aws

使用者指南

AWS Proton

AWS Proton: 使用者指南

Table of Contents

什麼是 AWS Proton？

AWS Proton是：

- 自動化基礎架構即程式碼佈建和部署無伺服器和容器型應用程式

  所以此AWS Proton服務是一個雙管齊下的自動化框架。身為管理員，您可以建立版本化服務模板為無伺服器和容器型應用程式定義標準化基礎架構和部署工具。作為應用程序開發人員，您可以從可用的中進行選擇服務模板自動化您的應用程式或服務部署。

  AWS Proton識別所有現有的服務執行個體正在為您使用過時的模板版本。身為管理員，您可以請求AWS Proton一鍵升級它們。

- 標準基礎設施

  平台團隊可以使用AWS Proton和版本化的基礎設施作為代碼模板。他們可以使用這些範本來定義和管理包含架構、基礎結構資源和 CI/CD 軟體部署管線的標準應用程式堆疊。

- 透過 CI/CD 整合的部署

  當開發人員使用AWS Proton自助服務介面來選取服務範本，他們為程式碼部署選取標準化的應用程式堆疊定義。AWS Proton自動佈建資源、設定 CI/CD 管線，並將程式碼部署到定義的基礎結構中。

AWS Proton適用於平台團隊

身為管理員，您或平台團隊的成員，建立環境樣板和服務模板包含基礎設施作為代碼。所以此環境範本定義多個應用程式或資源使用的共用基礎結構。所以此服務範本定義部署和維護單一應用程式或微服務所需的基礎結構類型環境。同時AWS Proton 服務是一個實例化服務範本，其中通常包括幾個服務執行個體和一個管道。同時AWS Proton 服務執行個體是一個實例化服務範本在特定環境。您或團隊中的其他人可以指定環境樣板與給定的兼容服務範本。如需有關 的詳細資訊模板，請參閱AWS Proton 範本。

您可以使用以下基礎設施做為程式碼供應商AWS Proton：

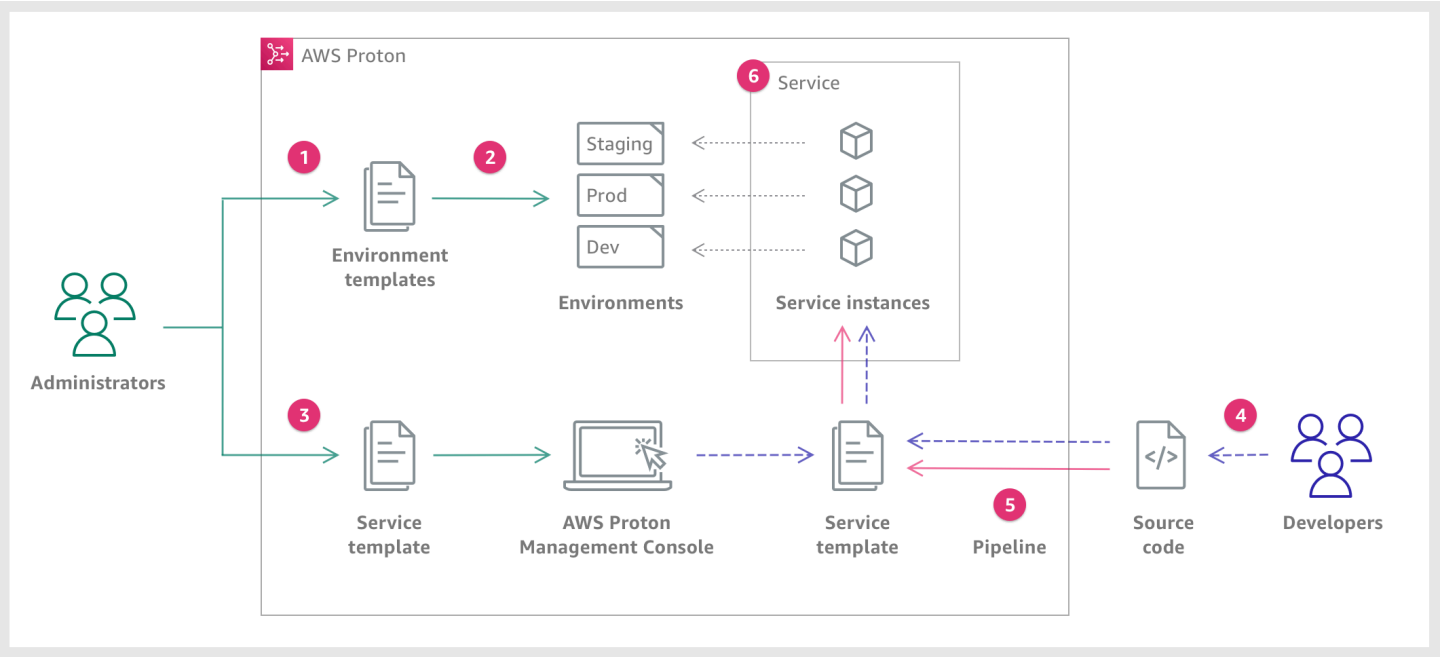
- AWS CloudFormation
- 地形

AWS Proton開發人員

身為應用程式開發人員，您可以選取標準化服務範本那個AWS Proton使用來建立服務部署和管理您的應用程式服務執行個體。同時AWS Proton 服務是一個實例化服務範本，其中通常包括幾個服務執行個體和一個管道。

AWS Proton工作流程

下圖是主要的可視化AWS Proton前一段所討論的概念。它還顯示什麼構成簡單的高層級概觀AWS Proton工作流程。



1 作為管理員，您建立並註冊環境範本取代為AWS Proton，它定義了共享資源。

2 AWS Proton部署一或多個環境，基於環境範本。

3 作為管理員，您建立並註冊服務範本取代為AWS Proton，其中定義了相關的基礎架構、監控和 CI/CD 資源，以及相容性環境範本。

4

作為開發人員，您選擇一個註冊服務範本並提供一個鏈接到您的來源碼儲存庫。

5

AWS Proton規定Service (服務)用一個CI/CD 管道為您的服務執行個體。

6

AWS Proton規定和管理Service (服務)與服務執行個體正在運行來源碼如已在選取的定義服務範本。一個服務執行個體是所選項目的實例化服務範本在一個環境對於一個單一階段管道（例如 Prod）。

設定

完成本節中的工作，以便您可以建立和註冊服務與環境範本。您需要這些來部署環境和服務 AWS Proton。

> **Note**
>
> 我們提供 AWS Proton 不收取額外費用。您可以免費建立、註冊和維護服務和環境範本。您也可以依靠自 AWS Proton 行管理自己的作業，例如儲存、安全性和部署。您在使用時產生的唯一費 AWS Proton 用如下。
>
> - 部署和使用您指示為您部署和維護的 AWS 雲端 AWS Proton 資源的成本。
> - 維護與程式碼儲存庫 AWS CodeStar 連線的成本。
> - 如果您使用儲存貯體提供輸入，則維護 Amazon S3 儲存貯體的成本 AWS Proton。如果您切換the section called “範本同步設定”使用 Git 儲存庫，您可以避免這些成本the section called “模板捆綁”。

主題

使用 IAM 進行設定

當您註冊時 AWS，系統會自動註冊您的 AWS 帳戶 所有服務 AWS，包括 AWS Proton。只需為您使用的服務和資源付費。

> **Note**
>
> 您和您的團隊，包括管理員和開發人員，都必須在同一個帳戶下。

註冊成為 AWS

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊時 AWS 帳戶，會建立AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

建立 IAM 使用者

若要建立管理員使用者，請選擇下列其中一個選項。

選擇一種管理管理員的方式	到	By	您也可以
在 IAM Identity Center (建議)	使用短期憑證存取 AWS。 這與安全性最佳實務一致。有關最佳實務的資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。	請遵循 AWS IAM Identity Center 使用者指南的入門中的說明。	AWS IAM Identity Center 在《使用AWS Command Line Interface 者指南》中設定 AWS CLI 要使用的，以設定程式設計方式存取。
在 IAM 中 (不建議使用)	使用長期憑證存取 AWS。	請遵循 IAM 使用者指南中建立您的第一個 IAM 管理員使用者和使用者群組的說明。	請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰，設定程式設計存取。

設定 AWS Proton 服務角色

您可能需要為 AWS Proton 解決方案的不同部分建立一些 IAM 角色。您可以使用 IAM 主控台預先建立它們，也可以使用 AWS Proton 主控台為您建立它們。

建立 AWS Proton 環境角色 AWS 服務， AWS Proton 以允許代表您對其他類似 AWS CloudFormation 以及各種計算和儲存服務進行 API 呼叫，以便為您佈建資源。 AWS CodeBuild當環境或其中執行的任何服務執行個體使用 AWS-AWS managed 佈建時，就需要-managed 佈建角色。當環境或其任何服務執行個體使用CodeBuild佈建時，就需要CodeBuild角色。若要進一步瞭解 AWS Proton 環境角色，請參閱the section called “IAM 角色”。建立環境時，您可以使用 AWS Proton 主控台為這兩個角色中的任何一個選擇現有角色，或為您建立具有管理權限的角色。

同樣地，建立 AWS Proton 管線角色， AWS Proton 以允許代表您對其他服務進行 API 呼叫，以便為您佈建 CI/CD 管道。若要深入瞭解 AWS Proton 管線角色，請參閱the section called “管線服務角色”。如需設定 CI/CD 設定的詳細資訊，請參閱。the section called “設定帳戶 CI/CD 管線設定”

> **Note**
>
> 由於我們不知道您將在 AWS Proton 範本中定義哪些資源，因此您使用主控台建立的角色具有廣泛的權限，而且可以同時用作 AWS Proton 管線服務角色和 AWS Proton 服務角色。對於生產部署，建議您針對 AWS Proton 管線服務角色和 AWS Proton 環境服務角色建立自訂原則，將權限範圍縮減為將部署的特定資源。您可以使用 AWS CLI 或 IAM 建立和自訂這些角色。如需詳細資訊，請參閱 的服務角色 AWS Proton 及 建立服務。

設定使用 AWS Proton

如果您想要使用 AWS CLI 執行 AWS Proton API，請確認您已安裝它。如果尚未安裝，請參閱正在設定 AWS CLI。

AWS Proton 具體配置：

- 若要建立和管理範本：
  - 如果您使用的是範本同步設定，請設定AWS CodeStar 連線。
  - 否則，請設置一個 Amazon S3 存儲桶。
- 若要佈建基礎結構：
  - 對於自我管理的佈建，您必須設定AWS CodeStar 連線。
- (選擇性) 若要佈建管線：

- 對於AWS受管佈建和CodeBuild基於佈建，請設定管線角色。
- 對於自我管理的佈建，請設定管線存放庫。

如需佈建方法的詳細資訊，請參閱the section called “AWS-管理佈建”。

設置一個 Amazon S3 存儲桶

若要設定 S3 儲存貯體，請按照建立您的第一個 S3 儲存貯體中的指示設定 S3 儲存貯體。將您的輸入放置 AWS Proton 在 AWS Proton 可以檢索它們的存儲桶中。這些輸入被稱為模板包。您可以在本指南的其他部分中了解有關它們的更多信息。

設定 AWS CodeStar 連線

要連接 AWS Proton 到存儲庫，您可以創建一個 AWS CodeStar 連接，該連接在第三方源代碼存儲庫上進行新提交時激活管道。

AWS Proton 使用連接：

- 在存儲庫源代碼上進行新提交時激活服務管道。
- 在基礎架構上作為程式碼儲存庫提出提取要求。
- 每當提交推送到更改其中一個模板的模板存儲庫（如果該版本尚不存在）時，請創建一個新的模板次要或主要版本。

您可以連接到 Bitbucket GitHub, GitHub 企業和 GitHub 企業服務器存儲庫. CodeConnections 若要取得更多資訊，請參閱《AWS CodePipeline 使用指南》CodeConnections中的。

若要設定 CodeStar 連線。

1. 開啟 AWS Proton 主控台。
2. 在功能窗格中，選取 [設定]，然後選取 [儲存庫連線]，將您帶到 [開發人員工具設定] 中的 [連線] 頁頁面會顯示連線清單。
3. 選擇 [建立連線] 並依照指示進行。

設定帳戶 CI/CD 管線設定

AWS Proton 可以佈建 CI/CD 管道，將應用程式程式碼部署到您的服務執行個體。管線佈建所需的 AWS Proton 設定取決於您為管線選擇的佈建方法。

AWS-託管和 CodeBuild基於佈建-設置管道角色

透過管AWS理的佈建和CodeBuild 佈建，為您 AWS Proton 佈建管道。因此， AWS Proton 需要提供佈建管道權限的服務角色。這兩種佈建方法中的每一個都使用自己的服務角色。這些角色會在所有 AWS Proton 服務管道之間共用，您可以在帳戶設定中進行一次設定。

使用主控台建立管線服務角色

1. 開啟 AWS Proton 主控台。
2. 在功能窗格中，選擇 [設定]，然後選擇 [帳戶設定]。
3. 在 [帳戶 CI/CD 設定] 頁面中，選擇 [設定]。
4. 執行以下任意一項：
   - 為您 AWS Proton 建立管線服務角色

     [若要啟用管線的管 AWS理佈建] 在 [設定帳戶設定] 頁面的 AWS-managed 佈建管線角色區段中：

     a. 選取 [新增服務角色]。
     b. 輸入角色的名稱，例如**`myProtonPipelineServiceRole`**。
     c. 核取此核取方塊以同意在您的帳戶中建立具有管理權限的 AWS Proton 角色。

     [若要啟用管線的 CodeBuild基礎佈建] 在 [設定帳戶設定] 頁面的 [CodeBuild管線角色] 區段中，選擇 [現有服務角色]，然後選擇您在CloudFormation 管線角色區段中建立的服務角色。或者，如果您未指派 CloudFormation 管線角色，請重複前三個步驟來建立新的服務角色。
   - 若要選擇現有的管線服務角色

     [若要啟用管道的管 AWS理佈建] 在 [設定帳戶設定] 頁面的 AWS-managed 佈建管線角色區段中，選擇 [現有服務角色]，然後選擇帳戶中的 AWS 服務角色。

     [若要啟用管線的 CodeBuild 佈建] 在 [設定帳戶設定] 頁面的 [CodeBuild管線佈建角色] 區段中，選擇 [現有服務角色]，然後選擇 AWS 帳戶中的服務角色。
5. 選擇儲存變更。

   您的新管線服務角色會顯示在 [帳戶設定] 頁面上。

自我管理佈建 — 設定管線儲存庫

透過自我管理佈建，可將提取要求 (PR) AWS Proton 傳送至您已設定的佈建存放庫，而您的自動化程式碼則負責佈建管線。因此， AWS Proton 不需要服務角色即可佈建管線。而是需要註冊的佈建存放庫。存放庫中的自動化程式碼必須擔任適當的角色，以提供佈建管道的權限。

使用主控台註冊管線佈建儲存區域

1. 如果尚未建立 CI/CD 管線佈建存放庫，請建立一個。如需自我管理佈建中管道的詳細資訊，請參閱the section called “自我管理的佈建”。
2. 在功能窗格中，選擇 [設定]，然後選擇 [帳戶設定]。
3. 在 [帳戶 CI/CD 設定] 頁面中，選擇 [設定]。
4. 在 [設定帳戶設定] 頁面的 [CI/CD 管線存放庫] 區段中：
   a. 選取新增儲存區域，然後選擇其中一個儲存區域提供者。
   b. 若要CodeStar 連線，請選擇其中一個連線。

      > **Note**
      >
      > 如果您尚未連線至相關的存放庫提供者帳戶，請選擇 [新增連線]，完成 CodeStar連線建立程序，然後選擇CodeStar連線功能表旁的 [重新整理] 按鈕。現在，您應該可以在菜單中選擇新的連接。

   c. 針對存放庫名稱，選擇您的管線佈建存放庫。下拉式功能表會顯示提供者帳戶中的儲存庫清單。
   d. 針對「分支名稱」，選擇其中一個儲存庫分支。
5. 選擇儲存變更。

   您的管道儲存庫會顯示在 [帳戶設定] 頁面上。

正在設定 AWS CLI

若要使用 AWS CLI 來進行 AWS Proton API 呼叫，請確認您已安裝最新版本的 AWS CLI. 如需詳細資訊，請參閱 AWS Command Line Interface 使用者指南中的 AWS CLI入門。然後，若要開始使用 AWS CLI 與 AWS Proton，請參閱the section called “開始使用 CLI”。

AWS Proton 入門

開始使用之前，請先設定使用AWS Proton並確認您是否符合入門先決條件。

選擇下列一或多個路徑即可開始使AWS Proton用：

- 透過文件連結，遵循引導式範例主控台或 CLI 工作流程。
- 執行引導式範例主控台工作流程。
- 執行引導式範例AWS CLI工作流程。

主題

先決條件

開始使用之前AWS Proton，請確定符合下列先決條件。如需詳細資訊，請參閱設定。

- 您擁有具有管理員權限的 IAM 帳戶。如需詳細資訊，請參閱使用 IAM 進行設定。
- 您擁有AWS Proton服務角色，而AWS Proton管線服務角色會附加至您的帳戶。如需詳細資訊，請參閱 設定 AWS Proton 服務角色 及 的服務角色 AWS Proton。
- 你有一個AWS CodeStar連接。如需詳細資訊，請參閱設定 AWS CodeStar 連線。
- 您熟悉建立AWS CloudFormation範本和 Jinja 參數化。如需詳細資訊，請參閱什麼是AWS CloudFormation？ 在AWS CloudFormation用戶指南和神社網站中。
- 您擁有AWS基礎設施服務的工作知識。
- 您已登入您的AWS 帳戶.

入門工作流程

透過遵循範例步驟和連結，了解如何建立範本服務包、建立和註冊範本，以及建立環境和服務。

開始之前，請確認您已建立AWS Proton服務角色。

如果您的服務範本包含AWS Proton服務管線，請確認您已建立AWS CodeStar連線和AWS Proton管線服務角色。

如需詳細資訊，請參閱AWS Proton服務 API 參考。

範例：開始使用工作流程

1. 如需AWS Proton輸入和輸出AWS Proton 的運作方式的高階檢視，請參閱中的圖表。
2. 建立環境服務包和服務範本套裝軟體。

   a. 識別輸入參數。

   b. 建立結構描述檔案。

   c. 建立基礎架構即程式碼 (IaC) 檔案。

   d. 若要包裝範本服務包，請建立資訊清單檔案，並在目錄中組織您的 IaC 檔案、資訊清單檔案和結構描述檔案。

   e. 讓您的範本套件可供存取AWS Proton。
3. 使用建立並註冊環境範本版本AWS Proton。

   當您使用主控台建立和註冊範本時，會自動建立範本版本。

   當您使用建AWS CLI立和註冊範本時：

   a. 建立環境範本。

   b. 建立環境範本版本。

   如需詳細資訊，請參閱 AWS ProtonAPI 參考資料CreateEnvironmentTemplateVersion中的CreateEnvironmentTemplate和。
4. 發佈您的環境範本以使其可供使用。

   如需詳細資訊，請參閱 AWS ProtonAPI 參考資料UpdateEnvironmentTemplateVersion中的。
5. 若要建立環境，請選取已發佈的環境範本版本，並為必要輸入提供值。

   如需詳細資訊，請參閱 AWS ProtonAPI 參考資料CreateEnvironment中的。
6. 使用建立並註冊服務範本版本AWS Proton。

   當您使用主控台建立和註冊範本時，會自動建立範本版本。

當您使用建AWS CLI立和註冊範本時：

a. 建立服務範本。

b. 建立服務範本版本。

如需詳細資訊，請參閱 AWS ProtonAPI 參考資料CreateServiceTemplateVersion中的CreateServiceTemplate和。

7. 發佈您的服務範本，使其可供使用。

如需詳細資訊，請參閱 AWS ProtonAPI 參考資料UpdateServiceTemplateVersion中的。

8. 若要建立服務，請選取已發佈的服務範本版本，並提供必要輸入的值。

如需詳細資訊，請參閱 AWS ProtonAPI 參考資料CreateService中的。

AWS Management Console入門

開始使用 AWS Proton

- 建立和檢視環境範本。
- 建立、檢視和發佈使用您剛建立之環境範本的服務範本。
- 建立環境和服務 (選用)。
- 刪除服務範本、環境範本、環境和服務 (如果已建立)。

步驟 1：開啟主AWS Proton控台

- 開啟 AWS Proton 主控台

步驟 2：準備使用範例範本

1. 創建一個 Codestar 連接到 Github 並命名連 my-proton-connection接。
2. 導覽至https://github.com/aws-samples/aws-proton-cloudformation-sample-templates。
3. 在您的 Github 帳戶中創建一個存儲庫的分支。

步驟 3：建立環境範本

在導覽窗格中，選擇 [環境範本]。

1. 在「環境範本」頁面中，選擇「建立環境範本」。
2. 在「建立環境樣板」頁面的「樣板選項」段落中，選擇建立用於啟動設定新環境的樣板。
3. 在 [範本套件來源] 區段中，選擇 [從 Git 同步範本套裝軟體]。
4. 在 [範本定義儲存庫] 區段中，選取 [選擇連結的 Git 儲存庫]。
5. my-proton-connection從「存放庫」清單中選取。
6. 從「分支」清單中選取「主要」。
7. 在 Proton 子環境模板詳細信息部分。

   a. 輸入樣板名稱為**`fargate-env`**。

   b. 將環境範本顯示名稱輸入為**`My Fargate Environment`**。

   c. (選擇性) 輸入環境範本的描述。
8. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。
9. 選擇「建立環境範本」。

   您現在進入了一個新頁面，其中會顯示新環境範本的狀態和詳細資料。這些詳細資料包括清單AWS和客戶管理的標籤。 AWS Proton當您建立AWS Proton資源時，會自動為您產生AWS受管理的標籤。如需詳細資訊，請參閱AWS Proton資源和標記。
10. 新環境範本狀態的狀態會以「草稿」狀態開始。您和擁有`proton:CreateEnvironment`權限的其他人都可以檢視和存取它。請按照下一步操作，使其他人可以使用該範本。
11. 在「範本版本」區段中，選擇您剛建立的範本次要版本左側的圓鈕 (1.0)。或者，您可以在資訊警示橫幅中選擇「發佈」，然後略過下一個步驟。
12. 在「範本版本」區段中，選擇「發佈」。
13. 範本狀態會變更為「已發佈」。因為它是模板的最新版本，所以它是推薦版本。
14. 在導覽窗格中，選取 [環境範本]。

    新頁面會顯示環境範本清單以及範本詳細資料。

步驟 4：建立服務範本

建立服務範本。

1. 在功能窗格中，選擇 [服務範本]。
2. 在 [服務範本] 頁面中，選擇 [建立服務範本]。
3. 在 [建立服務範本] 頁面的 [範本套件來源] 區段中，選擇 [從 Git 同步範本套裝軟體]。
4. 在「範本」區段中，選取「選擇連結的 Git 儲存庫」。
5. my-proton-connection從「存放庫」清單中選取。
6. 從「分支」清單中選取「主要」。
7. 在 Proton 子服務模板的詳細信息部分。

   a. 將服務樣板名稱輸入為**backend-fargate-svc**。

   b. 將服務樣板顯示名稱輸入為**My Fargate Service**。

   c. (選擇性) 輸入服務範本的說明。
8. 在「相容的環境範本」區段中。

   - 勾選環境範本 My Fargate 環境左側的核取方塊，以選取新服務範本的相容環境範本。
9. 對於 「加密」設定，請保留預設值。
10. 在「管線定義」區段中。

    - 保持選取「此範本包含 CI/CD 配管」按鈕。
11. 選擇 [建立服務範本]。

    您現在進入了一個新頁面，其中會顯示新服務範本的狀態和詳細資料，包括清單AWS和客戶管理的標籤清單。
12. 新服務範本狀況的狀態會以「草稿」狀態開始。只有管理員可以檢視和存取它。要使服務模板可供開發人員使用，請按照下一步操作。
13. 在「範本版本」區段中，選擇您剛建立的範本次要版本左側的圓鈕 (1.0)。或者，您可以在資訊警示橫幅中選擇「發佈」，然後略過下一個步驟。
14. 在「範本版本」區段中，選擇「發佈」。
15. 範本狀態會變更為「已發佈」。

    服務範本的第一個次要版本已發佈，並可供開發人員使用。因為它是模板的最新版本，所以它是推薦版本。

16. 在功能窗格中，選擇 [服務範本]。

    新頁面會顯示服務範本和詳細資料的清單。

步驟 5：建立環境

在導覽窗格中，選擇 Environments (環境)。

1. 選擇 Create environment (建立環境)。
2. 在 [選擇環境範本] 頁面中，選取您剛才建立的範本。它會命名為「我的 Fargate 環境」。然後，選擇配置。
3. 在「設定環境」頁面的「啟動設定」段落中，選擇啟動設定至AWS Proton。
4. 在 [部署帳戶] 區段中，選取 [這個] AWS 帳戶。
5. 在環境設定中，輸入環境名稱為**`my-fargate-environment`**。
6. 在 [環境角色] 區段中，選取 [新增服務角色]，或者，如果您已經建立AWS Proton服務角色，請選取現有服務角色。

   a. 選取 [新增服務角色] 以建立新角色。

      i. 輸入環境角色名稱為**`MyProtonServiceRole`**。

      ii. 核取此核取方塊以同意為您的帳戶建立具有系統管理權限的AWS Proton服務角色。

   b. 選取現有的服務角色以使用現有角色。

      - 在環境角色名稱下拉式欄位中選取您的角色。
7. 選擇下一步。
8. 在 [設定自訂設定] 頁面上，使用預設值。
9. 選擇 [下一步] 並檢閱您的輸入。
10. 選擇 建立 。

    檢視環境詳細資料和狀態，以及您環境的AWS受管理標籤和客戶管理標籤。
11. 在導覽窗格中，選擇 Environments (環境)。

    新頁面會顯示環境清單，以及狀態和其他環境詳細資訊。

步驟 6：選用-建立服務並部署應用程式

1. 開啟 AWS Proton 主控台。
2. 在功能窗格中，選擇 [服務]。
3. 在「服務」頁面中，選擇「建立服務」。
4. 在「選擇服務模板」頁面中，通過選擇模板卡右上角的單選按鈕來選擇「我的 Fargate 服務」模板。
5. 選擇頁面右下角的「設定」。
6. 在 [設定服務] 頁面的 [服務設定] 區段中，輸入服務名稱**`my-service`**。
7. (選擇性) 輸入服務的說明。
8. 在「服務儲存庫設定」區段中：

   a. 若要CodeStar 連線，請從清單中選擇您的連線。

   b. 對於「存放庫名稱」，請從清單中選擇原始程式碼儲存庫的名稱。

   c. 對於「分支名稱」，請從清單中選擇原始程式碼儲存庫分支的名稱。
9. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。然後選擇 Next (下一步)。
10. 在 [設定自訂設定值] 頁面的 [服務執行個體] 區段的 [新增執行個體] 區段中，遵循下列步驟，為您的服務執行個體參數提供自訂值。

    a. 輸入執行個體名稱**`my-app-service`**。

    b. 選擇服務執行個體的環境**`my-fargate-environment`**。

    c. 保留剩餘例證參數的預設值。

    d. 保留管線輸入的預設值。

    e. 選擇 [下一步] 並檢閱您的輸入。

    f. 選擇 [建立] 並檢視您的服務狀態和詳細資料。
11. 在服務詳細資訊頁面中，選擇「概觀」和「管線」標籤，以檢視服務執行個體和管線的狀態。在這些頁面上，您還可以查看AWS和客戶管理的標籤。 AWS Proton自動為您建立AWS受管理的標籤。選擇「管理標籤」以建立和修改客戶管理的標籤。如需標記的詳細資訊，請參閱AWS Proton資源和標記。
12. 服務處於作用中狀態之後，請在「總覽」索引標籤的「服務執行處理」區段中，選擇服務執行處理的名稱my-app-service。

    您現在位於服務執行個體詳細資訊頁面。

13. 若要檢視您的應用程式，請在 [輸出] 區段中，將ServiceEndpoint連結複製到您的瀏覽器。

    您會在網頁中看到一個AWS Proton圖形。

14. 建立服務後，在功能窗格中選擇 [服務] 以檢視您的服務清單。

步驟 7：清理。

1. 開啟 AWS Proton 主控台。
2. 刪除服務 (如果您已建立服務)

    a. 在功能窗格中，選擇 [服務]。

    b. 在「服務」頁面中，選擇服務名稱「我的服務」。

    您現在在我的服務的服務詳細信息頁面上。

    c. 在頁面的右上角，選擇「動作」，然後選擇「刪除」。

    d. 強制回應會提示您確認刪除動作。

    e. 按照說明進行操作，然後選擇是，刪除。

3. 刪除環境

    a. 在導覽窗格中，選擇 Environments (環境)。

    b. 在「環境」頁面中，選取您剛建立之環境左側的圓鈕。

    c. 選擇動作，然後選擇刪除。

    d. 強制回應會提示您確認刪除動作。

    e. 按照說明進行操作，然後選擇是，刪除。

4. 刪除服務範本

    a. 在功能窗格中，選擇 [服務範本]。

    b. 在「服務範本」頁面中，選取服務範本左側的圓鈕my-svc-template。

    c. 選擇動作，然後選擇刪除。

    d. 強制回應會提示您確認刪除動作。

    e. 按照說明進行操作，然後選擇是，刪除。這會刪除服務範本及其所有版本。

5. 刪除環境範本

    a. 在導覽窗格中，選擇 [環境範本]。

    b. 在「環境範本」頁面中，選取左側的圓鈕my-env-template。

c. 選擇動作，然後選擇刪除。

d. 強制回應會提示您確認刪除動作。

e. 按照說明進行操作，然後選擇是，刪除。這會刪除環境範本及其所有版本。

6. 刪除您的 Codestar 連線

AWS CLI入門

若要開始AWS Proton使用AWS CLI，請遵循本教學課程。本教程演示了一個基AWS Proton於. AWS Fargate 本教學課程也會佈建 CI/CD 管線，以部署具有顯示影像的靜態網站。

在開始之前，請確定您已正確設定。如需詳細資訊，請參閱 the section called “先決條件”。

步驟 1：註冊環境範本

在此步驟中，身為管理員，您可以註冊一個範例環境範本，該範本包含 Amazon Elastic Container Service (Amazon ECS) 叢集，以及具有兩個公有/私有子網路的 Amazon 虛擬私有雲端 (Amazon VPC)。

註冊環境範本

1. 將AWS Proton樣本 CloudFormation 模板存儲庫分支到您的 GitHub 帳戶或組織中。此儲存庫包含我們在本教學課程中使用的環境和服務範本。

   然後，使AWS Proton用註冊您的分叉存儲庫。如需詳細資訊，請參閱the section called “建立儲庫連結”。

2. 建立環境範本。

   環境範本資源會追蹤環境範本版本。

   ```
   $ aws proton create-environment-template \
     --name "fargate-env" \
     --display-name "Public VPC Fargate" \
     --description "VPC with public access and ECS cluster"
   ```

3. 建立範本同步設定。

   AWS Proton設定儲存庫與環境範本之間的同步關係。然後，它會在DRAFT狀態中創建模板版本 1.0。

```
$ aws proton create-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "environment-templates/fargate-env"
```

4. 等待環境範本版本成功註冊。

   當此命令返回時退出狀態為0，表示版本註冊完成。這在腳本中非常有用，以確保您可以在下一個步驟中成功運行命令。

```
$ aws proton wait environment-template-version-registered \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0"
```

5. 發佈環境範本版本，使其可用於建立環境。

```
$ aws proton update-environment-template-version \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

步驟 2：註冊服務範本

在此步驟中，身為管理員，您可以註冊一個範例服務範本，其中包含在負載平衡器和使用的 CI/CD 管道後佈建 Amazon ECS Fargate 服務所需的所有資源。AWS CodePipeline

若要註冊服務範本

1. 建立服務範本。

   服務範本資源會追蹤服務範本版本。

```
$ aws proton create-service-template \
  --name "load-balanced-fargate-svc" \
  --display-name "Load balanced Fargate service" \
```

```
  --description "Fargate service with an application load balancer"
```

2. 建立範本同步設定。

   AWS Proton設定儲存庫與服務範本之間的同步關係。然後，它會在DRAFT狀態中創建模板版本1.0。

```
$ aws proton create-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "service-templates/load-balanced-fargate-svc"
```

3. 等待服務範本版本成功註冊。

   當此命令返回時退出狀態為0，表示版本註冊完成。這在腳本中非常有用，以確保您可以在下一個步驟中成功運行命令。

```
$ aws proton wait service-template-version-registered \
  --template-name "load-balanced-fargate-svc" \
  --major-version "1" \
  --minor-version "0"
```

4. 發佈服務範本版本，使其可用於建立服務。

```
$ aws proton update-service-template-version \
  --template-name "load-balanced-fargate-svc" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

步驟 3：部署環境

在此步驟中，身為管理員，您可以從AWS Proton環境範本實例化環境。

若要部署環境

1. 取得您註冊之環境範本的範例規格檔案。

您可以`environment-templates/fargate-env/spec/spec.yaml`從範本範例存放庫下載檔案。或者，您可以在本地獲取整個存儲庫並從`environment-templates/fargate-env`目錄中運行create-environment命令。

2. 建立環境。

   AWS Proton從您的環境規格讀取輸入值，將它們與您的環境範本結合，並使用AWS Proton服務角色在AWS帳戶中佈建環境資源。

```
$ aws proton create-environment \
  --name "fargate-env-prod" \
  --template-name "fargate-env" \
  --template-major-version 1 \
  --proton-service-role-arn "arn:aws:iam::123456789012:role/AWSProtonServiceRole" \
  --spec "file://spec/spec.yaml"
```

3. 等待環境成功部署。

```
$ aws proton wait environment-deployed --name "fargate-env-prod"
```

第 4 步：部署服務 [應用程序開發人員]

在先前的步驟中，系統管理員已註冊並發佈服務範本，並部署環境。身為應用程式開發人員，您現在可以建立AWS Proton服務並將其部署到AWS Proton環境

若要部署服務

1. 取得管理員註冊之服務範本的範例規格檔案。

   您可以`service-templates/load-balanced-fargate-svc/spec/spec.yaml`從範本範例存放庫下載檔案。或者，您可以在本地獲取整個存儲庫並從`service-templates/load-balanced-fargate-svc`目錄中運行create-service命令。

2. 將AWS Proton範例服務儲存庫分配到您的 GitHub 帳戶或組織中。該存儲庫包括我們在本教程中使用的應用程序源代碼。

3. 建立服務。

   AWS Proton從您的服務規格讀取輸入值，將它們與您的服務範本結合，並在規格中指定的環境中佈建您AWS帳戶中的服務資源。AWS CodePipeline管線會從您在命令中指定的儲存庫部署應用程式程式碼。

```
$ aws proton create-service \
  --name "static-website" \
  --repository-connection-arn \
    "arn:aws:codestar-connections:us-east-1:123456789012:connection/your-codestar-connection-id" \
  --repository-id "your-GitHub-account/aws-proton-sample-services" \
  --branch-name "main" \
  --template-major-version 1 \
  --template-name "load-balanced-fargate-svc" \
  --spec "file://spec/spec.yaml"
```

4. 等待服務成功部署。

```
$ aws proton wait service-created --name "static-website"
```

5. 檢索輸出並查看您的新網站。

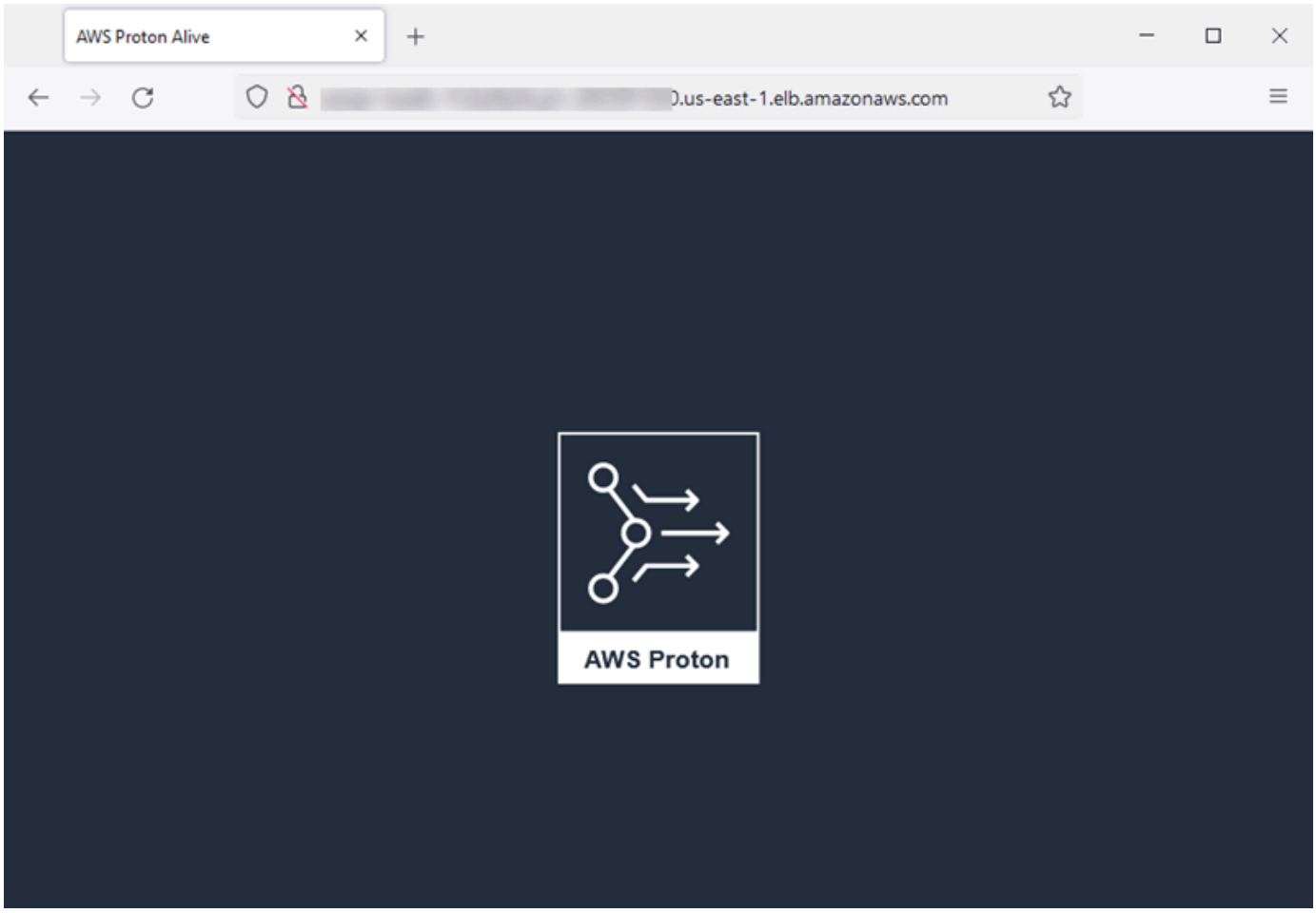
   執行以下命令：

```
$ aws proton list-service-instance-outputs \
  --service-name "static-website" \
  --service-instance-name load-balanced-fargate-svc-prod
```

   命令的輸出應該類似於以下內容：

```
{
    "outputs": [
        {
            "key": "ServiceURL",
            "valueString": "http://your-service-endpoint.us-east-1.elb.amazonaws.com"
        }
    ]
}
```

   ServiceURL執行個體輸出的值是新服務網站的端點。使用您的瀏覽器導航到它。您應該會在靜態頁面上看到下列圖形：



步驟 5：清理（可選）

在此步驟中，當您完成探索作為本教學課程一部分所建立的AWS資源，並節省與這些資源相關的成本時，您將其刪除。

刪除教學課程資源

1. 若要刪除服務，請執行下列命令：

```
$ aws proton delete-service --name "static-website"
```

2. 若要刪除環境，請執行下列命令：

```
$ aws proton delete-environment --name "fargate-env-prod"
```

