閱 觸發器的例子。

對應的 UI：視覺化編輯器/工作流程圖/觸發器/觸發器類型

Events

(Triggers/Events)

(如果觸發器設定Type為，則需要PULLREQUEST)

指定將啟動工作流程執行的提取要求事件類型。以下是有效值：

- 創建拉請求（可視化編輯器）或OPEN（YAML 編輯器）

  建立提取請求時，會啟動工作流程執行。
- 拉請求已關閉（可視化編輯器）或CLOSED（YAML 編輯器）

  當提取請求關閉時，會啟動工作流程執行。該CLOSED事件的行為很棘手，並且最好通過一個示例來理解。如需詳細資訊，請參閱範例：具有拉動、分支和「CLOSE」事件的觸發器。
- 新的修訂做了拉請求（可視化編輯器）或REVISION（YAML 編輯器）

  工作流程執行會在建立提取請求的修訂時啟動。建立提取請求時會建立第一個修訂版本。之後，每當有人將新提交推送到提取請求中指定的源分支時，都會創建一個新的修訂版本。如果您將REVISION事件包含在提取要求觸發程序中，您可以省略該OPEN事件，因為REVISION這是的OPEN超集合。

您可以在同一個提取請求觸發程序中指定多個事件。

如需範例，請參閱 觸發器的例子。

對應的 UI：可視化編輯器/工作流程圖/觸發器/提取請求的事件

Branches

(Triggers/Branches)

(選用)

指定觸發程式監視的來源儲存庫中的分支，以便知道何時開始工作流程執行。您可以使用正則表達式模式來定義分支名稱。例如，使用`main.*`來比對以開頭的所有分支`main`。

根據觸發器類型，要指定的分支會有所不同：

- 對於推送觸發器，請指定要推送到的分支，也就是目標分支。每個匹配的分支將使用匹配分支中的文件啟動一個工作流程運行。

  範例: `main.*`, `mainline`
- 對於提取要求觸發程序，請指定您要推送到的分支，也就是目的地分支。每個匹配的分支將使用工作流程定義文件和源文件（而不是匹配的分支）啟動一個工作流程運行。

  範例：`main.*`,`mainline`, `v1\-.*` (符合開頭為的分支`v1-`)
- 對於排程觸發器，請指定包含您要排程執行使用之檔案的分支。每個匹配的分支將使用工作流程定義文件和匹配分支中的源文件啟動一個工作流程運行。

  範例: `main.*`, `version\-1\.0`

> **Note**
>
> 如果您未指定分支，則觸發程序會監視來源儲存庫中的所有分支，並使用下列項目中的工作流程定義檔案和來源檔案啟動工作流程執行：
>
> - 您正在推送的分支（用於推送觸發器）。如需詳細資訊，請參閱 範例：簡單的程式碼推送觸發器。
> - 您從中提取的分支（用於拉取請求觸發器）。如需詳細資訊，請參閱 範例：簡單的拉取要求觸發程序。
> - 所有分支 (用於排程觸發器)。來源儲存庫中的每個分支都會啟動一個工作流程執行。如需詳細資訊，請參閱 範例：一個簡單的排程觸發。

如需有關分支和觸發程序的更多資訊，請參閱分支時的觸發考量。

如需更多範例，請參閱觸發器的例子。

對應的 UI：可視化編輯器/工作流程圖/觸發器/分支

FilesChanged

(Triggers/FilesChanged)

(如果觸發器設定Type為PUSH、或，則為選擇性PULLREQUEST。 如果觸發器設定Type為，則不支援SCHEDULE。)

指定觸發程式監視的來源儲存庫中的檔案或資料夾，以便知道何時開始執行工作流程。您可以使用規則運算式來比對檔案名稱或路徑。

如需範例，請參閱 觸發器的例子。

對應的 UI：視覺化編輯器/工作流程圖/觸發器/檔案已變更

Expression

(Triggers/Expression)

(如果觸發器設定Type為，則需要SCHEDULE)

指定描述何時執行排程工作流程執行的 cron 運算式。

中的 Cron 運算式 CodeCatalyst 使用下列六個欄位語法，其中每個欄位都以空格分隔：

####days-of-month#days-of-week#

cron 表達式的例子

分鐘	小時	月份中的天數	月	星期中的天數	年	意義
0	0	?	*	MON-FRI	*	在每個星期一至星期五的午夜 (UTC+0) 執行工作流程。

分鐘	小時	月份中的天數	月	星期中的天數	年	意義
0	2	*	*	?	*	每天在凌晨 2:00 (UTC +0) 執行工作流程。
15	22	*	*	?	*	每天在晚上 10:15 (UTC+0) 執行工作流程。
0/30	22-2	?	*	SAT-SUN	*	在開始日的晚上 10:00 至翌日凌晨 2:00 之間，每 30 分鐘執行一個工作流程 (UTC+0)。
45	13	L	*	?	2023-2027	在 2023 年到 2027 年 (含 2027 年) 之間每月最後一天的下午 1:45 (UTC +0) 執行工作流程。

在中指定 cron 運算式時 CodeCatalyst，請確定遵循下列準則：

- 為每個SCHEDULE觸發器指定一個 cron 運算式。
- 在 YAML 編輯器中以雙引號 ('') 括住 cron 運算式。

- 以國際標準時間 (UTC) 為單位指定時間。不支援其他時區。
- 設定執行間隔至少 30 分鐘。不支援較快的節奏。
- 指定*days-of-month*或*days-of-week*欄位，但不能同時指定兩者。如果您在其中一個欄位中指定值或星號 (*)，則必須在另一個欄位中使用問號 (?)。星號表示「全部」，問號表示「任何」。

如需 Cron 運算式的更多範例和萬用字元 (例如?、和L) 的相關資訊*，請參閱 Amazon EventBridge 使用者指南中的 Cron 運算式參考資料。Cron 表達式 EventBridge 和 CodeCatalyst 工作方式完全相同。

如需排程觸發器的範例，請參閱觸發器的例子。

對應的 UI：視覺化編輯器/工作流程圖/觸發器/排程

動作

此工作流程的一個或多個動作的序列。 CodeCatalyst 支援數種動作類型，例如建置和測試動作，這些動作提供不同類型的功能。每個動作類型都有：

- 指示動作唯一、硬式編碼 ID 的`Identifier`屬性。例如，`aws/build@v1`識別建置動作。
- 包含動作特定屬性的`Configuration`區段。

如需有關每個動作類型的詳細資訊，請參閱其參考文件集，可在工作流定義參考.

以下是工作流程定義檔案中動作和動作群組的 YAML 參考資料。

如需動作的詳細資訊，請參閱使用動作。

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Actions:
  action-name:
    Identifier: action-identifier
    Configuration:
    ...
  #Action groups
  action-group-name:
    Actions:
      ...
```

動作名稱

（ Actions/#### ）

(必要)

將####替換為您要提供動作的名稱。動作名稱在工作流程中必須是唯一的，且只能包含英數字元、連字號和底線。如需語法規則的詳細資訊，請參閱YAML 語法指南。

如需有關動作 (包括限制) 命名做法的詳細資訊，請參閱動作名稱。

對應的使用者介面：視覺化編輯器/動作**##/####**標籤/動作名稱或動作顯示名稱

action-group-name

(Actions/*action-group-name*)

(選用)

動作群組包含一或多個動作。將動作分組到動作群組中，可協助您保持工作流程井然有序，也可讓您配置不同動作群組之間的相依性。

以您要為動作群組指定的名稱取*action-group-name*代。動作群組名稱在工作流程中必須是唯一的，且只能包含英數字元、連字號和底線。如需語法規則的詳細資訊，請參閱YAML 語法指南。

如需有關動作群組的詳細資訊，請參閱將動作分組到動作群組中。

對應的用戶界面：無

建立和測試動作參考

以下是建置和測試動作的動作定義 YAML 參考。兩個動作有一個參考，因為它們的 YAML 屬性非常相似。

在下列程式碼中選擇 YAML 屬性，以查看是否有描述。

> **Note**
>
> 後續的大多數 YAML 屬性在視覺化編輯器中都有對應的 UI 元素。若要查詢使用者介面元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.
```

```
Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier: aws/build@v1 | aws/managed-test@v1
    DependsOn:
      - dependent-action-name-1
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Caching:
      FileCaching:
        key-name-1:
          Path: file1.txt
          RestoreKeys:
            - restore-key-1
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Outputs:
      Artifacts:
        - Name: output-artifact-1
          Files:
            - build-output/artifact-1.jar
            - "build-output/build*"
        - Name: output-artifact-2
          Files:
```

```
        - build-output/artifact-2.1.jar
        - build-output/artifact-2.2.jar
    Variables:
      - variable-name-1
      - variable-name-2
    AutoDiscoverReports:
      Enabled: true | false
      ReportNamePrefix: AutoDiscovered
      IncludePaths:
        - "**/*"
      ExcludePaths:
        - node_modules/cdk/junit.xml
      SuccessCriteria:
        PassRate: percent
        LineCoverage: percent
        BranchCoverage: percent
        Vulnerabilities:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisBug:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisSecurity:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisQuality:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisFinding:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
    Reports:
      report-name-1:
        Format: format
        IncludePaths:
          - "*.xml"
        ExcludePaths:
          - report2.xml
          - report3.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
```

```
            Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
            Number: whole-number
          StaticAnalysisBug:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisSecurity:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisQuality:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisFinding:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
  Configuration:
    Container:
      Registry: registry
      Image: image
    Steps:
      - Run: "step 1"
      - Run: "step 2"
    Packages:
      NpmConfiguration:
        PackageRegistries:
          - PackagesRepository: package-repository
            Scopes:
              - "@scope"
```

動作名稱

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

對應的 UI：組態索引標籤/動作名稱

Identifier

(*####*/) Identifier

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

用aws/build@v1於構建操作。

用aws/managed-test@v1於測試動作。

##############/####/aws/## @v1 | aws/ ##### @v1 ##

DependsOn

（ *####*/ ）DependsOn

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

（ *####*/ ）Compute

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

（ ***######***/Compute/ ）

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱關於運算類型。

對應的 UI：組態索引標籤/運算類型

Fleet

(*#### /Compute/* ***Fleet***)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如`Fleet`果省略，預設值為`Linux.x86-64.Large`。

對應的 UI：配置選項卡/計算機群

Timeout

(*####*/) Timeout

(選用)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Environment

(*####*/) Environment

(選用)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/環境

Name

(*#### /Environment/* ***Name***)

(選用)

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/環境名稱

Connections

(*#### /Environment/* ***Connections***)

(選用)

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線Environment。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/

Name

(*#### /Environment/Connections/* ***Name***)

(選用)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/

Role

(*#### /Environment/Connections/* ***Role***)

(選用)

指定此動作用於在 Amazon S3 和 Amazon ECR 等 AWS 服務中存取和操作的 IAM 角色名稱。確保此角色已添加到您的帳戶連接中。若要將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 如果角色具有足夠的權限，您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

> **Warning**
>
> 將權限限制為組建和測試動作所需的權限。使用具有更廣泛權限的角色可能會造成安全風險。

對應的 UI：配置選項卡/

Caching

（*####*/）Caching

(選用)

您可以在其中指定快取以儲存磁碟上的檔案，並在後續工作流程執行中從該快取還原它們的區段。

如需檔案快取的詳細資訊，請參閱使用檔案快取。

對應的用戶界面：配置選項卡/文件緩存- 可選

FileCaching

（*#### /Caching/* ***FileCaching***）

(選用)

指定一系列快取記憶體組態的段落。

對應的用戶界面：配置選項卡/文件緩存-可選/添加緩存

鍵名 -1

####### -1/Caching/FileCaching/#

(選用)

指定主要快取屬性名稱的名稱。快取屬性名稱在工作流程中必須是唯一的。每個動作最多可以有五個項目`FileCaching`。

對應的 UI：配置選項卡/文件緩存-可選/添加緩存/密鑰

Path

(*####/Caching/FileCaching/## -1/*) Path

(選用)

指定快取的相關路徑。

對應的 UI：配置選項卡/文件緩存-可選/添加緩存/路徑

RestoreKeys

(*####/Caching/FileCaching/## -1/*) RestoreKeys

(選用)

指定找不到主要快取屬性時要用作後援的還原金鑰。還原金鑰名稱在您的工作流程中必須是唯一的。每個快取中最多可以有五個項目`RestoreKeys`。

對應的用戶界面：配置選項卡/文件緩存-可選/添加緩存/恢復鍵-可選

Inputs

(*####/*) Inputs

(選用)

本Inputs節定義工作流程執行期間動作所需的資料。

> **Note**
>
> 每個建置動作或測試動作最多允許四個輸入 (一個來源和三個成品)。變數不會計入此總數。

如果您需要引用駐留在不同輸入中的文件（例如源和工件），則源輸入是主輸入，並且工件是輔助輸入。輔助輸入中的文件的引用採用特殊前綴來將它們從主輸入中剔除。如需詳細資訊，請參閱 範例：參考多個成品中的檔案。

對應的 UI：輸入索引標籤

Sources

（*#### /Inputs/* ***Sources*** ）

(選用)

指定代表動作所需之來源儲存庫的標籤。目前唯一支援的標籤是WorkflowSource，它代表儲存工作流程定義檔案的來源儲存庫。

如果您省略來源，則必須在下指定至少一個輸入成品*action-name*/Inputs/Artifacts。

如需來源的詳細資訊，請參閱 使用來源。

對應的用戶界面：無

Artifacts - input

（*#### /Inputs/* ***Artifacts*** ）

(選用)

指定您要提供作為此動作輸入的先前動作的人工因素。在先前動作中，必須將這些人工因素定義為輸出人工因素。

如果您未指定任何輸入人工因素，則必須在下指定至少一個來源儲存庫*action-name*/Inputs/Sources。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

> **Note**
>
> 如果無法使用「成品-選用」下拉式清單 (視覺化編輯器)，或者在驗證 YAML (YAML 編輯器) 時發生錯誤，可能是因為動作僅支援一個輸入。在此情況下，請嘗試移除來源輸入。

對應的 UI：輸入選項卡/加工品- 可選

Variables - input

(*#### /Inputs/* ***Variables***)

(選用)

指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的用戶界面：輸入選項卡/變量- 可選

Outputs

(*####/*) Outputs

(選用)

定義在工作流程執行期間由動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts - output

(*#### /Outputs/* ***Artifacts***)

(選用)

指定動作所產生的成品名稱。Artifact 名稱在工作流程中必須是唯一的，且僅限於英數字元 (a-z、A-Z、0-9) 和底線 (_)。不允許使用空格、連字號 (-) 和其他特殊字元。您無法使用引號來啟用輸出人工因素名稱中的空格、連字號和其他特殊字元。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/成品

Name

（*#### /Outputs/Artifacts/* ***Name*** ）

（如果包含Artifacts - output，則為必填）

指定動作所產生的成品名稱。Artifact 名稱在工作流程中必須是唯一的，且僅限於英數字元 (a-z、A-Z、0-9) 和底線 (_)。不允許使用空格、連字號 (-) 和其他特殊字元。您無法使用引號來啟用輸出人工因素名稱中的空格、連字號和其他特殊字元。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/人工因素/新輸出/構建工件名稱

Files

（*#### /Outputs/Artifacts/* ***Files*** ）

（如果包含Artifacts - output，則為必填）

指定動作輸出的加工品中 CodeCatalyst 包含的檔案。這些檔案會在工作流程動作執行時由工作流程動作產生，也可在來源存放庫中使用。檔案路徑可以位於來源儲存庫或上一個動作的人工因素中，且相對於來源儲存庫或人工因素根目錄。您可以使用全域模式來指定路徑。範例：

- 若要指定位於組建位置或來源存放庫位置根目錄中的單一檔案，請使用`my-file.jar`。
- 若要在子目錄中指定單一檔案，請使用`directory/my-file.jar`或`directory/subdirectory/my-file.jar`。
- 若要指定所有檔案，請使用`"**/*"`。**glob 模式指示匹配任意數量的子目錄。
- 若要指定名為的目錄中的所有檔案和目錄`directory`，請使用`"directory/**/*"`。**glob 模式指示匹配任意數量的子目錄。
- 若要指定名為的目錄中的所有檔案`directory`，但不指定其任何子目錄中的檔案，請使用`"directory/*"`。

> **Note**
>
> 如果檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

> **Note**
>
> 您可能需要在檔案路徑中新增前置詞，以指出要在其中尋找的成品或來源。如需詳細資訊，請參閱 參考來源儲存庫中的檔案 及 參考人工因素中的檔案。

對應的 UI：輸出選項卡/人造/新輸出/構建生成的文件

Variables - output

(`#### /Outputs/ Variables`)

(選用)

指定您要匯出動作的變數，以便後續動作可使用這些變數。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸出選項卡/變量/添加變量

變量名 -1

`######## -1/Outputs/Variables/#`

(選用)

指定您要匯出動作的變數名稱。此變數必須已在相同動作的`Inputs`或`Steps`區段中定義。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸出選項卡/變量/添加變量/名稱

AutoDiscoverReports

(`#### /Outputs/ AutoDiscoverReports`)

(選用)

定義自動探索功能的組態。

當您啟用自動探索時，會 CodeCatalyst 搜尋所有`Inputs`傳入動作的內容，以及動作本身產生的所有檔案，以尋找測試、程式碼涵蓋範圍和軟體組成分析 (SCA) 報告。針對找到的每個報表，將其

CodeCatalyst 轉換為 CodeCatalyst 報表。CodeCatalyst 報告是完全集成到 CodeCatalyst 服務中的報告，可以通過 CodeCatalyst 控制台查看和操作。

> **Note**
>
> 根據預設，自動探索功能會檢查所有檔案。您可以使用IncludePaths或ExcludePaths性質限制要檢查哪些檔案。

對應的 UI：輸出標籤/報告/自動探索報告

Enabled

（*#### /Outputs/AutoDiscoverReports/* ***Enabled*** ）

(選用)

啟用或停用自動探索功能。

有效值為 `true` 或 `false`。

如`Enabled`果省略，預設值為`true`。

對應的 UI：輸出標籤/報告/自動探索報告

ReportNamePrefix

（*#### /Outputs/AutoDiscoverReports/* ***ReportNamePrefix*** ）

(如果已包含並啟用，則AutoDiscoverReports為必要)

指定在找到的所有報告前面 CodeCatalyst 加上的首碼，以便命名其關聯 CodeCatalyst 的報告。例如，如果您指定的前置詞`AutoDiscovered`，並 CodeCatalyst自動探索兩個測試報告，`TestSuiteOne.xml`然後`TestSuiteTwo.xml`，關聯的 CodeCatalyst 報告將命名為`AutoDiscoveredTestSuiteOne`和。`AutoDiscoveredTestSuiteTwo`

對應的 UI：輸出選項卡/報告/前綴名稱

IncludePaths

（*#### /Outputs/AutoDiscoverReports/* ***IncludePaths*** ）

或

(*####/Outputs/Reports/##-## -1/*) IncludePaths

(如果AutoDiscoverReports已包含並啟用，或如果已包含，則Reports為必要)

指定搜尋原始報告時 CodeCatalyst 包含的檔案和檔案路徑。例如，如果您指定"/test/report/*"，會 CodeCatalyst 搜尋尋找/test/report/*目錄的動作所使用的整個組建映像。找到該目錄時， CodeCatalyst 會在該目錄中尋找報表。

> **Note**
>
> 如果檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

如果省略此性質，預設值為"**/*"，表示搜尋會包括所有路徑上的所有檔案。

> **Note**
>
> 對於手動設定的報告，IncludePaths必須是符合單一檔案的 glob 模式。

對應的介面：

- 輸出標籤/報告/自動探索報告/包含/排除路徑/包含路徑
- ***[##] ####/##/######/#### -1 /##/####/####***

ExcludePaths

(*#### /Outputs/AutoDiscoverReports/* ***ExcludePaths***)

或

(*####/Outputs/Reports/##-## -1/*) ExcludePaths

(選用)

指定搜尋原始報告時 CodeCatalyst 排除的檔案和檔案路徑。例如，如果您指定"/test/my-reports/**/*"， CodeCatalyst 將不會搜尋/test/my-reports/目錄中的檔案。要忽略目錄中的所有文件，請使用 **/* glob 模式。

> **Note**
>
> 如果檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

對應的介面：

- 輸出標籤/報告/自動探索報告/包含/排除路徑/排除路徑
- ***[##] ####/##/######/#### -1 /##/####/####***

SuccessCriteria

(*#### /Outputs/AutoDiscoverReports/* ***SuccessCriteria***)

或

(*####/Outputs/Reports/##-## -1/*) SuccessCriteria

(選用)

指定測試、程式碼涵蓋範圍、軟體組成分析 (SCA) 和靜態分析 (SA) 報告的成功準則。

如需詳細資訊，請參閱 設定報告的成功準則。

對應的 UI：輸出標籤/報告/成功條件

PassRate

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***PassRate***)

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****PassRate***

(選用)

指定測試報告中必須通過的測試百分比，這些測試百分比才能標記為通過的關聯 CodeCatalyst報告。有效值包括十進位數字。例如：50、60.5。合格率標準僅適用於測試報告。如需測試報告的詳細資訊，請參閱測試報告。

對應的用戶界面：輸出標籤/報告/成功標準/合格率

LineCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***LineCoverage***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****LineCoverage***

(選用)

指定程式碼涵蓋範圍報告中必須涵蓋的行百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。明細行涵蓋範圍條件僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

對應的用戶界面：輸出選項卡/報告/成功條件/行覆蓋範圍

BranchCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***BranchCoverage***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****BranchCoverage***

(選用)

指定程式碼涵蓋範圍報告中必須涵蓋的分支百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。分支涵蓋範圍標準僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

對應的用戶界面：輸出選項卡/報告/成功標準/分支覆蓋範圍

Vulnerabilities

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***Vulnerabilities***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****Vulnerabilities***

(選用)

針對要標記為通過的相關 CodeCatalyst 報告，指定 SCA 報告中允許的弱點數目上限和嚴重性。若要指定弱點，您必須指定：

- 您要包含在計數中的弱點的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL漏洞將被統計。
- 您要允許之指定嚴重性的弱點數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

弱點準則僅適用於 SCA 報告。如需有關 SCA 報告的詳細資訊，請參閱軟件成分分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

對應的使用者介面：輸出索引標籤/報告/成功準則/弱點

StaticAnalysisBug

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***StaticAnalysisBug*** ）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****StaticAnalysisBug***

(選用)

針對要標記為通過的相關 CodeCatalyst 報告，指定 SA 報告中允許的最大錯誤數目和嚴重性。若要指定錯誤，您必須指定：

- 您要包含在計數中的錯誤的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL錯誤將被計算。
- 您要允許的指定嚴重性的最大錯誤數目。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

錯誤條件僅適用於 PyLint 和 eSlint SA 報告。如需 SA 報告的詳細資訊，請參閱靜態分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

對應的用戶界面：輸出選項卡/報告/成功標準/錯誤

StaticAnalysisSecurity

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisSecurity***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**StaticAnalysisSecurity***

(選用)

指定 SA 報告中允許的相關 CodeCatalyst 報告標示為通過的安全性弱點數目上限和嚴重性。若要指定安全性弱點，您必須指定：

- 您要包含在計數中的安全性弱點的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL安全漏洞將被結算。
- 您要允許之指定嚴重性的最大安全性弱點數目。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

安全性弱點條件僅適用於 PyLint 和 eSlint SA 報告。如需 SA 報告的詳細資訊，請參閱靜態分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

對應的 UI：輸出標籤/報告/成功準則/安全漏洞

StaticAnalysisQuality

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisQuality***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**StaticAnalysisQuality***

(選用)

指定 SA 報告中允許將相關 CodeCatalyst 報告標示為通過的品質問題的最大數目和嚴重性。要指定質量問題，您必須指定：

- 您要包含在計數中的品質問題的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL質量問題將被結算。
- 您要允許之指定嚴重性的品質問題數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

「質量問題」條件僅適用于「eSlint SA」報表。 PyLint 如需 SA 報告的詳細資訊，請參閱靜態分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

對應的用戶界面：輸出選項卡/報告/成功準則/質量問題

StaticAnalysisFinding

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/**StaticAnalysisFinding***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**StaticAnalysisFinding***

(選用)

針對要標記為通過的關聯 CodeCatalyst 報告，指定 SA 報告中允許的發現項目數目上限和嚴重性。若要指定發現項目，您必須指定：

- 您要包含在計數中之發現項目的最小嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL調查結果將被統計。
- 您要允許之指定嚴重性的發現項目數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

發現項目僅套用至 SARIF SA 報表。如需 SA 報告的詳細資訊，請參閱靜態分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

對應的 UI：輸出標籤/報告/成功準則/發現項目

Reports

（*#### /Outputs/* ***Reports*** ）

(選用)

指定測試報告之組態的段落。

對應的 UI：輸出選項卡/報告

報告名稱 -1

（動作名稱報***#-## -1***/Outputs/Reports/ ）

（如果包含Reports，則為必填）

您要提供給將從原始 CodeCatalyst 報表產生之報表的名稱。

對應的 UI：輸出選項卡/報告/手動配置報告/報告名稱

Format

（*####/Outputs/Reports/##-## -1/* ）Format

（如果包含Reports，則為必填）

指定報告所使用的檔案格式。可能的值如下。

- 對於測試報告：
  - 對於「黃瓜 JSON」，請指定「黃瓜」(視覺化編輯器) 或 CUCUMBERJSON (YAML 編輯器)。
  - 對於 JUnit XML，請指定 JUnit（可視化編輯器）或JUNITXML（YAML 編輯器）。
  - 對於 NUnit XML，請指定 NUnit (視覺化編輯器) 或 NUNITXML (YAML 編輯器)。
  - 對於 NUnit 3 XML，請指定 nUnit3 (視覺化編輯器) 或 NUNIT3XML (YAML 編輯器)。
  - 若為視覺工作室 TRX，請指定視覺工作室 TRX (視覺化編輯器) 或 VISUALSTUDIOTRX (YAML 編輯器)。
  - 對於 TestNG 的 XML，請指定 TestNG（可視化編輯器）或TESTNGXML（YAML 編輯器）。

- 對於代碼覆蓋率報告：
  - 針對四葉草 XML，請指定四葉草 (視覺化編輯器) 或 CLOVERXML (YAML 編輯器)。
  - 對於 XML 編輯器，指定編輯器（可視化編輯器）或**COBERTURAXML**（YAML 編輯器）。
  - 對於 JaCoCo XML，請指定 JaCoCo(視覺化編輯器) 或 JACOCOXML (YAML 編輯器)。
  - 對於簡單生成的 SimpleCov JSON，而不是簡單的 co v-json，指定簡單的編輯器（可視化編輯器）或（YAML 編輯器）。SIMPLECOV
- 對於軟件成分分析（SCA）報告：
  - 針對 SARIF，請指定 SARIF (視覺化編輯器) 或 SARIFSCA (YAML 編輯器)。

UI: ####/##/######/##/####/#### -1/#########

Configuration

（*####*/）Configuration

(必要) 您可以在其中定義動作的組態特性的區段。

對應的 UI：組態索引標籤

Container

（*#### /Configuration/* ***Container***）

(選用)

指定動作用來完成其處理的 Docker 映像或容器。您可以指定隨附的其中一個作用中影像 CodeCatalyst，也可以使用自己的影像。如果您選擇使用自己的映像檔，它可以存放在 Amazon ECR、碼頭集線器或其他登錄中。如果您未指定 Docker 影像，動作會使用其中一個作用中的影像進行處理。若要取得有關預設使用哪個作用中影像的資訊，請參閱〈〉作用中影像。

如需有關指定您自己的 Docker 映像檔的詳細資訊，請參閱將自訂執行階段環境 Docker 影像指派給動作。

對應的用戶界面：運行時環境 Docker 映像- 可選

Registry

（*#### /Configuration/Container/* ***Registry***）

（如果包含Container，則為必填）

指定儲存映像檔的登錄。有效值包含：

- CODECATALYST(YAML 編輯器)

  該映像存儲在 CodeCatalyst 註冊表中。

- 碼頭集線器（可視化編輯器）或DockerHub（YAML 編輯器）

  該映像存儲在碼頭集線器映像註冊表中。

- 其他註冊表（可視化編輯器）或Other（YAML 編輯器）

  影像儲存在自訂映像登錄中。任何公開可用的註冊表都可以使用。

- Amazon 彈性容器註冊表（可視化編輯器）或ECR（YAML 編輯器）

  映像檔儲存在 Amazon 彈性容器登錄映像儲存庫中。若要在 Amazon ECR 儲存庫中使用映像檔，此動作需要存取 Amazon ECR。若要啟用此存取權，您必須建立包含下列許可和自訂信任政策的 IAM 角色。您可以視需要修改現有角色以包含權限和原則。）

  IAM 角色必須在其角色政策中包含以下許可：

  - ecr:BatchCheckLayerAvailability
  - ecr:BatchGetImage
  - ecr:GetAuthorizationToken
  - ecr:GetDownloadUrlForLayer

  IAM 角色必須包含下列自訂信任政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

如需建立 IAM 角色的詳細資訊，請參閱 IAM 使用者指南中的使用自訂信任政策 (主控台) 建立角色。

建立角色之後，您必須透過環境將其指派給動作。如需詳細資訊，請參閱 將環境、帳戶連線和 IAM 角色與工作流程動作建立關聯。

對應的用戶界面：Amazon 彈性容器註冊表，碼頭集線器和其他註冊表選項

Image

（*#### /Configuration/Container/* ***Image*** ）

（如果包含Container，則為必填）

請指定下列其中一項：

- 如果您使用CODECATALYST登錄，請將映像設定為下列其中一個作用中的影像：
  - CodeCatalystLinux_x86_64:2024_03
  - CodeCatalystLinux_x86_64:2022_11
  - CodeCatalystLinux_Arm64:2024_03
  - CodeCatalystLinux_Arm64:2022_11
  - CodeCatalystLinuxLambda_x86_64:2024_03
  - CodeCatalystLinuxLambda_x86_64:2022_11
  - CodeCatalystLinuxLambda_Arm64:2024_03
  - CodeCatalystLinuxLambda_Arm64:2022_11
  - CodeCatalystWindows_x86_64:2022_11
- 如果您使用的是 Docker Hub 註冊表，請將映像設置為 Docker Hub 映像名稱和可選標記。

  範例：postgres:latest
- 如果您使用的是 Amazon ECR 註冊表，請將映像設置為 Amazon ECR 註冊表 URI。

  範例：111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo
- 如果您使用自訂登錄，請將映像設定為自訂登錄所預期的值。

對應的用戶界面：運行時環境 docker 映像（如果註冊表是**CODECATALYST**），碼頭集線器映像（如果註冊表是 Docker Hub），ECR 映像 URL（如果註冊表是 Amazon 彈性容器註冊表）和圖像 URL（如果註冊表是其他註冊表）。

Steps

（*#### /Configuration/* ***Steps*** ）

(必要)

指定您要在動作期間執行的 shell 命令，以安裝、設定和執行建置工具。

以下是如何構建 npm 項目的示例：

```
Steps:
  - Run: npm install
  - Run: npm run build
```

以下是如何指定檔案路徑的範例：

```
Steps:
  - Run: cd $ACTION_BUILD_SOURCE_PATH_WorkflowSource/app  && cat file2.txt
  - Run: cd $ACTION_BUILD_SOURCE_PATH_MyBuildArtifact/build-output/  && cat file.txt
```

若要取得有關指定檔案路徑的更多資訊，請參閱參考來源儲存庫中的檔案和參考人工因素中的檔案。

對應的 UI：配置選項卡/命令介面命令

Packages

（*#### /Configuration/* ***Packages*** ）

(選用)

您可以在此段落指定動作用來解析相依性的套裝程式儲存區域。套件可讓您安全地儲存和共用用於應用程式開發的軟體套件。

如需封裝的詳細資訊，請參閱中的套件 CodeCatalyst。

對應的用戶界面：配置選項卡/包

NpmConfiguration

（*#### /Configuration/Packages/* ***NpmConfiguration***）

（如果包含Packages，則為必填）

定義 npm 套件格式設定的區段。此組態由工作流程執行期間的動作使用。

如需 npm 套件組態的詳細資訊，請參閱使用 NPM。

對應的用戶界面：配置選項卡/軟件包/添加配置/npm

PackageRegistries

（*#### /Configuration/Packages/NpmConfiguration/* ***PackageRegistries***）

（如果包含Packages，則為必填）

您可以在此段落定義套裝程式儲存區域序列的組態特性。

對應的用戶界面：配置選項卡/軟件包/添加配置/npm/ 添加軟件包庫

PackagesRepository

（*#### /Configuration/Packages/NpmConfiguration/PackageRegistries/* ***PackagesRepository***）

（如果包含Packages，則為必填）

指定您要動作使用的 CodeCatalyst 套裝程式儲存區域名稱。

如果您指定多個預設儲存庫，最後一個存放庫將優先處理。

如需套裝程式儲存庫的詳細資訊，請參閱Package 儲存庫。

對應的使用者介面：組態索引標籤/套 Package/新增組態 /NPM /新增套件儲存庫/套件儲存庫

Scopes

（*#### /Configuration/Packages/NpmConfiguration/PackageRegistries/* ***Scopes***）

(選用)

指定您要在套件登錄中定義的範圍序列。定義範圍時，指定的套裝程式儲存區域會設定為所有列出範圍的登錄。如果通過 npm 客戶端請求具有範圍的包，它將使用該存儲庫而不是默認的。每個範圍名稱必須加上前綴「@」。

如果您包含覆蓋範圍，則最後一個存儲庫將優先考慮。

如果省Scopes略，則會將指定的套裝程式儲存區域設定為動作所使用之所有套裝程式的預設登錄。

如需有關範圍的詳細資訊，請參閱Package 命名空間和範圍封裝。

對應的用戶界面：配置選項卡/軟件包/添加配置/NPM /添加包存儲庫/範圍-可選

「Amazon S3 發布」動作參考

以下是 Amazon S3 發佈動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱添加「Amazon S3 發布」操作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢使用者介面元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  S3Publish_nn:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      Sources:
        - source-name-1
```

```
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      SourcePath: my/source
      DestinationBucketName: s3-bucket-name
      TargetPath: my/target
```

S3Publish

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：S3Publish_nn。

對應的 UI：組態索引標籤/動作名稱

Identifier

(*S3Publish*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/s3-publish@v1。

對應的使用者介面：工作流程圖/S3Publish_nn/ aws/S3-發佈 @v1 標籤

DependsOn

(*S3Publish*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*S3Publish*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*S3Publish*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  針對動作執行期間的彈性進行優化
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的 UI：組態索引標籤/運算類型

Fleet

(*S3Publish*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如`Fleet`果省略，預設值為`Linux.x86-64.Large`。

對應的 UI：組態索引標籤/運算叢集

Timeout

(*S3Publish*/Timeout)

(必要)

指定動作在 CodeCatalyst 結束動作之前可執行的時間長度 (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Inputs

(*S3Publish*/Inputs)

(選用)

本`Inputs`節定義工作流程執行期間所`S3Publish`需的資料。

> **Note**
>
> 每個AWS CDK 部署動作最多允許四個輸入 (一個來源和三個成品)。變數不會計入此總數。

如果您需要引用駐留在不同輸入中的文件（例如源和工件），則源輸入是主輸入，而工件是輔助輸入。輔助輸入中的文件的引用採用特殊前綴，以將它們從主輸入中刪除。如需詳細資訊，請參閱 範例：參考多個成品中的檔案。

對應的 UI：輸入索引標籤

Sources

(*S3Publish*/Inputs/Sources)

(如果您要發佈到 Amazon S3 的檔案存放在來源儲存庫中，則需要)

如果您要發佈到 Amazon S3 的檔案存放在來源儲存庫中，請指定該來源儲存庫的標籤。目前，唯一支援的標籤是`WorkflowSource`。

如果您要發佈到 Amazon S3 的檔案不包含在來源儲存庫中，則它們必須位於由另一個動作產生的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的用戶界面：輸入選項卡/源- 可選

Artifacts - input

(*S3Publish*/Inputs/Artifacts)

(如果您要發佈到 Amazon S3 的檔案存放在先前動作的輸出成品中，則需要)

如果您要發佈到 Amazon S3 的檔案包含在先前動作所產生的成品中，請在此處指定該成品。如果您的檔案不包含在成品中，則它們必須位於來源儲存庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：組態索引標籤/成品- 選用

Variables - input

(*S3Publish*/Inputs/Variables)

(選用)

指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的用戶界面：輸入選項卡/變量- 可選

Environment

(*S3Publish*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：組態索引標籤/「環境/連線/角色」/「環境」

Name

(*S3Publish*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：組態索引標籤/「環境/連線/角色」/「環境」

Connections

(*S3Publish*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

對應的 UI：組態索引標籤/「環境/連線/角色」/「連線」

Name

(*S3Publish*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：組態索引標籤/「環境/連線/角色」/「連線」

Role

(*S3Publish*/Environment/Connections/Role)

(必要)

指定 Amazon S3 發佈動作用來存取檔案 AWS 並將檔案複製到 Amazon S3 的 IAM 角色名稱。請確定此角色包括：

- 下列權限原則：

> **Warning**
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

- 下列自訂信任原則：

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

```
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色與您的帳戶連線相關聯。若要進一步了解如何將 IAM 角色與帳戶連線建立關聯，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/連線/角色」/角色

Configuration

(*S3Publish*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

SourcePath

(*S3Publish*/Configuration/SourcePath)

(必要)

指定要發佈到 Amazon S3 之目錄或檔案的名稱和路徑。目錄或檔案可以位於來源儲存庫或上一個動作的人工因素中，且相對於來源儲存庫或人工因素根目錄。

範例：

指定會`./myFolder/`將的內容複製`/myFolder`到 Amazon S3，並保留基礎目錄結構。

只指定 Amazon S3 `myfile.txt` 的`./myFolder/myfile.txt`副本。（目錄結構被刪除。）

您不能使用萬用字元。

> **Note**
>
> 您可能需要在目錄或檔案路徑中新增前置詞，以指出要在其中尋找的成品或來源。如需詳細資訊，請參閱 參考來源儲存庫中的檔案 及 參考人工因素中的檔案。

對應的 UI：配置選項卡/源路徑

DestinationBucketName

(*S3Publish*/Configuration/DestinationBucketName)

(必要)

指定您要在其中發佈檔案的 Amazon S3 儲存貯體的名稱。

對應的 UI：配置選項卡/目標存儲桶- 可選

TargetPath

(*S3Publish*/Configuration/TargetPath)

(選用)

在 Amazon S3 中指定您要發佈檔案的目錄名稱和路徑。如果目錄不存在，則會建立該目錄。目錄路徑不得包含值區名稱。

範例：

```
myS3Folder
```

```
./myS3Folder/myS3Subfolder
```

對應的 UI：配置選項卡/目標目錄- 可選

「AWS CDK引導」動作參考

以下是啟動AWS CDK程序動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱添加「AWS CDK 引導」操作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKBootstrapAction_nn:
    Identifier: aws/cdk-bootstrap@v1
    DependsOn:
      - action-name
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_bootstrap_artifacts
```

```
        Files:
          - "cdk.out/**/*"
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      Region: us-west-2
      CdkCliVersion: version
```

CDKBootstrapAction

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：CDKBootstrapAction_nn。

對應的 UI：組態索引標籤/動作顯示名稱

Identifier

(*CDKBootstrapAction*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/cdk-bootstrap@v1。

對應的使用者介面：工作流程圖/CDKBootstrapAction_nn/ aws/cd-啟動程式 @v1 標籤

DependsOn

(*CDKBootstrapAction*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*CDKBootstrapAction*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*CDKBootstrapAction*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  優化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的使用者介面：組態索引標籤/進階-選用/運算類型

Fleet

(*CDKBootstrapAction*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範

例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

對應的使用者介面：組態索引標籤/進階-選用/運算叢集

Timeout

(*CDKBootstrapAction*/Timeout)

(必要)

指定動作在 CodeCatalyst 結束動作之前可執行的時間長度 (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Inputs

(*CDKBootstrapAction*/Inputs)

(選用)

Inputs本節定義AWS CDK啟動程序動作在工作流程執行期間所需的資料。

對應的 UI：輸入索引標籤

> **Note**
>
> 每個AWS CDK啟動程序動作只允許一個輸入 (來源或成品)。

Sources

(*CDKBootstrapAction*/Inputs/Sources)

（如果您的AWS CDK應用程序存儲在源存儲庫中，則需要）

如果您的AWS CDK應用程序存儲在源存儲庫中，請指定該源存儲庫的標籤。在啟動AWS CDK引導過程之前，引導操作會在此存儲庫中合成應用程序。目前，唯一支援的存放庫標籤為`WorkflowSource`。

如果您的AWS CDK應用程序不包含在源存儲庫中，則它必須位於由另一個操作生成的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的 UI：輸入選項卡/源- 可選

Artifacts - input

(*CDKBootstrapAction*/Inputs/Artifacts)

（如果您的AWS CDK應用程序存儲在上一個操作的輸出成品中，則需要）

如果您的AWS CDK應用程序包含在上一個操作生成的成品中，請在此處指定該成品。在啟動AWS CDK引導過程之前，引導操作會將指定成品中的應用程序合成為 CloudFormation 模板。如果您的AWS CDK應用程序不包含在成品中，則它必須位於源存儲庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸入選項卡/加工品- 可選

Outputs

(*CDKBootstrapAction*/Outputs)

(選用)

定義工作流程執行期間動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts - output

(*CDKBootstrapAction*/Outputs/Artifacts)

(選用)

指定動作產生的人工因素。您可以參考這些成品做為其他動作中的輸入。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/成品

Name

(*CDKBootstrapAction*/Outputs/Artifacts/Name)

（如果包含Artifacts - output，則為必填）

指定成品的名稱，該成品將包含由AWS CDK啟動程序動作在執行時期合成的AWS CloudFormation範本。預設值為 cdk_bootstrap_artifacts。如果您未指定成品，則動作會合成該範本，但不會將其儲存在成品中。請考慮將合成的範本儲存在成品中，以保留其記錄，以供測試或疑難排解之用。

對應的使用者介面：輸出索引標籤/人工因素/新增人工因素名稱

Files

(*CDKBootstrapAction*/Outputs/Artifacts/Files)

（如果包含Artifacts - output，則為必填）

指定要包含在人工因素中的檔案。您必須指"cdk.out/**/*"定包含AWS CDK應用程序的綜合AWS CloudFormation模板。

> **Note**
>
> cdk.out是儲存合成檔案的預設目錄。如果您在cdk.json檔案中指定了輸出目錄以外cdk.out的目錄，請在此處指定該目錄，而不是cdk.out。

對應的 UI：輸出選項卡/人造物/添加人造物/構建生成的文件

Environment

(*CDKBootstrapAction*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*CDKBootstrapAction*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

(*CDKBootstrapAction*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Name

(*CDKBootstrapAction*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Role

(*CDKBootstrapAction*/Environment/Connections/Role)

(必要)

指定AWS CDK啟動程序動作用來存取AWS和新增啟動程序堆疊的 IAM 角色名稱。請確定此角色包含下列原則：

> **Note**
>
> 下列權限原則中顯示的權限是`cdk bootstrap`指令執行其啟動載入所需的權限。如果更改其引導命令，這些權限可能會更改。AWS CDK

> **Warning**
>
> 僅將此角色與AWS CDK啟動程序動作搭配使用。這是非常寬鬆的，並且將其與其他操作一起使用可能會造成安全風險。

- 下列權限原則：

  > **Warning**
  >
  > 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "ssm:GetParameterHistory",
                "ecr:PutImageScanningConfiguration",
                "cloudformation:*",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "ssm:GetParameters",
                "iam:PutRolePolicy",
                "ssm:GetParameter",
                "ssm:DeleteParameters",
                "ecr:DeleteRepository",
```

```
                "ssm:PutParameter",
                "ssm:DeleteParameter",
                "iam:PassRole",
                "ecr:SetRepositoryPolicy",
                "ssm:GetParametersByPath",
                "ecr:DescribeRepositories",
                "ecr:GetLifecyclePolicy"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:aws-account:parameter/cdk-bootstrap/*",
                "arn:aws:cloudformation:aws-region:aws-account:stack/CDKToolkit/*",
                "arn:aws:ecr:aws-region:aws-account:repository/cdk-*",
                "arn:aws:iam::aws-account:role/cdk-*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:CreateUploadBucket",
                "cloudformation:ListExports",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:SetTypeDefaultVersion",
                "cloudformation:RegisterPublisher",
                "cloudformation:ActivateType",
                "cloudformation:ListTypes",
                "cloudformation:DeactivateType",
                "cloudformation:SetTypeConfiguration",
                "cloudformation:DeregisterType",
                "cloudformation:ListTypeRegistrations",
                "cloudformation:EstimateTemplateCost",
                "cloudformation:DescribeAccountLimits",
                "cloudformation:BatchDescribeTypeConfigurations",
                "cloudformation:CreateStackSet",
                "cloudformation:ListStacks",
                "cloudformation:DescribeType",
                "cloudformation:ListImports",
                "s3:*",
                "cloudformation:PublishType",
                "ecr:CreateRepository",
                "cloudformation:DescribePublisher",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:TestType",
```

```
                "cloudformation:ValidateTemplate",
                "cloudformation:ListTypeVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色時，請在資源策略陳述式中使用下列萬用字元，然後在可用資源名稱之後將策略範圍縮減。
>
> ```
> "Resource": "*"
> ```

- 下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色已新增至您的帳戶連線。若要進一步了解如何將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Configuration

(*CDKBootstrapAction*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

Region

(*CDKBootstrapAction*/Configuration/Region)

(必要)

指定AWS 區域將部署啟動程序堆疊的目標。此區域應與部署應AWS CDK用程式的區域相符。如需區域代碼的清單，請參閱區域端點。

對應的 UI：組態索引標籤/區域

CdkCliVersion

(*CDKBootstrapAction*/Configuration/CdkCliVersion)

(選用)

此屬性適用於AWS CDK部署動作的 1.0.13 版或更新版本，以及啟動程序動作的 1.0.8 版或更新版本AWS CDK。

請指定下列其中一項：

- 您希望此動作使用的AWS Cloud Development Kit (AWS CDK)命令列介面 (CLI) (也稱為AWS CDK 工具組) 的完整版本。範例：`2.102.1`。請考慮指定完整版本，以確保建置和部署應用程式時的一致性和穩定性。

  或

- `latest`。 請考慮指`latest`定以利用 CDK CLI 的最新功能和修正程式。

此動作會將指定的 AWS CDK CLI 版本 (或最新版本) 下載至 CodeCatalyst 組建映像，然後使用此版本執行部署 CDK 應用程式或啟動AWS環境所需的命令。

如需您可以使用的受支援 CDK CLI 版本清單，請參閱AWS CDK版本。

如果您省略此屬性，動作會使用下列其中一個主題所述的預設 AWS CDK CLI 版本：

- 「AWS CDK 部署」動作所使用的 CDK CLI 版本
- 「AWS CDK 引導程序」操作使用的 CDK CLI 版本

對應的用戶界面：配置選項卡/AWS CDKCLI 版本

「AWS CDK 部署」動作參考

以下是AWS CDK 部署動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱新增「AWS CDK 部署」動作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢使用者介面元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKDeploy_nn:
```

```
    Identifier: aws/cdk-deploy@v1
    DependsOn:
      - CDKBootstrap
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_artifact
          Files:
            - "cdk.out/**/*"
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      StackName: my-cdk-stack
      Region: us-west-2
      Tags: '{"key1": "value1", "key2": "value2"}'
      Context: '{"key1": "value1", "key2": "value2"}'
      CdkCliVersion: version
      CdkRootPath: directory/cdk.json
      CfnOutputVariables: '["CnfOutputKey1","CfnOutputKey2","CfnOutputKey3"]'
      CloudAssemblyRootPath: path-to-cdk.out
```

CDKDeploy

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：CDKDeploy_nn。

對應的 UI：組態索引標籤/動作名稱

Identifier

(*CDKDeploy*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/cdk-deploy@v1。

對應的使用者介面：工作流程圖/CDKDeploy_nn/ aws/cdk 部署 @v1 標籤

DependsOn

(*CDKDeploy*/DependsOn)

(選用)

指定必須順利執行才能執行AWS CDK 部署動作的動作或動作群組。我們建議在DependsOn屬性中指定AWS CDK 引導操作，如下所示：

```
CDKDeploy:
  Identifier: aws/cdk-deploy@v1
  DependsOn:
    - CDKBootstrap
```

> **Note**
>
> 啟動載入是部署應用程式的必要先決條件。 AWS CDK 如果您未在工作流程中包含 AWS CDK Bootstrap 動作，則必須在執行部署動作之前找到另一種部署啟動程 AWS CDK 序堆疊的AWS CDK 方法。如需詳細資訊，請參閱 新增「AWS CDK 部署」動作 中的 必要條件。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*CDKDeploy*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*CDKDeploy*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  針對動作執行期間的彈性進行優化
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱關於運算類型。

對應的使用者介面：組態索引標籤/進階-選用/運算類型

Fleet

(*CDKDeploy*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

對應的使用者介面：組態索引標籤/進階-選用/運算叢集

Timeout

(*CDKDeploy*/Timeout)

(必要)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Inputs

(*CDKDeploy*/Inputs)

(選用)

本`Inputs`節定義了工作流程執行期間所`CDKDeploy`需的資料。

> **Note**
>
> 每個AWS CDK 部署動作只允許一個輸入 (來源或成品)。

對應的 UI：輸入索引標籤

Sources

(*CDKDeploy*/Inputs/Sources)

（如果您要部署的 AWS CDK 應用程序存儲在源存儲庫中，則為必填）

如果您的 AWS CDK 應用程序存儲在源存儲庫中，請指定該源存儲庫的標籤。部AWS CDK 署動作會在開始部署程序之前，在此儲存庫中合成應用程式。目前，唯一支援的標籤是`WorkflowSource`。

如果您的 AWS CDK 應用程序不包含在源存儲庫中，則它必須位於由另一個操作生成的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的用戶界面：輸入選項卡/源- 可選

Artifacts - input

(*CDKDeploy*/Inputs/Artifacts)

（如果您要部署的 AWS CDK 應用程序存儲在上一個操作的輸出成品中，則為必填）

如果您的 AWS CDK 應用程序包含在上一個操作生成的成品中，請在此處指定該成品。部AWS CDK 署動作會在開始部署程序之前，將指定成品中的應用程式合成為 CloudFormation範本。如果您的 AWS CDK 應用程序不包含在成品中，則它必須位於源存儲庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸入選項卡/加工品- 可選

Outputs

(*CDKDeploy*/Outputs)

(選用)

定義在工作流程執行期間由動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts - output

(*CDKDeploy*/Outputs/Artifacts

(選用)

指定動作所產生的人工因素。您可以參考這些人工因素作為其他動作中的輸入。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/成品

Name

(*CDKDeploy*/Outputs/Artifacts/Name)

（如果包含Artifacts - output，則為必填）

指定成品的名稱，該成品將包含在執行時期由AWS CDK 部署動作所合成的 AWS CloudFormation 範本。預設值為 `cdk_artifact`。如果您未指定成品，則動作會合成該範本，但不會將其儲存在成品中。請考慮將合成的範本儲存在成品中，以保留其記錄，以供測試或疑難排解之用。

對應的使用者介面：輸出索引標籤/人工因素/新增人工因素名稱

Files

(*CDKDeploy*/Outputs/Artifacts/Files)

（如果包含Artifacts - output，則為必填）

指定要包含在人工因素中的檔案。您必須指"cdk.out/**/*"定包含 AWS CDK 應用程序的綜合 AWS CloudFormation 模板。

> **Note**
>
> cdk.out是儲存合成檔案的預設目錄。如果您在cdk.json檔案中指定了輸出目錄以外cdk.out的目錄，請在此處指定該目錄，而不是cdk.out。

對應的 UI：輸出選項卡/人造物/添加人造物/構建生成的文件

Environment

(*CDKDeploy*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*CDKDeploy*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

(*CDKDeploy*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線Environment。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與您的環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS 戶連接

Name

(*CDKDeploy*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS 戶連接

Role

(*CDKDeploy*/Environment/Connections/Role)

(必要)

指定部AWS CDK 署動作用來存取 AWS 和部署應用 AWS CDK 程式堆疊的 IAM 角色名稱。請確定此角色包括：

- 下列權限原則：

> **Warning**
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
```

```
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::aws-account:role/cdk-*"
        }
    ]
}
```

- 下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色已新增至您的帳戶連線。若要進一步了解如何將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳

戶和空間建立CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全性風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Configuration

(*CDKDeploy*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

StackName

(*CDKDeploy*/Configuration/StackName)

(必要)

AWS CDK 應用程序堆棧的名稱，因為它顯示在應用 AWS CDK 程序目錄中的入口點文件中。bin下列範例顯示 TypeScript入口點檔案的內容，堆疊名稱會以*##*斜體反白顯示。如果您的入口點文件使用不同的語言，它看起來會很相似。

```
import * as cdk from 'aws-cdk-lib';
import { CdkWorksopTypescriptStack } from '../lib/cdk_workshop_typescript-stack';

const app = new cdk.App();
new CdkWorkshopTypescriptStack(app, 'CdkWorkshopTypescriptStack');
```

您只能指定一個堆疊。

> **Tip**
>
> 如果您有多個堆疊，您可以使用巢狀堆疊建立父系堆疊。然後，您可以在此動作中指定父系堆疊，以部署所有堆疊。

對應的 UI：配置選項卡/堆棧名稱

Region

(*CDKDeploy*/Configuration/Region)

(必要)

指定 AWS 區域 要部署 AWS CDK 應用程式堆疊的目標。如需區域代碼的清單，請參閱區域端點。

對應的 UI：配置選項卡/區域

Tags

(*CDKDeploy*/Configuration/Tags)

(選用)

指定要套用至應用 AWS CDK 程式堆疊中 AWS 資源的標籤。標籤會套用至堆疊本身，以及堆疊中的個別資源。如需有關標記的詳細資訊，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南中的標記。

對應的用戶界面：配置選項卡/高級-可選/標籤

Context

(*CDKDeploy*/Configuration/Context)

(選用)

以索引鍵值配對的形式指定要與 AWS CDK 應用程式堆疊產生關聯的前後關聯。如需前後關聯的詳細資訊，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南中的執行階段內容

對應的 UI：配置選項卡/高級-可選/上下文

CdkCliVersion

(*CDKDeploy*/Configuration/CdkCliVersion)

(選用)

此屬性適用於AWS CDK 部署動作的 1.0.13 版或更新版本，以及啟動程序動作的 1.0.8 版或更新版本 AWS CDK 。

請指定下列其中一項：

- 您希望此動作使用的 AWS Cloud Development Kit (AWS CDK) 命令列介面 (CLI) (也稱為 AWS CDK 工具組) 的完整版本。範例：`2.102.1`。請考慮指定完整版本，以確保建置和部署應用程式時的一致性和穩定性。

  或

- `latest`。 請考慮指`latest`定以利用 CDK CLI 的最新功能和修正程式。

此動作會將指定的 AWS CDK CLI 版本 (或最新版本) 下載至 CodeCatalyst 組建映像，然後使用此版本執行部署 CDK 應用程式或啟動 AWS 環境所需的命令。

如需您可以使用的受支援 CDK CLI 版本清單，請參閱AWS CDK 版本。

如果您省略此屬性，動作會使用下列其中一個主題中所述的預設 AWS CDK CLI 版本：

- 「AWS CDK 部署」動作所使用的 CDK CLI 版本
- 「AWS CDK 引導程序」操作使用的 CDK CLI 版本

對應的用戶界面：配置選項卡/AWS CDK CLI 版本

CdkRootPath

(*CDKDeploy*/Configuration/CdkRootPath)

(選用)

包含 AWS CDK 專案檔案之目錄的路`cdk.json`徑。部AWS CDK 署動作會從此資料夾執行，且動作建立的任何輸出都會新增至此目錄。如果未指定，則AWS CDK 部署動作會假設`cdk.json`檔案位於專案的根 AWS CDK 目錄中。

對應的用戶界面：cdk.json 所在的配置選項卡/目錄

CfnOutputVariables

(*CDKDeploy*/Configuration/CfnOutputVariables)

(選用)

指定應用程 AWS CDK 式程式碼中要公開為工作流程輸出變數的`CfnOutput`建構。然後，您可以在工作流程中的後續動作中參考工作流程輸出變數。如需中變數的更多資訊 CodeCatalyst，請參閱使用變數。

例如，如果您的 AWS CDK 應用程式程式碼如下所示：

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'my-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'bucketName', {
      value: bucket.bucketName,
      description: 'The name of the s3 bucket',
      exportName: 'myBucket',
    });
    const table = new dynamodb.Table(this, 'todos-table', {
      partitionKey: {name: 'todoId', type: dynamodb.AttributeType.NUMBER},
      billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
      removalPolicy: RemovalPolicy.DESTROY,
    })
    new CfnOutput(this, 'tableName', {
      value: table.tableName,
      description: 'The name of the dynamodb table',
      exportName: 'myDynamoDbTable',
    });
    ...
  }
}
```

... 你的`CfnOutputVariables`財產看起來像這樣：

```
Configuration:
  ...
```

```
  CfnOutputVariables: '["bucketName","tableName"]'
```

... 然後動作會產生下列工作流程輸出變數：

金鑰	值
bucketName	bucket.bucketName
tableName	table.tableName

然後您可以在後續動作中參照bucketName和tableName變數。若要瞭解如何在後續動作中參考工作流程輸出變數，請參閱引用預定義的變量。

如果您未在CfnOutputVariables屬性中指定任何CfnOutput建構，則動作會將它找到的前四個 (或更少) CloudFormation 輸出變數公開為工作流程輸出變數。如需詳細資訊，請參閱 「AWS CDK部署」動作變數。

> **Tip**
>
> 若要取得動作產生的所有 CloudFormation 輸出變數清單，請執行一次包含AWS CDK 部署動作的工作流程，然後查看動作的 [記錄檔] 索引標籤。日誌包含與您的應 AWS CDK 用程序關聯的所有 CloudFormation 輸出變量的列表。一旦知道所有 CloudFormation 變數是什麼，就可以使用CfnOutputVariables屬性指定要轉換為工作流程輸出變數的變數。

有關 AWS CloudFormation 輸出變量的更多信息，請參閱CfnOutput構造的文檔，可在 AWS Cloud Development Kit (AWS CDK) API 參考中的類 CfnOutput （構造）中找到。

對應的 UI：配置選項卡/AWS CloudFormation 輸出變量

CloudAssemblyRootPath

(*CDKDeploy*/Configuration/CloudAssemblyRootPath)

(選用)

如果您已經將應用 AWS CDK 程序的堆棧合成到雲程序集中（使用該cdk synth操作），請指定 Cloud 程序集目錄（cdk.out）的根路徑。位於指定雲端組件目錄中的 AWS CloudFormation 範本將透過部AWS CDK 署動作部署到您 AWS 帳戶 使用指cdk deploy --app令。如果存在該--app選項，則不會cdk synth執行該操作。

如果您未指定雲端組合目錄，則AWS CDK 部署動作將在沒有--app選項的情況下執行指cdk deploy令。如果沒有--app選項，cdk deploy操作將合成（cdk synth）並將您的 AWS CDK 應用程序部署到您的 AWS 帳戶.