3. 若要刪除服務範本，請執行下列命令：

```
$ aws proton delete-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE"
$ aws proton delete-service-template --name "load-balanced-fargate-svc"
```

4. 若要刪除環境範本，請執行下列指令：

```
$ aws proton delete-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT"
$ aws proton delete-environment-template --name "fargate-env"
```

模AWS Proton板庫

AWS Proton專案團隊會在上維護範本範例資料庫 GitHub。此程式庫包含許多常見環境和應用程式基礎結構案例的基礎結構即程式碼 (IaC) 檔案範例。

模板庫存儲在兩個存儲 GitHub 庫中：

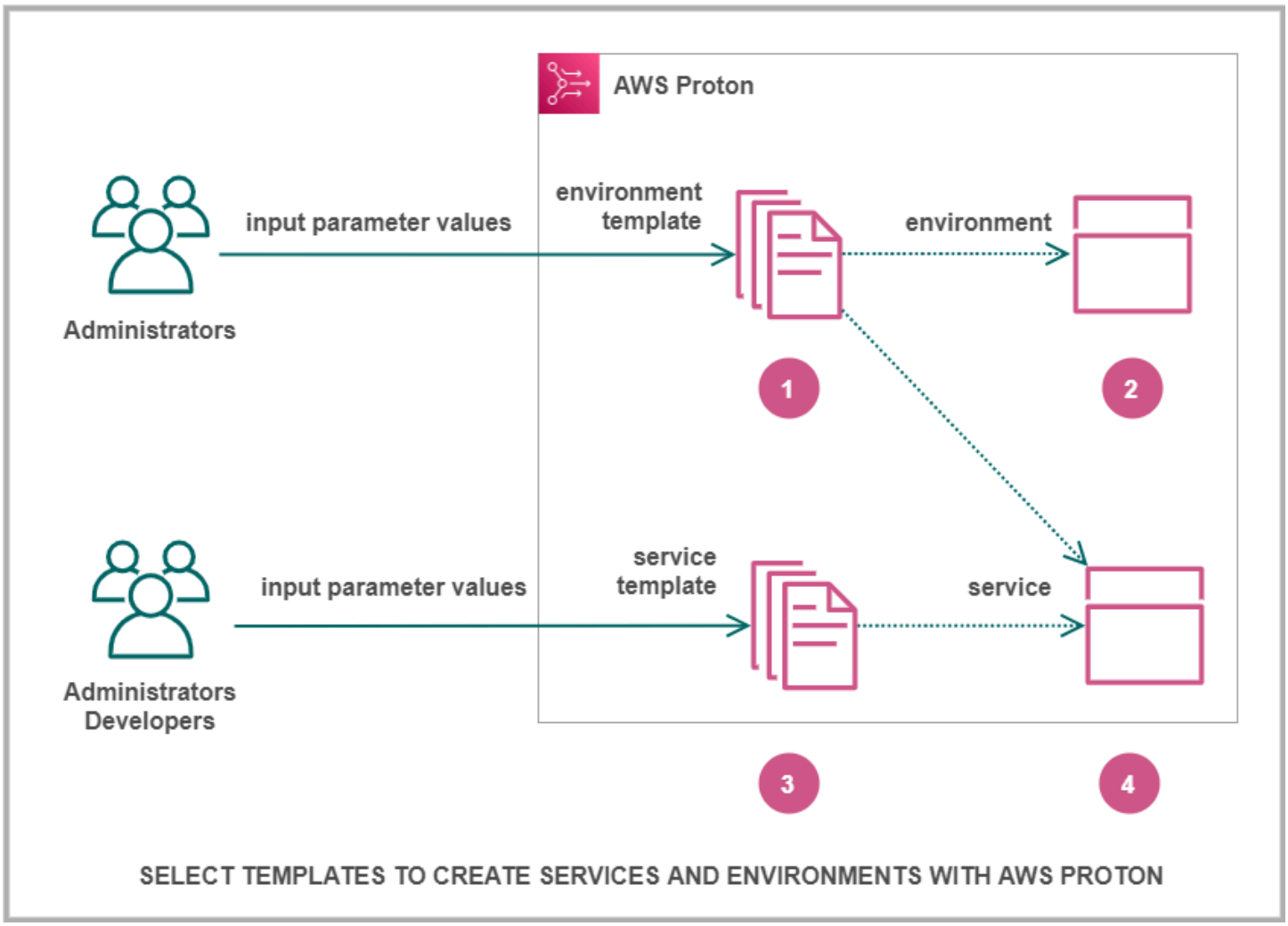
- aws-proton-cloudformation-sample- 模板-AWS CloudFormation與 Jinja 作為其 IaC 語言一起使用的模板包的示例。您可以將這些範例用於AWS-管理佈建環境。
- aws-proton-terraform-sample- 模板-使用 Terraform 作為其 Ia C 語言的模板包的示例。您可以將這些範例用於自我管理的佈建環境。

這些存儲庫中的每一個都有一個README文件，其中包含有關存儲庫內容和結構的完整信息。每個範例都包含範本涵蓋的使用案例、範例架構以及範本使用的輸入參數的相關資訊。

您可以通過將其中一個庫的存儲庫分配到您的 GitHub 帳戶中，直接使用此庫中的模板。或者，使用這些範例做為開發環境和服務範本的起點。

AWS Proton 的運作方式

使用AWS Proton，您可以佈建環境，然後在這些環境中執行的服務。環境和服務分別以您在已建立版本化的範本程式庫中選擇的環境和服務範AWS Proton本為基礎。



1 當

您以管理員身分選取環境範本時AWS Proton，您可以為必要的輸入參數提供值。

2 AWS

Proton使用環境範本和參數值來佈建您的環境。

3 當

您身為開發人員或管理員，使用選取服務範本時AWS Proton，您會提供必要輸入參數的值。您也可以選取要部署應用程式或服務的環境。

4

AWS

Proton使用服務範本，以及您的服務和選取的環境參數值來佈建服務。

您可以為輸入參數提供值，以自訂範本以供重複使用，以及多個使用案例、應用程式或服務。

若要進行這項工作，您可以建立環境或服務範本套裝軟體，並分別將它們上傳至已註冊的環境或服務範本。

模板包包含一切AWS Proton需要提供環境或服務。

當您建立環境或服務範本時，您會上傳範本套裝軟體，其中包含參數化的基礎結構即程式碼 (IAC) 檔案，可AWS Proton用來佈建環境或服務。

當您選取環境或服務範本來建立或更新環境或服務時，您會提供範本服務包 IaC 檔案參數的值。

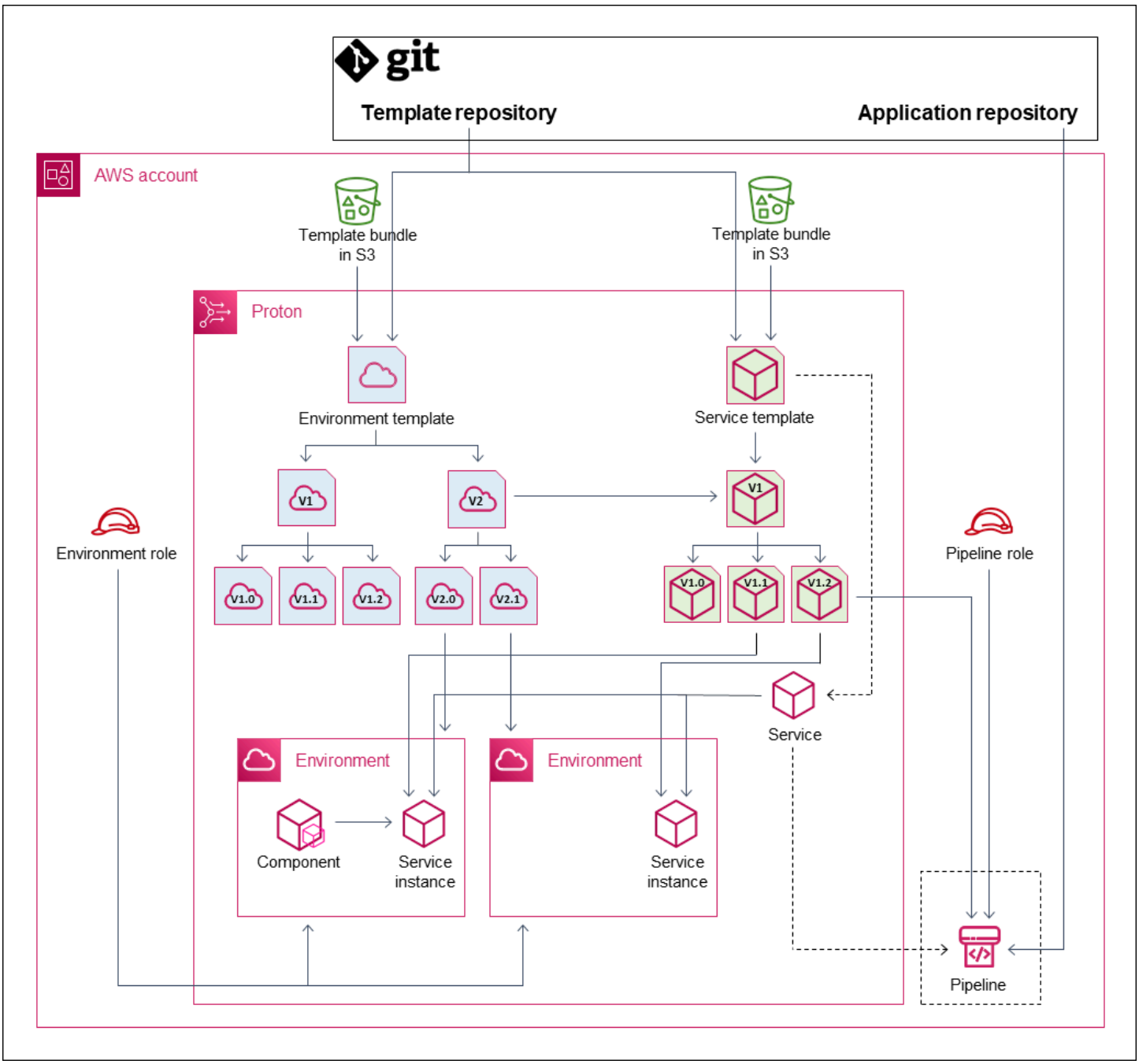
主題

- AWS Proton物件
- 如何AWS Proton佈建基礎設
- AWS Proton 術語

AWS Proton物件

下圖顯示了主要AWS Proton對象及其與其他AWS和第三方對象的關係。箭頭表示數據流的方向（依賴關係的反向方向）。

我們遵循這些AWS Proton對象的簡要描述和參考鏈接的圖表。



- 環境範本 — 可用來建立AWS Proton環境的環境範本版本集合。

  如需詳細資訊，請參閱 模板創作和捆綁 及 範本。

- 環境範本版本 — 環境範本的特定版本。從 S3 儲存貯體或 Git 儲存庫取得範本組合包做為輸入。該軟件包指定基礎設施作為代碼（IaC）和AWS Proton環境的相關輸入參數。

  如需詳細資訊，請參閱 the section called “版本”、the section called “發佈” 及 the section called “範本同步設定”。

- 環境 — AWS Proton 服務部署到的一組共用AWS基礎結構資源和存取原則。 AWS透過使用特定參數值叫用的環境範本版本來佈建資源。存取原則是以服務角色提供。

  如需詳細資訊，請參閱環境。

- 服務範本 — 可用來建立AWS Proton服務的服務範本版本集合。

  如需詳細資訊，請參閱 模板創作和捆綁 及 範本。

- 服務範本版本 — 服務範本的特定版本。從 S3 儲存貯體或 Git 儲存庫取得範本組合包做為輸入。該服務包指定基礎結構代碼（IaC）和相關的輸入參數的AWS Proton服務。

  服務範本版本也會根據版本指定服務執行個體的下列限制：

  - 相容的環境範本 — 執行個體只能在以這些相容環境範本為基礎的環境中執行。
  - 支援的元件來源 — 開發人員可與執行個體建立關聯的元件類型。

  如需詳細資訊，請參閱 the section called “版本”、the section called “發佈” 及 the section called “範本同步設定”。

- 服務 — 服務執行個體的集合，這些執行個體使用服務範本中指定的資源執行應用程式，以及選擇性地使用將應用程式程式碼部署到這些執行個體的 CI/CD 管線。

  在圖表中，Service 範本中的虛線表示服務會將範本傳遞至服務執行個體和管線。

  如需詳細資訊，請參閱服務。

- 服務執行個體 — 在特定AWS Proton環境中執行應用程式的一組AWS基礎結構資源。 AWS透過使用以特定參數值叫用的服務範本版本來佈建資源。

  如需詳細資訊，請參閱 服務 及 the section called “更新實體”。

- 管道 — 選用的 CI/CD 管線，可將應用程式部署到服務執行個體中，並具有佈建此管道的存取原則。存取原則是以服務角色提供。服務不一定會有關聯的管AWS Proton線，您可以選擇在以外的地方管理應用程式程式碼部署。AWS Proton

  在圖中，Service 的虛線和 Pipeline 周圍的虛線方塊表示，如果您選擇自行管理 CI/CD 部署，則可能無法建立管AWS Proton道，而且您自己的管道可能不在您的帳戶中。AWS

  如需詳細資訊，請參閱 服務 及 the section called “更新管道”。

- 元件 — 開發人員定義的服務執行個體擴充功能。除了環境和服務執行個體所提供的資源之外，指定特定應用程式可能需要的其他AWS基礎結構資源。平台團隊藉由將元件角色附加至環境來控制元件可佈建的基礎結構。

如需詳細資訊，請參閱元件。

如何AWS Proton佈建基礎設

AWS Proton可以使用以下幾種方式之一佈建基礎架構：

- AWS-管理佈建 — 代表您AWS Proton呼叫佈建引擎。此方法僅支持AWS CloudFormation模板包。如需詳細資訊，請參閱the section called “AWS CloudFormation 合家歡的文件”。
- CodeBuild佈建 — AWS Proton 用AWS CodeBuild於執行您提供的殼層命令。您的命令可以讀取AWS Proton提供的輸入，並負責佈建或取消佈建基礎結構，以及產生輸出值。此方法的範本組合包括資訊清單檔案中的命令，以及這些命令可能需要的任何程式、指令碼或其他檔案。

  作為使用CodeBuild佈建的範例，您可以包含使用佈建AWS資源的AWS Cloud Development Kit (AWS CDK)程式碼，以及安裝 CDK 並執行 CDK 程式碼的資訊清單。

  如需詳細資訊，請參閱the section called “CodeBuild 捆綁”。

  > **Note**
  >
  > 您可以將CodeBuild佈建與環境和服務搭配使用。目前您無法以這種方式佈建元件。

- 自我管理佈建 — 向您提供的儲存庫AWS Proton發出提取要求 (PR)，您自己的基礎結構部署系統會在其中執行佈建程序。此方法僅支持地形模板包。如需詳細資訊，請參閱the section called “地形 IAC 文件”。

AWS Proton分別決定並設定每個環境和服務的佈建方法。當您建立或更新環境或服務時，會AWS Proton檢查您提供的範本服務包，並決定範本服務包指示的佈建方法。您可以在環境層級提供環境及其潛在服務的佈建方法所需的參數，包括 AWS Identity and Access Management (IAM) 角色、環境帳戶連線或基礎結構存放庫。

無論佈建方法AWS Proton為何，用於佈建服務的開發人員都具有相同的體驗。開發人員不需要知道佈建方法，也不需要變更服務佈建程序中的任何項目。服務範本會設定佈建方法，以及開發人員部署服務的每個環境，以提供服務執行處理佈建的必要參數。

下圖總結了不同佈建方法的一些主要特徵。表格後面的段落提供有關每個方法的詳細資訊。

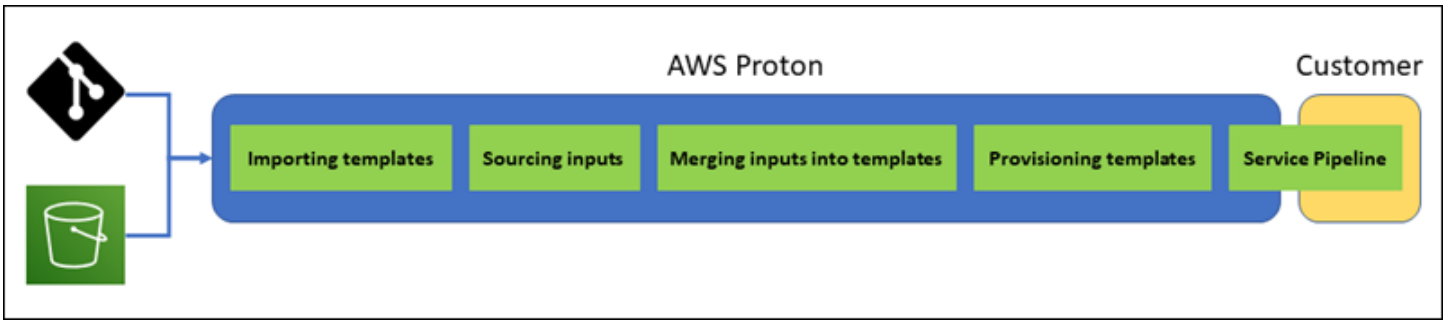
佈建方法	範本	佈建	狀態追蹤
AWS-管理	清單，模式，IaC 文件（ ）CloudFormation	AWS Proton(通過 CloudFormation)	AWS Proton(通過 CloudFormation)
CodeBuild	清單（使用命令），模式，命令依賴關係（例如AWS CDK代碼）	AWS Proton(通過 CodeBuild)	AWS Proton（您的命令返回狀態CodeBuild）
自我管理	清單，模式，IaC 文件（地形）	您的程式碼 (透過 Git 動作)	您的代碼（AWS通過 API 調用傳遞給）

AWS管理佈建的運作方式

當環境或服務使用AWS管理佈建時，基礎結構佈建如下：

1. AWS Proton客戶 (系統管理員或開發人員) 會建立AWS Proton資源 (環境或服務)。客戶選取資源的範本，並提供必要的參數。如需詳細資訊，請參閱下一節「」the section called “AWS受管佈建的考量”。
2. AWS Proton呈現用於佈建資源的完整AWS CloudFormation模板。
3. AWS Proton調AWS CloudFormation用使用呈現的模板開始佈建。
4. AWS Proton持續監控AWS CloudFormation部署。
5. 佈建完成時，會在發生故障時AWS Proton報告錯誤，並在成功的情況下擷取佈建輸出，例如 Amazon VPC ID。

下圖顯示了直接照AWS Proton顧這些步驟的大部分。



AWS受管佈建的考量

- 基礎架構佈建角色 — 當環境或其中執行的任何服務執行個體可能使用 AWS-managed 佈建時，管理員需要設定 IAM 角色 (直接或作為AWS Proton環境帳戶連線的一部分)。 AWS Proton使用此角色來佈建這些AWS受管佈建資源的基礎結構。角色應具有用AWS CloudFormation於建立這些資源範本所包含之所有資源的權限。

  如需詳細資訊，請參閱 the section called “IAM 角色” 及 the section called “服務角色政策範例”。

- 服務佈建 — 當開發人員將使用 AWS-managed 佈建的服務執行個體部署到環境時，AWS Proton會使用提供給該環境的角色來佈建服務執行個體的基礎結構。開發人員看不到這個角色，也無法變更它。

- 使用管線服務 — 使用 AWS-managed 佈建的服務範本可能包含寫入 AWS CloudFormation YAML 結構描述中的管線定義。 AWS Proton還通過調用創建管道AWS CloudFormation。AWS Proton用來建立管線的角色與每個個別環境的角色不同。此角色僅在AWS帳戶層級提供一次，並且用於佈建和管理所有管理的AWS管道。AWS Proton此角色應具有建立管道和管道所需的其他資源的權限。

  下列程序顯示如何將管道角色提供給AWS Proton。

  AWS Proton console

  提供管線角色

  1. 在AWS Proton主控台的功能窗格中，選擇 [設定] > [帳戶設定]，然後選擇 [設定]。
  2. 您可以使用管道管理的角色區段，為 AWS AWS-managed 佈建設定新的或現有的管線角色。

  AWS Proton API

  提供管線角色

  1. 使用 UpdateAccountSettingsAPI 動作。
  2. 在`pipelineServiceRoleArn`參數中提供管線的 Amazon Resource Name (ARN)。

  AWS CLI

  提供管線角色

  執行以下命令：

```
$ aws proton update-account-settings \
    --pipeline-service-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

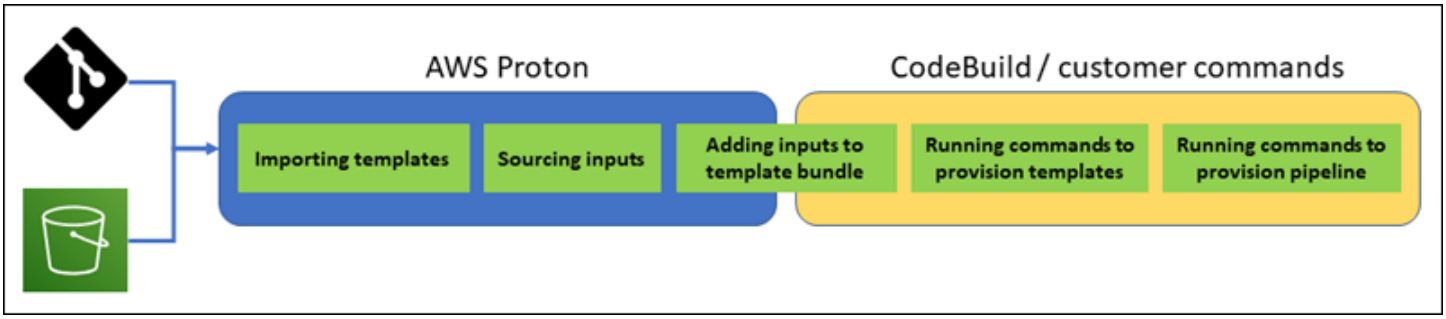
CodeBuild佈建的運作方式

當環境或服務使用CodeBuild佈建時，會以下列方式佈建基礎結構：

1. AWS Proton客戶 (系統管理員或開發人員) 會建立AWS Proton資源 (環境或服務)。客戶選取資源的範本，並提供必要的參數。如需詳細資訊，請參閱下一節「」the section called “CodeBuild佈建的考量”。
2. AWS Proton轉譯具有輸入參數值的輸入檔案，以供佈建資源。
3. AWS ProtonCodeBuild打電話開始工作。工CodeBuild作會執行範本中指定的客戶 shell 命令。這些命令會佈建所需的基礎結構，同時選擇性地讀取輸入值。
4. 佈建完成後，最終的客戶命令會將佈建狀態返回至CodeBuild並呼叫 NotifyResourceDeploymentStatusChangeAWS ProtonAPI 動作以提供輸出，例如 Amazon VPC ID (如果有的話)。

> **Important**
>
> 請確定您的命令正確地將佈建狀態傳回給CodeBuild並提供輸出。如果沒有，AWS Proton則無法正確追蹤佈建狀態，也無法為服務執行個體提供正確的輸出。

下圖說明了AWS Proton執行的步驟以及命令在CodeBuild工作中執行的步驟。



CodeBuild佈建的考量

- 基礎架構佈建角色 — 當環境或其中執行的任何服務執行個體可能使用CodeBuild基於佈建時，管理員需要設定 IAM 角色 (直接或作為AWS Proton環境帳戶連線的一部分)。 AWS Proton使用此角色來佈CodeBuild建這些佈建資源的基礎結構。該角色應具有權限，可用CodeBuild於在這些資源佈建的範本中建立命令的所有資源。

  如需詳細資訊，請參閱 the section called “IAM 角色” 及 the section called “服務角色政策範例”。

- 服務佈建 — 當開發人員將使用佈CodeBuild建的服務執行個體部署到環境時，AWS Proton會使用提供給該環境的角色來佈建服務執行個體的基礎結構。開發人員看不到這個角色，也無法變更它。
- 使用管道服務 — 使用CodeBuild佈建的服務範本可能包含用於佈建管線的命令。 AWS Proton還通過調用創建管道CodeBuild。AWS Proton用來建立管線的角色與每個個別環境的角色不同。此角色僅在AWS帳戶層級提供一次，並且用於佈建和管理所有CodeBuild基於管道的角色。AWS Proton此角色應具有建立管道和管道所需的其他資源的權限。

  下列程序顯示如何將管道角色提供給AWS Proton。

  AWS Proton console

  提供管線角色

  1. 在AWS Proton主控台的功能窗格中，選擇 [設定] > [帳戶設定]，然後選擇 [設定]。
  2. 您可以使用 Codebuild 管線佈建角色區段來設定用於CodeBuild佈建的新管線角色或現有的管線角色。

  AWS Proton API

  提供管線角色

  1. 使用 UpdateAccountSettingsAPI 動作。
  2. 在`pipelineCodebuildRoleArn`參數中提供管線的 Amazon Resource Name (ARN)。

  AWS CLI

  提供管線角色

  執行以下命令：

```
$ aws proton update-account-settings \
```

```
    --pipeline-codebuild-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

自我管理佈建的運作方式

當環境設定為使用自我管理佈建時，會依下列方式佈建基礎結構：

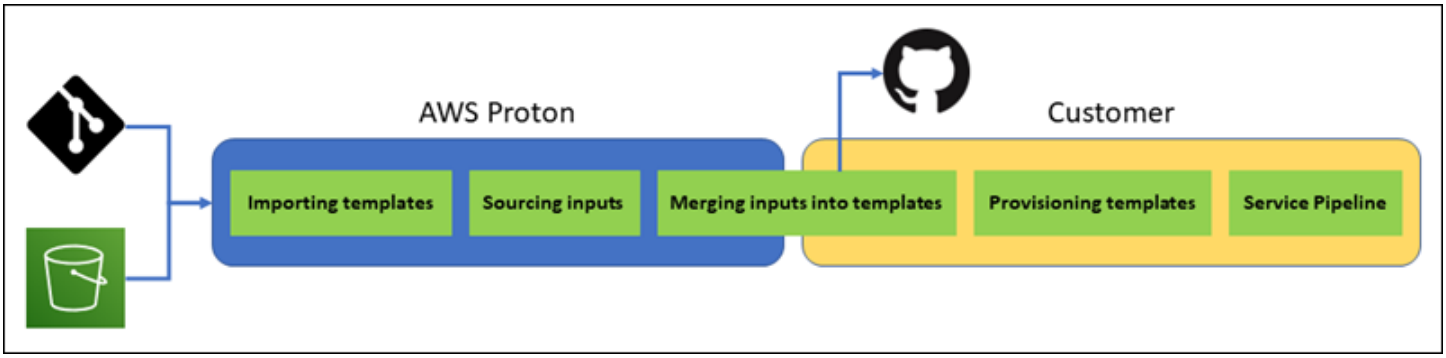
1. AWS Proton客戶 (系統管理員或開發人員) 會建立AWS Proton資源 (環境或服務)。客戶選取資源的範本，並提供必要的參數。對於環境，客戶也會提供連結的基礎架構儲存庫。如需詳細資訊，請參閱下一節「」the section called “自我管理佈建的考量”。
2. AWS Proton呈現一個完整的地形範本。它由一個或多個 Terraform 文件組成，可能在多個文件夾中，以及一個`.tfvars`變量文件。 AWS Proton將資源建立呼叫上提供的參數值寫入此變數檔案。
3. AWS Proton使用轉譯的 Terraform 範本，將 PR 提交至基礎結構儲存庫。
4. 當客戶 (系統管理員或開發人員) 合併 PR 時，客戶的自動化會觸發佈建引擎，使用合併的範本啟動佈建基礎結構。

> **Note**
>
> 如果客戶 (管理員或開發人員) 關閉 PR，則會將 PR AWS Proton 識別為已關閉，並將部署標示為已取消。

5. 佈建完成後，客戶的自動化會呼叫 NotifyResourceDeploymentStatusChangeAWS ProtonAPI 動作來指示完成、提供狀態 (成功或失敗)，並提供如 Amazon VPC ID 的輸出 (如果有的話)。

> **Important**
>
> 請確定您的自動化程式碼會AWS Proton重新呼叫佈建狀態和輸出。如果沒有，AWS Proton可能會將佈建視為擱置時間超過應該的時間，並持續顯示 [進行中] 狀態。

下圖說明AWS Proton執行的步驟，以及您自己的佈建系統執行的步驟。



自我管理佈建的考量

- 基礎結構存放庫 — 當管理員為自我管理佈建設定環境時，必須提供連結的基礎結構存放庫。 AWS Proton將 PRs 提交至此儲存區域，以佈建環境的基礎架構以及所有部署至該儲存區域的服務執行個體。存放庫中客戶擁有的自動化動作應假設具有權限的 IAM 角色，以建立環境和服務範本包含的所有資源，以及反映目標AWS帳戶的身分識別。如需擔任角色的GitHub動作範例，請參閱動作的「設定AWS認證」動GitHub作說明文件中的「假設角色」。
- 權限 — 您的佈建程式碼必須視需要使用帳號進行驗證 (例如，對AWS帳號進行驗證)，並提供資源佈建授權 (例如，提供角色)。
- 服務佈建 — 當開發人員將使用自我管理佈建的服務執行個體部署到環境時，將 PR 提AWS Proton交至與環境相關聯的存放庫，以佈建服務執行個體的基礎結構。開發人員看不到存儲庫，也無法更改它。

  > **Note**
  >
  > 無論佈建方法為何，建立服務的開發人員都會使用相同的程序，而且差異會從中抽取出來。但是，使用自我管理的佈建，開發人員可能會遇到較慢的回應，因為他們需要等到有人（可能不是本身）合併基礎結構存放庫中的 PR，然後才能啟動佈建。

- 使用管道服務 — 具有自我管理佈建的環境的服務範本可能包含以 Terraform HCL 撰寫的AWS CodePipeline管線定義 (例如管線)。若AWS Proton要啟用佈建這些管線，管理員會提供連結的管線存放庫AWS Proton。佈建管道時，存放庫中客戶擁有的自動化動作應假設具有佈建管道許可的 IAM 角色，以及反映目標AWS帳戶的身分。管線存放庫和角色與用於每個個別環境的存放庫和角色不同。連結的存放庫僅在AWS帳戶層級提供一次，並且用於佈建和管理所有管道。AWS Proton該角色應具有建立管道和管道所需的其他資源的權限。

  下列程序顯示如何將管道存放庫和角色提供給AWS Proton。

AWS Proton console

提供管線角色

1. 在AWS Proton主控台的功能窗格中，選擇 [設定] > [帳戶設定]，然後選擇 [設定]。
2. 您可以使用 CI/CD 配管儲存區域段落來設定新的或現有的儲存區域連結。

AWS Proton API

提供管線角色

1. 使用 UpdateAccountSettingsAPI 動作。
2. 在`pipelineProvisioningRepository`參數中提供管線存放庫的提供者、名稱和分支。

AWS CLI

提供管線角色

執行以下命令：

```
$ aws proton update-account-settings \
    --pipeline-provisioning-repository \
        "provider=GITHUB,name=my-pipeline-repo-name,branch=my-branch"
```

- 刪除自我管理的佈建資源 — Terraform 模組除了資源定義之外，還可能包含 Terraform 作業所需的組態元素。因此，AWS Proton無法刪除環境或服務執行個體的所有 Terraform 檔案。而是AWS Proton標記要刪除的檔案，並在 PR 中繼資料中更新旗標。您的自動化操作可以讀取該標誌並使用它來觸發地形摧毀命令。

AWS Proton 術語

環境範本

定義多個應用程式或資源所使用的共用基礎結構，例如 VPC 或叢集。

環境範本套件

您上載的檔案集合，以便在中建立和註冊環境範本AWS Proton。環境範本包包含下列項目：

1. 將基礎結構定義為程式碼輸入參數的結構描述檔案。

2. 一種基礎結構即程式碼 (IaC) 檔案，可定義共用基礎結構，例如 VPC 或叢集，可供多個應用程式或資源使用。
3. 列出 IaC 檔案的資訊清單檔案。

Environment (環境)

佈建的共用基礎結構，例如 VPC 或叢集，可供多個應用程式或資源使用。

服務範本

定義在環境中部署和維護應用程式或微服務所需的基礎結構類型。

服務範本

您上傳以在中建立和註冊服務範本的檔案集合AWS Proton。服務範本包含下列項目：

1. 將基礎結構定義為程式碼 (IaC) 輸入參數的結構描述檔案。
2. 定義在環境中部署和維護應用程式或微服務所需的基礎結構的 IaC 檔案。
3. 列出 IaC 檔案的資訊清單檔案。
4. 選用
   a. 定義服務管線基礎結構的 IaC 檔案。
   b. 列出 IaC 檔案的資訊清單檔案。

服務

在環境中部署和維護應用程式或微服務所需的佈建基礎結構。

服務執行個體

在環境中支援應用程式或微服務的佈建基礎結構。

服務管線

支援管線的佈建基礎架構。

範本

範本的主要或次要版本。如需詳細資訊，請參閱版本化模板。

輸入參數

在架構文件中定義，並在基礎設施中作為代碼（IaC）文件中使用，以便 IaC 文件可以重複使用，並用於各種用例。

結構描述檔

將基礎結構定義為代碼文件輸入參數。

規格文件

將基礎架構的值指定為程式碼檔案輸入參數，如結構描述檔案中所定義。

資訊清單檔案

將基礎結構列為程式碼檔案。

編寫範本和建立套裝軟體 AWS Proton

AWS Proton 根據基礎架構作為代碼（IaC）文件為您提供資源。您描述了可重複使用的 IaC 文件中的基礎 若要讓檔案在不同的環境和應用程式中重複使用，您可以將它們編寫為範本、定義輸入參數，並在 IaC 定義中使用這些參數。當您稍後建立佈建資源 (環境、服務執行個體或元件) 時， AWS Proton 會使用轉譯引擎，該引擎將輸入值與範本結合，以建立可供佈建的 IaC 檔案。

管理員將大多數範本編寫為範本服務包，然後將其上傳並註冊到中 AWS Proton。本頁的其餘部分將討論這些 AWS Proton 範本套裝軟體。直接定義的組件是一種異常-開發人員創建它們並直接提供 IaC 模板文件。如需元件的詳細資訊，請參閱元件。

主題

- 模板捆綁
- AWS Proton 參數
- AWS Proton 基礎架構即程式碼檔
- 結構描述檔
- 包裝的範本檔案 AWS Proton
- 範本服務包考量

模板捆綁

身為管理員，您可以使用建立和註冊範本 AWS Proton。您可以使用這些範本來建立環境和服務。當您建立服務時，會 AWS Proton 佈建服務執行個體並將其部署到選取的環境。如需詳細資訊，請參閱 AWS Proton適用於平台團隊。

若要在中建立並註冊範本 AWS Proton，請上傳範本套件包，其中包含基礎結構即程式碼 (IaC) 檔案，AWS Proton 需要佈建和環境或服務。

範本套件包含下列項目：

- 具有列出 IAC 檔案的資訊清單 YAML 檔案的基礎結構即程式碼 (I AC) 檔案。
- IAC 檔案輸入參數定義的結構描述 YAML 檔案。

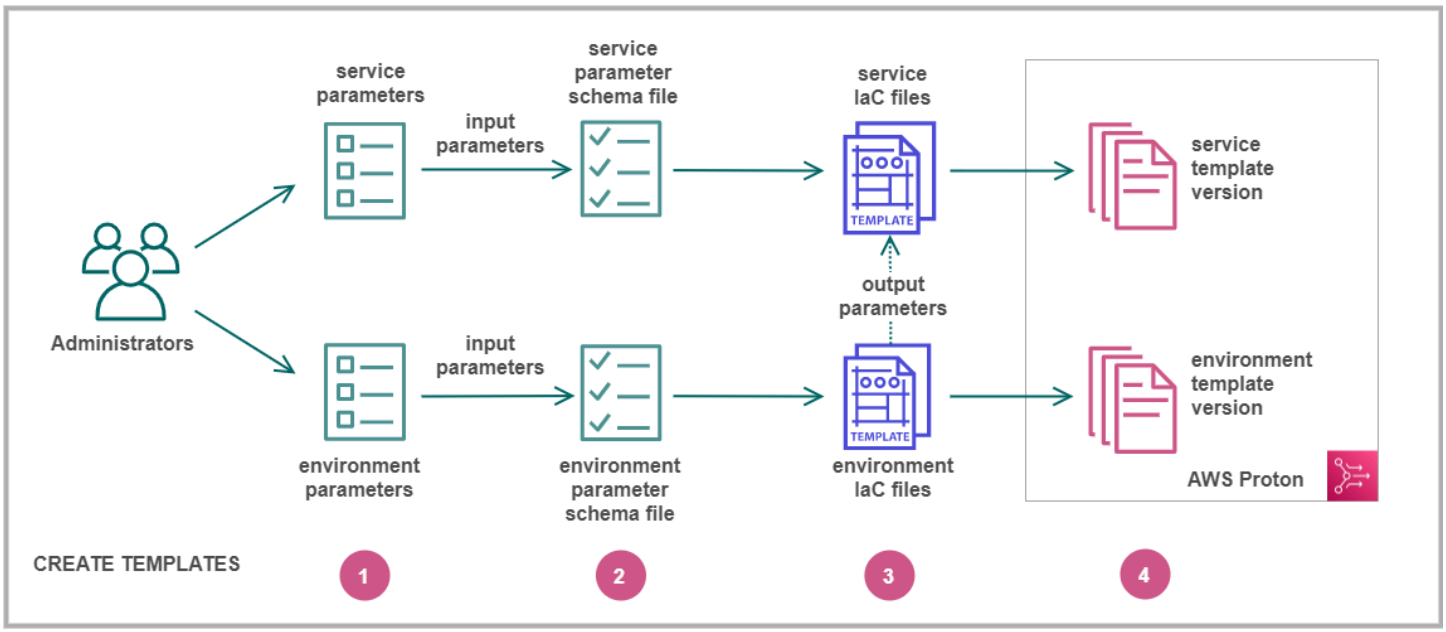
CloudFormation 環境範本套件包含一個 IaC 檔案。

CloudFormation 服務範本套件包含一個用於服務執行個體定義的 IaC 檔案，以及管線定義的另一個選用 IaC 檔案。

地形環境和服務模板包可以每個包含多個 IaC 文件。

AWS Proton 需要輸入參數結構描述檔案。當您使用建立 IaC 檔案時，您可 AWS CloudFormation 以使用 Jinja 語法來參考您的輸入參數。 AWS Proton 提供參數命名空間，您可以用來參考 IaC 檔案中的參數。

下圖顯示您可以為其建立範本所採取的步驟範例 AWS Proton。



1 識別輸入參數。

2 建立結構定義檔案以定義輸入參數。

3 創建引用您的輸入參數的 IaC 文件。您可以參考環境 IaC 文件輸出作為輸入為您的服務 IaC 文件。

4 註冊一個模板版本， AWS Proton 並上傳您的模板包。

AWS Proton 參數

您可以在基礎結構中定義和使用參數作為代碼（IaC）文件，以使其靈活且可重複使用。您可以透過參照參數命名空間中的參數名稱來讀取 IaC 檔案中的 AWS Proton 參數值。 AWS Proton 將參數值注入到它在資源佈建期間產生的轉譯 IaC 檔案中。要處理 AWS CloudFormation IaC 參數，請 AWS Proton 使用神社。若要處理 Terraform IAC 參數，請 AWS Proton 產生 Terraform 參數值檔案，並依賴 HCL 內建的參數化能力。