當「AWS CDK 部署」動作可以在執行階段執行合成時，為什麼要指定現有的、合成的雲端組件？

您可能想要將既有、合成的雲端組合指定為：

- 確保每次執行「AWS CDK 部署」動作時，部署完全相同的資源集

  如果您未指定雲端組件，則AWS CDK 部署動作可能會根據執行的時間來合成和部署不同的檔案。例如，AWS CDK 部署動作可能會在測試階段合成具有一組相依性的雲端組件，以及在生產階段期間 (如果這些相依性在階段之間變更)，合成另一組相依性。為了保證測試內容與部署的內容之間的確切同位檢查，我們建議合成一次，然後使用 [路徑到雲端組件目錄] 欄位 (視覺化編輯器) 或CloudAssemblyRootPath屬性 (YAML 編輯器) 來指定已經合成的雲端組件。

- 使用非標準軟件包管理器和工具與 AWS CDK 應用程序

  在synth作業期間，部AWS CDK 署動作會嘗試使用 npm 或 pip 等標準工具來執行您的應用程式。如果該操作無法使用這些工具成功運行您的應用程序，則不會發生合成，並且操作將失敗。若要解決此問題，您可以指定在應用程式cdk.json檔案中成功執行應用程 AWS CDK 式所需的確切命令，然後使用不涉及AWS CDK 部署動作的方法來合成應用程式。產生雲端組件之後，您可以在AWS CDK 部署動作的 [雲端組件目錄路徑] 欄位 (視覺化編輯器) 或CloudAssemblyRootPath屬性 (YAML 編輯器) 中指定它。

如需設定cdk.json檔案以包含 AWS CDK 用於安裝和執行應用程式之命令的詳細資訊，請參閱指定應用程式命令。

有關cdk deploy和cdk synth命令以及--app選項的詳細資訊，請參閱AWS Cloud Development Kit (AWS CDK) 開發人員指南中的部署堆疊、合成堆疊和略過合成。

如需有關雲端組件的資訊，請參閱 AWS Cloud Development Kit (AWS CDK) API 參考中的雲端組件。

對應的 UI：配置選項卡/雲組件目錄的路徑

「AWS Lambda 調用」操作引用

以下是AWS Lambda 呼叫動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱添加「AWS Lambda調用」操作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 元素將與其關聯的 YAML 屬性列出。

```yaml
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  LambdaInvoke_nn:
    Identifier: aws/lambda-invoke@v1
    DependsOn:
      - dependent-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - request-payload
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      Function: my-function|function-arn
      AWSRegion: us-west-2
      # Specify RequestPayload or RequestPayloadFile, but not both.
```

```
      RequestPayload: '{"firstname": "Li", lastname: "Jean", "company": "ExampleCo",
"team": "Development"}'
      RequestPayloadFile: my/request-payload.json
      ContinueOnError: true|false
      LogType: Tail|None
      ResponseFilters: '{"name": ".name", "company": ".department.company"}'
    Outputs:
      Artifacts:
        - Name: lambda_artifacts
          Files:
            - "lambda-response.json"
```

LambdaInvoke

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：Lambda_Invoke_Action_Workflow_nn。

對應的 UI：組態索引標籤/動作名稱

Identifier

(*LambdaInvoke*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/lambda-invoke@v1。

對應的用戶界面：工作流程圖/LambdaInvoke_nn/ aws/羊肉調用 @v1 標籤

DependsOn

(*LambdaInvoke*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*LambdaInvoke*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*LambdaInvoke*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  優化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的 UI：配置選項卡/計算類型

Fleet

(*LambdaInvoke*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

對應的 UI：組態索引標籤/運算叢集

Timeout

(*LambdaInvoke*/Timeout)

(必要)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Inputs

(*LambdaInvoke*/Inputs)

(必要)

Inputs本節定義了在工作流程執行期間AWS Lambda 呼叫動作所需的資料。

> **Note**
>
> 每個AWS Lambda 叫用動作只允許一個輸入 (來源或人工因素)。變數不會計入此總數。

對應的 UI：輸入索引標籤

Sources

(*LambdaInvoke*/Inputs/Sources)

（如果提供 RequestPayloadFile，則為必填）

如果您想要將請求承載 JSON 檔案傳遞至 AWS Lambda invoke 動作，且此承載檔案儲存在來源存放庫中，請指定該來源存放庫的標籤。目前，唯一支援的標籤是WorkflowSource。

如果您的請求承載檔案不包含在來源儲存庫中，則該檔案必須位於由另一個動作產生的成品中。

如需有關承載檔案的詳細資訊，請參閱RequestPayloadFile。

> **Note**
>
> 您可以使用RequestPayload屬性將承載的 JSON 程式碼直接新增至動作，而不是指定承載檔案。如需詳細資訊，請參閱 RequestPayload。

如需來源的詳細資訊，請參閱 使用來源。

對應的 UI：輸入選項卡/源- 可選

Artifacts - input

(*LambdaInvoke*/Inputs/Artifacts)

（如果提供 RequestPayloadFile，則為必填）

如果您想要將請求承載 JSON 檔案傳遞至 AWS Lambda invoke 動作，且此承載檔案包含在先前動作的輸出成品中，請在此指定該成品。

如需有關承載檔案的詳細資訊，請參閱RequestPayloadFile。

> **Note**
>
> 您可以使用RequestPayload屬性將承載的 JSON 程式碼直接新增至動作，而不是指定承載檔案。如需詳細資訊，請參閱 RequestPayload。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：組態索引標籤/成品- 選用

Variables - input

(*LambdaInvoke*/Inputs/Variables)

(選用)

指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸入選項卡/變量- 可選

Environment

(*LambdaInvoke*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*LambdaInvoke*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：組態索引標籤/「環境/連線/角色」/「環境」

Connections

(*LambdaInvoke*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與您的環境建立關聯的資訊，請參閱建立環境。

對應的 UI：組態索引標籤/「環境/連線/角色」/「連線」

Name

(*LambdaInvoke*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：組態索引標籤/「環境/連線/角色」/「連線」

Role

(*LambdaInvoke*/Environment/Connections/Role)

(必要)

指定叫用動作用來存取 AWS 和AWS Lambda 叫用 Lambda 函數的 IAM 角色名稱。請確定此角色包括：

- 下列權限原則：

> **Warning**
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:aws-region:aws-account:function:function-
name"
        }
    ]
}
```

- 下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色與您的帳戶連線相關聯。若要進一步了解如何將 IAM 角色與帳戶連線建立關聯，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全性風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/連線/角色」/角色

Configuration

(*LambdaInvoke*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

Function

(*LambdaInvoke*/Configuration/Function)

(必要)

指定此動作將呼叫的 AWS Lambda 函數。您可以指定函數的名稱，或其 Amazon 資源名稱（ARN）。您可以在 Lambda 主控台中找到名稱或 ARN。

> **Note**
>
> Lambda 函數所在的 AWS 帳戶可以與下指定的帳戶不同`Connections:`。

對應的 UI：配置選項卡/功能

AWSRegion

(*LambdaInvoke*/Configuration/AWSRegion)

(必要)

指定 AWS Lambda 函數所在的 AWS 區域。如需區域代碼的清單，請參閱 AWS 一般參考.

對應的 UI：配置選項卡/目標存儲桶- 可選

RequestPayload

(*LambdaInvoke*/Configuration/RequestPayload)

(選用)

如果要將請求有效負載傳遞給 AWS Lambda invoke 動作，請在此處以 JSON 格式指定請求有效負載。

請求有效負載示例：

```
'{ "key": "value" }'
```

如果您不想將請求裝載傳遞給 Lambda 函數，請省略此屬性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同時指定兩者。

如需要求承載的詳細資訊，請參閱 AWS Lambda API 參考中的 Invoke 主題。

對應的 UI：配置選項卡/請求有效負載- 可選

RequestPayloadFile

(*LambdaInvoke*/Configuration/RequestPayloadFile)

(選用)

如果要將請求有效負載傳遞給 AWS Lambda invoke 動作，請在此處指定此請求有效負載文件的路徑。檔案必須是 JSON 格式。

請求有效負載檔案可以位於來源儲存庫中，也可以位於先前動作的成品中。檔案路徑相對於來源儲存庫或人工因素根目錄。

如果您不想將請求裝載傳遞給 Lambda 函數，請省略此屬性。

> **Note**
>
> 您可以指定 `RequestPayload` 或 `RequestPayloadFile`，但不能同時指定兩者。

有關請求承載文件的詳細信息，請參閱 AWS Lambda API 參考中的調用主題。

對應的 UI：配置選項卡/請求有效負載文件- 可選

ContinueOnError

(*LambdaInvoke*/Configuration/RequestPayloadFile)

(選用)

指定即使叫用 AWS Lambda 函數失敗，您是否要將 AWS Lambda invoke 動作標示為成功。考慮將此屬性設定為，`true`以允許在 Lambda 失敗的情況下啟動工作流程中的後續動作。

預設值是如果 Lambda 函數失敗 (在可視化編輯器或 `false` YAML 編輯器中「關閉」)，則會失敗動作。

對應的用戶界面：配置選項卡/錯誤時繼續

LogType

(*LambdaInvoke*/Configuration/LogType)

(選用)

指定是否要在叫用 Lambda 函數後將錯誤記錄包含在來自 Lambda 函數的回應中。您可以在主控台的 Lambda 叫用動作的 [記錄] 索引標籤中檢視這些記 CodeCatalyst 錄。可能值為：

- `Tail`— 返回日誌
- None-不返回日誌

預設為尾巴。

如需有關記錄類型的詳細資訊，請參閱 AWS Lambda API 參考中的「叫用」主題。

如需檢視日誌檔案的詳細資訊，請參閱檢視工作流程執行狀態與詳細。

對應的 UI：配置選項卡/日誌類型

ResponseFilters

(*LambdaInvoke*/Configuration/ResponseFilters)

(選用)

指定 Lambda 回應承載中要轉換為輸出變數的索引鍵。然後，您可以在工作流程中的後續動作中參照輸出變數。如需中變數的更多資訊 CodeCatalyst，請參閱使用變數。

例如，如果您的響應有效負載如下所示：

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 您的響應過濾器如下所示：

```
Configuration:
  ...
  ResponseFilters: '{"name": ".name", "company": ".department.company"}'
```

... 然後動作會產生下列輸出變數：

金鑰	值
name	Saanvi
company	Amazon

然後您可以在後續動作中參照name和company變數。

如果您未在中指定任何索引鍵ResponseFilters，則動作會將 Lambda 回應中的每個頂層金鑰轉換為輸出變數。如需詳細資訊，請參閱「AWS Lambda調用」動作變量。

請考慮使用回應篩選器，將產生的輸出變數限制為只有您實際想要使用的變數。

對應的 UI：配置選項卡/響應過濾器- 可選

Outputs

(*LambdaInvoke*/Outputs)

(選用)

定義在工作流程執行期間由動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts

(*LambdaInvoke*/Outputs/Artifacts)

(選用)

指定動作產生的人工因素。您可以參考這些成品做為其他動作中的輸入。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/人工因素/構建成品名稱

Name

(*LambdaInvoke*/Outputs/Artifacts/Name)

(選用)

指定將包含 Lambda 函數傳回之 Lambda 回應承載之成品的名稱。預設值為 `lambda_artifacts`。如果您未指定成品，則可以在動作的記錄中檢視 Lambda 回應裝載，該記錄可在 CodeCatalyst 主控台中的動作的 [記錄] 索引標籤上找到。如需檢視日誌檔案的詳細資訊，請參閱檢視工作流程執行狀態與詳細。

對應的 UI：輸出選項卡/人工因素/構建成品名稱

Files

(*LambdaInvoke*/Outputs/Artifacts/Files)

(選用)

指定要包含在人工因素中的檔案。您必須指定`lambda-response.json`以便包含 Lambda 回應承載檔案。

對應的 UI：輸出選項卡/人工/構建生成的文件

「部署AWS CloudFormation堆疊」動作參考

以下是部署AWS CloudFormation堆疊動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱新增「部署 AWS CloudFormation 堆疊」動作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
```

```
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployRole
    Inputs:
      Sources:
        - source-name-1
      Artifacts:
        - CloudFormation-artifact
    Configuration:
      name: stack-name
      region: us-west-2
      template: template-path
      role-arn: arn:aws:iam::123456789012:role/StackRole
      capabilities: CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND
      parameter-overrides: KeyOne=ValueOne,KeyTwo=ValueTwo | path-to-JSON-file
      no-execute-changeset: 1|0
      fail-on-empty-changeset: 1|0
      disable-rollback: 1|0
      termination-protection: 1|0
      timeout-in-minutes: minutes
      notification-arns: arn:aws:sns:us-east-1:123456789012:MyTopic,arn:aws:sns:us-
east-1:123456789012:MyOtherTopic
      monitor-alarm-arns: arn:aws:cloudwatch::123456789012:alarm/
MyAlarm,arn:aws:cloudwatch::123456789012:alarm/MyOtherAlarm
      monitor-timeout-in-minutes: minutes
      tags: '[{"Key":"MyKey1","Value":"MyValue1"},{"Key":"MyKey2","Value":"MyValue2"}]'
```

DeployCloudFormationStack

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：DeployCloudFormationStack_nn。

對應的 UI：組態索引標籤/動作顯示名稱

Identifier

(*DeployCloudFormationStack*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：`aws/cfn-deploy@v1`。

對應的使用者介面：工作流程圖/DeployCloudFormationStack_nn/ aws/cfn-部署@v1 標籤

DependsOn

(*DeployCloudFormationStack*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*DeployCloudFormationStack*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*DeployCloudFormationStack*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  優化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的使用者介面：組態索引標籤/進階-選用/運算類型

Fleet

(*DeployCloudFormationStack*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

對應的使用者介面：組態索引標籤/進階-選用/運算叢集

Timeout

(*DeployCloudFormationStack*/Timeout)

(選用)

指定動作在 CodeCatalyst 結束動作之前可執行的時間長度 (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時（分鐘）- 可選

Environment

(*DeployCloudFormationStack*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*DeployCloudFormationStack*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

(*DeployCloudFormationStack*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Name

(*DeployCloudFormationStack*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Role

(*DeployCloudFormationStack*/Environment/Connections/Role)

(必要)

指定部署AWS CloudFormation堆疊動作用於存取AWS和AWS CloudFormation服務的 IAM 角色名稱。

請確定此角色包含下列原則：

- 下列權限原則：

> **Warning**
>
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色時，請在資源策略陳述式中使用下列萬用字元，然後在可用資源名稱之後將策略範圍縮小。
>
> ```
> "Resource": "*"
> ```

- 下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色已新增至您空間中的帳戶連線。若要進一步了解如何將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Inputs

(*DeployCloudFormationStack*/Inputs)

(選用)

本Inputs節定義工作流程執行期間所DeployCloudFormationStack需的資料。

> **Note**
>
> 每個「部署AWS CloudFormation堆疊」動作最多允許四個輸入 (一個來源和三個成品)。

如果您需要引用駐留在不同輸入中的文件（例如源和工件），則源輸入是主輸入，而工件是輔助輸入。輔助輸入中的文件的引用採用特殊前綴，以將它們從主輸入中刪除。如需詳細資訊，請參閱 範例：參考多個成品中的檔案。

對應的 UI：輸入索引標籤

Sources

(*DeployCloudFormationStack*/Inputs/Sources)

(如果您的 CloudFormation 或AWS SAM範本儲存在來源儲存庫中，則需要)

如果您的 CloudFormation 或AWS SAM範本儲存在來源儲存庫中，請指定該來源儲存庫的標籤。目前，唯一支援的標籤是`WorkflowSource`。

如果您的 CloudFormation 或AWS SAM範本未包含在來源儲存庫中，則該範本必須位於由其他動作產生的成品或 Amazon S3 儲存貯體中。

如需來源的詳細資訊，請參閱 使用來源。

對應的 UI：輸入選項卡/源- 可選

Artifacts - input

(*DeployCloudFormationStack*/Inputs/Artifacts)

(如果您的 CloudFormation 或AWS SAM範本儲存在先前動作的輸出成品中，則需要)

如果您要部署的 CloudFormation 或AWS SAM範本包含在先前動作所產生的成品中，請在此指定該成品。如果您的 CloudFormation 範本未包含在成品中，則該範本必須位於來源儲存庫或 Amazon S3 儲存貯體中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：配置選項卡/成品- 可選

Configuration

(*DeployCloudFormationStack*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

name

(*DeployCloudFormationStack*/Configuration/name)

(必要)

指定部署 CloudFormation 堆疊動作建立或更新的AWS CloudFormation堆疊名稱。

對應的 UI：配置選項卡/堆棧名稱

region

(*DeployCloudFormationStack*/Configuration/region)

(必要)

指定AWS 區域要部署堆疊的目標。如需區域代碼的清單，請參閱區域端點。

對應的 UI：配置選項卡/堆棧區域

template

(*DeployCloudFormationStack*/Configuration/template)

(必要)

指定 CloudFormation 或AWS SAM範本檔案的名稱和路徑。範本可以是 JSON 或 YAML 格式，而且可以位於來源儲存庫、先前動作的成品或 Amazon S3 儲存貯體中。如果範本檔案位於來源儲存庫或人工因素中，則路徑會相對於來源或人工因素根目錄。如果範本位於 Amazon S3 儲存貯體中，則路徑為範本的物件 URL 值。

範例：

```
./MyFolder/MyTemplate.json
```

```
MyFolder/MyTemplate.yml
```

```
https://MyBucket.s3.us-west-2.amazonaws.com/MyTemplate.yml
```

> **Note**
>
> 您可能需要在範本的檔案路徑中新增前置詞，以指出要在其中尋找的成品或來源。如需詳細資訊，請參閱 參考來源儲存庫中的檔案 及 參考人工因素中的檔案。

對應的 UI：配置選項卡/模板

role-arn

(*DeployCloudFormationStack*/Configuration/role-arn)

(必要)

指定堆疊角色的 Amazon 資源名稱 (ARN)。 CloudFormation 使用此角色存取和修改堆疊中的資源。例如：`arn:aws:iam::123456789012:role/StackRole`。

請確定堆疊角色包括：

- 一或多個權限原則。原則取決於您堆疊中擁有的資源。例如，如果您的堆疊包含AWS Lambda函數，則需要新增授與 Lambda 存取權的許可。如果您遵循中所述的教學課程教學課程：使用部署無伺服器應用程式 AWS CloudFormation，它會包含一個標題為的程序若要建立堆疊角色，列出當您部署典型的無伺服器應用程式堆疊時，堆疊角色所需的權限。

  > **Warning**
  >
  > 將權限限制為 CloudFormation 服務存取堆疊中資源所需的權限。使用具有更廣泛權限的角色可能會造成安全風險。

- 下列信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

選擇性地將此角色與您的帳戶連線產生關聯。若要進一步了解如何將 IAM 角色與帳戶連線建立關聯，請參閱將 IAM 角色新增至帳戶連線。如果您未將堆疊角色與帳戶連線產生關聯，則堆疊角色將不會出現在可視化編輯器的 [堆疊角色] 下拉式清單中；不過，仍可使用 YAML 編輯器在`role-arn`欄位中指定角色 ARN。

> **Note**
>
> 您可以在此處指定`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***`spaceName`***角色。瞭解`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：配置選項卡/堆棧角色- 可選

capabilities

(*`DeployCloudFormationStack`*/Configuration/capabilities)

(必要)

指定允許AWS CloudFormation建立特定堆疊所需的 IAM 功能清單。在大多數情況下，您可`capabilities`以保留預設值`CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND`。

如果您`##[error] requires capabilities: [`*`capability-name`*`]`在部署AWS CloudFormation堆疊動作的記錄檔中看到，請參閱以如何修正 IAM 功能錯誤？取得有關如何修正問題的詳細資訊。

如需 IAM 功能的詳細資訊，請參閱 IAM 使用者指南中的AWS CloudFormation範本確認 IAM 資源。

對應的使用者介面：組態索引標籤/進階/功能

parameter-overrides

(*`DeployCloudFormationStack`*/Configuration/parameter-overrides)

(選用)

在您的AWS CloudFormation或AWS SAM範本中指定沒有預設值的參數，或您要為其指定非預設值的參數。若要取得有關參數的更多資訊，請參閱《AWS CloudFormation使用指南》中的參數

該`parameter-overrides`物業接受：

- 包含參數和值的 JSON 檔案。
- 以逗號分隔的參數和值清單。

若要指定 JSON 檔案

1. 請確定 JSON 檔案使用下列其中一種語法：

```
{
  "Parameters": {
    "Param1": "Value1",
    "Param2": "Value2",
    ...
  }
}
```

或者...

```
[
  {
     "ParameterKey": "Param1",
     "ParameterValue": "Value1"
  },
  ...
]
```

（還有其他語法，但 CodeCatalyst 在撰寫本文時不支持它們。）如需有關在 JSON 檔案中指定 CloudFormation 參數的詳細資訊，請參閱AWS CLI命令參考中支援的 JSON 語法。

2. 使用下列其中一種格式指定 JSON 檔案的路徑：

   - 如果您的 JSON 檔案位於先前動作的輸出成品中，請使用：

     `file:///artifacts/`*`current-action-name`*`/`*`output-artifact-name`*`/`*`path-to-json-file`*

如需詳細資訊，請參閱範例 1

- 如果您的 JSON 文件駐留在源存儲庫中，請使用：

  `file:///sources/WorkflowSource/`*`path-to-json-file`*

  如需詳細資訊，請參閱範例 2

  範例 1 — JSON 檔案位於輸出成品中

```
##My workflow YAML
...
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ParamArtifact
          Files:
            - params.json
    Configuration:
    ...
  MyDeployCFNStackAction:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: file:///artifacts/MyDeployCFNStackAction/
ParamArtifact/params.json
```

  範例 2 — JSON 檔案位於來源儲存庫中，位於名為的資料夾中 `my/folder`

```
##My workflow YAML
...
Actions:
  MyDeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      parameter-overrides: file:///sources/WorkflowSource/my/folder/params.json
```

使用逗號分隔參數清單的步驟

- 使用下列格式在parameter-overrides屬性中新增參數名稱-值配對：

  param-1=*value-1*,*param-2*=*value-2*

  例如，假設下列AWS CloudFormation範本：

```
##My CloudFormation template

Description: My AWS CloudFormation template

Parameters:
  InstanceType:
    Description: Defines the Amazon EC2 compute for the production server.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t3.medium

Resources:
...
```

  ... 您可以按如下方式設置parameter-overrides屬性：

```
##My workflow YAML
...
Actions:
...
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: InstanceType=t3.medium,UseVPC=true
```

  > **Note**
  >
  > 您可以使用作為值來指定沒有對應值undefined的參數名稱。例如：
  > parameter-overrides: MyParameter=undefined

效果是在堆疊更新期間， CloudFormation 會使用指定參數名稱的現有參數值。

對應的介面：

- 組態標籤/進階/參數覆寫
- 組態標籤/進階/參數覆寫/使用檔案指定取代
- 組態標籤/進階/參數覆寫/使用值組指定取代

no-execute-changeset

(*DeployCloudFormationStack*/Configuration/no-execute-changeset)

(選用)

指定是否 CodeCatalyst 要建立變更集，然後在執行 CloudFormation 變更集之前停止變更集。這可讓您檢閱 CloudFormation 主控台中設定的變更。如果您確定變更集看起來不錯，請停用此選項，然後重新執行工作流程， CodeCatalyst 以便建立並執行變更集而不會停止。預設為建立並執行變更集而不停止。如需詳細資訊，請參閱《AWS CLI命令參考》中的AWS CloudFormation部署參數。若要取得有關檢視變更集的更多資訊，請參閱《AWS CloudFormation使用指南》中的〈檢視變更集〉。

對應的使用者介面：組態標籤/進階/無執行變更集

fail-on-empty-changeset

(*DeployCloudFormationStack*/Configuration/fail-on-empty-changeset)

(選用)

指定如果 CloudFormation 變更集 CodeCatalyst 為空，是否要讓「部署AWS CloudFormation堆疊」動作失敗。(如果變更集是空的，表示在最新部署期間沒有對堆疊進行任何變更。) 預設值是允許在變更集為空時繼續執行動作，並傳回UPDATE_COMPLETE訊息，即使堆疊未更新。

如需有關此設定的詳細資訊，請參閱《AWS CLI命令參考》中的 AWS CloudFormation deploy 參數。如需有關變更集的詳細資訊，請參閱使用指南中的使用變更集更新堆疊。AWS CloudFormation

對應的 UI：配置選項卡/高級/空變更集上失敗

disable-rollback

(*DeployCloudFormationStack*/Configuration/disable-rollback)

(選用)

指定 CodeCatalyst 堆疊部署失敗時是否要復原。回滾將堆棧返回到最後一個已知的穩定狀態。預設為啟用復原。如需有關此設定的詳細資訊，請參閱《AWS CLI命令參考》中的 AWS CloudFormation deploy 參數。

如需部署AWS CloudFormation堆疊動作如何處理復原的詳細資訊，請參閱設定復原。

若要取得有關復原堆疊的詳細資訊，請參閱《AWS CloudFormation使用指南》中的〈堆疊失敗選項〉。

對應的使用者介面：設定索引標籤/進階/停用復原

termination-protection

(*DeployCloudFormationStack*/Configuration/termination-protection)

(選用)

指定是否要 Deploy AWS CloudFormation 堆疊將終止保護新增至其部署的堆疊。如果使用者嘗試刪除啟用終止保護的堆疊，則刪除會失敗，且堆疊及其狀態保持不變。預設為停用終止保護。若要取得更多資訊，請參閱《AWS CloudFormation使用指南》中的〈保護堆疊不被刪除〉。

對應的使用者介面：組態索引標籤/進階/終止保護

timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/timeout-in-minutes)

(選用)

指定逾時堆疊建立作業並將堆疊狀態設定為之前 CloudFormation 應分配的時間量 (以分鐘為CREATE_FAILED單位)。如果 CloudFormation 無法在分配的時間內建立整個堆疊，操作便會因為逾時而失敗，並復原堆疊。

依預設，堆疊建立沒有逾時。不過，個別資源可能會因其實作的服務性質而有自己的逾時。例如，如果堆疊中的個別資源逾時，堆疊建立也會逾時，即使尚未達到您指定的堆疊建立逾時。

對應的使用者介面：組態索引標籤/進階/逾CloudFormation時

notification-arns

(*DeployCloudFormationStack*/Configuration/notification-arns)

(選用)

指定您要 CodeCatalyst 向其傳送通知訊息之 Amazon SNS 主題的 ARN。例如 `arn:aws:sns:us-east-1:111222333:MyTopic`。執行 Deploy AWS CloudFormation 堆疊動作時，請 CodeCatalyst 協調 CloudFormation 以在堆疊建立或更新程序期間發生的每個AWS CloudFormation事件傳送一個通知。事件會顯示在堆疊的AWS CloudFormation主控台的 [事件] 索引標籤中。) 您最多可以指定五個主題。如需詳細資訊，請參閱什麼是 Amazon VPC？。

對應的使用者介面：組態索引標籤/進階/通知 ARN

monitor-alarm-arns

(*DeployCloudFormationStack*/Configuration/monitor-alarm-arns)

(選用)

指定要用作回滾觸發器的 Amazon CloudWatch 警報的 Amazon 資源名稱 (ARN)。例如 `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`。您最多可以有五個回復觸發器。

> **Note**
>
> 如果您指定 CloudWatch 警報 ARN，您還需要配置其他權限以啟用該動作才能訪問 CloudWatch。如需詳細資訊，請參閱 設定復原。

對應的使用者介面：組態標籤/進階/監控警示 ARN

monitor-timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/monitor-timeout-in-minutes)

(選用)

指定從 0 到 180 分鐘的時間長度，在此期間 CloudFormation 監視指定的警報。在部署所有堆疊資源之後開始監視。如果警示在指定的監視時間內發生，則部署會失敗，並 CloudFormation 復原整個堆疊作業。

預設值：0。 CloudFormation 只會在部署堆疊資源時監控警示，而不會在之後監視警示。

對應的使用者介面：組態標籤/進階/監控時間

tags

(*DeployCloudFormationStack*/Configuration/tags)

(選用)

指定要附加到 CloudFormation 堆疊的標籤。標籤是任意索引鍵值配對，可用來識別堆疊，以達成成本分配等目的。如需有關哪些標籤以及如何使用這些標籤的詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的標記資源。若要取得有關在中加標籤的更多資訊 CloudFormation，請參閱《AWS CloudFormation使用指南》中的〈設定AWS CloudFormation堆疊

按鍵可以包含英數字元或空格，最多可包含 127 個字元。值可以包含英數字元或空格，最多可包含 255 個字元。

您最多可以為每個堆疊新增 50 個唯一的標籤。

對應的使用者介面：設定索引標籤/進階/標籤

「部署到 Amazon ECS」動作參考

以下是「部署到 Amazon ECS」動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱新增「部署到 Amazon ECS」動作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSDeployAction_nn:
    Identifier: aws/ecs-deploy@v1
    DependsOn:
```

```
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployToECS
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - task-definition-artifact
    Configuration:
      region: us-east-1
      cluster: ecs-cluster
      service: ecs-service
      task-definition: task-definition-path
      force-new-deployment: false|true
      codedeploy-appspec: app-spec-file-path
      codedeploy-application: application-name
      codedeploy-deployment-group: deployment-group-name
      codedeploy-deployment-description: deployment-description
```

ECSDeployAction

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：ECSDeployAction_nn。

對應的 UI：組態索引標籤/動作顯示名稱

Identifier

(*ECSDeployAction*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/ecs-deploy@v1。

對應的使用者介面：工作流程圖/ECS DeployAction _nn/ aws/EC-部署@v1 標籤

DependsOn

(*ECSDeployAction*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*ECSDeployAction*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*ECSDeployAction*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的使用者介面：組態索引標籤/進階-選用/運算類型

Fleet

(*ECSDeployAction*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如`Fleet`果省略，預設值為`Linux.x86-64.Large`。

對應的使用者介面：組態索引標籤/進階-選用/運算叢集

Timeout

(*ECSDeployAction*/Timeout)

(選用)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Environment

(*ECSDeployAction*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*ECSDeployAction*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

(*ECSDeployAction*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與您的環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Name

(*ECSDeployAction*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Role

(*ECSDeployAction*/Environment/Connections/Role)

(必要)

指定「部署到 Amazon ECS」動作用來存取AWS的 IAM 角色名稱。請確定此角色包含下列原則：

- 下列權限原則：

> **Warning**
>
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
```

```
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用角色時，請在資源策略陳述式中使用下列萬用字元，然後在策略可用之後使用資源名稱來定義策略的範圍。
>
> ```
> "Resource": "*"
> ```

- 下列自訂信任原則：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定此角色已新增至您的帳戶連線。若要進一步了解如何將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全性風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Inputs

(*ECSDeployAction*/Inputs)

(選用)

本Inputs節定義了工作流程執行期間所ECSDeployAction需的資料。

> **Note**
>
> 每個「部署到 Amazon ECS」動作只允許一個輸入 (來源或成品)。

對應的 UI：輸入索引標籤

Sources

(*ECSDeployAction*/Inputs/Sources)

(如果您的任務定義檔案儲存在來源儲存庫中，則需要)

如果您的任務定義檔案儲存在來源儲存庫中，請指定該來源儲存庫的標籤。目前，唯一支援的標籤是`WorkflowSource`。

如果您的任務定義檔案不包含在來源儲存庫中，則該檔案必須位於由其他動作產生的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的 UI：輸入選項卡/源- 可選

Artifacts - input

(*ECSDeployAction*/Inputs/Artifacts)

(如果您的任務定義檔案儲存在上一個動作的輸出成品中，則為必要)

如果您要部署的任務定義檔案包含在先前動作所產生的成品中，請在此指定該成品。如果您的任務定義檔案不包含在成品中，則該檔案必須位於來源儲存庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：配置選項卡/成品- 可選

Configuration

(*ECSDeployAction*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

region

(Configuration/region)

(必要)

指定您的 Amazon ECS 叢集和服務所在的AWS區域。如需區域代碼的清單，請參閱 AWS 一般參考.

對應的 UI：組態索引標籤/區域

cluster

(*ECSDeployAction*/Configuration/cluster)

(必要)

指定現有 Amazon ECS 叢集的名稱。「部署到 Amazon ECS」動作會將您的容器化應用程式做為任務部署到此叢集中。如需 Amazon ECS 叢集的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的叢集。

對應的 UI：組態索引標籤/叢集

service

(*ECSDeployAction*/Configuration/service)

(必要)

指定將實例化任務定義檔案的現有 Amazon ECS 服務的名稱。此服務必須位於`cluster`欄位中指定的叢集下方。如需 Amazon ECS 服務的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 服務。

對應的 UI：組態索引標籤/服務

task-definition

(*ECSDeployAction*/Configuration/task-definition)

(必要)

指定既有工作定義檔案的路徑。如果檔案位於來源儲存庫中，則路徑會相對於來源儲存庫根資料夾。如果您的檔案位於先前工作流程動作的人工因素中，則路徑會相對於人工因素根資料夾。如需有關任務定義檔案的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的任務定義。

對應的 UI：配置選項卡/任務定義

force-new-deployment

(*ECSDeployAction*/Configuration/force-new-deployment)

(必要)

如果啟用，Amazon ECS 服務就能在不變更服務定義的情況下啟動新的部署。強制部署會導致服務停止所有目前正在執行的工作並啟動新工作。如需有關強制執行新部署的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的更新服務。

預設：`false`

對應的用戶界面：配置選項卡/強制新的服務部署

codedeploy-appspec

(*ECSDeployAction*/Configuration/codedeploy-appspec)

(如果您已將 Amazon ECS 服務設定為使用藍/綠部署，則為必要項目，否則請省略)

指定現有 CodeDeploy 應用程式規格 (AppSpec) 檔案的名稱和路徑。此檔案必須位於來 CodeCatalyst 源儲存庫的根目錄中。若要取得有關 AppSpec 檔案的更多資訊，請參閱《AWS CodeDeploy使用指南》中的CodeDeploy 應用程式規格 (AppSpec) 檔案

> **Note**
>
> 只有在您已將 Amazon ECS 服務設定為執行藍/綠部署時，才提供 CodeDeploy 資訊。對於循環更新部署 (預設值)，請省略 CodeDeploy 資訊。如需 Amazon ECS 部署的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 部署類型。

> **Note**
>
> 這些CodeDeploy欄位可能會隱藏在視覺化編輯器中。若要讓它們出現，請參閱為什麼可視化編輯器中缺少 CodeDeploy 字段？。

對應的 UI：配置選項卡/CodeDeploy AppSpec

codedeploy-application

(*ECSDeployAction*/Configuration/codedeploy-application)

（如果包含`codedeploy-appspec`，則為必填）

指定現有 CodeDeploy 應用程式的名稱。若要取得有關 CodeDeploy 應用程式的更多資訊，請參閱使用指南 CodeDeploy中的〈AWS CodeDeploy使用應用程式〉

對應的 UI：配置選項卡/CodeDeploy 應用程序

codedeploy-deployment-group

(*ECSDeployAction*/Configuration/codedeploy-deployment-group)

（如果包含`codedeploy-appspec`，則為必填）

指定現有 CodeDeploy 部署群組的名稱。若要取得有關 CodeDeploy 部署群組的更多資訊，請參閱《使用指南》 CodeDeploy中的〈AWS CodeDeploy使用部署群組〉。

對應的 UI：組態索引標籤/CodeDeploy 部署群組

codedeploy-deployment-description

(*ECSDeployAction*/Configuration/codedeploy-deployment-description)

(選用)

指定此動作將建立之部署的描述。若要取得更多資訊，請參閱《使用指南》 CodeDeploy中的〈AWS CodeDeploy使用部署〉。

對應的 UI：組態索引標籤/CodeDeploy 部署說明

「部署至 Kubernetes 叢集」動作參考

以下是「部署至 Kubernetes」叢集動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱新增「部署至 Kubernetes 叢集」動作。

> **Note**
>
> 後續的大多數 YAML 屬性在視覺化編輯器中都有對應的 UI 元素。若要查詢使用者介面元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployToKubernetesCluster_nn:
    Identifier: aws/kubernetes-deploy@v1
    DependsOn:
      - build-action
    Compute:
        - Type: EC2 | Lambda
        - Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployToEKS
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - manifest-artifact
    Configuration:
      Namespace: namespace
      Region: us-east-1
      Cluster: eks-cluster
      Manifests: manifest-path
```

DeployToKubernetesCluster

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：DeployToKubernetesCluster_nn。

對應的 UI：組態索引標籤/動作顯示名稱

Identifier

(*DeployToKubernetesCluster*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/kubernetes-deploy@v1。

對應的使用者介面：工作流程圖/DeployToKubernetesCluster_nn/ aws/kubernetes-部署 @v1 標籤

DependsOn

(*DeployToKubernetesCluster*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*DeployToKubernetesCluster*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*DeployToKubernetesCluster*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  針對動作執行期間的彈性進行優化
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  最佳化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的使用者介面：組態索引標籤/進階-選用/運算類型

Fleet

(*DeployToKubernetesCluster*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：Linux.x86-64.Large、Linux.x86-64.XLarge。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如Fleet果省略，預設值為Linux.x86-64.Large。

對應的使用者介面：組態索引標籤/進階-選用/運算叢集

Timeout

(*DeployToKubernetesCluster*/Timeout)

(選用)

指定動作在 CodeCatalyst 結束動作之前可執行的時間長度 (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Environment

(*DeployToKubernetesCluster*/Environment)

(必要)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Name

(*DeployToKubernetesCluster*/Environment/Name)

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

(*DeployToKubernetesCluster*/Environment/Connections)

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與環境建立關聯的資訊，請參閱建立環境。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Name

(*DeployToKubernetesCluster*/Environment/Connections/Name)

(必要)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Role

(*DeployToKubernetesCluster*/Environment/Connections/Role)

(必要)

指定「部署到 Kubernetes」叢集動作用來存取的 IAM 角色名稱。AWS請確定此角色包含下列原則：

- 下列權限原則：

> **Warning**
> 將權限限制為以下策略中顯示的權限。使用具有更廣泛權限的角色可能會造成安全風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
> 第一次使用角色時，請在資源策略陳述式中使用下列萬用字元，然後在可用資源名稱之後將策略範圍縮減。
>
> ```
> "Resource": "*"
> ```

- 下列自訂信任原則：

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

```
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

請確定已將此角色新增至：

- 您的帳戶連接。若要進一步了解如何將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。
- 你的庫伯尼特 ConfigMap人 若要進一步了解如何將 IAM 角色新增至 ConfigMap，請參閱`eksctl`文件中的管理 IAM 使用者和角色。

> **Tip**
>
> 另請參閱以取教學課程：將應用程式部署到 Amazon EKS得將 am IAM 角色新增至帳戶連線和 ConfigMap.

> **Note**
>
> 您可以在此處指定`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色的名稱，如果您想要的話。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***`spaceName`***角色。瞭解`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色具有非常廣泛的權限，可能會造成安全風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。如果您決定使用該`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*角色，請確保按照`eksctl`文件中管理 IAM 使用者和角色中的指示將其新增到您的 ConfigMap 檔案中。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Inputs

(*DeployToKubernetesCluster*/Inputs)

(選用)

本Inputs節定義工作流程執行期間所DeployToKubernetesCluster需的資料。

> **Note**
>
> 每個「部署到 Amazon EKS」動作只允許一個輸入 (來源或成品)。

對應的 UI：輸入索引標籤

Sources

(*DeployToKubernetesCluster*/Inputs/Sources)

（如果您的清單文件存儲在源存儲庫中，則需要）

如果您的 Kubernetes 資訊清單檔案儲存在來源儲存庫中，請指定該來源儲存庫的標籤。目前，唯一支援的標籤是WorkflowSource。

如果您的資訊清單檔案不包含在來源儲存庫中，則它們必須位於由其他動作產生的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的用戶界面：輸入選項卡/源- 可選

Artifacts - input

(*DeployToKubernetesCluster*/Inputs/Artifacts)

（如果您的清單文件存儲在上一個操作的輸出成品中，則需要）

如果 Kubernetes 資訊清單檔案包含在先前動作所產生的成品中，請在此指定該成品。如果您的資訊清單檔案不包含在成品中，則它們必須位於來源儲存庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：組態索引標籤/成品- 選用

Configuration

(*DeployToKubernetesCluster*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

Namespace

(*DeployToKubernetesCluster*/Configuration/Namespace)

(選用)

指定要在其中部署您的 Kubernetes 應用程式的 Kubernetes 命名空間。`default`如果您沒有在叢集中使用命名空間，請使用此選項。如需有關命名空間的詳細資訊，請參閱 Kubernetes 說明文件中的使用 Kubernetes 命名空間細分叢集。

如果您省略命名空間，則會使`default`用的值。

對應的 UI：配置選項卡/命名空間

Region

(*DeployToKubernetesCluster*/Configuration/Region)

(必要)

指定 Amazon EKS 叢集和服務所在的AWS區域。如需區域代碼的清單，請參閱 AWS 一般參考.

對應的 UI：組態索引標籤/區域

Cluster

(*DeployToKubernetesCluster*/Configuration/Cluster)

(必要)

指定現有 Amazon EKS 叢集的名稱。「部署到 Kubernetes」叢集動作會將容器化應用程式部署到此叢集中。如需 Amazon EKS 叢集的詳細資訊，請參閱 Amazon EKS 使用者指南中的叢集。

對應的 UI：配置選項卡/集群

Manifests

(*DeployToKubernetesCluster*/Configuration/Manifests)

(必要)

在 Kubernetes 說明文件中指定 YAML 格式的 Kubernetes 資訊清單檔案的路徑，這些檔案稱為組態檔、組態檔或簡稱組態。

如果您使用多個清單文件，請將它們放在單個文件夾中並引用該文件夾。資訊清單檔案由 Kubernetes 以字母數字方式處理，因此請務必在檔案名稱前面加上數字或字母增加，以控制處理順序。例如：

```
00-namespace.yaml
```

```
01-deployment.yaml
```

如果您的資訊清單檔案位於來源儲存庫中，則路徑會相對於來源儲存庫根資料夾。如果檔案位於先前工作流程動作的人工因素中，則路徑會相對於人工因素根資料夾。

範例：

```
Manifests/
```

```
deployment.yaml
```

```
my-deployment.yml
```

請勿使用萬用字元 (*)。

> **Note**
>
> 不支援頭盔圖表和組合化檔案。

如需有關資訊清單檔案的詳細資訊，請參閱 Kubernetes 文件中的組織資源組態。

對應的用戶界面：配置選項卡/清單

「動GitHub 作」動作參考

以下是動作動作的動GitHub作定義 YAML 參考。

在下列程式碼中選擇 YAML 屬性，以查看是否有描述。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢使用者介面元素，請使用 Ctrl+F。 該元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier:  aws/github-actions-runner@v1
    DependsOn:
      - dependent-action-name-1
    Compute:
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
```

```
        Value: variable-value-2
    Outputs:
      Artifacts:
        - Name: output-artifact-1
          Files:
            - github-output/artifact-1.jar
            - "github-output/build*"
        - Name: output-artifact-2
          Files:
            - github-output/artifact-2.1.jar
            - github-output/artifact-2.2.jar
      Variables:
        - variable-name-1
        - variable-name-2
      AutoDiscoverReports:
        Enabled: true | false
        ReportNamePrefix: AutoDiscovered
        IncludePaths:
          - "**/*"
        ExcludePaths:
          - node_modules/cdk/junit.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
            Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
            Number: whole-number
      Reports:
        report-name-1:
          Format: format
          IncludePaths:
            - "*.xml"
          ExcludePaths:
            - report2.xml
            - report3.xml
          SuccessCriteria:
            PassRate: percent
            LineCoverage: percent
            BranchCoverage: percent
            Vulnerabilities:
              Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
              Number: whole-number
    Configuration
```

```
    Steps:
      - github-actions-code
```

動作名稱

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

UI######/####

Identifier

(*####*/) Identifier

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

用aws/github-actions-runner@v1於GitHub動作動作。

對應的使用者介面：工作流程圖/***####/aw*** s/ @v1 標籤 github-actions-runner

DependsOn

(*####*/) DependsOn

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需有關「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*####*/) Compute

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Fleet

（ *#### /Compute/* ***Fleet*** ）

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如`Fleet`果省略，預設值為`Linux.x86-64.Large`。

對應的 UI：配置選項卡/計算機群- 可選

Timeout

（ *####*/ ）Timeout

(選用)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Environment

（ *####*/ ）Environment

(選用)

指定要與動作搭配使用的 CodeCatalyst 環境。

若要取得有關環境的更多資訊，請參閱使用環境和建立環境。

對應的 UI：配置選項卡/環境/帳戶/角色

Name

（*#### /Environment/* ***Name*** ）

（如果包含Environment，則為必填）

指定您要與動作相關聯的現有環境名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/環境

Connections

（*#### /Environment/* ***Connections*** ）

（如果包含Environment，則為必填）

指定要與動作相關聯的帳號連線。您最多可以在下指定一個帳戶連線`Environment`。

如需有關帳戶連線的詳細資訊，請參閱管理 AWS 帳戶 空間。如需如何將帳戶連線與您的環境建立關聯的資訊，請參閱建立環境。

對應的用戶界面：無

Name

（*#### /Environment/Connections/* ***Name*** ）

(選用)

指定帳戶連線的名稱。

對應的 UI：配置選項卡/「環境/帳戶/角色」/帳AWS戶連接

Role

（*#### /Environment/Connections/* ***Role*** ）

(選用)

指定此動作用於在 Amazon S3 和 Amazon ECR 等AWS服務中存取和操作的 IAM 角色名稱。確保此角色已添加到您的帳戶連接中。若要將 IAM 角色新增至帳戶連線，請參閱將 IAM 角色新增至帳戶連線。

> **Note**
>
> 您可以在此處指定角色的名稱，前提是CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有足夠的權限。如需有關此角色的詳細資訊，請參閱 為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。瞭解CodeCatalystWorkflowDevelopmentRole-*spaceName*角色具有非常廣泛的權限，可能會造成安全性風險。我們建議您只在不太擔心安全性的教學課程和案例中使用此角色。

> **Warning**
>
> 將權限限制為「GitHub 動作」動作所需的權限。使用具有更廣泛權限的角色可能會造成安全風險。

對應的 UI：組態索引標籤/「環境/帳戶/角色」/角色

Inputs

（*####*/）Inputs

(選用)

本Inputs節定義工作流程執行期間動作所需的資料。

> **Note**
>
> 每個 GitHub 「動作」動作最多允許四個輸入 (一個來源和三個成品)。變數不會計入此總數。

如果您需要引用駐留在不同輸入中的文件（例如源和工件），則源輸入是主輸入，並且工件是輔助輸入。輔助輸入中的文件的引用採用特殊前綴，以將它們從主輸入中剔除。如需詳細資訊，請參閱 範例：參考多個成品中的檔案。

對應的 UI：輸入索引標籤

Sources

（*#### /Inputs/* ***Sources***）

(選用)

指定代表動作所需之來源儲存庫的標籤。目前唯一支援的標籤是WorkflowSource，它代表儲存工作流程定義檔案的來源儲存庫。

如果您省略來源，則必須在下指定至少一個輸入成品*action-name*/Inputs/Artifacts。

如需來源的詳細資訊，請參閱 使用來源。

對應的用戶界面：輸入選項卡/源- 可選

Artifacts - input

(*#### /Inputs/* ***Artifacts***)

(選用)

指定您要提供作為此動作輸入的先前動作的人工因素。在先前動作中，必須將這些人工因素定義為輸出人工因素。

如果您未指定任何輸入人工因素，則必須在下指定至少一個來源儲存庫*action-name*/Inputs/Sources。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

> **Note**
>
> 如果無法使用「成品-選用」下拉式清單 (視覺化編輯器)，或者在驗證 YAML (YAML 編輯器) 時發生錯誤，可能是因為動作僅支援一個輸入。在此情況下，請嘗試移除來源輸入。

對應的 UI：輸入選項卡/加工品- 可選

Variables - input

(*#### /Inputs/* ***Variables***)

(選用)

指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的用戶界面：輸入選項卡/變量- 可選

Outputs

(*####*/) Outputs

(選用)

定義在工作流程執行期間由動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts - output

(*#### /Outputs/* ***Artifacts***)

(選用)

指定動作所產生的人工因素名稱。Artifact 名稱在工作流程中必須是唯一的，且僅限於英數字元 (a-z、A-Z、0-9) 和底線 (_)。不允許使用空格、連字號 (-) 和其他特殊字元。您無法使用引號來啟用輸出人工因素名稱中的空格、連字號和其他特殊字元。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/成品

Name

(*#### /Outputs/Artifacts/* ***Name***)

(如果包含Artifacts - output，則為必填)

指定動作所產生的人工因素名稱。Artifact 名稱在工作流程中必須是唯一的，且僅限於英數字元 (a-z、A-Z、0-9) 和底線 (_)。不允許使用空格、連字號 (-) 和其他特殊字元。您無法使用引號來啟用輸出人工因素名稱中的空格、連字號和其他特殊字元。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的使用者介面：輸出索引標籤/人工因素/新增人工因素名稱

Files

(*#### /Outputs/Artifacts/* ***Files***)

（如果包含Artifacts - output，則為必填）

指定動作輸出的加工品中 CodeCatalyst 包含的檔案。這些檔案會在工作流程動作執行時由工作流程動作產生，也可在來源存放庫中使用。檔案路徑可以位於來源儲存庫或上一個動作的人工因素中，且相對於來源儲存庫或人工因素根目錄。您可以使用全域模式來指定路徑。範例：

- 若要指定位於組建位置或來源存放庫位置根目錄中的單一檔案，請使用`my-file.jar`。
- 若要在子目錄中指定單一檔案，請使用`directory/my-file.jar`或`directory/subdirectory/my-file.jar`。
- 若要指定所有檔案，請使用`"**/*"`。**glob 模式指示匹配任意數量的子目錄。
- 若要指定名為的目錄中的所有檔案和目錄`directory`，請使用`"directory/**/*"`。**glob 模式指示匹配任意數量的子目錄。
- 若要指定名為的目錄中的所有檔案`directory`，但不指定其任何子目錄中的檔案，請使用`"directory/*"`。

> **Note**
>
> 如果檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

> **Note**
>
> 您可能需要在檔案路徑中新增前置詞，以指出要在其中尋找的成品或來源。如需詳細資訊，請參閱 參考來源儲存庫中的檔案 及 參考人工因素中的檔案。

對應的 UI：輸出選項卡/人造物/添加人造物/構建生成的文件

Variables - output

（*#### /Outputs/* ***Variables*** ）

(選用)

指定您要匯出動作的變數，以便後續動作可使用這些變數。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸出選項卡/變量/添加變量

變量名 -1

（ ***####### -1***/Outputs/Variables ）

(選用)

指定您要匯出動作的變數名稱。此變數必須已在相同動作的`Inputs`或`Steps`區段中定義。

如需有關變數的更多資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸出選項卡/變量/添加變量/名稱

AutoDiscoverReports

（ *#### /Outputs/* ***AutoDiscoverReports*** ）

(選用)

定義自動探索功能的組態。

當您啟用自動探索時，會 CodeCatalyst 搜尋所有`Inputs`傳入動作的內容，以及動作本身產生的所有檔案，以尋找測試、程式碼涵蓋範圍和軟體組成分析 (SCA) 報告。針對找到的每個報表，將其 CodeCatalyst 轉換為 CodeCatalyst 報表。CodeCatalyst 報告是完全集成到 CodeCatalyst 服務中的報告，可以通過 CodeCatalyst 控制台查看和操作。

> **Note**
>
> 根據預設，自動探索功能會檢查所有檔案。您可以使用IncludePaths或ExcludePaths性質限制要檢查哪些檔案。

對應的用戶界面：無

Enabled

（ *#### /Outputs/AutoDiscoverReports/* ***Enabled*** ）

(選用)

啟用或停用自動探索功能。

有效值為 `true` 或 `false`。

如`Enabled`果省略，預設值為`true`。

對應的 UI：輸出選項卡/報告/自動發現報告

ReportNamePrefix

（*`####`* */Outputs/AutoDiscoverReports/* ***`ReportNamePrefix`*** ）

(如果已包含並啟用，則AutoDiscoverReports為必要)

指定在找到的所有報告前面 CodeCatalyst 加上的首碼，以便命名其關聯 CodeCatalyst 的報告。例如，如果您指定的前置詞`AutoDiscovered`，並 CodeCatalyst自動探索兩個測試報告，`TestSuiteOne.xml`然後`TestSuiteTwo.xml`，關聯的 CodeCatalyst 報告將命名為`AutoDiscoveredTestSuiteOne`和。`AutoDiscoveredTestSuiteTwo`

對應的 UI：輸出選項卡/報告/自動發現報告/報告前綴

IncludePaths

（*`####`* */Outputs/AutoDiscoverReports/* ***`IncludePaths`*** ）

或

（*`####/Outputs/Reports/##-## -1/`* ）IncludePaths

(如果AutoDiscoverReports已包含並啟用，或如果已包含，則Reports為必要)

指定搜尋原始報告時 CodeCatalyst 包含的檔案和檔案路徑。例如，如果您指定`"/test/report/*"`，會 CodeCatalyst 搜尋尋找`/test/report/*`目錄的動作所使用的整個組建映像。當它找到該目錄時， CodeCatalyst 會在該目錄中尋找報表。

> **Note**
>
> 如果您的檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

如果省略此性質，預設值為`"**/*"`，表示搜尋會包括所有路徑上的所有檔案。

> **Note**
>
> 對於手動設定的報告，IncludePaths必須是符合單一檔案的 glob 模式。

對應的介面：

- 輸出標籤/報表/自動探索報表 /'包含/排除路徑/ 包含路徑
- ***######/##/######/#### -1 /' ##/#### '/####***

ExcludePaths

(*#### /Outputs/AutoDiscoverReports/* ***ExcludePaths***)

或

(*####/Outputs/Reports/##-## -1/*) ExcludePaths

(選用)

指定搜尋原始報告時 CodeCatalyst 排除的檔案和檔案路徑。例如，如果您指定"/test/my-reports/**/*"， CodeCatalyst 將不會搜尋/test/my-reports/目錄中的檔案。要忽略目錄中的所有文件，請使用 **/* glob 模式。

> **Note**
>
> 如果您的檔案路徑包含一或多個星號 (*) 或其他特殊字元，請以雙引號 () 括住路徑。""如需特殊字元的詳細資訊，請參閱語法指南和慣例。

對應的介面：

- 輸出選項卡/報告/自動發現報告/「包含/排除路徑」/「排除路徑」
- ***######/##/######/#### -1 /' ##/#### '/####***

SuccessCriteria

(*#### /Outputs/AutoDiscoverReports/* ***SuccessCriteria***)

或

(*####/Outputs/Reports/##-## -1/*) SuccessCriteria

(選用)

指定測試、程式碼涵蓋範圍、軟體組成分析 (SCA) 和靜態分析 (SA) 報告的成功準則。

如需詳細資訊，請參閱 設定報告的成功準則。

對應的介面：

- 輸出標籤/報告/自動探索報告/成功條件
- ***[##] ####/##/######/#### -1/####***

PassRate

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***PassRate***)

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/****PassRate***

(選用)

指定測試報告中必須通過的測試百分比，這些測試百分比才能標記為通過的關聯 CodeCatalyst報告。有效值包括十進位數字。例如：50、60.5。合格率標準僅適用於測試報告。如需測試報告的詳細資訊，請參閱測試報告。

對應的介面：

- 輸出標籤/報告/自動探索報告/成功標準/合格率
- ***[##] ####/##/######/#### -1 /####/###***

LineCoverage

(*#### /Outputs/AutoDiscoverReports/SuccessCriteria/* ***LineCoverage***)

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**LineCoverage***

(選用)

指定程式碼涵蓋範圍報告中必須涵蓋的行百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。明細行涵蓋範圍條件僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

對應的介面：

- 輸出標籤/報告/自動探索報告/成功標準/線路覆蓋
- ***[##] ####/##/######/#### -1 /####/####***

BranchCoverage

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/ **BranchCoverage***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**BranchCoverage***

(選用)

指定程式碼涵蓋範圍報告中必須涵蓋的分支百分比，相關 CodeCatalyst 報表才會標示為已通過。有效值包括十進位數字。例如：50、60.5。分支涵蓋範圍標準僅適用於程式碼涵蓋範圍報告。如需程式碼涵蓋範圍報告的詳細資訊，請參閱代碼覆蓋率報告。

對應的介面：

- 輸出標籤/報告/自動探索報告/成功標準/分支覆蓋
- ***####/##/######/#### -1 /####/####***

Vulnerabilities

（*#### /Outputs/AutoDiscoverReports/SuccessCriteria/ **Vulnerabilities***）

或

*#######-## -1/Outputs/Reports/#/SuccessCriteria/**Vulnerabilities***

(選用)

針對要標記為通過的相關 CodeCatalyst 報告，指定 SCA 報告中允許的弱點數目上限和嚴重性。若要指定弱點，您必須指定：

- 您要包含在計數中的弱點的最低嚴重性。從最嚴重到最不嚴重的有效值為：CRITICAL、、HIGH、MEDIUM、LOW、、INFORMATIONAL。

  例如，如果你選擇HIGH，然後HIGH和CRITICAL漏洞將被統計。
- 您要允許之指定嚴重性的弱點數目上限。超過此數目會導致 CodeCatalyst 報告標記為失敗。有效值為整數。

弱點準則僅適用於 SCA 報告。如需有關 SCA 報告的詳細資訊，請參閱軟件成分分析報告。

若要指定最低嚴重性，請使用Severity內容。若要指定弱點數目上限，請使用Number內容。

如需有關 SCA 報告的詳細資訊，請參閱測試報告類型。

對應的介面：

- 輸出標籤/報告/自動探索報告/成功條件/漏洞
- ***[##] ####/##/######/#### -1 /####/##***

Reports

（*#### /Outputs/* ***Reports***）

(選用)

指定測試報告之組態的段落。

對應的 UI：輸出選項卡/報告

報告名稱 -1

（動作名稱報***#-## -1***/Outputs/Reports/）

（如果包含Reports，則為必填）

您要提供給將從原始 CodeCatalyst 報表產生之報表的名稱。

對應的 UI：輸出選項卡/報告/手動配置報告/報告名稱

Format

（*####/Outputs/Reports/##-## -1/*）Format

（如果包含Reports，則為必填）

指定報告所使用的檔案格式。可能的值如下。

- 對於測試報告：
  - 對於「黃瓜 JSON」，請指定「黃瓜」(視覺化編輯器) 或 CUCUMBERJSON (YAML 編輯器)。
  - 對於 JUnit XML，請指定 JUnit（可視化編輯器）或JUNITXML（YAML 編輯器）。
  - 對於 NUnit XML，請指定 NUnit (視覺化編輯器) 或 NUNITXML (YAML 編輯器)。
  - 對於 NUnit 3 XML，請指定 nUnit3 (視覺化編輯器) 或 NUNIT3XML (YAML 編輯器)。
  - 若為視覺工作室 TRX，請指定視覺工作室 TRX (視覺化編輯器) 或 VISUALSTUDIOTRX (YAML 編輯器)。
  - 對於 TestNG 的 XML，請指定 TestNG（可視化編輯器）或TESTNGXML（YAML 編輯器）。
- 對於代碼覆蓋率報告：
  - 針對四葉草 XML，請指定四葉草 (視覺化編輯器) 或 CLOVERXML (YAML 編輯器)。
  - 對於 XML 編輯器，指定編輯器（可視化編輯器）或**COBERTURAXML**（YAML 編輯器）。
  - 對於 JaCoCo XML，請指定 JaCoCo(視覺化編輯器) 或 JACOCOXML (YAML 編輯器)。
  - 對於簡單生成的 SimpleCov JSON，而不是簡單的 co v-json，請指定簡單的編輯器（可視化編輯器）或（YAML 編輯器）。SIMPLECOV
- 對於軟件成分分析（SCA）報告：
  - 針對 SARIF，請指定 SARIF (視覺化編輯器) 或 SARIFSCA (YAML 編輯器)。

########:####/##/######/####/#### -1/#########

Configuration

（*####/*）Configuration

(必要) 您可以在其中定義動作的組態特性的區段。

對應的 UI：組態索引標籤

Steps

(*#### /Configuration/* ***Steps***)

(必要)

指定您在 GitHub Marketplace 中 GitHub 動作詳細資訊頁面上顯示的動作程式碼。按照以下準則添加代碼：

1. 將「 GitHub 動作`steps:`」區段中的程式碼貼到工作 CodeCatalyst 流程的`Steps:`區段中。程式碼以破折號 (-) 開頭，看起來與下列類似。

   GitHub 要粘貼的代碼：

   ```
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: master
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

2. 檢閱您剛貼上的程式碼，並視需要加以修改，使其符合 CodeCatalyst標準。例如，使用前面的程式碼區塊時，您可以移除`###`體的程式碼，並以粗體加入程式碼。

   CodeCatalyst 工作流程:

   ```
   Steps:
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: false
          DEFAULT_BRANCH: mastermain
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

3. 對於 GitHub Action 中包含但不存在於`steps:`區段內的其他程式碼，請使用 CodeCatalyst相等程式碼將其新增至工作 CodeCatalyst 流程。您可以檢閱工作流定義參考以深入瞭解如何將 GitHub 程式碼移植到 CodeCatalyst。詳細的移轉步驟不在本指南的範圍內。

以下是如何在 GitHub 「動作」動作中指定檔案路徑的範例：

```
Steps:
```

```
  - name: Lint Code Base
    uses: github/super-linter@v4
    ...
  - run: cd /sources/WorkflowSource/MyFolder/  && cat file.txt
  - run: cd /artifacts/MyGitHubAction/MyArtifact/MyFolder/  && cat file2.txt
```

若要取得有關指定檔案路徑的更多資訊，請參閱參考來源儲存庫中的檔案和參考人工因素中的檔案。

對應的使用者介面：組態標籤/GitHub 動作 YAML

「渲染 Amazon ECS 任務定義」動作參考

以下是彩現 Amazon ECS 任務定義動作的動作定義 YAML 參考。若要瞭解如何使用此動作，請參閱新增「渲染 Amazon ECS 任務定義」動作。

> **Note**
>
> 接下來的大多數 YAML 屬性在可視化編輯器中都有對應的 UI 元素。若要查詢 UI 元素，請使用 Ctrl+F。 元素將與其關聯的 YAML 屬性列出。

```
# The workflow definition starts here.
# See #### for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSRenderTaskDefinition_nn:
    Identifier: aws/ecs-render-task-definition@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
```

```
        - task-definition-artifact
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Configuration
      task-definition: task-definition-path
      container-definition-name: container-definition-name
      image: docker-image-name
      environment-variables:
        - variable-name-1=variable-value-1
        - variable-name-2=variable-value-2
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files: "task-definition*"
      Variables:
        - task-definition
```

ECSRenderTaskDefinition

(必要)

指定動作的名稱。所有動作名稱在工作流程中都必須是唯一的。動作名稱僅限於英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用動作名稱中的特殊字元和空格。

預設：ECSRenderTaskDefinition_nn。

對應的 UI：組態索引標籤/動作名稱

Identifier

(*ECSRenderTaskDefinition*/Identifier)

(必要)

識別動作。除非您要變更版本，否則請勿變更此屬性。如需詳細資訊，請參閱 使用動作版本。

預設：aws/ecs-render-task-definition@v1。

對應的使用者介面：工作流程圖/ECSRenderTaskDefinition_nn/ aws/ @v1 標籤 ecs-render-task-definition

DependsOn

(*ECSRenderTaskDefinition*/DependsOn)

(選用)

指定必須順利執行才能執行此動作的動作或動作群組。

如需「依賴」功能的詳細資訊，請參閱。將動作配置為依賴其他動作

對應的用戶界面：輸入選項卡/取決於- 可選

Compute

(*ECSRenderTaskDefinition*/Compute)

(選用)

用來執行工作流程動作的計算引擎。您可以在工作流程層級或動作層級指定計算，但不能同時指定兩者。在工作流程層級指定時，計算組態會套用至工作流程中定義的所有動作。在工作流程層級，您也可以在相同的執行個體上執行多個動作。如需詳細資訊，請參閱 跨動作共用運算。

對應的用戶界面：無

Type

(*ECSRenderTaskDefinition*/Compute/Type)

（如果包含Compute，則為必填）

運算引擎的類型。您可以使用下列其中一個值：

- EC2 (視覺化編輯器) 或 EC2 (YAML 編輯器)

  優化了動作運行期間的靈活性。
- Lambda（可視化編輯器）或Lambda（YAML 編輯器）

  優化動作啟動速度。

如需運算類型的更多相關資訊，請參閱 關於運算類型。

對應的 UI：配置選項卡/計算類型

Fleet

(*ECSRenderTaskDefinition*/Compute/Fleet)

(選用)

指定將執行工作流程或工作流程動作的機器或叢集。對於隨需叢集，當動作開始時，工作流程會佈建所需的資源，並在動作完成時銷毀機器。隨選艦隊的範例：`Linux.x86-64.Large`、`Linux.x86-64.XLarge`。如需隨選叢集的詳細資訊，請參閱隨選叢集屬性。

透過佈建的叢集，您可以設定一組專用機器來執行工作流程動作。這些機器保持閒置狀態，可立即處理動作。如需已佈建叢集的詳細資訊，請參閱佈建的叢集屬性。

如`Fleet`果省略，預設值為`Linux.x86-64.Large`。

對應的 UI：組態索引標籤/運算叢集

Timeout

(*ECSRenderTaskDefinition*/Timeout)

(選用)

指定動作在 CodeCatalyst 結束動作之前可以執行的時間量 (以分鐘為單位) (YAML 編輯器) 或小時和分鐘 (視覺化編輯器)。最小值為 5 分鐘，最大值在中說明中工作流程的配額 CodeCatalyst。預設逾時與逾時上限相同。

對應的 UI：配置選項卡/超時- 可選

Inputs

(*ECSRenderTaskDefinition*/Inputs)

(選用)

本`Inputs`節定義了工作流程執行期間所`ECSRenderTaskDefinition`需的資料。

> **Note**
>
> 每個渲染 Amazon ECS 任務定義動作只允許一個輸入 (來源或成品)。變數不會計入此總數。

對應的 UI：輸入索引標籤

Sources

(*ECSRenderTaskDefinition*/Inputs/Sources)

(如果您的任務定義檔案儲存在來源儲存庫中，則需要)

如果您的任務定義檔案儲存在來源儲存庫中，請指定該來源儲存庫的標籤。目前，唯一支援的標籤是`WorkflowSource`。

如果您的任務定義檔案不包含在來源儲存庫中，則該檔案必須位於由其他動作產生的成品中。

如需來源的詳細資訊，請參閱 使用來源。

對應的 UI：輸入選項卡/源- 可選

Artifacts - input

(*ECSRenderTaskDefinition*/Inputs/Artifacts)

(如果您的任務定義檔案儲存在上一個動作的輸出成品中，則為必要)

如果您要部署的任務定義檔案包含在先前動作所產生的成品中，請在此指定該成品。如果您的任務定義檔案不包含在成品中，則該檔案必須位於來源儲存庫中。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：配置選項卡/成品- 可選

Variables - input

(*ECSRenderTaskDefinition*/Inputs/Variables)

(必要)

指定一系列名稱/值配對，這些配對定義您要讓動作可用的輸入變數。變數名稱限制為英數字元 (a-z、A-Z、0-9)、連字號 (-) 和底線 (_)。不允許空格。您無法使用引號來啟用變數名稱中的特殊字元和空格。

如需變數的詳細資訊 (包括範例)，請參閱使用變數。

對應的 UI：輸入選項卡/變量- 可選

Configuration

(*ECSRenderTaskDefinition*/Configuration)

(必要)

您可以在其中定義動作的組態屬性的區段。

對應的 UI：組態索引標籤

task-definition

(*ECSRenderTaskDefinition*/Configuration/task-definition)

(必要)

指定既有工作定義檔案的路徑。如果檔案位於來源儲存庫中，則路徑會相對於來源儲存庫根資料夾。如果您的檔案位於先前工作流程動作的人工因素中，則路徑會相對於人工因素根資料夾。如需有關任務定義檔案的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的任務定義。

對應的 UI：配置選項卡/任務定義

container-definition-name

(*ECSRenderTaskDefinition*/Configuration/container-definition-name)

(必要)

指定要在其中執行 Docker 映像檔的容器名稱。您可以在任務定義檔案的`containerDefinitions`、`name`欄位中找到此名稱。如需詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的名稱。

對應的 UI：配置選項卡/容器名稱

image

(*ECSRenderTaskDefinition*/Configuration/image)

(必要)

指定您希望彩現 Amazon ECS 任務定義動作新增至任務定義檔案的 Docker 映像名稱。動作會將此名稱新增至任務定義檔案中的containerDefinitions、image欄位。如果image欄位中已存在值，則動作會覆寫該值。您可以在影像名稱中包含變數。

範例：

如果您指定MyDockerImage:${WorkflowSource.CommitId}，動作*## #MyDockerImage:commit-id########## commit-id* 是工作流程在執行階段產生的提交 ID。

#####my-ecr-repo/image-repo:$(date +%m-%d-%y-%H-%m-%s)###### my-ecr-repo/image-repo: ## +%m-%D-%H-%m-%s ############ Amazon ####### (ECR) # URI### # +%m-%d-%y-%H-%m-%s my-ecr-repo##################### month-day-year-hour-minute-second

如需有關image欄位的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的映像。如需變數的更多資訊，請參閱使用變數。

對應的 UI：配置選項卡/映像名稱

environment-variables

(*ECSRenderTaskDefinition*/Configuration/environment-variables)

(必要)

指定您希望彩現 Amazon ECS 任務定義動作新增至任務定義檔案的環境變數。此動作會將變數新增至任務定義檔案中的containerDefinitions、environment欄位。如果變數已存在於檔案中，動作會覆寫現有變數的值，並新增任何新變數。如需 Amazon ECS 環境變數的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的指定環境變數。

對應的 UI：配置選項卡/環境變量- 可選

Outputs

(*ECSRenderTaskDefinition*/Outputs)

(必要)

定義工作流程執行期間動作輸出的資料。

對應的 UI：輸出索引標籤

Artifacts

(*ECSRenderTaskDefinition*/Outputs/Artifacts)

(必要)

指定動作產生的人工因素。您可以參考這些成品做為其他動作中的輸入。

如需人工因素的詳細資訊 (包括範例)，請參閱使用人工因素。

對應的 UI：輸出選項卡/成品

Name

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Name)

(必要)

指定將包含更新之工作定義檔案的人工因素名稱。預設值為 `MyTaskDefinitionArtifact`。然後，您必須將此成品指定為「部署到 Amazon ECS」動作的輸入。若要瞭解如何將此成品新增為「部署至 Amazon ECS」動作的輸入，請參閱範例工作流。

對應的 UI：輸出選項卡/人造物/名稱

Files

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Files)

(必要)

指定要包含在人工因素中的檔案。您必須指定，`task-definition-*`以便將更新的任務定義檔案 (開`task-definition-`頭為) 包括在內。

對應的 UI：輸出選項卡/人造物/文件

Variables

(*ECSRenderTaskDefinition*/Outputs/Variables)

(必要)

指定要透過渲染動作設定的變數名稱。render 動作會將此變數的值設定為更新工作定義檔案的名稱 (例如，`task-definition-`*`random-string`*`.json`)。然後，您必須在「部署到 Amazon ECS」動作

的「任務定義」(視覺化編輯器) 或 `task-definition` (yaml 編輯器) 屬性中指定此變數。若要瞭解如何將此變數新增至「部署至 Amazon ECS」動作，請參閱範例工作流。

預設：`task-definition`

對應的 UI：輸出標籤/變數/名稱欄位

中工作流程的配額 CodeCatalyst

下表說明 Amazon 中工作流程的配額和限制 CodeCatalyst。

如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

每個空間的最大工作流程數	800
工作流程定義檔大小上限	256 KB
單一來源事件中處理的工作流程檔案數目上限	50
單一來源事件中處理的檔案數目上限	4,000
每個空間的作用中叢集數目上限	10
每個叢集的最大作用中運算執行個體數	20
每個動作的輸入成品數目上限	10
每個動作的輸出成品數目上限	10
單一動作輸出變數的最大總大小	120 KB
輸出變數值的最大長度	500 個字符或更多，具體取決於發出值的操作。 **Note** 如果值超出動作的限制，則值可能會被截斷。
在工作流程執行期間保留產生的人工因素的最大天數	30

每個動作的最大報告數	50
每個測試報告的最大測試用例數	20,000
每個程式碼涵蓋範圍報告的最大檔案數	20,000
每份報告的軟體組成分析結果數目上限	20,000
每個靜態分析報告的最大檔案數	20,000
每個空間同時執行的工作流程數目上限	100
每個工作流程的最大動作數	50
每個工作流程同時執行的動作數目上限	50
每個空間同時執行的動作數目上限	200
動作可以執行的時間上限	對於構建和測試操作，超時是 8 小時。 對於所有其他動作，逾時為 1 小時。
與 AWS 帳戶 每個空間相關聯的環境數目上限	5,000
每個動作的密碼數目上限	5
每個空間的最大密碼數	500,000

中的問題 CodeCatalyst

在中 CodeCatalyst，您可以監視專案中涉及的功能、錯誤和任何其他工作。每件作品都保存在稱為問題的獨特記錄中。您可以通過添加任務檢查清單將問題分解為較小的目標。每個問題都可以具有說明、工作負責人、狀態和其他屬性，您可以搜尋、分組和篩選。您可以使用預設檢視來檢視您的問題，也可以使用自訂篩選、排序或群組來建立自己的檢視。如需與問題相關概念的詳細資訊，請參閱問題, 概念。若要瞭解如何建立第一期刊，請參閱在中建立問題 CodeCatalyst。

以下是使用問題的團隊可能的工作流程：

豪爾赫·索薩是一個項目工作的開發人員。他和他的項目成員李娟、馬特奧·傑克遜和王秀蘭合作，確定需要做什麼工作。每天，他和他的開發人員都會舉行由王秀蘭領導的同步會議。他們通過導航到董事會的其中一個團隊視圖來拉起董事會。透過建立檢視，使用者和團隊可以儲存篩選器、分組和問題排序，以輕鬆檢視符合其指定條件的問題。他們的觀點包含按指定人分組的問題，並按優先級排序，以顯示每個開發人員最重要的問題和問題的狀態。由於豪爾赫被分配任務來完成，他通過為每個任務創建一個問題來計劃自己的工作。當創建問題, 豪爾赫可以選擇合適的狀態, 優先級, 和工作估算努力. 對於較大的問題，豪爾赫增加了任務的問題，打破工作成較小的目標。豪爾赫（Jorge）以草稿狀態創建了自己的問題，例如積壓，因為他不打算立即開始。處於草稿狀態的問題會顯示在「草稿」檢視中，以便對其進行規劃和排定優先順序。一旦豪爾赫準備開始工作，他通過將其狀態更新為另一個類別（未開始，已開始或已完成）的狀態，將相應的問題移至董事會。由於每個任務正在處理中，團隊可以按標題，狀態，工作負責人，標籤，優先級和估計進行過濾，以找到與指定參數匹配的特定問題或類似問題。使用董事會，豪爾赫和他的團隊可以查看每個問題已完成的任務數量，並通過將每個問題從一個狀態拖動到下一個狀態來跟踪 day-to-day 進度，直到任務完成。隨著專案的進行，已完成的問題會累積在「已完成」狀態。王秀蘭決定通過使用快速存檔按鈕將其從視圖中刪除，以便開發人員可以專注於與當前和即將到來的工作有關的問題。

在規劃他們的工作時，在項目上工作的開發人員選擇排序依據和分組依據，以查找他們想要從積壓轉移到董事會的問題。他們可能會根據最高優先順序的客戶要求，選擇將問題新增至主機板，因此他們會依客戶要求標籤將主機板分組，然後依優先順序排序。他們也可能按估計排序，以確保他們正在採取的工作量，他們可以實現. 項目經理 Sanvi Sarkar 定期審查和新郎積壓，以幫助確保優先級準確地反映每個問題對項目成功的重要性。

主題

- 問題, 概念
- 在中建立問題 CodeCatalyst
- 編輯和協作的問題 CodeCatalyst

- 尋找及檢視問題
- 匯出問題
- 設定問題設定
- 中的問題配額 CodeCatalyst

問題, 概念

建立問題是追蹤專案中正在完成工作的快速且有效率的方法。您可以使用問題來協助您討論日常同步會議中的工作、排定工作優先順序等等。

本頁包含可協助您有效使用中問題的概念清單 CodeCatalyst。

作用中問題

作用中問題是指任何未處於「草稿」狀態或已封存的問題。換句話說，「作用中問題」是指下列任一狀態類別中狀態的問題：「未開始」、「已啟動」和「已完成」。如需狀態和狀態類別的詳細資訊，請參閱狀態和狀態類別。

您可以從預設的「作用中問題」檢視，檢視專案中的所有作用中問題。

封存的問題

封存的問題是指與您的專案不再相關的問題。例如，如果問題已完成，而您不再需要在「完成」欄中看到問題，或者建立錯誤，您可以將問題封存起來。如果需要，可以取消封存的問題。

受讓人

受指派人是指派問題的人員。如果該人在您搜尋時沒有出現在清單中，表示他們尚未新增到您的專案中。若要加入它們，請參閱邀請使用者加入您的專案。若要啟用問題的多個指派對象，請參閱。啟用或停用多個受指派人多個受指派人的問題會以不同顏色的頭像出現在您的圖版上，每個人都代表其中一個受指派人。

自訂欄位

自訂欄位可讓您根據追蹤和維護專案中問題的需求，自訂問題的不同屬性。例如，您可以新增路線圖、特定到期日或請求者欄位的欄位。

估計

在敏捷開發中，估計被稱為故事點。除了問題的模糊性和複雜性之外，使用估計值來包含所需的工作量。針對風險、難度和未知數較高的問題，請使用較高的估算值。您可以變更專案的估算類型。若要取得有關估計類型及其設定方式的更多資訊，請參閱〈〉設定問題工作量估算。

問題

問題是追蹤與專案相關工作的記錄。您可以針對與專案相關的功能、工作、錯誤或任何其他工作主體建立問題。如果您正在使用敏捷開發，則問題還可以描述史詩或用戶故事。

標籤

該標籤用於分組，排序和過濾問題。您可以輸入新標示名稱，或從填入的清單中選擇其中一個標示。此清單包含專案中最近使用的標籤。一個問題可以有多個標籤，並且可以從問題中移除一個標籤。若要自訂標示，請參閱標籤。

優先順序

優先順序是指問題的重要程度。有四個選項：「低」、「中」、「高」和「無優先順序」。

狀態和狀態類別

狀態是問題的目前狀態，可用來快速檢查問題在生命週期 (從發生到完成) 的進度。所有問題都必須有一個狀態，且每個狀態都屬於某個狀態類別。狀態類別可用來協助組織您的狀態，並填入預設問題檢視。

中有五種默認狀態和四個狀態類別 CodeCatalyst。您可以建立其他狀態，但無法建立其他狀態類別。以下列表包含默認狀態及其狀態類別：「待處理（草稿）」、「待辦事項（未開始）」、「進行中」（「已開始」）、「審閱中（已開始）」和「完成（已完成）」。

如需使用狀態的詳細資訊，請參閱狀態。

任務

可以將任務添加到問題中，以進一步分解和組織該問題的工作。您可以在建立問題時將工作新增至問題，或將工作新增至現有問題。檢視問題時，您可以重新排序、移除問題，或將其工作標示為已完成。

檢視

CodeCatalyst 專案中的問題會顯示在視圖中。檢視可以是以清單格式顯示問題的網格檢視，也可以是以問題狀態組織的欄中的並排顯示問題的看板檢視。有四種預設檢視，您可以使用自訂分組、篩選和排序來建立自己的檢視。下列清單包含四個預設檢視的詳細資料。

- 草稿視圖是一個網格視圖，顯示當前未處理的問題。以「草稿」狀態類別中的狀態建立的任何問題都會顯示在此檢視中。專案團隊可以使用此檢視來查看仍在定義哪些問題，或正在等待指派和處理的問題。
- 「使用中問題」檢視是目前正在處理的所有問題的看板檢視。狀態為 [未開始]、[已啟動] 或 [已完成] 狀態類別的任何問題都會顯示在此檢視中。
- 「所有問題」視圖是一個網格視圖，展示專案中的所有問題，包括草稿和作用中的問題。
- 已封存檢視會顯示所有已封存的問題。

在中建立問題 CodeCatalyst

Amazon 的生成式 AI 功能 CodeCatalyst 正在預覽版中，可能會有所變更。它們僅在美國西部 (奧勒岡) 區域提供。生成式 AI 功能的使用方式因層級而異。如需詳細資訊，請參閱 定價。

開發團隊創建問題以幫助跟踪和管理他們的工作。您可以根據自己的需求在專案中建立問題。例如，您可以建立問題來追蹤程式碼中變數的更新。您可以將問題分配給項目中的其他用戶，使用標籤來幫助您跟踪工作等。

請按照下列指示在中建立問題 CodeCatalyst。

若要建立問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 瀏覽至您要建立問題的專案。
3. 在專案首頁上，選擇 [建立問題]。或者，在導覽窗格中，選擇「問題」。
4. 選擇 [建立問題]。

> **Note**
>
> 您也可以在使用網格檢視時內嵌新增問題。

5. 輸入問題的標題。
6. (選擇性) 輸入「說明」。您可以使用降價來添加格式。
7. (選擇性) 選擇問題的 [狀態]、[優先順序] 和 [估計]。如果專案的估計設定設定為 [隱藏預估值]，則不會有 [估計] 欄位。
8. (選擇性) 將工作新增至問題。任務可用於將問題的工作細分為較小的目標。若要新增工作，請選擇 [+ 新增工作]。然後，在文本字段中輸入任務名稱，然後按 Enter 鍵。新增工作後，您可以透過選擇核取方塊將其標示為已完成，或透過從核取方塊左側選擇並拖曳工作來重新排序工作。
9. (選擇性) 新增現有標籤或建立新標籤，然後選擇 + 新增標籤來加入標籤。
    a. 若要新增現有標示，請從清單中選擇標示。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有標籤。
    b. 若要建立新標籤並新增標籤，請在搜尋欄位中輸入要建立的標籤名稱，然後按 Enter 鍵。
10. (選擇性) 選擇 + 新增工作負責人以新增工作負責人。您可以選擇 + 加入我，快速將自己新增為受指派人。

> **Tip**
>
> 您可以選擇將問題分配給 Amazon Q，讓 Amazon Q 嘗試解決問題。如需詳細資訊，請參閱 教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作。
> 此功能需要為空間啟用生成 AI 功能。如需詳細資訊，請參閱管理生成 AI 功能。

11. (選擇性) 新增現有的自訂欄位或建立新的自訂欄位。問題可以有多個自定義字段。
    a. 若要新增現有的自訂欄位，請從清單中選擇自訂欄位。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有自訂欄位。
    b. 要創建一個新的自定義字段並添加它，請在搜索字段中輸入要創建的自定義字段的名稱，然後按 Enter 鍵。然後選擇要創建的自定義字段的類型並設置一個值。
12. 選擇 [建立問題]。右下角會顯示通知：如果問題已成功建立，便會顯示確認訊息，指出問題已成功建立。如果問題未成功建立，則會顯示錯誤訊息，其中包含失敗原因。然後，您可以選擇「重試」以編輯並重試建立問題，或選擇「捨棄」以捨棄問題。這兩個選項都會關閉通知。

> **Note**
>
> 您無法在建立提取要求時將其連結至問題。不過，您可以在建立後對其進行編輯，以新增提取請求的連結。

建立和處理指派給 Amazon Q 的問題時的最佳實務

當你創建問題，有時他們中的一些徘徊。造成這種情況的原因可能是複雜且可變的。有時候是因為目前尚不清楚誰應該對它進行工作。其他時候，這個問題需要對代碼庫的特定部分進行研究或專業知識，以及工作的最佳候選人正忙於其他問題。往往還有其他緊急工作必須先參加。任何或所有這些原因都可能導致無法解決的問題。 CodeCatalyst 包括與名為 Amazon Q 的生成 AI 助理整合，該助理可以根據問題的標題和說明分析問題。如果您將問題指派給 Amazon Q，它會嘗試建立草稿解決方案供您評估。這可以幫助您和您的團隊專注於需要注意的問題並優化工作，而 Amazon Q 則針對您沒有資源可立即解決的問題提供解決方案。

> **Note**
>
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於將問題指派給 Amazon Q 功能開發功能是建立在 Amazon 基岩上，因此使用者可以充分利用 Amazon 基岩中實作的控制項來執行人工智慧 (AI) 的安全性、安全性和負責任的使用。

Amazon Q 在簡單問題和簡單的問題上表現最佳。為了獲得最佳效果，請使用簡單的語言清楚地解釋您想要完成的工作。以下是一些最佳實務，可協助您建立針對 Amazon Q 進行最佳化的問題。

- 保持簡單。Amazon Q 最擅長進行簡單的程式碼變更和修正，這些修正可在問題的標題和說明中加以說明。不要指定含糊標題或過於華麗或矛盾描述的問題。
- 要具體。您可以提供解決問題所需的確切變更的資訊越多，Amazon Q 就越有可能建立解決問題的解決方案。如果可能，請包括特定的詳細信息，例如要更改的 API 名稱，要更新的方法，需要更改的測試以及您可以想到的任何其他詳細信息。
- 在將問題指派給 Amazon Q 之前，請確認問題的標題和說明中已包含所有詳細資訊。 將問題指派給 Amazon Q 後，就無法變更問題的標題或說明，因此在將問題指派給 Amazon Q 之前，請確定您已取得問題中所需的所有資訊。
- 僅在單一來源儲存庫中指派需要變更程式碼的問題。Amazon Q 只能在中處理單一來源儲存庫中的程式碼 CodeCatalyst。不支援連結的儲存庫。在將問題指派給 Amazon Q 之前，請確定問題只需要在單一來源儲存庫中進行變更。
- 使用 Amazon Q 建議的預設核准每個步驟。根據預設，Amazon Q 需要您核准所需的每個步驟。這使您不僅可以在問題上的註解中與 Amazon Q 互動，還可以在其建立的任何提取請求上與 Amazon Q 互動。這為 Amazon Q 提供更具互動性的體驗，可協助您調整其方法並優化其建立的程式碼以解決問題。

> **Note**
>
> Amazon Q 不會回應問題或提取請求中的個別註解，但會在系統要求重新考慮其方法或建立修訂時進行審核。

- 始終仔細查看 Amazon Q 提出的方法。 一旦您核准其方法，Amazon Q 就會開始根據該方法產生程式碼。在告訴 Amazon Q 繼續之前，請確保方法看起來正確，並包含您期望的所有詳細資訊。
- 如果您沒有現有的工作流程可在審核之前部署工作流程，請確保僅允許 Amazon Q 處理工作流程。您的專案可能已設定為在提取要求事件上開始執行的工作流程。如果是這樣，Amazon Q 建立的任何提取請求 (包括建立或更新工作流程 YAML) 都可能會開始執行提取請求中包含的工作流程。最佳做法是，請勿選擇允許 Amazon Q 處理工作流程檔案，除非您確定專案中沒有工作流程會自動執行這些工作流程，然後再檢閱並核准其建立的提取請求。

如需詳細資訊，請參閱教學課程：使用 CodeCatalyst 生成式 AI 功能加速開發工作和管理生成 AI 功能。

編輯和協作的問題 CodeCatalyst

內容

編輯問題

請依照下列步驟編輯問題的標題、說明、狀態、受指派人、優先順序、估計或標籤。

若要編輯問題

1. 選擇您要編輯的問題，以檢視問題詳細資料。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 若要編輯問題標題，請選擇標題、輸入新標題，然後按 Enter 鍵。
3. 若要編輯描述，請選擇描述、輸入新描述，然後按 Enter。您可以使用降價來添加格式。
4. 在 [工作] 中，您可以檢視和管理問題的工作。如需詳細資訊，請參閱 管理有關問題的工作。
5. 若要編輯 [狀態]、[估計] 或 [優先順序]，請從個別的下拉式功能表中選擇選項。
6. 在「標示」中，您可以加入既有標示、建立新標示或移除標示。
   a. 若要新增現有標籤，請選擇 [+ 新增標籤]，然後從清單中選擇標籤。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有標籤。
   b. 要創建新標籤並添加它，請選擇 + 添加標籤在搜索字段中輸入要創建的標籤的名稱，然後按 Enter 鍵。
   c. 若要移除標籤，請選擇您要移除的標籤旁邊的 X 圖示。如果您從所有問題中移除標籤，該標籤將出現在問題設定的「標籤」區段中的「未使用的標籤」區段中。使用篩選器或在問題中新增標籤時，未使用的標籤會出現在標籤清單的末尾。您可以在問題設定中找到所有標籤 (已使用和未使用) 及其問題的概觀。
7. 若要指派問題，請在「受指派人」區段中選擇 [+ 新增工作負責人]，然後搜尋並從清單中選擇受指派人。您可以選擇 [+ 加入我]，快速將自己新增為受指派人。
8. 在「附件」 中，您可以新增、下載或移除附件。如需詳細資訊，請參閱 使用附件。
9. 若要連結提取請求，請選擇「連結提取請求」，然後從清單中選擇提取請求，或輸入其 URL 或 ID。若要取消提取請求的連結，請選擇「取消連結」圖示。

> **Tip**
>
> 在您新增問題的提取要求連結之後，您可以在連結的提取要求清單中選擇該連結的 ID，以快速瀏覽至該連結。您可以使用提取請求的 URL 來連結與問題看板不同專案中的提取請求，但只有屬於該專案成員的使用者才能檢視或導覽至該提取要求。

10. (選擇性) 新增和設定現有的自訂欄位、建立新的自訂欄位或移除自訂欄位。問題可以有多個自定義字段。

a. 若要新增現有的自訂欄位，請從清單中選擇自訂欄位。您可以在欄位中輸入搜尋字詞，以搜尋專案中包含該字詞的所有自訂欄位。

b. 要創建一個新的自定義字段並添加它，請在搜索字段中輸入要創建的自定義字段的名稱，然後按 Enter 鍵。然後選擇要創建的自定義字段的類型並設置一個值。

c. 若要移除自訂欄位，請選擇您要移除的自訂欄位旁邊的 X 圖示。如果您從所有問題中移除自訂欄位，自訂欄位將會被刪除，而您在篩選時將不會再看到該欄位。

使用附件

您可以將附件新增至中的問題， CodeCatalyst 以便於存取相關檔案。請使用下列步驟來管理問題的附件。

新增至問題的附件大小會計入您空間的儲存配額。如需有關檢視和管理專案附件的資訊，請參閱檢視和管理附件。

> **Important**
>
> Amazon 不會掃描或分析問題的附件 CodeCatalyst。任何使用者都可以將附件新增至可能包含惡意程式碼或內容的問題。在管理附件和防範惡意程式碼、內容或病毒時，請確定使用者知道最佳做法。

新增、下載或移除附件

1. 選擇您要管理附件的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 若要新增附件，請選擇 [上傳檔案]。導航到操作系統的文件資源管理器中的文件並選擇它。選擇 [開啟] 以將其新增為附件。如需配額資訊，例如附件大小上限，請參閱中的問題配額 CodeCatalyst。

   請注意下列對附件檔案名稱和內容類型的限制：

   - 檔案名稱中不允許使用下列字元：
     - 控制字符：`0x00-0x1f`和 `0x80-0x9f`
     - 保留字元：`/?<`、`>`、`\`、`:`、`*`、`|`、和 `"`
     - Unix 保留的文件名：`.`和 `..`
     - 後置週期和空格

- 視窗保留的檔案名稱:`CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9`
- 附件的內容類型必須符合下列媒體類型的模式：

```
media-type = type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

例如 `text/html; charset=UTF-8`。

3. 若要下載附件，請在您要下載的附件旁邊選擇省略符號選單，然後選擇 [下載]。
4. 若要複製附件的 URL，請選擇要複製 URL 的附件旁邊的省略符號選單，然後選擇「複製 URL」。
5. 若要移除附件，請選擇要移除之附件旁的省略符號選單，然後選擇 [刪除]。

管理有關問題的工作

您可以將任務添加到問題中，以進一步分解，組織和跟踪該問題的工作。

若要管理問題上的工作

1. 選擇您要管理工作的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 在 [工作] 中，您可以檢視和管理問題的工作。
    1. 若要新增工作，請在文字欄位中輸入工作名稱，然後按 Enter 鍵。
    2. 若要將工作標示為已完成，請選擇工作的核取方塊。
    3. 若要檢視或更新工作的詳細資訊，請從清單中選擇它。
    4. 若要重新排序工作，請從核取方塊左側選擇並拖曳工作。
    5. 若要移除工作，請選擇工作的省略符號功能表，然後選擇 [移除]。

將問題標記為已封鎖或解除封鎖

如果有問題阻止您解決問題，您可能需要將其標記為已阻止。例如，如果您的問題依賴於對尚未合併的代碼庫中另一部分的更改，則可能會阻止您的問題。

當您將問題標示為已封鎖時，會在問題中 CodeCatalyst 新增紅色的「封鎖」標籤，使其在待處理項目、封存或主機板上清晰可見。

當外部情況得到解決時，您可以解除封鎖此問題。

若要將問題標示為已封鎖

1. 開啟您要標記為已封鎖的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 選擇 [動作]，然後選擇 [標記為封鎖]。

若要解除封鎖問題

1. 開啟您要解除封鎖的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 選擇 [動作]，然後選擇 [標記為已解除封鎖]。

新增、編輯或刪除注釋

您可以在問題上發表評論。在註解中，您可以標記其他空間成員、空間中的其他專案、相關問題和程式碼。

若要為問題新增註解

1. 導航到您的項目。
2. 在導覽列中選擇「問題」。
3. 選擇您要新增註解的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
4. 在「註解」欄位中輸入註解。您可以使用降價來添加格式。
5. 選擇傳送。

編輯註解

您可以編輯您對問題所做的評論。您只能編輯您撰寫的註解。

1. 導航到您的項目。
2. 在導覽列中選擇「問題」。
3. 選擇您要編輯留言的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
4. 若要編輯註解，請尋找您要編輯的留言。

> **Tip**
>
> 您可以先按最舊或最新的評論排序。註解一次載入 10 個。

5. 選擇省略符號圖示，然後選擇 [編輯]。
6. 編輯註解。您可以使用降價來添加格式。
7. 選擇儲存。評論現在已更新。

刪除註解

您可以刪除您對問題所做的評論。您只能刪除您編寫的注釋。刪除留言後，您的使用者名稱會顯示出來，但這個註解已被刪除，以取代原始註解文字。

1. 導航到您的項目。
2. 在導覽列中選擇「問題」。
3. 選擇您要刪除留言的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
4. 選擇省略符號圖示，選擇 [刪除]，然後選擇 [確認]。

在評論中使用提及

您可以在註解中提及空間成員、空間中的其他專案、相關問題以及程式碼。這樣做會建立指向您提及的使用者或資源的快速連結。

收到註解中的 @mention

1. 導航到您的項目。
2. 在導覽列中選擇「問題」。
3. 選擇您要編輯的問題，以檢視問題詳細資料。如需尋找問題的說明，請參閱尋找及檢視問題。
4. 選擇 [新增註解] 文字方塊。
5. 鍵入@*user_name*以提及其他用戶。
6. 鍵入@*project_name*以提及項目。
7. 輸入@*issue_name*或@*issue_number*提及其他問題。
8. 鍵入@*file_name*以提及源存儲庫中的特定文件或代碼。

> **Note**
>
> 前 5 個項目（用戶，源存儲庫，項目等）的列表，其中包含與您的@mention將在您鍵入時填充匹配的術語。

9. 選擇您想要提及的所需項目。顯示項目所在位置的路徑將填入註解文字方塊中。
10. 完成評論，然後選擇「傳送」。

處理問題

每個問題都有生命週期。在中 CodeCatalyst，問題通常以待處理項目的草稿開始。當該問題的工作要開始時，它會移動到另一個狀態類別，並在不同的狀態之間移動，直到它完成為止，然後將其封存。您可以透過下列方式在問題的生命週期中移動或處理問題：

- 您可以在積壓和主機板之間移動問題。
- 您可以在不同的完成階段中移動進行中的問題。
- 您可以封存已完成的問題。

積壓和董事會之間

一旦您開始處理問題，您就可以將問題從積壓移至主機板。如果工作延期，您也可以將問題移回待處理。

若要在積壓與主機板之間移動問題

1. 導航到您的項目。
2. 在導覽窗格中，選擇 [問題]。預設檢視為電路板。
3. 若要將問題從主機板移至待處理項目，請執行下列動作：
   a. 選擇您要移動的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
   b. 從「狀態」下拉式功能表選擇「未交訂單」
4. 若要將問題從待處理項目移至主機板：
   a. 若要切換作業選項至未交訂單，請選擇「看板」，然後選擇「未交
   b. 選擇您要移動的問題。如需尋找問題的說明，請參閱尋找及檢視問題。

c. 選擇「新增至看板」，或選擇「未交訂單」以外的「狀態」。

在主機板上的生命週期階段中處理問題

您可以將問題移到看板中的不同狀態，直到完成為止。

若要在主機板內移動問題

1. 在導覽窗格中，選擇 [問題]。預設檢視為電路板。
2. 執行以下任意一項：
   - 將問題拖放到另一個狀態。
   - 選擇問題，然後從「狀態」下拉式功能表中選擇狀態。
   - 選擇問題，然後選擇「移至：######。

如需封存問題的相關資訊，請參閱封存問題。

在群組之間移動問題

您可以依各種參數將「所有問題」和「看板」檢視中的問題分組。如果問題已分組，您可以將問題從一個群組移至另一個群組。將問題從一個群組移到另一個群組時，會自動編輯問題分組的欄位，以符合目標群組。

例如，假設有一家公司使用 CodeCatalyst 的問題分配給兩個人，王秀蘭和 Sanvi Sarkar。看板依據分組`Assignee`，並有兩個群組，每個工作負責人各一個群組。將問題從王秀蘭小組移至 Sanvi Sarkar 小組將更新問題的受讓人到薩恩維·薩卡爾。

封存問題

> **Note**
>
> 不會在專案中刪除問題，而是封存這些問題。若要刪除問題，您必須刪除專案。

當您的專案不再需要問題時，您可以將問題歸檔。當您封存問題時， CodeCatalyst 會從所有篩選已封存問題的檢視中移除該問題。您可以在「已封存問題」預設檢視中檢視已封存的問題，視需要將問題取消封存。

您將問題封存在下列情況：

- 您已經完成問題，而且在「完成」欄中不再需要此問題。
- 你沒有計劃在它上面工作。
- 你錯誤地創建了它。
- 您已達到作用中問題的數目上限。

若要封存問題

1. 開啟您要封存的問題。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 選擇「動作」，然後選擇「移至封存」。
3. (選擇性) 若要快速封存具有 [已完成] 狀態的多個問題，請選擇主機板上任何 [已完成] 狀態頂端的垂直省略符號，然後選擇 [封存問題]。

若要取消封存問題

1. 開啟您要取消封存的問題。您可以從 [問題] 檢視切換器下拉式功能表中開啟 [已封存的問題] 檢視，以檢視已封存的問題清單。如需尋找問題的說明，請參閱尋找及檢視問題。
2. 選擇「取消封存」。

尋找及檢視問題

下列各節說明如何有效地搜尋及檢視 CodeCatalyst 專案中的問題。

CodeCatalyst 專案中的問題會顯示在視圖中。檢視可以是以清單格式顯示問題的網格檢視，也可以是以問題狀態組織的欄中的並排顯示問題的看板檢視。有四種預設檢視，您可以使用自訂分組、篩選和排序來建立自己的檢視。下列清單包含四個預設檢視的詳細資料。

- 草稿視圖是一個網格視圖，顯示當前未處理的問題。以「草稿」狀態類別中的狀態建立的任何問題都會顯示在此檢視中。專案團隊可以使用此檢視來查看仍在定義哪些問題，或正在等待指派和處理的問題。
- 「使用中問題」檢視是目前正在處理的所有問題的看板檢視。狀態為 [未開始]、[已啟動] 或 [已完成] 狀態類別的任何問題都會顯示在此檢視中。
- 「所有問題」視圖是一個網格視圖，展示專案中的所有問題，包括草稿和作用中的問題。
- 已封存檢視會顯示所有已封存的問題。

搜尋問題

您可以通過搜索特定參數找到問題。若要取得有關精簡搜尋的更多資訊，請參閱在中搜尋 CodeCatalyst。

若要搜尋問題

1. 導航到您的項目。
2. 使用搜尋列來搜尋問題或與問題相關的資訊。您可以使用查詢參數來精簡您的搜尋。如需詳細資訊，請參閱 在中搜尋 CodeCatalyst。

排序問題

依預設，中的問題 CodeCatalyst 會依「手動順序」排序。「手動排序」會依使用者移至問題的順序來顯示問題。您可以在以手動順序排序時拖放問題以更改其順序。此排序選項在梳理積壓問題和排定問題的優先順序時很有幫助。

下表顯示如何在網格視圖和電路板視圖中排序問題。

網格檢視排序選項	電路板檢視排序選項
手動下單	手動下單
上次更新	上次更新
優先順序	優先順序
估計	估計
Title	Title
ID	
Status	
封鎖	
自訂欄位	

請使用下列程序來變更問題的排序方式。

若要排序問題

1. 導航到您的項目。
2. 在導覽窗格中，選擇 [問題]。預設檢視為電路板。
3. (選擇性) 選擇 [作用中的問題] 以開啟 [問題] 檢視切換器下拉式功能表，以瀏覽至不同的問題檢視。
4. 若要排序網格檢視，有兩個選項：
   a. 選擇您要排序依據的欄位標題。選擇標題將在升序和降序之間循環。
   b. 選擇「排序依據」下拉式功能表，然後選擇要排序依據的參數。問題將按升序排序。
5. 若要排序看板檢視，請選擇「排序依據」下拉式功能表，然後選擇要依據的參數進行排序。問題將按升序排序。

分組問題

分組用於按照多個參數 (例如工作負責人、標籤和優先順序) 來組織電路板上的問題。

若要分組問題

1. 導航到您的項目。
2. 在導覽窗格中，選擇 [問題]。預設檢視為電路板。
3. (選擇性) 選擇 [作用中的問題] 以開啟 [問題] 檢視切換器下拉式功能表，以瀏覽至不同的問題檢視。
4. 選擇 [群組]。
5. 在「分組依據」中，選擇要依據分組的參數：
   - 如果您選擇「受指派人」或「優先順序」，請選擇群組順序。
   - 如果您選擇 [標籤]，請選擇標籤，然後選擇 [群組順序]。
6. (選擇性) 選擇「顯示空白群組」切換，以顯示或隱藏目前沒有指派任何問題的群組。
7. 檢視會在您做出選擇時更新。只有符合已設定參數的群組中才會出現問題。

篩選問題

使用篩選來尋找包含指定名稱、優先順序、標籤、自訂欄位或工作負責人的問題。

若要篩選問題

1. 導航到您的項目。
2. 在導覽窗格中，選擇 [問題]。
3. (選擇性) 選擇 [作用中的問題] 以開啟 [問題] 檢視切換器下拉式功能表，以瀏覽至不同的問題檢視。

> **Note**
>
> 若要根據問題名稱或說明中的字串進行篩選，請在問題搜尋列中輸入字串。

4. 選擇「篩選」，然後選擇「+ 加入濾鏡」。
5. 選擇要篩選的參數。您可以選擇多個過濾器和參數。您可以透過選取和或來設定篩選器，以顯示符合每個篩選器或任何個別篩選器的問題。檢視將會更新，以顯示符合篩選條件的問題。

建立問題檢視

您可以建立檢視以快速檢視符合特定篩選器集的問題。這可協助您節省時間，並快速檢視先前篩選、分組或排序依據的問題。

若要建立問題檢視

1. 在導覽窗格中，選擇 [問題]。
2. (選擇性) 根據您的使用案例，您可能想要從現有檢視建立檢視。若要切換作業選項至不同的檢視表，請選擇「作用中問題」以開啟「問題」檢視切換器下拉式功能表，然後選擇
3. (選擇性) 在建立檢視之前，先設定篩選器、分組和排序。您可以在建立視圖時加入這些視圖，但如果之前執行此操作，則可以在建立視圖之前預覽視圖中顯示的內容。
4. 從標題欄打開問題查看切換器下拉菜單。若要建立可根據狀態在欄中檢視問題的看板檢視，請在「看板」欄中選擇 +。若要建立在清單中檢視問題的網格檢視，請在「格點」欄中選擇 +。如果您改變主意，可以在建立視圖之前變更視圖類型。
5. 在「建立視圖」對話方塊中，輸入視圖的「名稱」。
6. 根據當前視圖的設置填寫「過濾器」，「問題分組依據」和「按字段排序問題」。如有必要，請更新它們。
7. 選擇 [建立檢視表] 以建立並切換至檢視表。

匯出問題

您可以將目前檢視中的問題匯出為 .xlsx 檔案。若要匯出問題，請執行下列步驟。

匯出問題的步驟

1. 導航到您的項目。
2. 在導覽列中選擇「問題」。
3. 選擇「作用中問題」以開啟「問題」檢視切換器下拉式功能表，並切換作業選項至包含您要匯出之問題的檢視表。只會匯出檢視中顯示的問題。
4. 選擇省略符號功能表，然後選擇「匯出至 Excel」。
5. .xlsx 檔案會下載。依預設，其標題為專案名稱和匯出完成日期。

設定問題設定

下列主題詳細說明如何在中設定問題的設定 CodeCatalyst。

主題

- 啟用或停用多個受指派人
- 設定問題工作量估算
- 狀態
- 標籤
- 自訂欄位
- 檢視和管理附件

啟用或停用多個受指派人

請遵循下列步驟，針對中的問題設定多個受指派人的設定。 CodeCatalyst

啟用或停用多個受指派人

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。
3. 在「基本設定」區段的「受指派人」中，切換指標以啟用將多個受指派人指派給相同問題。一個問題最多可以有 10 個受指派人。如果您未啟用此選項，您將只能為問題指派一位受指派對象。

設定問題工作量估算

請遵循下列步驟來配置中 CodeCatalyst問題的工作量估計設定。

若要設定問題的工作量估算

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。
3. 在 [基本設定] 區段的 [估計] 中，選擇估計值的顯示方式。可用的估算類型有 T 恤尺寸、斐波那契測序或隱藏估算值。當估計類型更新時，不會丟失任何數據，並且所有問題的估計值將被自動轉換。轉換對映如下表所示。

T 恤尺寸	斐波那契数
XS	1
XS	2
S	3
M	5
L	8
加大碼	13

狀態

您可以在圖版上新增自訂狀態。每個自訂狀態必須屬於下列其中一個類別：「草稿」、「未開始」、「已開始」或「已完成」。狀態類別用於協助組織狀態和填入預設檢視。如需狀態和狀態類別的詳細資訊，請參閱狀態和狀態類別和以取得有關檢視的詳細資訊，請參閱尋找及檢視問題。

若要建立狀態

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。
3. 在狀態中，選擇要顯示狀態的類別旁邊的加號圖示。
4. 命名狀態，然後選擇核取記號圖示。

> **Note**
>
> 選擇 X 圖示以取消新增狀態。

自訂狀態現在會顯示在您的圖版上，並在建立問題時顯示為選項。

編輯狀態

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。
3. 在 「狀態」中，選擇您要編輯或變更的狀態旁邊的「編輯」圖示。
4. 編輯狀態，然後選擇核取記號圖示。

編輯過的狀態現在會顯示在您的主機板上。

若要移動狀態

1. 在導覽窗格中，選擇 [問題]。
2. 選擇省略符號圖示，然後選擇 [設定]。
3. 在 「狀態」中，選擇您要移動的狀態。
4. 將狀態拖放到您想要的位置。

> **Note**
>
> 您只能在指定的類別中移動某個狀態。

狀態現在會在您的主機板上重新排序。

若要停用狀態

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。

3. 在狀態中，選擇您要停用的狀態。
4. 在您要停用的狀態上，選擇狀態上的切換。現在狀態會變成灰色。

> **Note**
>
> 停用狀態會顯示在主機板上，直到所有問題都移出該狀態為止。無法將問題新增至停用狀態。

5. 若要重新啟用停用狀態，請選擇狀態的開關。狀態不再呈現灰色。

> **Note**
>
> 每個類別中必須至少有一個有效狀態。如果類別中只有一種狀態，則無法停用該狀態。

標籤

您可以自訂問題的標籤。這包括編輯標籤和更改顏色。標籤可以幫助您對工作進行分類和組織。

建立標籤

在中 CodeCatalyst，您可以在建立新問題或編輯現有問題時加入標籤來建立標籤。如需詳細資訊，請參閱 在中建立問題 CodeCatalyst 及 編輯和協作的問題 CodeCatalyst。

編輯標籤

使用下列步驟變更現有標示的名稱或顏色。

編輯標示的步驟

1. 在導覽窗格中，選擇 [問題]。
2. 選擇活動問題以打開問題查看切換器下拉菜單，然後選擇設置。
3. 在標籤瓷磚是在項目中使用的標籤的列表。選擇您要編輯的標籤旁邊的編輯圖示。執行下列其中一項或多項：
   a. 編輯標籤的名稱。
   b. 若要變更顏色，請選擇色輪。使用挑選器來選擇新顏色。
4. 若要儲存對標籤所做的變更，請選擇核取記號圖示。

5. 變更的標籤現在會顯示在可用標籤清單中。您還可以查看有多少問題正在使用該標籤。

> **Note**
>
> 您可以選擇每個標籤旁顯示的數字，瀏覽至「所有問題」頁面，並查看包含該標籤的所有問題。

刪除標籤

您目前無法在中刪除問題標籤 CodeCatalyst。如果您從所有問題中移除標籤，該標籤將出現在問題設定的「標籤」區段中的「未使用的標籤」區段中。使用篩選器或在問題中新增標籤時，未使用的標籤會出現在標籤清單的末尾。您可以在問題設定中找到所有標籤 (已使用和未使用) 及其問題的概觀。

自訂欄位

您可以建立自訂欄位，以協助組織和檢視專案的工作。自訂欄位會新增至「篩選器」中的可用篩選器清單，以便您可以依自訂欄位篩選問題。自訂欄位是名稱和值配對。您可以依自訂欄位的名稱篩選，然後依該自訂欄位的值進行篩選。

一個問題可能有多個自定義字段。

建立自訂欄位

在中 CodeCatalyst，您可以在建立問題或編輯現有問題時新增自訂欄位來建立自訂欄位。如需詳細資訊，請參閱 在中建立問題 CodeCatalyst 及 編輯和協作的問題 CodeCatalyst。

刪除自訂欄位

刪除自定義字段，您必須從添加到的每個問題中刪除自定義字段。刪除自定義字段後，您將不再在「過濾器」中看到自定義字段。您可以使用篩選器來檢視自訂欄位的所有問題，並透過編輯問題將其移除。如需詳細資訊，請參閱 尋找及檢視問題 和 編輯問題

檢視和管理附件

您可以在問題設置中查看包含添加到項目中問題的每個附件的表格。此表格包含每個附件的詳細資訊，包括內容類型、新增時間、加入的問題及其狀態，以及檔案大小等資訊。

此表格可用來輕鬆識別已完成或封存問題的大型附件，以便將其移除，以釋放空間儲存空間。

> **Important**
>
> Amazon 不會掃描或分析問題的附件 CodeCatalyst。任何使用者都可以將附件新增至可能包含惡意程式碼或內容的問題。在管理附件和防範惡意程式碼、內容或病毒時，請確定使用者瞭解最佳作法。

若要檢視和管理專案中的所有問題附件

1. 在導覽窗格中，選擇 [問題]。
2. 選擇省略符號圖示，然後選擇 [設定]。
3. 選擇「附件」標簽。

中的問題配額 CodeCatalyst

下表說明 Amazon 中問題的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst..

資源	預設配額
作用中問題	每個項目最多 1,000 個。
附件大小	每個附件的最大容量為 500 MB。 最大附件儲存空間總數會受到空間整體儲存限制的影響。如需詳細資訊，請參閱 定價。
問題總數 (作用中和已封存)	每個項目最多 10 萬個。
已存視景	每個專案最多可儲存 50 個問題視圖。
您可以連結至問題的提取要求數	每個問題最多 50 個提取請求。
狀態 (每個專案)	每個項目最多 50 個。
狀態（每個問題）	每個問題最多 50 個。
標籤（每個項目）	每個項目最多 200 個。

資源	預設配額
標籤 (每期)	每個問題最多 50 個。
自定義字段（每個問題）	每個問題最多 50 個。
受讓人	每個問題最多 10 個。
說明	每個發行最多 1,000 個。
任務	每個問題最多 100 個。

中的身分識別、權限和存取 CodeCatalyst

當您第一次登入 Amazon CodeCatalyst 時，您會建立 AWS 產生器 ID。 AWS 中不存在建置器 ID AWS Identity and Access Management。您在第一次登入時選擇的使用者名稱會成為您身分識別的唯一使用者 ID。

在中 CodeCatalyst，您可以使用下列兩種方式之一首次登入：

- 作為創建空間的一部分。
- 作為接受中專案或空間邀請的一部分 CodeCatalyst。

與您的身份關聯的一個或多個角色決定了您可以在中執行的操作 CodeCatalyst。專案角色 (例如「專案管理員」和「參與者」) 是專案專屬的角色，因此您可以在一個專案中擁有一個角色，而在另一個專案中具有不同的角色。如果您建立空間， CodeCatalyst會自動為您指派 S pace 管理員角色。當使用者接受專案的邀請時，會將這些身分 CodeCatalyst 新增至空間，並為他們指派受限存取角色。當您邀請使用者加入專案時，您可以選擇您希望他們在專案中擁有的角色，這會決定他們可以在專案中執行哪些動作，以及不能在專案中執行哪些動作。大多數在專案上工作的使用者只需要「參與者」角色即可執行其工作。如需詳細資訊，請參閱 在 Amazon 中使用角色 CodeCatalyst。

除了專案角色之外，在使用 Git 用戶端或整合式開發環境 (IDE) 時，專案中的使用者還需要個人存取權杖 (PAT) 來存取專案的來源儲存庫。專案成員可將此 PAT 與第三方應用程式搭配使用，做為與其 CodeCatalyst 身分相關聯的應用程式特定密 例如，將來源存放庫複製到本機電腦時，必須提供 PAT 以及您的 CodeCatalyst 使用者名稱。

您可以在工作流程中部署動作時，使用服務角色執行動作 (例如存取 AWS CloudFormation 堆疊和資源)，以配置和資源之 CodeCatalyst 間的存取權限。 AWS 您必須為要執行的專案範本所包含的工作流程動作配置 CodeCatalyst 和 AWS 資源之間的存取權限。

主題

在 Amazon 中使用角色 CodeCatalyst

在 Amazon 中 CodeCatalyst，您可以同時為專案層級和空間層級的使用者指派角色。在專案中，角色會指定使用者可以在具有該專案資源的專案中執行的動作。當使用者加入專案時，會在空間中獲得成員資格。您可以新增或移除空間管理員的使用者。S pace 管理員角色擁有中任何角色最廣泛的權限。CodeCatalyst最佳作法是為使用者指派執行其工作所需的最窄權限。

您可以將角色指派給空間中的使用者。您也可以將角色指派給使用者身為成員的專案中的使用者。每個使用者在專案或空間中只能有一個角色，但使用者在每個專案和空間中可以有不同的角色。例如，使用者可能在一個專案中具有「專案管理員」角色，而在另一個專案中具有「參與者」角色。

主題

角色類型

空間有三個角色可用：

- 空間管理員
- 進階使用者
- 有限的訪問

接受專案邀請的使用者會在包含專案的空間中自動指派受限存取角色給他們。

專案中有四個角色可供成員使用：

- 專案管理員
- Contributor (作者群)
- 審稿人
- 唯讀

當您將使用者新增至專案時， CodeCatalyst 會自動將使用者授予受限存取角色給予他們。如果您從所有專案中移除使用者，則 CodeCatalyst會自動從該使用者移除「受限制」存取角色。

空間管理員角色

S pace 管理員角色是中最強大的角色 CodeCatalyst。僅將 S pace 管理員角色指派給需要管理空間各個層面的使用者，因為此角色具有中的所有權限 CodeCatalyst。具有 S pace 管理員角色的使用者是唯一可以從 S pace 管理員角色新增或移除其他使用者以及刪除空間的使用者。

當您建立空間時， CodeCatalyst 會自動為您指派 S pace 管理員角色。最佳作法是，我們建議您將此角色新增至至少一個可以擔任此角色的其他使用者，以防原始空間建立者無法使用。

進階使用者角色

進階使用者角色是 CodeCatalyst空間中第二強大的角色，但無法存取空間中的專案。它是專為需要能夠在空間中建立專案並協助管理空間使用者和資源的使用者所設計。將超級使用者角色指派給身為團隊領導者或經理的使用者，這些使用者需要在空間中建立專案和管理使用者做為其工作一部分的能力。

有限的存取角色

「受限存取存取」角色是大多數使用者在 CodeCatalyst 空間中所具有的角色。這是當使用者接受空間中專案的邀請時，自動指派給使用者的角色。它提供了在包含該專案的空間內工作所需的有限權限。將受限存取角色指派給您直接邀請到空間的使用者，除非他們的工作要求他們管理空間的某些層面。

專案管理員角色

專案管理員角色是專 CodeCatalyst案中最強大的角色。僅將此角色指派給需要管理專案各個層面的使用者，包括編輯專案設定、管理專案權限和刪除專案。

專案角色在空間層級沒有任何權限。因此，具有專案管理員角色的使用者無法建立其他專案。只有具有 S pace 管理員或超級使用者角色的使用者才能建立專案。

> **Note**
>
> S pace 管理員角色具有中的所有權限 CodeCatalyst。

貢獻者角色

「參與者」角色適用於 CodeCatalyst 專案中的大多數成員。將此角色指派給需要能夠在專案中處理程式碼、工作流程、問題和動作的使用者。

「審核者」角色

Reviewer 角色適用於需要能夠與專案中的資源互動的使用者，例如提取請求和問題，但不能建立和合併程式碼、建立工作流程，或在 CodeCatalyst 專案中啟動或停止工作流程執行。將 Reviewer 角色指派給需要能夠核准和註解提取請求、建立、更新、解決和註解問題的使用者，以及檢視專案中的程式碼和工作流程。

唯讀角色

「唯讀」角色適用於需要檢視資源和資源狀態，但不能與資源互動或直接對專案做出貢獻的使用者。具有此角色的使用者無法在中建立資源 CodeCatalyst，但是他們可以檢視資源並加以複製，例如複製儲存庫，以及將問題的附件下載到本機電腦。將「唯讀」角色指派給需要檢視資源和專案狀態但不直接與其互動的使用者。

每個角色的可用權限

下表顯示每個 CodeCatalyst 角色的可用權限。使用連結跳至適當的權限集。

- Space permissions
- Extensions permissions
- Project permissions
- Source repository permissions
- Dev Environment permissions
- Package repository and package permissions
- Workflow permissions
- Issues permissions
- Custom blueprint permissions
- Notifications permissions
- Search permissions

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
空間權限							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
創造空間	✓	✗	✗	✗	✗	✗	✗
編輯太空帳單詳情	✓	✗	✗	✗	✗	✗	✗
設定並啟用單一登入	✓	✗	✗	✗	✗	✗	✗
移除單一登入	✓	✗	✗	✗	✗	✗	✗
為空間啟用生成式 AI 功能	✓	✗	✗	✗	✗	✗	✗
停用空間的生成式 AI 功能	✓	✗	✗	✗	✗	✗	✗
刪除空間	✓	✗	✗	✗	✗	✗	✗
將其他使用者新增至 Space 管理員角色	✓	✗	✗	✗	✗	✗	✗
從 Space 管理員角色中移除其他使用者	✓	✗	✗	✗	✗	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
創建團隊							
刪除團隊							
更新團隊							
停用空間的機器資源							
啟用空間的機器資源							
建立專案							
將 AWS 帳戶連線與空間建立關聯							
更新 AWS 帳戶連線							
取消 AWS 帳戶與空間連線的關聯							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
刪除 AWS 帳戶連線並將其從空間中移除							
邀請其他人到這個空間							
建立 VPC 連線							
編輯 VPC 連線							
刪除 VPC 連線							
檢視空間中的活動記錄							
檢視 AWS 帳戶連線							
檢視的事件 CodeCatalyst							
查看空間							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
檢視團隊							
檢視 VPC 連線							
擴充功能權	**空間管理員角色**	**超級使用者角色**	**有限的存取角色**	**專案管理員角色**	**貢獻者角色**	**「審核者」角色**	**唯讀角色**
安裝擴展							
更新擴展							
刪除副檔名							
Connect GitHub 帳戶							
中斷連接 GitHub 帳戶							
Connect 吉拉網站							
斷開吉拉站點							
檢視已安裝的擴充功能的組態							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
查看擴展	✓	✓	✓	✓	✓	✓	✓

專案權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
編輯專案設定	✓	✗	✗	✓	✗	✗	✗
停用專案的機器資源	✓	✗	✗	✓	✗	✗	✗
啟用專案的機器資源	✓	✗	✗	✓	✗	✗	✗
刪除專案	✓	✗	✗	✓	✗	✗	✗
邀請使用者加入專案	✓	✗	✗	✓	✗	✗	✗
變更專案中使用者的角色	✓	✗	✗	✓	✗	✗	✗
從專案中移除使用者	✓	✗	✗	✓	✗	✗	✗
將團隊新增至專案	✓	✗	✗	✓	✗	✗	✗
從專案中移除團隊	✓	✗	✗	✓	✗	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
變更專案團隊的專案角色							
查看項目							
檢視專案活動							
檢視專案中的團隊							
檢視藍圖							

來源儲存庫權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立儲存庫							
連結儲存庫							
取消連結儲存庫							
刪除儲存庫							
編輯儲存庫設定							
檢視儲存庫							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
檢視儲存庫設定	✓	✗	✗	✓	✓	✓	✓
克隆存儲庫	✓	✗	✗	✓	✓	✓	✓
創建分支	✓	✗	✗	✓	✓	✗	✗
建立分支規則	✓	✗	✗	✓	✗	✗	✗
變更預設分支	✓	✗	✗	✓	✗	✗	✗
刪除分支	✓	✗	✗	✓	✓	✗	✗
合併分支	✓	✗	✗	✓	✓	✗	✗
更新分支規則	✓	✗	✗	✓	✗	✗	✗
檢視分行	✓	✗	✗	✓	✓	✓	✓
檢視分支規則	✓	✗	✗	✓	✓	✓	✓
建立資料夾	✓	✗	✗	✓	✓	✗	✗
刪除資料夾	✓	✗	✗	✓	✓	✗	✗
編輯資料夾	✓	✗	✗	✓	✓	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
檢視資料夾	✓	✗	✗	✓	✓	✓	✓
建立檔案	✓	✗	✗	✓	✓	✗	✗
刪除 檔案	✓	✗	✗	✓	✓	✗	✗
編輯檔案	✓	✗	✗	✓	✓	✗	✗
檢視檔案	✓	✗	✗	✓	✓	✓	✓
創建和推送提交	✓	✗	✗	✓	✓	✗	✗
查看提交	✓	✗	✗	✓	✓	✓	✓
建立提取請求	✓	✗	✗	✓	✓	✗	✗
建立提取要求的核准規則	✓	✗	✗	✓	✗	✗	✗
覆寫提取請求的合併需求	✓	✗	✗	✓	✗	✗	✗
更新提取請求	✓	✗	✗	✓	✓	✗	✗
更新提取請求的核准規則	✓	✗	✗	✓	✗	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
檢視提取請求							
檢視提取要求的核准規則							
關閉提取請求							
核准提取請求							
提取要求的註解							
在提取請求的註解中與 Amazon Q 互動							
為 Amazon Q 建立的提取請求建立修訂							
將問題連結至提取要求							
取消問題與提取請求的連結							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
開發環境權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立您自己的開發環境							
停止您自己的開發環境							
停止由其他使用者建立的開發環境							
恢復您自己的開發環境							
檢視您自己的開發環境							
檢視由其他使用者建立的開發環境							
編輯您自己的開發環境							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
編輯由其他使用者建立的開發環境							
刪除您自己的開發環境							
刪除其他使用者建立的開發環境							
為開發環境創建一個開發文件							
編輯開發環境的開發文件							
刪除開發環境的開發文件							
查看開發環境的開發文件							

Package 儲存區域和套件權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立套件儲存庫							
檢視套件儲存庫							
編輯套件儲存區							
刪除套裝程式庫							
建立閘道套件儲存庫							
檢視閘道套件儲存庫							
刪除閘道套件儲存庫							
新增上游套件儲存庫							
編輯上游儲存庫的搜尋順序							
移除上游套件儲存庫							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
Connect 至套裝程式儲存區域	✓	✗	✗	✓	✓	✓	✓
從套裝程式儲存區域讀取套件	✓	✗	✗	✓	✓	✓	✓
將套裝軟體發佈至套裝程式儲	✓	✗	✗	✓	✓	✗	✗
從上游儲存庫讀取及保留套件	✓	✗	✗	✓	✓	✓	✓
檢視套件	✓	✗	✗	✓	✓	✓	✓
檢視套件版本	✓	✗	✗	✓	✓	✓	✓
檢視套件版本資產	✓	✗	✗	✓	✓	✓	✓
列出套件版本相依性	✓	✗	✗	✓	✓	✓	✓
更新套件版本狀態	✓	✗	✗	✓	✗	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
更新套件原始設定							
刪除套件版本							

工作流權限	空間管理員角色	進階使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立工作流							
更新工作流							
刪除工作流							
開始工作流							
停止工作流							
建立工作流程密							
更新工作流程密							
刪除工作流程密							
建立環境							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
刪除環境							
建立車隊							
更新車隊							
刪除叢集							
管理其他帳戶的工作流程資源							
將 VPC 連線與環境建立關聯							
取消 VPC 連線與環境的關聯							
將 VPC 端連線的環境與工作流程建立關聯							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
取消與 VPC 連線環境與工作流程的關聯							
追蹤工作流程中的提交							
檢視環境							
檢視組建動作記錄							
檢視車隊							
檢視測試動作記錄							
檢視工作流							
檢視工作流執行							
檢視工作流執行結果							
檢視工作流程密							
問題權限	空間管理員角色	進階使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立問題							
更新問題							
檢視問題							
封存問題							
將問題分配給 Amazon Q							
在有關問題的評論中與 Amazon Q 互動							
從問題中取消分配 Amazon Q							
更新其他使用者建立的問題							
檢視某個問題的評論							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立問題的評論							
更新問題的評論							
建立標籤							
更新標籤							
視圖標示							
為問題新增標籤							
從問題中移除標籤							
建立問題的自訂狀態							
更新自訂狀態							
檢視自訂狀態							
移動自訂狀態							
停用自訂狀態							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
將附件新增至問題							
檢視問題附件							
從問題中移除附件							
將提取要求連結至問題							
取消提取請求與問題的連結							
鏈接一個吉拉項目							
取消 Jira 專案的連結							

自訂藍圖權限	空間管理員角色	進階使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
建立自訂藍圖專案							
發佈預覽自訂藍圖							
取消發佈預覽自訂藍圖							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
發佈自訂藍圖	✓	✓	✗	✗	✗	✗	✗
取消發佈自訂藍圖	✓	✓	✗	✗	✗	✗	✗
將自訂藍圖新增至空間藍圖目錄	✓	✓	✗	✗	✗	✗	✗
從空間藍圖目錄移除自訂藍圖	✓	✓	✗	✗	✗	✗	✗
管理自訂藍圖的發佈權限	✓	✓	✗	✗	✗	✗	✗
管理自訂藍圖的目錄版本	✓	✓	✗	✗	✗	✗	✗
更新自訂藍圖	✓	✓	✗	✗	✗	✗	✗
刪除自訂藍圖版本	✓	✓	✗	✗	✗	✗	✗
刪除自訂藍圖	✓	✓	✗	✗	✗	✗	✗
將自訂藍圖套用至專案	✓	✓	✗	✓	✗	✗	✗

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
取消自訂藍圖與專案的關聯							
更新已套用自訂藍圖的版本							
編輯自訂藍圖的別名							
檢視已發佈的自訂藍圖							

通知權限	空間管理員角色	進階使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
設定通知通道							
移除通知頻道							
編輯通知設定							
檢視通知設定							
自動接收有關 CodeCatalyst 事件的通知							

權限	空間管理員角色	超級使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
設定關聯電子郵件帳戶的電子郵件通知	✓	✓	✓	✓	✓	✓	✓

搜尋權限	空間管理員角色	進階使用者角色	有限的存取角色	專案管理員角色	貢獻者角色	「審核者」角色	唯讀角色
在專案內搜尋	✓	✗	✗	✓	✓	✓	✓
跨空間搜索	✓	✓	✗	✓	✓	✓	✓

檢視和變更使用者角色

您可以檢視指派給使用者的角色。這有助於您瞭解他們可以在專案中採取哪些動作。如果他們需要其他權限，您也可以變更他們的角色。

若要檢視專案中使用者的角色

1. 導覽至您要檢視與每個專案成員相關聯之角色的專案。

   > **Tip**
   >
   > 您可以選擇要在頂部導航欄中查看的項目。

2. 在導覽窗格中，選擇 [專案設定]。
3. 在 [成員] 索引標籤上，每個專案成員的角色都會顯示在 [角色] 中。

若要變更專案中的使用者角色

1. 導覽至您要變更與專案成員相關聯之角色的專案。

> **Tip**
>
> 您可以選擇要在頂部導航欄中查看的項目。

2. 在導覽窗格中，選擇 [專案設定]。
3. 在 [成員] 索引標籤的 [專案成員] 中，選擇您要變更其角色的使用者。選擇 [動作]，然後選擇 [編輯角色]。
4. 在 [角色] 中，選擇專案角色，然後選擇 [確認]。

檢視和變更空間中的角色

中接受專案邀請的所有使用者都會 CodeCatalyst 成為專案空間的成員。您可以檢視空間成員清單。您可以將使用者角色從有限的存取權限變更為 S pace 管理員，以更妥善管理您的空間及其資源。S pace 管理員角色是唯一允許使用者在中建立專案的角色 CodeCatalyst。

> **Warning**
>
> S pace 管理員角色是中最強大的角色 CodeCatalyst。具有此角色的使用者可以在中執行任何動作 CodeCatalyst，包括刪除空間。僅將此角色指派給需要此層級存取您空間的使用者。如需詳細資訊，請參閱 空間管理員角色。

若要變更使用者在空間中的角色

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至空間。

> **Tip**
>
> 如果您屬於多個空間，則可以選擇要在頂部導覽列中檢視的空間。

3. 選擇「成員」頁標。
4. 選擇您要變更其角色的使用者，然後選擇 [變更角色]。
5. 在 [變更角色] 中，選擇您要指派的角色，然後選擇 [確認]。

在 Amazon 中管理個人訪問令牌 CodeCatalyst

若要在具有 Git 用戶端或整合式開發環境 (IDE) 的本機電腦上存取某些 CodeCatalyst資源，例如來源儲存庫，您必須輸入應用程式特定的密碼。您可以建立個人存取權杖 (PAT) 以用於此目的。您建立的 PAT 會在中的所有空間和專案中 CodeCatalyst與您的使用者身分相關聯。您可以為您的 CodeCatalyst 身分建立多個 PAT。

您可以查看已創建的 PAT 的名稱和到期日，也可以刪除不再需要的 PAT。您只能在建立 PAT 密碼時複製它。

> **Note**
>
> 依預設，PAT 會在 1 年後到期。

建立 PATs

PAT 與您在 CodeCatalyst中的使用者身分相關聯。您只能在建立 PAT 密碼時複製它。

建立 PAT (主控台)

您可以使用控制台在中 CodeCatalyst建立 PAT。

若要建立個人存取權杖 (主控台)

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。 CodeCatalyst我的設定」 頁面隨即開啟。

   > **Tip**
   >
   > 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

3. 在「個人存取權杖」下，選擇「建立」

   便會顯示「建立 PAT」頁面。
4. 在 PAT 名稱中，輸入 PAT 的描述性名稱。

5. 在到期日中，保留預設日期，或選擇行事曆圖示以選取自訂日期。到期日預設為從目前日期算起 1 年。
6. 選擇建立。

> **Tip**
>
> 當您為來源儲存庫選擇複製儲存庫時，也可以建立此權杖。

7. 若要複製 PAT 密碼，請選擇「複製」。將 PAT 秘密存儲在您將能夠檢索它的位置。

> **Important**
>
> PAT 秘密僅顯示一次。關閉視窗之後，您無法擷取它。如果您未將 PAT 密碼儲存在安全的位置，您可以建立另一個密碼。

.

建立 PATs (CLI)

您可以使用 CLI 在 CodeCatalyst中建立 PATs。

若要建立個人存取權杖 (AWS CLI)

1. 在終端機或命令列中，執行下列create-access-token命令。