使用CodeBuild佈建， AWS Proton 產生您的程式碼可以匯入的輸入檔案。該文件是 JSON 或 HCL 文件，具體取決於模板的清單中的屬性。如需詳細資訊，請參閱 the section called “CodeBuild 佈建參數”。

您可以參考環境、服務和元件 IaC 檔案中的參數，或具有下列需求的佈建程式碼：

- 每個參數名稱的長度不超過 100 個字元。
- 參數命名空間和資源名稱組合的長度不超過資源名稱的字元限制。

AWS Proton 如果超過這些配額，佈建會失敗。

參數類型

下列參數類型可供您在 AWS Proton IaC 檔案中參考：

輸入參數

環境和服務執行個體可以採用您在與環境或服務範本建立關聯之綱要檔案中定義的輸入參數。您可以在資源的 IaC 檔案中參照資源的輸入參數。組件 IaC 文件可以參考該組件所附加到的服務實例的輸入參數。

AWS Proton 根據結構描述檔檢查輸入參數名稱，並將它們與 IaC 檔案中參照的參數進行比對，以便在資源佈建期間插入您在規格檔案中提供的輸入值。

輸出參數

您可以在任何 IaC 文件中定義輸出。例如，輸出可以是範本規定的其中一個資源的名稱、ID 或 ARN，也可以是透過範本輸入之一來傳遞的方式。您可以在其他資源的 IaC 文件中引用這些輸出。

在 CloudFormation IaC 檔案中，定義`Outputs:`區塊中的輸出參數。在地形 IaC 檔案中，使用陳述式定義每個輸出參數。`output`

資源參數

AWS Proton 自動建立 AWS Proton 資源參數。這些參數會公開 AWS Proton 資源物件的屬性。資源參數的範例為`environment.name`。

在 IaC 文件中使用 AWS Proton 參數

若要讀取 IaC 檔案中的參數值，請參考參數名稱空間中的 AWS Proton 參數名稱。對於 AWS CloudFormation IaC 文件，您可以使用 Jinja 語法，並用對大括號和引號圍住參數。

下表顯示每種受支援範本語言的參考語法，以及範例。

模板語言	語法	範例：名為「VPC」的環境輸入
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
地形	`var.parameter-name`	`var.environment.inputs.VPC` 產生的地形變數定義

> **Note**
>
> 如果您在 IaC 文件中使用CloudFormation 動態參數，則必須將其轉義以防止 Jinja 誤解錯誤。如需更多資訊，請參閱 AWS Proton 疑難排解

下表列出所有 AWS Proton 資源參數的命名空間名稱。每個範本檔案類型都可以使用不同的參數命名空間子集。

模板文件	參數類型	參數名稱	描述
Environment	資源	`environment.` **`name`**	環境名稱
	input	`environment.inputs.` *`input-name`*	結構描述定義的環境輸入

模板文件	參數類型	參數名稱	描述
服務	資源	environment. **name** environment. **account_id**	環境名稱和 AWS 帳戶 ID
	output	environment.outputs. *output-name*	環境 IAC 文件輸出
	資源	service.**branch_name** service.**name** service.**repository_connection_arn** service.**repository_id**	服務名稱與程式碼儲存庫
	資源	service_instance. **name**	服務實例名稱
	input	service_instance.inputs. *input-name*	結構定義的服務執行個體輸入
	資源	service_instance.components.default. **name**	貼附的預設元件名稱
	output	service_instance.components.default.outputs. *output-name*	附加的預設元件 IaC 檔案輸出
管道	資源	service_instance.environment. **name** service_instance.environment. **account_id**	服務執行個體環境名稱和 AWS 帳戶 ID
	output	service_instance.environment.outputs. *output-name*	服務實例環境 IaC 文件輸出
	input	pipeline.inputs. *input-name*	結構描述定義的管線輸入

模板文件	參數類型	參數名稱	描述
	資源	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	服務名稱與程式碼儲存庫
	input	`service_instance.inputs.` *`input-name`*	結構定義的服務執行個體輸入
	採集	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	您可以循環瀏覽的服務執行個體集合
元件	資源	`environment.` **`name`** `environment.` **`account_id`**	環境名稱和 AWS 帳戶帳號 ID
	output	`environment.outputs.` *`output-name`*	環境 IAC 文件輸出
	資源	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	服務名稱與程式碼儲存庫 (附加元件)
	資源	`service_instance.` **`name`**	服務執行個體名稱 (附加元件)
	input	`service_instance.inputs.` *`input-name`*	結構描述定義的服務執行個體輸入 (附加元件)
	資源	`component.` **`name`**	元件名稱

如需詳細資訊和範例，請參閱有關不同資源類型和範本語言的 IaC 範本檔案中參數的子主題。

主題

環境 CloudFormation IaC 文件參數的詳細信息和示例

您可以將環境基礎結構中的參數定義為程式碼 (IaC) 檔案。如需參數、參數類型、 AWS Proton 參數命名空間以及如何在 IaC 檔案中使用參數的詳細說明，請參閱the section called “參數”。

定義環境參數

您可以為環境 IaC 檔案定義輸入和輸出參數。

- 輸入參數 — 定義結構描述檔案中的環境輸入參數。

  下列清單包含典型使用案例的環境輸入參數範例。

  - VPC CIDR 值
  - 負載平衡器設定
  - 資料庫設定
  - 健康狀態檢查逾時

  身為管理員，您可以在建立環境時提供輸入參數的值：

  - 使用主控台填寫提供的結構描述式表 AWS Proton 單。
  - 使用 CLI 提供包含值的規格。
- 輸出參數 — 定義環境 IaC 檔案中的環境輸出。然後，您可以在其他資源的 IaC 文件中引用這些輸出。

讀取環境 IaC 檔案中的參數值

您可以在環境 IaC 文件中讀取與環境相關的參數。您可以在參數命名空間中參照參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 — 透過參考讀取環境輸入值`environment.inputs.`*`input-name`*。
- 資源參數 — 透過參考名稱來讀取 AWS Proton 資源參數，例如`environment.name`。

> **Note**
>
> 環境 IaC 檔案沒有其他資源的輸出參數可用。

示例環境和服務 IaC 文件參數

下面的例子演示了環境 IaC 文件中的參數定義和引用。然後，該示例顯示了如何在服務 IaC 文件中引用環境 IaC 文件中定義的環境輸出參數。

Example 環境 CloudFormation IAC 文件

請注意此範例中的下列事項：

- 命`environment.inputs.`名空間是指環境輸入參數。
- Amazon EC2 Systems Manager（SSM）參數`StoreInputValue`將環境輸入串聯起來。
- `MyEnvParameterValue`輸出會公開與輸出參數相同的輸入參數串連。另外三個輸出參數也會分別公開輸入參數。
- 六個額外的輸出參數公開了環境佈建的資源。

```
Resources:
  StoreInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ environment.inputs.my_sample_input }}
 {{ environment.inputs.my_other_sample_input}}
 {{ environment.inputs.another_optional_input }}"
              # input parameter references
```

```
# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'environment.outputs' namespace, for example,
 service_instance.environment.outputs.ClusterName.
Outputs:
  MyEnvParameterValue:                                    # output definition
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                     # output definition
    Value: "{{ environment.inputs.my_sample_input }}"         #   input parameter
 reference
  MyOtherSampleInputValue:                                # output definition
    Value: "{{ environment.inputs.my_other_sample_input }}"   #   input parameter
 reference
  AnotherOptionalInputValue:                              # output definition
    Value: "{{ environment.inputs.another_optional_input }}"  #   input parameter
 reference
  ClusterName:                                            # output definition
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'                              #   provisioned resource
  ECSTaskExecutionRole:                                   # output definition
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'             #   provisioned resource
  VpcId:                                                  # output definition
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'                                     #   provisioned resource
  PublicSubnetOne:                                        # output definition
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'                         #   provisioned resource
  PublicSubnetTwo:                                        # output definition
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'                         #   provisioned resource
  ContainerSecurityGroup:                                 # output definition
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'                  #   provisioned resource
```

Example 服務 CloudFormation IAC 文件

命environment.outputs.名空間是指環境 IaC 文件的環境輸出。例如，名稱會environment.outputs.ClusterName讀取ClusterName環境輸出參數的值。

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
```

```
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}' # resource parameter
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
 parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}'  # output
 reference to an environment infrastructure code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'  # resource parameter
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
```

```
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}' # resource parameter
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}' # resource parameter

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'  # resource parameter
      Cluster: '{{environment.outputs.ClusterName}}' # output reference to an
 environment infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'# input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output reference to an
 environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}' # output reference to an
 environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}' # output reference to an
 environment infrastructure as code file
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'  # resource parameter
          ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          TargetGroupArn: !Ref 'TargetGroup'
[...]
```

服務 CloudFormation IaC 文件參數的詳細信息和示例

您可以將服務和管線基礎結構中的參數定義為程式碼 (IaC) 檔案，並參考。如需參數、參數類型、AWS Proton 參數命名空間以及如何在 IaC 檔案中使用參數的詳細說明，請參閱the section called “參數”。

定義服務參數

您可以為服務 IaC 檔案定義輸入和輸出參數。

- 輸入參數 — 定義結構描述檔案中的服務執行個體輸入參數。

  下列清單包含典型使用案例的服務輸入參數範例。

  - 連線埠
  - 任務大小
  - 映像
  - 所需計數
  - 泊塢文件
  - 單元測試命令

  您可以在建立服務時提供輸入參數的值：

  - 使用主控台填寫提供的結構描述式表 AWS Proton 單。
  - 使用 CLI 提供包含值的規格。
- 輸出參數 — 在服務 IaC 檔案中定義服務執行個體輸出。然後，您可以在其他資源的 IaC 文件中引用這些輸出。

讀取服務 IaC 檔案中的參數值

您可以讀取與服務 IaC 文件中的服務和其他資源相關的參數。您可以在參數命名空間中參照參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 — 透過參照來讀取服務執行個體輸入值`service_instance.inputs.`*`input-name`*。
- 資源參數 — 透過參照`service.name`、`service_instance.name`和等名稱來讀取 AWS Proton 資源參數`environment.name`。
- 輸出參數 — 透過參照`environment.outputs.`*`output-name`*或來讀取其他資源的輸出`service_instance.components.default.outputs.`*`output-name`*。

具有參數的示例服務 IaC 文件

下列範例是服務 CloudFormation IaC 檔案中的程式碼片段。命`environment.outputs.`名空間是指環境 IaC 文件的輸出。`service_instance.inputs.`命名空間是指服務執行個體輸入參數。該`service_instance.name`屬性是指資 AWS Proton 源參數。

```
Resources:
  StoreServiceInstanceInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ service.name }} {{ service_instance.name }}
 {{ service_instance.inputs.my_sample_service_instance_required_input }}
 {{ service_instance.inputs.my_sample_service_instance_optional_input }}
 {{ environment.outputs.MySampleInputValue }}
 {{ environment.outputs.MyOtherSampleInputValue }}"
              #  resource parameter references               # input parameter
 references
                                           # output references to an environment
 infrastructure as code file
Outputs:
  MyServiceInstanceParameter:                                                    #
 output definition
    Value: !Ref StoreServiceInstanceInputValue
  MyServiceInstanceRequiredInputValue:                                           #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_required_input }}"  #
 input parameter reference
  MyServiceInstanceOptionalInputValue:                                           #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_optional_input }}"  #
 input parameter reference
  MyServiceInstancesEnvironmentSampleOutputValue:                                #
 output definition
    Value: "{{ environment.outputs.MySampleInputValue }}"                        #
 output reference to an environment IaC file
  MyServiceInstancesEnvironmentOtherSampleOutputValue:                           #
 output definition
    Value: "{{ environment.outputs.MyOtherSampleInputValue }}"                   #
 output reference to an environment IaC file
```

組件 CloudFormation IaC 文件參數的詳細信息和示例

您可以將元件基礎結構中的參數定義並參考為程式碼 (IaC) 檔案。如需參數、參數類型、 AWS Proton 參數命名空間以及如何在 IaC 檔案中使用參數的詳細說明，請參閱the section called “參數”。如需元件的詳細資訊，請參閱元件。

定義元件輸出參數

您可以在元件 IaC 檔案中定義輸出參數。然後，您可以在服務 IaC 文件中引用這些輸出。

> **Note**
>
> 您無法定義元件 IaC 檔案的輸入。連接的組件可以從它們所連接的服務實例獲取輸入。分離的組件沒有輸入。

讀取元件 IaC 檔案中的參數值

您可以在元件 IaC 檔案中讀取與元件和其他資源相關的參數。您可以在參數命名空間中參照參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 — 透過參照來讀取附加的服務執行個體輸入值`service_instance.inputs.`*`input-name`*。
- 資源參數 — 藉由參照名稱 (例如`component.name`、`service.name`、`service_instance.name`和) 來讀取 AWS Proton 資源參數`environment.name`。
- 輸出參數 — 透過參考讀取環境輸出`environment.outputs.`*`output-name`*。

具有參數的示例組件和服務 IaC 文件

下列範例顯示佈建 Amazon Simple Storage Service (Amazon S3) 貯體和相關存取政策的元件，並將兩個資源的 Amazon 資源名稱 (ARN) 公開為元件輸出。服務 IaC 範本會將元件輸出新增為 Amazon 彈性容器服務 (Amazon ECS) 任務的容器環境變數，讓輸出可供在容器中執行的程式碼使用，並將儲存貯體存取政策新增至任務的角色。值區名稱是以環境、服務、服務執行個體和元件的名稱為基礎，這表示儲存貯體與元件範本的特定執行個體結合，延伸特定服務執行個體。開發人員可以根據此元件範本建立多個自訂元件，以針對不同的服務執行個體和功能需求佈建 Amazon S3 儲存貯體。

此範例顯示如何使用 Jinja `{{ ... }}` 語法來參照服務 IaC 檔案中的元件和其他資源參數。
只有當元件附加至服務執行個體時，您才可以使用`{% if ... %}`陳述式來新增陳述式區塊。

關proton_cfn_*鍵字是可用來清理和格式化輸出參數值的篩選器。如需篩選條件的詳細資訊，請參閱the section called “CloudFormation 參數篩選”。

身為系統管理員，您撰寫服務 IaC 範本檔案。

Example 服務 CloudFormation IaC 文件使用一個組件

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

身為開發人員，您撰寫元件 IaC 範本檔案。

Example 組件 CloudFormation IAC 文件

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

為您的服務執行個體 AWS Proton 轉譯 AWS CloudFormation 範本，並以實際值取代所有參數時，範本可能看起來像下列檔案。

Example 服務實例 CloudFormation 呈現的 IaC 文件

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
```

```
          Environment:
            - Name: BucketName
              Value: arn:aws:s3:us-
east-1:123456789012:environment_name-service_name-service_instance_name-component_name
            - Name: BucketAccessPolicyArn
              Value: arn:aws:iam::123456789012:policy/cfn-generated-policy-name
  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

CloudFormation IaC 檔案的參數篩選器

當您在 AWS CloudFormation IaC 文件中引用AWS Proton 參數時，可以使用稱為過濾器的 Jinja 修飾符來驗證，過濾和格式化參數值，然後將參數值插入渲染的模板中。篩選器驗證在參照元件輸出參數時特別有用，因為元件建立和附件是由開發人員完成的，而在服務執行個體範本中使用元件輸出的管理員可能想要驗證其存在性和有效性。但是，您可以在任何 Jinja IaC 文件中使用過濾器。

以下各節說明和定義可用的參數篩選器，並提供範例。 AWS Proton 定義了大部分的篩選器。該`default`過濾器是 Jinja 內置過濾器。

格式化 Amazon ECS 任務的環境屬性

宣言

```
dict # proton_cfn_ecs_task_definition_formatted_env_vars (raw: boolean = True) # YAML
 list of dicts
```

Description

此篩選器會格式化 Amazon 彈性容器服務 (Amazon ECS) 任務定義ContainerDefinition區段中環境屬性中要使用的輸出清單。

設定raw為False以同時驗證參數值。在這種情況下，需要該值才能匹配正則表達式^[a-zA-Z0-9_-]*$。如果該值失敗此驗證，則模板渲染失敗。

範例

使用以下自定義組件模板：

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

以下服務模板：

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            {{ service_instance.components.default.outputs
              | proton_cfn_ecs_task_definition_formatted_env_vars }}
```

呈現的服務模板如下：

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
```

```
          # ...
          Environment:
            - Name: Output1
              Value: hello
            - Name: Output2
              Value: world
```

格式化 Lambda 函數的環境屬性

宣言

```
dict # proton_cfn_lambda_function_formatted_env_vars (raw: boolean = True) # YAML dict
```

Description

此篩選器會格式化 AWS Lambda 函數定義Properties區段中 Envi stance 屬性中要使用的輸出清單。

設定raw為False以同時驗證參數值。在這種情況下，需要該值才能匹配正則表達式^[a-zA-Z0-9_-]*$。如果該值失敗此驗證，則模板渲染失敗。

範例

使用以下自定義組件模板：

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

以下服務模板：

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
```

```
        Variables:
          {{ service_instance.components.default.outputs
            | proton_cfn_lambda_function_formatted_env_vars }}
```

呈現的服務模板如下：

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          Output1: hello
          Output2: world
```

擷取要包含在 IAM 角色中的 IAM 政策 ARN

宣言

```
dict # proton_cfn_iam_policy_arns # YAML list
```

Description

此篩選器會格式化要在 AWS Identity and Access Management (IAM) 角色定義`Properties`區段中ManagedPolicyArns 屬性中使用的輸出清單。篩選器會使用規則運算式`^arn:[a-zA-Z-]+:iam::\d{12}:policy/`，從輸出參數清單中擷取有效的 IAM 政策 ARN。您可以使用此篩選器，將輸出參數值中的政策附加到服務範本中的 IAM 角色定義。

範例

使用以下自定義組件模板：

```
Resources:
  # ...
  ExamplePolicy1:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
  ExamplePolicy2:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

```
  # ...

Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
  PolicyArn1:
    Description: "ARN of policy 1"
    Value: !Ref ExamplePolicy1
  PolicyArn2:
    Description: "ARN of policy 2"
    Value: !Ref ExamplePolicy2
```

以下服務模板：

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

呈現的服務模板如下：

```
Resources:

  # ...
```

```
  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-1
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-2

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

清理屬性值

宣言

```
string # proton_cfn_sanitize # string
```

Description

這是一個通用的過濾器。使用它來驗證參數值的安全性。篩選器會驗證值是否符合規則運算式，^[a-zA-Z0-9_-]*$或是有效的 Amazon 資源名稱 (ARN)。如果該值失敗此驗證，則模板渲染失敗。

範例

使用以下自定義組件模板：

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example of valid output"
    Value: "This-is_valid_37"
  Output2:
    Description: "Example incorrect output"
    Value: "this::is::incorrect"
  SomeArn:
    Description: "Example ARN"
    Value: arn:aws:some-service::123456789012:some-resource/resource-name
```

- 服務範本中的下列參考資料：

```
# ...
  {{ service_instance.components.default.outputs.Output1
    | proton_cfn_sanitize }}
```

呈現如下：

```
# ...
  This-is_valid_37
```

- 服務範本中的下列參考資料：

```
# ...
  {{ service_instance.components.default.outputs.Output2
    | proton_cfn_sanitize }}
```

具有以下渲染錯誤的結果：

```
Illegal character(s) detected in "this::is::incorrect". Must match regex ^[a-zA-
Z0-9_-]*$ or be a valid ARN
```

- 服務範本中的下列參考資料：

```
# ...
  {{ service_instance.components.default.outputs.SomeArn
    | proton_cfn_sanitize }}
```

呈現如下：

```
# ...
  arn:aws:some-service::123456789012:some-resource/resource-name
```

為不存在的參照提供預設值

Description

當命名空間參考不存在時，`default`篩選器會提供預設值。使用它來編寫強大的模板，即使您引用的參數丟失也可以渲染而不會失敗。

範例

如果服務實例沒有附加的直接定義（默認）組件，或者連接的組件沒有名為的輸出，則服務模板中的以下引用會導致模板渲染失敗`test`。

```
# ...
  {{ service_instance.components.default.outputs.test }}
```

若要避免此問題，請新增`default`篩選器。

```
# ...
  {{ service_instance.components.default.outputs.test | default("[optional-value]") }}
```

CodeBuild 佈建參數詳細資料和範例

您可以在範本中為 CodeBuild基礎的 AWS Proton 資源定義參數，並在佈建程式碼中參考這些參數。如需參數、參數類型、 AWS Proton 參數命名空間以及如何在 IaC 檔案中使用參數的詳細說明，請參閱the section called “參數”。

> **Note**
>
> 您可以將 CodeBuild 佈建與環境和服務搭配使用。目前您無法以這種方式佈建元件。

輸入參數

當您建立 AWS Proton 資源 (例如環境或服務) 時，您可以為範本結構描述檔中定義的輸入參數提供值。當您建立的資源使用時CodeBuild佈建，會將這些輸入值 AWS Proton 轉譯為輸入檔案。您的佈建程式碼可以從此檔案匯入和取得參數值。

如需範本的 CodeBuild 範例，請參閱the section called “CodeBuild 捆綁”。如需資訊清單檔案的相關資訊，請參閱the section called “清單和包裝”。

下列範例是在服務執行個體佈建期間 CodeBuild產生的 JSON 輸入檔案。

範例：使用 AWS CDK 與 CodeBuild 佈建

```
{
  "service_instance": {
    "name": "my-service-staging",
```

```
    "inputs": {
      "port": "8080",
      "task_size": "medium"
    }
  },
  "service": {
    "name": "my-service"
  },
  "environment": {
    "account_id": "123456789012",
    "name": "my-env-staging",
    "outputs": {
      "vpc-id": "hdh2323423"
    }
  }
}
```

輸出參數

若要將資源佈建輸出傳回 AWS Proton，您的佈建程式碼可以產生名為的 JSON 檔案，其中`proton-outputs.json`包含範本結構描述檔中定義的輸出參數值。例如，`cdk deploy`命令具有`--outputs-file`引數，可指示產生具有佈建輸出的 JSON 檔案。 AWS CDK 如果您的資源使用 AWS CDK，請在 CodeBuild 範本資訊清單中指定下列命令：

```
aws proton notify-resource-deployment-status-change
```

AWS Proton 查找這個 JSON 文件。如果在佈建程式碼順利完成後存在該檔案，請從中 AWS Proton 讀取輸出參數值。

地形基礎架構代碼（IaC）文件參數詳細信息和示例

您可以在模板包中的`variable.tf`文件中包含 Terraform 輸入變量。您也可以建立結構描述來建立 AWS Proton 受管理的變數。 AWS Proton `.tf files`從您的架構檔案建立變數。如需詳細資訊，請參閱 the section called “地形 IAC 文件”。

若要在基礎結構中參考結構描述定義的 AWS Proton 變數`.tf files`，請使用 Terraform IaC 資料 AWS Proton 表的參數和命名空間中顯示的命名空間。例如，您可以使用 `var.environment.inputs.vpc_cidr`. 在引號內，用單括號括住這些變數，並在第一個大括號前加上貨幣符號 (例如，“`${var.environment.inputs.vpc_cidr}`”)。

下列範例會示範如何使用命名空間在環境`.tf file`中包含 AWS Proton 參數。

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

AWS Proton 基礎架構即程式碼檔

範本服務包的主要部分是基礎結構即程式碼 (IaC) 檔案，可定義您要佈建的基礎結構資源和屬性。AWS CloudFormation 和其他基礎結構作為代碼引擎使用這些類型的文件來佈建基礎結構資源。

> **Note**
>
> 一個 IaC 文件也可以獨立地使用模板包，作為直接輸入到直接定義的組件。如需元件的詳細資訊，請參閱元件。

AWS Proton 目前支持兩種類型的 IaC 文件：

- CloudFormation檔案 — 用於AWS管理佈建。 AWS Proton 在 CloudFormation 範本檔案格式之上使用 Jinja 進行參數化。
- 地形 HCL 檔案 — 用於自我管理佈建。HCL 本地支持參數化。

您無法使用佈建方法的組合佈建 AWS Proton 資源。您必須使用其中一個。您無法將 AWS受管理的佈建服務部署到自我管理的佈建環境，反之亦然。

如需詳細資訊，請參閱 the section called “佈建方法”、環境、服務 和 元件。

AWS CloudFormation 合家歡的文件

了解如何將 AWS CloudFormation 基礎結構當做程式碼檔案使用 AWS Proton。 AWS CloudFormation 是一種基礎結構即程式碼 (IaC) 服務，可協助您建立 AWS 資源的模型和設定。您可以在範本中定義基礎結構資源，並在範本檔案格式之上 CloudFormation 使用 Jinja 進行參數化。 AWS Proton 展開參數並呈現完整的 CloudFormation 模板。 CloudFormation 將定義的資源規定為 CloudFormation 堆棧。若要取得更多資訊，請參閱《 AWS CloudFormation 使用指南》 AWS CloudFormation中的「內容」。

AWS Proton 支援 CloudFormation I AWS aC 的管理佈建。

從您自己的現有基礎結構作為代碼文件開始

您可以將自己現有的基礎結構調整為程式碼 (IaC) 檔案，以便搭配 AWS Proton使用。

下列 AWS CloudFormation 範例 [範例 1] 和 [範例 2] 代表您自己現有的 CloudFormation IaC 檔案。CloudFormation 可以使用這些文件創建兩個不同的 CloudFormation堆棧。

在範例 1 中， CloudFormation IaC 檔案會設定為佈建要在容器應用程式之間共用的基礎結構。在此範例中，會新增輸入參數，以便您可以使用相同的 IaC 檔案來建立多組佈建的基礎結構。每個集合可以有不同的名稱以及一組不同的 VPC 和子網路 CIDR 值。身為系統管理員或開發人員，您可以在使用 IaC 檔案佈建基礎結構資源時，提供這些參數的 CloudFormation值。為了您的方便起見，這些輸入參數標有註釋，並在示例中多次引用。輸出在模板的末尾定義。它們可以在其他 CloudFormation IaC 文件中引用。

在範例 2 中， CloudFormation IaC 檔案會設定為將應用程式部署至範例 1 所佈建的基礎結構。為了您的方便起見，這些參數被評論。

範例 1： CloudFormation IAC 檔案

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery namespaces.
Parameters:
   VpcCIDR:       # input parameter
        Description: CIDR for VPC
        Type: String
        Default: "10.0.0.0/16"
   SubnetOneCIDR: # input parameter
        Description: CIDR for SubnetOne
        Type: String
        Default: "10.0.0.0/24"
   SubnetTwoCIDR: # input parameters
        Description: CIDR for SubnetTwo
        Type: String
        Default: "10.0.1.0/24"
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock:
        Ref: 'VpcCIDR'

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
         Ref: 'SubnetOneCIDR'
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 1
         - Fn::GetAZs: {Ref: 'AWS::Region'}
```

```
      VpcId: !Ref 'VPC'
      CidrBlock:
        Ref: 'SubnetTwoCIDR'
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster

  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
```

```
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values will be available to other templates to use.
Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSCluster"
  ECSTaskExecutionRole:                                  # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSTaskExecutionRole"
  VpcId:                                                 # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-VPC"
  PublicSubnetOne:                                       # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
    Export:
```

```
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetOne"
  PublicSubnetTwo:                                          # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetTwo"
  ContainerSecurityGroup:                                   # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ContainerSecurityGroup"
```

範例 2： CloudFormation IAC 檔案

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Parameters:
    ContainerPortInput:  # input parameter
        Description: The port to route traffic to
        Type: Number
        Default: 80
    TaskCountInput:      # input parameter
        Description: The default number of Fargate tasks you want running
        Type: Number
        Default: 1
    TaskSizeInput:       # input parameter
        Description: The size of the task you want to run
        Type: String
        Default: x-small
    ContainerImageInput: # input parameter
        Description: The name/url of the container image
        Type: String
        Default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
    TaskNameInput:       # input parameter
        Description: Name for your task
        Type: String
        Default: "my-fargate-instance"
    StackName:           # input parameter
        Description: Name of the environment stack to deploy to
```

```
        Type: String
        Default: "my-fargate-environment"
Mappings:
  TaskSizeMap:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName:
        Ref: 'TaskNameInput' # input parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref 'TaskNameInput'
      Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
      Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn:
        Fn::ImportValue:
          !Sub "${StackName}-ECSTaskExecutionRole"    # output parameter from another
 CloudFormation template
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: !Ref 'TaskNameInput'
```

```
 # The service_instance. The service is a resource which allows you to run multiple
 # copies of a type of task, and gather up their logs and metrics, as well
 # as monitor the number of running tasks and replace any that have crashed
 Service:
   Type: AWS::ECS::Service
   DependsOn: LoadBalancerRule
   Properties:
     ServiceName: !Ref 'TaskNameInput'
     Cluster:
       Fn::ImportValue:
         !Sub "${StackName}-ECSCluster"  # output parameter from another
CloudFormation template
     LaunchType: FARGATE
     DeploymentConfiguration:
       MaximumPercent: 200
       MinimumHealthyPercent: 75
     DesiredCount: !Ref 'TaskCountInput'
     NetworkConfiguration:
       AwsvpcConfiguration:
         AssignPublicIp: ENABLED
         SecurityGroups:
           - Fn::ImportValue:
               !Sub "${StackName}-ContainerSecurityGroup"    # output parameter from
another CloudFormation template
         Subnets:
           - Fn::ImportValue:r CloudFormation template
     TaskRoleArn: !Ref "AWS::NoValue"
     ContainerDefinitions:
       - Name: !Ref 'TaskNameInput'
         Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
         Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
         Image: !Ref 'ContainerImageInput'             # input parameter
         PortMappings:
           - ContainerPort: !Ref 'ContainerPortInput'  # input parameter

         LogConfiguration:
           LogDriver: 'awslogs'
           Options:
             awslogs-group: !Ref 'TaskNameInput'
               !Sub "${StackName}-PublicSubnetOne"    # output parameter from another
CloudFormation template
           - Fn::ImportValue:
```

```
                !Sub "${StackName}-PublicSubnetTwo"    # output parameter from another
CloudFormation template
     TaskDefinition: !Ref 'TaskDefinition'
     LoadBalancers:
       - ContainerName: !Ref 'TaskNameInput'
         ContainerPort: !Ref 'ContainerPortInput'  # input parameter
         TargetGroupArn: !Ref 'TargetGroup'

 # A target group. This is used for keeping track of all the tasks, and
 # what IP addresses / port numbers they have. You can query it yourself,
 # to use the addresses yourself, but most often this target group is just
 # connected to an application load balancer, or network load balancer, so
 # it can automatically distribute traffic across all the targets.
 TargetGroup:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     HealthCheckIntervalSeconds: 6
     HealthCheckPath: /
     HealthCheckProtocol: HTTP
     HealthCheckTimeoutSeconds: 5
     HealthyThresholdCount: 2
     TargetType: ip
     Name: !Ref 'TaskNameInput'
     Port: !Ref 'ContainerPortInput'
     Protocol: HTTP
     UnhealthyThresholdCount: 2
     VpcId:
       Fn::ImportValue:
         !Sub "${StackName}-VPC"    # output parameter from another CloudFormation
template

 # Create a rule on the load balancer for routing traffic to the target group
 LoadBalancerRule:
   Type: AWS::ElasticLoadBalancingV2::ListenerRule
   Properties:
     Actions:
       - TargetGroupArn: !Ref 'TargetGroup'
         Type: 'forward'
     Conditions:
       - Field: path-pattern
         Values:
           - '*'
     ListenerArn: !Ref PublicLoadBalancerListener
     Priority: 1
```

```
  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"  # output parameter from another
 CloudFormation template
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
```

```
      AdjustmentType: 'ChangeInCapacity'
      StepAdjustments:
        - MetricIntervalUpperBound: 0
          ScalingAdjustment: -1
      MetricAggregationType: 'Average'
      Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60

  # Create alarms to trigger these policies
  LowCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
```

```
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - low-cpu
            - !Ref 'TaskNameInput'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "Low CPU utilization for service"
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 20
      ComparisonOperator: LessThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleDownPolicy

  HighCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - high-cpu
            - !Ref 'TaskNameInput'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
```

```
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId:
        Fn::ImportValue:
          !Sub "${StackName}-ContainerSecurityGroup"
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
```

```
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetOne"
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetTwo"
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
# These output values will be available to other templates to use.
Outputs:
  ServiceEndpoint:        # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

您可以調整這些檔案以配合使用 AWS Proton。

將您的基礎架構即程式碼帶入 AWS Proton

只要稍作修改，您就可以使用範例 1 做為基礎結構即程式碼 (IaC) 檔案，以供 AWS Proton 用來部署環境的環境範本套裝軟體 (如範例 3 所示)。

您可以使用 Jinja 語法來 CloudFormation 參考您在 Open API 架構檔案中定義的參數，而不是使用參數。這些輸入參數被註釋為了您的方便，並在 IaC 文件中多次引用。這樣， AWS Proton 可以審核和檢查參數值。它還可以將一個 IaC 文件中的輸出參數值匹配並插入到另一個 IaC 文件中的參數。

身為管理員，您可以將 AWS Proton `environment.inputs.`命名空間新增至輸入參數。當您在服務 IaC 檔案中參考環境 IaC 檔案輸出時，您可以將`environment.outputs.`命名空間新增至輸出 (例如，`environment.outputs.ClusterName`)。最後，你用花括號和引號圍住它們。

通過這些修改，您的 CloudFormation IaC 文件可以被 AWS Proton使用。

範例 3： AWS Proton 環境基礎結構作為程式碼檔案

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery prefixes.
Mappings:
  # The VPC and subnet configuration is passed in via the environment spec.
  SubnetConfig:
    VPC:
      CIDR: '{{ environment.inputs.vpc_cidr}}'        # input parameter
    PublicOne:
      CIDR: '{{ environment.inputs.subnet_one_cidr}}' # input parameter
    PublicTwo:
      CIDR: '{{ environment.inputs.subnet_two_cidr}}' # input parameter
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock: !FindInMap ['SubnetConfig', 'VPC', 'CIDR']

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicOne', 'CIDR']
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      AvailabilityZone:
         Fn::Select:
         - 1
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicTwo', 'CIDR']
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
```

```
  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'service_instance.environment.outputs.' namespace, for example,
 service_instance.environment.outputs.ClusterName.

Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:                                 # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:                                                # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
  PublicSubnetOne:                                      # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
  PublicSubnetTwo:                                      # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
```

```
  ContainerSecurityGroup:                                  # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
```

範例 1 和範例 3 中的 IaC 檔案會產生稍微不同的 CloudFormation 堆疊。參數在堆疊範本檔案中的顯示方式不同。範例 1 CloudFormation 堆疊樣板檔案會在堆疊樣板視圖中顯示參數標示 (關鍵字)。範例 3 AWS Proton CloudFormation 基礎結構堆疊範本檔案會顯示參數值。 AWS Proton 輸入參數不會出現在控制台 CloudFormation堆棧參數視圖中。

在範例 4 中， AWS Proton 服務 IaC 檔案對應於範例 2。

範例 4： AWS Proton 服務執行個體 IaC 檔案

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
```

```
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}'
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}' # output from an
environment infrastructure as code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}'
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}'

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'
      Cluster: '{{environment.outputs.ClusterName}}' # output from an environment
infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'       # input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
```

```
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output from an
environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}'        # output from an
environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}'
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'
          ContainerPort: '{{service_instance.inputs.port}}'
          TargetGroupArn: !Ref 'TargetGroup'

  # A target group. This is used for keeping track of all the tasks, and
  # what IP addresses / port numbers they have. You can query it yourself,
  # to use the addresses yourself, but most often this target group is just
  # connected to an application load balancer, or network load balancer, so
  # it can automatically distribute traffic across all the targets.
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Name: '{{service_instance.name}}'
      Port: '{{service_instance.inputs.port}}'
      Protocol: HTTP
      UnhealthyThresholdCount: 2
      VpcId: '{{environment.outputs.VpcId}}' # output from an environment
infrastructure as code file

  # Create a rule on the load balancer for routing traffic to the target group
  LoadBalancerRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      Conditions:
        - Field: path-pattern
          Values:
```

```
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}' # output from an environment
 infrastructure as code file
            - '{{service_instance.name}}'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
```

```
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60

  # Create alarms to trigger these policies
  LowCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
```

```
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - low-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "Low CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 20
      ComparisonOperator: LessThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleDownPolicy

  HighCpuUsageAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName:
        Fn::Join:
          - '-'
          - - high-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
```

```
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: '{{environment.outputs.ContainerSecurityGroup}}'
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: '{{environment.outputs.VpcId}}'
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1

  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
```

```
        # from the internet can reach the load balancer directly via the internet
 gateway
        - '{{environment.outputs.PublicSubnetOne}}'
        - '{{environment.outputs.PublicSubnetTwo}}'
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
Outputs:
  ServiceEndpoint:         # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