```
aws codecatalyst create-access-token
```

如果成功，命令會傳回建立 PAT 的相關資訊，如下列範例所示。

```
{
    "secret": "value",
    "name": "marymajor-22222EXAMPLE",
    "expiresTime": "2024-02-04T01:56:04.402000+00:00"
}
```

2.

建立 PAT 時，您只能檢視一次 PAT 密碼。如果您放錯了 PAT 密碼，或者擔心它沒有安全地存儲，則可以創建另一個密碼。

您可以使用檢視與您的使用者帳戶相關聯的 PAT。 AWS CLI您只能檢視 PAT 的相關資訊，而不能檢視 PAT 密碼本身的值。

> **Note**
>
> 請確定您使用的是最新版本的 AWS CLI 來使用 CodeCatalyst。早期版本可能不包含這些 CodeCatalyst 命令。您必須先設定您的設定 AWS CLI 檔，才能搭配使用它 CodeCatalyst。如需詳細資訊，請參閱 設定以使用AWS CLI與 CodeCatalyst。

檢視 PAT

您可以在 CodeCatalyst中檢視 PAT。此清單會顯示您與使用者身分相關聯的所有 PAT。您的 PAT 與中的所有空間和專案中的使用者紀要相關聯 CodeCatalyst。過期的 PATs 不會顯示，因為它們會在到期後被刪除。

檢視 PAT (主控台)

您可以使用主控台來檢視中 CodeCatalyst與您的使用者身分相關聯的 PAT。

若要檢視您的個人存取權杖 (主控台)

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。 CodeCatalyst 我的設定」 頁面隨即開啟。

   > **Tip**
   >
   > 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

3. 在「個人存取權杖」下，檢視目前 PAT 的名稱和到期日期。

檢視 PATs (CLI)

您可以使用 CLI 來檢視中 CodeCatalyst與您的使用者身分相關聯的 PAT。

若要檢視您的個人存取權杖 (AWS CLI)

- 在終端機或命令列中，執行下列list-access-tokens命令。

```
aws codecatalyst list-access-tokens
```

如果成功，命令會傳回與您的使用者帳戶相關聯之 PAT 的相關資訊，如下列範例所示。

```
{
    "items": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "marymajor-22222EXAMPLE",
            "expiresTime": "2024-02-04T01:56:04.402000+00:00"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "marymajor-11111EXAMPLE",
            "expiresTime": "2023-03-12T01:58:40.694000+00:00"
        }
    ]
}
```

刪除 PATs

您可以在 CodeCatalyst中刪除與您的使用者身份相關聯的 PAT。

刪除 PATs（控制台）

您可以使用控制台刪除中 CodeCatalyst的 PAT。

若要刪除個人存取權杖 (主控台)

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在頂端選單列中，選擇您的設定檔徽章，然後選擇 [我的設定]。 CodeCatalyst 我的設定」 頁面隨即開啟。

> **Tip**
>
> 您還可以通過轉到項目或空間的成員頁面並從成員列表中選擇您的名稱來查找您的用戶個人資料。

3. 在 [個人存取權杖] 下，選擇您要刪除之 PAT 旁的選取器，然後選擇 [刪除]。

   在刪除 PAT:? <name> 頁面中，若要確認刪除，請在文字欄位中鍵入 delete。選擇刪除。

刪除程式 (CLI)

您可以使用刪除與您的使用者身分相關聯的 PAT AWS CLI。若要執行此操作，您必須提供 PAT 的 ID，您可以使用delete-access-token指令來檢視該 ID。

> **Note**
>
> 請確定您使用的是最新版本的 AWS CLI 來使用 CodeCatalyst。早期版本可能不包含這些 CodeCatalyst 命令。若要取得有關使用 AWS CLI 與的更多資訊 CodeCatalyst，請參閱〈〉設定以使用AWS CLI與 CodeCatalyst。

若要刪除個人存取權杖 (AWS CLI)

- 在終端機或命令列上，執行delete-access-token命令，提供您要刪除之 PAT 的 ID。例如，執行下列命令以刪除識別碼為 *123* 範例的PAT。

  ```
  aws codecatalyst delete-access-token --id a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb
  ```

  如果成功，此命令不會傳回任何回應。

Amazon 中的多因素身份驗證（MFA）CodeCatalyst

無論您是建立供個人使用或專業使用的 AWS Builder ID 設定檔，我們都建議您將多重要素驗證 (MFA) 設定為另一層安全性。如果您是 Space 的成員，並與其他人合作進行專案，我們特別建議您設定 MFA。由於一個以上的人可以存取專案，因此安全漏洞的機會就會有更多。

啟用 MFA 時，您必須 CodeCatalyst 使用電子郵件和密碼登入 Amazon。登入的這部分是第一個因素，您會使用您知道的內容。然後，您可以使用代碼或安全金鑰登入。這是第二個因素，這是你有。第二個因素可能是您的移動設備或通過點擊或按下連接到計算機的安全密鑰生成的身份驗證碼。綜合而言，這些多個因素通過防止未經授權的訪問提高了安全性。

如何註冊設備以使用多因素身份驗證

使用 [我的設定檔] > [多重要素驗證] 上的下列程序，為您的新裝置註冊多重要素驗證 (MFA)。

> **Note**
>
> 我們建議您先將適當的驗證器應用程式下載到您的裝置上，然後再開始執行此程序中的步驟。如需可用於 MFA 裝置的應用程式清單，請參閱驗證器應用程式。

註冊您的裝置以便與 MFA 搭配使用

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [使用者設定檔]。「設 CodeCatalyst定檔」頁面會開啟。
3. 在設定檔頁面上，選擇 [管理設定檔和安全性]。AWS 生成器 ID 配置文件頁面打開。
4. 在頁面左側，選擇 [安全性]。
5. 在 [多因素驗證] 頁面上，選擇 [註冊裝置]。
6. 在 [註冊 MFA 裝置] 頁面上，選擇下列其中一種 MFA 裝置類型，然後依照指示進行：

   - 安全金鑰或內建驗證器

     1. 在 [註冊使用者的安全金鑰] 頁面上，遵循瀏覽器或平台提供給您的指示。

        > **Note**
        >
        > 這種體驗會根據您的操作系統和瀏覽器而有所不同，因此請按照瀏覽器或平台顯示的說明進行操作。成功註冊設備後，您將可以選擇將友好的顯示名稱與新註冊的設備關聯。如果您要變更此設定，請選擇 [重新命名]，輸入新名稱，然後選擇 [儲存]。

   - 驗證器應用程式

1. 在 [設定驗證器應用程式] 頁面上，您可能會注意到新 MFA 裝置的設定資訊，包括 QR 碼圖形。該圖形是密鑰的表示形式，該密鑰可用於在不支持 QR 碼的設備上手動輸入。
2. 使用實體 MFA 裝置，執行下列動作：
   a. 開啟相容的 MFA 驗證器應用程式。如需可與 MFA 裝置搭配使用的已測試應用程式清單，請參閱測試驗證器應用程式。如果 MFA 應用程式支援多個裝置，請選擇建立新 MFA 裝置的選項。
   b. 確定 MFA 應用程式是否支援 QR 碼，然後在「設定驗證器應用程式」頁面上執行下列其中一項操作：
      i. 選擇 [顯示 QR 碼]，然後使用應用程式掃描 QR 碼。例如，您可以選擇相機圖示或選擇類似於「掃描程式碼」的選項。然後使用設備的相機掃描代碼。
      ii. 選擇顯示密鑰，然後將該密鑰輸入到您的 MFA 應用程序中。

      > **Important**
      >
      > 當您為 AWS 產生器 ID 設定 MFA 裝置時，請將 QR 碼或私密金鑰的複本儲存在安全的地方。如果您遺失手機或必須重新安裝 MFA 驗證器應用程式，這可能會有所幫助。如果其中一種情況發生，您可以快速重新配置應用程序以使用相同的 MFA 配置。

3. 在 [設定驗證器應用程式] 頁面的 [驗證器代碼] 下，輸入目前顯示在實體 MFA 裝置上的一次性密碼。

   > **Important**
   >
   > 產生代碼之後立即提交您的請求。如果您產生代碼，然後等待太長時間才能提交請求，則 MFA 裝置已成功與您的 AWS Builder ID 設定檔建立關聯，但 MFA 裝置不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

4. 選擇 Assign MFA (指派 MFA)。MFA 裝置現在可以開始產生一次性密碼，現在可供使用。

驗證器應用程式

身份驗證器應用程序是基於一次性密碼（OTP）的第三方身份驗證器。使用者可以使用安裝在行動裝置或平板電腦上的驗證器應用程式做為授權的 MFA 裝置。協力廠商驗證器應用程式必須符合 RFC 6238，RFC 6238 是標準型 TOTP (以時間為基礎的一次性密碼) 演算法，可產生六位數驗證碼。

當系統提示輸入 MFA 時，使用者必須在顯示的輸入方塊中輸入驗證器應用程式的有效代碼。每個指派給使用者的 MFA 裝置都必須是唯一的。任何指定使用者都可以註冊兩個驗證器應用程式。

測試驗證器應用程式

雖然任何符合 TOTP 標準的應用程式都可以與 IAM 身分中心 MFA 搭配使用，但下表列出可供選擇的知名第三方驗證器應用程式。

作業系統	測試驗證器應用程式
Android	Authy，雙核移動，身份驗證器，Microsoft LastPass 身份驗證器，谷歌身份驗證器
iOS	Authy，雙核移動，身份驗證器，Microsoft LastPass 身份驗證器，谷歌身份驗證器

變更您的 MFA 裝置

註冊 MFA 裝置後，您可以變更其名稱或刪除裝置。我們建議至少啟用一個 MFA 裝置，以獲得額外的安全性。您最多可以註冊五個裝置。若要瞭解如何新增更多項目，請參閱如何註冊設備以使用多因素身份驗證。

重新命名 MFA 裝置

若要重新命名您的 MFA 裝置

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [使用者設定檔]。「設 CodeCatalyst定檔」頁面會開啟。
3. 在設定檔頁面上，選擇 [管理設定檔和安全性]。AWS 生成器 ID 配置文件頁面打開。
4. 選擇頁面左側的多因素驗證。當您到達頁面時，您會看到「重新命名」顯示為灰色。
5. 選取您要變更的 MFA 裝置。選擇 Rename (重新命名)。然後彈出一個模態。
6. 在開啟的提示中，在 MFA 裝置名稱中輸入新名稱，然後選擇 [重新命名]。重新命名的裝置會出現在多重要素驗證裝置 (MFA) 下。

刪除 MFA 裝置

若要刪除 MFA 裝置

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 在右上角，選擇第一個首字母圖示旁邊的箭頭，然後選擇 [使用者設定檔]。「設 CodeCatalyst定檔」頁面會開啟。
3. 在設定檔頁面上，選擇 [管理設定檔和安全性]。AWS 生成器 ID 配置文件頁面打開。
4. 選擇頁面左側的多因素驗證。當您到達頁面時，您會看到「刪除」顯示為灰色。
5. 選取您要變更的 MFA 裝置。選擇刪除。出現一個模態，顯示刪除 MFA 設備？ 。依照指示刪除裝置。
6. 選擇刪除。刪除的設備不再出現在多因素身份驗證設備（MFA）下。

Amazon 的安全 CodeCatalyst

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足最敏感安全性空間的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端安全性 — AWS 負責保護中執行 AWS 服務的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。若要深入瞭解適用於的規範遵循計劃 CodeCatalyst，請參閱合規計劃的AWS 服務範圍範圍)。
- 雲端中的安全性 — 您的責任取決於您使用的 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

本文件可協助您了解如何在使用 Amazon 時應用共同的責任模型 CodeCatalyst。它會說明如何設定 CodeCatalyst 以符合安全性和合規性目標。您還將學習如何使用其他 AWS 服務來幫助您監控和保護您的 CodeCatalyst 資源。

目錄

- Amazon 基礎設施安全 CodeCatalyst
- Amazon 中的配置和漏洞分析 CodeCatalyst
- 您在 Amazon 的數據和隱私 CodeCatalyst
- Amazon 中工作流程動作的最佳實務 CodeCatalyst
- CodeCatalyst 信任模型

Amazon 的數據保護 CodeCatalyst

安全與合規是 Amazon CodeCatalyst 與客戶之間的共同責任，正如共同責任模型AWS共同責任模適用於您使用工作流程中的AWS資源一樣。如此模型所述， CodeCatalyst 負責保護服務的全球基礎結構。您必須負責維護在此基礎設施上託管之內容的控制權。此共同責任模型適用於中的資料保護 CodeCatalyst。

基於資料保護目的，我們建議您保護您的帳戶憑證，並在登入時設定多重要素驗證。如需詳細資訊，請參閱Amazon 中的多因素身份驗證（MFA） CodeCatalyst。

請勿在標籤或自由格式欄位 (例如姓名) 欄位中輸入有效或敏感的資料，例如客戶的電子郵件地址。這包括資源名稱以及您輸入的任何其他識別碼，以及任何連線的識別碼AWS 帳戶。 CodeCatalyst例如，請勿輸入機密或敏感性資訊做為空間、專案或部署叢集名稱的一部分。您在標籤、名稱或任意格式欄位中輸入的任何資料，用於名稱，都可能會用於帳單或診斷記錄，也可能包含在 URL 路徑中。這適用於使用控制台，APIAWS CLI， CodeCatalyst 操作開發工具包或任何 AWS SDK。

如果您向外部伺服器提供 URL，我們強烈建議您不要在 URL 中包含任何安全認證資訊，以驗證您對該伺服器的要求。

CodeCatalyst 來源儲存庫會在靜態時自動加密。不需要客戶採取任何行動。 CodeCatalyst 也會使用 HTTPS 通訊協定加密傳輸中的儲存庫資料。

CodeCatalyst 支持 MFA。如需詳細資訊，請參閱Amazon 中的多因素身份驗證（MFA） CodeCatalyst。

資料加密

CodeCatalyst 在服務中安全地存儲和傳輸數據。所有數據都在傳輸和靜態中進行加密。服務建立或儲存的任何資料，包括服務的任何中繼資料，都會以原生方式儲存在服務中並加密。

> **Note**
>
> 雖然問題的相關資訊會安全地儲存在服務中，但有關未解決問題的資訊也會儲存在瀏覽器的本機快取中，您可以在其中檢視問題看板、待處理項目和個別問題。為了獲得最佳的安全性，請務必清除瀏覽器快取以移除此資訊。

如果您使用連結至 CodeCatalyst的資源 (例如中的帳號連線AWS 帳戶或連結儲存庫) GitHub，則從該連結資源傳輸 CodeCatalyst 到該連結資源的資料會加密，但該連結資源中的資料處理則由該連結服務管理。如需詳細資訊，請參閱連結服務和Amazon 中工作流程動作的最佳實務 CodeCatalyst.

金鑰管理

CodeCatalyst 不支援金鑰管理。

網際網路流量隱私權

在中建立空間時 CodeCatalyst，您可以選擇儲存該空間資料和資源的AWS 區域位置。項目數據和元數據永遠不會離開該AWS 區域數據 不過，為了支援內部導覽 CodeCatalyst，系統會在分割區中的所有空間中複製一組有限的空間、專案和使用者AWS 區域中繼資料。它不會被複製到該分區AWS 區域之外。例如，如果您在建立空間AWS 區域時選擇美國西部 (奧勒岡)，您的資料將不會複寫到中國區域或 AWS GovCloud (US)。如需詳細資訊，請參閱管理AWS 區域、AWS全域基礎結構和AWS服務端點。

在分割區AWS 區域內複製的資料包括：

- 加密的雜湊值，代表空間名稱，以確保空間名稱的唯一性。這個值不是人類可讀的，也不會暴露空格的實際名稱
- 空間的唯一識別碼
- 協助跨空間導覽之空間的中繼資料
- 空AWS 區域間所在的位置
- 空間中所有專案的唯一 ID
- 指示使用者在空間或專案中角色的角色 ID
- 註冊時 CodeCatalyst，有關註冊過程的數據和元數據，包括：
  - 的唯一識別碼 AWS 建構家 ID
  - 使用者在其中的顯示名稱 AWS 建構家 ID
  - 使用者在他們的別名 AWS 建構家 ID
  - 使用者註冊其時使用的電子郵件地址 AWS 建構家 ID

- 註冊過程的進度
- 如果建立空間做為註冊程序的一部分，則為該空間的帳單帳戶使用的 AWS 帳戶 ID

空間名稱在中是唯一的 CodeCatalyst。請確定不要在空間名稱中包含敏感資料。

使用連結的資源和連線的帳戶 (例如與AWS 帳戶或 GitHub 存放庫的連線) 時，我們建議您設定來源和目標位置，使用每個帳戶都支援的最高安全性層級。 CodeCatalyst 使用傳輸層安全性 (TLS) 1.2 來保護AWS 區域、和可用區域之AWS 帳戶間的連線。

Identity and Access Management 和 Amazon CodeCatalyst

在 Amazon 中 CodeCatalyst，您可以建立並使用AWS產生器 ID 來登入和存取您的空間和專案。AWS 產生器 ID 不是 AWS Identity and Access Management (IAM) 中的身分，也不存在於AWS 帳戶. 但是， CodeCatalyst 在驗證用於計費目的的空間時，以及連接到以創建和使用其中的資源時，與 IAM 集成AWS 帳戶。AWS 帳戶

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全控制對 AWS 資源的存取權限。IAM 管理員可以控制誰能「完成身分驗證」(已登入) 和「獲得授權」(具有許可) 而得以使用資源。IAM 是一種您可以免費使用的 AWS 服務。

當您在 Amazon 中建立空間時 CodeCatalyst，您必須連接AWS 帳戶為您的空間的帳單帳戶。您必須在中具有管理員權限AWS 帳戶才能驗證 CodeCatalyst 空間，或具有權限。您也可以選擇為您的空間新增 IAM 角色， CodeCatalyst 以便在已連線的空間中建立和存取資源AWS 帳戶。這稱為服務角色。您可以選擇建立與多個帳戶的連線，AWS 帳戶並為 CodeCatalyst 每個帳戶建立服務角色。

> **Note**
>
> 的帳單 CodeCatalyst 會在AWS 帳戶指定為帳單帳戶中進行。不過，如果您在該角色AWS 帳戶或任何其他連線中建立 CodeCatalyst 服務角色AWS 帳戶，則該 CodeCatalyst 服務角色所建立和使用的資源將以該連線方式計費AWS 帳戶。如需詳細資訊，請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。

主題

- IAM 中以身分識別為基礎的政策
- IAM 中的政策動作
- IAM 中的政策資源

- IAM 中的政策條件金鑰
- 連線的身分識別原則範例 CodeCatalyst
- 使用標記控制對帳號連線資源的存取
- CodeCatalyst 權限參考
- 使用 CodeCatalyst 的服務連結角色
- AWSAmazon 的受管政策 CodeCatalyst
- Amazon CodeCatalyst 可存取AWS資源的 IAM 角色

IAM 中以身分識別為基礎的政策

以身分識別為基礎的原則是 JSON 權限原則文件，您可以附加至身分識別。該身分可以是使用者、使用者群組或角色。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。若要瞭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。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至附加的使用者或角色。如要瞭解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考。

CodeCatalyst 的身分型政策範例

若要檢視以 CodeCatalyst 身分為基礎的原則範例，請參閱。連線的身分識別原則範例 CodeCatalyst

IAM 中的政策動作

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，哪些主體可以在什麼樣的資源，以及在什麼條件下執行哪些操作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作需要政策中的多個動作。這些額外的動作稱為相依動作。

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "prefix:action1",
      "prefix:action2"
         ]
```

IAM 中的政策資源

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，哪些主體可以在什麼樣的資源，以及在什麼條件下執行哪些操作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

IAM 中的政策條件金鑰

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，哪些主體可以在什麼樣的資源，以及在什麼條件下執行哪些操作。

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。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 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 AWS 全域條件內容索引鍵。

連線的身分識別原則範例 CodeCatalyst

在中 CodeCatalyst，需AWS 帳戶要管理空間的帳單以及存取專案工作流程中的資源。帳戶連線用於授權新增AWS 帳戶至空間。連線中會使用以身分識別為基礎的原則。AWS 帳戶

根據預設，使用者和角色不具備建立或修改 CodeCatalyst 資源的權限。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授與使用者和角色對其所需資源執行動作的許可。管理員接著必須將這些政策連接至需要這些許可的使用者。

下列 IAM 政策範例會針對與帳戶連線相關的動作授予許可。使用它們來限制將帳戶連接到的存取 CodeCatalyst。

範例 1：允許使用者接受單一連線要求 AWS 區域

下列權限原則僅允許使用者檢視和接受 CodeCatalyst 和之間的連線要求AWS 帳戶。此外，政策使用條件只允許在 us-West -2 區域中執行動作，而不允許其他動作。AWS 區域若要檢視和核准請求，使用者必須使用AWS Management Console與要求中指定的相同帳戶登入。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:GetPendingConnection"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-west-2"
        }
      }
    }
  ]
}
```

範例 2：允許在主控台中管理單一連線 AWS 區域

下列權限原則可讓使用者管理單一區域之AWS 帳戶間 CodeCatalyst和之間的連線。此原則會使用條件，只允許在 us-West -2 區域中執行動作，而不允許其他動作。AWS 區域建立連線之後，您可以在中選擇選項來建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色AWS Management Console。在範例策略中，`iam:PassRole`動作的條件包括的服務主參與者 CodeCatalyst。只有具有該存取權限的角色才會在中建立AWS Management Console。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "codecatalyst:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-west-2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

範例 3：拒絕管理連線

下列權限原則會拒絕使用者管理 CodeCatalyst 和AWS 帳戶之間連線的任何能力。

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

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecatalyst:*"
            ],
            "Resource": "*"
        }
    ]
}
```

使用標記控制對帳號連線資源的存取

標籤可以附加至資源，或將要求傳遞給支援標記的服務。策略中的資源可以有標籤，策略中的某些動作可以包含標籤。標籤條件鍵包括aws:RequestTag和aws:ResourceTag條件鍵。建立 IAM 政策時，可使用標籤條件索引鍵來控制以下項目：

- 哪些使用者可以根據連線資源已有的標籤，對連線資源執行動作。
- 可在動作請求中傳遞的標籤。
- 請求中是否可使用特定的標籤鍵。

下列範例示範如何在策略中為 CodeCatalyst帳號連線使用者指定標籤條件。如需條件索引鍵的詳細資訊，請參閱 IAM 中的政策條件金鑰。

範例 1：允許根據要求中的標籤執行動作

下列原則會授與使用者核准帳號連線的權限。

若要這樣做，它會在請求指定名為 Project 且值為 ProjectA 的標籤時允許 AcceptConnection 和 TagResource 動作。(aws:RequestTag 條件索引鍵用來控制哪些標籤可在 IAM 請求中傳遞。) aws:TagKeys 條件可確保標籤索引鍵區分大小寫。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:TagResource"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

範例 2：允許根據資源標籤執行動作

下列策略授與使用者對帳號連線資源執行動作及取得相關資訊的權限。

為此，如果連接具有以值命名Project的標籤，則允許特定操作ProjectA。(aws:ResourceTag 條件索引鍵用來控制哪些標籤可在 IAM 請求中傳遞。)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:GetConnection",
        "codecatalyst:DeleteConnection",
        "codecatalyst:AssociateIamRoleToConnection",
        "codecatalyst:DisassociateIamRoleFromConnection",
        "codecatalyst:ListIamRolesForConnection",
        "codecatalyst:PutBillingAuthorization"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "ProjectA"
        }
      }
    }
  ]
}
```

CodeCatalyst 權限參考

本節為與連線的帳號連線資源搭配使用的動作AWS 帳戶提供權限參考 CodeCatalyst。下節說明與連線帳號相關的僅限權限動作。

帳戶連線所需的權限

使用帳戶連線需要下列權限。

CodeCatalyst 帳戶連線的權限	所需的許可	資源
AcceptConnection	需要接受將此帳戶連接到CodeCatalyst空間的請求。這只是 IAM 政策許可，不是 API 動作。	僅支援政策 Resource 元素中的萬用字元 (*)。
AssociateIamRoleToConnection	將 IAM 角色與帳戶連線相關聯時需要。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DeleteConnection	刪除帳戶連線時需要。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DisassociateIamRoleFromConnection	必須取消 IAM 角色與帳戶連線的關聯。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetBillingAuthorization	描述帳戶連線的帳單授權時需要。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetConnection	需要取得帳戶連線。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_I*

CodeCatalyst 帳戶連線的權限	所需的許可	資源
		D :/connect ions/ *connection_ID*
GetPendingConnection	需要取得擱置的要求才能將此帳戶連線至 CodeCatalyst空間。這只是 IAM 政策許可，不是 API 動作。	僅支援政策 Resource 元素中的萬用字元 (*)。
ListConnections	必須列出未擱置的帳號連線。這只是 IAM 政策許可，不是 API 動作。	僅支援政策 Resource 元素中的萬用字元 (*)。
ListIamRolesForConnection	必須列出與帳戶連線相關聯的 IAM 角色。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*
ListTagsForResource	必須列出與帳號連線相關聯的標籤。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*
PutBillingAuthorization	需要建立或更新帳戶連線的帳單授權。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*
RejectConnection	需要拒絕將此帳戶連線到 CodeCatalyst空間的要求。這只是 IAM 政策許可，不是 API 動作。	僅支援政策 Resource 元素中的萬用字元 (*)。
TagResource	需要建立或編輯與帳戶連線相關聯的標籤。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/connect ions/ *connection_ID*

CodeCatalyst 帳戶連線的權限	所需的許可	資源
UntagResource	需要移除與帳戶連線相關聯的標籤。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*

IAM 身分中心應用程式所需的許可

使用 IAM 身分中心應用程式需要下列許可。

CodeCatalyst IAM 身分中心應用程式的許可	所需的許可	資源
AssociateIdentityCenterApplicationToSpace	將 IAM 身分中心應用程式與 CodeCatalyst 空間建立關聯時需要。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
AssociateIdentityToIdentityCenterApplication	必須將身分與 CodeCatalyst空間的 IAM 身分中心應用程式建立關聯。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
BatchAssociateIdentitiesToIdentityCenterApplication	需要將多個身分與 CodeCatalyst 空間的 IAM 身分中心應用程式建立關聯。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
BatchDisassociateIdentitiesFromIdentityCenterApplication	必須取消多個身分與 IAM 身分中心應用程式的 CodeCatalyst	arn:aws:codecatalyst:region: *account_I*

CodeCatalyst IAM 身分中心應用程式的許可	所需的許可	資源
	空間關聯。這只是 IAM 政策許可，不是 API 動作。	*D* :/identity-center-applicati ons/ *identity-center-application_ID*
CreateIdentityCenterApplication	建立 IAM 身分中心應用程式所需。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
CreateSpaceAdminRoleAssignment	為指定 CodeCatalyst空間和 IAM 身分中心應用程式建立管理員角色指派時需要。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
DeleteIdentityCenterApplication	刪除 IAM 身分中心應用程式所需。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*
DisassociateIdentityCenterApplicationFromSpace	必須取消 IAM 身分中心應用程式與 CodeCatalyst 空間的關聯。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecataly st:region: *account_I D* :/identity-center-applicati ons/ *identity-center-application_ID*

CodeCatalyst IAM 身分中心應用程式的許可	所需的許可	資源
DisassociateIdentityFromIdentityCenterApplication	必須取消身分與 IAM 身分中心應用程式的 CodeCatalyst 空間關聯。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
GetIdentityCenterApplication	取得 IAM 身分中心應用程式相關資訊的必要條件。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
ListIdentityCenterApplications	需要查看帳戶中的 Alliam 身份中心應用程序列表。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
ListIdentityCenterApplicationsForSpace	需要按 CodeCatalyst 空間檢視 IAM 身分中心應用程式清單。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
ListSpacesForIdentityCenterApplication	需要透過 IAM 身分中心應用程式檢視 CodeCatalyst 空間清單。這只是 IAM 政策許可，不是 API 動作。	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*

CodeCatalyst IAM 身分中心應用程式的許可	所需的許可	資源
SynchronizeIdentityCenterApplication	需要將 IAM 身分中心應用程式與支援身分存放區同步。這只是 IAM 政策許可，不是 API 動作。	`arn:aws:codecatalyst:region: account_ID :/identity-center-applications/ identity-center-application_ID`
UpdateIdentityCenterApplication	需要更新 IAM 身分中心應用程式。這只是 IAM 政策許可，不是 API 動作。	`arn:aws:codecatalyst:region: account_ID :/identity-center-applications/ identity-center-application_ID`

使用 CodeCatalyst 的服務連結角色

Amazon CodeCatalyst 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結到 CodeCatalyst的唯一 IAM 角色類型。服務連結角色由預先定義， CodeCatalyst 並包含服務代表您呼叫其他服AWS務所需的所有權限。

服務連結角色可讓您 CodeCatalyst 更輕鬆地設定，因為您不需要手動新增必要的權限。 CodeCatalyst 定義其服務連結角色的權限，除非另有定義，否則只 CodeCatalyst 能擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這樣可以保護您的 CodeCatalyst 資源，因為您無法不小心移除存取資源的權限。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的 AWS 服務》，尋找服務連結角色欄中顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

服務連結角色權限 CodeCatalyst

CodeCatalyst 使用名為的服務連結角色 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization— 允許 Amazon 代表您存取應用程式執行個體設定檔和關聯目錄使用者和群組的 CodeCatalyst 唯讀權限。

AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 服務連結角色信任下列服務以擔任角色：

- codecatalyst.amazonaws.com

名為的角色權限原則 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy CodeCatalyst 允許對指定的資源完成下列動作：

- 動作：View application instance profiles and associated directory users and groups對於 CodeCatalyst spaces that support identity federation and SSO users and groups

您必須設定許可，以允許您的使用者、群組或角色建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

建立服務連結角色 CodeCatalyst

您不需要手動建立一個服務連結角色。當您在AWS Management Console、或 AWS API 中建 CodeCatalyst 立空間時AWS CLI，會為您建立服務連結角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。此外，如果您在 2023 年 11 月 17 日之前使用該 CodeCatalyst 服務，則該服務開始支援服務連結角色時，請在您的帳戶中 CodeCatalyst 建立該 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization角色。若要進一步了解，請參閱在我的 AWS 帳戶 中顯示新角色。

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。當您建立空間時，請再次為您建 CodeCatalyst 立服務連結角色。

您也可以使用 IAM 主控台，透過 View 應用程式執行個體設定檔以及關聯的目錄使用者和群組使用案例建立服務連結角色。在 AWS CLI CLI 或 AWS API 中，建立一個服務名稱為 codecatalyst.amazonaws.com 的服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的「建立服務連結角色」。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯下列項目的服務連結角色 CodeCatalyst

CodeCatalyst 不允許您編輯 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除下列項目的服務連結角色 CodeCatalyst

您不需要手動刪除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。當您刪除AWS Management Console、或 AWS API 中的空間時AWS CLI，會為您 CodeCatalyst 清除資源並刪除服務連結角色。

您也可以使用 IAM 主控台、AWS CLI 或 AWS API 來手動刪除服務連結角色。若要執行此操作，您必須先手動清除服務連結角色的資源，然後才能手動刪除它。

> **Note**
>
> 當您嘗試刪除資源時，如果 CodeCatalyst 服務正在使用此角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

若要刪除使用的 CodeCatalyst 資源 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization

- 刪除空間。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

支援 CodeCatalyst 服務連結角色的區域

CodeCatalyst 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱AWS區域和端點。

CodeCatalyst 不支援在每個提供服務的區域中使用服務連結角色。您可以在下列區域中使用 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 角色。

區域名稱	區域身分	中的 Support CodeCatalyst
美國東部 (維吉尼亞北部)	us-east-1	否
美國東部 (俄亥俄)	us-east-2	否
美國西部 (加利佛尼亞北部)	us-west-1	否
美國西部 (奧勒岡)	us-west-2	是
非洲 (開普敦)	af-south-1	否
亞太區域 (香港)	ap-east-1	否
亞太區域 (雅加達)	ap-southeast-3	否
亞太區域 (孟買)	ap-south-1	否
亞太區域 (大阪)	ap-northeast-3	否
亞太區域 (首爾)	ap-northeast-2	否
亞太區域 (新加坡)	ap-southeast-1	否
亞太區域 (雪梨)	ap-southeast-2	否
亞太區域 (東京)	ap-northeast-1	否
加拿大 (中部)	ca-central-1	否
歐洲 (法蘭克福)	eu-central-1	否
歐洲 (愛爾蘭)	eu-west-1	是
歐洲 (倫敦)	eu-west-2	否
歐洲 (米蘭)	eu-south-1	否
歐洲 (巴黎)	eu-west-3	否
歐洲 (斯德哥爾摩)	eu-north-1	否

區域名稱	區域身分	中的 Support CodeCatalyst
中東 (巴林)	me-south-1	否
中東 (阿拉伯聯合大公國)	me-central-1	否
南美洲 (聖保羅)	sa-east-1	否
AWS GovCloud (美國東部)	us-gov-east-1	否
AWS GovCloud (美國西部)	us-gov-west-1	否

AWSAmazon 的受管政策 CodeCatalyst

AWS 管理的政策是由 AWS 建立和管理的獨立政策。AWS 管理的政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請謹記，AWS 管理的政策可能不會授予您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 管理的政策中定義的許可。如果 AWS 更新 AWS 管理的政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 管理的政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管政策：AmazonCodeCatalystSupportAccess

這是一項政策，可授予所有空間管理員和空間成員使用與太空計費帳戶相關聯的 Business 或 Enterprise 進階支援計劃的權限。這些權限允許空間管理員和成員針對他們在 CodeCatalyst 權限策略中具有權限的資源實用高級支持計劃。

許可詳細資訊

此政策包含以下許可。

- support— 授予權限以允許使用者搜尋、建立和解決 Sup AWS port 案例。此外，也授予描述通訊、嚴重性等級、附件及相關支援案例詳細資料的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "support:DescribeAttachment",
        "support:DescribeCaseAttributes",
        "support:DescribeCases",
        "support:DescribeCommunications",
        "support:DescribeIssueTypes",
        "support:DescribeServices",
        "support:DescribeSeverityLevels",
        "support:DescribeSupportLevel",
        "support:SearchForCases",
        "support:AddAttachmentsToSet",
        "support:AddCommunicationToCase",
        "support:CreateCase",
        "support:InitiateCallForCase",
        "support:InitiateChatForCase",
        "support:PutCaseAttributes",
        "support:RateCaseCommunication",
        "support:ResolveCase"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 受管政策：AmazonCodeCatalystFullAccess

這是一項政策，授與在中的 Amazon S CodeCatalyst paces 頁面中管理您的 CodeCatalyst 空間和連接帳戶的權限AWS Management Console。此應用程式用於設AWS 帳戶定連接到中的空間 CodeCatalyst。

許可詳細資訊

此政策包含以下許可。

- `codecatalyst`— 授予中 Amazon CodeCatalyst 空間頁面的完整許可AWS Management Console。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeCatalystResourceAccess"
            "Effect": "Allow",
            "Action": [
                "codecatalyst:*",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeCatalystAssociateIAMRole"
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
    ]
```

```
}
```

AWS 受管政策：AmazonCodeCatalystReadOnlyAccess

這是一項政策，授與在中的 Amazon CodeCatalyst Spaces 頁面中檢視和列出空間和連線帳戶資訊的權限AWS Management Console。此應用程式用於設AWS 帳戶定連接到中的空間 CodeCatalyst。

許可詳細資訊

此政策包含以下許可。

- `codecatalyst`— 授與中的 Amazon CodeCatalyst 空間頁面的唯讀權限AWS Management Console。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecatalyst:Get*",
                "codecatalyst:List*",
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策：AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy

您無法將AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy; 附加到您的 IAM 實體。此原則附加至服務連結角色，可 CodeCatalyst 代表您執行動作。如需詳細資訊，請參閱 使用 CodeCatalyst 的服務連結角色。

此原則可讓客戶在 CodeCatalyst管理中的空間時，檢視應用程式執行個體設定檔和關聯的目錄使用者和群組 客戶在管理支援身分同盟和 SSO 使用者和群組的空間時，將檢視這些資源。

許可詳細資訊

此政策包含以下許可。

- sso— 授予許可，以允許使用者檢視在 IAM 身分中心針對中關聯空間管理的應用程式執行個體設定檔 CodeCatalyst。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid":
 "AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy",
   "Effect": "Allow",
   "Action": [
    "sso:ListInstances",
    "sso:ListApplications",
    "sso:ListApplicationAssignments",
    "sso:DescribeInstance",
    "sso:DescribeApplication"
   ],
   "Resource": "*"
  }
 ]
}
```

CodeCatalyst AWS受管理策略的更新

檢視 CodeCatalyst 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，請訂閱「 CodeCatalyst 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
AmazonCodeCatalyst ServiceRoleForIdentityCente	CodeCatalyst 添加了策略。 授與權限，以允許 CodeCatalyst 使用者檢視應用程式執行個	2023 年 11 月 17 日

變更	描述	日期
rApplicationSynchronization Policy – 新政策	體設定檔以及相關聯的目錄使用者	
AmazonCodeCatalyst SupportAccess – 新政策	CodeCatalyst 添加了策略。 授與權限，以允許 CodeCatalyst 使用者搜尋、建立和解決支援案例，以及檢視相關通訊和詳細資料。	2023 年 4 月 20 日
AmazonCodeCatalyst FullAccess – 新政策	CodeCatalyst 添加了策略。 授予完整的存取權限 CodeCatalyst。	2023 年 4 月 20 日
AmazonCodeCatalyst ReadOnlyAccess – 新政策	CodeCatalyst 添加了策略。 授與的唯讀存取權 CodeCatalyst。	2023 年 4 月 20 日
CodeCatalyst 開始追蹤變更	CodeCatalyst 開始追蹤其AWS受管理策略的變更。	2023 年 4 月 20 日

Amazon CodeCatalyst 可存取AWS資源的 IAM 角色

CodeCatalyst 可以通過連接到 CodeCatalyst 空間AWS 帳戶來訪問AWS資源。然後，您可以建立下列服務角色，並在連線帳戶時建立關聯這些角色。

如需有關您在 JSON 政策中使用之元素的詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考。

- 若要存取 CodeCatalyst 專案和工作流程中的資源，您必須先授與權限， CodeCatalyst 才能代表您存取這些資源。AWS 帳戶若要這麼做，您必須在連線的服務角色中建立AWS帳戶 CodeCatalyst可代表空間中的使用者和專案承擔的服務角色。您可以選擇建立和使用CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，也可以建立自訂服務角色並手動設定這些 IAM 政策和角色。最佳作法是為這些角色指派最少的必要權限。

> **Note**
>
> 若為自訂服務角色，則需要 CodeCatalyst 服務主體。如需有關 CodeCatalyst 服務主體和信任模型的詳細資訊，請參閱CodeCatalyst 信任模型。

- 若要透過連線來管理空間的支援AWS 帳戶，您可以選擇建立和使用允許使用 CodeCatalyst 者存取支援的AWSRoleForCodeCatalystSupport服務角色。如需 CodeCatalyst 空間支援的更多資訊，請參閱AWS Support對於 Amazon CodeCatalyst。

了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色

您可以為您的空間新增 IAM 角色， CodeCatalyst 以便在已連線中建立和存取資源AWS帳戶。這稱為服務角色。建立服務角色的最簡單方法是在建立空間時新增一個角色，並選擇該角色的CodeCatalystWorkflowDevelopmentRole-***spaceName***選項。這不僅會建立`AdministratorAccess`附加的服務角色，而且還會建立信任原則，以 CodeCatalyst 便代表空間中專案中的使用者承擔角色。服務角色的範圍是空間，而不是個別專案。若要建立角色，請參閱為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。您只能為每個帳戶中的每個空間建立一個角色。

> **Note**
>
> 此角色僅建議與開發帳戶搭配使用，並使用`AdministratorAccess`AWS受管理的策略，讓其具有完整存取權，以便在其中建立新政策和資源AWS 帳戶。

附加至CodeCatalystWorkflowDevelopmentRole-***spaceName***角色的原則旨在處理使用空間中使用藍圖建立的專案。它允許這些項目中的用戶使用連接中的資源開發，構建，測試和部署代碼AWS 帳戶。如需詳細資訊，請參閱建立AWS服務的角色。

附加到CodeCatalystWorkflowDevelopmentRole-***spaceName***角色的策略是中的`AdministratorAccess`受管理策略AWS。這是一項政策，可授與所有AWS動作和資源的完整存取權。若要在 IAM 主控台中檢視 JSON 政策文件，請參閱AdministratorAccess。

下列信任原則 CodeCatalyst 允許擔任CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。如需 CodeCatalyst 信任模型的詳細資訊，請參閱CodeCatalyst 信任模型。

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

為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色

請依照下列步驟建立將用於空間中工作流程的CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。對於要在專案中使用 IAM 角色的每個帳戶，您必須新增一個角色，例如開發人員角色。

在開始之前，您必須具有您的管理權限，AWS 帳戶或能夠與您的管理員合作。如需中如何使用AWS帳戶和 IAM 角色的詳細資訊 CodeCatalyst，請參閱管理 AWS 帳戶 空間。

若要建立並新增 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-***spaceName***

1. 在 CodeCatalyst 主控台中啟動之前，請開啟AWS Management Console，然後確定您已使用相同 AWS 帳戶的空間登入。
2. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
3. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
4. 選擇您要建立角色之AWS 帳戶位置的連結。AWS 帳戶詳細資訊頁面隨即顯示。
5. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登錄才能訪問該頁面。

6. 選擇在 IAM 中建立 CodeCatalyst 開發管理員角色。此選項會建立包含開發角色之權限原則和信任原則的服務角色。該角色將具有一個名稱`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。

   > **Note**
   >
   > 此角色僅建議與開發人員帳戶搭配使用，並使用`AdministratorAccess`AWS受管理的政策，讓其具有完整存取權，以便在其中建立新政策和資源AWS 帳戶。

7. 選擇 [建立開發角色]。
8. 在「連線」頁面的「可用的 IAM 角色」下 CodeCatalyst**`CodeCatalystWorkflowDevelopmentRole-`*****`spaceName`**，檢視新增至您帳戶的 IAM 角色清單中的角色。
9. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

了解服AWSRoleForCodeCatalystSupport務角色

您可以為空間新增 IAM 角色，空間中的使用 CodeCatalyst 者可以用來建立和存取支援案例。這稱為支援服務角色。建立支援服務角色的最簡單方法是在建立空間並選擇該角色的`AWSRoleForCodeCatalystSupport`選項時新增一個角色。這不僅會建立原則和角色，還會建立信任原則，以 CodeCatalyst 便代表空間中專案中的使用者擔任角色。服務角色的範圍是空間，而不是個別專案。若要建立角色，請參閱為您的帳戶和空間建立AWSRoleForCodeCatalystSupport角色。

附加至`AWSRoleForCodeCatalystSupport`角色的原則是受管理的原則，可提供支援權限的存取權。如需詳細資訊，請參閱 AWS 受管政策：AmazonCodeCatalystSupportAccess。

策略的信任角色允 CodeCatalyst 許擔任該角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codecatalyst.amazonaws.com",
                    "codecatalyst-runner.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

為您的帳戶和空間建立AWSRoleForCodeCatalystSupport角色

請依照下列步驟建立將AWSRoleForCodeCatalystSupport用於您空間中支援案例的角色。角色必須新增至空間的指定帳單帳戶。

在開始之前，您必須具有您的管理權限，AWS 帳戶或能夠與您的管理員合作。如需中如何使用AWS帳戶和 IAM 角色的詳細資訊 CodeCatalyst，請參閱管理 AWS 帳戶 空間。

若要建立並新增 CodeCatalyst AWSRoleForCodeCatalystSupport

1. 在 CodeCatalyst 主控台中啟動之前，請開啟AWS Management Console，然後確定您已使用相同AWS 帳戶的空間登入。
2. 導覽至您的 CodeCatalyst 空間。選擇 Settings (設定)，然後選擇 AWS 帳戶。
3. 選擇您要建立角色之AWS 帳戶位置的連結。AWS 帳戶詳細資訊頁面隨即顯示。
4. 選擇 [管理角色來源] AWS Management Console。

   將 IAM 角色新增至 Amazon CodeCatalyst 空間頁面會在中開啟AWS Management Console。這是 Amazon CodeCatalyst 空間頁面。您可能需要登入才能存取此頁面。
5. 在CodeCatalyst 空間詳細資料下，選擇新增 S CodeCatalyst upport 角色。此選項會建立包含預覽開發角色的權限原則和信任原則的服務角色。該角色將具有附加唯AWSRoleForCodeCatalystSupport一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服AWSRoleForCodeCatalystSupport務角色。
6. 在 [新增 Sup CodeCatalyst port 角色] 頁面上，保持選取的預設值，然後選擇 [建立角色]。
7. 在「可用的 IAM 角色」下 CodeCatalyst**CodeCatalystWorkflowDevelopmentRole-*spaceName***，檢視新增至您帳戶的 IAM 角色清單中的角色。
8. 要返回您的空間，請選擇前往 Amazon CodeCatalyst。

設定工作流程動作的 IAM 角色 CodeCatalyst

本節詳細說明您可以建立以搭配 CodeCatalyst帳戶使用的 IAM 角色和政策。如需建立實例角色的指示，請參閱手動建立工作流程動作的角色。建立 IAM 角色後，請複製角色 ARN 以將 IAM 角色新增至

您的帳戶連線，並將其與您的專案環境建立關聯。如需進一步了解，請參閱 將 IAM 角色新增至帳戶連線。

CodeCatalyst 為 Amazon S3 存取建立角色

對於 CodeCatalyst 工作流程建置動作，您可以使用預設CodeCatalystWorkflowDevelopmentRole-*spaceName*服務角色，也可以建立名為 CodeCatalystBuildRoleforS3 Access 的 IAM 角色。此角色使用具有範圍權限的策略，該策略 CodeCatalyst 需要對AWS CloudFormation. AWS 帳戶

此角色授予執行下列作業的權限：

- 寫入 Amazon S3 存儲桶。
- Support 資源建設AWS CloudFormation. 這需要 Amazon S3 訪問。

此角色使用下列原則：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 建置角色 AWS CloudFormation

對於 CodeCatalyst 工作流程建置動作，您可以使用預設CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，也可以建立具有必要許可的 IAM 角色。此角色使用具有範圍權限的策略，該策略 CodeCatalyst 需要對AWS CloudFormation. AWS 帳戶

此角色授予執行下列作業的權限：

- Support 資源建設AWS CloudFormation. 這是必需的，以及 Amazon S3 存取的 CodeCatalyst建置角色和的 CodeCatalyst 部署角色AWS CloudFormation。

下列AWS受管理的原則應附加至此角色：

- AWSCloudFormationFullAccess
- IAM FullAccess
- 亞馬遜 3 FullAccess
- 亞馬遜 API GatewayAdministrator
- AWSLambdaFullAccess

CodeCatalyst CDK 的構建角色

對於執行 CDK 建置動作的工作 CodeCatalyst 流程，例如現代三層 Web 應用程式，您可以使用預設CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，或者建立具有必要權限的 IAM 角色。此角色使用具有範圍權限的原則， CodeCatalyst 需要針對AWS CloudFormation. AWS 帳戶

此角色授予執行下列作業的權限：

- 寫入 Amazon S3 存儲桶。
- Support CDK 構建和AWS CloudFormation資源堆棧的構建。這需要存取 Amazon S3 以取得成品儲存、Amazon ECR 以支援映像儲存庫，以及 SSM 以進行虛擬執行個體的系統控管和監控。

此角色使用下列原則：

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

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 部署角色 AWS CloudFormation

對於使用的工作 CodeCatalyst 流程部署動作AWS CloudFormation，您可以使用預設CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，或者您可以使用具有範圍權限的原則，這些原則 CodeCatalyst 需要在中的AWS CloudFormation資源上執行工作。AWS 帳戶

此角色授予執行下列作業的權限：

- 允許調 CodeCatalyst 用函數以通過AWS CloudFormation執行藍/綠部署。
- 允許 CodeCatalyst 在中創建和更新堆棧和更改集。AWS CloudFormation

此角色使用下列原則：

```
{"Action": [
```

```
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
    "Effect": "Allow"
}
```

Note

第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。

```
"Resource": "*"
```

CodeCatalyst 部署 Amazon EC2 的角色

CodeCatalyst 工作流程部署動作使用具有必要許可的 IAM 角色。此角色使用具有範圍許可 CodeCatalyst 的政策，該政策 Amazon EC2 在您的. AWS 帳戶該CodeCatalystWorkflowDevelopmentRole-*spaceName*角色的預設政策不包括適用於 Amazon EC2 或 Amazon EC2 Auto Scaling 的許可。

此角色授予執行下列作業的權限：

- 建立 Amazon EC2 部署。
- 閱讀執行個體上的標籤，或透過 Auto Scaling 群組名稱識別 Amazon EC2 執行個體。
- 讀取、建立、更新和刪除 Amazon EC2 Auto Scaling 群組、生命週期勾點和擴展政策。
- 將資訊發佈到 Amazon SNS 主題。
- 擷取有關 CloudWatch 警報的資訊。
- 讀取並更新 Elastic Load Balancing。

此角色使用下列原則：

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
 "autoscaling:CompleteLifecycleAction",
"autoscaling:DeleteLifecycleHook",
"autoscaling:DescribeAutoScalingGroups",
"autoscaling:DescribeLifecycleHooks",
"autoscaling:PutLifecycleHook",
"autoscaling:RecordLifecycleActionHeartbeat",
"autoscaling:CreateAutoScalingGroup",
"autoscaling:UpdateAutoScalingGroup",
"autoscaling:EnableMetricsCollection",
"autoscaling:DescribePolicies",
"autoscaling:DescribeScheduledActions",
"autoscaling:DescribeNotificationConfigurations",
"autoscaling:SuspendProcesses",
"autoscaling:ResumeProcesses",
"autoscaling:AttachLoadBalancers",
"autoscaling:AttachLoadBalancerTargetGroups",
"autoscaling:PutScalingPolicy",
"autoscaling:PutScheduledUpdateGroupAction",
"autoscaling:PutNotificationConfiguration",
"autoscaling:PutWarmPool",
"autoscaling:DescribeScalingActivities",
"autoscaling:DeleteAutoScalingGroup",
"ec2:DescribeInstances",
"ec2:DescribeInstanceStatus",
"ec2:TerminateInstances",
"tag:GetResources",
"sns:Publish",
"cloudwatch:DescribeAlarms",
"cloudwatch:PutMetricAlarm",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
"elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
"elasticloadbalancing:DescribeTargetGroups",
"elasticloadbalancing:DescribeTargetHealth",
"elasticloadbalancing:RegisterTargets",
```

```
"elasticloadbalancing:DeregisterTargets"
],
"Resource": "resource_ARN"
      }
   ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon ECS 的部署角色

對於 CodeCatalyst 工作流程動作，您可以建立具有必要許可的 IAM 角色。您可以使用預設CodeCatalystWorkflowDevelopmentRole-*spaceName*服務角色，也可以為部 CodeCatalyst 署動作建立 IAM 角色，以用於 Lambda 部署。此角色使用具有範圍許可的政策，該政策 CodeCatalyst 需要在您的 Amazon ECS 資源上執行任務。AWS 帳戶

此角色授予執行下列作業的權限：

- 代表 CodeCatalyst 使用者在 CodeCatalyst 連線中指定的帳戶中啟動滾動 Amazon ECS 部署。
- 讀取、更新和刪除 Amazon ECS 任務集。
- 更新 Elastic Load Balancing 目標群組、監聽器和規則。
- 叫用 Lambda 函數。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。
- 將資訊發佈到 Amazon SNS 主題。

此角色使用下列原則：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
```

```
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
    ],
    "Resource":"*",
    "Effect":"Allow"
```

```
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 部署角色

對於 CodeCatalyst 工作流程動作，您可以建立具有必要許可的 IAM 角色。您可以使用預設CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，或者建立 IAM 角色來部CodeCatalyst署動作以用於 Lambda 部署。此角色使用具有範圍權限的政策，該政 Lambda CodeCatalyst 需要在. AWS 帳戶

此角色授予執行下列作業的權限：

- 讀取、更新和叫用 Lambda 函數和別名。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取 CloudWatch 事件警示的相關資訊。
- 將資訊發佈到 Amazon SNS 主題。

此角色使用下列原則：

```
*{*
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 部署角色

對於 CodeCatalyst 工作流程動作，您可以使用預設CodeCatalystWorkflowDevelopmentRole-*spaceName*服務角色，也可以建立具有必要許可的 IAM 角色。此角色使用具有範圍權限的政策，該政 Lambda CodeCatalyst 需要在. AWS 帳戶

此角色授予執行下列作業的權限：

- 讀取、更新和叫用 Lambda 函數和別名。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。
- 將資訊發佈到 Amazon SNS 主題。

此角色使用下列原則：

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
```

```
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 部署角色 AWS SAM

對於 CodeCatalyst 工作流程動作，您可以使用預設CodeCatalystWorkflowDevelopmentRole-***spaceName***服務角色，也可以建立具有必要許可的 IAM

角色。此角色使用具有範圍權限的策略，該策略 CodeCatalyst 需要AWS CloudFormation在. AWS SAM AWS 帳戶

此角色授予執行下列作業的權限：

- 允許叫 CodeCatalyst 用 Lambda 函數來執行無伺服器和 AWS SAM CLI 應用程式的部署。
- 允許 CodeCatalyst 在中創建和更新堆棧和更改集。AWS CloudFormation

此角色使用下列原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Note

第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。

```
"Resource": "*"
```

CodeCatalyst Amazon EC2 的只讀角色

對於 CodeCatalyst 工作流程動作，您可以建立具有必要許可的 IAM 角色。此角色使用具有範圍許可 CodeCatalyst 的政策，該政策 Amazon EC2 在您的. AWS 帳戶 CodeCatalystWorkflowDevelopmentRole-*spaceName*服務角色不包含 Amazon EC2 的許可或 Amazon 所描述的動作 CloudWatch。

此角色授予執行下列作業的權限：

- 取得 Amazon EC2 執行個體的狀態。
- 取得 Amazon EC2 執行個體的 CloudWatch 指標。

此角色使用下列原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe",
            "Resource": "resource_ARN"
        }
    ]
```

```
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon ECS 的唯讀角色

對於 CodeCatalyst 工作流程動作，您可以建立具有必要許可的 IAM 角色。此角色使用具有範圍許可的政策，該政策 CodeCatalyst 需要在您的 Amazon ECS 資源上執行任務。AWS 帳戶

此角色授予執行下列作業的權限：

- 閱讀 Amazon ECS 任務集。
- 擷取有關 CloudWatch 警報的資訊。

此角色使用下列原則：

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
       {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeRules"
            ],
            "Resource": "resource_ARN",
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam:::role/ecsTaskExecutionRole",
                "arn:aws:iam:::role/ECSTaskExecution"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Lambda 的唯讀角色

對於 CodeCatalyst 工作流程動作，您可以建立具有必要許可的 IAM 角色。此角色使用具有範圍權限的政策，該政 Lambda CodeCatalyst 需要在. AWS 帳戶

此角色授與下列項目的權限：

- 讀取 Lambda 函數和別名。
- 存取 Amazon S3 儲存貯體中的修訂版檔案。
- 擷取有關 CloudWatch 警報的資訊。

此角色使用下列 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
```

```
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

手動建立工作流程動作的角色

CodeCatalyst 工作流程動作使用您建立的 IAM 角色，稱為建置角色、部署角色和堆疊角色。

請依照下列步驟在 IAM 中建立這些角色。

若要建立部署角色

1. 建立角色的策略，執行方式如下：
    a. 登入 AWS。
    b. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
    c. 在導覽窗格中，選擇政策。
    d. 選擇 Create policy (建立政策)。
    e. 請選擇 JSON 標籤。
    f. 刪除現有的代碼。
    g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
```

```
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-deploy-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立部署角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋codecatalyst-deploy-policy並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-deploy-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst deploy role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的部署角色。

3. 取得部署角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-deploy-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的部署角色，並取得其 ARN。

若要建立建置角色

1. 建立角色的策略，執行方式如下：

   a. 登入 AWS。

   b. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

   c. 在導覽窗格中，選擇政策。

   d. 選擇 Create policy (建立政策)。

   e. 請選擇 JSON 標籤。

   f. 刪除現有的代碼。

   g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

   h. 選擇下一步：標籤。

   i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 在 [權限] 原則中，搜尋codecatalyst-build-policy並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-build-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst build role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

3. 取得建置角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-build-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的組建角色，並取得其 ARN。

若要建立堆疊角色

> **Note**
> 您不必建立堆疊角色，雖然出於安全性考量，建議您這麼做。如果您不建立堆疊角色，則需要將此程序中進一步描述的權限原則新增至部署角色。

1. AWS使用您要部署堆疊的帳戶登入。
2. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
3. 在瀏覽窗格中，選擇 [角色]，然後選擇 [建立角色]。
4. 在頂部，選擇AWS服務。
5. 從服務清單中選擇CloudFormation。
6. 選擇 Next: Permissions (下一步：許可)。
7. 在搜尋方塊中，新增存取堆疊中資源所需的任何原則。例如，如果您的堆疊包含AWS Lambda函數，則需要新增授與 Lambda 存取權的政策。

> **Tip**
>
> 如果您不確定要新增哪些原則，可以暫時省略它們。當您測試動作時，如果您沒有正確的權限，AWS CloudFormation會產生錯誤，顯示您需要新增的權限。

8. 選擇下一步：標籤。
9. 選擇下一步：檢閱。
10. 在「角色名稱」中，輸入：

```
codecatalyst-stack-role
```

11. 選擇建立角色。
12. 若要取得堆疊角色的 ARN，請執行下列動作：

    a. 在導覽窗格中，選擇角色。

    b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-stack-role)。

    c. 從清單中選擇角色。

    d. 在 [摘要] 頁面上，複製角色 ARN 值。

用AWS CloudFormation於在 IAM 中建立政策和角色

您可以選擇建立和使用AWS CloudFormation範本來建立所需的策略和角色，以便在 CodeCatalyst 專案和工作流程中存取資源。AWS 帳戶 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，而將更多時間專注於執行的應用程式AWS。如果您打算建立多個角色AWS 帳戶，建立範本可協助您更快地執行此工作。

下列範例範本會建立部署動作角色和原則。

```
Parameters:
  CodeCatalystAccountId:
    Type: String
    Description: Account ID from the connections page
  ExternalId:
    Type: String
    Description: External ID from the connections page
Resources:
  CrossAccountRole:
    Type: 'AWS::IAM::Role'
```

```
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS:
                - !Ref CodeCatalystAccountId
            Action:
              - 'sts:AssumeRole'
            Condition:
              StringEquals:
                sts:ExternalId: !Ref ExternalId
      Path: /
      Policies:
        - PolicyName: CodeCatalyst-CloudFormation-action-policy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - 'cloudformation:CreateStack'
                  - 'cloudformation:DeleteStack'
                  - 'cloudformation:Describe*'
                  - 'cloudformation:UpdateStack'
                  - 'cloudformation:CreateChangeSet'
                  - 'cloudformation:DeleteChangeSet'
                  - 'cloudformation:ExecuteChangeSet'
                  - 'cloudformation:SetStackPolicy'
                  - 'cloudformation:ValidateTemplate'
                  - 'cloudformation:List*'
                  - 'iam:PassRole'
                Resource: '*'
```

手動建立 Web 應用程式藍圖的角色

CodeCatalyst Web 應用程式藍圖使用您建立的 IAM 角色，稱為 CDK 的建置角色、部署角色和堆疊角色。

請依照下列步驟在 IAM 中建立角色。

若要建立建置角色

1. 建立角色的策略，執行方式如下：

a. 登入 AWS。

b. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

c. 在導覽窗格中，選擇政策。

d. 選擇建立政策。

e. 請選擇 JSON 標籤。

f. 刪除現有的代碼。

g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-webapp-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 將權限原則附加至組建角色。在 [新增權限] 頁面的 [權限原則] 區段中，搜尋codecatalyst-webapp-build-policy並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-webapp-build-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst Web app build role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

3. 將權限原則附加至組建角色，如下所示：

a. 在導覽窗格中，選擇 [角色]，然後搜尋codecatalyst-webapp-build-role。

b. 選擇codecatalyst-webapp-build-role顯示其詳細資訊。

c. 在 [權限] 索引標籤中，選擇 [新增權限]，然後選擇 [附加原則]。

d. 搜尋codecatalyst-webapp-build-policy，選取其核取方塊，然後選擇 [附加原則]。

您現在已將權限原則附加至組建角色。組建角色現在有兩個原則：權限原則和信任原則。

4. 取得建置角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-webapp-build-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的組建角色，並取得其 ARN。

手動建立 SAM 藍圖的角色

S CodeCatalyst AM 藍圖使用您建立的 IAM 角色稱為建置角色，以 CloudFormation及 S AM 的部署角色。

請依照下列步驟在 IAM 中建立角色。

若要建立下列項目的建置角色 CloudFormation

1. 建立角色的策略，執行方式如下：

a. 登入 AWS。

b. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

c. 在導覽窗格中，選擇政策。

d. 選擇建立政策。

e. 請選擇 JSON 標籤。

f. 刪除現有的代碼。

g. 貼上以下程式碼：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudformation:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-SAM-build-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

   a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

   b. 選擇 [自訂信任原則]。

   c. 刪除現有的自訂信任原則。

   d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   e. 選擇下一步。

   f. 將權限原則附加至組建角色。在 [新增權限] 頁面的 [權限原則] 區段中，搜尋`codecatalyst-SAM-build-policy`並選取其核取方塊。

   g. 選擇下一步。

   h. 在「角色名稱」中，輸入：

```
codecatalyst-SAM-build-role
```

   i. 對於「角色」描述，請輸入：

```
CodeCatalyst SAM build role
```

   j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

3. 將權限原則附加至組建角色，如下所示：

   a. 在導覽窗格中，選擇 [角色]，然後搜尋codecatalyst-SAM-build-role。

   b. 選擇codecatalyst-SAM-build-role顯示其詳細資訊。

   c. 在 [權限] 索引標籤中，選擇 [新增權限]，然後選擇 [附加原則]。

   d. 搜尋codecatalyst-SAM-build-policy，選取其核取方塊，然後選擇 [附加原則]。

   您現在已將權限原則附加至組建角色。組建角色現在有兩個原則：權限原則和信任原則。

4. 取得建置角色 ARN，如下所示：

   a. 在導覽窗格中，選擇角色。

   b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-SAM-build-role)。

   c. 從清單中選擇角色。

   此時會顯示角色的 [摘要] 頁面。

   d. 在頂端複製 ARN 值。

   您現在已建立具有適當權限的組建角色，並取得其 ARN。

若要建立 SAM 的部署角色

1. 建立角色的策略，執行方式如下：

   a. 登入 AWS。

   b. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

   c. 在導覽窗格中，選擇政策。

   d. 選擇建立政策。

   e. 請選擇 JSON 標籤。

   f. 刪除現有的代碼。

   g. 貼上以下程式碼：

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 第一次使用角色執行工作流程動作時，請在資源策略陳述式中使用萬用字元，然後在策略可用之後使用資源名稱縮小策略的範圍。
>
> ```
> "Resource": "*"
> ```

h. 選擇下一步：標籤。

i. 選擇下一步：檢閱。

j. 在名稱中，輸入：

```
codecatalyst-SAM-deploy-policy
```

k. 選擇建立政策。

您現在已建立權限原則。

2. 建立組建角色，如下所示：

a. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

b. 選擇 [自訂信任原則]。

c. 刪除現有的自訂信任原則。

d. 新增下列自訂信任原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 選擇下一步。

f. 將權限原則附加至組建角色。在 [新增權限] 頁面的 [權限原則] 區段中，搜尋`codecatalyst-SAM-deploy-policy`並選取其核取方塊。

g. 選擇下一步。

h. 在「角色名稱」中，輸入：

```
codecatalyst-SAM-deploy-role
```

i. 對於「角色」描述，請輸入：

```
CodeCatalyst SAM deploy role
```

j. 選擇建立角色。

您現在已建立具有信任原則和權限原則的組建角色。

3. 將權限原則附加至組建角色，如下所示：

a. 在導覽窗格中，選擇 [角色]，然後搜尋codecatalyst-SAM-deploy-role。

b. 選擇codecatalyst-SAM-deploy-role顯示其詳細資訊。

c. 在 [權限] 索引標籤中，選擇 [新增權限]，然後選擇 [附加原則]。

d. 搜尋codecatalyst-SAM-deploy-policy，選取其核取方塊，然後選擇 [附加原則]。

您現在已將權限原則附加至組建角色。組建角色現在有兩個原則：權限原則和信任原則。

4. 取得建置角色 ARN，如下所示：

a. 在導覽窗格中，選擇角色。

b. 在搜尋方塊中，輸入您剛建立的角色名稱 (codecatalyst-SAM-deploy-role)。

c. 從清單中選擇角色。

此時會顯示角色的 [摘要] 頁面。

d. 在頂端複製 ARN 值。

您現在已建立具有適當權限的組建角色，並取得其 ARN。

Amazon 的合規驗證 CodeCatalyst

要瞭解 AWS 服務 是否在特定法規遵循方案範圍內，請參閱法規遵循方案範圍內的 AWS 服務，並選擇您感興趣的法規遵循方案。如需一般資訊，請參閱 AWS 法規遵循方案。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱 AWS Artifact 中的下載報告。

您使用 AWS 服務 時的法規遵循責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理法規遵循事宜：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心的基準環境的架構考量和步驟。
- Amazon Web Services 的 HIPAA 安全與法規遵循架構：本白皮書說明公司可如何運用 AWS 來建立符合 HIPAA 規定的應用程式。

> **Note**
>
> 並非全部的 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 法規遵循資源：這組手冊和指南可能適用於您的產業和位置。
- AWS客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務護指引並對應至安全控制的最佳實務。
- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 此 AWS 服務 可供您全面檢視 AWS 中的安全狀態。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- AWS Audit Manager – 此 AWS 服務 可協助您持續稽核 AWS 使用情況，以簡化管理風險與法規與業界標準的法規遵循方式。

Amazon 的韌性 CodeCatalyst

AWS 全球基礎設施是以 AWS 區域 與可用區域為中心而建置。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。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。

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。若要深入瞭解哪些 CodeCatalyst 資料要複寫AWS 區域，請參閱Amazon 的數據保護 CodeCatalyst。

Amazon 基礎設施安全 CodeCatalyst

作為一項受管服務，Amazon CodeCatalyst 受到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來設計您的 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。

您可以使用AWS已發佈的 API 呼叫透 CodeCatalyst 過網路進行存取。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，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon 中的配置和漏洞分析 CodeCatalyst

組態和 IT 控制是 AWS 與身為我們客戶的您共同的責任。如需詳細資訊，請參閱 AWS 共同的責任模型。

您在 Amazon 的數據和隱私 CodeCatalyst

Amazon 非常重視您的隱私，而您的資訊安全是我們的首 CodeCatalyst 要任務。您可以在AWS隱私聲明中查看更多有關我們如何處理您的信息的信息。

若要請求和檢視您的資料，請參閱AWS 一般參考。

刪除您的AWS產生器 ID 設定檔

刪除您的個人資料是無法復原的永久動作。刪除程序會在您選擇 [刪除] 之後立即開始。Amazon CodeCatalyst 開始刪除您的個人資料和所有相關的個人資訊。此程序最多可能需要 90 天才能完成。

刪除設定檔後，您將無法存取或復原 Amazon 中的資料 CodeCatalyst。這包括個人訪問令牌，角色，用戶成員資格以及您唯一成員的任何 Amazon CodeCatalyst 空間。您無法再登錄 Amazon CodeCatalyst。

如需有關如何刪除您的AWS產生器 ID 設定檔的資訊，請參閱中的刪除您的AWS產生器 ID AWS 一般參考。

Amazon 中工作流程動作的最佳實務 CodeCatalyst

在中開發工作流程時，有許多安全性最佳作法需要考慮 CodeCatalyst。以下是一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

主題

- 敏感資訊
- 授權條款
- 不受信任的代碼
- GitHub 動作

敏感資訊

請勿在 YAML 中內嵌敏感資訊。我們建議您使用機密，而不是在 YAML 中內嵌認證、金 CodeCatalyst 鑰或權杖。機密提供了一種簡單的方法來存儲和引用您的 YAML 中的敏感信息。

授權條款

請務必注意您選擇使用的動作的授權條款。

不受信任的代碼

動作通常是獨立的單一用途模組，可在專案、空間或更廣泛的社群之間共用。使用其他人的代碼可以帶來很大的便利性和效率，但也引入了新的威脅媒介。請檢閱以下各節，確保您遵循最佳做法，以確保 CI/CD 工作流程的安全。

GitHub 動作

GitHub 行動是開源的，由社區構建和維護。我們遵循共同的責任模型，並將 GitHub Actions 源代碼視為您負責的客戶數據。 GitHub 您可以授與動作存取機密、儲存庫權杖、原始程式碼、帳戶連結和您的運算時間。確保您對計劃 GitHub 執行動作的可信度和安全性充滿信心。

更具體的 GitHub 動作指引和安全性最佳做法：

- 安全性強化
- 防止 pwn 請求
- 不受信任的輸入
- 如何信任您的建構區塊

CodeCatalyst 信任模型

Amazon CodeCatalyst 信任模型允 CodeCatalyst 許在連線中扮演服務角色AWS 帳戶。模型會連接 IAM 角色、 CodeCatalyst 服務主體和 CodeCatalyst 空間。信任原則會使用`aws:SourceArn`條件索引鍵，將權限授與條件索引鍵中指定的 CodeCatalyst 空間。如需有關此條件金鑰的詳細資訊，請參閱 IAM 使用者指南SourceArn中的 aws:。

信任原則是 JSON 原則文件，您可以在其中定義您信任擔任該角色的主參與者。角色信任政策是在 IAM 中連接至角色的以資源為基礎的必要政策。如需詳細資訊，請參閱 IAM 使用者指南中的術語和概念。如需有關的服務主參與者的詳細資訊 CodeCatalyst，請參閱下列項目的服務主體 CodeCatalyst。

在下列信任原則中，Principal元素中列出的服務主體會從以資源為基礎的原則授與權限，而Condition區塊則用來限制對縮短資源的存取。

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

在信任政策中， CodeCatalyst 服務主體會透過aws:SourceArn條件金鑰授予存取權，其中包含 CodeCatalyst 空間 ID 的 Amazon 資源名稱 (ARN)。ARN 使用下列格式：

```
arn:aws:codecatalyst:::space/spaceId/project/*
```

> **Important**
>
> 僅在條件索引鍵中使用空格 ID，例如aws:SourceArn。請勿使用 IAM 政策陳述式中的空間 ID 做為資源 ARN。

最佳作法是在原則中盡可能減少權限的範圍。

- 您可以在aws:SourceArn條件鍵中使用萬用字元 (*)，用來指定空間中的所有專案project/*。
- 您可以在空間中的特定項目的aws:SourceArn條件鍵中指定資源級權限。project/*projectId*

下列項目的服務主體 CodeCatalyst

您可以在以資源為基礎的 JSON 政策中使用`Principal`元素來指定允許或拒絕存取資源的主體。您在信任政策中可指定的主體包含使用者、角色、帳戶和服務。您無法在以身分識別為基礎的政策中使用`Principal`元素；同樣地，您無法將使用者群組識別為原則中的主體 (例如以資源為基礎的政策)，因為群組與權限相關，而非驗證，而主體則是經過驗證的 IAM 實體。

在信任策略中，您可以AWS 服務在以資源為基礎的策略的`Principal`元素中或在支援主體的條件索引鍵中指定。服務主體由服務定義。以下是針 CodeCatalyst對下列項目定義的服務主體：

- 代碼催化器。Amazonaws.com-此服務主體用於將授予訪問權限的角色。 CodeCatalyst AWS
- 代碼催化器運行者 .amazonaws.com-此服務主體用於將授予工作流部署資源訪問權限的角色。CodeCatalyst AWS CodeCatalyst

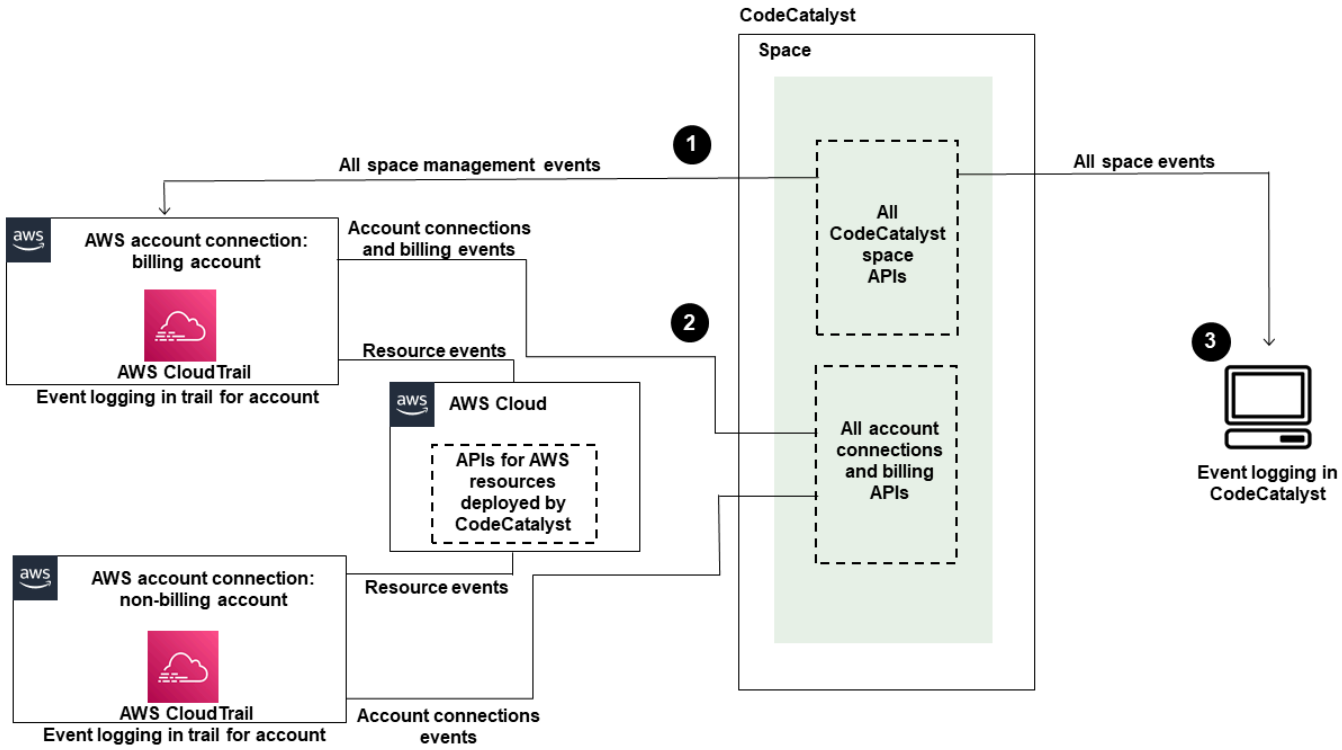
如需詳細資訊，請參閱 IAM 使用者指南中的 AWSJSON 政策元素：主體。

在 Amazon 監控 CodeCatalyst

在 Amazon 中 CodeCatalyst，空間的管理事件是由空間的帳單帳戶收集 AWS CloudTrail 並記錄在追蹤中。 CloudTrail 記錄是管理 CodeCatalyst 事件記錄的主要方法，次要方法是檢視事件登入 CodeCatalyst。

帳戶中的事件會與設定的追蹤和指定值區一起記錄 AWS 帳戶。

下圖顯示如何針對帳單帳戶登入該空間的所有管理事件，同時 CloudTrail 針對個別帳戶登入帳戶連線/帳單事件和 AWS 資源事件。 CloudTrail



此圖說明了下列步驟：

1. 建立空間後，會連 AWS 帳戶 接至空間，並指定為帳單帳戶。使用的追蹤是針對帳單帳戶建立 CloudTrail 的追蹤，其中會記錄空間事件。 CloudTrail 擷取空間或代表 CodeCatalyst 空間發出的 API 呼叫和相關事件，並將日誌檔傳送到您指定的 S3 儲存貯體。如果帳單帳戶變更為另一個 AWS 帳戶，則該帳戶的追蹤和儲存貯體中會記錄空間事件。如需有關由記錄的 CodeCatalyst 管理事件的詳細資訊 CloudTrail，請參閱CodeCatalyst 中的資訊 CloudTrail。
2. 與該空間連線的其他帳戶 (包括帳單帳戶) 會記錄帳戶連線和帳單事件的事件子集。 CodeCatalyst 為該帳戶部署的資 AWS 源產生帳號事件的工作流程也會記錄在的追蹤和值區中 AWS 帳戶。 CloudTrail 擷取空間或代表 CodeCatalyst空間發出的 API 呼叫和相關事件，並將日誌檔傳送到您指定的 S3 儲存貯體。如需有關由記錄的 CodeCatalyst 管理事件的詳細資訊 CloudTrail，請參閱存取記錄的事件 CodeCatalyst。
3. 您也可以使用list-event-logs指令，在空間中的特定時間內監視空間中的 CodeCatalyst 動作 AWS CLI。如需詳細資訊，請參閱 Amazon CodeCatalyst API 參考指南。您必須具有 S pace 管理員角色，才能呼叫空間中 CodeCatalyst 動作的事件清單。如需詳細資訊，請參閱 存取記錄的事件 CodeCatalyst。

> **Note**
>
> `ListEventLogs`保證指定空間中過去 30 天的事件。您也可以檢視事件歷史記錄，或建立追蹤以建立和維護超過 90 天的事件記錄， CodeCatalyst 在 AWS CloudTrail 主控台中檢視和擷取過去 90 天內的管理事件清單。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄和使用 CloudTrail 追蹤。

> **Note**
>
> AWS 針對 CodeCatalyst 工作流程部署到連線帳戶中的資源，不會記錄為 CodeCatalyst 空間 CloudTrail 記錄的一部分。例如， CodeCatalyst 資源包括空間或專案。 AWS 資源包括 Amazon ECS 服務或 Lambda 函數。您必須為每個資源部署到的 AWS 帳戶 位置分別配置 CloudTrail 記錄。

以下是中事件監視的一個可能流程 CodeCatalyst。

Mary Major 是空間的管理員，可以檢視中所有的管理事件，以 CodeCatalyst 取得登入的空間中的空間層級和專案層級資源。 CodeCatalyst CloudTrail如CodeCatalyst 中的資訊 CloudTrail需已登入的事件範例，請參閱 CloudTrail。

對於在中 CodeCatalyst建立的資源 (例如開發環境)，Mary 會檢視空間的帳單帳戶中的事件歷史記錄，並調查由中 CodeCatalyst的專案成員建立開發環境的事件。此事件會為建立開發環境的使用者提供識別存放區 IAM 身分類型和認證的 AWS Builder ID。對於在中的工作流程部署 AWS 時所建立的 CodeCatalyst資源 (例如無伺服器部署的 Lambda 函數)， AWS 帳戶 擁有者可以檢視與工作流程部署動作的個別追蹤相關聯 AWS 帳戶 (也是連線帳戶 CodeCatalyst) 的事件歷史記錄。

若要進一步調查，Mary 也可以使用中的list-event-logs命令檢視空間中所有 CodeCatalyst API 的事件 AWS CLI。

主題

- AWS 帳戶使用連線記錄 CodeCatalyst API 呼叫 AWS CloudTrail
- 存取記錄的事件 CodeCatalyst

AWS 帳戶使用連線記錄 CodeCatalyst API 呼叫 AWS CloudTrail

Amazon CodeCatalyst 與服務整合AWS CloudTrail，可提供使用者、角色或使用者所採取的動作記錄 AWS 服務。 CloudTrail 捕獲AWS 帳戶作為事件連接的 CodeCatalyst 代表進行的 API 調用。如果您建立追蹤，您可以啟用 CloudTrail 事件持續傳遞至 S3 儲存貯體，包括 CodeCatalyst. 如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。

CodeCatalyst 支援將下列動作記錄為記 CloudTrail 錄檔中的事件：

- CodeCatalyst 空間的管理事件將記錄AWS 帳戶在空間的指定帳單帳戶中。如需詳細資訊，請參閱CodeCatalyst 太空事件。

  > **Note**
  >
  > CodeCatalyst 空間的資料事件可透過使用 CLI 存取，如中所述存取記錄的事件 CodeCatalyst。

- 發生在連接的 CodeCatalyst 工作流程動作中使用的資源的事件AWS 帳戶將記錄為其中的事件AWS 帳戶。如需詳細資訊，請參閱CodeCatalyst 帳戶連線和帳單事件。

> **Important**
>
> 雖然多個帳戶可以與一個空間相關聯，但 CodeCatalyst 空間和專案中的事件 CloudTrail 記錄僅適用於帳單帳戶。

太空計費帳戶是您針AWS 帳戶對免費方案以外的 CodeCatalyst 資源所支付的AWS費用。多個帳戶可以連接到一個空間，而只有一個帳戶可以是指定的帳單帳戶。空間的帳單帳戶或其他連線帳戶可以具有用於從 CodeCatalyst 工作流程部署AWS資源和基礎設施 (例如 Amazon ECS 叢集或 S3 儲存貯體) 的 IAM 角色。您可以使用工作流程 YAML 來識別您部署的AWS 帳戶目標。

> **Note**
>
> AWS針對 CodeCatalyst 工作流程部署到連線帳戶中的資源，不會記錄為 CodeCatalyst 空間 CloudTrail 記錄的一部分。例如， CodeCatalyst資源包括空間或專案。 AWS資源包括 Amazon ECS 服務或 Lambda 函數。 CloudTrail 必須針對資源部署到的每個AWS 帳戶位置單獨配置記錄。

CodeCatalyst 登入連線的帳戶包括下列考量事項：

- CloudTrail 事件的存取權限是透過連線帳戶中的 IAM 管理，而不是在中 CodeCatalyst。
- 第三方連線 (例如連結至 GitHub 儲存庫) 會導致協力廠商資源名稱記錄在記 CloudTrail 錄中。

> **Note**
>
> CloudTrail CodeCatalyst 事件記錄位於空間層級，不會依專案邊界隔離事件。

若要取得有關的更多資訊 CloudTrail，請參閱AWS CloudTrail使用者指南。

> **Note**
>
> 本節說明 CloudTrail 記錄在某個 CodeCatalyst空間中記錄的所有事件，以及連線到的事件 CodeCatalyst。AWS 帳戶此外，若要檢閱 CodeCatalyst 空間中記錄的所有事件，您也可以使用AWS CLI和指aws codecatalyst list-event-logs令。如需詳細資訊，請參閱存取記錄的事件 CodeCatalyst。

CodeCatalyst 太空事件

中 CodeCatalyst 用於管理空間層級和專案層級資源的動作會記錄在空間的帳單帳戶中。對於 CodeCatalyst 空間的 CloudTrail 記錄，會記錄事件，並考量下列事件。

- CloudTrail 事件適用於整個空間，並且不限於任何單一專案。
- 當您連接AWS 帳戶到 CodeCatalyst 空間時，帳戶連接的可記錄事件將被記錄在該空間中。AWS 帳戶啟用此連線後，就無法停用它。
- 當您連接AWS 帳戶到 CodeCatalyst 空間並將其指定為該空間的帳單帳戶時，會在該空間中記錄事件AWS 帳戶。啟用此連線後，就無法停用它。

  空間層級和專案層級資源的事件只會記錄在帳單帳戶中。若要變更 CloudTrail 目的地帳戶，請在中更新帳單帳戶 CodeCatalyst。在下個月計費週期開始時，變更會在中的新帳單帳戶生效 CodeCatalyst。之後， CloudTrail目標帳戶就會更新。

以下是中AWS與管理空間層級和專案層級資源 CodeCatalyst的動作相關的事件範例。下列 API 是透過 SDK 和 CLI 發行的。事件將記錄在AWS 帳戶指定為 CodeCatalyst 空間的帳單帳戶中。

- CreateDevEnvironment
- CreateProject
- DeleteDevEnvironment
- GetDevEnvironment
- GetProject
- GetSpace
- GetSubscription
- ListDevEnvironments
- ListDevEnvironmentSessions
- ListEventLogs
- ListProjects
- ListSourceRepositories
- StartDevEnvironment
- StartDevEnvironmentSession
- StopDevEnvironment
- StopDevEnvironmentSession
- UpdateDevEnvironment

CodeCatalyst 帳戶連線和帳單事件

以下是中AWS與帳戶連線或帳單動作相關的事件範例： CodeCatalyst

- AcceptConnection
- AssociateIAMRoletoConnection
- DeleteConnection
- DissassociateIAMRolefromConnection
- GetBillingAuthorization
- GetConnection
- GetPendingConnection
- ListConnections
- ListIAMRolesforConnection

- PutBillingAuthorization
- RejectConnection

CodeCatalyst 中的資訊 CloudTrail

CloudTrail 在您建立該帳戶AWS 帳戶時啟用。當您將該空間連接AWS 帳戶到 CodeCatalyst 空間時，發生在該空間的事件會記錄AWS 帳戶在該 AWS 帳戶的 CloudTrail 日誌中。中的可登入事件會記錄 CodeCatalyst 為 CloudTrail 連線帳戶的記錄 CloudTrail 檔和 CloudTrail 主控台的事件歷程記錄中的事件，以及該帳戶中的其他可登入AWS事件。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 請求是否由具有其AWS生成器 ID 的用戶提出。
- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

存取 CloudTrail 事件

若要進行中的AWS 帳戶事件記錄 (包括中 CodeCatalyst活動的事件)AWS 帳戶，請建立追蹤。線索可讓 CloudTrail 將日誌檔案傳遞到 S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail 日誌文件

追蹤是一種組態，可讓事件以日誌檔的形式傳遞至您指定的 S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參

數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

範例中的 CodeCatalyst 帳戶連線事件 AWS

下列範例顯示示範`ListConnections`動作的 CloudTrail 記錄項目。對AWS 帳戶於連接到空間的，用`ListConnections`於查看與此的所有帳戶連接 CodeCatalyst AWS 帳戶。事件將記錄在中AWS 帳戶指定的`accountId`，且的值`arn`將是用於動作之角色的 Amazon 資源名稱 (ARN)。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "role-ARN",
        "accountId": "account-ID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "arn": "role-ARN",
                "accountId": "account-ID",
                "userName": "user-name"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-09-06T15:04:31Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-09-06T15:08:43Z",
    "eventSource": "account-ID",
    "eventName": "ListConnections",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.168.0.1",
    "userAgent": "aws-cli/1.18.147 Python/2.7.18 Linux/5.4.207-126.363.amzn2int.x86_64
 botocore/1.18.6",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
```

```
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-ID",
    "eventCategory": "Management"
}
```

範例 CodeCatalyst 專案資源事件 AWS

下列範例顯示示範CreateDevEnvironment動作的 CloudTrail 記錄項目。連接到空間並且是空間的指定帳單帳戶，用於空間中的專案層級事件，例如建立開發環境。AWS 帳戶

在accountId欄位中的下userIdentity方，這是主控所有 AWS Builder ID 身分識別身分識別身分識別的身分集區的 IAM 身分識別中心帳戶 ID (432677196278)。此帳號 ID 包含下列有關事件 CodeCatalyst 使用者的資訊。

- 此type欄位會指出要求的 IAM 實體類型。對於空間和專案資源的 CodeCatalyst 事件，此值為IdentityCenterUser。此accountId欄位會指定擁有用來取得認證之實體的帳戶。
- 該userId字段包含用戶的AWS生成器 ID 標識符。
- 此identityStoreArn欄位包含識別身分存放區帳戶和使用者的角色 ARN。

此recipientAccountId欄位包含空間帳單帳戶的帳戶識別碼，此處的範例值為 111122223333。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "IdentityCenterUser",
  "accountId": "432677196278",
  "onBehalfOf": {
   "userId": "user-ID",
   "identityStoreArn": "arn:aws:identitystore::432677196278:identitystore/d-9067642ac7"
  },
  "credentialId": "ABCDefGhiJKLMn11Lmn_1AbCDEFgHijk-AaBCdEFGHIjKLmnOPqrs11abEXAMPLE"
 },
 "eventTime": "2023-05-18T17:10:50Z",
 "eventSource": "codecatalyst.amazonaws.com",
 "eventName": "CreateDevEnvironment",
```

```
 "awsRegion": "us-west-2",
 "sourceIPAddress": "192.168.0.1",
 "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101
 Firefox/102.0",
 "requestParameters": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
  "ides": [{
   "runtime": "public.ecr.aws/q6e8p2q0/cloud9-ide-runtime:2.5.1",
   "name": "Cloud9"
  }],
  "instanceType": "dev.standard1.small",
  "inactivityTimeoutMinutes": 15,
  "persistentStorage": {
   "sizeInGiB": 16
  }
 },
 "responseElements": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 "
 },
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "readOnly": false,
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "eventCategory": "Management"
}
```


> **Note**
>
> 在某些事件中，用戶代理可能不知道。在這種情況下， CodeCatalyst 將在 CloudTrail 事件Unknown中的userAgent字段中提供的值。

查詢您的 CodeCatalyst 事件追蹤

您可以使用 Amazon Athena 中的查詢表建立和管理 CloudTrail 日誌的查詢。如需有關建立查詢的詳細資訊，請參閱 Amazon Athena 使用者指南中的查詢AWS CloudTrail日誌。

存取記錄的事件 CodeCatalyst

當使用者在 Amazon 中執行動作時 CodeCatalyst，這些動作會記錄為事件。您可以使 AWS CLI 用檢視指定時間範圍內空間中的事件記錄。您可以檢視這些事件以檢閱在空間中採取的動作，包括動作的日期和時間、執行動作的使用者名稱，以及使用者提出要求的 IP 位址。

> **Note**
>
> 已連線的帳單帳戶會登入 CodeCatalyst 空間 CloudTrail 的管理事件。如需有關由記錄的 CodeCatalyst 管理事件的詳細資訊 CloudTrail，請參閱CodeCatalyst 中的資訊 CloudTrail。

若要檢視空間的事件記錄，您必須已安裝並設定 AWS CLI 的設定檔 CodeCatalyst，且您必須具有該空間的 Space 管理員角色。如需詳細資訊，請參閱 設定以使用AWS CLI與 CodeCatalyst 及 空間管理員角色。

> **Note**
>
> 若要檢視代表連線 CodeCatalyst 中發生的事件記錄 AWS 帳戶，或檢視連線帳戶中空間或專案資源的事件記錄，您可以使用 AWS CloudTrail。如需詳細資訊，請參閱 AWS 帳戶使用連線記錄 CodeCatalyst API 呼叫 AWS CloudTrail。

1. 開啟終端機或命令列並執行指aws codecatalyst list-event-logs令，並指定：

   - 帶有**--space-name**選項的空間名稱。
   - 您要開始審核事件的日期和時間，採用 RFC 3339 中指定的協調世界時間 (UTC) 時間戳記格式，並提供選項。**--start-time**
   - 您想要停止審核事件的日期和時間，採用 RFC 3339 中指定的協調世界時間 (UTC) 時間戳記格式，並提供選項。**--end-time**
   - (選擇性) 在單一回應中傳回的結果數目上限 (含選**--max-results**項)。如果結果數量大於您指定的數目，則響應將包含一個nextToken元素，您可以使用該元素返回下一個結果。
   - (選擇性) 使用選**--event-name**項，將結果限制為您要傳回的特定事件類型。

   此範例會傳回*ExampleCorp*從 *2022-11-30 # 2022- 12-01* 期間命名的空間中的記錄事件，並在回應中傳回最多 *2* 個事件。

```
aws codecatalyst list-event-logs --space-name ExampleCorp --start-time 2022-11-30
 --end-time 2022-12-01 --event-name list-event-logs --max-results 2
```

2. 如果事件發生在此時間範圍內，命令會傳回類似下列內容的結果：

```
{
    "nextToken": "EXAMPLE",
    "items": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "eventName": "listEventLogs",
            "eventType": "AwsApiCall",
            "eventCategory": "MANAGEMENT",
            "eventSource": "manage",
            "eventTime": "2022-12-01T22:47:24.605000+00:00",
            "operationType": "READONLY",
            "userIdentity": {
                "userType": "USER",
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111"
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"startTime\":
\"2022-12-01T00:00:00Z\",\"endTime\":\"2022-12-10T00:00:00Z\",\"maxResults\":
\"2\"}"
            },
            "sourceIpAddress": "127.0.0.1",
            "userAgent": "aws-cli/2.9.0 Python/3.9.11 Darwin/21.3.0 exe/x86_64
 prompt/off command/codecatalyst.list-event-logs"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "eventName": "createProject",
            "eventType": "AwsApiCall",
            "eventCategory": "MANAGEMENT",
            "eventSource": "manage",
            "eventTime": "2022-12-01T09:15:32.068000+00:00",
            "operationType": "MUTATION",
            "userIdentity": {
                "userType": "USER",
```

```
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111",
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\"}"
            },
            "responsePayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\",\"id\":\"a1b2c3d4-5678-90ab-cdef-
EXAMPLE4444\"}"
            },
            "sourceIpAddress": "192.0.2.23",
            "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0)
 Gecko/20100101 Firefox/102.0"
        }
    ]
}
```


3. 使用--next-token選項和傳回的 Token 值再次執行list-event-logs命令，以擷取下一組符合要求的記錄事件。

中的身分識別、權限和存取配額 CodeCatalyst

下表說明 Amazon 中身分識別、許可和存取的配額和限制 CodeCatalyst。如需 Amazon 中配額的詳細資訊 CodeCatalyst，請參閱的配額 CodeCatalyst。

資源	資訊
別名 CodeCatalyst	長度介於 3 到 100 個字元之間且必須以字母開頭的任何允許字元組合。有效字元：A-Z、a-z 和 0-9。別名不能： • 包含少於 3 個字符 • 包含空格或以下任何字元：? ^ * [\ ~ :
使用者每天傳送的邀請數目上限	500

資源	資訊
每天傳送至電子郵件地址的邀請數上限	25
每位使用者的個人存取權杖 (PAT) 數目上限	100
密碼 CodeCatalyst	長度介於 8 到 64 個字元之間的任何允許字元組合。有效字元：A-Z、a-z 和 0-9。您的密碼可以包含下列非英數字元：(~ ! @ # $ % ^ & * _ - + = ` | \ { } [ ] : ; " ' < > , . ? /)
帕特名稱 CodeCatalyst	1 到 100 個字元之間允許的任何字元組合
專案成員邀請到期的時間	24 小時後到期
空間成員邀請到期前的時間	24 小時後到期
直到電子郵件地址驗證到期的時間	發送後 10 分鐘到期

故障診斷

本節可協助您解決存取 Amazon 個 CodeCatalyst 人檔案時可能遇到的一些常見問題。

註冊時遇到問題

註冊時可能會遇到一些問題。我們有一些解決方案。

我的電子郵件地址已在使用

如果您輸入的電子郵件已在使用中，並且您將其識別為您自己的電子郵件，那麼您可能已經擁有了我們的個人資料。使用此現有身分登入。如果您不擁有現有的電子郵件，請使用其他未使用的電子郵件進行註冊。