在範例 5 中， AWS Proton 管線 IaC 檔案會佈建管線基礎結構，以支援範例 4 所佈建的服務執行個體。

範例 5： AWS Proton 服務管線 IaC 檔案

```
Resources:
  ECRRepo:
    Type: AWS::ECR::Repository
    DeletionPolicy: Retain
  BuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: true
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: repo_name
```

```
          Type: PLAINTEXT
          Value: !Ref ECRRepo
        - Name: service_name
          Type: PLAINTEXT
          Value: '{{ service.name }}'    # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - PublishRole
          - Arn
      Source:
        BuildSpec:
          Fn::Join:
            - ""
            - - >-
                {
                  "version": "0.2",
                  "phases": {
                    "install": {
                      "runtime-versions": {
                        "docker": 18
                      },
                      "commands": [
                        "pip3 install --upgrade --user awscli",
                        "echo
 'f6bd1536a743ab170b35c94ed4c7c4479763356bd543af5d391122f4af852460  yq_linux_amd64' >
 yq_linux_amd64.sha",
                        "wget https://github.com/mikefarah/yq/releases/download/3.4.0/
yq_linux_amd64",
                        "sha256sum -c yq_linux_amd64.sha",
                        "mv yq_linux_amd64 /usr/bin/yq",
                        "chmod +x /usr/bin/yq"
                      ]
                    },
                    "pre_build": {
                      "commands": [
                        "cd $CODEBUILD_SRC_DIR",
                        "$(aws ecr get-login --no-include-email --region
 $AWS_DEFAULT_REGION)",
                        "{{ pipeline.inputs.unit_test_command }}",    # input parameter
                      ]
                    },
                    "build": {
                      "commands": [
                        "IMAGE_REPO_NAME=$repo_name",
```

```
                        "IMAGE_TAG=$CODEBUILD_BUILD_NUMBER",
                        "IMAGE_ID=
              - Ref: AWS::AccountId
              - >-
                .dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG",
                        "docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG -f
 {{ pipeline.inputs.dockerfile }} .",     # input parameter
                        "docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_ID;",
                        "docker push $IMAGE_ID"
                      ]
                    },
                    "post_build": {
                      "commands": [
                        "aws proton --region $AWS_DEFAULT_REGION get-service --name
 $service_name | jq -r .service.spec > service.yaml",
                        "yq w service.yaml 'instances[*].spec.image' \"$IMAGE_ID\" >
 rendered_service.yaml"
                      ]
                    }
                  },
                  "artifacts": {
                    "files": [
                      "rendered_service.yaml"
                    ]
                  }
                }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% for service_instance in service_instances %}
  Deploy{{loop.index}}Project:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: false
        Type: LINUX_CONTAINER
        EnvironmentVariables:
```

```
        - Name: service_name
          Type: PLAINTEXT
          Value:  '{{service.name}}'          # resource parameter
        - Name: service_instance_name
          Type: PLAINTEXT
          Value: '{{service_instance.name}}'  # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - DeploymentRole
          - Arn
      Source:
        BuildSpec: >-
          {
            "version": "0.2",
            "phases": {
              "build": {
                "commands": [
                  "pip3 install --upgrade --user awscli",
                  "aws proton --region $AWS_DEFAULT_REGION update-service-instance
 --deployment-type CURRENT_VERSION --name $service_instance_name --service-name
 $service_name --spec file://rendered_service.yaml",
                  "aws proton --region $AWS_DEFAULT_REGION wait service-instance-
deployed --name $service_instance_name --service-name $service_name"
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
```

```
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
```

```
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/
                - Ref: BuildProject
                - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
```

```
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole

  DeploymentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
```

```
            - logs:PutLogEvents
          Effect: Allow
          Resource:
            - Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":logs:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :log-group:/aws/codebuild/Deploy*Project*
            - Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":logs:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :log-group:/aws/codebuild/Deploy*Project:*
        - Action:
            - codebuild:CreateReportGroup
            - codebuild:CreateReport
            - codebuild:UpdateReport
            - codebuild:BatchPutTestCases
          Effect: Allow
          Resource:
            Fn::Join:
              - ""
              - - "arn:"
                - Ref: AWS::Partition
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/Deploy*Project
                - -*
        - Action:
            - proton:UpdateServiceInstance
            - proton:GetServiceInstance
          Effect: Allow
          Resource: "*"
        - Action:
```

```
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
```

```
            - kms:Enable*
            - kms:List*
            - kms:Put*
            - kms:Update*
            - kms:Revoke*
            - kms:Disable*
            - kms:Get*
            - kms:Delete*
            - kms:ScheduleKeyDeletion
            - kms:CancelKeyDeletion
            - kms:GenerateDataKey
            - kms:TagResource
            - kms:UntagResource
          Effect: Allow
          Principal:
            AWS:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
```

```
            Principal:
              AWS:
                Fn::GetAtt:
                  - PublishRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - PublishRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
        Version: "2012-10-17"
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineArtifactsBucket:
```

```
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'
      TargetKeyId:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
```

```
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
```

```
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'
                FullRepositoryId: '{{ service.repository_id }}'
                BranchName: '{{ service.branch_name }}'
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
```

```
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
```

```
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":iam::"
                  - Ref: AWS::AccountId
                  - :root
      Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
```

```
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"

                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
```

```
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
```

```
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
      Version: "2012-10-17"
    PolicyName: PublishRoleDefaultPolicy
    Roles:
      - Ref: PublishRole

DeploymentRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Action: sts:AssumeRole
          Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
      Version: "2012-10-17"
DeploymentRoleDefaultPolicy:
  Type: AWS::IAM::Policy
```

```
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/Deploy*Project
                  - -*
```

```
          - Action:
              - proton:UpdateServiceInstance
              - proton:GetServiceInstance
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
```

```
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
              - kms:Enable*
              - kms:List*
              - kms:Put*
              - kms:Update*
              - kms:Revoke*
              - kms:Disable*
              - kms:Get*
              - kms:Delete*
              - kms:ScheduleKeyDeletion
              - kms:CancelKeyDeletion
              - kms:GenerateDataKey
              - kms:TagResource
              - kms:UntagResource
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - PipelineRole
                  - Arn
            Resource: "*"
          - Action:
```

```
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
```

```
                - Arn
            Resource: "*"
        Version: "2012-10-17"
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineArtifactsBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'     # resource
parameter
      TargetKeyId:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
```

```
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
```

```
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'   # resource
parameter
                FullRepositoryId: '{{ service.repository_id }}'            # resource
parameter
                BranchName: '{{ service.branch_name }}'                    # resource
parameter
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
```

```
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'         # resource parameter
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
```

```
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"
```

CodeBuild 佈建範本套件

CodeBuild 隨著佈建，而不是使用 IaC 範本來呈現 IaC 檔案，並使用 IaC 佈建引擎執行它們， AWS Proton 只需執行您的殼層命令。為此，請在環境帳戶中為環境創 AWS Proton 建一個 AWS CodeBuild 項目，並啟動一項工作以為每次 AWS Proton 資源創建或更新運行命令。當您編寫範本服務包時，您會提供資訊清單，以指定基礎結構佈建和取消佈建命令，以及這些命令可能需要的任何程式、指令碼和

其他檔案。您的命令可以讀取 AWS Proton 提供的輸入，並負責佈建或取消佈建基礎結構，以及產生輸出值。

清單還指定了 AWS Proton 如何呈現代碼可以輸入並從中獲取輸入值的輸入文件。它可以被渲染成 JSON 或 HCL。如需輸入參數的更多資訊，請參閱the section called “CodeBuild 佈建參數”。如需資訊清單檔案的相關資訊，請參閱the section called “清單和包裝”。

> **Note**
>
> 您可以將 CodeBuild 佈建與環境和服務搭配使用。目前您無法以這種方式佈建元件。

範例：使用 AWS CDK 與 CodeBuild 佈建

作為使用 CodeBuild 佈建的範例，您可以包含使用佈建 (部署) 和取消佈建 (銷毀) AWS 資源的程式碼，以及安裝 CDK 並執行 CDK 程式碼的資訊清單。 AWS Cloud Development Kit (AWS CDK)

下列各節列出您可以包含在使用佈建環境的 CodeBuild 佈建範本套裝軟體中的範例檔案 AWS CDK。

清單檔案

下列資訊清單檔案會指定 CodeBuild 佈建，並包含安裝和使用輸出 AWS CDK、輸出檔案處理以及回報輸出所需的命令 AWS Proton。

Example 基礎設施/表現。

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
```

```
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

結構描述

下列結構描述檔案定義了環境的參數。您的 AWS CDK 程式碼可以在部署期間參考這些參數的值。

Example 模式/方案/亞姆

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "MyEnvironmentInputType"
  types:
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:
          type: string
          description: "Another sample input"
      required:
        - my_other_sample_input
```

AWS CDK 文件

下列檔案是 Node.js CDK 專案的範例。

Example 基礎結構/封裝.json

```
{
  "name": "ProtonEnvironment",
```

```
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironmente": "bin/ProtonEnvironment.js"
  },
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk"
  },
  "devDependencies": {
    "@types/jest": "^28.1.7",
    "@types/node": "18.7.6",
    "jest": "^28.1.3",
    "ts-jest": "^28.0.8",
    "aws-cdk": "2.37.1",
    "ts-node": "^10.9.1",
    "typescript": "~4.7.4"
  },
  "dependencies": {
    "aws-cdk-lib": "2.37.1",
    "constructs": "^10.1.77",
    "source-map-support": "^0.5.21"
  }
}
```

Example 基礎結構/設定.json

```
{
  "compilerOptions": {
    "target": "ES2018",
    "module": "commonjs",
    "lib": [
      "es2018"
    ],
    "declaration": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noImplicitThis": true,
    "alwaysStrict": true,
    "noUnusedLocals": false,
    "noUnusedParameters": false,
```

```
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": false,
    "inlineSourceMap": true,
    "inlineSources": true,
    "experimentalDecorators": true,
    "strictPropertyInitialization": false,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "typeRoots": [
      "./node_modules/@types"
    ]
  },
  "exclude": [
    "node_modules",
    "cdk.out"
  ]
}
```

Example 基礎設施

```
{
  "app": "npx ts-node --prefer-ts-exts bin/ProtonEnvironment.ts",
  "outputsFile": "proton-outputs.json",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
      "yarn.lock",
      "node_modules",
      "test"
    ]
  },
  "context": {
    "@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId": true,
    "@aws-cdk/core:stackRelativeExports": true,
    "@aws-cdk/aws-rds:lowercaseDbIdentifier": true,
```

```
    "@aws-cdk/aws-lambda:recognizeVersionProps": true,
    "@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2_2021": true,
    "@aws-cdk-containers/ecs-service-extensions:enableDefaultLogDriver": true,
    "@aws-cdk/aws-ec2:uniqueImdsv2TemplateName": true,
    "@aws-cdk/core:target-partitions": [
      "aws",
      "aws-cn"
    ]
  }
}
```

Example 基礎設施//.ts ProtonEnvironment

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { ProtonEnvironmentStack } from '../lib/ProtonEnvironmentStack';

const app = new cdk.App();
new ProtonEnvironmentStack(app, 'ProtonEnvironmentStack', {});
```

Example 基礎設施/庫/ProtonEnvironmentStack. TS

```
import { Stack, StackProps } from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import input from '../proton-inputs.json';

export class ProtonEnvironmentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, { ...props, stackName: process.env.STACK_NAME });

    const ssmParam = new ssm.StringParameter(this, "ssmParam", {
      stringValue: input.environment.inputs.my_sample_input,
      parameterName: `${process.env.STACK_NAME}-Param`,
      tier: ssm.ParameterTier.STANDARD
    })

    new cdk.CfnOutput(this, 'ssmParamOutput', {
      value: ssmParam.parameterName,
      description: 'The name of the ssm parameter',
      exportName: `${process.env.STACK_NAME}-Param`
```

```
    });
  }
}
```

渲染輸入文件

使用 CodeBuild以基礎的佈建範本建立環境時， AWS Proton 會使用您提供的輸入參數值轉譯輸入檔案。您的程式碼可以參考這些值。下面的文件是一個例子渲染的輸入文件。

Example 基礎結構/質子輸入.json

```
{
  "environment": {
    "name": "myenv",
    "inputs": {
      "my_sample_input": "10.0.0.0/16",
      "my_other_sample_input": "11.0.0.0/16"
    }
  }
}
```

地形 IAC 文件

了解如何使用. AWS Proton地形是一種廣泛使用的開源 IaC 引擎，由. HashiCorp Terraform 模組以 HCL 語言 HashiCorp開發，並支援多個後端基礎設施供應商，包括 Amazon Web Services。

AWS Proton 支援 Terraform IAC 的自我管理佈建。

有關響應提取請求並實現基礎結構佈建的佈建存儲庫的完整示例，請參閱上的 Terraform OpenSource GitHub 動作自動化模板。 AWS Proton GitHub

自我管理佈建如何與 Terraform IaC 範本套件組合檔案搭配使用：

1. 當您從 Terraform 範本組合包建立環境時，請使用主控台或`spec file`輸入參數 AWS Proton 編譯`.tf`檔案。
2. 它提出了一個拉取請求，將編譯的 IaC 文件合併到您已註冊的存儲庫中。 AWS Proton
3. 如果要求已核准，就 AWS Proton 會等待您提供的佈建狀態。
4. 如果要求遭到拒絕，則會取消環境建立。
5. 如果提取請求逾時，則環境建立不完成。

AWS Proton 與地形 IaC 的考慮因素：

- AWS Proton 不管理您的 Terraform 配置。
- 您必須使用註冊啟動設定存放庫 AWS Proton。 AWS Proton 在此儲存庫上提出提取要求。
- 您必須建立 CodeStar 連線才能 AWS Proton 與佈建存放庫連線。
- 若要從 AWS Proton 編譯的 IaC 檔案進行佈建，您必須回應提 AWS Proton 取要求。 AWS Proton 在環境和服務建立和更新動作之後發出提取要求。如需詳細資訊，請參閱 AWS Proton 環境 及 AWS Proton 服務。
- 若要從 AWS Proton 編譯的 IaC 檔案佈建管線，您必須建立 CI/CD 管線存放庫。
- 以提取要求為基礎 AWS Proton 的佈建自動化必須包含通知任何已佈建 AWS Proton 資源狀態變更的步驟。您可以使用 AWS Proton NotifyResourceDeploymentStatusChange API。
- 您無法將從 IaC 檔案建立的服務、管線和元件部署到從 Terraform CloudFormation IaC 檔案建立的環境中。
- 您無法將從 Terraform IaC 檔案建立的服務、管線和元件部署到從 IaC 檔案建立的環境中 CloudFormation 。

準備您的 Terraform IaC 檔案時 AWS Proton，您會將命名空間附加至輸入變數，如下列範例所示。如需詳細資訊，請參閱參數。

範例 1： AWS Proton 環境地形 IaC 檔案

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
```

```
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

編譯基礎架構即程式

當您建立環境或服務時，會使用主控台或spec file輸入將基礎結構 AWS Proton 編譯為程式碼檔案。它會為您的輸入創建proton.*resource-type*.variables.tf和proton.auto.tfvars.json文件，可以通過 Terraform 中使用，如下面的例子。這些檔案位於與環境或服務執行個體名稱相符的資料夾中的指定儲存庫中。

此範例顯示如何在變數定義和變數值中 AWS Proton 包含標籤，以及如何將這些 AWS Proton 標籤傳播至已佈建的資源。如需詳細資訊，請參閱 the section called “標籤傳播至已佈建資源”。

範例 2：針對名為「dev」的環境編譯的 IaC 檔案。

開發/環境 .tf：

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}
```

```
// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

開發/質子環境變量 .tf :

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

開發/質子. 自動 .tfvar.json :

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
```

```
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

儲存庫路徑

AWS Proton 使用來自環境或服務的控制台或規格輸入創建操作來查找存儲庫和路徑，它是找到編譯的 IaC 文件。輸入值會傳遞至命名空間的輸入參數。

AWS Proton 支援兩種儲存庫路徑配置。在下列範例中，路徑是由來自兩個環境的命名空間資源參數來命名。每個環境都有兩個服務的服務執行個體，而其中一個服務的服務執行個體已直接定義元件。

資源類型	名稱參數	=	資源名稱
Environment	environment.name		"env-prod"
Environment	environment.name		"env-staged"
服務	service.name		"service-one"
服務執行個體	service_instance.name		"instance-one-prod"
服務執行個體	service_instance.name	=	"instance-one-staged"
服務	service.name		"service-two"
服務執行個體	service_instance.name		"instance-two-prod"
元件	service_instance.components.default.name		"component-prod"

資源類型	名稱參數	=	資源名稱
服務執行個體	service_instance.name		"instance-two-staged"
元件	service_instance.components.default.name		"component-staged"

Layout 1

如果 AWS Proton 找到含有environments資料夾的指定儲存庫，則會建立一個資料夾，其中包含已編譯的 IaC 檔案，並以. environment.name

如果 AWS Proton 找到指定的儲存庫，其中包含與服務執行個體相容環境名稱相容的資料environments夾名稱相符的資料夾，則會建立包含已編譯執行個體 IaC 檔案的資料夾，並以. service_instance.name

```
/repo
    /environments
        /env-prod                               # environment folder
            main.tf
            proton.environment.variables.tf
            proton.auto.tfvars.json

            /service-one-instance-one-prod    # instance folder
                main.tf
                proton.service_instance.variables.tf
                proton.auto.tfvars.json

            /service-two-instance-two-prod    # instance folder
                main.tf
                proton.service_instance.variables.tf
                proton.auto.tfvars.json

            /component-prod                   # component folder
                main.tf
                proton.component.variables.tf
                proton.auto.tfvars.json
```

```
        /env-staged                             # environment folder
            main.tf
            proton.variables.tf
            proton.auto.tfvars.json

            /service-one-instance-one-staged  # instance folder
                main.tf
                proton.service_instance.variables.tf
                proton.auto.tfvars.json

            /service-two-instance-two-staged  # instance folder
                main.tf
                proton.service_instance.variables.tf
                proton.auto.tfvars.json

            /component-staged                 # component folder
                main.tf
                proton.component.variables.tf
                proton.auto.tfvars.json
```

Layout 2

如果 AWS Proton 找到沒有`environments`文件夾的指定存儲庫，它會創建一個`environment.name`文件夾，它定位編譯的環境 IaC 文件。

如果 AWS Proton 找到具有與服務實例兼容環境名稱匹配的文件夾名稱的指定存儲庫，則會創建一個`service_instance.name`文件夾，其中它位於編譯的實例 IaC 文件。

```
/repo
    /env-prod                               # environment folder
        main.tf
        proton.environment.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json
```

```
        /component-prod                  # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                          # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

結構描述檔

身為系統管理員，當您使用「開放 API 資料模型 (結構描述)」區段定義範本服務包的參數結構描述 YAML 檔案時， AWS Proton 可以根據您在結構描述中定義的需求驗證參數值輸入。

如需有關格式和可用關鍵字的詳細資訊，請參閱 OpenAPI 的結構描述物件一節。

環境範本組合的結構描述需求

您的結構描述必須遵循 OpenAPI 的資料模型 (架構) 區段，採用 YAML 格式。它也必須是環境範本包的一部分。

對於您的環境結構描述，您必須包含格式化的標頭，以確定您正在使用 Open API 的資料模型 (結構描述) 區段。在下列環境結構描述範例中，這些標頭會出現在前三行中。

必environment_input_type須使用您提供的名稱包含和定義。在下列範例中，這是在第 5 行定義的。透過定義此參數，您可以將其與 AWS Proton 環境資源產生關聯。

若要遵循 Open API 結構描述模型，您必須包含types. 在下面的例子中，這是第 6 行。

接下來types，您必須定義一個environment_input_type類型。您可以將環境的輸入參數定義為的性質environment_input_type。您必須包含至少一個屬性，其名稱與環境基礎結構中列出的至少一個參數相符，作為與結構描述相關聯的程式碼 (IaC) 檔案。

當您建立環境並提供自訂參數值時， AWS Proton 會使用結構描述檔案來比對、驗證，並將其插入至關聯 CloudFormation IaC 檔案中的大括弧參數。對於每個性質 (參數)，提供name和type。您也可以選擇性地提供descriptiondefault、和pattern。

下列範例標準環境範本結構描述所定義的參數包括vpc_cidrsubnet_one_cidr、subnet_two_cidr與default關鍵字和預設值。當您使用此環境範本套件結構描述建立環境時，您可以接受預設值或提供您自己的預設值。如果參數沒有預設值且列為required屬性 (參數)，您必須在建立環境時提供該參數的值。

第二個範例標準環境範本結構描述會列出required參數my_other_sample_input。

您可以為兩種類型的環境樣板建立結構描述。如需詳細資訊，請參閱 註冊和發佈範本。

- 標準環境範本

  在下列範例中，使用描述和輸入性質定義環境輸入類型。此結構描述範例可與範例 3 中所示的 AWS Proton CloudFormation IaC 檔案搭配使用。

  標準環境範本的範例結構描述：

```
schema:                             # required
  format:                           # required
    openapi: "3.0.0"                # required
  # required              defined by administrator
  environment_input_type: "PublicEnvironmentInput"
  types:                            # required
    # defined by administrator
    PublicEnvironmentInput:
      type: object
      description: "Input properties for my environment"
      properties:
        vpc_cidr:                    # parameter
          type: string
```

```
          description: "This CIDR range for your VPC"
          default: 10.0.0.0/16
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_one_cidr:            # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.0.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_two_cidr:            # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.1.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
```

包含required參數的標準環境範本的範例結構描述：

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "MyEnvironmentInputType"
  types:                          # required
    # defined by administrator
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:          # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:    # parameter
          type: string
          description: "Another sample input"
        another_optional_input:   # parameter
          type: string
          description: "Another optional input"
          default: "!"
      required:
        - my_other_sample_input
```

- 客戶管理的環境範本

在下列範例中，結構描述只包含複寫您用來佈建客戶管理基礎結構之 IaC 輸出的輸出清單。您需要將輸出值類型定義為僅字符串（而不是列表，數組或其他類型）。例如，下一個程式碼片段會顯示外部 AWS CloudFormation 範本的輸出區段。這是從示例 1 中顯示的模板。它可用於為範例 4 所建立的 AWS Proton Fargate 服務建立外部客戶管理的基礎架構。

> **Important**
>
> 身為系統管理員，您必須確定已佈建和受管理的基礎結構，以及所有輸出參數都與相關聯的客戶受管理環境範本相容。 AWS Proton 無法代表您考慮變更，因為這些變更不會顯示給您 AWS Proton。不一致會導致失敗。

客戶管理環境範本的範例 CloudFormation IaC 檔案輸出：

```
// Cloudformation Template Outputs
[...]
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

下列範例顯示對應 AWS Proton 客戶管理環境範本套裝軟體的結構描述。每個輸出值被定義為一個字符串。

客戶受管理環境範本的範例結構描述：

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "EnvironmentOutput"
  types:                          # required
    # defined by administrator
```

```
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:               # parameter
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:      # parameter
          type: string
          description: "The ARN of the ECS role"
        VpcId:                     # parameter
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

服務範本組合的結構描述需求

您的結構描述必須遵循 OpenAPI 的 YAML 格式的資料模型 (結構描述) 區段，如下列範例所示。您必須在服務範本包中提供結構描述檔案。

在下列服務結構描述範例中，您必須包含格式化的標頭。在下面的例子中，這是在前三行。這是為了確定您正在使用 Open API 的數據模型（模式）部分。

`service_input_type`必須使用您提供的名稱包含並定義 A。在下面的例子中，這是在第 5 行。這會將參數與 AWS Proton 服務資源產生關聯。

當您使用主控台或 CLI 建立 AWS Proton 服務時，依預設會包含服務管線。當您為服務包含服務管道時，必須包`pipeline_input_type`含您提供的名稱。在下面的例子中，這是在第 7 行。如果您不包含 AWS Proton 服務管線，請勿包含此參數。如需詳細資訊，請參閱 註冊和發佈範本。

若要遵循 Open API 結構描述模型，您必須`types`在下列範例中包含在第 9 行中。

接下來`types`，您必須定義一個`service_input_type`類型。您可以將服務的輸入參數定義為的屬性`service_input_type`。您必須包含至少一個具有名稱的屬性，該屬性與服務基礎結構中列出的至少一個參數相符，作為與結構描述相關聯的程式碼 (IaC) 檔案。

若要定義服務管線，您必須在定`service_input_type`義之下定義一個`pipeline_input_type`. 如上所述，您必須包含至少一個屬性，其名稱與結構描述相關聯的管線 IaC 檔案中列出的至少一個參數相符。如果您不包含 AWS Proton 服務管線，請勿包含此定義。

當您身為系統管理員或開發人員，建立服務並提供自訂參數值時， AWS Proton 會使用結構描述檔來比對、驗證，並將其插入關聯的 CloudFormation IaC 檔案的大括弧參數中。對於每個性質 (參數)，提供name和type。您也可以選擇性地提供descriptiondefault、和pattern。

範例結構描述的定義參數包括portdesired_count、task_sizeimage和default關鍵字和預設值。當您使用此服務範本套件結構描述建立服務時，您可以接受預設值或提供您自己的預設值。此參數unique_name也包含在範例中，且沒有預設值。它會列示為required性質 (參數)。您身為管理員或開發人員，必須在建立服務時提供required參數值。

如果您要使用服務管線建立服務範本，請在結構描述pipeline_input_type中加入。

包含服務管線之服務的範例服務結構描述檔案。 AWS Proton

此結構描述範例可與範例 4 和範例 5 中所示的 AWS Proton IaC 檔案搭配使用。包括一個服務管道。

```
schema:                             # required
  format:                           # required
    openapi: "3.0.0"                # required
  # required           defined by administrator
  service_input_type: "LoadBalancedServiceInput"
  # only include if including AWS Proton service pipeline, defined by administrator
  pipeline_input_type: "PipelineInputs"

  types:                            # required
    # defined by administrator
    LoadBalancedServiceInput:
      type: object
      description: "Input properties for a loadbalanced Fargate service"
      properties:
        port:                       # parameter
          type: number
          description: "The port to route traffic to"
          default: 80
          minimum: 0
          maximum: 65535
        desired_count:              # parameter
          type: number
          description: "The default number of Fargate tasks you want running"
          default: 1
          minimum: 1
        task_size:                  # parameter
          type: string
          description: "The size of the task you want to run"
```

```
          enum: ["x-small", "small", "medium", "large", "x-large"]
          default: "x-small"
        image:                      # parameter
          type: string
          description: "The name/url of the container image"
          default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
          minLength: 1
          maxLength: 200
        unique_name:                # parameter
          type: string
          description: "The unique name of your service identifier. This will be used
 to name your log group, task definition and ECS service"
          minLength: 1
          maxLength: 100
      required:
        - unique_name
    # defined by administrator
    PipelineInputs:
      type: object
      description: "Pipeline input properties"
      properties:
        dockerfile:                 # parameter
          type: string
          description: "The location of the Dockerfile to build"
          default: "Dockerfile"
          minLength: 1
          maxLength: 100
        unit_test_command:          # parameter
          type: string
          description: "The command to run to unit test the application code"
          default: "echo 'add your unit test command here'"
          minLength: 1
          maxLength: 200
```

如果您想要建立不含服務管線的服務範本，請勿pipeline_input_type在結構描述中包含，如下列範例所示。

不包含服務管線之服務的範例服 AWS Proton 務結構描述檔案

```
schema:                             # required
  format:                           # required
    openapi: "3.0.0"                # required
  # required            defined by administrator
```

```
  service_input_type: "MyServiceInstanceInputType"

  types:                            # required
    # defined by administrator
    MyServiceInstanceInputType:
      type: object
      description: "Service instance input properties"
      required:
        - my_sample_service_instance_required_input
      properties:
        my_sample_service_instance_optional_input:   # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_sample_service_instance_required_input:   # parameter
          type: string
          description: "Another sample input"
```

包裝的範本檔案 AWS Proton

準備您的環境和服務基礎結構作為代碼（IaC）文件及其各自的模式文件後，您必須將它們組織在目錄中。您也必須建立資訊清單 YAML 檔案。清單文件列出了目錄中的 IaC 文件，渲染引擎，以及用於在此模板中開發 IaC 的模板語言。

> **Note**
>
> 資訊清單檔案也可以獨立於範本包使用，做為直接定義之元件的直接輸入。在這種情況下，它總是為 CloudFormation 和 Terraform 指定單個 IaC 模板文件。如需元件的詳細資訊，請參閱元件。

資訊清單檔案必須遵循下列範例中顯示的格式和內容。

CloudFormation 清單文件格式：

使用 CloudFormation，您可以列出單一檔案。

```
infrastructure:
  templates:
    - file: "cloudformation.yaml"
      rendering_engine: jinja
```

```
      template_language: cloudformation
```

地形清單文件格式：

使用 terraform，您可以明確列出單個文件或使用通配符*列出目錄中的每個文件。

> **Note**
>
> 萬用字元只包含名稱以結尾的檔案.tf。其他檔案會被忽略。

```
infrastructure:
  templates:
    - file: "*"
      rendering_engine: hcl
      template_language: terraform
```

CodeBuild基於佈建資訊清單檔案格式：

使用 CodeBuild基於佈建，您可以指定佈建和取消佈建 shell 命令。

> **Note**
>
> 除了清單之外，您的軟件包還應包含命令依賴的任何文件。

下列範例資訊清單使用 CodeBuild基於佈建，使用 () 佈建 (部署) 和取消佈建 AWS Cloud Development Kit (AWS CDK) (銷毀AWS CDK) 資源。模板包還應包括 CDK 代碼。

在佈建期間，AWS Proton 會建立一個輸入檔案，其中包含您在範本結構描述中以名稱定義的輸入參數值proton-input.json。

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
```

```
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

為環境或服務範本套件組合設定目錄和資訊清單檔案後，您可以將目錄壓縮到 tar ball 中，然後將它們上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體， AWS Proton 以便在其中擷取它們，或到範本同步 Git 儲存庫。

當您建立註冊的環境或服務範本的次要版本時 AWS Proton，您會提供位於 S3 儲存貯體中的環境或服務範本 bundle tar ball 的路徑。 AWS Proton 將其與新範本次要版本一起儲存。您可以選取新的範本次要版本來建立或更新環境或服務 AWS Proton。

環境模板包裝包裝

您可以為其建立兩種類型的環境範本套裝軟體 AWS Proton。

- 若要建立標準環境範本的環境範本套裝軟體，請將結構描述、基礎結構作為程式碼 (IaC) 檔案和資訊清單檔案組合 (如下列環境範本組合包目錄結構所示) 組織在目錄中。
- 若要為客戶受管理的環境範本建立環境範本套裝軟體，請僅提供結構描述檔案和目錄。請勿包含基礎結構目錄和檔案。 AWS Proton 如果包含基礎結構目錄和檔案，則會擲回錯誤。

如需詳細資訊，請參閱 註冊和發佈範本。

CloudFormation 環境模板包目錄結構：

```
/schema
```

```
  schema.yaml
/infrastructure
  manifest.yaml
  cloudformation.yaml
```

地形環境模板捆綁目錄結構：

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  environment.tf
```

服務模板捆綁包裝

若要建立服務範本套裝軟體，您必須將結構描述、基礎結構即程式碼 (IaC) 檔案以及資訊清單檔案組織到目錄中，如服務範本組合包目錄結構範例所示。

如果您沒有在範本包中包含服務管線，請不要包含管線目錄和檔案，並在建立要與此範本套件組合關聯的服務範本`"pipelineProvisioning": "CUSTOMER_MANAGED"`時進行設定。

> **Note**
>
> 您無法在建立服務範本`pipelineProvisioning`之後進行修改。

如需詳細資訊，請參閱 註冊和發佈範本。

CloudFormation 服務範本包目錄結構：

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  cloudformation.yaml
/pipeline_infrastructure
  manifest.yaml
  cloudformation.yaml
```

地形服務模板捆綁目錄結構：

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  instance.tf
/pipeline_infrastructure
  manifest.yaml
  pipeline.tf
```

範本服務包考量

- 基礎結構即程式碼 (IaC) 檔案

  AWS Proton 稽核正確檔案格式的樣板。但是， AWS Proton 不會檢查模板開發，依賴項和邏輯錯誤。例如，假設您在 AWS CloudFormation IaC 檔案中指定建立 Amazon S3 儲存貯體做為服務或環境範本的一部分。服務是根據這些範本建立的。現在，假設在某些時候您要刪除該服務。如果指定的 S3 儲存貯體不為空，且 CloudFormation IaC 檔案未將其標記為`Retain`在中`DeletionPolicy`，則服務刪除作業會 AWS Proton 失敗。
- 套裝軟體檔案大小限制和格式
  - 您可以在中找到套裝軟體檔案大小、計數和名稱大小限制AWS Proton 配額。
  - 文件的模板捆綁目錄壓縮到焦油球中，並位於亞馬遜簡單存儲服務（Amazon S3）存儲桶中。
  - 服務包中的每個檔案都必須是有效的格式化 YAML 檔案。
- S3 儲存貯體範本套件加密

  如果您想要加密 S3 儲存貯體中靜態範本服務包中的敏感資料，請使用 SSE-S3 或 SSE-KMS 金鑰 AWS Proton 來允許擷取這些資料。

AWS Proton 範本

若要將範本組合新增至AWS Proton範本資源庫，請建立範本次要版本並使用註冊AWS Proton。建立範本時，提供您的 Amazon S3 儲存貯體名稱和範本服務包的路徑。範本發佈後，可以由平台團隊成員和開發人員選取這些範本。選取之後，AWS Proton會使用範本來建立和佈建基礎結構和應用程式。

身為管理員，您可以使用建立和註冊環境範本AWS Proton。然後，此環境範本可用於部署多個環境。例如，它可用於部署「開發」，「登台」和「生產」環境。「dev」環境可能包括具有私有子網路的 VPC，以及對所有資源的限制性存取原則。環境輸出可以用作服務的輸入。

您可以建立和註冊環境範本，以建立兩種不同類型的環境。您和開發人員都可以用AWS Proton來將服務部署到這兩種類型。

- 註冊並發佈標準環境範本，該範本AWS Proton用於建立佈建和管理環境基礎結構的標準環境。
- 註冊並發佈客戶受管環境範本，該範本AWS Proton用於建立連線至現有佈建基礎結構的客戶管理環境。 AWS Proton不會管理您現有的佈建基礎結構。

您可以建立並註冊服務範本，以AWS Proton便將服務部署到環境。必須先建立AWS Proton環境，才能將服務部署至該環境。

下列清單說明如何使用建立和管理範本AWS Proton。

- (選用) 準備 IAM 角色以控制開發人員對AWS Proton API 呼叫和AWS Proton IAM 服務角色的存取權限。如需詳細資訊，請參閱the section called “IAM 角色”。
- 撰寫範本包。如需詳細資訊，請參閱模板捆綁。
- 在範本服務包組合構成、壓縮並儲存在 Amazon S3 儲存貯體AWS Proton之後，使用建立並註冊範本。您可以在主控台中或使用來執行這項作業AWS CLI。
- 測試並使用範本，在向其註冊後建立和管理AWS Proton已佈建的資源AWS Proton。
- 在範本的整個生命週期內建立和管理範本的主要和次要版本。

您可以手動或使用範本同步設定來管理範本版本：

- 使用主AWS Proton控台並AWS CLI建立新的次要或主要版本。
- 建立範本同步配置，當偵測到您定義的儲存庫中範本套件的變更時，可AWS Proton自動建立新的次要或主要版本。

如需其他資訊，請參閱AWS Proton服務 API 參考。

主題

版本化模板

身為系統管理員或平台小組的成員，您可以定義、建立及管理用於佈建基礎結構資源的版本化範本的程式庫。範本版本有兩種類型：次要版本和主要版本。

- 次要版本 — 對具有向後相容結構描述的範本所做的變更。這些更改不需要開發人員在更新到新模板版本時提供新信息。

  當您嘗試進行次要版本變更時，AWS Proton會盡最大努力判斷新版本的結構描述是否與範本的舊次要版本向下相容。如果新結構描述不向後相容，則新次要版本的註冊AWS Proton失敗。

  > **Note**
  >
  > 相容性僅根據結構描述決定。 AWS Proton不檢查模板包基礎結構代碼（IaC）文件是否與以前的次要版本向後兼容。例如，AWS Proton不會檢查新的 IaC 檔案是否會對範本的舊次要版本佈建的基礎結構上執行的應用程式造成中斷變更。

- 主要版本-對模板的更改可能不向後兼容。這些更改通常需要開發人員的新輸入，並且通常涉及模板模式更改。

  您有時可能會根據團隊的營運模式，選擇將向後相容的變更指定為主要版本。

AWS Proton決定範本版本請求是次要版本還是主要版本的方式，取決於追蹤範本變更的方式：

- 當您明確提出建立新範本版本的要求時，您可以透過指定主要版本號碼來要求主要版本，而不指定主要版本號碼來請求次要版本。

- 當您使用範本同步處理 (因此您沒有提出明確的範本版本要求) 時，會AWS Proton嘗試針對現有 YAML 檔案中發生的範本變更建立新的次要版本。 AWS Proton當您為新範本變更建立新目錄時 (例如，從 v1 移至 v2)，會建立主要版本。

> **Note**
>
> 如果AWS Proton判斷變更不向後相容，則以範本同步處理為基礎的新次要版本註冊仍會失敗。

當您發佈範本的新版本時，如果範本是最高的主要和次要版本，則該範本會變成「建議」版本。新AWS Proton資源是使用新的建議版本建立的，並AWS Proton提示管理員使用新版本並更新使用過期版本的現有AWS Proton資源。

註冊和發佈範本

您可以使用註冊和發行環境和服務範本AWS Proton，如下列各節所述。

您可以使用主控台或建立新版本的範本AWS CLI。

或者，您可以使用控制台或AWS CLI建立範本並為其設定範本同步。此配置允許從位於已定義的已註冊 git 存儲庫中的模板包進行AWS Proton同步。每當將提交推送到更改模板包之一的存儲庫時，如果該版本尚不存在，則會創建模板的新次要或主要版本。若要深入瞭解範本同步設定必要條件和需求，請參閱範本同步設定。

註冊和發佈環境範本

您可以註冊和發佈以下類型的環境樣板。

- 註冊並發佈AWS Proton用於部署和管理環境基礎結構的標準環境範本。
- 註冊並發佈客戶受管環境範本，該範本AWS Proton用於連線至您管理的現有佈建基礎結構。 AWS Proton不會管理您現有的佈建基礎結構。

> **Important**
>
> 身為系統管理員，請確定已佈建和受管理的基礎結構及所有輸出參數都與相關的客戶受管理環境範本相容。 AWS Proton無法代表您考慮變更，因為這些變更不會顯示給您AWS Proton。不一致性會造成失敗。

您可以使用主控台或註冊和發佈環境範本。AWS CLI

AWS Management Console

使用主控台註冊並發佈新的環境範本。

1. 在AWS Proton主控台中，選擇 [環境範本]。
2. 選擇建立環境範本。
3. 在「建立環境樣板」頁面的「樣板選項」段落中，選擇兩個可用的樣板選項之一。
   - 建立用於佈建新環境的範本。
   - 建立範本以使用您管理的已佈建基礎結構。
4. 如果您選擇「建立用於啟動設定新環境的樣板」，請在「範本套裝軟體來源」區段中，選擇三個可用的樣板套裝軟體來源選項之一。若要進一步瞭解同步範本的需求和必要條件，請參閱範本同步設定。
   - 使用我們的示例模板包之一。
   - 使用您自己的模板包。
   - 從 Git 同步範本。
5. 提供範本服務包的路徑。
   a. 如果您選擇使用我們的範例範本組合包之一：

      在「範例範本套裝軟體」區段中，選取範例範本套裝軟體。
   b. 如果您在「原始程式碼」區段中選擇「從 Git 同步範本」：
      i. 選取範本同步配置的儲存庫。
      ii. 輸入要同步的來源存放庫分支的名稱。
      iii. (選擇性) 輸入目錄名稱，以限制範本套裝軟體的搜尋。
   c. 否則，在 S3 服務包位置區段中，提供範本服務包的路徑。
6. 在「範本詳細資料」區段中。
   a. 輸入「範本」名稱。
   b. (選用性) 輸入範本顯示名稱。
   c. (選用性) 輸入範本的描述。

7. (選擇性) 在「加密設定」區段中勾選「自訂加密設定 (進階)」核取方塊，以提供您自己的加密金鑰。
8. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。
9. 選擇「建立環境範本」。

   您現在進入了一個新頁面，其中會顯示新環境範本的狀態和詳細資料。這些詳細資料包括清單AWS和客戶管理的標籤。 AWS Proton當您建立AWS Proton資源時，會自動為您產生AWS受管理的標籤。如需詳細資訊，請參閱AWS Proton資源和標記。
10. 新環境範本狀態的狀態會以「草稿」狀態開始。您和擁有`proton:CreateEnvironment`權限的其他人都可以檢視和存取它。請按照下一步操作，使其他人可以使用該範本。
11. 在「範本版本」區段中，選擇您剛建立的範本次要版本左側的圓鈕 (1.0)。作為替代方法，您可以在信息警報中選擇「發布」並跳過下一步。
12. 在「範本版本」區段中，選擇「發佈」。
13. 範本狀態會變更為「已發佈」。因為它是模板的最新版本，所以它是推薦版本。
14. 在導覽窗格中，選取 [環境範本] 以檢視環境範本和詳細資料的清單。

使用主控台註冊環境範本的新主要和次要版本。

如需詳細資訊，請參閱版本化模板。

1. 在AWS Proton主控台中，選擇「環境範本」。
2. 在環境樣板清單中，選擇您要為其建立主要或次要版本之環境樣板的名稱。
3. 在環境範本詳細資料檢視中，選擇 [範本版本] 區段中的 [建立新版本]。
4. 在「建立新環境範本版本」頁面的「範本套裝軟體來源」區段中，選擇兩個可用的範本套裝軟體來源選項之一。

   - 使用我們的示例模板包之一。
   - 使用您自己的模板包。
5. 提供所選範本套裝軟體的路徑。

   - 如果您選擇「使用我們的範例範本套裝軟體」，請在「範例範本套件組合」區段中，選取範例範本套件組合。
   - 如果您選擇 [使用您自己的範本服務包]，請在 S3 套件組合位置區段中選擇範本服務包的路徑。
6. 在「範本詳細資料」區段中。

a. (選用性) 輸入範本顯示名稱。

b. (選用性) 輸入服務範本的描述。

7. 在範本詳細資料區段中，選擇下列其中一個選項。

- 若要建立次要版本，請將核取方塊保留勾選以建立新的主要版本為空。
- 若要建立主要版本，請勾選核取方塊以建立新的主要版本。

8. 繼續執行主控台步驟以建立新的次要或主要版本，然後選擇 [建立新版本]。

AWS CLI

使用 CLI 來註冊和發佈新的環境範本，如下列步驟所示。

1. 透過指定區域、名稱、顯示名稱 (選擇性) 和說明 (選用)，建立標準 OR 客戶管理的環境範本。

a. 建立標準環境範本。

執行以下命令：

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access"
```

回應：

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env"
    }
}
```

b. 透過新增具有值的`provisioning`參數來建立客戶管理的環境範本`CUSTOMER_MANAGED`。

執行以下命令：

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access" \
    --provisioning "CUSTOMER_MANAGED"
```

回應：

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env",
        "provisioning": "CUSTOMER_MANAGED"
    }
}
```

2. 建立環境範本主要版本 1 的次要版本 0

對於標準和客戶管理的環境範本，這和其餘步驟都是相同的。

包含範本名稱、主要版本，以及包含環境範本組合包的儲存貯體的 S3 儲存貯體名稱和金鑰。

執行以下命令：

```
$ aws proton create-environment-template-version \
    --template-name "simple-env" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}"
```

回應：

```
{
    "environmentTemplateVersion": {
```

```
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "simple-env"
    }
}
```

3. 使用 get 命令來檢查註冊狀態。

   執行以下命令：

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

   回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
```

```
        "templateName": "simple-env"
    }
}
```

4. 透過提供範本名稱以及主要和次要版本，發佈環境範本的主要版本 1 的次要版本 0。這個版本是Recommended版本。

   執行以下命令：

```
$ aws proton update-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   回應：

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

使用建立新範本後AWS CLI，您可以檢視AWS和客戶管理的標籤清單。 AWS Proton自動為您產生AWS受管理的標籤。您也可以使用修改和建立客戶管理的標籤AWS CLI。如需詳細資訊，請參閱AWS Proton資源和標記。

執行以下命令：

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment-
template/simple-env"
```

註冊和發佈服務範本

當您建立服務範本版本時，您可以指定相容的環境範本清單。這樣，當開發人員選擇服務模板時，他們可以選擇將其服務部署到哪個環境。

從服務範本建立服務之前，或在發佈服務範本之前，請確認已從列出的相容環境範本部署環境。

如果服務部署到從已移除的相容環境範本建置的環境中，則無法將服務更新為新的主要版本。

若要新增或移除服務範本版本的相容環境範本，請建立新的主要版本。

您可以使用主控台或註冊和發行服務範本。AWS CLI

AWS Management Console

使用主控台註冊並發佈新的服務範本。

1. 在AWS Proton主控台中，選擇 [服務範本]。
2. 選擇 [建立服務範本]。
3. 在「建立服務範本」頁面的「範本套裝軟體來源」區段中，選擇其中一個可用的範本選項。
   - 使用您自己的模板包。
   - 從 Git 同步範本。
4. 提供範本服務包的路徑。
   a. 如果您選擇從 Git 同步模板，請在源代碼存儲庫部分中：
      i. 選取範本同步配置的儲存庫。
      ii. 輸入要同步的來源存放庫分支的名稱。
      iii. (選擇性) 輸入目錄名稱，以限制範本套裝軟體的搜尋。

b. 否則，在 S3 服務包位置區段中，提供範本服務包的路徑。

5. 在「範本詳細資料」區段中。

   a. 輸入「範本」名稱。

   b. (選用性) 輸入範本顯示名稱。

   c. (選用性) 輸入服務範本的描述。

6. 在「相容的環境範本」區段中，從相容的環境範本清單中選擇。

7. (選擇性) 在「加密設定」區段中，選擇「自訂加密設定 (進階)」以提供您自己的加密金鑰。

8. (選用) 在「管線」區段中：

   如果您未在服務範本中包含服務管線定義，請取消勾選頁面底部的 Pipeline-選用核取方塊。您無法在建立服務範本之後變更此設定。如需詳細資訊，請參閱模板捆綁。

9. (選擇性) 在「支援的元件來源」段落中，對於「元件來源」，請選擇直接定義，將直接定義的元件附加至服務執行處理。

10. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。

11. 選擇 [建立服務範本]。

    您現在進入了一個新頁面，其中顯示新服務模板的狀態和詳細信息。這些詳細資料包括清單AWS和客戶管理的標籤。 AWS Proton當您建立AWS Proton資源時，會自動為您產生AWS受管理的標籤。如需詳細資訊，請參閱AWS Proton資源和標記。

12. 新服務範本狀況的狀態會以「草稿」狀態開始。您和擁有`proton:CreateService`權限的其他人都可以檢視和存取它。請按照下一步操作，使其他人可以使用該範本。

13. 在「範本版本」區段中，選擇您剛建立的範本次要版本左側的圓鈕 (1.0)。作為替代方法，您可以在信息警報中選擇「發布」並跳過下一步。

14. 在「範本版本」區段中，選擇「發佈」。

15. 範本狀態會變更為「已發佈」。因為它是模板的最新版本，所以它是推薦版本。

16. 在瀏覽窗格中，選取 [服務範本] 以檢視服務範本和詳細資料的清單。

使用主控台註冊服務範本的新主要和次要版本。

如需詳細資訊，請參閱版本化模板。

1. 在AWS Proton主控台中，選擇 [服務範本]。

2. 在服務範本清單中，選擇您要為其建立主要或次要版本的服務範本名稱。

3. 在服務範本詳細資料檢視中，選擇 [範本版本] 區段中的 [建立新版本]。
4. 在 [建立新服務範本版本] 頁面的 [套裝軟體來源] 區段中，選取 [使用您自己的範本套裝軟體]。
5. 在 S3 服務包位置區段中，選擇範本服務包的路徑。
6. 在「範本詳細資料」區段中。

   a. (選用性) 輸入範本顯示名稱。

   b. (選用性) 輸入服務範本的描述。
7. 在範本詳細資料區段中，選擇下列其中一個選項。

   - 若要建立次要版本，請將核取方塊保留勾選以建立新的主要版本為空。
   - 若要建立主要版本，請勾選核取方塊以建立新的主要版本。
8. 繼續執行主控台步驟以建立新的次要或主要版本，然後選擇 [建立新版本]。

AWS CLI

若要建立在沒有服務管線的情況下部署服務的服務範本，請將參數和值新增`--pipeline-provisioning "CUSTOMER_MANAGED"`至`create-service-template`命令。依照建立和中所述設定範本套裝軟模板捆綁體服務範本組合的結構描述需求。

> **Note**
>
> 您無法在建立服務範本`pipelineProvisioning`之後進行修改。

1. 使用 CLI 來註冊和發佈新的服務範本 (含或不含服務管線)，如下列步驟所示。

   a. 使用 CLI 建立具有服務管線的服務範本。

   指定名稱、顯示名稱 (選擇性) 和說明 (選擇性)。

   執行以下命令：

   ```
   $ aws proton create-service-template \
       --name "fargate-service" \
       --display-name "Fargate" \
       --description "Fargate-based Service"
   ```

   回應：

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service"
    }
}
```

b. 建立不含服務管線的服務範本。

新增 `--pipeline-provisioning`。

執行以下命令：

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service" \
    --pipeline-provisioning "CUSTOMER_MANAGED"
```

回應：

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

2. 建立服務範本主要版本 1 的次要版本 0。

   包含範本名稱、相容環境範本、主要版本，以及包含服務範本組合包之儲存貯體的 S3 儲存貯體名稱和金鑰。

   執行以下命令：

```
$ aws proton create-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}" \
    --compatible-environment-templates '[{"templateName":"simple-
env","majorVersion":"1"}]'
```

   回應：

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

3. 使用 get 命令檢查註冊狀態。

   執行以下命令：

```
$ aws proton get-service-template-version \
```

```
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

回應：

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

4. 使用 update 指令將狀態變更為來發佈服務範本"PUBLISHED"。

執行以下命令：

```
$ aws proton update-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

回應：


```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello pipeline
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"\n",
```

```
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

5. 使用 get 命令擷取服務範本詳細資料，檢查是否AWS Proton已發佈 1.0 版。

   執行以下命令：

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   回應：

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:03:04.767000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
```

```
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

檢視範本資料

您可以使用AWS Proton主控台和來檢視包含詳細資料的範本清單，並檢視包含詳細資料的個別範本AWS CLI。

客戶管理的環境範本資料包括具有值的`provisioned`參數`CUSTOMER_MANAGED`。

如果服務範本未包含服務管線，則服務範本資料會包含該值的`pipelineProvisioning`參數`CUSTOMER_MANAGED`。

如需詳細資訊，請參閱註冊和發佈範本。

您可以使用主控台或AWS CLI來列出和檢視範本資料。

AWS Management Console

使用控制台列出和查看模板。

1. 若要檢視範本清單，請選擇 (環境或服務) 範本。
2. 若要檢視詳細資料，請選擇範本名稱。

   檢視範本的詳細資料、範本的主要和次要版本清單、使用範本版本和範本標籤部署的AWS Proton資源清單。

   建議的主要版本和次要版本會標示為「建議」。

AWS CLI

使用列AWS CLI示和檢視樣板。

執行以下命令：

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

回應：

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-10T18:35:08.293000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-10T18:35:11.162000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

執行以下命令：

```
$ aws proton list-environment-templates
```

回應：

```
{
    "templates": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-3",
```

```
            "createdAt": "2020-11-10T18:35:05.763000+00:00",
            "description": "VPC with Public Access",
            "displayName": "VPC",
            "lastModifiedAt": "2020-11-10T18:35:05.763000+00:00",
            "name": "simple-env-3",
            "recommendedVersion": "1.0"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-1",
            "createdAt": "2020-11-10T00:14:06.881000+00:00",
            "description": "Some SSM Parameters",
            "displayName": "simple-env-1",
            "lastModifiedAt": "2020-11-10T00:14:06.881000+00:00",
            "name": "simple-env-1",
            "recommendedVersion": "1.0"
        }
    ]
}
```

檢視服務範本的次要版本。

執行以下命令：

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

回應：

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
```

```
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
       my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
  my_sample_pipeline_required_input:\n          type: string\n          description:
 \"Another sample input\"\n\n    MyServiceInstanceInputType:\n      type: object
\n      description: \"Service instance input properties\"\n      required:\n
       - my_sample_service_instance_required_input\n      properties:\n
 my_sample_service_instance_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
     my_sample_service_instance_required_input:\n          type: string\n
 description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

檢視沒有服務管線的服務範本，如下一個範例命令和回應所示。

執行以下命令：

```
$ aws proton get-service-template \
    --name "simple-svc-template-cli"
```

回應：

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/simple-svc-
template-cli",
        "createdAt": "2021-02-18T15:38:57.949000+00:00",
        "displayName": "simple-svc-template-cli",
        "lastModifiedAt": "2021-02-18T15:38:57.949000+00:00",
        "status": "DRAFT",
        "name": "simple-svc-template-cli",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
```

```
    }
}
```

更新範本

您可以按照以下清單中的描述更新範本。

- 使用主控台`description`或時，請編輯範本`display name`的或AWS CLI。您無法編輯`name`範本的。
- 當您使用主控台或時，請更新範本次要版本的狀態AWS CLI。您只能將狀態從變更`DRAFT`為`PUBLISHED`。
- 使用時，編輯範本次要或主要版本的顯示名稱和描述AWS CLI。

AWS Management Console

使用主控台編輯範本說明和顯示名稱，如下列步驟所述。

在模板列表中。

1. 在AWS Proton主控台中，選擇 (環境或服務) 範本。
2. 在範本清單中，選擇您要更新描述或顯示名稱之範本左側的圓鈕。
3. 選擇動作，然後選擇編輯。
4. 在 [編輯 (環境或服務) 範本] 頁面的 [範本詳細資料] 區段中，在表單中輸入您的編輯，然後選擇 [儲存變更]。

使用主控台變更範本次要版本的狀態，如下所述發佈範本。您只能將狀態從變更`DRAFT`為`PUBLISHED`。

在（環境或服務）模板詳細信息頁面中。

1. 在AWS Proton主控台中，選擇 (環境或服務) 範本。
2. 在範本清單中，選擇您要將次要版本狀態從「草稿」更新為「已發佈」的範本名稱。
3. 在 (環境或服務) 範本詳細資料頁面的「範本版本」區段中，選取您要發佈之次要版本左側的圓鈕。
4. 在「範本版本」區段中選擇「發佈」。狀態會從「草稿」變更為「已發佈」。

AWS CLI

下列範例指令和回應展示如何編輯環境範本的描述。

執行下列命令。

```
$ aws proton update-environment-template \
    --name "simple-env" \
    --description "A single VPC with public access"
```

回應：

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-28T22:02:10.651000+00:00",
        "description": "A single VPC with public access",
        "displayName": "simple-env",
        "lastModifiedAt": "2020-11-29T16:11:18.956000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

您也可以使用AWS CLI來更新服務範本。如需更新次要版本服務範本狀態的範例，請註冊和發佈服務範本參閱步驟 5。

刪除範本

您可以使用主控台和刪除範本AWS CLI。

如果沒有部署到該版本的環境，您可以刪除該環境範本的次要版本。

如果服務範本沒有部署到該版本的服務執行個體或管道，您可以刪除次要版本的服務範本。您的管道可以部署到與服務實例不同的模板版本。例如，如果您的服務實例從 1.0 更新為 1.1 版，並且管道仍部署到 1.0 版，則無法刪除服務模板 1.0。

AWS Management Console

您可以使用主控台刪除整個範本，或刪除範本的個別次要和主要版本。

使用主控台刪除範本，如下所示。

> **Note**
>
> 使用主控台刪除範本。
>
> - 當您刪除整個範本時，您也會刪除該範本的主要版本和次要版本。

在（環境或服務）模板列表中。

1. 在AWS Proton主控台中，選擇 (環境或服務) 範本。
2. 在範本清單中，選取您要刪除的範本清單按鈕。

   只有在沒有AWS Proton資源部署到其版本的情況下，您才能刪除整個範本。
3. 選擇「動作」，然後選擇「刪除」以刪除整個範本。
4. 強制回應提示您確認刪除動作。
5. 按照說明進行操作，然後選擇是，刪除。

在（環境或服務）模板詳細信息頁面中。

1. 在AWS Proton主控台中，選擇 (環境或服務) 範本。
2. 在範本清單中，選擇您要刪除的範本名稱，或刪除其個別主要或次要版本的名稱。

3. 刪除整個範本。

   只有在沒有AWS Proton資源部署到其版本的情況下，您才能刪除整個範本。

   a. 選擇頁面右上角的「刪除」。
   b. 強制回應提示您確認刪除動作。
   c. 按照說明進行操作，然後選擇是，刪除。

4. 若要刪除範本的主要或次要版本。

   如果沒有部署到該版本的AWS Proton資源，則只能刪除該範本的次要版本。

   a. 在範本版本區段中，選取您要刪除的版本選項按鈕。
   b. 在 [範本版本] 區段中選擇 [刪除]。
   c. 強制回應提示您確認刪除動作。
   d. 按照說明進行操作，然後選擇是，刪除。

AWS CLI

AWS CLI範本刪除作業不包括刪除範本的其他版本。使用時AWS CLI，刪除具有下列條件的範本。

- 如果範本沒有次要或主要版本，請刪除整個範本。
- 刪除最後一個剩餘的次要版本時，請刪除主要版本。
- 如果沒有部署到該版本的AWS Proton資源，請刪除該範本的次要版本。
- 如果範本沒有其他次要版本存在，且沒有部署至該版本的AWS Proton資源，請刪除建議的次要版本。

下列範例指令和回應展示如何使用AWS CLI刪除範本。

執行以下命令：

```
$ aws proton delete-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

回應：

```
{
```

```
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

執行以下命令：

```
$ aws proton delete-environment-template \
    --name "simple-env"
```

回應：

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with Public Access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-12T00:23:22.339000+00:00",
        "name": "simple-env",
        "recommendedVersion": "1.0"
    }
}
```

執行以下命令：

```
$ aws proton delete-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

回應：

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [{"majorVersion": "1", "templateName":
 "simple-env"}],
        "createdAt": "2020-11-28T22:07:05.798000+00:00",
        "lastModifiedAt": "2020-11-28T22:19:05.368000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

範本同步設定

了解如何配置模板以允許從您定義的已註冊 git 存儲庫中的模板包進行 AWS Proton 同步。將提交推送至您的儲存庫時， AWS Proton 檢查儲存庫範本服務包是否有變更。如果偵測到範本套件組合變更，則會建立範本的新次要或主要版本 (如果該版本尚未存在)。 AWS Proton 目前支援 GitHub「 GitHub 企業 BitBucket」和。

將提交推送到同步的模板包

當您將提交推送到由其中一個模板跟踪的分支時，請 AWS Proton 克隆您的存儲庫並確定需要同步的模板。它會掃描目錄中的檔案，以尋找符合慣例的目錄`{template-name}/{major-version}/`。

在 AWS Proton 確定哪些模板和主要版本與您的存儲庫和分支相關聯後，它會開始嘗試 parallel 同步所有這些模板。

在每次同步到特定範本時， AWS Proton 首先檢查範本目錄的內容是否在上次成功同步後變更。如果內容沒有變更，則會 AWS Proton 略過註冊重複的套裝軟體。這樣可確保在範本服務包的內容發生變更時，會建立新的範本次要版本。如果範本套裝軟體的內容已變更，則會使用註冊套裝軟體 AWS Proton。

註冊範本套裝軟體之後，會 AWS Proton 監視註冊狀態，直到註冊完成為止。

在單一指定時間內，特定範本次要和主要版本只能進行一次同步。在進行同步時可能已推送的任何提交都會進行批次處理。批次提交會在先前的同步嘗試完成後同步。

同步服務範本

AWS Proton 可以從 git 存儲庫同步環境和服務模板。要同步您的服務模板，您可以添加一個名為的附加文件`.template-registration.yaml`到模板包中的每個主要版本目錄。此檔案包含在提交後為您建立服務範本版本時所 AWS Proton 需的其他詳細資料：相容環境和支援的元件來源。

檔案的完整路徑為*`service-template-name`*`/`*`major-version`*`/.template-registration.yaml`。如需詳細資訊，請參閱 the section called “同步服務範本”。

範本同步組態考量

檢閱下列使用範本同步設定的考量事項。

- 儲存庫的大小不得超過 250 MB。
- 若要設定範本同步，請先將存放庫連結至 AWS Proton。如需詳細資訊，請參閱 the section called “建立儲庫連結”。
- 從同步範本建立新的範本版本時，該範本會處於DRAFT狀態。
- 如果下列條件之一成立，則會建立範本的新次要版本：
  - 範本服務包內容與上次同步處理範本次要版本的內容不同。
  - 上次同步的範本次要版本已刪除。
- 同步無法暫停。
- 新的次要版本或主要版本都會自動同步。
- 範本同步設定無法建立新的頂層範本。
- 您無法使用模板同步配置從多個存儲庫同步到一個模板。
- 您不能使用標籤而不是分支。
- 建立服務範本時，您可以指定相容的環境範本。
- 您可以建立環境範本，並在同一次提交中將其新增為服務範本的相容環境。
- 同步到單個模板主要版本一次運行一個。在同步過程中，如果檢測到任何新的提交，它們將在活動同步結束時進行批處理並應用。同步到不同的模板主要版本 parallel 發生。
- 如果您更改了模板正在同步的分支，則從舊分支進行的任何正在進行的同步都會首先完成。然後同步從新分支開始。
- 如果您變更範本同步來源的儲存庫，舊儲存庫中的任何進行中同步可能會失敗或執行完成。這取決於它們所處的同步階段。

如需詳細資訊，請參閱 AWS Proton 服務 API 參考。

主題

- 建立範本同步設定
- 檢視範本同步設定詳情
- 編輯範本同步設定
- 刪除範本同步設定

建立範本同步設定

瞭解如何使用建立範本同步設定 AWS Proton。

建立範本同步設定先決條件：

- 您已將儲存庫與 AWS Proton.
- 範本包位於您的存放庫中。

存放庫連結由下列項目組成：

- 授與存取存放庫及訂閱其通知之 AWS Proton 權限的 CodeConnections 連線。
- 服務連結的角色。當您連結存放庫時，會為您建立服務連結角色。

在您建立第一個範本同步組態之前，請將範本組合包推送至您的儲存庫，如下列目錄配置所示。

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
/templates/my-env-template/v1/schema/
```

在您建立第一個範本同步設定之後，當您推送提交 (在新版本下新增更新的範本組合包/my-env-template/v2/) 時，系統會自動建立新的範本版本。

```
/templates/                                                 # subdirectory (optional)
/templates/my-env-template/                                 # template name
/templates/my-env-template/v1/                              # template version
/templates/my-env-template/v1/infrastructure/               # template bundle
```

```
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

您可以在單次提交中包含一個或多個同步配置的範本的新範本套件版本。 AWS Proton 為提交中包含的每個新模板包版本創建一個新的模板版本。

建立範本同步組態之後，您仍然可以在主控台中手動建立範本的新版本，或透 AWS CLI 過從 S3 儲存貯體上傳範本組合。範本同步只能在一個方向上運作：從儲存庫到 AWS Proton. 手動建立的範本版本不會同步。

設定範本同步組態後， AWS Proton 會偵聽儲存庫的變更。每當推送變更時，都會尋找與範本名稱相同的任何目錄。然後，它會在該目錄中查找看起來像主要版本的任何目錄。 AWS Proton 將範本套裝軟體註冊到對應的範本主要版本。新版本始終處於狀DRAFT態。您可以使用控制台或發布新版本 AWS CLI。

例如，假設您有一個名為`my-env-template`配置為使用以下佈局`my-repo/templates`的分支`main`同步的模板。

```
/code
/code/service.go
README.md
/templates/
/templates/my-env-template/
/templates/my-env-template/v1/
/templates/my-env-template/v1/infrastructure/
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

AWS Proton 將的內容`my-env-template:1`與`/templates/my-env-template/v1/`的內容同步`/templates/my-env-template/v2/`到`my-env-template:2`。如果它們不存在，它會創建這些主要版本。

AWS Proton 找到符合範本名稱的第一個目錄。您可以在建立或編輯範本同步配置`subdirectoryPath`時指定一個來限制目錄 AWS Proton 搜尋。例如，您可以指定`/production-templates/`為`subdirectoryPath`。

您可以使用主控台或 CLI 建立範本同步設定。

AWS Management Console

使用主控台建立範本和範本同步設定。

1. 在AWS Proton 主控台中，選擇 [環境範本]。
2. 選擇建立環境範本。
3. 在「建立環境樣板」頁面的「樣板選項」段落中，選擇建立樣板以啟動設定新環境。
4. 在 [範本套件來源] 區段中，選擇 [從 Git 同步範本]。
5. 在「原始程式碼儲存庫」區段中：
   a. 對於「存放庫」，選取包含範本服務包的連結存放庫。
   b. 對於「分支」，請選取要同步的來源儲存庫分支。
   c. (選擇性) 對於 Tem plate bundle 目錄，請輸入目錄的名稱，以涵蓋範本服務包的搜尋範圍。
6. 在「範本詳細資料」區段中。
   a. 輸入範本名稱。
   b. (選擇性) 輸入「範本」顯示名稱。
   c. (選擇性) 輸入環境範本的範本說明。
7. (選擇性) 勾選「加密設定」區段中的「自訂加密設定 (進階)」核取方塊，以提供您自己的加密金鑰。
8. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。
9. 選擇「建立環境範本」。

   您現在進入了一個新頁面，其中會顯示新環境範本的狀態和詳細資料。這些詳細資料包括 AWS 受管理和客戶管理的標籤清單。 AWS Proton 當您建立 AWS Proton 資源時，會自動為您產生 AWS 受管理的標籤。如需詳細資訊，請參閱 AWS Proton資源和標記。
10. 在範本詳細資料頁面中，選擇 [同步] 索引標籤以檢視範本同步設定詳細資料。
11. 選擇「範本版本」標籤以檢視具有狀態詳細資訊的範本版本。
12. 新環境範本狀態的狀態會以「草稿」狀態開始。您和擁有`proton:CreateEnvironment`權限的其他人都可以檢視和存取它。請按照下一步操作，使其他人可以使用該範本。
13. 在「範本版本」區段中，選擇您剛建立之範本次要版本左側的圓鈕 (1.0)。作為替代方法，您可以在信息警報中選擇「發布」並跳過下一步。
14. 在「範本版本」區段中，選擇「發佈」。

15. 範本狀態會變更為「已發佈」。這是模板的最新和推薦版本。
16. 在導覽窗格中，選取 [環境範本] 以檢視環境範本和詳細資料的清單。

建立服務範本和範本同步設定的程序類似。

AWS CLI

使用建立範本和範本同步設定 AWS CLI。

1. 建立範本。在此範例中，會建立環境範本。

   執行下列命令。

```
$ aws proton create-environment-template \
    --name "env-template"
```

   回應如下。

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:us-east-1:123456789012:environment-template/env-template",
        "createdAt": "2021-11-07T23:32:43.045000+00:00",
        "displayName": "env-template",
        "lastModifiedAt": "2021-11-07T23:32:43.045000+00:00",
        "name": "env-template",
        "status": "DRAFT",
        "templateName": "env-template"
    }
}
```

2. 透過提供下列資訊 AWS CLI 來建立範本同步配置：

   - 您要同步到的範本。建立範本同步設定之後，您仍然可以在主控台中手動建立新版本，或使用 AWS CLI.
   - 範本名稱。
   - 範本類型。
   - 您要同步的來源連結儲存庫。
   - 連結的儲存庫提供者。
   - 範本束所在的分支。

- (選擇性) 包含範本套件的目錄路徑。依預設，會 AWS Proton 尋找符合範本名稱的第一個目錄。

執行下列命令。

```
$ aws proton create-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-name "myrepos/templates" \
    --repository-provider "GITHUB" \
    --branch "main" \
    --subdirectory "env-template/"
```

回應如下。

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

3. 若要發佈範本版本，請參閱註冊和發佈範本。

同步服務範本

上述範例顯示如何同步環境範本。服務範本類似。若要同步服務範本，請在範本包中的每個主要版本目錄中新增一個名為.template-registration.yaml的其他檔案。此檔案包含在提交後為您建立服務範本版本時所 AWS Proton 需的其他詳細資料。當您使用 AWS Proton 主控台或 API 明確建立服務範本版本時，請將這些詳細資料作為輸入提供，而此檔案會取代這些輸入以進行範本同步處理。

```
./templates/                                                    # subdirectory (optional)
 /templates/my-svc-template/                                    # service template name
 /templates/my-svc-template/v1/                                 # service template version
 /templates/my-svc-template/v1/.template-registration.yaml      # service template version
 properties
```

```
/templates/my-svc-template/v1/instance_infrastructure/      # template bundle
/templates/my-svc-template/v1/schema/
```

該.template-registration.yaml文件包含以下詳細信息：

- 相容環境 [必要] — 以這些環境範本和主要版本為基礎的環境與以此服務範本版本為基礎的服務相容。
- 支援的元件來源 [選用] — 使用這些來源的元件與以此服務範本版本為基礎的服務相容。如果未指定，則無法將組件附加到這些服務。如需元件的詳細資訊，請參閱元件。

該文件的 YAML 語法如下所示：

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
  - DIRECTLY_DEFINED
```

指定一或多個環境範本/主要版本組合。指定supported_component_sources是選擇性的，唯一支援的值為DIRECTLY_DEFINED。

Example . 模板註冊.

在此範例中，服務範本版本與my-env-template環境範本的主要版本 1 和 2 相容。它也與another-env-template環境模板的主要版本 1 和 3 兼容。該文件未指定supported_component_sources，因此組件無法附加到基於此服務模板版本的服務。

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
  - another-env-template:3
```

> **Note**
>
> 先前， AWS Proton 定義了一個不同的檔案.compatible-envs，以指定相容的環境。 AWS Proton 仍然支持該文件及其格式以實現向後兼容性。我們不建議再使用它，因為它不可擴展，也無法支持更新的功能，如組件。

檢視範本同步設定詳情

使用主控台或 CLI 檢視範本同步組態詳細資料。

AWS Management Console

使用主控台檢視範本同步設定詳細資料。

1. 在瀏覽窗格中，選擇 (環境或服務) 範本。
2. 若要檢視詳細資料，請選擇您為其建立範本同步設定的範本名稱。
3. 在範本的詳細資料頁面中，選取 [同步] 索引標籤以檢視範本同步設定詳細資料。

AWS CLI

使用檢 AWS CLI 視已同步的範本。

執行下列命令。

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

回應如下。

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
        "templateType": "SERVICE"
    }
}
```

使用取 AWS CLI 得範本同步狀態。

在中`template-version`，輸入範本主要版本。

執行下列命令。

```
$ aws proton get-template-sync-status \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

編輯範本同步設定

您可以編輯除`template-name`和以外的任何範本同步組態參數`template-type`。

了解如何使用主控台或 CLI 編輯範本同步設定。

AWS Management Console

使用控制台編輯模板同步配置分支。

在模板列表中。

1. 在AWS Proton 主控台中，選擇 (環境或服務) 範本。
2. 在範本清單中，選擇具有您要編輯之範本同步設定的範本名稱。
3. 在範本詳細資料頁面中，選擇 [範本同步] 索引標籤。
4. 在範本同步詳細資料區段中，選擇編輯。
5. 在「編輯」頁面的「原始程式碼儲存區域」段落中，選取分支做為「分支」，然後選擇「儲存組態」。

AWS CLI

下列範例命令和回應顯示如何**branch**使用 CLI 編輯範本同步組態。

執行下列命令。

```
$ aws proton update-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-provider "GITHUB" \
    --repository-name "myrepos/templates" \
    --branch "fargate" \
    --subdirectory "env-template"
```

回應如下。

```
{
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

同樣地，您也可以使用 AWS CLI 來更新已同步的服務範本。

刪除範本同步設定

使用主控台或 CLI 刪除範本同步設定。

AWS Management Console

使用主控台刪除範本同步設定。

1. 在範本詳細資料頁面中，選擇 [同步] 索引標籤。
2. 在同步詳細資料區段中，選擇「中斷連線」

AWS CLI

下列範例指令和回應顯示如何使用刪除已同步的範本設定。 AWS CLI

執行下列命令。

```
$ aws proton delete-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT"
```

回應如下。

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
```

```
    }
}
```

服務同步配置

透過服務同步，您可以使用 Git 設定和部署AWS Proton服務。您可以使用服務同步，透過 Git 儲存庫中定義的組態來管理AWS Proton服務的初始部署和更新。透過 Git，您可以使用版本追蹤和提取要求等功能來設定、管理和部署您的服務。服務同步結合AWS Proton和 Git 可協助您佈建透過AWS Proton範本定義和管理的標準化基礎結構。它會管理 Git 儲存庫中的服務定義，並減少工具切換。與單獨使用 Git 相比，中的範本和部署的標準化可AWS Proton協助您減少管理基礎結構的時間。 AWS Proton還為開發人員和平台團隊提供更高的透明度和可稽核性。

AWS Proton行動檔案

此`proton-ops`檔案會定義在何處AWS Proton尋找用來更新服務執行個體的規格檔案。它也會定義更新服務執行個體的順序，以及何時將變更從一個執行個體升級到另一個執行個體。

該文`proton-ops`件支持使用 spec 文件或在鏈接存儲庫中找到的多個規格文件來同步服務實例。您可以透過在`proton-ops`檔案中定義同步區塊來執行此操作，如下列範例所示。

範例。 /配置/質子操作劑:

```
sync:
  services:
      frontend-svc:
          alpha:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          beta:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          gamma:
              branch: pre-prod
              spec: ./frontend-svc/pre-prod/frontend-spec.yaml
          prod-one:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-two:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

```
        prod-three:
            branch: prod
            spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

在上述範例中，frontend-svc是服務名稱alpha，且beta、gamma、prod-oneprod-two、和prod-three是執行個體。

spec檔案可以是proton-ops檔案中定義的所有例證或例證的子集。但是，至少它必須在分支中定義實例以及與其同步的規範。如果proton-ops檔案中沒有定義執行個體，而且具有特定分支和spec檔案位置，服務同步將不會建立或更新這些執行個體。

下列範例顯示檔spec案的外觀。請記住，該文proton-ops件是從這些spec文件同步的。

範例**./frontend-svc/test/frontend-spec.yaml**：

```
proton: "ServiceSpec"
instances:
- name: "alpha"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "beta"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

範例**./frontend-svc/pre-prod/frontend-spec.yaml**：

```
proton: "ServiceSpec"
instances:
- name: "gamma"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
```

```
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

範例**./frontend-svc/prod/frontend-spec-second.yaml**：

```
proton: "ServiceSpec"
instances:
- name: "prod-one"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-two"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-three"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

如果執行個體沒有同步處理，而且在嘗試同步時仍然存在問題，呼叫 GetServiceInstanceSyncStatusAPI 可能有助於解決問題。

> **Note**
>
> 使用服務同步的客戶仍受到AWS Proton限制的限制。

阻斷劑

透過使用服務同步來同步AWS Proton服務，您可以更新服務規格，並從 Git 儲存庫建立和更新服務執行個體。不過，有時候您可能需要透過或手動更新服務或執行個體AWS CLI。AWS Management Console

AWS Proton協助避免覆寫您透過AWS Management Console或所做的任何手動變更AWS CLI，例如更新服務執行個體或刪除服務執行個體。若要達成此目標，請在偵測到手AWS Proton動變更時停用服務同步處理，藉此自動建立服務同步封鎖程式。

要獲取與服務相關聯的所有阻止程序，您必須對與服務`serviceInstance`相關的每個阻止程序執行以下操作：

- 只`getServiceSyncBlockerSummary`使用`serviceName`.
- 使用`serviceName`和呼叫 `getServiceSyncBlockerSummary` API `serviceInstanceName`。

這將返回最新的阻止程序列表以及與它們相關的狀態。如果有任何阻止程序被標記`resolvedReason`為 ACTIVE，則必須通過使用`blockerId`和調用每個阻止程序的 `UpdateServiceSyncBlocker` API 來解決它們。

如果您手動更新或建立服務執行個體，請在服務執行個體上建AWS Proton立服務同步封鎖程式。AWS Proton繼續同步所有其他服務實例，但禁用此服務實例的同步，直到阻止程序解決為止。如果您從服務刪除服務執行個體，請在服務上AWS Proton建立服務同步封鎖程式。這樣可以防AWS Proton止同步任何服務實例，直到阻止程序解決為止。

擁有所有活動阻止程序後，您必須通過使用`blockerId`和`resolvedReason`調用每個活動阻止程序的 `UpdateServiceSyncBlocker` API 來解決它們。

使用AWS Management Console，您可以瀏覽至AWS Proton並選擇 [服務同步] 索引標籤，以判斷服務同步是否已停用。如果服務或服務執行個體遭到封鎖，就會出現「啟用」按鈕。若要啟用服務同步，請選擇 [啟用]。

主題

建立服務同步設定

您可以使用主控台或建立服務同步處理組態AWS CLI。

AWS Management Console

1. 在 [選擇服務範本] 頁面上，選取範本並選擇 [設定]。
2. 在 [設定服務] 頁面的 [服務詳細資料] 區段中，輸入新的服務名稱。
3. (選用) 輸入服務的描述。
4. 在 [應用程式原始程式碼儲存庫] 區段中，選擇 [選擇連結的 Git 儲存庫] 以選取您已連結的儲存庫AWS Proton。如果您還沒有連結的儲存庫，請選擇「連結其他 Git 儲存庫」，然後依照建立儲存庫連結中的指示操作。
5. 對於「儲存庫」，請從清單中選擇原始程式碼儲存庫的名稱。
6. 對於「分支」，請從清單中選擇原始程式碼的儲存庫分支名稱。
7. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。
8. 選擇 下一步。
9. 在 [設定服務執行個體] 頁面的 [服務定義來源] 區段中，選取 [從 Git 同步您的服務]。
10. 在「服務定義檔案」區段中，如果AWS Proton要建立`proton-ops`檔案，請選取「我要 AWS Proton 建立檔案」。使用此選項，AWS Proton在您指定的位置建立`spec`和`proton-ops`檔案。選取 [我提供自己的檔案] 來建立您自己的 OPS 檔案。
11. 在 [服務定義儲存庫] 區段中，選擇 [選擇連結的 Git 儲存庫] 以選取您已連結的儲存庫AWS Proton。
12. 對於「存放庫名稱」，請從清單中選擇原始程式碼儲存庫的名稱。
13. 對於**`proton-ops`**文件分支，請從列表中選擇AWS Proton將放置 OPS 和 spec 文件的分支名稱。
14. 在「服務執行個體」區段中，系統會根據`proton-ops`檔案中的值自動填入每個欄位。
15. 選擇 [下一步] 並檢閱您的輸入。
16. 選擇 建立 。

AWS CLI

使用建立服務同步設定 AWS CLI

- 執行下列命令。

```
$ aws proton create-service-sync-config \
    --resource "service-arn" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
```

```
    --ops-file-branch "main" \
    --proton-ops-file "./configuration/custom-proton-ops.yaml" (optional)
```

回應如下下。

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

檢視服務同步的組態詳細資訊

您可以使用主控台或檢視服務同步的組態詳細資料資料AWS CLI。

AWS Management Console

使用主控台檢視服務同步的組態詳細資訊

1. 在導覽窗格中，選擇。
2. 若要檢視詳細資料，請選擇您為其建立服務同步組態的服務名稱。
3. 在服務的詳細資料頁面中，選取服務同步索引標籤，以檢視服務同步的組態詳細資料。

AWS CLI

使用取AWS CLI得同步的服務。

執行下列命令。

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

回應如下下。

```
{
```

```
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

使用取AWS CLI得服務同步狀態。

執行下列命令。

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

編輯服務同步化組態

您可以使用主控台或來編輯服務同步處理組態AWS CLI。

AWS Management Console

使用主控台編輯服務同步處理組態。

1. 在導覽窗格中，選擇。
2. 若要檢視詳細資料，請選擇您為其建立服務同步組態的服務名稱。
3. 在服務詳細資料頁面上，選擇 [服務同步] 索引標籤。
4. 在 [服務同步] 區段中，選擇 [編輯]。
5. 在 [編輯] 頁面上，更新您要編輯的資訊，然後選擇 [儲存]。

AWS CLI

下列範例命令和回應顯示如何使用編輯服務同步配置AWS CLI。

執行下列命令。

```
$ aws proton update-service-sync-config \
    --service-name "service name" \
```