我無法完成電子郵件驗證

如果您尚未收到驗證電子郵件

1. 檢查垃圾郵件、垃圾郵件和已刪除郵件資料夾。

> **Note**
>
> 此驗證電子郵件來自地址no-reply@signin.aws或no-reply@login.awsapps.com。我們建議您設定您的郵件系統，使其接受來自這些寄件者電子郵件地址的電子郵件，並且不會將其視為垃圾郵件或垃圾郵件處理。

2. 等待 5 分鐘，然後刷新您的收件箱。再次檢查您的垃圾郵件，垃圾郵件和已刪除郵件文件夾。
3. 如果仍然看不到驗證電子郵件，請選擇「重新傳送驗證碼」。如果您已經退出該頁面，請重新啟動您的工作流程以便在 Amazon CodeCatalyst 註冊。

我的密碼不符合最低要求

為了安全起見，您的密碼必須包含 8-20 個字符，包括大寫和小寫字母以及數字。

登入時發生問題

我忘記了我的密碼

請遵循 我忘記了密碼 中的步驟。

我的密碼無法使用

每當您設定或變更密碼時，您都必須遵循下列要求：

- 密碼區分大小寫。
- 密碼的長度必須介於 8 到 64 個字元之間，包括大寫和小寫字母、數字，以及至少一個非英數字元。
- 您不能重複使用最後三個密碼 。

我無法啟用 MFA

若要啟用 MFA，請依照中的步驟將一或多個 MFA 裝置新增至您的設定檔。Amazon 中的多因素身份驗證（MFA） CodeCatalyst

我無法新增 MFA 裝置

如果您發現無法新增其他 MFA 裝置，則該裝置可能已達到您可以註冊的 MFA 裝置上限。您可能需要先移除現有的 MFA 裝置，然後再新增一個 MFA 裝置。

我無法移除 MFA 裝置

如果您打算停用 MFA，請依照中的步驟繼續移除 MFA 裝置。刪除 MFA 裝置但是，如果您要保持 MFA 啟用狀態，則應先新增另一個 MFA 裝置，然後再嘗試刪除現有的 MFA 裝置。如需新增其他 MFA 裝置的詳細資訊，請參閱如何註冊設備以使用多因素身份驗證。

登出時發生問題

我找不到要登出的地方

在頁面右上角，選擇 [登出]。

「登出」不會完全將我登出

系統的設計可立即登出，但完全登出最多可能需要一個小時。

我得到一個失敗的工作流程的角色不存在錯誤

問題：從 Web 應用程式或無伺服器藍圖建立專案後，工作流程會失敗，並出現下列錯誤：

用戶端錯誤：角色不存在

可能的解決方案：設定具有執行工作流程許可的 IAM 角色之後，並且已將 IAM 角色新增至工作流程 YAML，工作流程仍然失敗，因為可能需要將 IAM 角色新增至您的帳戶連線。將 IAM 角色新增至您空間的帳戶連線，如中所述將 IAM 角色新增至帳戶連線。

我收到失敗工作流程的角色錯誤

問題：從 Web 應用程式或無伺服器藍圖建立專案後，工作流程會失敗，並出現下列錯誤：

客戶端錯誤：角色設置不正確或不存在

可能的解決方案：建立專案的空間可能需要設定 AWS 帳戶 連線，或者可能需要完成帳戶連線要求。如果您的空間已有作用中 AWS 帳戶 連線，請建立並新增具有執行工作流程動作許可的 IAM 角色。將 IAM 角色新增至您的帳戶連線，如中所述將 IAM 角色新增至帳戶連線。

可能的解決方案：如果專案建立時未指定連線，則帳戶連線必須與部署環境建立關聯。如果您的空間已有作用中 AWS 帳戶 連線並新增 IAM 角色，則必須將具有 IAM 角色的帳戶連線新增至部署環境，如中所述將帳戶連線和 IAM 角色新增至您的部署環境。

我需要更新專案工作流程中的 IAM 角色

如果 AWS 帳戶 連線已完全設定，且 IAM 角色已建立並新增至帳戶連線，您可以在專案工作流程中更新 IAM 角色。

1. 選擇 CI/CD 選項，然後選擇您的工作流程。選擇 YAML 按鈕。
2. 選擇編輯。
3. 在`ActionRoleArn:`欄位中，將 IAM 角色 ARN 取代為更新後的 IAM 角色 ARN。選擇「驗證」。
4. 選擇 Commit (遞交)。

   如果在主線分支上，工作流程會自動啟動。否則，若要重新執行工作流程，請選擇「執行」。

如何填寫支援表格？

你可以去 Amazon CodeCatalyst 或填寫 Sup port 反饋表。在「請求信息」部分的「我們如何為您提供幫助」下，包括您是 Amazon CodeCatalyst 客戶。盡可能提供詳細資訊，以便我們能夠最有效地解決您的問題。

中的擴展 CodeCatalyst

Amazon CodeCatalyst 包含可協助您新增功能並與外部產品整合的擴充功能 CodeCatalyst。使用 CodeCatalyst 目錄中的擴充功能，團隊可以在中自訂其體驗 CodeCatalyst。

主題

擴展概念

以下是在中使用擴充功能時應瞭解的一些概念和術語 CodeCatalyst。

延伸模組

擴充功能是一種附加元件，您可以將其安裝到您的 CodeCatalyst 空間中，以便為您的專案新增功能，並與外部的服務整合 CodeCatalyst。您可以從 CodeCatalyst 目錄中瀏覽和安裝擴充功能。

CodeCatalyst 目錄

目 CodeCatalyst 錄是中所有可用擴充功能的集中清單 CodeCatalyst。您可以瀏覽目 CodeCatalyst 錄以找到可以改善團隊在來源、工作流程 CodeCatalyst 等領域體驗的擴充功能。

安裝和解除安裝擴充功能 CodeCatalyst

擴充功能是一種附加元件，您可以將其安裝到您的 CodeCatalyst 空間中，以便為您的專案新增功能，並與外部的服務整合 CodeCatalyst。您可以從 CodeCatalyst 目錄瀏覽和安裝擴充功能。

主題

安裝擴充功能

您可以為您的 CodeCatalyst 空間安裝擴充功能，為該空間中的專案新增功能。您可以選擇「 CodeCatalyst 目錄」圖示來檢視目錄 。

> **Important**
> 若要安裝擴充功能，您必須使用在空間中具有 S pace 管理員角色的帳戶登入。

若要從 CodeCatalyst 目錄安裝擴充功能

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。
3. 選擇頂端功能表 中的「目錄」圖示，以導覽至目錄。 CodeCatalyst 您可以根據類別搜尋副檔名或篩選擴充功能。
4. (選擇性) 選擇擴充功能的名稱，以查看有關擴充功能的詳細資訊，例如擴充功能將擁有的權限。
5. 選擇 Install (安裝)。檢閱擴充功能所需的權限，如果您要繼續，請再次選擇 [安裝]。

安裝擴展程序後，您將看到已安裝擴展程序的詳細信息頁面。瀏覽選項卡以獲取有關擴展程序的更多信息。如果需要，詳細信息頁面也是您將在其中執行擴展的進一步配置。

卸載擴展

您可以解除安裝先前安裝在您的 CodeCatalyst 空間中的擴充功能。解除安裝擴充功能可能會從您的 CodeCatalyst 空間或專案移除與該擴充功能相關的資源。

> **Important**
> 若要解除安裝擴充功能，您必須使用在空間中具有 S pace 管理員角色的帳戶登入。

若要從您的 CodeCatalyst 空間解除安裝擴充功能

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。

2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您的空間已安裝擴充功能的清單：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

      單  中

      的目錄圖標。
4. 在您要解除安裝的擴充功能上選擇 [設定]。
5. 在擴充功能詳細資料頁面上選擇解除安
6. 檢閱 [解除安裝擴充功能] 對話方塊中的資訊。依照指示操作，然後選擇 [解除安裝] 以解除安裝擴充功能。

在中使用 GitHub 儲存庫 CodeCatalyst

GitHub 是一種基於雲的服務，可幫助開發人員存儲和管理其代碼。GitHub 儲存庫擴充功能可讓您在 Amazon CodeCatalyst 專案中使用連結的 GitHub 儲存庫。您也可以在建立新 CodeCatalyst 專案時連結 GitHub 儲存庫。如需詳細資訊，請參閱 建立具有連結 GitHub儲存庫的專案。

> **Note**
>
> 您無法在 CodeCatalyst 專案中使用空白或封存的 GitHub 儲存庫。GitHub 儲存庫擴充功能與 GitHub 企業伺服器不相容。

安裝並配置GitHub 存儲庫擴展後，您將能夠：

- 在以下位置的來源 GitHub 儲存庫清單中檢視您的儲存庫 CodeCatalyst
- 在儲存庫中 GitHub 儲存和管理工作流程定義檔案。
- 從 CodeCatalyst 開發環境建立、讀取、更新和刪除儲存在連結儲存 GitHub 庫中的檔案
- 當 CodeCatalyst 程式碼推送至 GitHub 儲存庫時，啟動工作流程自動執行
- 在 CodeCatalyst 工作流程中使用連結的 GitHub 儲存庫來源檔
- 在 CodeCatalyst 工作流程中讀取和 GitHub 執行動作

下列主題說明如何安裝和設定GitHub 儲存庫擴充功能，以及如何在 GitHub 儲存庫連結至您的 CodeCatalyst 專案時使用儲存庫。

主題

快速入門：在中使用 GitHub 儲存庫 CodeCatalyst

執行以下步驟來安裝GitHub 儲存庫擴充功能、連線到您的 GitHub 帳戶，以及將 GitHub 儲存庫連結至現有 CodeCatalyst 專案。

您還可以在創建新項目時安裝GitHub 存儲庫擴展，連接到您的 GitHub 帳戶，鏈接 GitHub 存儲 CodeCatalyst 庫。如需詳細資訊，請參閱 建立具有連結 GitHub儲存庫的專案。

內容

步驟 1：從 CodeCatalyst 目錄安裝 GitHub 擴充功能

在中使用 GitHub 儲存庫的第一步 CodeCatalyst 是從 CodeCatalyst 目錄安裝GitHub 儲存庫擴充功能。要安裝擴展程序，請執行以下步驟，選擇GitHub 存儲庫擴展。

> **Important**
>
> 作為安裝和配置GitHub 存儲庫擴展的一部分，您必須在 GitHub 帳戶中安裝擴展程序。若要這麼做，您必須同時是 GitHub 帳戶管理員和 S CodeCatalyst pace 管理員。

若要從 CodeCatalyst 目錄安裝擴充功能

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。
3. 選擇頂端功能

   表 中
   的「目錄」圖示，以導覽至目錄。 CodeCatalyst CodeCatalyst 您可以搜索GitHub 存儲庫或根據類別過濾擴展。
4. (選擇性) 若要查看有關擴充功能的更多詳細資訊，例如擴充功能將擁有的權限，請選擇GitHub 儲存庫擴充功能名稱。
5. 選擇 Install (安裝)。檢閱擴充功能所需的權限，如果您要繼續，請再次選擇 [安裝]。

安裝GitHub 儲存庫擴充功能之後，您會前往GitHub 儲存庫擴充功能詳細資料頁面，您可以在此檢視和管理連線的 GitHub 帳戶和連結的 GitHub 儲存庫。

步驟 2：將您的 GitHub 帳戶 Connect 到您的 CodeCatalyst 空間

安裝GitHub 儲存庫擴充功能後，下一步就是將您的 GitHub 帳戶連線到您的 CodeCatalyst 空間。

> **Important**
>
> 若要將 GitHub 帳戶連線到您的 CodeCatalyst 空間，您必須同時是 GitHub 帳戶管理員和 S CodeCatalyst pace 管理員。

將您的 GitHub 帳戶連接到 CodeCatalyst

1. 在 [已連線的 GitHub 帳戶] 索引標籤中，選擇 [Connect GitHub 帳戶] 以移至其外部網站 GitHub。
2. 使用您 GitHub 的 GitHub 憑據登錄到您的帳戶，然後選擇要安裝 Amazon 的帳戶 CodeCatalyst。

> **Tip**
>
> 如果您之前已將 GitHub 帳戶連接到其他空間，則不會提示您重新授權。如果您是多個帳戶的成員或協作者，則會看到一個對話方塊，詢問您要在何處安裝擴充功能；如果您只屬於一個 GitHub 帳戶，則會看到 Amazon CodeCatalyst 應用程式的設定頁面。 GitHub 針

對您要允許的存放庫存取設定應用程式，然後選擇 [儲存]。如果中沒有使用「儲存」按鈕 GitHub，請變更組態，然後再試一次。

3. 選擇是否允許存 CodeCatalyst 取所有目前和 future 的儲存庫，或選擇要在其中使用的特定 GitHub 儲存庫 CodeCatalyst。預設選項是 GitHub 帳戶中的所有 GitHub 儲存庫。
4. 檢閱授予的權限 CodeCatalyst，然後選擇 [安裝]。

連接到您的 GitHub 帳戶後 CodeCatalyst，您可以在GitHub 存儲庫擴展詳細信息頁面的GitHub 帳戶選項卡中查看已連接的帳戶。

第 3 步：將 GitHub 存儲庫鏈接到 CodeCatalyst 項目

在中使用儲存 GitHub 庫的第三個也是最後一步 CodeCatalyst 是將存放庫連結至您要在其中使用它的 CodeCatalyst 專案。

Important

雖然您可以以參與者的身份連結 GitHub 存放庫，但您只能以 S pace 管理員或專案管理員的身分取消 GitHub 存放庫的連結。如需詳細資訊，請參閱 取消 GitHub 儲存庫與專案的 CodeCatalyst 連結。

若要從儲 GitHub 存庫擴充功能詳細資訊頁面將存放 GitHub 庫連結至 CodeCatalyst 專案

1. 在「連結的 GitHub 儲存庫」頁籤中，選擇「連結儲 GitHub 存庫
2. 從GitHub 帳戶下拉式功能表中，選擇包含您要連結之存放庫的 GitHub 帳戶。
3. 從GitHub 存放庫下拉式功能表中，選擇要連結至 CodeCatalyst 專案的存放庫。

Tip

如果儲存庫的名稱顯示為灰色，則無法連結該儲存庫，因為該儲存庫已連結至空間中的另一個專案。

4. 從CodeCatalyst 專案下拉式功能表中，選擇您要連結 GitHub 存放庫的 CodeCatalyst 專案。
5. 選擇 Link (連結)。

您也可以從程式碼中的原始碼儲 GitHub 存庫將儲存庫連結至專案。如需詳細資訊，請參閱 管理 GitHub 儲存庫 CodeCatalyst。

後續步驟

安裝GitHub 儲存庫擴充功能、連接您的 GitHub 帳戶，並將 GitHub 儲存庫連結至 CodeCatalyst 專案之後，您就可以在 CodeCatalyst 工作流程和開發環境中使用它。如需更多詳細資訊，請參閱 在 CodeCatalyst 工作流中使用 GitHub 儲存 及 建立開發環境。

管理 GitHub 帳戶 CodeCatalyst

若要在中使用 GitHub 儲存庫 CodeCatalyst，您必須將 GitHub 帳戶連線至您的 CodeCatalyst 空間。如果您不想再在中使用帳戶的儲存庫 CodeCatalyst，您可以中斷該 GitHub 帳戶的連線。當帳戶中斷連線時，帳戶儲存庫中的事件將不會啟動工作流程執行，而且您將無法在開 CodeCatalyst 發環境中使用這些存放庫。

> **Important**
>
> 若要連線或中斷連線您的 GitHub 帳戶和 CodeCatalyst 空間，您必須同時是 GitHub 帳戶管理員和 S CodeCatalyst pace 管理員。

內容

將 GitHub 帳戶 Connect 到 CodeCatalyst 空間

執行以下步驟將 GitHub 帳戶連線到您的 CodeCatalyst 空間。

將您的 GitHub 帳戶連接到 CodeCatalyst

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝GitHub 儲存庫擴充功能：

    a. 選擇 [設定]，然後選擇 [安裝的擴充功能

b. 選擇頂部菜單  中的目錄圖標。

4. 在GitHub 儲存庫中，選擇設定。
5. 在 [已連線的 GitHub 帳戶] 索引標籤中，選擇 [Connect GitHub 帳戶] 以移至其外部網站 GitHub。
6. 使用您 GitHub 的 GitHub 憑據登錄到您的帳戶，然後選擇要安裝 Amazon 的帳戶 CodeCatalyst。

> **Tip**
>
> 如果您先前已將 GitHub 帳戶連線到空間，則不會提示您重新授權。如果您是多個空間的成員或協作者，則會看到一個對話方塊，詢問您要在何處安裝擴充功能；如果您只屬於一個 GitHub空間，則會看到 Amazon CodeCatalyst 應用程式的設定頁面。 GitHub 針對您要允許的存放庫存取設定應用程式，然後選擇 [儲存]。如果 [儲存] 按鈕未啟用，請變更組態，然後再試一次。

7. 選擇是否允許存 CodeCatalyst 取所有目前和 future 的儲存庫，或選擇要在其中使用的特定 GitHub 儲存庫 CodeCatalyst。預設選項是 GitHub 空間中的所有 GitHub 儲存庫。
8. 檢閱授予的權限 CodeCatalyst，然後選擇 [安裝]。

安裝GitHub 儲存庫擴充功能之後，您會前往GitHub 儲存庫擴充功能詳細資料頁面，您可以在此檢視和管理連線的 GitHub 帳戶和連結的 GitHub 儲存庫。

在建立專案期 CodeCatalyst 間將 GitHub 帳戶 Connect 至空間

建立新 CodeCatalyst 專案時，您可以將 GitHub 帳戶連接到您的 CodeCatalyst 空間。如需詳細資訊，請參閱 建立具有連結 GitHub儲存庫的專案。

中斷 GitHub 帳戶與 CodeCatalyst 空間的連線

執行下列步驟以中斷 GitHub 帳戶與 CodeCatalyst 空間的連線。一旦帳戶斷開連線，帳戶儲存庫中的事件將不會啟動工作流程執行，而且您將無法在開 CodeCatalyst 發環境中使用這些存放庫。

> **Note**
>
> 若要中斷 GitHub 帳戶的連結，您必須先取消所有連結 GitHub 儲存庫與該帳戶的連結。如需詳細資訊，請參閱 取消 GitHub 儲存庫與專案的 CodeCatalyst 連結。

中斷連接 GitHub 帳戶

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您 CodeCatalyst 空間的已安裝GitHub 儲存庫擴充功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

      單

      的目錄圖標。

      中

4. 在GitHub 儲存庫中，選擇設定。
5. 在 [GitHub 帳戶] 索引標籤中，選擇您要中斷連線的 GitHub 帳戶。
6. 選擇中斷連接 GitHub 帳戶。
7. 在 [中斷連線] 對話方塊中，檢閱中斷帳號連線的效果。
8. 在文字輸入欄位中輸入斷線，然後選擇 [中斷連線]。

管理 GitHub 儲存庫 CodeCatalyst

若要在中使用 GitHub 儲存庫 CodeCatalyst，您必須先將儲存庫連結至您的 CodeCatalyst專案。連結 GitHub 存放庫之前，您必須先將存放庫所屬的 GitHub 帳戶與您的 CodeCatalyst 空間連線。如需詳細資訊，請參閱 將 GitHub 帳戶 Connect 到 CodeCatalyst 空間。

如果您不想再在中使用 GitHub 存放庫 CodeCatalyst，可以將其與 CodeCatalyst 專案取消連結。取消鏈接存儲庫時，該存儲庫中的事件將不會啟動工作流程運行，並且您將無法將該存儲庫與開 CodeCatalyst 發環境一起使用。

內容

- 將 GitHub 儲存庫連結至專 CodeCatalyst 案
- 在 CodeCatalyst 專案建立期間將 GitHub 儲存庫連結至專案
- 取消 GitHub 儲存庫與專案的 CodeCatalyst 連結

將 GitHub 儲存庫連結至專 CodeCatalyst 案

您可以在工作流程中使用連結的存放 GitHub 庫，其中連結 GitHub 存放庫中的事件會啟動可能會建置、測試或部署程式碼的工作流程，具體取決於工作流程組態。使用連結存 GitHub 放 GitHub 庫之工

作流程的工作流程組態檔儲存在連結的存放庫中。鏈接的 GitHub 存儲庫也可以與開發環境一起使用，以創建，更新和刪除鏈接 GitHub 存儲庫中的文件。您可以從 GitHub 存放GitHub 庫擴充功能的詳細資訊頁面，或從 CodeCatalyst 專案本身的程式碼中的 [程式碼] 檢視中將儲存庫連結至專案。

> **Note**
>
> 一個 GitHub 儲存庫只能連結至一個空間中的一個 CodeCatalyst 專案。您無法連結與專案中另一個儲存庫具有相同名稱的儲存庫。

若要從儲 GitHub 存庫擴充功能詳細資訊頁面將存放 GitHub 庫連結至 CodeCatalyst 專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝GitHub 儲存庫擴充功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

      單

      的目錄圖標。

      中
4. 在GitHub 儲存庫中，選擇設定。
5. 在「連結的 GitHub 儲存庫」頁籤中，選擇「連結儲 GitHub 存庫
6. 從GitHub 帳戶下拉式清單中，選擇包含您要連結之儲存庫的 GitHub 帳戶。
7. 從GitHub 存放庫下拉式清單中，選擇要連結至 CodeCatalyst 專案的儲存庫。

   > **Tip**
   >
   > 如果儲存庫的名稱顯示為灰色，則無法連結該儲存庫，因為該儲存庫已連結至空間中的另一個專案。

8. (選擇性) 如果您在儲 GitHub 存庫清單中看不到儲存庫，可能尚未針對中的 Amazon CodeCatalyst 應用程式中的儲存庫存取進行設定 GitHub。您可以設定連線帳戶 CodeCatalyst 中可以使用哪些 GitHub 儲存庫。

   a. 導覽至您的GitHub帳戶，選擇 [設定]，然後選擇 [應用程式]。

   b. 在已安裝的 GitHub 應用程式索引標籤中，選擇 Amazon CodeCatalyst 應用程式的設定。

c. 執行下列其中一項動作，以設定您要連結的 GitHub 儲存庫存取權限 CodeCatalyst：

- 若要提供對所有目前和 future 儲存庫的存取權，請選擇所有儲存庫。
- 若要提供特定儲存區域的存取權，請選擇 [僅選取儲存區域]，選擇 [選取儲存區域] 下拉式清單，然後選擇您要允許連結的儲存區域 CodeCatalyst。

> **Note**
>
> - 您無法在 CodeCatalyst 專案中使用空白或封存的 GitHub 儲存庫。
> - 您無法連結與專案中 GitHub 儲存庫名稱相同的儲存 CodeCatalyst 庫。
> - GitHub 儲存庫擴充功能與 GitHub 企業伺服器儲存庫不相容。

9. 從CodeCatalyst 專案下拉式選單中，選擇您要連結 GitHub 儲存庫的 CodeCatalyst 專案。
10. 選擇 Link (連結)。

若要從 CodeCatalyst 專案的來源儲 GitHub 存庫頁面將存放庫連結至專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的 CodeCatalyst 項目。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。
4. 選擇新增儲存庫，然後選擇連結儲存庫。
5. 從存放庫提供者下拉式功能表中，選擇GitHub。
6. 從GitHub 帳戶下拉式選單中，選擇包含您要連結之儲存庫的 GitHub 帳戶。
7. 從GitHub 存儲庫下拉菜單中，選擇要鏈接 CodeCatalyst 項目的 GitHub 存儲庫。

> **Tip**
>
> 如果儲存庫的名稱顯示為灰色，則無法連結該儲存庫，因為該儲存庫已連結至 Amazon CodeCatalyst 中的另一個專案。

8. (選擇性) 如果您在 GitHub 儲存庫下拉式清單中沒有看到GitHub 儲存庫，可能尚未在中的 space 應用程式中將其設定為存放庫存取 GitHub。您可以設定連線帳戶 CodeCatalyst 中可以使用哪些 GitHub 儲存庫。

   a. 導覽至您的GitHub帳戶，選擇 [設定]，然後選擇 [應用程式]。

b. 在已安裝的 GitHub 應用程式索引標籤中，選擇 Amazon CodeCatalyst 應用程式的設定。

c. 執行下列其中一項動作，以設定您要連結的 GitHub 儲存庫存取權限 CodeCatalyst：

- 若要提供對所有目前和 future 儲存庫的存取權，請選擇所有儲存庫。
- 若要提供特定儲存區域的存取權，請選擇 [僅選取儲存區域]，選擇 [選取儲存區域] 下拉式清單，然後選擇您要允許連結的儲存區域 CodeCatalyst。

> **Note**
>
> - 您無法在 CodeCatalyst 專案中使用空白或封存的 GitHub 儲存庫。
> - 您無法連結與專案中 GitHub 儲存庫名稱相同的儲存 CodeCatalyst 庫。
> - GitHub 儲存庫擴充功能與 GitHub 企業伺服器儲存庫不相容。

9. 選擇 Link (連結)。

在 CodeCatalyst 專案建立期間將 GitHub 儲存庫連結至專案

建立新 CodeCatalyst 專案時，您可以將 GitHub 存放庫連結至新 CodeCatalyst 專案。如需詳細資訊，請參閱 建立具有連結 GitHub儲存庫的專案。

取消 GitHub 儲存庫與專案的 CodeCatalyst 連結

取消鏈接存儲庫不會刪除 GitHub 存儲庫或對其進行任何更改。它不會刪除儲存在該連結存放庫中的任何工作流程設定檔。但是，一旦取消鏈接 GitHub 存儲庫，該存儲庫中的事件將不再啟動工作流運行，並且您無法將存儲庫與開發環境一起使用。

> **Important**
>
> 若要取消 GitHub 儲存庫與 CodeCatalyst 專案的連結，您必須是 S pace 管理員或專案管理員。

若要取消儲存庫的 GitHub 連結

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

3. 執行下列其中一個動作，以檢視您空間的已安裝GitHub 儲存庫擴充功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

      單  中

      的目錄圖標。

4. 在GitHub 儲存庫中，選擇設定。
5. 在 [連結的 GitHub 儲存庫] 索引標籤中，選擇您要取消連結的儲 GitHub 存庫。
6. 選擇取消連結 GitHub 儲存庫。
7. 在「取消連結」(Unlink) 對話方塊中，檢閱解除連結儲存庫的效果。
8. 在文字輸入欄位中輸入取消連結，然後選擇 [取消連結]。

檢視連結的 GitHub 儲存庫於 CodeCatalyst

您可以在專案的來源 GitHub 儲存庫清單中，或從儲存庫擴充功能詳細資訊頁面檢視連結的GitHub 儲存庫。從存儲庫列表中選擇它們不會在中打開它們 CodeCatalyst。相反地，它們會在中開啟 GitHub，您可以在其中檢視和處理連結儲存庫中的程式碼。

若要檢視連結的 GitHub 儲存庫 CodeCatalyst

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 導航到您的 CodeCatalyst 項目。
3. 在瀏覽窗格中，選擇 [程式碼]，然後選擇 [原始碼儲存庫]。

若要從擴充功能詳細資料頁面檢視連結的 GitHub 儲存庫

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 瀏覽至您的 CodeCatalyst 空間，然後選擇 [已安裝的擴充功能] 索引標籤。
3. 在GitHub 儲存庫中，選擇設定。
4. 選擇 [連結的 GitHub 儲存庫] 索引標籤，以檢視連接到您 CodeCatalyst 空間中 CodeCatalyst 專案的所有 GitHub 存放庫

連結至您專案的 GitHub 儲存庫會顯示在清單中。選擇要在其中檢視和編輯檔案的 GitHub 儲存庫 GitHub。

> **Note**
>
> 如果工作流程在來源動作中使用 GitHub 存放庫，您在視覺化編輯器或 YAML 編輯器中對工作流程 YAML 所做的變更 CodeCatalyst 將會自動認可並推送至存放庫。 GitHub

在 CodeCatalyst 工作流中使用 GitHub 儲存

您可以使用連結 GitHub 存放庫作為工作流程的來源，在此工作流程中變更連結 GitHub 存放庫中的指定分支會自動啟動工作流程執行。

工作流程是一種自動化程序，描述如何在持續整合和持續交付 (CI/CD) 系統中建置、測試及部署程式碼。工作流程會定義工作流程執行期間要採取的一系列步驟或動作。工作流程也會定義導致工作流程啟動的事件或觸發器。若要設定工作流程，您可以使用 CodeCatalyst 主控台的視覺化或 YAML 編輯器建立工作流程定義檔案。

> **Tip**
>
> 若要快速瞭解如何在專案中使用工作流程，請使用藍圖建立專案。每個藍圖都會部署運作正常的工作流程，您可以檢閱、執行和試驗。

當您將工作流程設定為使用連結存 GitHub 放庫時，工作流程組態檔案會儲存在該儲存 GitHub 庫中。工作流程設定是定義工作流程名稱、觸發程序、資源、成品和動作的 YAML 檔案。如需有關工作流程組態檔案的詳細資訊，請參閱工作流定義參考。

工作流程組態檔案必須位於 GitHub 存放庫的`./codecatalyst/workflows/`目錄中。

您可以使用工作流程編輯器來建立和設定工作流程。如需更多資訊，請參閱開始使用中的工作流程 CodeCatalyst及使用來源。

在 GitHub 存放庫事件後自動啟動工作流程執行

您可以將 CodeCatalyst 工作流程設定為在程式碼推送至 GitHub 存放庫的指定分支時自動啟動執行。若要自動開始工作流程執行，請將觸發器新增至工作流程組態檔案的`Triggers`區段。

範例：簡單的程式碼推送觸發器

下列範例顯示每當程式碼推送至來源儲存庫中的任何分支時，就會啟動工作流程執行的觸發程序。

```
Triggers:
  - Type: PUSH
```

範例：簡單的提取要求觸發程序

下列範例顯示每當針對來源儲存庫中的任何分支建立提取要求時，啟動工作流程執行的觸發程序。

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
```

如需更多詳細資訊，請參閱 使用觸發程序。

在 GitHub 企業雲中使用 IP 位址存取限制

Amazon CodeCatalyst GitHub 儲存庫擴充功能與 GitHub 企業雲端 IP 存取限制相容。將 GitHub Enterprise Cloud 組織設定為限制對特定 IP 位址的存取時，您也可以啟用 GitHub 應用程式設定允許清單，以便自動 CodeCatalyst 註冊其 IP 位址 GitHub。或者，您可以手動新增 CodeCatalyst IP 位址。如需詳細資訊，請參閱 GitHub 儲存庫擴充功能使用的 IP 位址。

如果 CodeCatalyst IP 位址不在 GitHub 組織的允許清單中，Amazon CodeCatalyst GitHub 應用程式將無法存取您的 GitHub 儲存庫。

GitHub 儲存庫擴充功能使用的 IP 位址

GitHub 儲存庫擴充功能會使用下列 IP 位址來存取 GitHub 組織的資源：

```
us-west-2
  52.32.242.246
  54.148.176.49
  35.164.118.94
eu-west-1
  34.241.64.10
  34.246.255.80
  3.248.38.7
```

當工作流程失敗時封鎖提 GitHub 取要求合併

將 GitHub 存放庫連結至後 CodeCatalyst，您可以為提取要求新增 CodeCatalyst 工作流程。一或多個工作流程執行可能會在特定的提交中發生，而中每個工作流程的執行狀態 CodeCatalyst 也會反映為

中提交狀態的一部分 GitHub。當推送新的提交時，新的工作流程運行狀態會反映在 GitHub 該新提交中。如果您再次執行某個提交的工作流程，則新的工作流程執行狀態會覆寫該提交和工作流程的先前狀態。

您可以在中設定分支保護規則， GitHub 以便在最新的提交具有失敗的工作流程執行狀態時封鎖提取要求合併。使用分支保護規則時，最新提交的狀態會影響合併提取請求的能力 GitHub。如需詳細資訊，請參閱 GitHub的文件關於狀態檢查和關於受保護的分支。若要進一步瞭解工作流程，請參閱使用梯段和使用觸發程序。

在中使用吉拉問題 CodeCatalyst

Jira 是一個軟件應用程序，可幫助敏捷開發團隊計劃，分配，跟踪，報告和管理工作。Jira 軟件擴展允許您在 Amazon CodeCatalyst 項目中使用 Jira 項目。

> **Note**
>
> CodeCatalyst 僅與 Jira 軟件雲兼容。

安裝並設定 Amazon CodeCatalyst 專案的 Jira 軟體擴充功能後，您將能夠：

1. CodeCatalyst 通過將它們鏈接到 CodeCatalyst 項目來訪問 Jira 項目
2. 使用 CodeCatalyst 提取要求更新 Jira 問題
3. 查看 Jira 問題中鏈接的 CodeCatalyst 提取請求的狀態和工作流程運行

下列主題說明如何安裝和設定 Jira 軟體延伸功能以在中使用 Jira。 CodeCatalyst

主題

- 快速入門：在中使用 Jira 問題 CodeCatalyst
- 管理吉拉網站 CodeCatalyst
- 管理 Jira 專案 CodeCatalyst
- 將 Jira 問題鏈接到 CodeCatalyst 拉取請求
- 查看吉拉的 CodeCatalyst 事件
- 搜尋吉拉問題 CodeCatalyst

快速入門：在中使用 Jira 問題 CodeCatalyst

執行以下步驟來安裝 Jira 軟體擴充功能、將 Jira 網站連接到您的 CodeCatalyst 空間，並將 Jira 專案連結至您的專案。 CodeCatalyst

內容

步驟 1：從目錄中安裝 Jira 軟件擴展 CodeCatalyst

管理 Jira 的第一步 CodeCatalyst 是從目錄中安裝 Jira 軟體擴充功能。 CodeCatalyst 要安裝擴展程序，請執行以下步驟，選擇 Jira 軟件擴展。

> **Important**
>
> 作為安裝和設定 Jira 軟體延伸功能的一部分，您必須擁有 Atlassian 帳戶和現有的 Jira 網站，但是稍後可以建立 Jira 專案。您必須是 Jira 網站管理員和 CodeCatalyst 空間管理員。

若要從 CodeCatalyst 目錄安裝擴充功能

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。
3. 選擇頂端功能

   表 中的「目錄」圖示，以導覽至目錄。 CodeCatalyst CodeCatalyst 您可以搜索 Jira 軟件或根據類別過濾擴展。
4. (選擇性) 若要查看有關擴充功能的更多詳細資訊，例如擴充功能所擁有的權限，請選擇 Jira Software 擴充功能名稱。
5. 選擇 Install (安裝)。檢閱擴充功能所需的權限，如果您要繼續，請再次選擇 [安裝]。

安裝 Jira 軟體擴充功能後，您將前往 Jira 軟體擴充功能詳細資料頁面，您可以在此檢視和管理連接的 Jira 網站和連結的 Jira 專案。

步驟 2：將您的 Jira 網站 Connect 到您的 CodeCatalyst 空間

安裝 Jira 軟件擴展後，下一步是將您的 Jira 站點連接到您 CodeCatalyst 的空間。

> **Important**
>
> 若要將您的 Jira 網站連線到您的 CodeCatalyst 空間，您必須同時是 Jira 網站管理員和 S CodeCatalyst pace 管理員。

將您的 Jira 網站連接到 CodeCatalyst

1. 在「已連線的 Jira 網站」索引標籤中，選擇「Connect Jira 網站」以前往 Atlassian Marketplace 的外部網站。
2. 選擇「立即取得」以開始在您的 Jira 網站 CodeCatalyst 上進行安裝。
3. 根據您的角色，執行下列任一項作業：

    1. 如果您是 Jira 網站管理員，請從網站下拉式功能表中選擇 Jira 網站以安裝應用程式，然後選擇「安裝 CodeCatalyst 應用程式」。

        > **Note**
        >
        > 如果您有一個 Jira 網站，則不會顯示此步驟，並且會自動將您導向到下一個步驟。

    2. a. 如果您不是 Jira 管理員，請從站點下拉菜單中選擇 Jira 站點以安裝 CodeCatalyst 應用程序，然後選擇請求應用程序。有關安裝 Jira 應用程序的更多信息，請參閱誰可以安裝應用程序？ 。

        b. 在輸入文字欄位 CodeCatalyst 中輸入您需要安裝的原因，或保留預設文字，然後選擇 [提交請求]。
4. 檢閱安裝應用程式 CodeCatalyst 時所執行的動作，然後選擇 [立即取得]。
5. 安裝應用程式之後，請選擇 [返回 CodeCatalyst至] 以返回 CodeCatalyst。

將您的 Jira 站點連接到後 CodeCatalyst，您可以在 Jira 軟件擴展詳細信息頁面的「已連接的 Jira 站點」選項卡中查看連接的站點。

第 3 步：將您的 Jira 項目鏈接到您的 CodeCatalyst 項目

在中管理 Jira 項目的第三步 CodeCatalyst 是將 Jira 項目鏈接到要在其中使用它的 CodeCatalyst 項目。

> **Note**
>
> 一個 CodeCatalyst 專案只能連結至一個 Jira 專案。一個 Jira 項目可以鏈接到多個 CodeCatalyst 項目。

> **Important**
>
> 要將您的 Jira 項目鏈接到 CodeCatalyst 項目，您必須是 S CodeCatalyst pace 管理員或 CodeCatalyst 項目管理員。

若要從 Jira 軟體延伸功能詳細資訊頁面將 Jira CodeCatalyst 專案連結至專案

1. 在「連結的 Jira 專案」標籤中，選擇「連結 Jira 專案」。
2. 從「Jira 網站」下拉式功能表中，選擇包含您要連結之專案的 Jira 網站。
3. 從 Jira 項目下拉菜單中，選擇要鏈接到項 CodeCatalyst 目的項目。
4. 從CodeCatalyst 項目下拉菜單中，選 CodeCatalyst 擇要鏈接到 Jira 項目的項目。
5. 選擇 Link (連結)。

一旦 Jira 專案連結到專案，就會完全停用對 CodeCatalyst 問題的存取權， CodeCatalyst 導覽窗格中的「問題」將會取代為連結至 Jira CodeCatalyst 專案的 Jira 問題項目。

步驟 4：將您的 Jira 問題鏈接到 CodeCatalyst 拉取請求

將 Jira 項目鏈接到 CodeCatalyst 項目後，您可以通過創建提取請求來鏈接 CodeCatalyst 問題，並將其顯示為提取請求的屬性。

> **Note**
>
> 如果沒有在 CodeCatalyst 項目中具有兩個分支的源代碼存儲庫，則無法創建提取請求。如需提取要求的詳細資訊，請參閱使用中的提取要求 CodeCatalyst。

將 Jira 問題連結至 CodeCatalyst 提取要求

1. 在功能窗格中，選擇 [程式碼]，然後選擇 [提取要求]。
2. 選擇「建立提取請求」，以輸入提取請求明細。
3. 從「來源存放庫」下拉式功能表中，選擇您要連結提取要求的來源存放庫。
4. 從「來源」分支下拉式功能表中，選擇包含您要檢閱之變更的分支。
5. 從「目標」分支下拉式功能表中，選擇您要合併已檢閱變更的分支。
6. 在提取請求標題文本輸入字段中，輸入提取請求的標題。
7. 選擇「Jira 問題」的「鏈接問題」-可選字段，選擇下拉菜單，然后從鏈接的 Jira 項目中搜索要添加的 Jira 問題。
8. 選取您要新增至提取要求的 Jira 問題。
9. 選擇「建立」以建立提取請求。

提取請求的摘要狀態和相關 CodeCatalyst 工作流程事件的狀態都會反映在您的 Jira 問題中。

後續步驟

安裝 Jira 軟件擴展後，連接您的 Jira 站點，將您的 Jira 項目鏈接到項目並鏈接拉請求，從更新反映在您的 Jira 項 CodeCatalyst 目中。 CodeCatalyst 如需在 Jira 中檢視 CodeCatalyst 事件的詳細資訊，請參閱查看吉拉的 CodeCatalyst 事件。

管理吉拉網站 CodeCatalyst

若要管理中的 Jira 專案 CodeCatalyst，您必須將 Jira 網站連接到您的 CodeCatalyst 空間。如果您不想再在中使用 Jira 站點的項目 CodeCatalyst，則可以斷開該 Jira 站點的連接。當 Jira 站點斷開連接時，該站點項目中的 Jira 問題將無法在項目中使用，並且 CodeCatalyst 問題將再次成為問題提供者。CodeCatalyst

> **Important**
>
> 若要連線或中斷 Jira 網站和 CodeCatalyst 空間的連線，您必須同時是 Jira 網站管理員和 S CodeCatalyst pace 管理員。

內容

- 中斷 Jira 站台與空間的連接 CodeCatalyst

將 Jira 站台連接到空間 CodeCatalyst

執行以下步驟將 Jira 站台連接到您的 CodeCatalyst 空間。

將您的 Jira 網站連接到 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝 Jira 軟體延伸功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

   單 

   的目錄圖標。

   中

4. 在 Jira 軟體中，選擇設定。
5. 在「已連線的 Jira 網站」索引標籤中，選擇「Connect Jira 網站」以前往 Atlassian Marketplace 的外部網站。
6. 選擇「立即取得」以開始在您的 Jira 網站 CodeCatalyst 上進行安裝。

   > **Note**
   >
   > 如果您之前已安裝 CodeCatalyst 到 Jira 網站，您將收到通知。選擇 [開始] 以進行最後一步。

7. 根據您的角色，執行下列任一項作業：

   1. 如果您是 Jira 網站管理員，請從網站下拉式功能表中選擇 Jira 網站以安裝應用程式，然後選擇「安裝 CodeCatalyst 應用程式」。

      > **Note**
      >
      > 如果您有一個 Jira 網站，則不會顯示此步驟，並且會自動將您導向到下一步。

2. a. 如果您不是 Jira 管理員，請從站點下拉菜單中選擇 Jira 站點以安裝 CodeCatalyst 應用程序，然後選擇請求應用程序。有關安裝 Jira 應用程序的更多信息，請參閱誰可以安裝應用程序？ 。

   b. 在輸入文字欄位 CodeCatalyst 中輸入您需要安裝的原因，或保留預設文字，然後選擇 [提交請求]。

8. 檢閱安裝應用程式 CodeCatalyst 時所執行的動作，然後選擇 [立即取得]。
9. 安裝應用程式之後，請選擇 [返回 CodeCatalyst至] 以返回 CodeCatalyst。

將您的 Jira 站點連接到後 CodeCatalyst，您可以在 Jira 軟件擴展詳細信息頁面的「已連接的 Jira 站點」選項卡中查看連接的站點。

中斷 Jira 站台與空間的連接 CodeCatalyst

執行以下步驟以中斷 Jira 站台與 CodeCatalyst 空間的連線。

> **Note**
>
> 若要中斷 Jira 網站的連結，您必須先取消所有連結的 Jira 專案與該帳戶的連結。如需詳細資訊，請參閱 取消 Jira 專案與專案的連結 CodeCatalyst。

若要中斷 Jira 網站的連線

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝 Jira 軟體延伸功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜

      單

      的目錄圖標。

      中

4. 在 Jira 軟體中，選擇設定。
5. 在「已連接的 Jira 地點」標簽中，選擇要中斷連接的 Jira 站點。
6. 選擇斷開 Jira 網站的連接。
7. 在「中斷連線」對話方塊中，檢閱中斷站台連線的效果。

8. 在文字輸入欄位中輸入斷線，然後選擇 [中斷連線]。

管理 Jira 專案 CodeCatalyst

若要管理中的 Jira 專案 CodeCatalyst，您必須先將 Jira 專案連結至您的 CodeCatalyst專案。然後，您可以將 Jira 問題添加並使用到 CodeCatalyst 項目中的提 CodeCatalyst 取請求。連結 Jira 專案之前，您必須先將專案所屬的 Jira 網站與您 CodeCatalyst 的空間連接起來。如需詳細資訊，請參閱 將 Jira 站台連接到空間 CodeCatalyst。

如果您不想再在中使用 Jira 專案 CodeCatalyst，您可以將其與專案取消連結。 CodeCatalyst取消連結 Jira 專案時， CodeCatalyst專案中將無法使用 Jira 問題，而「問題」將再次成為 CodeCatalyst 問題提供者。

> **Important**
>
> 要將您的 Jira 項目鏈接到 CodeCatalyst 項目，您必須是 S CodeCatalyst pace 管理員或 CodeCatalyst 項目管理員。

內容

將 Jira 項目鏈接到項目 CodeCatalyst

您可以使用連結的 Jira 專案來管理問題，並將 CodeCatalyst 提取要求連結至 Jira 問題。拉取請求的摘要狀態和相關 CodeCatalyst 工作流程事件的狀態都會反映在您的 Jira 問題中。

> **Note**
>
> 一個 CodeCatalyst 專案只能連結至一個 Jira 專案。一個 Jira 項目可以鏈接到多個 CodeCatalyst 項目。

若要從 Jira 軟體延伸功能詳細資訊頁面將 Jira CodeCatalyst 專案連結至專案

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/

2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝 Jira 軟體延伸功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜單  中的目錄圖標。

4. 在 Jira 軟體中，選擇設定。
5. 在「連結的 Jira 專案」標籤中，選擇「連結 Jira 專案」。
6. 從「Jira 網站」下拉式功能表中，選擇包含您要連結之專案的 Jira 網站。
7. 從 Jira 項目下拉菜單中，選擇要鏈接到項 CodeCatalyst 目的項目。
8. 從CodeCatalyst 項目下拉菜單中，選 CodeCatalyst 擇要鏈接到 Jira 項目的項目。
9. 選擇 Link (連結)。

一旦 Jira 專案連結到專案，就會完全停用對 CodeCatalyst 問題的存取權， CodeCatalyst 導覽窗格中的「問題」將會取代為連結至 Jira CodeCatalyst 專案的 Jira 問題項目。

取消 Jira 專案與專案的連結 CodeCatalyst

取消連結專案並不會刪除 Jira 專案，包括計劃項目或開發資訊，也不會對其進行任何變更。但是，一旦您取消連結 Jira 專案，該專案的 Jira 問題將無法再連結至專案。 CodeCatalyst

若要取消 Jira 專案的連結

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。
3. 執行下列其中一個動作，以檢視您空間的已安裝 Jira 軟體延伸功能：

   a. 選擇 [設定]，然後選擇 [安裝的擴充功能

   b. 選擇頂部菜單  中的目錄圖標。

4. 在連結的 Jira 專案索引標籤中，選擇您要取消連結的 Jira 專案。
5. 選擇「取消連結 Jira 專案」。

6. 在「取消連結」(Unlink) 對話方塊中，檢閱解除連結儲存庫的效果。
7. 在文字輸入欄位中輸入取消連結，然後選擇 [取消連結]。

將 Jira 問題鏈接到 CodeCatalyst 拉取請求

您可以將在 CodeCatalyst 來源儲存庫中建立的提取要求連結至 Jira 問題。連結 Jira 問題後，問題會顯示為提取要求的屬性。因此，提取要求事件、工作流程事件和部署事件會傳送至 Jira 並新增至 Jira 問題。提取請求可以鏈接到一個或多個 Jira 問題。您只能鏈接 CodeCatalyst 源存儲庫中的提取請求，而不能鏈接第三方存儲庫中的請求 GitHub。您的 Jira 專案必須先連結至專案，才能將 Jira 問題連結至提取要求。 CodeCatalyst 若要取得有關將 Jira 專案連結至專案的更多資訊，請參閱〈〉管理吉拉網站 CodeCatalyst。 CodeCatalyst

> **Note**
>
> 如果沒有在 CodeCatalyst 項目中具有兩個分支的源代碼存儲庫，則無法創建提取請求。如需提取要求的詳細資訊，請參閱使用中的提取要求 CodeCatalyst。

將 Jira 問題連結至 CodeCatalyst 提取要求

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的 CodeCatalyst 項目。
3. 在功能窗格中，選擇 [程式碼]，然後選擇 [提取要求]。
4. 選擇「建立提取請求」，以輸入提取請求明細。
5. 從「來源存放庫」下拉式功能表中，選擇您要連結提取要求的來源存放庫。
6. 從 [來源] 分支下拉式功能表中，選擇包含您要檢閱之變更的分支。
7. 從「目標」分支下拉式功能表中，選擇您要合併已檢閱變更的分支。
8. 在提取請求標題文本輸入字段中，輸入提取請求的標題。
9. 選擇「Jira 問題」的「鏈接問題」-可選字段，選擇下拉菜單，然后從鏈接的 Jira 項目中搜索要添加的 Jira 問題。
10. 選取您要新增至提取要求的 Jira 問題。
11. 選擇「建立」以建立提取請求。

將 Jira 問題連結至提 CodeCatalyst 取要求後，即可取得提取要求的摘要。摘要包括工作流程執行、連結的問題、必要的審核者、選擇性審核者和作者。

> **Note**
>
> 中不提供與 Jira 問題相關聯的「工作負責人」與「建立者」資訊。 CodeCatalyst

鏈接拉請求後，同步的 CodeCatalyst 項目和 Jira 項目允許從 CodeCatalyst 更新反映在您的 Jira 項目中。在 Jira 中查看鏈接的提取請求的狀態以及與提取請求相關的任何工作流程事件都會顯示在 Jira 問題中。如需在 Jira 中檢視 CodeCatalyst 事件的詳細資訊，請參閱查看吉拉的 CodeCatalyst 事件。

查看吉拉的 CodeCatalyst 事件

如果您的 CodeCatalyst 項目和 Jira 項目已鏈接，那么您的 Jira 問題中會反映提取請求的摘要狀態和相關 CodeCatalyst 工作流程事件的狀態。例如，如果您關閉或合併中的提取請求 CodeCatalyst，則狀態更新會反映在 Jira 問題中。 CodeCatalyst 與提 CodeCatalyst 取要求相關的工作流程 CI/CD 事件會同步處理，因此也會將成功的工作流程執行傳送至 Jira 問題。

若要檢視 Jira 問題中的 CodeCatalyst 事件

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的 CodeCatalyst 項目。
3. 在 CodeCatalyst 瀏覽窗格中，選擇 [程式碼]，選擇 [提取要求]，然後選擇您要在 Jira 專案中檢視之 Jira 問題的提取要求。
4. 在「其他資訊」窗格中，選擇您要在 Jira 專案中檢視的 Jira 問題。
5. 從 Jira 專案的 [詳細資料] 窗格中，選擇 [開發] 列出的提取要求，以查看提取要求的詳細資料。
6. (選擇性) 若要查看最新的組建，請選擇 [組建] 索引標籤。
7. (選擇性) 若要查看開發狀態，請選擇部署索引標籤。

搜尋吉拉問題 CodeCatalyst

連結 Jira 專案後，您可以使用 CodeCatalyst 全域搜尋列來搜尋連結的 Jira 專案中的問題。您也可以在中搜尋 Jira 問題， CodeCatalyst 同時從提取要求連結到問題。如需將 Jira 問題連結至提 CodeCatalyst 取要求的詳細資訊，請參閱將 Jira 問題鏈接到 CodeCatalyst 拉取請求。

在連結的 Jira 專案中搜尋 Jira 問題

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導航到您的 CodeCatalyst 項目。

3. 在全域搜尋列中，搜尋連結的 Jira 專案，找出您想要連結至提取要求的問題或 Jira 問題。

在中搜尋 CodeCatalyst

使用中的搜尋列或專用的搜尋結果視窗 CodeCatalyst 來搜尋程式碼、問題、專案和使用者 CodeCatalyst。

您可以在搜尋列中輸入名稱、描述和狀態等查詢，來尋找空間和專案中的資源。您也可以使用搜尋查詢語言來縮小搜尋查詢。

主題

若要搜尋

1. 在頂端導覽列的搜尋列中，輸入搜尋查詢。
2. (選擇性) 使用的搜尋查詢語言 CodeCatalyst來縮小搜尋查詢。如需詳細資訊，請參閱精簡您的搜尋查詢。
3. 執行下列任意一項：
   - 若要搜尋目前所在專案內的資源，請選擇 [此專案]。
   - 若要搜尋您目前所在空間中的所有專案中的資源，請選擇此空間。
4. 執行下列任一項作業，在專用的搜尋結果視窗中檢視搜尋結果：
   - 在快速搜索結果窗口的底部，選擇「查看所有結果（在計划名稱中）」|「空格名稱」以查看所有搜索結果。
   - 按 Enter 鍵可檢視所有搜尋結果。

> **Tip**
>
> 在提取要求註解或說明中，或在問題註解或說明中，使用 @ 符號後面加上顯示名稱或使用者名稱，提及其他專案使用者。您也可以使用 @符號後面接問題或代碼檔案的名稱來連結到問題或程式碼檔等資源。

精簡您的搜尋查詢

如果您在搜尋後找不到您要尋找的內容，您可以使用專門的查詢語言 CodeCatalyst來縮小搜尋範圍。個別欄位沒有字元限制，但整體查詢的限制為 1,024 個字元。

主題

按類型精煉

若要將搜尋範圍縮小為特定類型的資訊，請包含*type:result-type*在搜尋中，其中*###*#為codeissue、project、或user。

範例：

- type:code AND java— 在包含「java」的程式碼相關欄位中顯示程式碼結果。

  如需詳細資訊，請參閱程式碼欄位。
- type:issue AND Bug— 在包含「Bug」的問題相關欄位中顯示問題結果。

  如需詳細資訊，請參閱問題欄位。
- type:user AND MaryMajor— 在包含「MaryMajor」的使用者相關欄位中顯示使用者結果。

  如需詳細資訊，請參閱使用者欄位。
- type:project AND Datafeeder— 顯示包含「資料傳送器」的專案結果。

  如需詳細資訊，請參閱專案欄位。

按字段精煉

若要將搜尋範圍縮小為特定欄位，請*field-name:query*在搜尋中加入*###*#為titleusernameprojectdescription、、、等等，*##*是您要搜尋的文字。如需欄位清單，請參閱可搜尋欄位參考。您可以使用括號搜尋多個查詢。

範例：

- `title:bug`— 顯示標題包含「bug」的結果。
- `username:John`— 顯示使用者名稱包含「John」的結果。
- `project:DataFeeder`— 在專案「DataFeeder」中顯示結果。查詢不區分大小寫。
- `description:overview`— 顯示描述包含「概述」的結果。

使用布林運算子精簡

若要指定搜尋片語的限制，您可以使用布林運算子AND、OR、和NOT。如果您列出多個片語，默認情況OR下將它們 CodeCatalyst連接起來。您可以使用括號將搜尋詞組成群組。

- `exception AND type:code`— 僅顯示「異常」的代碼結果。
- `path:README.md AND repo:ServerlessAPI`— 顯示具有「Readme.md」路徑的結果，其中存儲庫被命名為「無服務器 API」。
- `buildspec.yml AND (repo:ServerlessAPI OR ServerlessWebApp)`-顯示「構建規格 .yml」的結果，其中存儲庫是「無服務器 API」或「」。ServerlessWebApp
- `path:java NOT (path:py OR path:ts)`— 顯示路徑包含「java」但不包含「py」或「ts」的結果。

按項目精煉

若要將搜尋範圍縮小為特定專案，請包含*`project:name AND query`*在您的搜尋中，其中 *`name`* 是您要搜尋的專案，而*`##`*是您要搜尋的內容。

- `project:name AND query`— 顯示路徑包含查詢和專案名稱的結果。

使用搜尋時的考量

延遲內容更新 — 內容更新 (例如名稱變更或重新指派問題) 可能需要數分鐘的時間才會反映在搜尋結果中。大型更新 (例如程式碼庫移轉) 可能需要較長的時間才會顯示在搜尋結果中。

逸出特殊字元 — 下列特殊字元需要在搜尋查詢中特別考量：`+ - & & || ! ( ) { } [ ] ^ " ~ * ? : \`特殊字元不會影響查詢，您必須將其移除或逸出。若要逸出字元，請在字元前面加上反斜線 (\)。例如，搜尋查詢 [功能] 應為 [特徵] 或 [功能\]。

縮小搜尋範圍 — 搜尋不區分大小寫。以全部小寫字母搜尋可防止您的查詢在大小寫變更時分割文字。例如，若要查詢MyService和僅查詢MyService，請考慮查詢myservice以避免僅包含my或的結果service。

搜索默認情況下將單詞和單詞的部分與或明智的結合聯接起來。例如，new function可以傳回包含new和function且只有new或的結果function。為了避免後者，請將多個單詞與AND. 例如，您可以搜尋new AND function。

預設分支 — 搜尋只會傳回來源儲存庫預設分支上最新提交的程式碼結果。要查找其他分支或提交上的代碼，請考慮在本地克隆存儲庫，在開發環境中打開分支，或在 CodeCatalyst UI 中查看分支和詳細信息。更改默認分支會導致更新通過搜索發現的文件。如需詳細資訊，請參閱檢視和變更儲存庫的預設分支。

可搜尋欄位參考

CodeCatalyst 當您輸入搜尋查詢時，會搜尋下列欄位。別名是您可以用來參考進階查詢語言中欄位的另一個名稱。

程式碼欄位

欄位	別名	說明
分行名稱	分支	代碼文件所在的分支名稱。
code	N/A	有關代碼內容的信息，以代碼片段的形式表示與搜索匹配的源代碼的部分。
同意	N/A	返回代碼最後更新的提交的提交 ID。可能是也可能不是在中指定的分支名稱提示的提交 ID。branchName
提交留言	N/A	上次更新程式碼的提交訊息。可能是也可能不是在中指定的分支名稱提示的提交消息。branchName 如果沒有提供提交訊息，這個值將是一個空字串。

欄位	別名	說明
filePath	路徑	此代碼文件的文件路徑。
lastUpdatedBy	N/A	CodeCatalyst 上次更新代碼 FILE 的用戶。如果使用者名稱不可用，這個值將會是 Git 設定中設定的使用者的電子郵件地址。
lastUpdatedBy識別碼	N/A	上次更新代碼的系統生成的用戶的唯一 ID。如果使用者 ID 無法使用，則此值可能是使用者的電子郵件地址。
lastUpdatedTime	N/A	上次使用包含代碼的提交更新搜尋資料的時間 (以協調世界時間 (UTC) 時間戳記表示)。
projectId	N/A	系統產生的專案唯一 ID。
projectName	專案名稱，專案	顯示項目的名稱，該項目包含已提交代碼 FILE 的源存儲庫。
儲存庫	RepoID	系統產生的來源儲存庫的唯一 ID。
儲存庫名稱	儲存庫, 回購	已提交程式碼欄位的來源儲存庫的顯示名稱。

問題欄位

欄位	別名	說明
受讓人	指定	指派給問題之使用者的系統產生的唯一 ID。

欄位	別名	說明
受讓人	受讓人	指定給問題的使用者的使用者名稱。
創建	N/A	顯示建立問題的使用者名稱。
createdById	N/A	系統產生問題之使用者的唯一 ID。
創建時間	N/A	建立問題的時間 (以國際標準時間 (UTC) 時間戳記表示)。
description	N/A	問題的描述。
是封存	archived	Boolean 值，指出是否要在封存狀態下建立問題。
被封鎖	blocked	Boolean 值，指出問題是否標記為已封鎖。
標籤	標籤	系統針對問題產生的標籤唯一 ID。
lastUpdatedBy	N/A	顯示上次更新問題的使用者名稱。
lastUpdatedBy識別碼	N/A	系統產生的上次更新問題之使用者的唯一 ID。
lastUpdatedTime	N/A	上次更新問題的時間 (以國際標準時間 (UTC) 時間戳記表示)。
priority	N/A	問題的優先順序 (如果已指派)。
projectId	N/A	系統產生的專案唯一 ID。
projectName	專案名稱，專案	可以發現此問題的項目。

欄位	別名	說明
短時間	N/A	縮短、自動遞增的問題識別碼。
status	N/A	問題的狀態，指出問題是否在待處理或列中。
雕像	N/A	狀態的系統識別碼。
title	N/A	問題的標題。

專案欄位

欄位	別名	說明
description	N/A	專案的描述。
lastUpdatedTime	N/A	上次更新專案中繼資料的時間 (以協調世界時間 (UTC) 時間戳記表示)。
projectName	project	空間中專案的名稱。
專案路徑	N/A	專案的 URL 可傳遞名稱，在專案建立期間解除。用於需要專案名稱的 URL。

使用者欄位

欄位	別名	說明
displayName	N/A	中用於使用者的名稱 CodeCatalyst。顯示名稱不是唯一的。
email	N/A	使用者的電子郵件地址。

欄位	別名	說明
lastUpdatedTime	N/A	上次更新使用者中繼資料的時間 (以協調世界時間 (UTC) 時間戳記表示)。
使用者名稱	username	使用者在註冊時選擇的使用者名稱 CodeCatalyst。與顯示名稱不同，使用者名稱無法變更。

Amazon 故障 CodeCatalyst

下列資訊可協助您疑難排解中的常見問題 CodeCatalyst。您也可以使用 Amazon 運作 CodeCatalyst 狀況報告來判斷是否存在可能影響您的體驗的服務問題。

主題

解決一般存取問題

我忘記了密碼

問題：我忘記了用於AWS生成器 ID 和 Amazon 的密碼 CodeCatalyst。

可能的修復：解決此問題的最簡單方法是重置密碼。

1. 打開 Amazon CodeCatalyst 並輸入您的電子郵件地址。然後選擇 Continue (繼續)。
2. 選擇忘記密碼？
3. 我們將向您發送一封電子郵件，其中包含更改密碼的鏈接。如果您在收件匣中沒有看到該電子郵件，請檢查垃圾郵件資料夾。

Amazon 的部分或全部 CodeCatalyst 不可用

問題：我導航到或跟隨到 CodeCatalyst 控制台的鏈接，但我看到一個錯誤。

可能的修正：造成此問題的最常見原因是您追蹤了專案的連結，或是您尚未受邀前往的空間，或是服務存在一般可用性問題。檢查 Health 報告，瞭解服務是否有任何已知問題。如果沒有，請聯絡邀請您加入專案或空間的人員，並要求另一個邀請。如果您尚未受邀加入任何專案或空間，您可以註冊並建立自己的空間和專案。

我無法在中創建項目 CodeCatalyst

問題：我想要建立專案，但 [建立專案] 按鈕顯示為無法使用，或是收到錯誤訊息。

可能的修正：造成此問題的最常見原因是您使用不具有 S pace 管理員角色的 AWS Builder ID 登入主控台。您必須具有此角色才能在空間中建立專案。

如果您具有此角色，且按鈕未顯示為可用，則服務可能存在暫時性問題。重新整理瀏覽器，然後再試一次。

解決支援問題

當我訪問 AWS Support Amazon 時出現錯誤 CodeCatalyst

問題：當我選擇 Amazon CodeCatalyst 選項時，我收到以下錯誤消息：AWS Support