```
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --ops-file "./configuration/custom-proton-ops.yaml"
```

回應如下下。

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

刪除服務同步設定

您可以使用主控台或來刪除服務同步組態AWS CLI。

AWS Management Console

使用主控台刪除服務同步處理組態

1. 在服務詳細資料頁面上，選擇服務同步索引標籤。
2. 在 [服務同步詳細資料] 區段中，選擇 [中斷連線] 以中斷儲存庫 在您的儲存庫中斷連線後，我們將不再從該儲存庫同步服務。

AWS CLI

下列範例命令和回應顯示如何使用刪AWS CLI除服務同步的組態。

執行下列命令。

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

回應如下下。

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> 服務同步不會刪除服務實例。它只會刪除組態。

AWS Proton 環境

對於AWS Proton，環境代表一組共享資源和策略AWS Proton 服務部署到. 它們可以包含預期共享的任何資源AWS Proton服務實例。這些資源可以包括 VPC、叢集和共用負載平衡器或 API 閘道。一個AWS Proton必須先建立環境，才能將服務部署至該環境。

本節說明如何使用建立、檢視、更新和刪除作業來管理環境。如需 > 其他資訊，請參閱該AWS Proton服務 API 參考資料。

主題

IAM 角色

使用AWS Proton，您可以為您擁有和管理的AWS資源提供 IAM 角色和AWS KMS金鑰。這些稍後會套用至開發人員擁有和管理的資源，並由其使用。您可以建立 IAM 角色來控制開發人員團隊對AWS Proton API 的存取權。

AWS Proton 服務角色

建立新環境時，您會提供相關的 IAM 服務角色。角色包含更新環境範本和服務範本中定義之所有已佈建基礎結構所需的所有權限。如需角色範例，請參閱AWS Proton 佈建使用的服務角色 AWS CloudFormation。如果您使用環境帳戶連線和環境帳戶，則會在選取的環境帳戶中建立角色。如需詳細資訊，請參閱在一個帳戶中建立環境，並在另一個帳戶中佈建及環境帳戶連線。

您如何提供此服務角色以及擔任該角色的人員，取決於您環境的佈建方法。

- AWS-受管理的佈建 — 您可以AWS Proton在建立環境時直接提供角色，或透過帳戶連線間接提供角色。 AWS Proton承擔相關帳戶中的角色，以提供環境和服務基礎設施。

- 自我管理佈建 — 當提取請求 (PR) 觸發佈建動作時，您有責任將佈建自動化配置為使用適當的憑證承擔適當的角色。如需擔任角色的 GitHub 動作範例，請參閱動作的「設定AWS認證」動 GitHub 作說明文件中的「假設角色」。

如需佈建範本的詳細資訊，請參閱the section called “佈建方法”。

建立環境

學習如何創造AWS Proton環境。

您可以建立AWS Proton環境有兩種方式之一：

- 使用建立、管理及佈建標準環境標準環境樣板。AWS Proton為您的環境佈建基礎結構。
- 連接AWS Proton至客戶管理的基礎架構，透過使用客戶管理的環境範本。您在以外佈建自己的共用資源AWS Proton，然後您提供佈建輸出AWS Proton可以使用。

建立環境時，您可以選擇數種佈建方法之一。

- AWS管理佈建— 在單一帳戶中建立、管理和佈建環境。AWS Proton提供您的環境。

  此方法僅支持CloudFormation基礎設施代碼（IaC）模板。
- AWS管理佈建到另一個帳戶— 在單一管理帳戶中，建立並管理在具有環境帳戶連線的另一個帳戶中佈建的環境。AWS Proton在其他帳戶中佈建您的環境。如需詳細資訊，請參閱 在一個帳戶中建立環境，並在另一個帳戶中佈建 及 環境帳戶連線。

  此方法僅支持CloudFormation合家歡的模板。
- 自我管理佈建—AWS Proton將佈建提取請求提交至具有您自己的佈建基礎結構的連結存放庫。

  此方法僅支持地形 IaC 模板。
- CodeBuild服務開通—AWS Proton使用AWS CodeBuild以執行您提供的殼層命令。您的命令可以讀取輸入AWS Proton提供並負責佈建或取消佈建基礎架構，以及產生輸出值。此方法的範本組合包括資訊清單檔案中的命令，以及這些命令可能需要的任何程式、指令碼或其他檔案。

  作為使用的示例CodeBuild佈建時，您可以包含使用AWS Cloud Development Kit (AWS CDK)提供AWS資源，以及安裝 CDK 並運行 CDK 代碼的清單。

  如需詳細資訊，請參閱the section called “CodeBuild 捆綁”。

> **Note**
>
> 您可以使用CodeBuild使用環境和服務進行佈建。目前您無法以這種方式佈建元件。

同AWS託管配置（在同一帳戶和另一個帳戶中），AWS Proton直接呼叫以佈建您的資源。

透過自我管理佈建，AWS Proton提出提取要求，以提供 IaC 引擎用來佈建資源的編譯 IaC 檔案。

如需詳細資訊，請參閱 the section called “佈建方法”、the section called “模板捆綁” 及 the section called “環境綱要需求”。

主題

- 在同一帳戶中建立和佈建標準環境
- 在一個帳戶中建立環境，並在另一個帳戶中佈建
- 使用自我管理的佈建建立和佈建環境

在同一帳戶中建立和佈建標準環境

使用控制台或AWS CLI在單一帳戶中建立和佈建環境。佈建由管理AWS。

AWS Management Console

使用主控台在單一帳戶中建立和佈建環境

1. 在AWS Proton安慰，選擇環境。
2. 選擇 Create environment (建立環境)。
3. 在選擇環境範本頁面上，選擇一個模板，然後選擇配置。
4. 在配置環境頁面，在佈建區段中，選擇AWS管理佈建。
5. 在部署帳戶區段中，選擇這個AWS 帳戶。
6. 在配置環境頁面，在環境設定區段中，輸入環境名稱。
7. (選擇性) 輸入環境的說明。
8. 在環境角色」區段中，選取AWS Proton您建立為其一部分的服務角色設定 AWS Proton 服務角色。
9. (選擇性) 在元件角色」段落中，選取可讓直接定義的元件在環境中執行的服務角色，並縮減其可佈建之資源的範圍。如需詳細資訊，請參閱元件。

10. (選擇性) 在标签區段中，選擇新增標籤並輸入金鑰和值以建立客戶管理的標籤。
11. 選擇 下一步。
12. 在設定環境自訂設定頁面中，您必須輸入`required`參數。您可以輸入`optional`參數或在給定時使用默認值。
13. 選擇下一步並檢查您的輸入。
14. 選擇 建立 。

    檢視環境詳細資訊和狀態，以及AWS適用於您環境的受管理標籤和客戶管理標籤。
15. 在導覽窗格中，選擇 Environments (環境)。

    新頁面會顯示環境清單，以及狀態和其他環境詳細資訊。

AWS CLI

使用AWS CLI在單一帳戶中建立和佈建環境。

若要建立環境，請指定AWS Proton服務角色ARN，規格文件的路徑，環境名稱，環境模板 ARN，主要和次要版本以及描述（可選）。

下面的例子顯示YAML格式化規格檔案，指定在環境範本結構描述檔案中定義的兩個輸入值。您可以使用`get-environment-template-minor-version`用於檢視環境範本結構描述的指令。

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

執行下列命令來建立環境。

```
$ aws proton create-environment \
    --name "MySimpleEnv" \
    --template-name simple-env \
    --template-major-version 1 \
    --proton-service-role-arn "arn:aws:iam::123456789012:role/AWSProtonServiceRole"
 \
    --spec "file://env-spec.yaml"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateName": "simple-env"
    }
}
```

建立新環境之後，您可以檢視清單AWS和客戶管理的標籤，如下面的示例命令所示。AWS Proton自動產生AWS為您管理的標籤。您也可以修改和建立客戶管理的標籤AWS CLI。如需詳細資訊，請參閱AWS Proton資源和標記。

命令：

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv"
```

在一個帳戶中建立環境，並在另一個帳戶中佈建

使用控制台或AWS CLI在管理帳戶中建立標準環境，在另一個帳戶中佈建環境基礎結構。佈建由管理AWS。

使用主控台或 CLI 之前，請先完成下列步驟。

1. 識別AWS 帳戶管理和環境帳戶的 ID，並複製它們以供日後使用。
2. 在環境帳戶中，建立AWS Proton具有要建立之環境之最低權限的服務角色。如需詳細資訊，請參閱AWS Proton 佈建使用的服務角色 AWS CloudFormation。

AWS Management Console

使用主控台在一個帳戶中建立環境，並在另一個帳戶中進行佈建。

1. 在環境帳戶中，建立環境帳戶連線，並使用它來傳送連線至管理帳戶的要求。
   a. 在AWS Proton安慰，選擇環境帳戶連線在導航窗格中。

b. 在環境帳戶連線頁面上，選擇請求連接。

> **Note**
>
> 驗證列在帳戶 ID環境帳號連線頁面標題與您預先識別的環境帳號 ID 相符。

c. 在請求連接頁面，在環境角色區段中，選取現有服務角色以及您為環境建立的服務角色名稱。

d. 在連線至管理帳戶」區段中，輸入管理帳戶識別碼和一個環境名稱為您的AWS Proton環境。複製名稱以備日後使用。

e. 選擇請求連接在頁面的右下角。

f. 您的請求在中顯示為待處理傳送至管理帳戶的環境連線表和模式顯示了如何接受來自管理帳戶的請求。

2. 在管理帳戶中，接受從環境帳戶連線的要求。

a. 登入您的管理帳戶並選擇環境帳戶連線在AWS Proton控制台。

b. 在環境帳戶連線頁面，在環境帳戶連線要求表中，選取環境帳戶 ID 與您預先識別的環境帳戶 ID 相符的環境帳戶連線。

> **Note**
>
> 驗證列在帳戶 ID環境帳號連線頁面標題與您預先識別的管理帳戶 ID 相符。

c. 選擇 Accept (接受)。狀態從「待處理」變更為「已連線」。

3. 在管理帳戶中，建立環境。

a. 在導覽窗格中，選擇環境模板。

b. 在環境模板頁面上，選擇建立環境範本。

c. 在選擇環境範本頁面上，選擇環境範本。

d. 在配置環境頁面，在佈建區段中，選擇AWS管理佈建。

e. 在部署帳戶區段中，選擇另一個AWS帳戶;。

f. 在環境詳情」區段中，選取您的環境帳號連線和環境名稱。

g. 選擇 下一步。

h. 填寫表格並選擇下一步直到你到達檢閱和建立頁面。

i. 檢閱並選擇建立環境。

AWS CLI

使用AWS CLI在一個帳戶中創建環境並在另一個帳戶中進行配置。

在環境帳戶中，創建一個環境帳戶連接並請求通過運行以下命令進行連接。

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role" \
    --management-account-id "111111111111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

在管理帳戶中，執行下列命令以接受環境帳戶連線要求。

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
```

```
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

執行下列命令來檢視您的環境帳戶連線。

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

在管理帳戶中，執行下列命令來建立環境。

```
$ aws proton create-environment \
    --name "simple-env-connected" \
    --template-name simple-env-template \
    --template-major-version "1" \
    --template-minor-version "1" \
    --spec "file://simple-env-template/specs/original.yaml" \
```

```
    --environment-account-connection-id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-
connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
        "templateName": "simple-env-template"
    }
}
```

使用自我管理的佈建建立和佈建環境

當您使用自我管理佈建時，AWS Proton將佈建提取請求提交至具有您自己的佈建基礎結構的連結存放庫。拉取請求會啟動您自己的工作流程，該工作流程AWS服務; 提供基礎設施。

自我管理佈建考量：

- 建立環境之前，請先設定自我管理佈建的儲存庫資源目錄。如需詳細資訊，請參閱AWS Proton 基礎架構即程式碼檔。
- 建立環境之後，AWS Proton等待接收有關基礎結構佈建狀態的非同步通知。您的佈建程式碼必須使用AWS Proton `NotifyResourceStateChange`將這些非同步通知傳送至的 APIAWS Proton。

您可以在主控台中使用自我管理佈建，或與AWS CLI。下列範例說明如何搭配 Terraform 使用自我管理的佈建。

AWS Management Console

使用主控台建立使用自我管理佈建的 Terraform 環境。

1. 在AWS Proton安慰，選擇環境。
2. 選擇 Create environment (建立環境)。

3. 在選擇環境範本頁面上，選擇一個地形範本，然後選擇配置。
4. 在配置環境頁面，在佈建區段中，選擇自我管理佈建。
5. 在啟動設定儲存區區段：

   a. 如果您還沒有將您的佈建存放庫連結至AWS Proton，選擇新儲存庫，選擇其中一個儲存區域提供者，然後針對CodeStar連接，選擇您的連線之一。

   > **Note**
   >
   > 如果您尚未連線到相關的儲存庫提供者帳戶，請選擇添加一個新的CodeStar連接。然後，建立連線，然後選擇 [重新整理] 按鈕CodeStar連接菜單。現在，您應該可以在菜單中選擇新的連接。

   如果您已將存儲庫鏈接到AWS Proton，選擇現有儲存庫。

   b. 對於儲存庫名稱，選擇一個存放庫。下拉式功能表會顯示連結的儲存庫現有儲存庫或提供者帳戶中的存儲庫列表新儲存庫。

   c. 對於分行名稱」下，選擇其中一個儲存庫分支。
6. 在環境設定區段中，輸入環境名稱。
7. (選擇性) 輸入環境的說明。
8. (選擇性) 在标签區段中，選擇新增標籤並輸入金鑰和值以建立客戶管理的標籤。
9. 選擇 下一步。
10. 在設定環境自訂設定頁面中，您必須輸入`required`參數。您可以輸入`optional`參數或在給定時使用默認值。
11. 選擇下一步並檢查您的輸入。
12. 選擇創建發送提取請求。

    - 如果您核准提取要求，則部署正在進行中。
    - 如果您拒絕提取請求，則會取消環境建立。
    - 如果提取請求逾時，則建立環境不是完成。
13. 檢視環境詳細資訊和狀態，以及AWS適用於您環境的受管理標籤和客戶管理標籤。
14. 在導覽窗格中，選擇 Environments (環境)。

    新頁面會顯示環境清單，以及狀態和其他環境詳細資訊。

AWS CLI

當您使用自我管理的佈建來建立環境時，您加該provisioningRepository參數並省略ProtonServiceRoleArn和environmentAccountConnectionId參數。

使用AWS CLI創建具有自我管理配置的 Terraform 環境。

1. 建立環境並將提取要求傳送至儲存庫以供檢閱和核准。

   下面的例子顯示YAML格式化規格檔案，根據環境範本結構描述檔案定義兩個輸入的值。您可以使用get-environment-template-minor-version用於檢視環境範本結構描述的指令。

   規格:

   ```
   proton: EnvironmentSpec
   spec:
     ssm_parameter_value: "test"
   ```

   執行下列命令來建立環境。

   ```
   $ aws proton create-environment \
       --name "pr-environment" \
       --template-name "pr-env-template" \
       --template-major-version "1" \
       --provisioning-repository="branch=main,name=myrepos/env-
   repo,provider=GITHUB" \
       --spec "file://env-spec.yaml"
   ```

   回應內容：>

   ```
   {
       "environment": {
           "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
   environment",
           "createdAt": "2021-11-18T17:06:58.679000+00:00",
           "deploymentStatus": "IN_PROGRESS",
           "lastDeploymentAttemptedAt": "2021-11-18T17:06:58.679000+00:00",
           "name": "pr-environment",
           "provisioningRepository": {
   ```

```
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateName": "pr-env-template"
    }
```

2. 檢閱請求。

   - 如果您核准請求，則正在進行佈建。
   - 如果您拒絕請求，則會取消環境建立。
   - 如果提取請求逾時，則建立環境不是完成。

3. 以非同步方式提供佈建狀態給AWS Proton。下面的例子通知AWS Proton成功佈建的。

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

檢視環境資料

您可以使用AWS Proton主控台或AWS CLI。

AWS Management Console

您可以檢視包含詳細資料的環境清單，以及具有詳細資料的個別環境，方法是使AWS Proton安慰。

1. 若要檢視您的環境清單，請選擇環境在導航窗格中。
2. 若要檢視詳細資料，請選擇環境的名稱。

   檢視您的環境詳細資料。

AWS CLI

使用AWS CLI 得到或者名單環境詳細資料。

執行以下命令：

```
$ aws proton get-environment \
    --name "MySimpleEnv"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "lastDeploymentSucceededAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\nspec:\n  my_sample_input: \"the first\"\n
 my_other_sample_input: \"the second\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

更新環境

如果AWS Proton環境與環境帳戶連接相關聯，不要更新或包含protonServiceRoleArn用於更新或連接到環境帳戶連接的參數。

只有在下列兩項都成立時，您才能更新為新的環境帳戶連線：

- 環境帳戶連線是在建立目前環境帳戶連線的相同環境帳戶中建立的。
- > 環境帳戶連線與目前環境相關聯。

如果環境不是與環境帳戶連接相關聯，不要更新或包含environmentAccountConnectionId參數。

您可以更新environmentAccountConnectionId或者protonServiceRoleArn參數和值。您無法同時更新兩者。

如果您的環境使用自我管理佈建，不要更新provisioning-repository參數和忽略該environmentAccountConnectionId和protonServiceRoleArn參數。

更新環境有四種模式，如下列清單所述。使用時AWS CLI，該deployment-type字段定義模式。使用主控台時，這些模式會對應至編輯,更新,更新次要，以及主要更新從下拉式清單的動作動作。

NONE

在此模式下，部署沒有發生。只會更新要求的中繼資料參數。

CURRENT_VERSION

在此模式下，環境會部署並使用您提供的新規格進行更新。只會更新要求的參數。不要使用此參數時包括次要或主要版本參數deployment-type。

MINOR_VERSION

在此模式下，依預設，會使用目前主要版本的已發佈、建議 (最新) 次要版本來部署和更新環境。您也可以指定目前正在使用的主要版本的不同次要版本。

MAJOR_VERSION

在此模式下，依預設，會使用目前範本的已發佈、建議 (最新) 主要和次要版本來部署和更新環境。您也可以指定高於使用中的主要版本和次要版本 (選用) 的不同主要版本。

主題

更新一個AWS受管佈建環境

標準佈建僅由以下項目提供的環境支援AWS CloudFormation。

使用控制台或AWS CLI以更新您的環境。

AWS Management Console

如下列步驟所示，使用主控台更新環境。

1. 選擇以下 2 個步驟之一。
   a. 在環境列表中。
      i. 在AWS Proton安慰，選擇環境。
      ii. 在環境清單中，選擇您要更新之環境左側的圓鈕。
   b. 在控制台環境詳細信息頁面中。
      i. 在AWS Proton安慰，選擇環境。
      ii. 在環境清單中，選擇您要更新之環境的名稱。
2. 選擇接下來 4 個步驟中的一個來更新您的環境。
   a. 進行不需要環境部署的編輯。
      i. 例如，若要變更描述。

         選擇 編輯 。
      ii. 填寫表格並選擇下一步。
      iii. 查看您的編輯並選擇更新。
   b. 僅對中繼資料輸入進行更新。
      i. 選擇動作然後更新。
      ii. 填寫表格並選擇編輯。
      iii. 填寫表格並選擇下一步直到你到達評論頁面。
      iv. 查看您的更新並選擇更新。
   c. 更新其環境範本的新次要版本。
      i. 選擇動作然後更新次要。
      ii. 填寫表格並選擇下一步。
      iii. 填寫表格並選擇下一步直到你到達評論頁面。
      iv. 查看您的更新並選擇更新。

d. 要更新其環境模板的新主要版本。

i. 選擇動作然後主要更新。

ii. 填寫表格並選擇下一步。

iii. 填寫表格並選擇下一步直到你到達評論頁面。

iv. 查看您的更新並選擇更新。

AWS CLI

使用AWS Proton AWS CLI將環境更新為新的次要版本。

執行下列命令來更新您的環境：

```
$ aws proton update-environment \
        --name "MySimpleEnv" \
        --deployment-type "MINOR_VERSION" \
        --template-major-version "1" \
        --template-minor-version "1" \
        --proton-service-role-arn arn:aws:iam::123456789012:role/service-
role/ProtonServiceRole \
        --spec "file:///spec.yaml"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:29:55.472000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

執行下列命令以取得並確認狀態：

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "MySimpleEnv",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

更新自我管理的佈建環境

只有使用 Terraform 佈建的環境才支援自我管理的佈建。

使用控制台或AWS CLI以更新您的環境。

AWS Management Console

如下列步驟所示，使用主控台更新環境。

1. 選擇以下 2 個步驟之一。

    a. 在環境列表中。

        i. 在AWS Proton安慰，選擇環境。

        ii. 在環境清單中，選擇您要更新之環境範本左側的圓鈕。

b. 在控制台環境詳細信息頁面中。

i. 在AWS Proton安慰，選擇環境。

ii. 在環境清單中，選擇您要更新之環境的名稱。

2. 選擇接下來 4 個步驟中的一個來更新您的環境。

a. 進行不需要環境部署的編輯。

i. 例如，若要變更描述。

選擇 編輯 。

ii. 填寫表格並選擇下一步。

iii. 查看您的編輯並選擇更新。

b. 僅對中繼資料輸入進行更新。

i. 選擇動作然後更新。

ii. 填寫表格並選擇編輯。

iii. 填寫表格並選擇下一步直到你到達評論頁面。

iv. 查看您的更新並選擇更新。

c. 更新其環境範本的新次要版本。

i. 選擇動作然後更新次要。

ii. 填寫表格並選擇下一步。

iii. 填寫表格並選擇下一步直到你到達評論頁面。

iv. 查看您的更新並選擇更新。

d. 要更新其環境模板的新主要版本。

i. 選擇動作然後主要更新。

ii. 填寫表格並選擇下一步。

iii. 填寫表格並選擇下一步直到你到達評論頁面。

iv. 查看您的更新並選擇更新。

AWS CLI

使用AWS CLI將 Terraform 環境更新為具有自我管理佈建的新次要版本。

1. 執行下列命令來更新您的環境：

```
$ aws proton update-environment \
    --name "pr-environment" \
    --deployment-type "MINOR_VERSION" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --provisioning-repository "branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec-mod.yaml"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:09:15.745000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "pr-env-template"
    }
}
```

2. 執行下列命令以取得並確認狀態：

```
$ aws proton get-environment \
```

```
    --name "pr-environment"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:25:41.998000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "spec": "proton: EnvironmentSpec\nspec:\n   ssm_parameter_value: \"test
\"\n ssm_another_parameter_value: \"update\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "pr-env-template"
    }
}
```

3. 檢閱由傳送的提取要求AWS Proton。

   - 如果您核准請求，則正在進行佈建。
   - 如果您拒絕請求，則會取消環境建立。
   - 如果提取請求逾時，則環境建立不完成。

4. 將佈建狀態提供給AWS Proton。

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

取消進行中的環境部署

您可以嘗試取消環境更新部署，如果deploymentStatus在IN_PROGRESS。AWS Proton嘗試取消部署。成功取消不是保證。

當您取消更新部署時，AWS Proton嘗試取消部署，如下列步驟所列。

同AWS-管理佈建，AWS Proton執行以下操作：

- 將部署狀態設定為CANCELLING。
- 停止進行中的部署，並刪除部署在下列情況下建立的任何新資源IN_PROGRESS。
- 將部署狀態設定為CANCELLED。
- 將資源的狀態還原為開始部署之前的狀態。

透過自我管理佈建，AWS Proton執行以下操作：

- 嘗試關閉提取請求以防止將更改合併到存儲庫中。
- 將部署狀態設定為CANCELLED如果拉請求已成功關閉。

如需如何取消環境部署的指示，請參閱CancelEnvironmentDeployment在AWS ProtonAPI 參考資料。

您可以使用主控台或 CLI 取消進行中的環境。

AWS Management Console

使用主控台取消環境更新部署，如下列步驟所示。

1. 在AWS Proton安慰，選擇環境在導航窗格中。
2. 在環境清單中，選擇您要取消之部署更新之環境的名稱。
3. 如果您的更新部署狀態為進行中，在「環境詳細資訊」頁面中，選擇動作然後取消部署。
4. 強制回應會提示您確認要取消。選擇取消部署。
5. 您的更新部署狀態設定為取消然後已取消完成取消。

AWS CLI

使用AWS Proton AWS CLI，以取消對新次要版本 2 的 IN_PROGRESS 環境更新部署。

等待條件包含在用於此範例的範本中，以便在更新部署成功之前開始取消。

執行下列命令以取消更新：

```
$ aws proton cancel-environment-deployment \
        --environment-name "MySimpleEnv"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

執行下列命令以取得並確認狀態：」

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
```

```
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

刪除環境

您可以刪除AWS Proton環境:使用AWS Proton主控台或AWS CLI。

> **Note**
>
> 您無法刪除具有任何關聯元件的環境。若要刪除此類環境，您應該先刪除環境中執行的所有元件。如需有關元件的更多資訊，請參閱元件。

AWS Management Console

使用主控台刪除環境，如下列兩個選項所述。

在環境列表中。

1. 在AWS Proton安慰，選擇環境。
2. 在環境清單中，選取您要刪除之環境左側的圓鈕。
3. 選擇動作然後刪除。
4. 強制回應會提示您確認刪除動作。
5. 按照說明進行操作並選擇是，刪除。

在環境詳細資訊頁面中。

1. 在AWS Proton安慰，選擇環境。
2. 在環境清單中，選擇您要刪除的環境名稱。

3. 在「環境詳細資訊」頁面中，選擇動作然後刪除。
4. 強制回應會提示您確認要刪除。
5. 按照說明進行操作並選擇是，刪除。

AWS CLI

使用AWS CLI以刪除環境。

不要如果服務或服務執行個體已部署至環境，請刪除環境。

執行以下命令：

```
$ aws proton delete-environment \
    --name "MySimpleEnv"
```

回應：

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "DELETE_IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

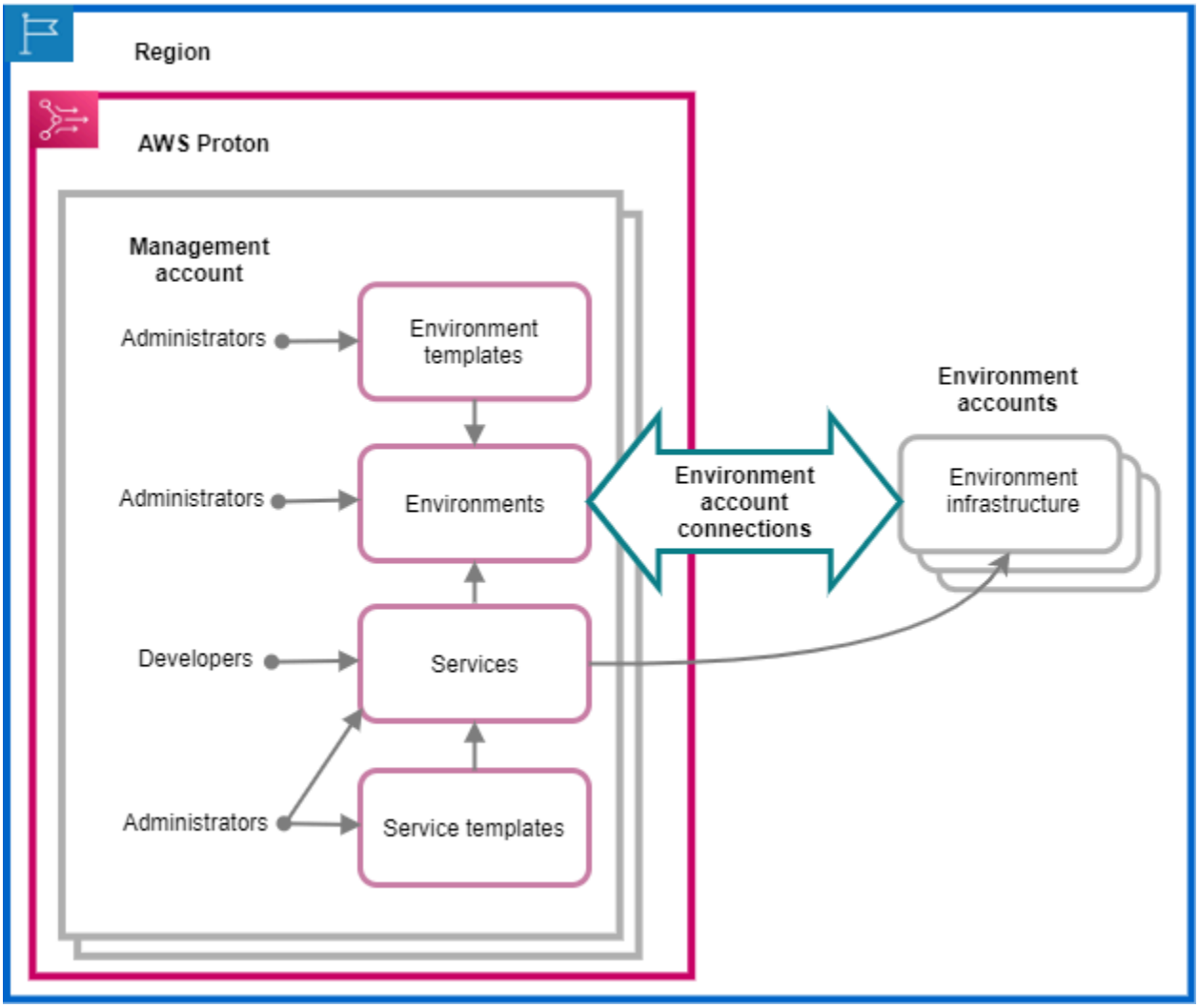
環境帳戶連線

概觀

瞭解如何建立和管理AWS Proton環境在一個帳戶中，並在另一個帳戶中佈建其基礎結構資源。這有助於提高大規模的可見性和效率。環境帳戶連線僅支援標準佈建AWS CloudFormation基礎設施即程式碼。

> **Note**
>
> 本主題中的資訊與以下項目設定的環境有關AWS管理佈建。使用環境配置自我管理佈建,AWS Proton不會直接佈建您的基礎結構。相反地，它會將提取要求 (PR) 傳送到您的儲存庫以供佈建。您有責任確保您的自動化程式碼具有正確的身分和角色。
> 如需佈建方法的詳細資訊，請參閱the section called “佈建方法”。

術語



同AWS Proton 環境帳戶連線，您可以建立AWS Proton從一個帳戶環境，並在另一個帳戶中佈建其基礎設施。

管理帳戶

作為管理員，您可以在其中創建一個單一帳戶AWS Proton在另一個環境中佈建基礎結構資源環境帳戶。

環境帳戶

當您建立環境基礎結構時，佈建環境基礎結構的帳戶AWS Proton另一個帳戶中的環境。

環境帳號連線

之間的安全雙向連接管理帳戶和一個環境帳戶。它維護授權和權限，如以下各節進一步所述。

當您在特定區域的環境帳戶中建立環境帳戶連線時，只有相同區域中的管理帳戶可以看到並使用環境帳戶連線。這意味著AWS Proton在管理帳戶中建立的環境，以及在環境帳戶中佈建的環境基礎結構必須位於相同區域。

環境帳戶連線考量

- 對於要在環境帳戶中佈建的每個環境，您需要一個環境帳戶連線。
- 如需環境帳戶連線配額的相關資訊，請參閱AWS Proton 配額。

標記

在環境帳戶中，使用主控台或AWS CLI檢視和管理環境帳戶連線客戶管理的標籤。AWS受管理標籤不是為環境帳戶連接生成。如需詳細資訊，請參閱標記。

在一個帳戶中建立環境，並在另一個帳戶中佈建其基礎結構

若要從單一管理帳戶建立及佈建環境，請為您打算建立的環境設定環境帳戶。

在環境帳戶中啟動並創建連接。

在環境帳戶中，建立AWS Proton服務角色的範圍僅限於佈建環境基礎結構資源所需的權限。如需詳細資訊，請參閱AWS Proton 佈建使用的服務角色 AWS CloudFormation。

然後，建立環境帳戶連線要求並傳送至您的管理帳戶。當請求被接受時，AWS Proton可以使用關聯的 IAM 角色，該角色允許在關聯的環境帳戶中佈建環境資源。

在管理帳戶中，接受或拒絕環境帳戶連線。

在管理帳戶中，接受或拒絕環境帳戶連線要求。你不能從管理帳戶刪除環境帳戶連線。

如果您接受請求，AWS Proton可以使用允許在關聯環境帳戶中佈建資源的關聯 IAM 角色。

環境基礎結構資源會佈建在關聯的環境帳戶中。您只能使用AWS Proton從您的管理帳戶存取和管理您的環境及其基礎架構資源的 API。如需詳細資訊，請參閱 在一個帳戶中建立環境，並在另一個帳戶中佈建 及 更新環境。

在您拒絕要求之後，您不能接受或使用拒絕的環境帳戶連線。

> **Note**
>
> 你不能拒絕連線至環境的環境帳戶連線。若要拒絕環境帳戶連線，您必須先刪除關聯的環境。

在環境帳戶中，存取佈建的基礎結構資源。

在環境帳戶中，您可以檢視和存取佈建的基礎結構資源。例如，您可以使用CloudFormation如有需要，可監視和清理堆疊的 API 動作。你不能使用AWS Proton用於存取或管理AWS Proton用來佈建基礎結構資源的環境。

在環境帳戶中，您可以刪除已在環境帳戶中建立的環境帳戶連線。你不能接受或拒絕它們。如果您刪除正在使用的環境帳戶連線AWS Proton環境，AWS Proton在接受環境帳戶和具名環境的新環境連接之前，將無法管理環境基礎結構資源。您負責清理沒有環境連線的已佈建資源。

使用主控台或 CLI 管理環境帳戶連線

您可以使用主控台或 CLI 來建立和管理環境帳戶連線。

AWS Management Console

使用主控台建立環境帳戶連線，並將要求傳送至管理帳戶，如下列步驟所示。

1. 決定您計劃在管理帳戶中建立的環境名稱，或選擇需要環境帳戶連線的現有環境名稱。
2. 在環境帳戶中，AWS Proton安慰，選擇環境帳戶連線在導航窗格中。
3. 在環境帳戶連線頁面上，選擇請求連接。

> **Note**
>
> 驗證中列出的帳戶 ID環境帳號連線頁面標題。請確定它符合您要在其中佈建具名環境之環境帳戶的帳戶 ID。

4. 在請求連接頁面：

   a. 在連線至管理帳戶」區段中，輸入管理帳戶識別碼和環境名稱您在步驟 1 中輸入的。

   b. 在環境角色區段中，選擇新服務角色和AWS Proton會自動為您建立新角色。或者，選擇現有服務角色以及您先前建立的服務角色名稱。

   > **Note**
   >
   > 角色AWS Proton自動為您創建具有廣泛的權限。建議您將角色範圍縮小為佈建環境基礎結構資源所需的權限。如需詳細資訊，請參閱AWS Proton 佈建使用的服務角色 AWS CloudFormation。

   c. (選擇性) 在标签區段中，選擇新增標籤為您的環境帳戶連線建立客戶管理的標籤。

   d. 選擇請求連接。

5. 您的請求在中顯示為待處理傳送至管理帳戶的環境連線表格和模式可讓您知道如何接受來自管理帳戶的請求。

接受或拒絕環境帳戶連線要求。

1. 在管理帳戶中，AWS Proton安慰，選擇環境帳戶連線在導航窗格中。
2. 在環境帳戶連線頁面，在環境帳戶連線要求表格中，選擇要接受或拒絕的環境連線要求。

   > **Note**
   >
   > 驗證中列出的帳戶 ID環境帳號連線頁面標題。請確定它符合要拒絕之環境帳戶連線關聯之管理帳戶的帳戶識別碼。拒絕此環境帳戶連線之後，您不能接受或使用拒絕的環境帳戶連線。

3. 選擇拒絕或者接受。

   - 如果您已選取拒絕，狀態變更為待決至拒絕。
   - 如果您已選取接受，狀態變更為待決至連線。

刪除環境帳戶連線。

1. 在環境帳戶中，AWS Proton安慰，選擇環境帳戶連線在導航窗格中。

> **Note**
>
> 驗證中列出的帳戶 ID環境帳號連線頁面標題。請確定它符合要拒絕之環境帳戶連線關聯之管理帳戶的帳戶識別碼。刪除此環境帳號連線之後，AWS Proton 不能管理環境帳戶中的環境基礎設施資源。只有在管理帳戶接受環境帳戶和具名環境的新環境帳戶連線之後，才能對其進行管理。

2. 在環境帳戶連線頁面，在傳送連線至管理帳戶的要求區段中，選擇刪除。
3. 強制回應會提示您確認要刪除。選擇 刪除 。

AWS CLI

決定您計劃在管理帳戶中建立的環境名稱，或選擇需要環境帳戶連線的現有環境名稱。

在環境帳戶中建立環境帳戶連線。

執行以下命令：

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role",
        "status": "PENDING"
    }
```

```
}
```

接受或拒絕管理帳戶中的環境帳戶連接，如下面的命令和響應所示。

> **Note**
>
> 如果您拒絕此環境帳戶連線，您將無法接受或使用已拒絕的環境帳戶連線。

如果您指定拒絕，狀態變更為待決至拒絕。

如果您指定接受，狀態變更為待決至連線。

執行下列命令以接受環境帳戶連線：

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

執行下列命令以拒絕環境帳戶連線：

```
$ aws proton reject-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-reject",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role"
    }
}
```

檢視環境帳戶的連線。你可以得到或者名單環境帳戶連線。

執行下列 get 命令：

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

刪除環境帳戶中的環境帳戶連線。

> **Note**
>
> 如果您刪除此環境帳戶連線，AWS Proton在環境帳戶和具名環境接受新的環境連接之前，將無法管理環境帳戶中的環境基礎結構資源。您負責清理沒有環境連線的已佈建資源。

執行以下命令：

```
$ aws proton delete-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

回應：

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

客戶管理的環境

透過客戶管理的環境，您可以使用已部署為您的現有基礎架構，例如 VPCAWS Proton環境。使用客戶管理的環境時，您可以在以外佈建自己的共用資源AWS Proton。但是，您仍然可以允許AWS Proton使用相關的佈建輸出作為輸入AWS Proton部署時的服務。如果輸出可以改變，AWS Proton能夠接受更新。AWS Proton但是，由於佈建是在以外管理的，因此無法直接更改環境AWS Proton。

建立環境之後，您必須負責提供相同的輸出給AWS Proton如果這將被創建AWS Proton已經取得了環境，如亞馬遜 ECS 集群名稱或亞馬遜 VPC ID。

使用此功能，您可以部署和更新AWS Proton服務資源AWS Proton此環境的服務模板。但是，環境本身不會通過模板更新進行修改AWS Proton。您負責對環境執行更新並更新這些輸出AWS Proton。

您可以在一個帳戶中擁有多個環境，這些環境可以混合AWS Proton受管理和客戶管理的環境。您也可以連結第二個帳號並使用AWS Proton主帳戶中的範本，用於在該第二個連結帳戶中對環境和服務執行部署和更新。

如何使用客戶管理的環境

系統管理員需要做的第一件事就是註冊匯入的客戶管理環境範本。請勿在範本組合包中提供資訊清單或基礎結構檔案。僅提供綱要。

下面的架構概述了使用 open API 格式的輸出列表，並從AWS CloudFormation範本。

> **Important**
>
> 輸出只允許字符串輸入。

下列範例是輸出區段的程式碼片段AWS CloudFormation相應的法門模板的模板。

```
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

對應的模式AWS Proton匯入的環境與以下內容類似。請勿在結構描述中提供預設值。

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentOutput"
  types:
    EnvironmentOutput:
```

```
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:
          type: string
          description: "The ARN of the ECS role"
        VpcId:
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

註冊範本時，表示已匯入此範本，並提供服務包的 Amazon S3 儲存貯體位置。AWS Proton驗證結構描述只包含`environment_input_type`和沒有AWS CloudFormation範本參數，然後再將範本置於草稿中。

您可以提供以下內容來建立匯入的環境。

- 進行部署時要使用的 IAM 角色。
- 包含所需輸出值的規格。

您可以透過主控台或AWS CLI使用類似於一般環境部署的程序。

CodeBuild佈建角色建立

基礎架構即程式碼 (IaaS) 工具，例如AWS CloudFormation和 Terraform 需要許多不同類型的權限AWS資源。例如，如果 IaaS 範本宣告 Amazon S3 儲存貯體，則需要建立、讀取、更新和刪除 Amazon S3 儲存貯體的許可。將角色限制為所需的最低權限被視為安全性最佳做法。鑑於廣度AWS資源，建立 IaaS 範本的最低權限原則是具有挑戰性，尤其是當這些範本所管理的資源可能稍後變更時。例如，在您對管理的範本的最新編輯中AWS Proton，您可以新增 RDS 資料庫資源。

設定正確的權限可協助您的 IaC 順利部署。AWS ProtonCodeBuild佈建會執行客戶提供的任意 CLI 命令CodeBuild位於客戶帳戶中的專案。通常，這些命令會使用基礎結構即程式碼 (IaaS) 工具來建立和刪除基礎結構，例如AWS CDK。當AWS範本使用的資源部署CodeBuild佈建，AWS將啟動一個構建CodeBuild專案管理者AWS。角色傳遞給CodeBuild，其中CodeBuild假設執行命令。這個角色，稱為CodeBuild佈建角色由客戶提供，並包含佈建基礎結構所需的權限。這意味著只能由CodeBuild甚至AWS Proton不能假設它。