```
Unable to assume role

To access support cases, you must add the role
AWSRoleForCodeCatalystSupport to the AWS ## that is the billing account for
the space.
```

可能的修正：將必要角色新增至空間的帳單帳戶。AWS 帳戶指定為空間帳單帳戶的帳戶會使用AWSRoleForCodeCatalystSupport角色和AmazonCodeCatalystSupportAccess受管理的政策。如需詳細資訊，請參閱為您的帳戶和空間建立AWSRoleForCodeCatalystSupport角色。

> **Note**
>
> AWSBuilder ID 只能取得對其進行驗證的別名的支援，而且只能根據中的權限取得資源的支援 CodeCatalyst。空間中的所有使用者均可使用帳戶和帳單支援。不過，建置人員只能取得其在中擁有權限的資源和資訊的支援 CodeCatalyst。

我無法為我的空間建立技術支援案例

問題：我無法為我的空間建立技術支援案例。

修正：必須將商業 Support 援或企業支援方案新增至太空計費帳戶，才能讓空間中的使用者建立技術支援案例。請您的空間管理員將AWS Support方案新增至您的空間帳單帳戶，或造訪 https://repost.aws/ 以詢問AWS社群。

我的支援案例帳戶不再連結至我的空間 CodeCatalyst

問題：我的支援案例帳戶不再連線到中的空間 CodeCatalyst。

修正：如果具有 S pace 管理員角色的使用者切換了太空計費帳戶，這將會中斷AWS Support方案和所有相關案例與空間的連線。Amazon 中AWS Support將不再顯示與舊太空帳單帳戶相關聯的AWS Support案例 CodeCatalyst。該帳單帳戶的 root 使用者可以從中檢視和解決舊案例，也可以為AWS Support其他使用者設定 IAM 許可，以便其他使用者檢視和解決舊案例。AWS Management Console您將無法透過舊的 Space 帳單帳戶繼續 CodeCatalyst 取得技術支援AWS Management Console，但是在您的AWS Support方案取消之前，您可以獲得其他服務的技術支援。

如需詳細資訊，請參閱AWS Support使用指南中的更新、解決和重新開啟案例。

我無法為 Amazon 打開另一個AWS 服務支持案例 AWS Support CodeCatalyst

問題：我無法AWS 服務AWS Support為 CodeCatalyst.

可能的修正：您只能從中開啟 CodeCatalyst 支援案AWS Support例 CodeCatalyst。如果您需要從 CodeCatalyst 其他服務AWS、Amazon 或其他第三方服務部署的服務或資源的支援，則需要透過AWS Management Console或第三方服務支援管道建立案例。如需詳細資訊，請參閱AWS Support使用指南中的建立支援案例和案例管理。

Amazon 的部分或全部 CodeCatalyst 不可用

問題：我導航到或跟隨到 CodeCatalyst 控制台的鏈接，但我看到一個錯誤。

可能的修正：造成此問題的最常見原因是您追蹤了專案的連結，或是您尚未受邀前往的空間，或是服務存在一般可用性問題。檢查 Health 報告，瞭解服務是否有任何已知問題。如果沒有，請聯絡邀請您加入專案或空間的人員，並要求另一個邀請。如果您尚未受邀加入任何專案或空間，您可以註冊並建立自己的空間和專案。

我無法在中創建項目 CodeCatalyst

問題：我想要建立專案，但 [建立專案] 按鈕顯示為無法使用，或是收到錯誤訊息。

可能的修正：造成此問題的最常見原因是您使用不具有 S pace 管理員角色的 AWS Builder ID 登入主控台。您必須具有此角色才能在空間中建立專案。

如果您具有此角色，且按鈕未顯示為可用，則服務可能存在暫時性問題。重新整理瀏覽器，然後再試一次。

我想在以下位置提交意見反應 CodeCatalyst

問題：我在中發現了一個錯誤 CodeCatalyst ，我想提交反饋。

可能的修正：您可以直接在中提交意見反應 CodeCatalyst。

1. 開啟主 CodeCatalyst 控台，網址為 https://codecatalyst.aws/。
2. 在導覽窗格中，選擇 [提供意見反應]。
3. 從下拉式功能表中選擇意見反應類型，然後輸入您的意見反應。

疑難排解來源儲存庫問題

下列資訊可協助您疑難排解中來源儲存庫的常見問題 CodeCatalyst。

主題

- 我已達到空間的最大儲存空間，並看到警告或錯誤
- 我在嘗試複製或推送至 Amazon CodeCatalyst 來源儲存庫時收到錯誤訊息
- 我在嘗試提交或推送至 Amazon CodeCatalyst 來源儲存庫時收到錯誤訊息
- 我需要我的項目的源代碼庫
- 我的源代碼庫是全新的，但包含一個提交
- 我想要一個不同的分支作為我的默認分支
- 我收到有關提取請求中活動的電子郵件
- 我忘記了我的個人訪問令牌（PAT）
- 拉取請求不會顯示我期望的更改
- 提取請求顯示「不可合併」的狀態

我已達到空間的最大儲存空間，並看到警告或錯誤

問題：我想將代碼提交到中的一個或多個源存儲庫 CodeCatalyst，但我看到一個錯誤。在控制台中，我在源存儲庫頁面上看到一條消息，表明我已達到空間的存儲限制。

可能的修正：視您在專案或空間中的角色而定，您可以減少一或多個來源儲存庫的大小、刪除未使用的來源儲存庫，或將帳單層變更為具有更多儲存空間的帳單層。

- 若要減少專案中來源儲存庫的大小，您可以刪除未使用的分支。如需詳細資訊，請參閱 刪除分支（控制台） 及 貢獻者角色。
- 若要減少空間的整體儲存空間，您可以刪除未使用的來源儲存庫。如需詳細資訊，請參閱 刪除來源儲存庫 及 專案管理員角色。
- 若要增加空間的可用儲存空間，您可以將帳單層變更為具有更多儲存空間的計費層級。如需詳細資訊，請參閱 Amazon CodeCatalyst 管理員指南中的變更 CodeCatalyst 帳單層。

我在嘗試複製或推送至 Amazon CodeCatalyst 來源儲存庫時收到錯誤訊息

問題：當我嘗試將源存儲庫克隆到本地計算機或集成開發環境（IDE）時，我收到權限錯誤。

可能的修正：您的 AWS Builder ID 可能沒有個人存取權杖 (PAT)，您可能尚未使用 PAT 設定認證管理系統，或者您的 PAT 可能已過期。請嘗試下列一或多個解決方案：

- 建立個人存取權杖 (PAT)。如需詳細資訊，請參閱 在 Amazon 中管理個人訪問令牌 CodeCatalyst。
- 請確定您已接受邀請加入包含來源儲存庫的專案，且您仍然是該專案的成員。如果您不是該專案的作用中成員，則無法複製來源儲存庫。登入主控台，並嘗試瀏覽至您嘗試複製來源儲存庫的空間和專案。如果您在空間的專案清單中看不到該專案，表示您不是該專案的成員，或者您尚未接受該專案的邀請。如需詳細資訊，請參閱 接受邀請並建立您的AWS產生器 ID。
- 確保您的克隆命令格式正確，並包含您的 AWS 生成器 ID。例如：

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

- 使用 AWS CLI 來確定您有與您的 AWS 產生器 ID 相關聯的 PAT，而且它尚未過期。如果您沒有或 PAT 已過期，請建立一個。如需詳細資訊，請參閱 在 Amazon 中管理個人訪問令牌 CodeCatalyst。
- 嘗試創建一個開發環境以使用源代碼存儲庫中的代碼，而不是將其克隆到本地存儲庫或 IDE。如需詳細資訊，請參閱 建立開發環境。

我在嘗試提交或推送至 Amazon CodeCatalyst 來源儲存庫時收到錯誤訊息

問題：當我嘗試推送到源存儲庫時，收到權限錯誤。

可能的修正：您在專案中可能沒有可讓您提交和推送程式碼變更至專案的角色。在您嘗試將變更推送至來源儲存庫的專案中檢視您的角色。如需詳細資訊，請參閱 檢視專案中的成員 及 在 Amazon 中使用角色 CodeCatalyst。

如果您有一個允許提交和推送更改的角色，則您嘗試提交更改的分支可能會為其配置分支規則，以防止您將代碼更改推送到該分支。嘗試創建一個分支並將代碼推送到該分支。如需詳細資訊，請參閱 使用分支規則管理分支允許的操作。

我需要我的項目的源代碼庫

問題：我的項目要么沒有源存儲庫，要么我的項目需要另一個源存儲庫。

可能的修復：某些項目創建沒有任何資源。如果您是專案的成員，您可以在中建立該專案的來源儲存庫 CodeCatalyst。如果具有 S pace 管理員角色的人安裝GitHub 存放庫並將其連接到 GitHub 帳戶，則您可以連結到可用的 GitHub 存放庫以將其新增至您的專案 (如果您具有專案管理員角色)。如需詳細資訊，請參閱建立來源儲存庫和連結來源儲存庫。

我的源代碼庫是全新的，但包含一個提交

問題：我剛剛創建了一個源代碼存儲庫。它應該是空的，但它有一個提交，一個分支和一個`README.md`文件。

可能的修正：這是預期的行為。中的所有來源存放庫都 CodeCatalyst 包含初始提交，該提交會將預設分支設定為`main`並包含範例程式碼 (如果存放庫是使用包含範例程式碼的藍圖為專案建立的)，或是儲存庫 README 檔案的範本 markdown 檔案。您可以在主控台和 Git 用戶端中建立其他分支。您可以在控制台中創建和編輯文件，並在開發環境和 Git 客戶端中刪除文件。

我想要一個不同的分支作為我的默認分支

問題：我的源代碼存儲庫附帶了一個名為的默認分支`main`，但我想要一個不同的分支作為我的默認分支。

可能的修正：您無法在中變更或刪除來源儲存庫中的預設分支 CodeCatalyst。您可以建立其他分支，並在工作流程的來源動作中使用這些分支。您還可以選擇鏈接 GitHub 存儲庫並將其用作項目的存儲庫。

我收到有關提取請求中活動的電子郵件

問題：我沒有註冊或設定關於提取請求活動的電子郵件通知，但我仍然會收到它們。

可能的修正：會自動傳送有關提取要求活動的電子郵件通知。如需詳細資訊，請參閱 在 Amazon 中使用拉取請求 CodeCatalyst。

我忘記了我的個人訪問令牌（PAT）

問題：我一直在使用 PAT 來克隆，推送和提取源代碼存儲庫，但是我丟失了令牌的值，並且在 CodeCatalyst 控制台中找不到它。

可能的修復：解決此問題的最快方法是創建另一個 PAT 並配置憑據管理器或 IDE 以使用此新 PAT。我們只會在您建立 PAT 時顯示它的值。如果您遺失此值，則無法擷取該值。如需詳細資訊，請參閱 在 Amazon 中管理個人訪問令牌 CodeCatalyst。

拉取請求不會顯示我期望的更改

問題：我創建了一個拉請求，但我沒有看到我希望在源分支和目標分支之間看到的更改。

可能的修復：這可能是由許多問題引起的。請嘗試下列一或多個解決方案：

- 您可能正在檢閱較舊版本之間的變更，或者您可能沒有檢視最新的變更。重新整理瀏覽器，並確定您已選擇要檢視的修訂版本之間的比較。
- 並非提取要求中的所有變更都可以顯示在主控台中。例如，您無法在主控台中檢視 Git 子模組，因此無法檢視提取要求中子模組的差異。某些差異可能太大而無法顯示。如需詳細資訊，請參閱 來源儲存庫的配額 CodeCatalyst 及 檢視檔案。
- 提取請求會顯示合併基礎與您選擇的任何修訂之間的差異。當您建立提取要求時，顯示的差異在於來源分支的尖端與目標分支尖端之間的差異。一旦建立了提取請求，顯示的差異就是修訂版本與其合併基礎之間。合併基礎是創建修訂時，作為目標分支提示的提交。合併基準可以在修訂之間進行變更。有關 Git 中差異和合併基礎的更多信息，請參閱 Git 文檔git-merge-base中的。

提取請求顯示「不可合併」的狀態

問題：我想合併一個拉請求，但其狀態顯示為不可合併。

可能的修復：這可能是由一個或多個問題引起的：

- 您提取請求的所有必要審核者都必須先核准提取請求，才能合併提取請求。檢閱名稱旁邊有時鐘圖示的任何審核者的必要審核者清單。時鐘圖示表示審核者尚未核准提取請求。

> **Note**
>
> 如果在核准提取請求之前，已從專案中移除必要的複查者，您就無法合併提取請求。關閉提取請求並創建一個新的提取請求。

- 來源分支和目的地分支之間可能會發生合併衝突。 CodeCatalyst 不支持所有可能的 Git 合併策略和選項。您可以評估開發環境中的合併衝突分支，或複製存放庫，並使用 IDE 或 Git 工具來查找和解決合併衝突。如需更多詳細資訊，請參閱 合併提取請求。

疑難排解專案和藍圖

本節可協助您疑難排解在 Amazon CodeCatalyst 中處理專案和藍圖時可能遇到的一些常見問題。

Java API 與AWS Fargate藍圖缺少阿帕奇馬文 -3.8.6 的依賴關係

問題：對於從具有AWS Fargate藍圖的 Java API 建立的專案，工作流程會失敗，並顯示遺失apache-maven-3.8.6相依性的錯誤。工作流程失敗，輸出類似下列範例：

```
Step 8/25 : RUN wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-
maven-3.8.6-bin.tar.gz -P /tmp
---> Running in 1851ce6f4d1b
[91m--2023-03-10 01:24:55--  https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz
[0m[91mResolving dlcdn.apache.org (dlcdn.apache.org)...
[0m[91m151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443...
[0m[91mconnected.
[0m[91mHTTP request sent, awaiting response... [0m[91m404 Not Found
2023-03-10 01:24:55 ERROR 404: Not Found.
[0mThe command '/bin/sh -c wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz -P /tmp' returned a non-zero code: 8
[Container] 2023/03/10 01:24:55 Command failed with exit status 8
```

解決方案：使用下列步驟更新藍圖 Docker 檔案。

1. 在搜索欄中，輸入apache-maven-3.8.6以在使用具有藍圖的 Java API 創建的項目中找到碼頭文件。AWS Fargate
2. 更新 Dockerfile (/static-assets/app/Dockerfile) maven:3.9.0-amazoncorretto-11 作為基礎映像檔使用，並移除對套件的相依性。apache-maven-3.8.6

3. (建議使用) 我們也建議您將 Maven 堆積大小更新為 6 GB。

下面是一個例子碼頭文件。

```
FROM maven:3.9.0-amazoncorretto-11 AS builder

COPY ./pom.xml ./pom.xml
COPY src ./src/

ENV MAVEN_OPTS='-Xmx6g'

RUN mvn -Dmaven.test.skip=true clean package

FROM amazoncorretto:11-alpine

COPY –from=builder target/CustomerService-0.0.1.jar CustomerService-0.0.1.jar
EXPOSE 80
CMD ["java","-jar","-Dspring.profiles.active=prod","/CustomerService-0.0.1.jar", "–
server.port=80"]
```

現代三層 Web 應用程式藍圖工作流程OnPullRequest失敗，並顯示 Amazon 的許可錯誤 CodeGuru

問題：當我嘗試為我的專案執行工作流程時，工作流程無法執行，並顯示下列訊息：

```
Failed at codeguru_codereview: The action failed during runtime. View the action's logs
 for more details.
```

解決方案：此動作失敗的一個可能原因可能AWS 帳戶是由於 IAM 角色政策中缺少許可，因為您在連線 CodeCatalyst 中使用的服務角色版本缺少 codeguru_codereview 動作成功執行的必要權限。若要修正此問題，必須使用必要的許可更新服務角色，或者您必須將用於工作流程的服務角色變更為具有 Amazon CodeGuru 和 Amazon CodeGuru Reviewer 所需許可的服務角色。使用下列步驟尋找您的角色並更新角色原則權限，以允許工作流程成功執行。

> **Note**
>
> 這些步驟適用於下列中的工作流程 CodeCatalyst：

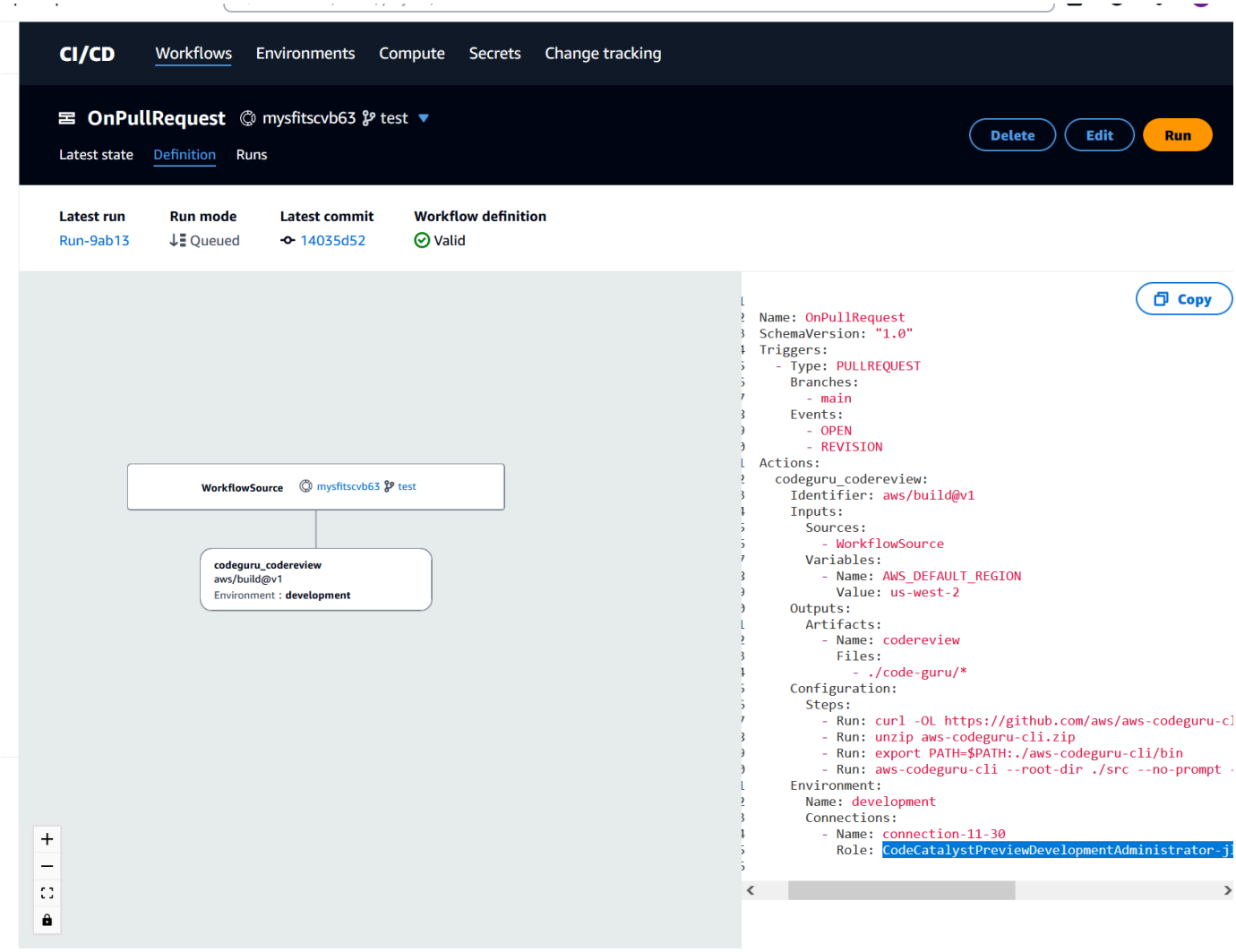
- 針對使用中 CodeCatalyst的 Modern 三層 Web 應用OnPullRequest程式藍圖建立的專案所提供的工作流程。
- 使用可存取 Amazon CodeGuru 或 Amazon CodeGuru 審核者 CodeCatalyst 的動作新增至專案的工作流程。

每個專案都包含工作流程，其中包含動作，這些動作使用AWS 帳戶連接至中的專案所提供的角色和環境 CodeCatalyst。包含動作及其指定原則的工作流程會儲存在 /.codealys/工作流程目錄中的來源儲存庫中。除非您要將新的角色識別碼新增至現有的工作流程，否則不需要修改工作流程 YAML。如需 YAML 範本元素和格式的詳細資訊，請參閱工作流定義參考。

這些是編輯角色原則和驗證工作流程 YAML 所需遵循的高階步驟。

在工作流程 YAML 中參照您的角色名稱並更新原則

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。導航到您的項目。
3. 選擇 CI/CD，然後選擇「工作流程」。
4. 選擇標題為的工作流程OnPullRequest。選擇 Definition (定義) 索引標籤。
5. 在工作流程 YAML 中，在 codeguru_codereview 動作下的`Role:`欄位中，記下角色名稱。這是您將在 IAM 中修改之政策的角色。下列範例顯示角色名稱。



6. 執行下列任意一項：

- (建議) 使用 Amazon CodeGuru 和 Amazon CodeGuru 審核者所需的許可更新與專案連線的服務角色。該角色將具有附加唯CodeCatalystWorkflowDevelopmentRole-*spaceName*一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。繼續執行後續步驟以更新 IAM 中的政策。

  > **Note**
  >
  > 您必須具有角色和策略AWS 帳戶的AWS管理員存取權。

- 將用於工作流程的服務角色變更為具有 Amazon CodeGuru 和 Amazon CodeGuru 審核者所需許可的服務角色，或使用所需許可建立新角色。

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

   在 IAM 主控台中，尋找步驟 5 中的角色，例如CodeCatalystPreviewDevelopmentRole。

8. 在步驟 5 的角色中，變更權限原則以包含codeguru-reviewer:*和codeguru:*權限。新增這些權限之後，權限原則看起來應該類似下列內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "apigateway:*",
                "ecr:*",
                "ecs:*",
                "ssm:*",
                "codedeploy:*",
                "s3:*",
                "iam:DeleteRole",
                "iam:UpdateRole",
                "iam:Get*",
                "iam:TagRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:CreatePolicyVersion",
                "iam:DeletePolicyVersion",
                "iam:PutRolePermissionsBoundary",
                "iam:DeleteRolePermissionsBoundary",
                "sts:AssumeRole",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule",
                "cloudwatch:DescribeAlarms",
                "sns:Publish",
```

```
                "sns:ListTopics",
                "codeguru-reviewer:*",
                "codeguru:*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

9. 進行原則更正後，請返回 CodeCatalyst 並重新啟動工作流程。

還在尋找解決您的問題嗎？

你可以去 Amazon CodeCatalyst 或填寫 Sup port 反饋表。在「請求信息」部分的「我們如何為您提供幫助」下，包括您是 Amazon CodeCatalyst 客戶。盡可能提供詳細資訊，以便我們能夠最有效地解決您的問題。

工作流程的疑難排解

請參閱以下各節，以疑難排解與 Amazon 中工作流程相關的問題 CodeCatalyst。如需工作流程的相關詳細資訊，請參閱 ???。

主題

- 如何修正「工作流程處於非作用中狀態」訊息？
- 如何修復「工作流定義有 n 個錯誤」錯誤？
- 如何修復「找不到憑據」和「ExpiredToken」錯誤？
- 如何修復「無法連接到服務器」錯誤？
- 為什麼可視化編輯器中缺少 CodeDeploy 字段？
- 如何修正 IAM 功能錯誤？
- 我如何解決「npm 安裝」錯誤？
- 為什麼多個工作流程具有相同的名稱？
- 我可以將工作流程定義檔案儲存在其他資料夾中嗎？
- 如何將動作依序新增至我的工作流程？
- 為什麼我的工作流程在執行階段成功驗證但失敗？

- 自動探索不會針對我的動作探索任何報告
- 設定成功準則後，我的動作在自動探索的報表上失敗
- 「自動探索」會產生我不想要的報告
- 自動探索會針對單一測試架構產生許多小型報告
- CI/CD 下列出的工作流程與來源儲存庫中的工作流程不符
- 我無法建立或更新工作流程

如何修正「工作流程處於非作用中狀態」訊息？

問題：在 CodeCatalyst 主控台的 CI/CD、工作流程下，您的工作流程會顯示下列訊息：

`Workflow is inactive.`

此訊息指出工作流程定義檔包含不適用於您目前所在分支的觸發程序。例如，您的工作流程定義文件可能包含引用您的`main`分支的`PUSH`觸發器，但您位於功能分支上。由於您在功能分支中進行的更改不適用於`main`，並且不會啟動工作流運行`main`，因此 CodeCatalyst 取消委任分支上的工作流並將其標記為`Inactive`。

可能的修正：

如果要在功能分支上啟動工作流程，可以執行以下操作：

- 在功能分支的工作流程定義檔案中，從`Triggers`區段中移除`Branches`性質，使其看起來像這樣：

  ```
  Triggers:
    - Type: PUSH
  ```

  此配置導致觸發器在推送到任何分支（包括功能分支）時激活。如果觸發器已啟動，則 CodeCatalyst 會使用工作流程定義檔案和來源檔案，在您要推送的任何分支中啟動工作流程執行。
- 在功能分支的工作流程定義檔案中，移除`Triggers`區段並手動執行工作流程。
- 在您的特徵分支中，在工作流程定義檔案中，變更`PUSH`截面，使其參照您的特徵分支`main`，而不是其他分支 (例如)。

> **Important**
>
> 如果您不打算將它們合併回`main`分支，請小心不要提交這些更改。

若要取得有關編輯工作流程定義檔的更多資訊，請參閱若要編輯工作流程。

關於觸發條件的詳細資訊，請參閱 使用觸發程序。

如何修復「工作流定義有 *n* 個錯誤」錯誤？

問題：您看到下列任何錯誤訊息：

錯誤 1：

在 [C I/CD, 工作流程] 頁面的工作流程名稱下，您會看到：

```
Workflow definition has n errors
```

錯誤二：

編輯工作流程時，請選擇「驗證」按鈕， CodeCatalyst 主控台頂端會顯示下列訊息：

```
The workflow definition has errors. Fix the errors and choose Validate to
verify your changes.
```

錯誤三：

導覽至工作流程的詳細資訊頁面後，您會在「工作流程定義」欄位中看到下列錯誤：

```
n errors
```

可能的修正：

- 選擇 CI/CD，選擇「工作流程」，然後選擇發生錯誤的工作流程名稱。在頂部附近的「工作流程定義」欄位中，選擇錯誤的連結。有關錯誤的詳細資訊會顯示在頁面底部。請依照錯誤中的疑難排解提示來修正問題。
- 請確定工作流程定義檔案是 YAML 檔案。
- 請確定工作流程定義檔案中的 YAML 屬性嵌套在正確的層級。若要查看工作流程定義檔案中的性質應如何巢狀化工作流定義參考，請參閱，或參閱連結至的動作文件新增動作。
- 確保正確逸出星號 (*) 和其他特殊字元。要將它們轉義，請添加單引號或雙引號。例如：

```
Outputs:
  Artifacts:
```

```
    - Name: myartifact
      Files:
        - "**/*"
```

如需工作流程定義檔案中特殊字元的詳細資訊，請參閱語法指南和慣例。

- 請確定工作流程定義檔案中的 YAML 屬性使用正確的大小寫。如需大小寫規則的詳細資訊，請參閱語法指南和慣例。若要判斷每個屬性的正確大小寫工作流定義參考，請參閱，或參閱連結至的動作文件新增動作。
- 請確定SchemaVersion屬性存在，並在工作流程定義檔案中設定為正確的版本。如需詳細資訊，請參閱 SchemaVersion。
- 請確保工作流程定義文件中的Triggers部分包含所有必需的屬性。若要判斷必要的屬性，請在視覺化編輯器中選擇觸發程式，然後尋找缺少資訊的欄位，或參閱中的觸發程式參照文件Triggers。
- 請確保工作流程定義文件中的DependsOn屬性配置正確，并且不引入循環相依性。如需詳細資訊，請參閱 將動作配置為依賴其他動作。
- 請確定工作流程定義檔案中的Actions區段至少包含一個動作。如需詳細資訊，請參閱 動作。
- 請確定每個動作都包含所有必要的屬性。若要判斷必要的屬性，請在視覺化編輯器中選擇動作，然後尋找缺少資訊的欄位，或參閱您動作的文件 (連結至來源) 新增動作。
- 請確定所有輸入成品都有對應的輸出加工品。如需詳細資訊，請參閱 定義輸出人工因素。
- 請確定已匯出在某個動作中定義的變數，以便在其他動作中使用這些變數。如需詳細資訊，請參閱匯出變數，以便其他動作可以使用它。

如何修復「找不到憑據」和「ExpiredToken」錯誤？

問題：進行工作時教學課程：將應用程式部署到 Amazon EKS，您會在開發電腦的終端機視窗中看到下列其中一個或兩個錯誤訊息：

```
Unable to locate credentials. You can configure credentials by running "aws configure".
```

```
ExpiredToken: The security token included in the request is expired
```

可能的修正：

這些錯誤表示您用來存取 AWS 服務的認證已過期。在此情況下，請勿執行aws configure命令。相反，請使用以下說明刷新您的 AWS 訪問密鑰和會話令牌。

若要重新整理存 AWS 取金鑰和工作階段權杖

1. 請確定您擁有完整的 Amazon EKS 教學 (codecatalyst-eks-user) 使用者的 AWS 存取入口網站 URL、使用者名稱和密碼。完成教學課程後，您應該已步驟 1：設定您的開發機器設定這些項目。

   > **Note**
   >
   > 如果您沒有此信息，請轉到 IAM 身份中心的codecatalyst-eks-user詳細信息頁面，選擇「重置密碼」，「生成一次性密碼 [...] ，然後再次重設密碼以在螢幕上顯示資訊。

2. 執行以下任意一項：

   - 將 AWS 訪問門戶網址粘貼到瀏覽器的地址欄中。

     或

   - 如果已載入，請重新整理 AWS 存取入口網站頁面。

3. 如果您尚未登入，請使用codecatalyst-eks-user的使用者名稱和密碼登入。
4. 選擇 AWS 帳戶，然後選擇您 AWS 帳戶 為其指派codecatalyst-eks-user使用者和權限集的名稱。
5. 在權限集名稱 (codecatalyst-eks-permission-set) 旁，選擇 [命令列] 或 [程式設計存取]。
6. 複製頁面中間的命令。它們看起來類似於以下內容：

   ```
   export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
   export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
   export AWS_SESSION_TOKEN="session-token"
   ```

   ... 其中####是一個長隨機字符串。
7. 將命令粘貼到開發計算機上的終端提示符中，然後按 Enter 鍵。

   新的密鑰和會話令牌被加載。

   您現在已重新整理認證。 AWS CLIeksctl、和kubectl指令現在應該可以運作。

如何修復「無法連接到服務器」錯誤？

問題：在完成中所述的教學課程時教學課程：將應用程式部署到 Amazon EKS，您會在開發電腦的終端機視窗中看到類似下列內容的錯誤訊息：

`Unable to connect to the server: dial tcp: lookup long-string.gr7.us-west-2.eks.amazonaws.com on 1.2.3.4:5: no such host`

可能的修正：

此錯誤通常表示公用kubectl程式用於連接 Amazon EKS 叢集的登入資料已過期。若要解決此問題，請在終端機提示字元中輸入下列命令來重新整理認證：

```
aws eks update-kubeconfig --name codecatalyst-eks-cluster --region us-west-2
```

其中：

- *codecatalyst-eks-cluster*已取代為您的 Amazon EKS 叢集的名稱。
- *us-west-2* 會取代為部署叢集的 AWS 區域。

為什麼可視化編輯器中缺少 CodeDeploy 字段？

問題：您正在使用「部署到 Amazon ECS」動作，但在工作流程的視覺化編輯器CodeDeploy AppSpec中看不到這些 CodeDeploy 欄位。因為您在「服務」欄位中指定的 Amazon ECS 服務未設定為執行藍/綠部署，可能會發生此問題。

可能的修正：

- 在「部署到 Amazon ECS」動作的「組態」索引標籤上選擇不同的 Amazon ECS 服務。如需詳細資訊，請參閱 新增「部署到 Amazon ECS」動作。
- 將選取的 Amazon ECS 服務設定為執行藍/綠部署。如需有關設定藍/綠部署的詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南 CodeDeploy中的藍/綠部署。

如何修正 IAM 功能錯誤？

問題：您正在使用「部署 AWS CloudFormation 堆疊」動作，而且您會`##[error] requires capabilities: [capability-name]`在部署 AWS CloudFormation 堆疊動作的記錄檔中看到。

可能的修正：請完成下列程序，將功能新增至工作流程定義檔案。如需 IAM 功能的詳細資訊，請參閱 IAM 使用者指南中的 AWS CloudFormation 範本確認 IAM 資源。

Visual

使用視覺化編輯器新增 IAM 功能

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 [視覺]。
7. 在工作流程圖中，選擇您的部署 AWS CloudFormation 堆疊動作。
8. 選擇 Configuration (組態) 索引標籤。
9. 在底部，選擇 [進階]-[選用]。
10. 在 [權能] 下拉式清單中，選取錯誤訊息中提到的功能旁邊的核取方塊。如果該功能在清單中沒有可用，請使用 YAML 編輯器加入它。
11. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
12. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。
13. 如果新的工作流程執行未自動啟動，請手動執行工作流程以查看變更是否可修正錯誤。如需有關手動執行工作流程的更多資訊，請參閱啟動工作流程執行。

YAML

若要使用 YAML 編輯器新增身分與存取權管理功能

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 選擇您的專案。
3. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。
4. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。
5. 選擇編輯。
6. 選擇 YAML。
7. 在部署 AWS CloudFormation 堆疊動作中，新增`capabilities`屬性，如下所示：

```
DeployCloudFormationStack:
  Configuration:
    capabilities: capability-name
```

將功####代為錯誤訊息中顯示的 IAM 功能名稱。請使用逗號且不能使用空格來列出多個權能。如需詳細資訊，請參閱中的capabilities性質說明「部署AWS CloudFormation堆疊」動作參考。

8. (選擇性) 選擇 [驗證]，在認可之前驗證工作流程的 YAML 程式碼。
9. 選擇「確認」，輸入確認訊息，然後再次選擇「確認」。
10. 如果新的工作流程執行未自動啟動，請手動執行工作流程以查看變更是否可修正錯誤。如需有關手動執行工作流程的更多資訊，請參閱啟動工作流程執行。

我如何解決「npm 安裝」錯誤？

問題：您的AWS CDK 部署動作或AWS CDK 啟動程序動作失敗，並顯示錯npm install誤。可能會發生此錯誤，是因為您將 AWS CDK 應用程序依賴項存儲在無法通過操作訪問的私有節點包管理器（npm）註冊表中。

可能的修復：使用以下說明使用其他註冊表和身份驗證信息更新 AWS CDK 應用程序的cdk.json文件。

開始之前

1. 為您的身份驗證信息創建密碼。您將在cdk.json檔案中參考這些密碼，而不是提供純文字對等項。要創建秘密：
   a. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
   b. 選擇您的專案。
   c. 在功能窗格中，選擇 CI/CD，然後選擇 [密碼]。
   d. 使用下列屬性建立兩個密碼：

第一個秘密	第二個秘密
名稱: npmUsername	名稱: npmAuthToken

第一個秘密	第二個秘密
值：*npm-###### nPM-### ########* *npm* 註冊表進行身份驗證的用戶名。 (選擇性) 說明：`The username used to authenticate to the private npm registry.`	值：*npm-auth-token* ，其中*npm-auth-token* 是用於向私人 npm 註冊表進行身份驗證的訪問令牌。有關 npm 訪問令牌的更多信息，請參閱關於 npm 文檔中的訪問令牌。 (選擇性) 說明：`The access token used to authenticate to the private npm registry.`

如需有關密碼的詳細資訊，請參閱與秘密一起工作。

2. 將密碼作為環境變數新增至您的 AWS CDK 動作。動作會在執行時以實數值取代變數。若要新增密碼：

   a. 在瀏覽窗格中，選擇 CI/CD，然後選擇 [工作流程]。

   b. 選擇工作流程的名稱。您可以依定義工作流程的來源儲存庫或分支名稱進行篩選，或依工作流程名稱進行篩選。

   c. 選擇編輯。

   d. 選擇 [視覺]。

   e. 在工作流程圖中，選擇您的 AWS CDK 動作。

   f. 選擇輸入索引標籤。

   g. 添加兩個具有以下屬性的變量：

第一個變數	第二個變數
名稱: NPMUSER Value (值)：`${Secrets.npmUsername}`	名稱: NPMTOKEN Value (值)：`${Secrets.npmAuthToken}`

您現在有兩個包含秘密參照的變數。

您的工作流程定義檔 YAML 程式碼看起來應該類似下列：

> **Note**
>
> 下列程式碼範例來自啟動程AWS CDK 序動作；AWS CDK 部署動作看起來會類似。

```
Name: CDK_Bootstrap_Action
SchemaVersion: 1.0
Actions:
  CDKBootstrapAction:
    Identifier: aws/cdk-bootstrap@v1
    Inputs:
      Variables:
        - Name: NPMUSER
          Value: ${Secrets.npmUsername}
        - Name: NPMTOKEN
          Value: ${Secrets.npmAuthToken}
      Sources:
        - WorkflowSource
    Environment:
      Name: Dev2
      Connections:
        - Name: account-connection
          Role: codecatalystAdmin
    Configuration:
      Parameters:
        Region: "us-east-2"
```

您現在可以在cdk.json檔案中使用NPMUSER和NPMTOKEN變數了。轉到下一個程序。

若要更新您的檔案

1. 切換到 AWS CDK 項目的根目錄，然後打開cdk.json文件。
2. 找到"app":屬性，並將其更改為包含以####顯示的代碼：

> **Note**
>
> 下列範例程式碼來自 TypeScript 專案。如果您使用的是 JavaScript 項目，代碼看起來會相似，但不相同。

```
{
  "app": "npm set registry=https://your-registry/folder/CDK-package/ --
userconfig .npmrc && npm set //your-registry/folder/CDK-package/:always-auth=true
 --userconfig .npmrc && npm set //your-registry/folder/CDK-package/:_authToken=
\"${NPMUSER}\":\"${NPMTOKEN}\" && npm install  && npx ts-node --prefer-ts-exts bin/
hello-cdk.ts|js",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
...
```

3. 在####突出顯示的代碼中，替換：

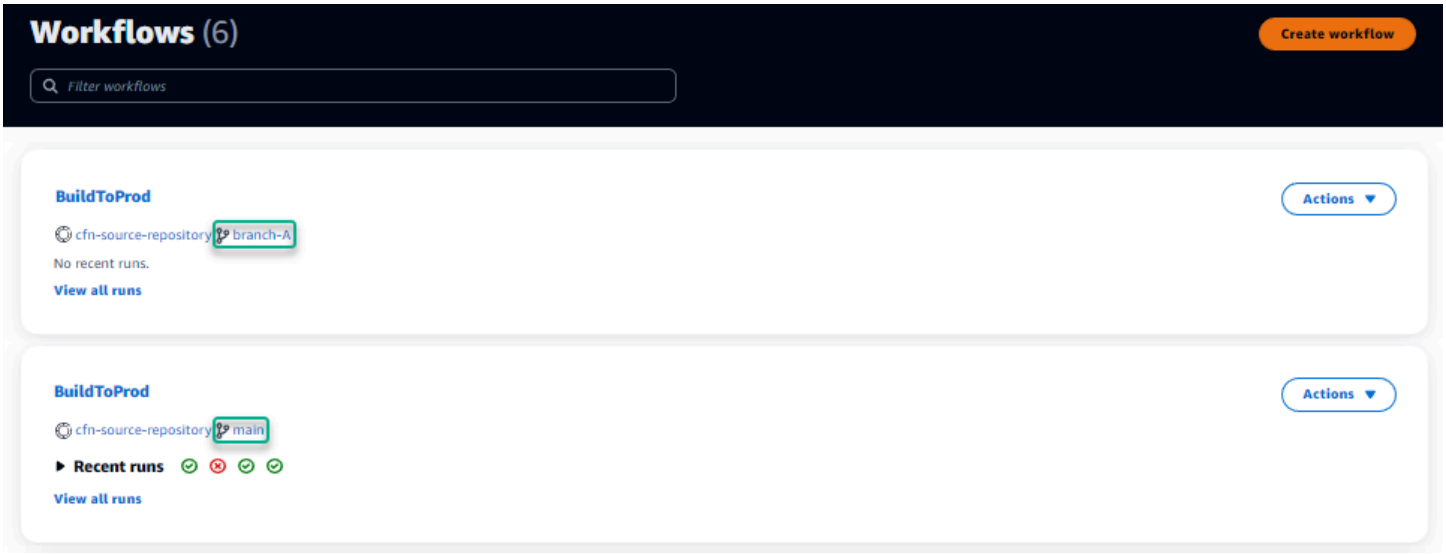
   - *#####/###/CDK #/#################*徑。 AWS CDK
   - 使用您的入口點*####### cdk.ts|.js#*這可能是.ts（TypeScript）或.js（JavaScript）文件，具體取決於您使用的語言。

   > **Note**
   >
   > 該操作將替換 ***NPMUSER # NPM TOKEN*** 變量與您在秘密中指定的 NPM 用戶名和訪問令牌。

4. 儲存您的 cdk.json 檔案。
5. 手動重新執行動作，以查看變更是否可修正錯誤。如需有關手動執行動作的詳細資訊，請參閱啟動工作流程執行。

為什麼多個工作流程具有相同的名稱？

工作流程儲存在每個儲存庫的每個分支 如果兩個不同的工作流程存在於不同分支中，則可以具有相同的名稱。在「工作流程」頁面中，您可以透過查看分支名稱來區分相同名稱的工作流程。如需詳細資訊，請參閱 與 Amazon 的分支機構合作 CodeCatalyst。



我可以將工作流程定義檔案儲存在其他資料夾中嗎？

否，您必須將所有工作流程定義檔案儲存在資.codecatalyst/workflows料夾中。如果您在多個邏輯專案中使用 mono 存放庫，請將所有工作流程定義檔案放在.codecatalyst/workflows資料夾中，然後使用觸發器內的FilesChanged屬性來觸發指定專案路徑的工作流程。如需詳細資訊，請參閱 使用觸發程序。

如何將動作依序新增至我的工作流程？

根據預設，當您將動作新增至工作流程時，該動作將沒有相依性，而且會與其他動作 parallel 執行。

如果您想要依序排列動作，您可以透過設定DependsOn欄位來設定另一個動作的相依性。您也可以配置動作以使用作為其他動作輸出的成品或變數。如需詳細資訊，請參閱 將動作配置為依賴其他動作。

為什麼我的工作流程在執行階段成功驗證但失敗？

如果您使用Validate按鈕驗證您的工作流程，但您的工作流程仍然失敗，這可能是因為驗證器中的限制。

在提交期間，工作流程配置中對 CodeCatalyst 資源（如機密，環境或叢集）引用的任何錯誤都不會註冊。如果使用任何無效的參考，則只有在執行工作流程時才會識別錯誤。同樣地，如果您的操作配置中

存在任何錯誤，例如缺少必填字段或操作屬性中的錯別字，則只有在運行工作流程時才會識別它們。如需詳細資訊，請參閱 建立、編輯和刪除工作流程。

自動探索不會針對我的動作探索任何報告

問題：我為執行測試的動作設定了自動探索，但未發現任何報告。 CodeCatalyst

可能的修復：這可能是由許多問題引起的。請嘗試下列一或多個解決方案：

- 確保用於運行測試的工具以 CodeCatalyst 理解的格式之一產生輸出。例如，如果您想pytest要允許 CodeCatalyst 探索測試和程式碼覆蓋率報告，請包含下列引數：

  ```
  --junitxml=test_results.xml --cov-report xml:test_coverage.xml
  ```

  如需詳細資訊，請參閱 測試報告類型。
- 請確定輸出的副檔名與所選格式一致。例如，當設定為pytest以JUnitXML格式產生結果時，請檢查副檔名是否為.xml。如需詳細資訊，請參閱 測試報告類型。
- 請確定已設定IncludePaths為包含整個檔案系統 (**/*)，除非您故意排除某些資料夾。同樣地，請確定ExcludePaths不排除您預期報表所在的目錄。
- 如果您手動將報告設定為使用特定輸出檔案，則會從自動探索中排除該報告。如需詳細資訊，請參閱範例：設定報表。
- 自動探索可能找不到報告，因為動作在產生任何輸出之前失敗。例如，在運行任何單元測試之前，構建可能已經失敗。

設定成功準則後，我的動作在自動探索的報表上失敗

問題：當我啟用自動探索並設定成功準則時，有些報告不符合成功準則，且動作會因此失敗。

可能的修正：若要解決此問題，請嘗試下列一或多個解決方案：

- 修ExcludePaths改IncludePaths或排除您不感興趣的報告。
- 更新成功條件以允許所有報告通過。例如，如果發現兩份報告，其中一個報告的線路涵蓋範圍為 50%，另一個則為 70%，請將最小線路涵蓋範圍調整為 50%。如需更多資訊，請參閱成功條件
- 將失敗報告轉換為手動設定的報告。這可讓您為該特定報告設定不同的成功準則。如需詳細資訊，請參閱 設定報告的成功準則。

「自動探索」會產生我不想要的報告

問題：當我啟用自動探索時，它會產生我不想要的報告。例如，為存儲在中的應用程序依賴項中包含的文件 CodeCatalyst 生成代碼覆蓋率報告node_modules。

可能的修復：您可以調整ExcludePaths配置以排除不需要的文件。例如，要排除node_modules，請添加node_modules/**/*。如需詳細資訊，請參閱 包含/排除路徑。

自動探索會針對單一測試架構產生許多小型報告

問題：當我使用某些測試和代碼覆蓋率報告框架時，我注意到自動發現會生成大量報告。例如，當使用 Maven Surefire 外掛程式時，自動探索會為每個測試類別產生不同的報告。

可能的修復：您的框架可能能夠將輸出彙總到單個文件中。例如，如果您使用的是 Maven Surefire 插件，則可以使用npx junit-merge手動聚合文件。完整的表達式可能如下所示：

```
mvn test; cd test-package-path/surefire-reports && npx junit-merge -d ./ && rm
 *Test.xml
```

CI/CD 下列出的工作流程與來源儲存庫中的工作流程不符

問題：「CI/CD，工作流程」頁面上顯示的工作流程與來源儲存庫中~/.codecatalyst/workflows/資料夾中的工作流程不符。您可能會看到下列不相符項目：

- 工作流程會顯示在「工作流程」頁面上，但來源儲存庫中不存在對應的工作流程定義檔案。
- 工作流程定義檔案存在於您的來源儲存庫中，但對應的工作流程不會顯示在「工作流程」頁面上。
- 工作流程同時存在於來源儲存庫和「工作流程」頁面中，但兩者不同。

如果 [工作流程] 頁面沒有時間重新整理，或超過工作流程配額，就可能會發生這個問題。

可能的修正：

- 等候。在「工作流程」頁面上看到「變更」之前，您通常必須等待兩到三秒鐘。
- 如果您已超出工作流程配額，請執行下列其中一項操作：

> **Note**
>
> 若要判斷是否超過工作流程配額中工作流程的配額 CodeCatalyst，請針對來源儲存庫或「工作流程」頁面中的工作流程檢閱並交叉檢查記錄的配額。沒有錯誤訊息表示已超出配額，因此您必須自行調查。

- 如果您已超過每個空間配額的工作流程數目上限，請刪除某些工作流程，然後針對工作流程定義檔執行測試認可。測試提交的一個例子可能是向文件添加一個空格。
- 如果您已超過工作流程定義檔案大小上限配額，請變更工作流程定義檔案以縮短其長度。
- 如果您已超過單一來源事件配額中處理的工作流程檔案數目上限，請執行數次測試認可。修改少於每次提交中最大數目的工作流程。
- 開啟付費方案計費，以增加工作流程配額。如需詳細資訊，請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。

我無法建立或更新工作流程

問題：我想要建立或更新工作流程，但是在嘗試提交變更時看到錯誤訊息。

可能的修正：視您在專案或空間中的角色而定，您可能沒有將程式碼推送至專案中原始碼儲存庫的權限。工作流程的 YAML 檔案會儲存在儲存庫中。如需詳細資訊，請參閱 工作流程定義檔。S pace 系統管理員角色、專案管理員角色和參與者角色都具有提交和推送程式碼至專案中存放庫的權限。

如果您具有 Contributor 角色，但無法在特定分支中建立或認可對工作流程 YAML 的變更，則可能會針對該分支設定分支規則，防止具有該角色的使用者將程式碼推送至該特定分支。嘗試在不同的分支中創建工作流程，或將您的更改提交到不同的分支。如需更多詳細資訊，請參閱 使用分支規則管理分支允許的操作。

疑難排解搜尋中的問題 CodeCatalyst

請參閱下列各節，以疑難排解與中搜尋相關的問題 CodeCatalyst。如需工作流程的相關詳細資訊，請參閱 在中搜尋 CodeCatalyst。

主題

- 我在專案中找不到使用者
- 我在專案或空間中看不到我要尋找的內容

我在專案中找不到使用者

問題：當我嘗試查看用戶的詳細信息時，我在項目中看不到他們的信息。

可能的修正：搜尋目前不支援在專案中搜尋使用者。若要搜尋可存取您空間的使用者，請切換至中的這個空間 QuickSearch，或移除您可能使用進階查詢語言指定的任何專案篩選器。

我在專案或空間中看不到我要尋找的內容

問題：當我嘗試搜索特定信息時，結果不會顯示。

可能的修正：內容更新可能需要幾秒鐘的時間才能在搜尋結果中更新。大型更新可能需要幾分鐘的時間。

對於最近尚未更新的資源，您可能需要縮小搜尋範圍。您可以新增更多關鍵字或使用進階查詢語言來精簡化。如需有關精簡查詢的更多資訊，請參閱精簡您的搜尋查詢。

當我瀏覽頁面時，搜索結果的數量不斷變化

問題：當我進入下一頁時，搜索結果的數量似乎會發生變化，因此不清楚總結果有多少。

可能的修正：瀏覽搜尋結果頁面時，您可能會看到符合您查詢條件的搜尋結果數目有所變更。結果數量可能會更新，以反映在您瀏覽頁面時發現的更準確的相符項目數。

瀏覽結果時，您可能會看到下列訊息：「test」沒有結果。如果您無法存取剩餘的結果，您將會收到訊息。

我的搜尋查詢尚未完成

問題：我的搜索查詢的結果沒有顯示出來，並且似乎花費了太長時間。

可能的修正：當空間中同時進行許多搜尋時 (無論是以程式設計方式或是因為高團隊活動)，您的搜尋可能無法完成。如果您正在執行程式化搜尋，請暫停或減少搜尋。否則，請在幾秒鐘後再試一次。

疑難排解與您的空間相關聯的帳戶問題

在中 CodeCatalyst，您可以在您的空間中新增一個AWS 帳戶，以授與資源的權限以及用於計費目的。下列資訊可協助您疑難排解中關聯帳戶的常見問題 CodeCatalyst。

主題

我的AWS 帳戶連接請求收到了無效的令牌錯誤

問題：使用連接令牌創建連接請求時，頁面不接受令牌並顯示錯誤，指出令牌無效。

可能的修正：請確定您提供要新增至空間的帳戶 ID。您必須具有您的系統管理權限，AWS 帳戶或能夠與管理員合作才能新增帳戶。

當您選擇驗證帳戶時，將在中打開一個新的瀏覽器窗口AWS Management Console。需要在控制台端登錄相同的帳戶。請在確認下列項目後再試一次：

- 您已使AWS Management Console用您要新增至空間AWS 帳戶的相同方式登入。
- 您已選取美國西部 (奧勒岡) AWS 區域 (`us-west-2`) 登入。AWS Management Console
- 如果您已從帳單頁面到達，且想要將空間的帳單帳戶新增AWS 帳戶為指定的帳單帳戶，請確定該帳戶不是另一個空間的帳單帳戶。

我的 Amazon CodeCatalyst 專案工作流程失敗，並顯示設定的帳戶、環境或 IAM 角色發生錯誤

問題:當工作流程執行且找不到與您的空間相關聯的已設定帳戶或 IAM 角色時，您必須在工作流程 YAML 中手動填寫角色、連線和環境欄位。檢視失敗的工作流程動作，並注意錯誤訊息是否如下：

- 角色無法用於與環境相關聯的連線。
- 動作未成功。狀態：失敗；提供的帳戶連線或環境值無效。確認連接與您的空間相關聯，並且環境與您的專案相關聯。
- 動作未成功。狀態：失敗；提供的 IAM 角色值無效。驗證名稱是否存在，IAM 角色會新增至您的帳戶連線，且連線已與您的 Amazon CodeCatalyst 空間相關聯

可能的修正：請確定工作流程 YAML 欄位具有正確的環境、連線和角色值。需要環境的 CodeCatalyst 工作流程動作包括執行AWS資源或產生資AWS源堆疊的建置或部署動作。

選擇失敗的工作流程動作區塊，然後選擇 [視覺]。選擇 Configuration (組態) 索引標籤。如果未填入「環境」、「連線名稱」和「角色名稱」欄位，則您將需要手動更新工作流程。請使用下列步驟來編輯您的工作流程 YAML：

- 展開目/.codecatalyst錄，然後展開目/workflows錄。開啟工作流程 YAML 檔案。請確定已在您為工作流程設定的 YAML 中指定 IAM 角色和帳戶資訊。範例：

```
Actions:
  cdk_bootstrap:
    Identifier: action-@v1
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: Staging
      Connections:
        - Name: account-connection
          Role: build-role
```

  若要執行工作 CodeCatalyst 流程建置和部署具有AWS資源的動作，必須具備「環境」、「連線」和「角色」內容 如需範例，請參閱環境、連線和角色的 CodeCatalyst 建置動作參考 YAML 參數。
- 請確定您的空間已新增一個帳戶，並確定該帳戶具有新增至該帳戶的適當 IAM 角色或角色。如果您具有 S pace 管理員角色，則可以調整或新增帳戶。如需詳細資訊，請參閱 管理 AWS 帳戶 空間。

我需要相關聯的帳戶、角色和環境來建立專案

問題：在專案建立選項中，我的專案在我的空間中沒有可用的新增帳戶，或者我需要將另一個帳戶新增至我的空間以供專案使用。

可能的修復：如果您具有 Space 管理員角色，則可以為您的空間添加授權AWS 帳戶以將其添加到您的項目中。您還必須擁有具有管理權限的AWS 帳戶地方，或者可以與AWS管理員合作。

若要確保專案建立畫面中可以使用帳戶和角色，您必須先新增帳戶和角色。如需詳細資訊，請參閱 管理 AWS 帳戶 空間。

您可以選擇使用稱為角色原則的角色原則來建立服務CodeCatalystWorkflowDevelopmentRole-*spaceName*角色。該角色將具有附加

唯CodeCatalystWorkflowDevelopmentRole-*spaceName*一標識符的名稱。如需有關角色和角色原則的詳細資訊，請參閱了解服CodeCatalystWorkflowDevelopmentRole-***spaceName***務角色。如需建立角色的步驟，請參閱為您的帳戶和空間建立CodeCatalystWorkflowDevelopmentRole-***spaceName***角色。角色會新增至您的帳戶，並可在中的專案建立頁面中使用 CodeCatalyst。

我無法訪問 Amazon CodeCatalyst 空間頁面 AWS Management Console

問題：當我嘗試訪問中的 Amazon CodeCatalyst 頁面以將帳戶添加AWS Management Console到我的 CodeCatalyst 空間或向帳戶添加角色時AWS，我收到許可錯誤。

可能的修正：

對於您的空間，如果您具有 S pace 管理員角色，則可以添加授權AWS 帳戶以將其添加到您的項目中。您還必須擁有具有管理權限的AWS 帳戶地方，或者可以與AWS管理員合作。您必須先確定您已使AWS Management Console用您要管理的相同帳戶登入。登入後，您可以開啟主機AWS Management Console，然後再試一次。

在以下位置打開 Amazon CodeCatalyst 頁AWS Management Console面 https://us-west-2.console.aws.amazon.com/codecatalyst/home?region=us-west-2#/.

我想要一個不同的帳戶作為我的帳單帳戶

問題：當我設定 CodeCatalyst 登入時，我完成了幾個步驟來設定我的空間並關聯授權AWS 帳戶。現在，我想授權一個不同的帳戶進行計費。

可能的修正：如果您具有 Space 管理員角色，您可以針對您的空間授權帳戶。您還必須擁有具有管理權限的AWS 帳戶地方，或者可以與AWS管理員合作。

如需詳細資訊，請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。

疑難排解開發環境的問題

請參閱下列各節，以疑難排解與開發環境相關的問題。如需有關開發環境的詳細資訊，請參閱開發環境 CodeCatalyst。

主題

- 由於配額問題，我的開發環境創建未成功
- 我無法將更改從我的開發環境推送到存儲庫中的特定分支
- 我的開發環境沒有恢復
- 我的開發環境已中斷

- 我的 VPC 連接開發環境失敗
- 我找不到我的項目所在的目錄
- 我無法通過 SSH 連接到我的開發環境
- 我無法通過 SSH 連接到我的開發環境，因為我的本地 SSH 配置丟失
- 我無法通過 SSH 連接到我的開發環境，因為我的codecatalyst配置文件有問題 AWS Config
- IDE 的疑難排解問題
- 開發檔案的疑難排解問題

由於配額問題，我的開發環境創建未成功

問題：我想在中創建一個開發環境 CodeCatalyst，但我看到一個錯誤。在主控台中，我在 [開發環境] 頁面上看到一則訊息，表示我已達到空間的儲存限制。

可能的修正：視您在專案或空間中的角色而定，您可以刪除一或多個自己的開發環境，或者如果您具有 Space 管理員角色，則可以刪除其他使用者建立的未使用的開發環境。您也可以決定將計費層級變更為包含更多儲存空間的方案。

- 若要檢視儲存限制，請檢視 Amazon CodeCatalyst 空間的 [帳單] 索引標籤，查看用量配額是否已達到允許的上限。如果配額已達到上限，請聯絡具有 Space 管理員角色的人員以移除不需要的開發環境，或考慮變更帳單層。
- 若要移除您建立不再需要的任何開發環境，請參閱刪除開發環境。

如果問題仍然存在，而您在 IDE 中出現錯誤，請檢查您是否具有可讓您建立開發環境的 CodeCatalyst 角色。S pace 管理員角色、專案管理員角色和參與者角色都具有建立開發環境的權限。如需詳細資訊，請參閱在 Amazon 中使用角色 CodeCatalyst。

我無法將更改從我的開發環境推送到存儲庫中的特定分支

問題：我想提交並推送我的開發環境中的代碼更改到源存儲庫中的分支，但我看到一個錯誤。

可能的修正：視您在專案或空間中的角色而定，您可能沒有將程式碼推送至專案中原始碼儲存庫的權限。S pace 系統管理員角色、專案管理員角色和參與者角色都具有將程式碼推送至專案中存放庫的權限。

如果您有 Contributor 角色，但無法將程式碼推送至特定分支，則可能會針對特定分支設定分支規則，防止具有該角色的使用者將程式碼推送至該特定分支。嘗試將更改推送到不同的分支，或創建一個分支，然後將代碼推送到該分支。如需詳細資訊，請參閱使用分支規則管理分支允許的操作。

我的開發環境沒有恢復

問題：停止後，我的開發環境沒有恢復。

可能的修正：若要修正問題，請檢視 Amazon S CodeCatalyst pace 的 [帳單] 索引標籤，查看使用量配額是否已達到上限。如果配額已達到上限，請聯絡您的 Space 管理員以提高帳單層級。

我的開發環境已中斷

問題：我的開發環境在我使用它時斷開連接。

可能的修正：若要修正問題，請檢查您的網際網路連線。如果您沒有連線到網際網路，請在開發環境中連線並繼續工作。

我的 VPC 連接開發環境失敗

問題：我將 VPC 連接關聯到我的開發環境，並且遇到錯誤。

可能的修正：Docker 使用稱為橋接網路的連結層裝置，可讓連接到相同橋接器網路的容器進行通訊。預設橋接器通常會使用 `172.17.0.0/16` 子網路來進行容器網路連線。如果環境執行個體的 VPC 子網路使用 Docker 已經使用的相同地址範圍，可能會發生 IP 地址衝突。若要解決由 Amazon VPC 造成的 IP 位址衝突，並Docker使用相同的 IPv4 CIDR 地址區塊，請設定與之不同的 CIDR 區塊。`172.17.0.0/16`

> **Note**
>
> 您無法變更現有 VPC 或子網路的 IP 位址範圍。

我找不到我的項目所在的目錄

問題：我找不到我的項目所在的目錄。

可能的修正：若要尋找您的專案，請將目錄變更為`/projects`。這是您可以找到項目的目錄。

我無法通過 SSH 連接到我的開發環境

若要疑難排解透過 SSH 連線至開發環境的問題，您可以使用`-vvv`選項執行`ssh`命令，以顯示有關如何解決問題的詳細資訊：

```
ssh -vvv codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
```

我無法通過 SSH 連接到我的開發環境，因為我的本地 SSH 配置丟失

如果您的本機 SSH 設定 (`~/.ssh/config`) 遺失或`Host codecatalyst-dev-env*`區段的內容已過期，您將無法透過 SSH 連線至您的開發環境。若要解決此問題，請刪除`Host codecatalyst-dev-env*`區段，然後再次從 SSH Access 模式執行第一個命令。如需詳細資訊，請參閱透過 SSH 連線至開發環境。

我無法通過 SSH 連接到我的開發環境，因為我的**codecatalyst**配置文件有問題 AWS Config

請確定設定`codecatalyst`檔的 AWS Config (`~/.aws/config`) 符合中所述的設定檔設定以使用 AWS CLI與 CodeCatalyst。如果沒有，請刪除的設定`codecatalyst`檔，然後再次從 SSH Access 強制回應執行第一個命令。如需詳細資訊，請參閱 透過 SSH 連線至開發環境。

IDE 的疑難排解問題

請參閱下列各節，以疑難排解與中 IDE 相關的問題 CodeCatalyst。如需 IDE 的詳細資訊，請參閱搭配 IDE 使用開發環境。

主題

- 我有不匹配的運行時圖像版本 AWS Cloud9
- 我無法訪問我的/projects/projects文件 AWS Cloud9
- 我無法AWS Cloud9使用自定義開發文件啟動我的開發環境
- 我遇到了問題 AWS Cloud9
- 在中 JetBrains，我無法通過連接到我的開發環境 CodeCatalyst
- 我無法AWS 工具組為我的 IDE 安裝
- 在我的 IDE 中，我無法啟動我的開發環境

我有不匹配的運行時圖像版本 AWS Cloud9

AWS Cloud9正在使用不同版本的前端資產和後端運行時圖像。使用不同的版本可能會導致 Git 擴充功能並無AWS 工具組法正常運作。若要修正此問題，請瀏覽至 [開發環境] 儀表板，停止您的開發環境，然後再次啟動它。若要使用 API 修正此問題，請使用 `UpdateDevEnvironment` API 更新執行階段。如需詳細資訊，請參閱 Amazon CodeCatalyst API 參考資料UpdateDevEnvironment中的。

我無法訪問我的**/projects/projects**文件 AWS Cloud9

編AWS Cloud9輯器無法存取目錄中的檔案`/projects/projects`。若要修正此問題，請使用AWS Cloud9終端機存取您的檔案，或將檔案移至其他目錄。

我無法AWS Cloud9使用自定義開發文件啟動我的開發環境

您的 devfile 映像檔可能與AWS Cloud9. 要解決此問題，請查看存儲庫和相應的開發環境中的開發文件，然後創建一個新的以繼續。

我遇到了問題 AWS Cloud9

對於其他問題，請查看使AWS Cloud9用指南中的疑難排解部分。

在中 JetBrains，我無法通過連接到我的開發環境 CodeCatalyst

若要修正此問題，請檢查您是否只 JetBrains 安裝了最新版本。如果您有多個版本，請解除安裝舊版本，然後關閉 IDE 和瀏覽器，再次註冊通訊協定處理常式。然後開啟 JetBrains 並再次註冊通訊協定處理常式。

我無法AWS 工具組為我的 IDE 安裝

若要修正 VS Code 的這個問題，請AWS Toolkit for Visual Studio Code從手動安裝GitHub。

若要修正此問題 JetBrains，請AWS Toolkit for JetBrains從手動安裝GitHub。

在我的 IDE 中，我無法啟動我的開發環境

若要修正 VS Code 的這個問題，請檢查您是否已AWS Toolkit for Visual Studio Code安裝最新版本的 VS Code。如果您沒有最新版本，請更新並啟動您的開發環境。有關更多信息，請參 CodeCatalyst 閱 Amazon VS 代碼。

若要修正此問題 JetBrains，請檢查您是否已AWS Toolkit for JetBrains安裝最新版本的 JetBrains 。如果您沒有最新版本，請更新並啟動您的開發環境。如需詳細資訊，請參閱 Amazon CodeCatalyst 的 JetBrains.

開發檔案的疑難排解問題

請參閱下列各節，以疑難排解與中 CodeCatalyst的 devfile 相關的問題。如需開發檔案的詳細資訊，請參閱設定您的開發環境。

主題

即使我在自定義 devfile 中實現了自定義映像，我的開發環境正在使用默認的通用開發文件

如果在啟動使用自訂開發檔案的開發環境時 CodeCatalyst 遇到錯誤，開發環境會預設為預設的通用開發檔案。要解決此問題，您可以在下的日誌中檢查確切的錯誤`/aws/mde/logs/devfile.log`。您還可以檢查日誌中是否成功`postStart`執行：`/aws/mde/logs/devfileCommand.log`。

我的項目沒有使用默認的通用開發文件在我的開發環境中構建

要解決此問題，請檢查您沒有使用自定義開發文件。如果您不使用自定義 devfile，請在項目的源存儲庫中查看該`devfile.yaml`文件以查找並修復任何錯誤。

我想移動一個開發環境的存儲庫開發文件

您可以將默認的 devfile 移動`/projects/devfile.yaml`到源代碼存儲庫中。要更新開發文件的位置，請使用以下命令：`/aws/mde/mde start --location` *`repository-name`*`/devfile.yaml`。

我在啟動我的開發文件時遇到問題

如果啟動 devfile 時出現問題，它將進入恢復模式，以便您仍然可以連接到您的環境並修復您的 devfile。在恢復模式下，運行`/aws/mde/mde status`不會包含您的 devfile 的位置。

```
{
    "status": "STABLE"
}
```

您可以檢查日誌中的錯誤`/aws/mde/logs`，修復 devfile，然`/aws/mde/mde start`後再次嘗試運行。

我不確定如何檢查我的開發文件狀態

您可以通過運/aws/mde/mde status行來檢查您的開發文件狀態。執行此命令之後，您可能會看到下列其中一項：

- {"status": "STABLE", "location": "devfile.yaml" }

  這表明您的開發文件是正確的。

- {"status": "STABLE" }

  這表明您的 devfile 無法啟動並且已進入恢復模式。

您可以在下的日誌中檢查確切的錯誤/aws/mde/logs/devfile.log。

您還可以檢查日誌中是否成功postStart執行：/aws/mde/logs/devfileCommand.log。

如需詳細資訊，請參閱通用開發文件圖像。

我的 devfile 與最新映像中提供的工具不相容

在您的開發環境中，devfile或devfile postStart者如果latest工具沒有特定項目所需的工具，則可能會失敗。若要修正此問題，請執行下列動作：

1. 導航到您的開發文件。
2. 在您的latest開發文件中，更新為精細的圖像版本，而不是. 它看起來可能類似於以下內容：

```
components:
  - container:
      image: public.ecr.aws/amazonlinux/universal-image:1.0
```

3. 使用更新的開發文件創建一個新的開發環境。

疑難排解問題

下列資訊可協助您疑難排解中問題的常見問題 CodeCatalyst。

主題

- 我無法為我的問題選擇受指派人

我無法為我的問題選擇受指派人

問題：建立問題時，工作負責人清單為空白。

可能的修正：工作負責人清單會直接連結至列為專案成員的 CodeCatalyst 使用者。若要確認使用者設定檔存取是否正常運作，請選擇設定檔圖示，然後選擇 [使用者設定檔]。如果未填入使用者設定檔資訊，請檢查健全狀況報告是否有任何事件。如果它確實填充，請提交服務票證。

Amazon CodeCatalyst 和AWS軟件開發套件之間的故障排除問題 AWS CLI

下列資訊可協助您疑難排解使用 CodeCatalyst 和/AWS CLI或 AWS SDK 時的常見問題。

主題

當我在命令行或終端機輸入aws codecatalyst時收到錯誤信息，說這是一個無效的選擇

問題：當我嘗試使AWS CLI用 and 時 CodeCatalyst，一個或多個命aws codecatalyst令不被識別為有效。

解決方案：造成此問題的最常見原因是您使用AWS CLI的版本不包含最新服務和命令的最新更新。請更新您的安裝，AWS CLI然後再試一次。如需詳細資訊，請參閱設定以使用AWS CLI與 CodeCatalyst。

我在執行aws codecatalyst命令時收到認證錯誤

問題：當我嘗試使用 and 時 CodeCatalyst，我收到一條消息，說明`You can configure credentials by running "aws configure".`或`Unable to locate authorization token.` AWS CLI

解法：您必須設定設定AWS CLI檔才能使用 CodeCatalyst 指令。如需詳細資訊，請參閱 設定以使用 AWS CLI與 CodeCatalyst。

CodeCatalyst 健康報告

Amazon 運作 CodeCatalyst 狀態報告是公開儀表板，可為使用者提供有關資源效能和具有廣泛影響之服務可用性的 CodeCatalyst 匯總 up-to-the-minute 通知清單。您可以查看哪些資源發生問題，可能會影響中的應用程式 CodeCatalyst。這使您可以跟踪系統範圍內的服務中斷和其他資源停機時間。當事件發生時，健全狀況報告圖示上會顯示藍色指示器。此外，還 CodeCatalyst 會自動傳送警示和電子郵件通知給專案中具有 S pace 管理員角色的所有使用者，以近乎即時的方式提供詳細資訊和事件歷史記錄。

儀表板提供所有使用中事件的清單，以及過去 30 天內先前發生 100 個事件的記錄。您可以根據事件的更新日期來組織事件清單。您也可以重新整理事件清單，以取得最新的更新。

以下是使用 CodeCatalyst 健康報告的可能工作流程：

Mateo Jackson 是具有空間管理員權限的萌芽空間開發人員。嘗試創建拉取請求時，他不斷收到錯誤消息。他檢查自己的電子郵件，發現自己收到一封自動產生的系統事件電子郵件，其中 CodeCatalyst 提供了影響其空間的系統問題的詳細歷史記錄 他選擇 [檢視更新] 並前往 CodeCatalyst 健全狀況報告，以檢視所有系統報告的事件。他從列表中選擇事件以了解更多信息。分割畫面隨即開啟，其中提供上次更新、歷程記錄、受影響功能、開始時間和事件目前狀態的時間戳記。他還可以看到問題正在進行中，但服務團隊已經開始對此進行工作。每次更新事件的歷史記錄或狀態時，他都會收到一封電子郵件。如果他無法存取自己的電子郵件，他可以選擇頂端面板中的鈴鐺圖示以取得 CodeCatalyst 健康報告。

CodeCatalyst 健康報告概念

學習下列概念可協助您瞭解 CodeCatalyst 健全狀況報告，以及這些報告如何讓您追蹤應用程式、服務和資源的健全狀況。

事件

事件是影響其中應用程式和資源的系統事件 CodeCatalyst。您可以選擇事件來檢視事件的詳細歷史記錄，包括事件開始的時間，以及服務團隊是否正在進行解決。

狀態

狀態是事件的即時狀態。它將顯示為正在進行或已解決。

受影響功能

受影響的功能是受事件影響的資源或應用程式。單一事件可能會影響系統中的多個區域，包括提取要求、問題、工作流程、測試、部署和來源。

更新於

更新日期會提供事件上次更新的時間戳記。

AWS Support對於 Amazon CodeCatalyst

建立空間時，您必須連線AWS 帳戶並將其指定為您的空間的帳單帳戶。AWS 帳戶您指定為帳單帳戶也是您訪問 Amazon AWS Support 計劃的地方 CodeCatalyst。如果您需要支援，您可以從此指定項目建立支援案例AWS 帳戶。

CodeCatalyst 空間中的使用者可以使用AWS Support適用於 Amazon CodeCatalyst 頁面 CodeCatalyst 來管理支援案例。您可以升級至企業支援或企業支援等AWS Support方案，以建立和管理中的 CodeCatalyst 技術支援案例 CodeCatalyst。Support 可通過電話，網絡或聊天獲得支持案例。

只有特定於 CodeCatalyst 服務和資源的案例才能透過 Amazon AWS Support 獲得支援 CodeCatalyst。 CodeCatalyst 資源包括中使用者部署的資源 CodeCatalyst，但這些資源不包括為其他AWS或協力廠商服務部署的資源。 CodeCatalyst 如果您需要任何其他AWS服務的支援，您必須透過 AWS Management Console.

若要變更您的支援方案，請參閱變更支援方案。

> **Note**
>
> 開發人員 Support 計劃並非針對生產環境而設計。如果太空計費帳戶具有開發人員 Support 方案，則此方案不會重疊顯示給AWS Support中的所有空間管理員和空間成員 CodeCatalyst。

Amazon 帳單 AWS Support CodeCatalyst

當您在中建立空間時 CodeCatalyst，該空間中的使用者可以從 AWS Support Amazon 建立和管理支援案例 CodeCatalyst。您可以建立兩種類型的客戶案例：

- 空間中的所有 CodeCatalyst 使用者均可使用帳戶和帳單支援案例。您可以根據您的權限取得帳單和帳戶問題的相關協助 CodeCatalyst。
- 技術支援案例可讓您與技術支援工程師聯繫，以取得與服務相關的技術問題和第三方應用程式延伸的協助。如果您擁有基本支援計劃，將無法建立技術支援案例。

  AWS 帳戶指定為該空間的帳單帳戶必須具有商業 Support 或企業 Support 方案，才能用AWS Support於 CodeCatalyst 技術案例的空間。

> **Note**
>
> 如果您的AWS Support空間 CodeCatalyst 從沒有商業支援或企業支 Support 計劃的帳戶用於 Amazon，您仍然可以將 Amazon 用AWS Support CodeCatalyst 於帳戶和帳單案例。

如需技術支援，您必須透過 CodeCatalyst 主控台開啟所有案例。您無法 CodeCatalyst 從AWS Support中建立的技術支援案例AWS Management Console。

> **Note**
>
> Amazon 不提供服務限制提高請求 CodeCatalyst。AWS Support這些要求只能由 root 使用者針對中的空間帳單帳戶提交AWS Support Center Console。

AWS SupportAmazon CodeCatalyst 的支援協議與下列考量相同：AWS Support

- AWS Support適用於支援案例的嚴重性清單、回應時間和 SLA CodeCatalyst，AWS Support詳情請參閱選擇嚴重性。
- 空間管理員和空間成員無法使用 Slack 中的 AWS Support API 或 AWS SDK 或AWS Support應用程式來建立案例 CodeCatalyst。 CodeCatalyst 支援案例只能從中提交 CodeCatalyst。

> **Note**
>
> CodeCatalyst 未完全整合AWS Trusted Advisor或AWS事件偵測與回應。驗證如何整合，以確保您的業務實務與目前的整合保持 CodeCatalyst 一致。

您必須是要求支援的空間中的使用者。

> **Note**
>
> 如果您的空間中有多個建置器，我們建議您購買商業 Support 或企業 Support 方案。這些計劃為最多 5,000 名建築商的空間提供技術支持。

AWS 帳戶指定為空間的帳單帳戶會使用AWSRoleForCodeCatalystSupport角色和AmazonCodeCatalystSupportAccess受管理的政策。這允許空間中的 CodeCatalyst AWS Support用戶訪問 Amazon CodeCatalyst 頁面。如需有關此角色和策略的更多資訊，請參閱AmazonCodeCatalystSupportAccess。有關計費的其他考量，請參閱 Amazon 管理 CodeCatalyst 員指南中的管理帳單。

以下是在以下情況下創建支持案例的構建器的可能流程 CodeCatalyst：

馬特奧·傑克遜是一個項目的 CodeCatalyst開發人員. 註冊與 AWS Support Amazon 帳單管理帳單 CodeCatalyst 並升級到商業 Support 計劃之後，AWS 帳戶該領域中的所有建設者都可以建立技術支援案例。Mateo 針對專案中失敗的工作流程提交技術支援案例。Mateo 使用適用AWS Support於 Amazon 的 CodeCatalyst 頁面填寫表單並建立案例，並在請求中提供工作流程 ID 和其他詳細資訊。案例是使用案例 ID 建立的，並包含AWS 帳戶指定為帳單帳戶的帳戶 ID，並與空間的支援方案相關聯。

雖然所有建置商都可以在中建立支援案AWS Support例 CodeCatalyst，但不會針對每個建立的案例向您收費。根據您在 Space 帳單帳戶上購買的 AWS Support Premium 方案，您可以開啟幾乎無限制的案例和聯絡人。

> **Note**
>
> 空間帳單帳戶是針對 CodeCatalyst 使用者和資源向您收費的帳戶。AWS 帳戶如果您已部署到其他服務AWS 帳戶，請AWS Support透過連絡以AWS Management Console取得部署至其他服務的資源的協助。
> 您可以從工作流程識別部署到的目標。AWS 帳戶

為 Amazon 設置您AWS Support的空間 CodeCatalyst

AWS SupportAmazon CodeCatalyst 管理支持案例，作為與 CodeCatalyst. AWS Support

AWSRoleForCodeCatalystSupport角色是用於空間中支援案例的服務角色。角色必須新增至空間的指定帳單帳戶。如需建立角色的詳細資訊，請參閱為您的帳戶和空間建立AWSRoleForCodeCatalystSupport角色。

> **Note**
>
> 對於在 2023 年 4 月 20 日之前建立的空間，您必須建立角色，才能支 CodeCatalyst 援您的空間。如果在 2023 年 4 月 20 日之後建立空間，您可以在建立空間期間、在中的 [帳單詳細資料] 頁面上建立角色 CodeCatalyst，或按一下中的支援橫幅連結來建立角色。 CodeCatalyst

若要為您的空間設定支援

1. 建立 CodeCatalyst 空間時，系統會指示您連結帳單帳戶。空間的指定帳單帳戶將由收費AWS。如需建立空間的更多資訊，請參閱註冊以創建您的第一個空間和您的發展角色。
2. 當您建立 CodeCatalyst 空間時，可使用此選項來建立允許 CodeCatalyst 使用者存取支援的`AWSRoleForCodeCatalystSupport`服務角色。角色使用受管理的策略`AmazonCodeCatalystSupportAccess`。角色必須新增至AWS 帳戶指定為空間的帳單帳戶。如需建立此角色的詳細資訊，請參閱為您的帳戶和空間建立AWSRoleForCodeCatalystSupport角色。
3. 對於空間的指定帳單帳戶，建議空間管理員購買的商業 Support 或企業 Support 方案AWS 帳戶。該領域中的所有成員都可以管理 Amazon 的支援案例 CodeCatalyst，並且支援管道將與您在整合完成時購買的AWS Support方案保持一致。AWS Support
4. 若要在中建立和管理支援案例 CodeCatalyst，請參閱在中建立 CodeCatalyst 支援案例 CodeCatalyst。

存取 CodeCatalyst 中的支援 AWS Management Console

如果空間已啟用支援的帳單帳戶中斷連線，Amazon 將不再顯示與先前 Space 帳單帳戶和相關支援方AWS Support案相關聯的案AWS Support例 CodeCatalyst。該帳單帳戶的 root 使用者可以從中檢視和解決舊案例，也可以為AWS Support其他使用者設定 IAM 許可，以便其他使用者檢視和解決舊案例。AWS Management Console您仍然可以參與所有其他支援計劃的好處，AWS 服務並完成先前未解決的任何 CodeCatalyst 支援案例。AWS Management Console

如需詳細資訊，請參閱AWS Support使用指南中的更新、解決和重新開啟案例。

有關一般操作方法資訊的 Support 案例也 CodeCatalyst 可以在中開啟AWS Management Console，但無法透過此通道接收的技術支援 CodeCatalyst。如需詳細資訊，請參閱AWS Support使用指南中的建立支援案例和案例管理。

以下是解決支援案例的使用者的 CodeCatalyst 可能流程AWS Management Console：

雖然所有建置商都可以透AWS Support過 Amazon 建立支援案例 CodeCatalyst，但支援請求會從指定為該空間帳單帳戶的帳戶計費。Mateo Jackson 是專案的開發人員，在專案中 CodeCatalyst 為失敗的工作流程開發技術支援案例。但是，已向 Amazon 註冊 CodeCatalyst 並購買商業 Support 方案AWS Support的空間帳單帳戶已與空間中斷連線。Mateo 檢視最新通訊並解決開立案例的唯一方法 CodeCatalyst 是從AWS Support中心管理案例 ID。AWS Management Console為此，Mateo 會獲得先前太空帳單帳戶根使用者的 IAM 許可，並透過AWS Support主控台解決此案例。

> **Important**
>
> 如果您變更空間的指定帳單帳戶，您的AWS Support方案仍然可以在月底透過AWS Management Console唯一存取。您必須重新購買更新AWS Support的帳單帳戶，才能繼續存取先前在中建立的支援案例 CodeCatalyst。我們建議您等待解決所有支援案例，以變更太空計費帳戶，以避免對透過 AWS Support Amazon 存取支援案例造成任何影響 CodeCatalyst。

在中建立 CodeCatalyst 支援案例 CodeCatalyst

您可以在 Amazon CodeCatalyst 頁面中創建一個支持案例。AWS Support

AWSBuilder ID 只能取得對其進行驗證的別名的支援，而且只能根據其權限取得資源的支援。帳戶和帳單選項適用於所有空間管理員和空間成員。不過，使用者只能取得他們有權存取的資源，而不能取得 CodeCatalyst 與管理帳戶帳單有關的資源支援。

您可以使用空間的 [用於 CodeCatalyst] 頁面，為您的 CodeCatalyst 資源建立帳戶和帳單案例或技術支援案例。AWS Support

> **Note**
>
> 只有特定於 CodeCatalyst 服務和資源的案例才能透過 Amazon AWS Support 獲得支援 CodeCatalyst。 CodeCatalyst 資源包括中使用者部署的資源 CodeCatalyst，但這些資源不包括為其他AWS或協力廠商服務部署的資源。 CodeCatalyst 如果您需要任何其他AWS服務的支援，您必須透過AWS Management Console.

若要在中建立支援案例 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 在頁面頂端，選擇？圖示，然後選擇 [Sup port]。
4. 選擇 Create case (建立案例)。
5. 請選擇下列其中一個選項：

   - 帳戶和帳單
   - 技術

   > **Note**
   >
   > 在 AWS Support Amazon 中 CodeCatalyst，如果將商業 Support 或企業 Support 方案新增至太空計費帳戶，則所有 Space 管理員和太空會員都可以使用 CodeCatalyst 技術案例支援。如需故障診斷資訊，請參閱我無法為我的空間建立技術支援案例。
   > AWS Support平面不跨越空間。如果您是多個空間的成員，您的空間管理員將需要為每個空間購買 AWS Support Premium 方案，才能獲得所有空間的技術支援。

6. 選擇 Service (服務)、Category (類別) 和 Severity (嚴重性)。如需有關選擇嚴重性的資訊，請參閱選擇嚴重性。

   - 一般指導方針
   - 系統效能不佳
   - 生產系統效能不佳
   - 生產系統當機
   - 商業關鍵系統當機

7. 選擇 Next step: Additional information (下一步：其他資訊)。
8. 在 Additional information (其他資訊) 頁面上的 Subject (主旨)，輸入與您問題相關的標題。
9. 請在 Description (描述) 欄位中，依照提示來描述您的案例，如下所示：

   - 疑難排解特定的資訊 CodeCatalyst，例如工作流程 ID、記錄或螢幕擷取畫面
   - 您收到的錯誤訊息
   - 您依照哪些疑難排解步驟操作

> **Note**
>
> 請勿在情況下分享任何敏感信息，例如憑據，信用卡，簽名的 URL 或個人身份信息。

10. (選用) 選擇 Attach files (附加檔案) 來將任何相關文件新增至您的案例，例如錯誤日誌或螢幕擷取畫面。您最多可以連接三個檔案。每個檔案最多可達 5 MB。
11. 在空間名稱中，會顯示空間的名稱。
12. 在 [建置器名稱] 中，會自動填入與您的AWS建置器 ID 相關聯的全名。
13. （可選）在「項目名稱」中選擇項目（如果適用）。

> **Note**
>
> 您只會看到您有權限的專案。如果您需要存取其他專案，請先要求專案管理員為您提供存取權，然後再建立支援案例。

14. 選擇下一步：聯絡我們。
15. 在偏好的聯絡人語言中，選擇預設值。目前僅提供英語服務。
16. 選擇聯絡方式的 [網路]、[電話] 或 [聊天] 選項。
17. 檢閱您的案例詳細資料，然後選擇 [提交]。您的案例 ID 編號和摘要隨即出現。

    支援案例是在空間層級建立，並且可由具有存取權限 (如果已選取) 的所有成員檢視，這些成員在您的支援案例中定義。目前無法省略個別使用者的支援案例。

解決支援案例 CodeCatalyst

您可以從AWS Support適用於 Amazon 的 CodeCatalyst 頁面解決未解決的支援案例。

您必須在要解決支援案例的空間中具有 Space 管理員或 Space 成員角色。如果您沒有 S pace 管理員角色，或者在建立案例時選取了專案，您也需要擁有專案的成員資格才能檢視和解決案例。

若要解決開啟的支援案例 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 在頁面頂端，選擇? 圖標，然後選擇 AWS SupportAmazon CodeCatalyst。
4. 選擇您要管理的支援案例連結。選擇 Resolve case (解決案例)。

重新開啟支援案例 CodeCatalyst

您可以使用從 Amazon CodeCatalyst 頁面重新打開已解決的AWS Support支持案例。

> **Note**
>
> 您最多可以在問題解決後 14 天內重新開啟支援案例。不過，您無法重新開啟已結案超過 14 天的案例。如果您無法重新開啟個案，請開啟新案例，並提供先前的案例 ID 作為參考。
> 如果您重新開啟一個現有案例，但資訊與目前的問題不同，支援客服人員可能會要求您建立一個新案例。

若要重新開啟支援案例 CodeCatalyst

1. 請在以下位置開啟 CodeCatalyst 主控台。 https://codecatalyst.aws/
2. 導覽至您的 CodeCatalyst 空間。

> **Tip**
>
> 如果您屬於多個空間，請在上方導覽列中選擇一個空格。

3. 在頁面頂端，選擇? 圖示，然後選擇適用於的 AWS Support CodeCatalyst。
4. 選擇您要管理的支援案例連結。選擇 Reopen (重新開啟)。在確認畫面上選擇「確定」，然後選擇「送出」。
5. 在描述中填寫有關同一問題的最新信息。請勿在情況下分享任何敏感信息，例如憑據，信用卡，簽名的 URL 或個人身份信息。

的配額 CodeCatalyst

下表說明 Amazon 的配額和限制 CodeCatalyst。您可以 CodeCatalyst 在下列主題中找到的特定方面的其他資訊：

- 來源儲存庫的配額 CodeCatalyst
- 中的身分識別、權限和存取配額 CodeCatalyst
- 中工作流程的配額 CodeCatalyst
- 開發環境的配額 CodeCatalyst
- 以下項目的配額 CodeCatalyst
- 中藍圖的配額 CodeCatalyst
- 中空格的配額 CodeCatalyst
- 中的問題配額 CodeCatalyst

帳戶中的最大空格數	5
使用者在行事曆月份可建立的最大空格數	5
空間的AWS 帳戶最小數目	1
空間的帳戶連線數目上限	5,000
空間的虛擬私人雲端連線數目上限	100
空間中的專案數目上限	100
使用者可以屬於的專案數目上限	1,000
空間描述	空間描述是可選的。如果指定，它們的長度必須介於 0 到 200 個字元之間。它們可以包含字母、數字、空格、句點、底線、逗號、破折號和下列特殊字元的任意組合： `? & $ % + = / \ ; : \n \t \r`
空間名稱	空間名稱在中必須是唯一的 CodeCatalyst。您無法重複使用已刪除空格的名稱。

空格名稱的長度必須介於 3 到 63 個字元之間。它們還必須以字母數字字符開頭。空格名稱可以包含字母、數字、句號、底線和破折號的任意組合。它們不能包含下列任何字元：

```
! ? @ # $ % ^ & * ( ) + = { } [ ]
| 八 > < ~ ` ' " ; :
```

Amazon 的文檔歷史 CodeCatalyst

下表說明的整體文件的文件歷程記錄和更新 CodeCatalyst。

變更	描述	日期
新內容：使用任務將問題分解為更小的目標	已新增內容，以支援啟動問題中的工作。您可以將任務添加到問題中，以進一步分解，組織和跟踪該問題的工作。	2024年4月4日
更新內容：中 GitHub 動作的限制 CodeCatalyst	已更新主中 GitHub 動作的限制 CodeCatalyst題，指出 GitHub 動作會在較舊的執行階段環境 Docker 映像上執行。	2024年4月2日
更新內容：使用人工因素	更新主使用人工因素題以包含兩個新的子主題：我可以共享工件而不將它們指定為輸出和輸入嗎？和 可以在工作流程之間共用成品嗎？	2024年4月2日
新內容：「AWS CDK 部署」動作參考	已將新`CloudAssemblyRootPath` 屬性新增至「AWS CDK 部署」動作參考.	2024年4月1日
更新內容：使用執行階段環境 Docker 影像	已更新主使用執行階段環境 Docker 影像題，以包含有關新 2024 年 3 月執行階段環境影像的資訊。	2024年3月26日
更新內容：使用角色	將角色權限資訊合併到單一資料表中。表格位於新每個角色的可用權限主題中。	2024年3月18日
新內容：檢視使用者的所有空間和專案	已新增在使用者首頁上檢視清單的相關資訊，以顯示登入使	2024年3月18日

	用者的每個 CodeCatalyst 空間或專案。 CodeCatalyst請參閱檢視使用者的所有空間和專案。	
新內容：範例：具有拉動和分支的觸發器	新增了拉取要求觸發程序的範例。在整個使用觸發程序主題中做了一些小的更正。	2024年3月11日
更新內容：使用角色	已更新角色的文件，以包含建立、刪除和檢視環境的權限。	2024年3月4日
更新內容：教學課程：使用生成式 AI 功能	更新教學以反映建立問題並將問題指派給 Amazon Q 時所做的變更。	2024年3月4日
新內容：當工作流程失敗時阻止提 GitHub 取請求合併	已新增有關如何在工作流程失敗時封鎖與分支保護規則的提 GitHub 取要求合併的新內容。	2024年3月4日
新內容：問題元件	已新增有關如何以自訂藍圖開發人員身分處理問題元件的新內容。	2024年2月27日
更新內容：動作類型	已更新CodeCatalyst 動作主題以包含 CodeCatalyst 實驗室動作清單。	2024年2月21 日
更新內容：使用提取請求	已更新文件，以反映新功能，其中包含核准規則和合併提取要求的覆寫要求。	2024年2月15日
更新內容：合併提取請求	已新增提取請求的文件，以包含覆寫合併需求的相關資訊，以合併尚未收到必要審核者核准或符合核准規則的提取請求。	2024年2月15日

新內容：管理核准規則	已新增提取要求的文件，以包含建立和管理核准規則的相關資訊。	2024年2月15日
更新內容：使用角色	已更新角色的文件，以包含處理核准規則和提取要求的權限。	2024年2月14日
更新內容：如何修復「工作流定義有 *n* 個錯誤」錯誤？	已更新此如何修復「工作流定義有 *n* 個錯誤」錯誤？區段以包含更多疑難排解秘訣。	2024年2月9 日
新內容：檢視工作流程狀態	添加了描述工作流程狀態的部分。	2024年2月9 日
更新內容：中工作流程的配額 CodeCatalyst	更新了每個 AWS 帳戶 工作流程的最大動作數和與每個空間配額相關聯的環境數目上限的中工作流程的配額 CodeCatalyst主題。	2024年2月7日
更新內容：建立環境	已更新建立環境區段，指出每個環境最多可以使用一個帳戶連線。	2024 年 1 月 31 日
新內容：自訂藍圖儲 GitHub 存庫	已新增可公開使用之 GitHub 儲存庫的新內容。	2024 年 1 月 10 日
更新內容：使用 npm 配置 CodeCatalyst	更新了使用 npm 的一般配置說明 CodeCatalyst，並增加了`always-auth=true` 選項的清晰度。	2024年1月5日
更新內容：使用提取請求	已更新文件，以反映中生成 AI 功能的新功能 CodeCatalyst。	2023 年 11 月 28 日
更新內容：建立問題	已更新文件，以反映中生成 AI 功能的新功能 CodeCatalyst。	2023 年 11 月 28 日

新內容：教學課程：使用生成式 AI 功能	已新增在 Amazon 中使用生成 AI 功能的教學課程 CodeCatalyst。	2023 年 11 月 28 日
新內容：自訂藍圖和生命週期管理	已新增 Amazon 中使用自訂藍圖和生命週期管理功能的新內容 CodeCatalyst。	2023 年 11 月 27 日
更新內容：教學課程：使用現代三層 Web 應用程式藍圖建立專案	使用修正和疑難排解資訊更新了教學課程。	2023 年 11 月 22 日
更新內容：使用觸發程序	修正了一些與拉取要求觸發程序相關的範例和說明。已新增分支時的觸發考量區段。	2023 年 11 月 22 日
新內容：使用 SSO 登入	已新增有關使用單一登入 (SSO) 登入的資訊，以及設定和管理支援身分同盟之 CodeCatalyst 空間的相關資訊連結。請參閱 正在設定 CodeCatalyst 和 使用 SSO 登入。	2023 年 11 月 17 日
更新內容：使用角色	已更新角色的文件，以包含使用團隊、VPC 連線、單一登入和機器資源的權限。	2023 年 11 月 16 日
更新內容：使用提取請求	已更新文件，以反映如何顯示提取要求變更的變更。	2023 年 11 月 16 日
更新內容：配額 CodeCatalyst	更新了空間配額中 VPC 連線數目上限的的配額 CodeCatalyst主題。	2023 年 11 月 16 日
新內容：透過 VPC 連線使用開發環境	已新增透過 VPC 連線使用開發環境的文件。	2023 年 11 月 16 日

新內容：管理空間和 CodeCatalyst 專案的團隊	已新增有關使用團隊搭配空間的資訊。請參閱 管理團隊 和管理專案團隊。	2023 年 11 月 16 日
新內容：管理空間中藍圖和工作流程的機器資源	已新增有關搭配空格使用機器資源的資訊。請參閱管理機器資源。	2023 年 11 月 16 日
新內容：管理專案中藍圖和工作流程的機器資源 CodeCatalyst	已新增有關在專案中使用機器資源的 CodeCatalyst 資訊。請參閱管理機器資源。	2023 年 11 月 16 日
新內容：建立 VPC 連線與環境的關聯	已新增關聯 VPC 連線與環境 (可在工作流程中使用) 的文件。	2023 年 11 月 16 日
新內容	Amazon CodeCatalyst 管理員指南的初始發布。	2023 年 11 月 16 日
新內容：「AWS CDK 部署」動作參考	已將新`CdkCliVersion` 屬性新增至「AWS CDK 部署」動作參考和「AWS CDK引導」動作參考。	2023 年 11 月 14 日
更新內容：使用角色	已更新角色的文件，以包含使用分支規則的權限。	2023 年 11 月 13 日
更新內容：疑難排解來源儲存庫、工作流程和開發環境的問題	更新疑難排解主題，以包含使用分支規則的相關資訊。	2023 年 11 月 13 日
更新內容：建立和測試動作參考	更新了`Environment` 屬性的文檔。現在，它是用於構建和測試操作的可選字段。	2023 年 11 月 13 日
新內容：管理分支規則	已新增分支文件，以包括檢視來源儲存庫中分支的任何規則，以及建立和管理分支規則的相關資訊。	2023 年 11 月 13 日

更新內容：使用提取請求	已更新文件，以反映拉取要求相關資訊顯示方式的變更。	2023 年 11 月 10 日
更新內容：使用檔案快取	更新了文檔以包含文件緩存限制。	2023 年 11 月 10 日
更新內容：教學課程：將應用程式部署到 Amazon EKS	更新文件以提及 EKS 應用程式部署藍圖。	2023 年 11 月 9 日
新內容：套件 CodeCatalyst	增加了在中使用軟件包的文檔 CodeCatalyst。	2023 年 11 月 1 日
新內容和更新內容：使用角色	更新以下四個新角色的文件 CodeCatalyst：進階使用者、受限存取權、審核者和唯讀。	2023 年 11 月 1 日
更新內容：使用 GitHub 動作輸出參數	已更新範例以使用`GITHUB_OUTPUT` 環境檔案而非指`set-output` 令。建議使用環境檔案設定輸出參數 GitHub的方法。	2023 年 10 月 24 日
新內容：使用觸發程序	增加了對計劃觸發器的文檔。	2023 年 10 月 16 日
更新內容：「部署至 Kubernetes 叢集」動作參考	已新增有關使用`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 角色至「部署至 Kubernetes 叢集」動作參考和主教學課程：將應用程式部署到 Amazon EKS題的資訊。	2023 年 9 月 22 日

更新內容：角色的CodeCatalystWorkflowDevelopmentRole-*spaceName* 新角色名稱和策略	更新開發人員角色名稱變更為的步驟和角色描述CodeCatalystWorkflowDevelopmentRole-*spaceName* 。開發人員角色現在會使用受AdministratorAccess AWS 管理的政策。請參閱 了解服CodeCatalystWorkflowDevelopmentRole-*spaceName* 務角色 和 註冊以創建您的第一個空間和您的發展角色。	2023 年 9 月 20 日
更新內容：使用變數	引入了兩個新概念：用戶定義變量和預定義的變量。這些概念應該使使用變數章節更易於閱讀和理解。	2023 年 9 月 19 日
更新內容：使用提交	更新文件以反映顯示資訊中的變更，並提供有關檢視多個父項提交的詳細資訊。	2023 年 9 月 7 日
新內容：使用觸發程序	已將下列範例新增至使用觸發程序主題：示例：一個簡單的「推送到主」觸發器	2023 年 9 月 6 日
更新內容：使用提取請求	已更新文件，以反映建立提取要求時來源分支和目的地分支的顯示順序變更。	2023 年 8 月 30 日
新內容：查看和更改默認分支	已新增分支文件，以包含檢視和變更來源儲存庫預設分支的相關資訊。	2023 年 8 月 30 日
更新內容：「部署至 Kubernetes 叢集」動作參考	已在中的屬性描述中新增有關「頭盔」和「使用Manifests設定」的備註。「部署至 Kubernetes 叢集」動作參考	2023 年 8 月 15 日

新內容：管理問題附件	已新增處理及管理問題附件的文件。	2023 年 8 月 15 日
更新內容：使用觸發程序	改善和擴充與工作流程觸發器相關的文件。	2023 年 8 月 11 日
新內容：疑難排解角色權限	新增有關更新角色許可以執行需要存取 Amazon 的工作流程的資訊 CodeGuru。請參閱現代三層 Web 應用程式藍圖工作流程OnPullRequest失敗，並顯示 Amazon 的許可錯誤 CodeGuru。	2023 年 8 月 11 日
新內容：如何修正「工作流程處於非作用中狀態」訊息？	已新增下列疑難排解主題：如何修正「工作流程處於非作用中狀態」訊息？	2023 年 8 月 11 日
新內容：新增「部署至 Kubernetes 叢集」動作	已新增部署至 Kubernetes 叢集動作的文件。如需詳細資訊，請參閱 新增「部署至 Kubernetes 叢集」動作 及 教學課程：將應用程式部署到 Amazon EKS。	2023 年 7 月 27 日
CodeCatalyst 空間的管理事件記錄方式的更新	已新增有關如何針對 CodeCatalyst空間中的特定動作記錄管理事件的資訊 AWS CloudTrail。已新增有關如何使用 list-event-logs 指令檢視空間中所有事件的資訊。請參閱在 Amazon 監控 CodeCatalyst。	2023 年 7 月 20 日
更新內容：使用觸發程序	已更新文件，指出來 GitHub 源儲存庫現在支援提取要求觸發程序。以前，只有 CodeCatalyst 來源儲存庫才支援提取要求觸發程序。	2023 年 7 月 14 日

更新內容：工作流定義參考	修正程式Compute碼區塊中的格式錯誤。	2023 年 6 月 27 日
更新內容：中工作流程的配額 CodeCatalyst	更新了動作可以執行配額的時間上限的中工作流程的配額 CodeCatalyst主題。	2023 年 6 月 27 日
更新內容：資料保護	已更新文件以包含有關資料複製的其他資訊。	2023 年 6 月 26 日
新內容：使用動作版本	已新增使用動作版本主題。	2023 年 6 月 21 日
更新內容：使用環境	澄清了部哪些動作支援環境？分。	2023 年 6 月 14 日
更新內容：重組問題的文檔	重組了大部分的問題文件，以更好地與整體文件集和使用者流程保持一致。	2023 年 5 月 31 日
更新內容：問題查看切換	更新了各種用戶流程，以配合更新的問題視圖切換器。	2023 年 5 月 31 日
更新內容：管理通知	已更新通知的文件，以包含設定個人 Slack 通知的相關資訊。	2023 年 5 月 30 日
更新內容：管理通知	已更新通知的文件，以包含設定個人 Slack 通知的相關資訊。	2023 年 5 月 30 日
新內容： CodeCatalyst 信任模型	已新增包含信任模型相關資訊的新主題，該主題 CodeCatalyst 允許在連線中承擔服務角色 AWS 帳戶。已新增有關的已定義服務主體的新區段 CodeCatalyst。請參閱CodeCatalyst 信任模型。	2023年5月20日

更新內容：使用來源	簡化了中的指示參考來源儲存庫中的檔案。	2023 年 5 月 10 日
更新內容：中工作流程的配額 CodeCatalyst	已使用輸出變數值配額的最大長度更新中工作流程的配額 CodeCatalyst主題。	2023 年 5 月 10 日
新內容：使用人工因素	增加了兩個例子：範例：參照單一人工因素中的檔案和範例：存在人工因素時參照檔案 WorkflowSource 。	2023 年 5 月 10 日
更新內容：使用提取請求	已更新提取要求的文件，以包含設定提取要求事件之電子郵件偏好設定的相關資訊。	2023 年 4 月 21 日
更新內容：管理通知	已更新通知的文件，以包含設定提取要求事件之電子郵件偏好設定的相關資訊。	2023 年 4 月 21 日
受管理策略的更新	新增AWS 受管政策：AmazonCodeCatalystFullAccessAWS 受管政策：AmazonCodeCatalystReadOnlyAccess、和AWS 受管政策：AmazonCodeCatalystSupportAccess受管理的政策。請參閱CodeCatalyst AWS受管理策略的更新。	2023 年 4 月 20 日
新內容：移除部署目標	已新增移除部署目標主題。	2023 年 4 月 20 日
新內容：動作類型	已新增CodeCatalyst 實驗室動作主題。	2023 年 4 月 20 日

管理空間中具有 Space 管理員角色之使用者的更新	已新增有關移除或變更在空間中具有 Space 管理員角色之使用者角色的資訊。請參閱移除或變更具有 S pace 管理員角色之使用者的角色。	2023 年 4 月 19 日
用於管理開發環境的更新	已新增有關以 Space 管理員身分管理開發環境的資訊。請參閱管理空間的開發環境。	2023 年 4 月 19 日
更新內容：尋找和檢視問題	重新組織了主尋找及檢視問題題和子主題。	2023 年 4 月 19 日
更新內容：使用計算	增加了對 ARM64 架構的支持 Amazon Linux 2.	2023 年 4 月 19 日
新內容：在群組內移動問題	已新增關於「主機板」和「所有問題」檢視中移動群組內問題的文件。	2023 年 4 月 19 日
更新內容：中工作流程的配額 CodeCatalyst	已更新缺少配額的中工作流程的配額 CodeCatalyst主題，並將單一動作輸出變數配額的總大小上限更新為 120 KB (從 2 KB)。	2023 年 4 月 18 日
新內容：檢視動作的原始程式碼	已新增檢視動作的原始程式碼主題。	2023 年 4 月 18 日
新內容：重試測試用例	已新增重試測試用例主題。	2023 年 4 月 11 日
新內容：停止工作流程執行	已新增停止工作流程執行主題。	2023 年 4 月 10 日
新內容：添加了標記 AWS 和 Amazon 之間帳戶連接資源的部分 CodeCatalyst	新增標記帳戶連線資源和管理連線資源的 IAM 政策的資訊。請參閱 使用標記控制對帳號連線資源的存取 和 CodeCatalyst 權限參考。	2023 年 4 月 6 日

新內容：動作類型	已新增動作類型主題。	2023 年 4 月 6 日
更新內容：「部署AWS CloudFormation堆疊」動作參考	更新了`parameter-overrides` 屬性描述。它現在支持 JSON 文件。	2023 年 4 月 5 日
新內容： CodeCatalyst 使用連結的 GitHub 儲存庫在中建立專案	已新增在 Amazon 創建一個項目 CodeCatalyst標題的新章節，其中建立具有連結 GitHub 儲存庫的專案包含建立連結至 GitHub 儲存庫的專案的指示。	2023 年 4 月 5 日
更新內容：使用通知	已更新通知的文件，以包含設定專案事件相關電子郵件的相關資訊。	2023 年 3 月 31 日
新內容	Amazon CodeCatalyst 行動開發工具包指南的初始發布。	2023 年 3 月 31 日
更新內容：在 Amazon 中重組了空間部分 CodeCatalyst	透過移除著陸頁面和合併主題來更新「空間」區段。	2023 年 3 月 29 日
更新內容：教學課程：將應用程式部署到 Amazon ECS	已更步驟 1：設定 AWS 使用者和 AWS CloudShell改為描述如何在中創建用戶 AWS IAM Identity Center 而不是 AWS Identity and Access Management。不再建議建立 IAM 使用者。	2023 年 3 月 23 日
更新內容：使用角色	已更新 Space 管理員、專案管理員和參與者角色的文件，以包含將問題連結至提取要求的權限。	2023 年 3 月 13 日
更新內容：使用提取請求	已更新提取要求的文件，以包含連結問題至提取要求的相關資訊。	2023 年 3 月 13 日

更新內容：處理問題	已更新問題的文件，以包含連結問題至提取要求的相關資訊。	2023 年 3 月 13 日
新內容：檢視專案中所有執行的狀態和詳細資訊	已新增說明新彙總工作流程執行頁面的區段。	2023 年 3 月 8 日
新內容：如何修復「工作流定義有 *n* 個錯誤」錯誤？	已新增關於如何疑難排解「工作流程定義有錯誤」錯誤的章節。	2023 年 3 月 7 日
更新內容：建立、編輯和刪除工作流程	更新指示以反映新 UI。	2023 年 3 月 3 日
新內容：使用 universal-test-runner	已新增使用 universal-test-runner主題。	2023 年 3 月 3 日
更新內容：使用中的工作流程來建置、測試和部署 CodeCatalyst	已更新各個區段，以反映「工作流程摘要」頁面上的新來源儲存庫、分支和工作流程名稱篩選器。	2023 年 3 月 2 日
新內容：透過提交檢視程式碼品質和部署狀態 CodeCatalyst	已新增關於透過提交檢視程式碼品質和部署狀態的區段。	2023 年 2 月 27 日
新內容：「BranchName" 和 "CommitId" 變數	添加了一個新的`BranchName` 預定義變量。	2023 年 2 月 16 日
更新內容：管理 Amazon 中的太空會員 CodeCatalyst	根據使用者中指派的角色，更新了有關變更成員角色、邀請成員以及移除兩個新表格中成員的資訊 CodeCatalyst。	2023 年 2 月 15 日
更新內容：在 Amazon CodeCatalyst 主控台新增 PAT 管理步驟	已新增在主控台中檢視、建立和刪除 PAT 的步驟。	2023 年 2 月 15 日
更新內容：使用執行階段環境 Docker 影像	在「預設影像工具版本」表格中新增了更多工具。	2023 年 1 月 10 日

更新內容：使用人工因素	修正了一個神器路徑。	2023 年 1 月 3 日
更新內容：使用來源	修正了源路徑。	2023 年 1 月 3 日
更新內容：「動GitHub 作」動作參考	修正了部分中的代碼片`Steps`段。	2023 年 1 月 3 日
更新內容：更新提取請求	已更新文件，以包含更新提取要求之必要或選用檢閱者的相關資訊。	2022 年 12 月 23 日
新內容：使用檔案快取	已新增工作流程中檔案快取的頁面。	2022 年 12 月 20 日
更新內容：使用提取請求	已更新提取要求的文件，以包含通知的相關資訊。	2022 年 12 月 16 日
新內容：「AWS CDK 部署」動作參考	添加了一個新的`CdkRootPath` 屬性。	2022 年 12 月 16 日
新內容：跨動作共用運算	已新增跨動作共用運算主題。	2022 年 12 月 14 日
更新內容：使用人工因素	已修正示範如何指定輸入成品的範例。	2022 年 12 月 13 日
新內容：「動GitHub 作」動作參考	已新增 GitHub 「動作」動作的專用參考頁面。	2022 年 12 月 13 日
更新內容：中專案的配額 CodeCatalyst	更新了一個空間中最多 100 個專案的文件。	2022 年 12 月 2 日
新內容	Amazon CodeCatalyst 用戶指南的初始發布。	2022 年 12 月 1 日

AWS 詞彙表

有關最新 AWS 術語，請參閱AWS 詞彙表 參考文獻中的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。