建立 角色

該CodeBuild佈建角色可以在 IAM 主控台中建立，也可以在AWS CLI。若要在中建立它AWS CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AWSProtonCodeBuildProvisioningBasicAccess
```

這也附加了AWSProtonCodeBuildProvisioningBasicAccess，其中包含CodeBuild運行構建的服務。

如果您希望使用控制台，請在創建角色時確保以下事項：

1. 對於信任的實體，請選取AWS服務，然後選擇CodeBuild。
2. 在「新增權限」步驟中，選取AWSProtonCodeBuildProvisioningBasicAccess以及您要附加的任何其他政策。

管理員存取

如果您附上AdministratorAccess政策的CodeBuild佈建角色，它將保證任何 IaaS 範本不會因為缺少權限而失敗。這也表示任何可以建立環境範本或服務範本的使用者都可以執行系統管理員層級的動作，即使該使用者不是系統管理員也一樣。AWS Proton不建議使用AdministatorAccess與CodeBuild佈建角色。如果您決定使用AdministratorAccess與CodeBuild佈建角色，請在沙箱環境中執行此操作。

您可以使用以下方式建立角色AdministratorAccess在 IAM 控制台中或通過執行以下命令：

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

建立最小範圍角色

如果您想要建立具有最低權限的角色，有多種方法：

- 使用管理員權限進行部署，然後縮小角色的範圍。我們建議使用IAM 存取分析器。

- 使用受管政策授與您計劃使用之服務的存取權。

AWS CDK

如果您正在使用AWS CDK與AWS Proton，你跑了`cdk bootstrap`在每個環境帳戶/區域上，則已存在`cdk deploy`。在這種情況下，請將以下策略附加到CodeBuild佈建角色：

```
{
    "Action": "sts:AssumeRole",
    "Resource": [
        "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
        "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
    ],
    "Effect": "Allow"
}
```

自訂虛擬電腦

如果你決定跑CodeBuild在一個自訂 VPC，您將需要以下權限CodeBuild角色：

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:network-interface/*",
        "arn:aws:ec2:region:account-id:subnet/*",
        "arn:aws:ec2:region:account-id:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:*/*"
    ]
},
{
    "Effect": "Allow",
```

```
    "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:AuthorizedService": "codebuild.amazonaws.com"
        }
    }
}
```

您也可以使用AmazonEC2FullAccess受管理策略，雖然包括您可能不需要的權限。如果要使用 CLI 連結受管理的原則：

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-
policy-document '{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

AWS Proton 服務

AWS Proton服務是一個服務模板的實例化，通常包括多個服務實例和一個管道。AWS Proton服務實例是在特定環境中實例化服務模板。服務範本是服務基礎結構和選用服務管線的完整定義。AWS Proton

部署服務執行個體之後，您可以透過提示 CI/CD 管線的原始程式碼推送，或將服務更新為其服務範本的新版本來更新這些執行個體。 AWS Proton當服務範本的新版本可供使用時，會提示您將服務更新為新版本。更新服務時，請AWS Proton重新部署服務和服務執行個體。

本章介紹如何使用創建，查看，更新和刪除操作來管理服務。如需其他資訊，請參閱AWS Proton服務 API 參考。

主題

建立服務

若要以開發人員身分部署應用程式，您可以建立服務並提供下列輸入。AWS Proton

1. 由平台小組發佈的AWS Proton服務範本名稱。
2. 服務的名稱。
3. 您要部署的服務執行個體數。
4. 您想要使用的環境。
5. 如果您使用的服務模板包含服務管道（可選），則與代碼存儲庫的連接。

什麼是在一個服務？

在建立AWS Proton服務時，您可以從兩種不同類型的服務範本中進行選擇：

- 包含服務管線 (預設值) 的服務範本。
- 不包含服務管線的服務範本。

在建立服務時，您必須至少建立一個服務執行個體。

服務執行個體和選用管線與服務相關聯。您只能在服務建立和刪除動作的前後關聯內建立或刪除管線。若要了解如何從服務新增和移除執行個體的詳細資訊，請參閱編輯服務。

> **Note**
>
> 您的環境已設定為AWS-或自我管理的佈建。 AWS Proton使用與環境使用相同的佈建方法，在環境中佈建服務。創建或更新服務實例的開發人員看不到差異，他們的體驗在兩種情況下都是相同的。
> 如需佈建方法的詳細資訊，請參閱the section called “佈建方法”。

服務範本

主要和次要版本的服務模板都可用。使用主控台時，請選取服務範本的最新Recommended主要和次要版本。當您使用AWS CLI且只指定服務範本的主要版本時，您會隱含地指定其最新的Recommended次要版本。

以下說明主要和次要範本版本及其使用方式之間的差異。

- 模板的新版本在獲得Recommended平台團隊成員的批准後立即變為。這表示新服務是使用該版本建立的，而且系統會提示您將現有服務更新為新版本。
- 通過AWS Proton，平台團隊可以自動將服務實例更新為服務模板的新次要版本。次要版本必須向後相容。
- 由於主要版本要求您提供新的輸入作為更新過程的一部分，您需要將服務更新為其服務模板的主要版本。主要版本不向後兼容。

建立服務

下列程序顯示如何使用主AWS Proton控台或AWS CLI建立具有或不含服務管線的服務。

AWS Management Console

建立如下列主控台步驟所示的服務。

1. 在AWS Proton主控台中，選擇 [服務]。
2. 選擇 Create service (建立服務)。
3. 在 [選擇服務範本] 頁面中，選取範本並選擇 [設定]。

   如果您不想使用已啟用的管道，請為您的服務選擇標記為「排除管道」的範本。
4. 在 [設定服務] 頁面的 [服務設定] 區段中，輸入服務名稱。
5. (選用) 輸入服務的描述。
6. 在「服務儲存庫設定」區段中：

   a. 對於「CodeStar 連線」，請從清單中選擇您的連線。

   b. 對於「儲存庫 ID」，請從清單中選擇原始程式碼儲存庫的名稱。

   c. 對於「分支名稱」，請從清單中選擇原始程式碼儲存庫分支的名稱。
7. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和值以建立客戶管理的標籤。
8. 選擇 Next (下一步)。
9. 在 [設定自訂設定] 頁面的 [服務執行個體] 區段的 [新增執行個體] 區段中。您必須輸入`required`參數的值。您可以輸入`optional`參數值，或在指定時使用預設值。
10. 在「管線輸入」區段中，您必須輸入`required`參數值。您可以輸入`optional`參數值，或在指定時使用預設值。
11. 選擇 [下一步] 並檢閱您的輸入。
12. 選擇 Create (建立)。

    檢視服務詳細資訊和狀態，以及您服務的AWS受管理標籤和客戶管理標籤。
13. 在導覽窗格中，選擇 Services (服務)。

    新頁面會顯示您的服務清單，以及狀態和其他服務詳細資料。

AWS CLI

使用時AWS CLI，您可以在位於原始程式碼目錄中的 YAML 格式`spec`檔案中指定服務輸入。`.aws-proton/service.yaml`

您可以使用 CLIget-service-template-minor-version 命令來檢視您在 spec 檔案中提供值的結構描述必要參數和選用參數。

如果您想要使用具有的服務範本pipelineProvisioning: "CUSTOMER_MANAGED"，請勿在規格中包含該pipeline:區段，也不要在create-service命令中包含-repository-connection-arn-repository-id、和-branch-name參數。

使用服務管線建立服務，如下列 CLI 步驟所示。

1. 設定管線的服務角色，如下列 CLI 範例命令所示。

   命令：

   ```
   $ aws proton update-account-settings \
           --pipeline-service-role-arn
    "arn:aws:iam::123456789012:role/AWSProtonServiceRole"
   ```

2. 下列清單顯示以服務範本結構描述為基礎的範例規格，其中包括服務管線和執行個體輸入。

   Spec (格)

   ```
   proton: ServiceSpec

   pipeline:
     my_sample_pipeline_required_input: "hello"
     my_sample_pipeline_optional_input: "bye"

   instances:
     - name: "acme-network-dev"
       environment: "ENV_NAME"
       spec:
         my_sample_service_instance_required_input: "hi"
         my_sample_service_instance_optional_input: "ho"
   ```

   使用管線建立服務，如下列 CLI 範例命令和回應所示。

   命令：

   ```
   $ aws proton create-service \
           --name "MySimpleService" \
           --branch-name "mainline" \
           --template-major-version "1" \
   ```

```
        --template-name "fargate-service" \
        --repository-connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
        --repository-id "myorg/myapp" \
        --spec "file://spec.yaml"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

建立沒有服務管線的服務，如下列 CLI 範例命令和回應所示。

以下顯示不包含服務管線輸入的範例規格。

Spec (格)

```
proton: ServiceSpec

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

若要在沒有佈建服務管線的情況下建立服務，您需要提供的路徑，而**spec.yaml**且不包含存放庫參數，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton create-service \
        --name "MySimpleServiceNoPipeline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --spec "file://spec-no-pipeline.yaml"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/
MySimpleServiceNoPipeline",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleServiceNoPipeline",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service-no-pipeline"
    }
}
```

檢視服務資料

您可以使用AWS Proton主控台或檢視和列出服務詳細資訊資料AWS CLI。

AWS Management Console

使用AWS Proton主控台列出並檢視服務詳細資訊，如下列步驟所示。

1. 若要檢視您的服務清單，請在功能窗格中選擇 [服務]。
2. 若要檢視詳細資料，請選擇服務名稱。

   檢視服務詳細資料。

AWS CLI

檢視具有服務管線之服務的詳細資訊，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton get-service \
```

```
    --name "simple-svc"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "repositoryId": "myorg/myapp",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

檢視沒有服務管線的服務詳細資訊，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton get-service \
    --name "simple-svc-no-pipeline"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc-without-
pipeline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc-without-pipeline",
        "spec": "proton: ServiceSpec\ninstances:\n- name: instance-svc-simple\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_required_input:
 hi\n    my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple-no-pipeline"
    }
}
```

編輯服務

您可以對AWS Proton服務進行以下編輯。

- 編輯服務描述。
- 透過新增和移除服務執行個體來編輯服務。

編輯服務說明

您可以使用主控台或AWS CLI來編輯服務描述。

AWS Management Console

使用主控台編輯服務，如下列步驟所述。

在服務列表中。

1. 在AWS Proton主控台中，選擇 [服務]。
2. 在服務清單中，請選擇您要更新的服務左側的選項按鈕。

3. 選擇 Edit (編輯)。
4. 在 [設定服務] 頁面中，填寫表單，然後選擇 [下一步]。
5. 在 [設定自訂設定] 頁面中，選擇 [下一步]。
6. 檢閱您的編輯並選擇 [儲存變更]。

在服務詳細資訊頁面中。

1. 在AWS Proton主控台中，選擇 [服務]。
2. 在服務清單中，請選擇您要編輯的服務名稱。
3. 在服務詳細資訊頁面中，選擇編輯。
4. 在 [設定服務] 頁面中，填寫表單，然後選擇 [下一步]。
5. 在 [設定自訂設定] 頁面中，填寫表單，然後選擇 [下一步]。
6. 檢閱您的編輯並選擇 [儲存變更]。

AWS CLI

編輯描述，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton update-service \
    --name "MySimpleService" \
    --description "Edit by updating description"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2021-03-12T22:44:21.975000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "ACTIVE",
```

```
        "templateName": "fargate-service"
    }
}
```

編輯服務以新增或移除服務執行個體

對於AWS Proton服務，您可以提交已編輯的規格，以新增或刪除服務執行個體。若要成功請求必須滿足下列條件：

- 當您提交編輯請求時，您的服務和管道尚未被編輯或刪除。
- 您編輯的規格不包括修改服務管道或對不要刪除的現有服務實例進行編輯的編輯。
- 您編輯的規格不會移除任何具有附加元件的現有服務執行個體。若要刪除這類服務執行個體，您應該先更新元件，以將其與其服務執行個體中斷連結。如需元件的詳細資訊，請參閱元件。

刪除失敗的執行個體是DELETE_FAILED狀態中的服務執行個體。當您要求編輯服務時，會AWS Proton嘗試移除刪除失敗的執行個體，做為編輯程序的一部分。如果您的任何服務執行個體無法刪除，則可能仍有與執行個體相關聯的資源，即使在主控台或中看不到這些資源AWS CLI。檢查刪除失敗的執行個體基礎結構資源並清理它們，AWS Proton以便為您移除它們。

如需服務的服務執行個體配額，請參閱AWS Proton 配額。建立服務之後，您也必須維護至少 1 個服務執行個體。在更新程序期間，AWS Proton會計算現有服務執行個體以及要新增或移除的執行個體。刪除失敗的執行個體包含在此計數中，您必須在編輯時考慮這些執行個體spec。

使用主控台或新AWS CLI增或移除服務執行個體

AWS Management Console

使用主控台編輯服務以新增或移除服務執行個體。

在AWS Proton主控台中

1. 在導覽窗格中，選擇 Services (服務)。
2. 選取您要編輯的服務。
3. 選擇 Edit (編輯)。
4. (選擇性) 在 [設定服務] 頁面上，編輯服務名稱或說明，然後選擇 [下一步]。
5. 在 [設定自訂設定] 頁面上，選擇 [刪除] 以刪除服務執行個體，然後選擇 [新增執行個體] 以新增服務執行個體並填寫表單。

6. 選擇 Next (下一步)。
7. 檢閱更新，然後選擇 [儲存變更]。
8. 強制回應會要求您驗證服務執行個體的刪除。按照說明進行操作，然後選擇是，刪除。
9. 在服務詳細資訊頁面上，檢視服務的狀態詳細資料。

AWS CLI

新增和刪除服務執行個體**spec**，AWS CLI如下列範例指令和回應所示。

使用 CLI 時，spec必須排除要刪除的服務執行個體，並包含要新增的服務執行個體和尚未標示為要刪除的現有服務執行個體。

下列清單顯示編輯spec前的範例，以及規格所部署的服務執行個體清單。此規格在前面的範例中用於編輯服務描述。

Spec (格)

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "def"
      my_sample_service_instance_required_input: "456"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

下列範例 CLIlist-service-instances 命令和回應會顯示新增或刪除服務執行個體之前的作用中執行個體。

命令：

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

回應：

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "example-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.160000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.160000+00:00",
            "name": "my-instance",
            "serviceName": "example-svc",
            "serviceTemplateArn": "arn:aws:proton:region-id:123456789012:service-
template/fargate-service",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

下列清單顯示spec用於刪除和新增執行個體的已編輯範例。已移除名為my-instance的現有執行個體，並新增名為的新執行yet-another-instance個體。

Spec (格)

```
proton: ServiceSpec
```

```
pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

如果值存在於中spec，您可以使用"${Proton::CURRENT_VAL}"指示要保留原始參數值的參數值spec。用get-service於檢視服務spec的原始檔案，如中所述檢視服務資料。

下列清單顯示如何用"${Proton::CURRENT_VAL}"來確保您spec不包含要保留現有服務執行個體的參數值變更。

Spec (格)

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

下一個清單顯示 CLI 命令和編輯服務的回應。

命令：

```
$ aws proton update-service
    --name "MySimpleService" \
    --description "Edit by adding and deleting a service instance" \
    --spec "file://spec.yaml"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by adding and deleting a service instance",
        "lastModifiedAt": "2021-03-12T22:55:48.169000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "UPDATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

下列list-service-instances指令和回應會確認已移除名為my-instance的現有執行個體，並新增名為的新執行yet-another-instance個體。

命令：

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

回應：

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/yet-another-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
```

```
            "lastDeploymentAttemptedAt": "2021-03-12T22:56:01.565000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:56:01.565000+00:00",
            "name": "yet-another-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```


新增或移除服務執行個體時會如何

在您提交服務編輯以刪除及新增服務執行個體之後，請執AWS Proton行下列動作。

- 將服務設定為UPDATE_IN_PROGRESS。
- 如果服務具有管線，請將其狀態設定為IN_PROGRESS並封鎖管線動作。
- 設定要刪除的任何服務執行個體DELETE_IN_PROGRESS。
- 封鎖服務動作。
- 封鎖標示為要刪除的服務執行個體上的動作。
- 建立新的服務執行個體。
- 刪除您列出要刪除的例證。
- 嘗試移除刪除失敗的執行個體。
- 新增和刪除完成後，重新佈建服務管線 (如果有的話)、將您的服務設定為，ACTIVE並啟用服務和管線動作。

AWS Proton嘗試重新調解失敗模式，如下所示。

- 如果一或多個服務執行個體無法建立，請AWS Proton嘗試取消佈建所有新建立的服務執行個體，並將其還原spec為先前的狀態。它不會刪除任何服務實例，也不會以任何方式修改管道。
- 如果一或多個服務執行個體無法刪除，請AWS Proton重新佈建管線而不刪除的執行個體。會spec更新以包含新增的執行個體，並排除標記為要刪除的執行個體。
- 如果管線佈建失敗，則不會嘗試復原，且服務和管線都會反映失敗的更新狀態。

標記和服務編輯

當您將服務執行個體新增為服務編輯的一部分時，AWS受管理的標籤會傳播到新執行個體和佈建的資源，並自動建立。如果您建立新標籤，這些標籤只會套用至新的例項。現有服務客戶受管標籤也會傳播至新的執行個體。如需詳細資訊，請參閱 AWS Proton資源和標記。

刪除服務

您可以使用AWS Proton主控台或刪除AWS Proton服務及其執行個體和管線AWS CLI。

您無法刪除具有任何具有連接元件的服務執行個體的服務。若要刪除此類服務，您應該先更新所有連結的元件，以將它們與其服務執行個體中斷連結。如需元件的詳細資訊，請參閱元件。

AWS Management Console

使用主控台刪除服務，如下列步驟所述。

在服務詳細資訊頁面中。

1. 在AWS Proton主控台中，選擇 [服務]。
2. 在服務清單中，請選擇您要刪除的服務名稱。
3. 在服務詳細資訊頁面上，選擇動作，然後選擇刪除。
4. 強制回應會提示您確認刪除動作。
5. 按照說明進行操作，然後選擇是，刪除。

AWS CLI

刪除服務，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton delete-service \
    --name "simple-svc"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2020-11-29T00:30:39.248000+00:00",
        "name": "simple-svc",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "DELETE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

檢視服務實例資料

瞭解如何檢視AWS Proton服務執行個體詳細資料。您可以使用主控台或AWS CLI。

服務執行個體屬於服務。您只能在服務編輯、建立和刪除動作的前後關聯內建立或刪除執行個體。若要了解如何從服務中新增和移除除的執行個體，請參閱編輯服務。

AWS Management Console

使用AWS Proton主控台列出並檢視服務執行個體詳細資料，如下列步驟所示。

1. 若要檢視服務執行個體清單，請在導覽窗格中選擇 [服務執行個體]。
2. 若要檢視詳細資料，請選擇服務執行處理的名稱。

   檢視您的服務執行個體詳細資料。

AWS CLI

列出並檢視服務執行個體詳細資訊，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton list-service-instances
```

回應：

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
service-instance/instance-one",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentArn": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "name": "instance-one",
            "serviceName": "simple-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

命令：

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

回應：

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
```

```
        "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: hello world\n
 my_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_optional_input:
 Ola\n    my_sample_service_instance_required_input: Ciao\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

更新服務執行個體

瞭解如何更新AWS Proton服務執行個體並取消更新。

服務執行個體屬於服務。您只能在服務編輯、建立和刪除動作的前後關聯內建立或刪除執行個體。若要了解如何從服務中新增和移除除的執行個體，請參閱編輯服務。

更新服務執行個體有四種模式，如下列清單所述。使用時AWS CLI，deployment-type欄位會定義模式。使用主控台時，這些模式會對應至 「編輯」和「更新為最新次要版本」，以及「更新為最新的主要版本」動作 (從服務執行處理詳細資訊頁面中的「動作」 下拉式清單)。

NONE

在此模式中，不會發生部署。只會更新要求的中繼資料參數。

CURRENT_VERSION

在此模式中，服務執行個體會部署並以您提供的新規格進行更新。只更新了請求參數。使用此參數時，請勿包含次要或主要版本參數deployment-type。

MINOR_VERSION

在此模式中，依預設，服務執行個體會以目前使用中主要版本的已發佈建議 (最新) 次要版本進行部署和更新。您也可以指定目前正在使用的主要版本的不同次要版本。

MAJOR_VERSION

在此模式中，服務執行個體預設會以目前範本的已發佈、建議 (最新) 主要和次要版本進行部署和更新。您也可以指定高於使用中的主要版本和次要版本 (選用) 的不同主要版本。

如果deploymentStatus是，您可以嘗試取消服務執行個體更新部署IN_PROGRESS。 AWS Proton嘗試取消部署。不能保證取消成功。

取消更新部署時，會AWS Proton嘗試取消部署，如下列步驟所列。

- 將部署狀態設定為CANCELLING。
- 停止處理中的部署，並刪除部署在時間建立的所有新資源IN_PROGRESS。
- 將部署狀態設定為CANCELLED。
- 將資源的狀態還原為開始部署之前的狀態。

如需取消服務執行個體部署的詳細資訊，請參閱 AWS ProtonAPI 參考CancelServiceInstanceDeployment中的。

使用主控台或AWS CLI進行更新或取消更新部署。

AWS Management Console

請遵循下列步驟，使用主控台更新服務執行個體。

1. 在AWS Proton主控台中，選擇導覽窗格中的 [服務執行個體]。
2. 在服務例證清單中，請選擇您要更新的服務執行個體名稱。
3. 選擇「動作」，然後選擇其中一個更新選項，選擇 「編輯」以更新規格或動作，然後選擇「更新為最新次要版本」，或「更新為最新的主要版本」。
4. 填寫每個表格，然後選擇「下一步」，直到到達「評論」頁面。
5. 檢閱您的編輯並選擇 [更新]。

AWS CLI

將服務實例更新為新的次要版本，如 CLI 示例命令和響應中所示。

當您使用已修改的服務執行個體來更新服務執行個體時spec，如果值存在於中spec，您可以使用"${Proton::CURRENT_VAL}"來指出要保留的參數值與原始參數值相比spec。用get-service於檢視服務執行spec個體的原始資料，如中所述檢視服務資料。

以下範例示範如何"${Proton::CURRENT_VAL}"在spec.

Spec (格)

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

命令：更新

```
$ aws proton update-service-instance \
    --name "instance-one" \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

回應：

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentName": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "name": "instance-one",
```

```
        "serviceName": "simple-svc",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

命令：取得並確認狀態

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

回應：

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

AWS Management Console

使用主控台取消服務執行個體部署，如下列步驟所示。

1. 在AWS Proton主控台中，選擇導覽窗格中的 [服務執行個體]。
2. 在服務例證清單中，選擇您要取消的部署更新的服務執行個體名稱。
3. 如果您的更新部署狀態為進行中，請在服務執行個體詳細資訊頁面中選擇動作，然後選擇取消部署。
4. 模式要求您確認取消。選擇 [取消部署]。
5. 您的更新部署狀態設定為「取消」，然後設定為「已取消」以完成取消。

AWS CLI

取消 IN_PROGRESS 服務執行個體部署更新至新的次要版本 2，如下列 CLI 範例命令和回應所示。

等待條件包含在用於此範例的範本中，以便在更新部署成功之前開始取消。

命令：取消

```
$ aws proton cancel-service-instance-deployment \
    --service-instance-name "instance-one" \
    --service-name "simple-svc"
```

回應：

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: abc\n  my_sample_pipeline_required_input:
 '123'\ninstances:\n- name: my-instance\n  environment: MySimpleEnv
\n  spec:\n    my_sample_service_instance_optional_input: def\n
 my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
 environment: MySimpleEnv\n  spec:\n    my_sample_service_instance_required_input:
 '789'\n",
        "templateMajorVersion": "1",
```

```
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

命令：取得並確認狀態

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

回應：

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

更新服務管道

了解如何更新AWS Proton服務管道並取消更新。

服務管線屬於服務。您只能在服務建立和刪除動作的前後關聯內建立或刪除管線。

更新服務管線有四種模式，如下列清單所述。使用時AWS CLI，deployment-type欄位會定義模式。當您使用主控台時，這些模式會對應至 「編輯管線」 和「更新至建議的版本」。

NONE

在此模式中，不會發生部署。只會更新要求的中繼資料參數。

CURRENT_VERSION

在此模式中，服務管線會部署並使用您提供的新規格進行更新。只更新了請求參數。使用此參數時，請勿包含次要或主要版本參數deployment-type。

MINOR_VERSION

在此模式下，依預設，服務管線會以目前使用的主要版本的已發佈建議 (最新) 次要版本進行部署和更新。您也可以指定目前正在使用的主要版本的不同次要版本。

MAJOR_VERSION

在此模式下，依預設，服務管線會以目前範本的已發佈、建議 (最新) 主要和次要版本進行部署和更新。您也可以指定高於使用中的主要版本和次要版本 (選用) 的不同主要版本。

如果deploymentStatus是，您可以嘗試取消服務管線更新部署IN_PROGRESS。 AWS Proton嘗試取消部署。不能保證取消成功。

取消更新部署時，會AWS Proton嘗試取消部署，如下列步驟所列。

- 將部署狀態設定為CANCELLING。
- 停止處理中的部署，並刪除部署在時間建立的所有新資源IN_PROGRESS。
- 將部署狀態設定為CANCELLED。
- 將資源的狀態還原為開始部署之前的狀態。

如需取消服務管線部署的詳細資訊，請參閱 AWS ProtonAPI 參考CancelServicePipelineDeployment中的。

使用主控台或AWS CLI進行更新或取消更新部署。

AWS Management Console

使用主控台更新服務管線，如下列步驟所述。

1. 在AWS Proton主控台中，選擇 [服務]。
2. 在服務清單中，請選擇您要更新的服務名稱。
3. 服務詳細資訊頁面上有兩個標籤：「概觀」和「管線」。選擇「管線」。
4. 如果您想要更新規格，請選擇 [編輯管線] 並填寫每個表單，然後選擇 [下一步]，直到您完成最終表單為止，然後選擇 [更新管道]。

   如果您想要更新為新版本，而且在 Pipeline 範本中有一個資訊圖示指出新版本可用，請選擇新範本版本的名稱。

   a. 選擇「更新至建議版本」。
   b. 填寫每份表格，然後選擇「下一步」，直到您完成最終表格並選擇「更新」。

AWS CLI

將服務管線更新為新的次要版本，如下列 CLI 範例命令和回應所示。

當您使用已修改的服務管線來更新服務管線時`spec`，如果值存在於中`spec`，您可以使用`"${Proton::CURRENT_VAL}"`指示要保留原始參數值的參數值`spec`。用`get-service`於檢視服務管線`spec`的原始資料，如中所述檢視服務資料。

以下範例示範如何`"${Proton::CURRENT_VAL}"`在`spec`.

Spec (格)

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
```

```
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

命令：更新

```
$ aws proton update-service-pipeline \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

回應：

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"my-instance\"\n    environment: \"MySimpleEnv
\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"MySimpleEnv\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

命令：取得並確認狀態

```
$ aws proton get-service \
```

```
    --name "simple-svc"
```

回應：

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

AWS Management Console

使用主控台取消服務管線部署，如下列步驟所示。

1. 在AWS Proton主控台中，選擇功能窗格中的 [服務]。
2. 在服務清單中，選擇您要取消的部署更新的管道名稱。
3. 在服務詳細資訊頁面上，選擇 Pipeline (管道) 標籤
4. 如果您的更新部署狀態為 [進行中]，請在服務管線詳細資料頁面中選擇 [取消部署]。
5. 模式要求您確認取消。選擇 [取消部署]。
6. 您的更新部署狀態設定為「取消」，然後設定為「已取消」以完成取消。

AWS CLI

將 IN_PROGRESS 服務管線部署更新取消至次要版本 2，如下列 CLI 範例命令和回應所示。

等待條件包含在用於此範例的範本中，以便在更新部署成功之前開始取消。

命令：取消

```
$ aws proton cancel-service-pipeline-deployment \
    --service-name "simple-svc"
```

回應：

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

命令：取得並確認狀態

```
$ aws proton get-service \
    --name "simple-svc"
```

回應：


```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "CANCELLED",
            "deploymentStatusMessage": "User initiated cancellation.",
            "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
```

```
    }
}
```

AWS Proton 元件

組件是一種AWS Proton資源。他們增加了服務模板的靈活性。元件為平台團隊提供擴充核心基礎架構模式的機制，並定義保護措施，讓開發人員能夠管理其應用程式基礎架構的各個層面。

在AWS Proton管理員中，定義跨開發團隊和應用程式使用的標準基礎結構。但是，開發團隊可能需要為其特定使用案例包含其他資源，例如 Amazon Simple Queue Service (Amazon SQS) 佇列或 Amazon DynamoDB 表。這些應用程式特定的資源可能會經常變更，特別是在早期應用程式 在管理員編寫的範本中維護這些頻繁變更可能很難管理和擴展 — 管理員需要維護更多的範本，而不需要真正的系統管理員附加價值。另一種方法是讓應用程式開發人員為其應用程式撰寫範本，也不理想，因為它會消除系統管理員標準化主要架構元件 (例如AWS Fargate工作) 的能力。這是組件進來的地方。

使用元件，開發人員可以將補充資源新增至其應用程式，超出管理員在環境和服務範本中定義的範圍之外。然後，開發人員將組件附加到服務實例。 AWS Proton佈建元件所定義的基礎結構資源，就像它為環境和服務執行個體佈建資源一樣。

組件可以讀取服務實例輸入並提供輸出到服務實例，以獲得完全集成的體驗。例如，如果元件新增 Amazon Simple Storage Service (Amazon S3) 貯體供服務執行個體使用，則元件範本可以將環境和服務執行個體名稱納入考量以命名儲存貯體。在AWS Proton呈現服務範本以佈建服務執行個體時，服務執行個體可以參考值區並使用它。

AWS Proton目前支援的元件是直接定義的元件。您可以將定義元件基礎結構的基礎結構直接傳遞至 AWS Proton API 或主控台的基礎結構即程式碼 (IaC) 檔案。這與環境或服務不同，您可以在模板包中定義 IaC 並將包註冊為模板資源，然後使用模板資源來創建環境或服務。

> **Note**
>
> 直接定義的元件可讓開發人員定義額外的基礎架構並進行佈建 AWS Proton使用相同AWS Identity and Access Management (IAM) 角色佈建在相同環境中執行的所有直接定義元件。

管理員可以通過兩種方式控制開發人員可以使用組件執行的操作：

- 支援的元件來源 — 管理員可以根據服務範本版本的屬性，允許將元件附加至AWS Proton服務執行個體。根據預設，開發人員無法將元件附加到服務執行個體。

  如需有關此屬性的詳細資訊，請supportedComponentSources參閱 CreateServiceTemplateVersionAPI 參考中 API 動作的AWS Proton參數。

> **Note**
>
> 當您使用範本同步時，AWS Proton會在您將變更提交至存放庫中的服務範本組合時，隱含地建立服務範本版本。在這種情況下，您不需要在建立服務範本版本期間指定支援的元件來源，而是在與每個服務範本主要版本相關聯的檔案中指定此屬性。如需詳細資訊，請參閱 the section called “同步服務範本”。

- 元件角色 — 管理員可以將元件角色指派給環境。 AWS Proton當它佈建由環境中直接定義的元件定義的基礎結構時，會承擔此角色。因此，元件角色會限制開發人員可以使用環境中直接定義的元件新增的基礎結構。在沒有組件角色的情況下，開發人員無法在環境中創建直接定義的組件。

  有關指派元件角色的詳細資訊，請componentRoleArn參閱 CreateEnvironmentAPI 參考中 API 動作的AWS Proton參數。

> **Note**
>
> 組件角色不在自我管理的佈建環境中使用。

主題

組件與其他AWS Proton資源相比如何？

在許多方面，組件類似於其他AWS Proton資源。它們的基礎結構是在 IaC 範本檔案中定義，以AWS CloudFormation YAML 或地形 HCL 格式撰寫。 AWS Proton可以使用AWS管理佈建或自我管理佈建來佈建元件基礎結構。

但是，組件在以下幾個方面與其他AWS Proton資源不同：

- 已分離狀態 — 元件的設計目的是連接到服務執行個體並擴充其基礎結構，但也可以處於分離狀態，在此狀態中它們不會連接到任何服務執行個體。如需元件狀態的詳細資訊，請參閱the section called “元件狀態”。
- 沒有模式-組件沒有像模板包那樣的關聯模式。組件輸入由服務定義。當元件連接至服務執行個體時，元件可能會消耗輸入。
- 沒有客戶管理的元件 —AWS Proton 永遠為您佈建元件基礎架構。沒有自帶的資源版本的組件。如需客戶管理環境的詳細資訊，請參閱the section called “建立”。
- 無範本資源 — 直接定義的元件沒有類似於環境和服務範本的關聯範本資源。您可以直接提供給元件的 IaC 範本檔案。同樣地，您可以直接提供定義範本語言和轉譯引擎的資訊清單，以佈建元件的基礎結構。您可以使用類似於撰寫範本服務包的方式撰寫範本檔案和資訊清單。但是，使用直接定義的組件，沒有要求將 IaC 文件存儲為捆綁在特定位置，並且不會在 IaC 文件中AWS Proton創建模板資源。
- 無 CodeBuild基於佈建 — 您無法使用自己的自訂佈建指令碼 (稱為CodeBuild基礎佈建) 佈建直接定義的元件。如需詳細資訊，請參閱 the section called “CodeBuild佈建”。

AWS Proton控制台中的組件

使用主AWS Proton控台建立、更新、檢視和使用AWS Proton元件。

下列主控台頁面與元件有關。我們包括直接鏈接到頂級控制台頁面。

- 元件 — 檢視您AWS帳戶中的元件清單。您可以建立新元件，以及更新或刪除現有元件。在清單上傳元件名稱，以檢視其詳細資訊頁面。

  環境詳細資料和服務執行個體詳細資料頁面上也有類似的清單。這些清單只會顯示與正在檢視之資源相關聯的元件。當您從這些清單之一建立元件時，請在「建立元件」頁面AWS Proton預先選取相關的環境。
- 元件詳細資訊 — 若要檢視元件詳細資訊頁面，請在「元件」清單中選擇元件名稱。

  在詳細資訊頁面上，檢視元件詳細資訊和狀態，以及更新或刪除元件。檢視和管理輸出清單 (例如佈建的資源 ARN)、佈建的AWS CloudFormation堆疊和指派的標籤。
- 建立元件 — 建立元件。輸入元件名稱和說明、選擇關聯的資源、指定元件來源 IaC 檔案，然後指定標籤。
- 更新元件 — 若要更新元件，請在 [元件] 清單中選取元件，然後在 [動作] 功能表上選擇 [更新元件]。或者，在 [元件詳細資訊] 頁面上，選擇 [更新]。

您可以更新大部分元件的詳細資料。因此您無法更新元件名稱。您也可以選擇是否要在成功更新之後重新部署元件。

- 配置環境 — 建立或更新環境時，您可以指定元件角色。此角色控制在環境中執行直接定義元件的能力，並提供佈建元件的權限。
- 建立新的服務範本版本 — 當您建立服務範本版本時，您可以指定範本版本的支援元件來源。這可控制根據此範本版本將元件附加至服務的服務執行個體的能力。

AWS ProtonAPI 中的元件和AWS CLI

使用AWS Proton API 或建立、更新、檢視和使用AWS Proton元件。AWS CLI

下列 API 動作可直接管理AWS Proton元件資源。

- CreateComponent— 建立AWS Proton元件。
- DeleteComponent— 刪除元AWS Proton件。
- GetComponent— 取得元件的詳細資料。
- ListComponentOutputs— 獲取組件基礎架構作為代碼列表 (IaC) 輸出.
- ListComponentProvisionedResources— 列出具有詳細資訊的元件佈建資源。
- ListComponents-列出具有摘要數據的組件。您可以依環境、服務或單一服務執行個體來篩選結果清單。

其他AWS Proton資源的下列 API 動作具有與元件相關的某些功能。

- CreateEnvironment、UpdateEnvironment— 用於指`componentRoleArn`定 IAM 服務角色的 Amazon 資源名稱 (ARN)，該角色在此環境中佈建直接定義的元件時AWS Proton使用。它決定了一個直接定義的組件可以佈建的基礎結構的範圍。
- CreateServiceTemplateVersion— 用`supportedComponentSources`於指定支援的元件來源。具有支援來源的元件可以根據此服務範本版本附加至服務執行個體。

元件

什麼是組件的生命週期？

元件可以處於貼附或分離狀態。它們被設計為連接到服務實例，並在大多數情況下增強其基礎架構。分離的組件處於過渡狀態，可讓您以受控且安全的方式刪除組件或將其附加到另一個服務實例。如需詳細資訊，請參閱 the section called “元件狀態”。

為什麼我無法刪除貼附的元件？

解決方案：若要刪除附加元件，請更新元件以將其與服務執行個體中斷連結，驗證服務執行個體的穩定性，然後刪除該元件。

為什麼這是必需的？ 附加元件會提供應用程式執行其執行階段功能所需的額外基礎結構。服務執行個體可能正在使用元件輸出來偵測並使用此基礎結構的資源。刪除元件，藉此移除其基礎結構資源，可能會對連接的服務執行個體造成干擾。

這是一項額外的安全措施，您AWS Proton必須先更新元件並將其從其服務執行個體中分離，然後才能刪除它。然後，您可以驗證您的服務執行個體，以確保其繼續部署和正常運作。如果您偵測到問題，您可以快速將元件重新連接至服務執行個體，然後解決問題。當您確信您的服務實例對組件沒有任何依賴性時，可以安全地刪除該組件。

為什麼我不能直接更改組件的附加服務實例？

解決方案：若要變更附件，請更新元件以將其與服務執行個體中斷連結，驗證元件和服務執行個體的穩定性，然後將元件附加至新的服務執行個體。

為什麼這是必需的？ 組件被設計為連接到服務實例。您的元件可能會使用服務執行個體輸入來進行基礎結構資源命名和 變更附加的服務執行個體可能會對元件造成干擾 (除了服務執行個體可能中斷之外，如前面的常見問題集所述，為什麼我無法刪除附加的元件？). 例如，它可能會導致重新命名，甚至可能取代元件的 IaC 範本中定義的資源。

作為一項額外的安全措施，您AWS Proton需要先更新元件並將其從其服務執行個體中分離，然後才能將其連接到另一個服務執行個體。然後，您可以先驗證元件和服務執行個體的穩定性，然後再將元件附加到新的服務執行個體。

元件狀態

AWS Proton組件可以處於兩種根本不同的狀態：

- 附加 — 元件會附加至服務執行個體。它定義了支持服務實例的運行時功能的基礎結構。該組件使用開發人員定義的基礎結構擴展了環境和服務模板中定義的基礎結構。

  典型元件在其生命週期中大部分有用的部分都處於貼附狀態。

- 已分離 — 元件與AWS Proton環境相關聯，且未附加至環境中的任何服務執行個體。

  這是一種過渡狀態，用於將組件的生命週期延長到單個服務實例之外。

下表提供不同元件狀態的最上層比較。

	Attached	Detached
州政府的主要目的	擴充服務執行個體的基礎結構。	為了維護服務實例附件之間的組件的基礎結構。
與相關聯	服務執行個體和環境	環境
索引鍵特定屬性	• 服務名稱 • 服務實例名稱 • Spec (規格)	• 環境名稱
可以刪除	X 否	✓ 是
可以更新到另一個服務實例	X 否	✓ 是
可以讀取輸入	✓ 是	X 否

組件的主要目的是連接到服務實例，並使用其他資源擴展其基礎結構。連接的組件可以根據規格讀取服務實例的輸入。您無法直接刪除組件或將其附加到不同的服務實例。您也無法刪除其服務執行個體或相關的服務和環境。若要執行上述任何操作，請先更新元件，以將其與其服務執行個體分離。

若要將元件的基礎結構維持在單一服務執行個體的生命週期之外，您可以移除服務和服務執行個體名稱來更新元件，並將其從其服務執行個體中分離。這種分離狀態是一個過渡狀態。元件沒有輸入。它的基礎結構保持佈建狀態，您可以更新它。您可以刪除元件在連接時所關聯的資源 (服務執行個體、服務)。您可以刪除元件或更新元件，以便再次連接至服務執行個體。

組件基礎結構作為代碼文

組件基礎結構代碼（IaC）文件類似於其他AWS Proton資源。在這裡了解一些特定於組件的詳細信息。若要取得有關為的編寫 IaC 檔案的完整資訊AWS Proton，請參閱模板創作和捆綁。

搭配元件使用參數

AWS Proton參數命名空間包括一些參數，服務 IaC 文件可以引用以獲取關聯組件的名稱和輸出。命名空間還包括組件 IaC 文件可以參考的參數，以從組件相關聯的環境，服務和服務實例中獲取輸入，輸出和資源值。

組件沒有自己的輸入-它從連接到的服務實例獲取其輸入。組件也可以讀取環境輸出。

如需有關在元件和相關服務 IaC 檔案中使用參數的更多資訊，請參閱the section called “組件 CloudFormation IAC 參數”。如需有關AWS Proton參數的一般資訊和參數命名空間的完整參考，請參閱the section called “參數”。

撰寫強大的 IaC 檔案

身為管理員，當您建立服務範本版本時，您可以決定是否要允許從範本版本建立的服務執行個體附加元件。請參閱 API supportedComponentSources參考資料中 CreateServiceTemplateVersionAPI 動作的參數。AWS Proton不過，對於任何 future 的服務執行個體，建立執行個體的人員決定是否要將元件附加至該執行個體，而且 (如果是直接定義的元件) 編寫元件 IaC 通常是不同的人員，也就是使用您的服務範本的開發人員。因此，您無法保證元件會附加至服務執行個體。您也無法保證特定元件輸出名稱是否存在，或是這些輸出值的有效性和安全性。

AWS Proton而 Jinja 語法可協助您解決這些問題，並透過下列方式撰寫強大的服務範本，這些範本不會失敗地呈現：

- AWS Proton參數篩選器 — 當您參照元件輸出屬性時，您可以使用參數篩選器 — 驗證、篩選和格式化參數值的修飾符。如需詳細資訊和範例，請參閱 the section called “CloudFormation 參數篩選”。
- 單一屬性預設值 — 當您參考組件的單一資源或輸出屬性時，您可以保證使用`default`篩選器呈現服務範本不會失敗，不論是否為預設值。如果組件或您引用的特定輸出參數不存在，則會改為呈現默認值（或空字符串，如果尚未指定默認值），並且渲染成功。如需詳細資訊，請參閱 the section called “提供預設值”。

  範例：

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

> **Note**
>
> 請勿將指定直接定義之元件的命名空間.default部分與篩選器混淆，default篩選器會在參考的屬性不存在時提供預設值。

- 整個物件參考 — 當您參考整個組件或組件輸出的集合時，會AWS Proton傳回空白物件{}，並因此保證呈現服務範本不會失敗。因此您不一定要使用任何篩選條件。請務必在可以取得空物件的前後關聯中建立參考，或使用{{ if .. }}條件來測試空白物件。

  範例：

  - `{{ service_instance.components.default }}`
  - `{{ service_instance.components.default.outputs }}`

元件AWS CloudFormation範例

以下是AWS Proton直接定義的組件以及如何在AWS Proton服務中使用它的完整示例。元件佈建 Amazon Simple Storage Service (Amazon S3) 儲存貯體。服務實例可以引用此存儲桶並使用它。值區名稱是以環境、服務、服務執行個體和元件的名稱為基礎，這表示儲存貯體與元件範本的特定執行個體結合，延伸特定服務執行個體。開發人員可以根據此元件範本建立多個元件，以針對不同的服務執行個體和功能需求佈建 Amazon S3 儲存貯體。

這個範例涵蓋撰寫各種必要的AWS CloudFormation基礎結構做為程式碼 (IaC) 檔案，以及建立必要 AWS Identity and Access Management (IAM) 角色。此範例會依擁有人員角色來分組步驟。

管理員步驟

讓開發人員能夠將元件與服務搭配使用

1. 建立AWS Identity and Access Management (IAM) 角色，將資源範圍縮減直接定義在您環境中執行的元件可佈建的資源。 AWS Proton稍後假定此角色，以便在環境中佈建直接定義的元件。

   這個範例例例如下列政策：

   Example 直接定義的元件角色

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

```
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

2. 提供您在前一個步驟中建立的角色。在AWS Proton主控台中，在 [設定環境] 頁面上指定元件角色。如果您使用的是AWS Proton APIAWS CLI，或指componentRoleArn定CreateEnvironment或 UpdateEnvironmentAPI 動作。
3. 建立服務範本，參照連接至服務執行個體的直接定義元件。

   該示例顯示瞭如何編寫強大的服務模板，如果組件未附加到服務實例，該模板不會中斷。

   Example 服務 CloudFormation IaC 文件使用一個組件

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

4. 建立將直接定義的元件宣告為受支援的新服務範本次要版本。

- Amazon S3 中的範本服務包 — 在AWS Proton主控台中，當您建立服務範本版本時，對於支援的元件來源，請選擇直接定義。如果您使用的是AWS Proton APIAWS CLI，或者DIRECTLY_DEFINED在CreateServiceTemplateVersion或UpdateServiceTemplateVersionAPI 動作的supportedComponentSources參數中指定。
- 範本同步 — 提交變更至您的服務範本組合包存放庫，您可在其supported_component_sources:中指定DIRECTLY_DEFINED為主要版本目錄中.template-registration.yaml檔案中的項目。如需有關此檔案的詳細資訊，請參閱the section called “同步服務範本”。

5. 發佈新的服務範範本範本本本本本本本本本 如需詳細資訊，請參閱 the section called “發佈”。
6. 請務必允許使用此服務範本之開發人員的 IAM 角色。proton:CreateComponent

開發人員步

使用直接定義的元件搭配服務執行處理

1. 建立使用系統管理員透過元件支援建立的服務範本版本的服務。或者，更新其中一個現有的服務執行個體以使用最新的範本版本。
2. 撰寫佈建 Amazon S3 儲存貯體和相關存取政策的元件 IaC 範本檔案，並將這些資源公開為輸出。

   Example 組件 CloudFormation IAC 文件

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
```

```
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

3. 如果您使用的是AWS Proton APIAWS CLI，或者，請為組件編寫資訊清單檔案。

   Example 直接定義的組件清單

   ```
   infrastructure:
     templates:
       - file: "cloudformation.yaml"
         rendering_engine: jinja
         template_language: cloudformation
   ```

4. 建立直接定義的元件。 AWS Proton假設管理員定義用來佈建元件的元件角色。

   在AWS Proton主控台的 [元件] 頁面上，選擇 [建立元件]。對於 「元件」設定，請輸入元件名稱和可選元件描述。在「元件附件」中，選擇「將元件附加至服務執行處理」。 選取您的環境、服務和服務執行個體。針對 [元件來源] AWS CloudFormation，選擇，然後選擇元件 IaC 檔案。

   > **Note**
   >
   > 您不需要提供資訊清單，主控台會為您建立資訊清單。

   如果您使用的是AWS Proton APIAWS CLI，或使用 CreateComponentAPI 動作。設置一個組件`name`和可選`description`。設定`environmentName`、`serviceName`、和`serviceInstanceName`。設置`templateSource`和`manifest`您創建的文件的路徑。

   > **Note**
   >
   > 當您指定服務和服務執行個體名稱時，指定環境名稱是選擇性的。這兩者的組合在您的AWS帳戶中是唯一的，並且AWS Proton可以從服務實例確定環境。

5. 更新您的服務實例以重新部署它。 AWS Proton使用轉譯服務執行個體範本中元件的輸出，讓您的應用程式能夠使用該元件佈建的 Amazon S3 儲存貯體。

使用 git 存儲庫AWS Proton

AWS Proton使用 git 存儲庫用於各種目的。下列清單分類與AWS Proton資源相關聯的存放庫類型。對於重複連接到存放庫以將內容推送到儲存庫或從中提取內容的AWS Proton功能，您必須在AWS帳戶AWS Proton中註冊存放庫連結。存放庫連結是一組AWS Proton可在連線至存放庫時使用的屬性。AWS Proton目前支援GitHub「GitHub企業BitBucket」和。

開發者儲存

程式碼儲存庫 — 開發人員用來儲存應用程式程式碼的儲存庫。用於代碼部署。 AWS Proton不會直接與此儲存庫互動。當開發人員佈建包含管線的服務時，他們會提供存放庫名稱和分支來讀取其應用程式程式碼。 AWS Proton將此資訊傳遞至其佈建的管線。

如需詳細資訊，請參閱the section called “建立”。

管理員儲存

範本存放庫 — 管理員儲存AWS Proton範本組合的存放庫。用於模板同步。當管理員在中建立範本時AWS Proton，他們可以指向範本存放庫，並使新範本與其AWS Proton保持同步。當管理員更新存放庫中的範本服務包時，AWS Proton會自動建立新的範本版本。請先將範本儲存庫連結至，AWS Proton然後才能使用它進行同步。

如需詳細資訊，請參閱the section called “範本同步設定”。

> **Note**
>
> 如果您繼續將範本上傳到 Amazon Simple Storage Service (Amazon S3)，並呼叫範本管理 API 來建立新的AWS Proton範本或範本版本，則不需要範本儲存庫。

自我管理的佈建庫

基礎結構存放庫 — 託管轉譯基礎結構範本的存放庫。用於資源基礎結構的自我管理佈建。當管理員為自我管理的佈建建立環境時，他們會提供存放庫。 AWS Proton提交提取要求 (PR) 至此儲存庫，以建立環境的基礎結構，以及任何部署至環境的服務執行個體的基礎結構。將基礎結構存放庫連結至，AWS Proton然後才能將其用於自我管理的基礎結構佈建。

管道存放庫 — 用於建立管道的存放庫。用於管線的自我管理佈建。使用額外的存放庫佈建管道可讓您AWS Proton獨立於任何個別環境或服務來儲存管線組態。您只需要為所有自我管理的佈建服務提供單一管道存放庫。將管線存放庫連結至，AWS Proton然後才能將其用於自我管理的管線佈建。

如需詳細資訊，請參閱the section called “AWS-管理佈建”。

主題

- 建立存放庫的連結
- 檢視連結儲存庫資料
- 刪除儲庫連結

建立存放庫的連結

您可以使用主控台或 CLI 建立使用主控台或 CLI 建立您的儲庫連結。當您建立存放庫連結時，AWS Proton會為您建立服務連結角色。

AWS Management Console

如下列主控台步驟所示，建立存放庫的連結。

1. 在AWS Proton主控台中，選擇 [儲存庫]。
2. 選擇 Create repository (建立儲存庫)。
3. 在「連結新儲存區域」頁面的「儲存區域詳細資訊」段落中：
   a. 選擇您的儲存庫提供者。
   b. 選擇您現有的其中一個連線。如果您沒有連線，請選擇 [新增CodeStar連線] 以建立連線，然後返回AWS Proton主控台，重新整理連線清單，然後選擇您的新連線。
   c. 從連接的源代碼存儲庫中進行選擇。
4. [選擇性] 在「標記」區段中，選擇「新增標籤」一次或多次，然後輸入「金鑰」和「值」配對。
5. 選擇 Create repository (建立儲存庫)。
6. 檢視連結儲存庫的詳細資料。

AWS CLI

建立並註冊存放庫的連結。

執行以下命令：

```
$ aws proton create-repository \
```

```
    --name myrepos/environments \
    --connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
    --provider "GITHUB" \
    --encryption-key "arn:aws:kms:region-id:123456789012:key/bPxRfiCYEXAMPLEKEY" \
    --tags key=mytag1,value=value1 key=mytag2,value=value2
```

最後兩個參數--encryption-key和--tags是可選的。

回應：

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
        "connectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/2ad03b28-a7c4-EXAMPLE11111",
        "encryptionKey": "arn:aws:kms:region-id:123456789012:key/
bPxRfiCYEXAMPLEKEY",
        "name": "myrepos/environments",
        "provider": "GITHUB"
    }
}
```

建立儲存庫連結後，您可以檢視以AWS及客戶管理的標籤清單，如下列範例指令所示。 AWS Proton自動為您產生AWS受管理的標籤。您也可以使用修改和建立客戶管理的標籤AWS CLI。如需詳細資訊，請參閱AWS Proton資源和標記。

命令：

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments"
```

檢視連結儲存庫資料

您可以使用主控台或列出並檢視連結的儲存庫詳細資訊AWS CLI。對於用來同步 git 儲存庫的儲存庫連結AWS Proton，您可以使用AWS CLI.

AWS Management Console

使用主控台列出並檢視連結的儲存庫詳細資AWS Proton訊。

1. 若要列出連結的儲存庫，請在導覽窗格中選擇「儲存庫」。
2. 若要檢視詳細資料，請選擇儲存庫的名稱。

AWS CLI

列出連結的儲存庫。

執行以下命令：

```
$ aws proton list-repositories
```

回應：

```
{
    "repositories": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
            "name": "myrepos/templates",
            "provider": "GITHUB"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
            "name": "myrepos/environments",
            "provider": "GITHUB"
        }
    ]
}
```

檢視連結儲存庫的詳細資訊。

執行以下命令：

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

回應：

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

列出已同步的儲存庫。

下列範例會列出您為範本同步所設定的儲存庫。

執行以下命令：

```
$ aws proton list-repository-sync-definitions \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

檢視儲存庫同步狀態。

以下範例會擷取範本同步庫的同步狀態。

執行以下命令：

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

回應：

```
{
    "latestSync": {
        "events": [
            {
                "event": "Clone started",
```

```
                "time": "2021-11-21T00:26:35.883000+00:00",
                "type": "CLONE_STARTED"
            },
            {
                "event": "Updated configuration",
                "time": "2021-11-21T00:26:41.894000+00:00",
                "type": "CONFIG_UPDATED"
            },
            {
                "event": "Starting syncs for commit 62c03ff86eEXAMPLE1111111",
                "externalId": "62c03ff86eEXAMPLE1111111",
                "time": "2021-11-21T00:26:44.861000+00:00",
                "type": "STARTING_SYNC"
            }
        ],
        "startedAt": "2021-11-21T00:26:29.728000+00:00",
        "status": "SUCCEEDED"
    }
}
```

刪除儲庫連結

您可以使用主控台或來刪除儲庫連結AWS CLI。

> **Note**
>
> 刪除儲存庫連結只會移除您AWS帳戶中AWS Proton已註冊的連結。不會刪除儲庫中的儲庫中。

AWS Management Console

使用主控台刪除存放庫連結。

在存放庫詳細資訊頁面中。

1. 在AWS Proton主控台中，選擇 [儲存庫]。
2. 在儲庫清單中，選擇您要刪除的儲庫左側的選項按鈕。
3. 選擇 刪除 。
4. 強制回應會提示您確認刪除動作。

5. 按照說明進行操作，然後選擇是，刪除。

AWS CLI

刪除存放庫連結。

執行以下命令：

```
$ aws proton delete-repository \
    --name myrepos/templates \
    --provider"GITHUB"
```

回應：

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

監控 AWS Proton

監控是維持 AWS 解決方案的可靠性、可用性和效能的 AWS Proton 重要組成部分。下節說明您可以搭配使用的監視工具 AWS Proton。

AWS Proton 使用自動化 EventBridge

您可以在 Amazon 監控 AWS Proton 事件 EventBridge。 EventBridge 從您自己的應用程式、software-as-a-service (SaaS) 應用程式和 AWS 服務. 您可以設定事件以回應資 AWS 源狀態變更。EventBridge 然後將此數據路由到目標服務，例如 AWS Lambda 和 Amazon 簡單通知服務。這些事件與 Amazon 活動中出現的 CloudWatch 事件相同。 CloudWatch Events 提供近乎即時的系統事件串流，用來描述 AWS 資源變更。有關更多信息，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

用 EventBridge 於在 AWS Proton 佈建工作流程中收到狀態變更的通知。

事件類型

事件是由包含事件模式和目標的規則所組成。您可以選擇事件模式和目標物件來設定規則：

事件模式

每個規則都以事件模式表示，其中包含要監視的事件來源和類型以及事件目標。若要監視事件，請使用您正在監視的服務建立規則作為事件來源。例如，您可以建立具有事件模式的規則，該規則會用 AWS Proton 作事件來源，以便在部署狀態發生變更時觸發規則。

目標

規則會接收選取的服務做為事件目標。您可以設定目標服務來傳送通知、擷取狀態資訊、採取更正動作、起始事件或採取其他動作。

事件物件包含 ID、帳號、詳細資料類型 AWS 區域、來源、版本、資源、時間 (選用) 的標準欄位。詳細資料欄位是包含事件自訂欄位的巢狀物件。

AWS Proton 事件會以最佳的方式發出。盡力傳遞意味著服務會嘗試將所有事件傳送至 EventBridge，但在極少數情況下，事件可能無法傳遞。

下表會針對每個可發出事件的 AWS Proton 資源，列出詳細資料類型值、明細欄位，以及 (如果有的話) `status` 和`previousStatus`明細欄位值清單的參照。刪除資源時，`status`明細欄位值為`DELETED`。

資源	詳細類型值	詳細資料欄
EnvironmentTemplate	AWS Proton 環境範本狀態變更	name status previousStatus
EnvironmentTemplateVersion	AWS Proton 環境範本版本狀態變更	name majorVersion minorVersion status previousStatus 狀態值
ServiceTemplate	AWS Proton 服務範本狀態變更	name status previousStatus
ServiceTemplateVersion	AWS Proton 服務範本版本狀態變更	name majorVersion minorVersion status

資源	詳細類型值	詳細資料欄
		previousStatus 狀態值
Environment	AWS Proton 環境狀態變更	name status previousStatus
Service	AWS Proton 服務狀態變更	name status previousStatus 狀態值
ServiceInstance	AWS Proton 服務實例狀態變更	name serviceName status previousStatus
ServicePipeline	AWS Proton 服務管道狀態變更	serviceName status previousStatus

資源	詳細類型值	詳細資料欄
EnvironmentAccountConnection	AWS Proton 環境帳戶連線狀態變更	id status previousStatus 狀態值
Component	AWS Proton 元件狀態變更	name status previousStatus

AWS Proton 事件範例

下列範例顯示 AWS Proton 可將事件傳送至的方式 EventBridge。

服務範本

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-service-template-name"],
    "detail": {
        "name": "sample-service-template-name",
        "status": "PUBLISHED",
        "previousStatus": "DRAFT"
    }
}
```

服務範本版本

```
{
```

```
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Version Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name:1.0"],
    "detail": {
        "name": "sample-service-template-name",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_FAILED",
        "previousStatus": "REGISTRATION_IN_PROGRESS"
    }
}
```

Environment (環境)

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Environment Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:environment/sample-
environment"],
    "detail": {
        "name": "sample-environment",
        "status": "DELETE_FAILED",
        "previousStatus": "DELETE_IN_PROGRESS"
    }
}
```

EventBridgeTutorial：針對服務狀態變更傳送 Amazon 簡易通知 AWS Proton 服務警示

在本教學課程中，您會使用 AWS Proton 預先設定的事件規則來擷取 AWS Proton 服務的狀態變更。EventBridge將狀態變更傳送至 Amazon SNS 主題。您訂閱主題後，Amazon SNS 就會傳送 AWS Proton 服務的狀態變更電子郵件給您。

必要條件

您擁有`Active`狀態的現有 AWS Proton 服務。在本教學課程中，您可以將服務執行個體新增至此服務，然後刪除執行個體。

如果您需要建立 AWS Proton 服務，請參閱入門。如需詳細資訊，請參閱 AWS Proton 配額 及 the section called “編輯”。

步驟 1：建立並訂閱 Amazon SNS 主題

建立 Amazon SNS 主題，做為您在步驟 2 中建立的事件規則的事件目標。

建立 Amazon SNS 主題

1. 登入並開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 [主題] > [建立主題]。
3. 在建立主題頁面中：
   a. 選擇「類型標準」。
   b. 在「名稱」中，輸入**tutorial-service-status-change**並選擇「建立主題」。
4. 在tutorial-service-status-change詳細資料頁面中，選擇 [建立訂閱]。
5. 在「建立訂閱」頁面中：
   a. 對於通訊協定，選擇電子郵件。
   b. 對於 Endpoint (端點)，輸入您目前能存取的電子郵件地址，並選擇 Create subscription (建立訂閱)。
6. 檢查您的電子郵件帳戶，並等待接收訂閱確認電子郵件訊息。收到後，將其打開並選擇確認訂閱。

步驟 2：註冊事件規則

註冊可擷取 AWS Proton 服務狀態變更的事件規則。如需詳細資訊，請參閱 必要條件。

建立事件規則。

1. 打開 Amazon EventBridge 控制台。
2. 在導覽窗格中，選擇 Events (事件) 、Rules (規則)。
3. 在「規則」頁面的「規則」區段中，選擇「建立規則」。
4. 在「建立規則」頁面中：
   a. 在「名稱和說明」區段中，輸入做為「名稱」**tutorial-rule**。
   b. 在「定義模式」區段中，選擇「事件模式」。

i. 在 Event matching pattern (事件比對模式) 中，選擇 Pre-defined by service (依服務預先定義)。

ii. 針對服務供應商，選擇 AWS。

iii. 對於 Service Name (服務名稱) 中，選擇 AWS Proton。

iv. 對於事件類型，請選擇AWS Proton 服務狀態變更。

事件模式會出現在文字編輯器中。

v. 開啟 AWS Proton 主控台。

vi. 在導覽窗格中，選擇服務。

vii. 在「服務」頁面中，選擇您的 AWS Proton 服務名稱。

viii. 在服務詳細資料頁面中，複製服務 Amazon 資源名稱 (ARN)。

ix. 導覽回EventBridge 主控台和教學課程規則，然後在文字編輯器中選擇 [編輯]。

x. 在文字編輯器中`"resources":`，輸入您在步驟 viii 中複製的服務 ARN。

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

xi. 儲存事件模式。

c. 在 「選取目標」 區段中：

i. 在 Target (目標)，選擇 SNS topic (SNS 主題)。

ii. 對於「主題」，請選擇tutorial-service-status-change。

d. 選擇建立。

步驟 3：測試您的事件規則

將執行個體新增至 AWS Proton 服務，以確認您的事件規則正常運作。

1. 切換至主AWS Proton 控台。
2. 在導覽窗格中，選擇服務。
3. 在「服務」頁面中，選擇您的服務名稱。

4. 在服務詳細資訊頁面中，選擇編輯。
5. 在設定服務頁面中，選擇下一步。
6. 在 [設定自訂設定] 頁面的 [服務執行個體] 區段中，選擇 [新增執行個體]。
7. 為您的新實例填寫表單：
   a. 輸入新執行個體的「名稱」。
   b. 選取您為現有執行個體選擇的相容環境。
   c. 輸入所需輸入的值。
   d. 選擇下一步。
8. 檢閱您的輸入並選擇 [更新]。
9. 服務狀態為後`Active`，請檢查您的電子郵件，以確認您收到的 AWS 通知提供狀態更新。

```
{
    "version": "0",
    "id": "af76c382-2b3c-7a0a-cf01-936dff228276",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:40:16Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "ACTIVE",
        "status": "UPDATE_IN_PROGRESS",
        "name": "your-service"
    }
}
```

```
{
    "version": "0",
    "id": "87131e29-ad95-bda2-cd30-0ce825dfb0cd",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:42:27Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "UPDATE_IN_PROGRESS",
```

```
        "status": "ACTIVE",
        "name": "your-service"
    }
}
```

步驟 4：清理

刪除您的 Amazon SNS 主題和訂閱，並刪除您的 EventBridge 規則。

刪除您的 Amazon SNS 主題和訂閱。

1. 導覽至 Amazon SNS 主控台。
2. 在瀏覽面板中，選擇 Subscriptions (訂閱)。
3. 在「訂閱」頁面中，選擇您對名為的主題所做的訂閱，`tutorial-service-status-change`然後選擇「刪除」。
4. 在導覽面板中，選擇「主題」。
5. 在「主題」頁面中，選擇名為的主題，`tutorial-service-status-change`然後選擇「刪除」。
6. 強制回應會提示您確認刪除。按照指示操作，然後選擇刪除。

刪除您的 EventBridge 規則。

1. 導航到 Amazon EventBridge 控制台。
2. 在導覽窗格中，選擇 Events (事件) 、Rules (規則)。
3. 在「規則」頁面中，選擇名為的規則，`tutorial-rule`然後選擇「刪除」。
4. 強制回應會提示您確認刪除。選擇刪除。

刪除新增的服務執行個體。

1. 導覽至 AWS Proton 主控台。
2. 在導覽窗格中，選擇服務。
3. 在「服務」頁面中，選擇您的服務名稱。
4. 在服務詳細資訊頁面中，選擇編輯，然後選擇下一步。
5. 在 [設定自訂設定] 頁面的 [服務執行個體] 區段中，針對您在本教學課程中建立的服務執行個體選擇 [刪除]，然後選擇 [下一步]。

6. 檢閱您的輸入並選擇 [更新]。
7. 強制回應會提示您確認刪除。按照說明進行操作，然後選擇是，刪除。

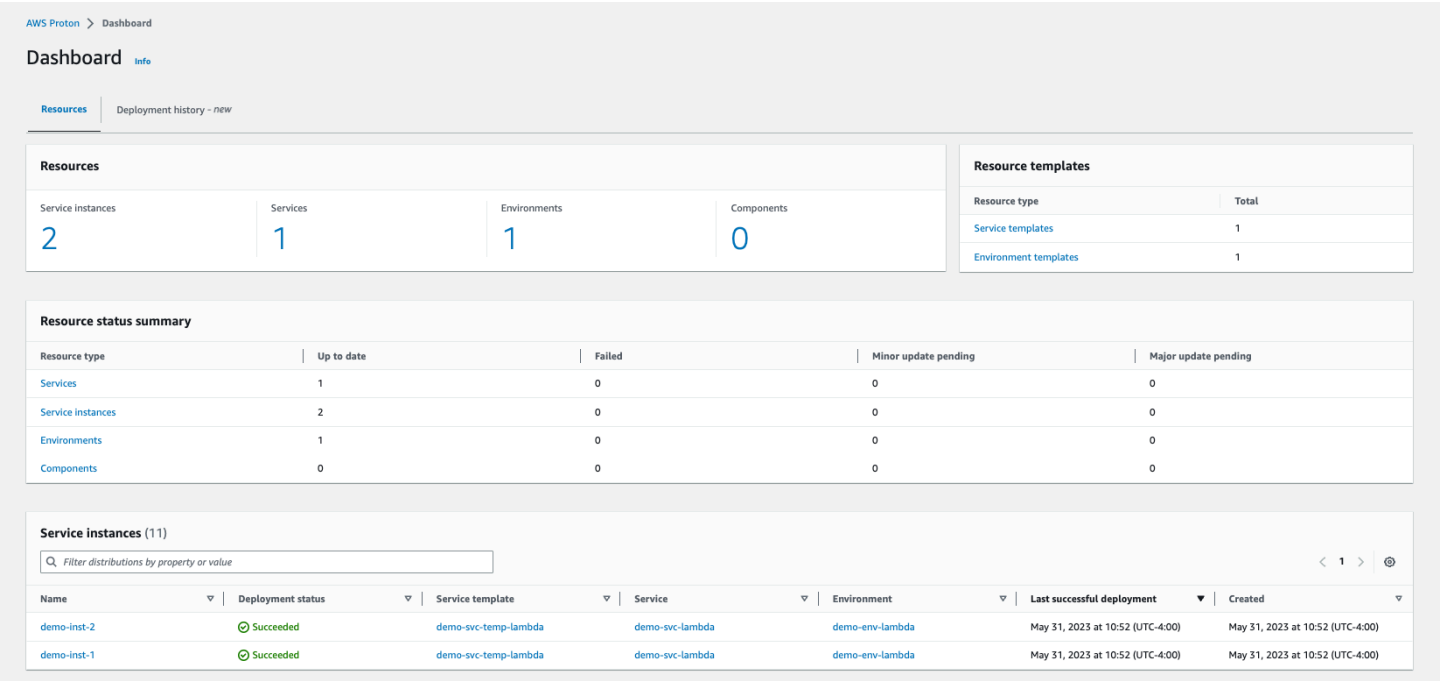
使用 AWS Proton 儀表板讓基礎架構保持最新狀態

AWS Proton 儀表板提供您 AWS 帳戶中 AWS Proton 資源的摘要，特別著重於過時 — 更新部署資源的方式。部署的資源在使用其關聯範本的建議版本時為最新狀態。 out-of-date 已部署的資源可能需要主要或次要範本版本更新。

在 AWS Proton 主控台中檢視儀表板

若要檢視 AWS Proton 儀表板，請開啟AWS Proton 主控台，然後在導覽窗格中選擇 [儀表板]。

資源

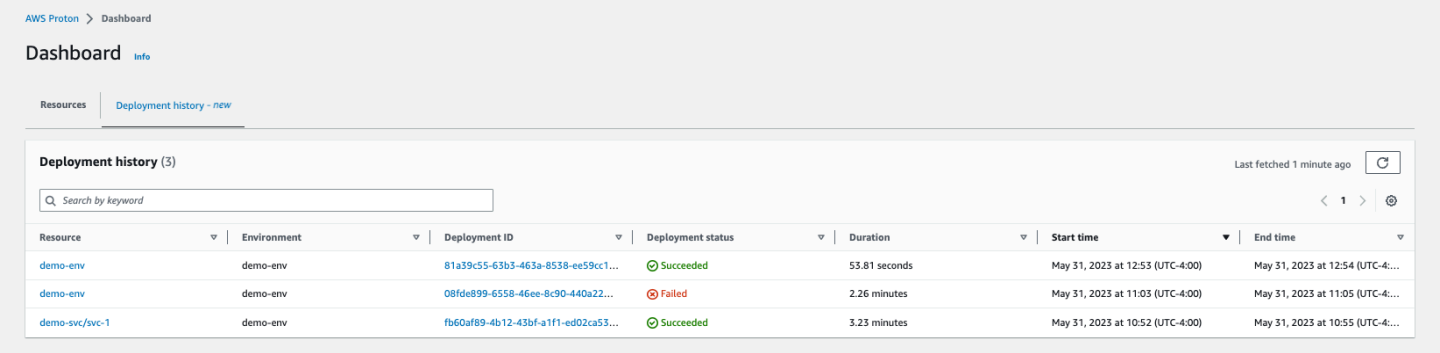


儀表板的第一個索引標籤會顯示帳戶中所有資源的計數。「資源」索引標籤會顯示服務執行個體、服務、環境和元件的數量，以及您的資源範本。它也會依該類型的資源狀態劃分每個已部署資源類型的資源計數。服務執行處理表格會顯示每個服務執行處理的詳細資訊，包括其建置狀態、與其相關聯的 AWS Proton 資源、可用的更新，以及一些時間戳記。

您可以依任何資料表屬性來篩選服務執行個體清單。例如，您可以篩選以查看在特定時間範圍內部署的服務執行個體，或是相對於主要或次要版本建議過期的服務執行個體。

選擇服務執行處理名稱，以瀏覽服務執行處理詳細資訊頁面，您可以在此處進行適當的版本更新。選擇任何其他 AWS Proton 資源名稱，以切換作業選項至其明細頁面，或選擇資源類型以切換作業選項至個別的資源清單。

部署歷史記錄



部署歷史記錄索引標籤可讓您查看有關部署的詳細資料。在部署歷史記錄表格中，您可以追蹤部署狀態，以及環境和部署 ID。您可以選擇資源名稱或部署 ID 來查看更多詳細資料，例如部署狀態訊息和資源輸出。此表格也可讓您篩選任何資料表屬性。

AWS Proton 中的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護在 AWS 雲端 中執行 AWS 服務 的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計劃的一部分。若要了解適用於 AWS Proton 的合規計劃，請參閱合規計劃的 AWS 服務 範圍。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 AWS Proton 時套用共同責任模型。下列主題說明如何將 AWS Proton 設定為達到您的安全及合規目標。您也會了解如何使用其他 AWS 服務 來協助監控並保護 AWS Proton 資源。

主題

- 的 Identity and Access Management AWS Proton
- AWS Proton 中的組態與漏洞分析
- AWS Proton 中的資料保護
- 基礎結構安全 AWS Proton
- AWS Proton 中的記錄和監控
- AWS Proton 中的恢復能力
- AWS Proton 的安全最佳實務
- 預防跨服務混淆代理人
- CodeBuild 佈建自訂亞馬遜 VPC 支援

的 Identity and Access Management AWS Proton

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 AWS Proton 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在進行的工作 AWS Proton。

服務使用者 — 如果您使用 AWS Proton 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 AWS Proton 功能來完成工作時，您可能需要其他權限。了解存取的管理方式可協助您向管理員請求正確的許可。若您無法存取 AWS Proton中的某項功能，請參閱 疑難排解 AWS Proton 身分和存取。

服務管理員 — 如果您負責公司的 AWS Proton 資源，您可能擁有完整的存取權 AWS Proton。決定您的服務使用者應該存取哪些 AWS Proton 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步瞭解貴公司如何搭配使用 IAM AWS Proton，請參閱如何與 IAM AWS Proton 搭配使用。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 AWS Proton存取權的詳細資訊。若要檢視可在 IAM 中使用的 AWS Proton 基於身分的政策範例，請參閱。以身分識別為基礎的原則範例 AWS Proton

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料

都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要

擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

- 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。

- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

如何與 IAM AWS Proton 搭配使用

在您使用 IAM 管理存取權限之前 AWS Proton，請先了解哪些 IAM 功能可搭配使用 AWS Proton。

您可以搭配使用的 IAM 功能 AWS Proton

IAM 功能	AWS Proton 支持
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是

IAM 功能	AWS Proton 支持
主體許可	是
服務角色	是
服務連結角色	是

若要深入瞭解如何以 AWS Proton 及其他如何 AWS 服務 使用大多數 IAM 功能 AWS 服務 ，請參閱 IAM 使用者指南中的 IAM。

以身分識別為基礎的原則 AWS Proton

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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，因為這會套用至連接的使用者或角色。如要了解您在 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。

以身分識別為基礎的原則範例 AWS Proton

若要檢視以 AWS Proton 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Proton

以資源為基礎的政策 AWS Proton

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

的政策動作 AWS Proton

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

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。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 AWS Proton 動作清單，請參閱服務授權參考 AWS Proton中所定義的動作。

中的策略動作在動作之前 AWS Proton 使用下列前置詞：

```
proton
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `List` 文字的所有動作，請包含以下動作：

```
"Action": "proton:List*"
```

若要檢視以 AWS Proton 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Proton

的政策資源 AWS Proton

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

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 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 AWS Proton 資源類型及其 ARN 的清單，請參閱服務授權參考 AWS Proton中所定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS Proton定義的動作。

若要檢視以 AWS Proton 身分為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Proton

的政策條件索引鍵 AWS Proton

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 AWS Proton 條件索引鍵清單，請參閱服務授權參考 AWS Proton中的條件金鑰。若要瞭解您可以使用條件索引鍵的動作和資源，請參閱定義的動作 AWS Proton。

若要檢視限制資源存取權的範例 condition-key-based 政策，請參閱以條件金鑰為基礎的政策範例 AWS Proton。

中的存取控制清單 (ACL) AWS Proton

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

存取控制清單 (ACLs) 是您可以附加資源的授權清單。他們授與帳戶和要附加這些政策的資源的存取許可。

以屬性為基礎的存取控制 (ABAC) 搭配 AWS Proton

支援 ABAC (政策中的標籤)	是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

如需標記 AWS Proton 資源的更多資訊，請參閱AWS Proton資源和標記。

使用臨時登入資料 AWS Proton

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

的跨服務主體權限 AWS Proton

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

的服務角色 AWS Proton

支援服務角色	是

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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

如需詳細資訊，請參閱 AWS Proton IAM 服務角色政策範例。

> **Warning**
>
> 變更服務角色的權限可能會中斷 AWS Proton 功能。只有在 AWS Proton 提供指引時才編輯服務角色。

服務連結角色 AWS Proton

支援服務連結角色	是

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊 AWS 服務 ，請參閱使用 IAM。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

政策範例 AWS Proton

請參閱以下各節中的 AWS Proton IAM 政策範例。

主題

- 以身分識別為基礎的原則範例 AWS Proton
- AWS Proton IAM 服務角色政策範例
- 以條件金鑰為基礎的政策範例 AWS Proton

以身分識別為基礎的原則範例 AWS Proton

根據預設，使用者和角色不具備建立或修改 AWS Proton 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需有關由定義的動作和資源類型的詳細資訊 AWS Proton，包括每個資源類型的 ARN 格式，請參閱服務授權參考 AWS Proton中的動作、資源和條件索引鍵。

主題

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 AWS Proton 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

連結至以識別為基礎的原則範例 AWS Proton

連結至以識別為基礎的範例原則範例 AWS Proton

- AWS 受管理的政策 AWS Proton
- AWS Proton IAM 服務角色政策範例

- 以條件金鑰為基礎的政策範例 AWS Proton

AWS Proton IAM 服務角色政策範例

系統管理員擁有並管理由環境和服務範本所定義 AWS Proton 建立的資源。他們會將 IAM 服務角色附加 AWS Proton 到允許代表其建立資源的帳戶。管理員提供 IAM 角色和 AWS Key Management Service 金鑰，這些資源在後來由開發人員在 AWS Proton 環境中 AWS Proton 部署應用程式作為 AWS Proton 服務時，由開發人員擁有和管理的資源。若要取得有關 AWS KMS 和資料加密的更多資訊，請參閱AWS Proton 中的資料保護。

服務角色是 Amazon Web Services (IAM) 角色，可 AWS Proton 代表您撥打資源呼叫。如果您指定服務角色， AWS Proton 就會使用角色的憑證。使用服務角色可明確指定 AWS Proton 可以執行的動作。

您使用 IAM 服務建立服務角色及其許可政策。如需有關建立服務角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將權限委派給 AWS 服務。

AWS Proton 佈建使用的服務角色 AWS CloudFormation

身為平台小組的成員，您可以作為系統管理員建立 AWS Proton 服務角色，並在建立環境作為環境的 CloudFormation 服務角色 (CreateEnvironmentAPI 動作的`protonServiceRoleArn`參數) AWS Proton 時提供服務角色。當環境或其中執行的任何服務執行個體使用 AWS-managed 佈建並 AWS CloudFormation 佈建基礎結構時，此角色可 AWS Proton 讓您代表您對其他服務進行 API 呼叫。

建議您針對 AWS Proton 服務角色使用下列 IAM 角色和信任政策。當您使用主 AWS Proton 控台建立環境並選擇建立新角色時，這是新 AWS Proton 增至其為您建立的服務角色的原則。設定此原則的權限範圍時，請記住，錯誤時會 AWS Proton 失敗。`Access Denied`

> **Important**
>
> 請注意，下列範例中顯示的策略會將管理員權限授予任何可向您帳戶註冊範本的任何人。由於我們不知道您將在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的權限。我們建議您將權限範圍縮小到將在您的環境中部署的特定資源。

AWS Proton 服務角色原則範例 AWS CloudFormation

*123456789012*用您的 AWS 帳戶 身份證替換。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "NotAction": [
        "organizations:*",
        "account:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeOrganization",
        "account:ListRegions"
```

```
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton 服務信任政策

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

已設定範圍 AWS管理的佈建服務角色原則

以下是範圍停用 AWS Proton 服務角色政策的範例，如果您只需要佈建 S3 資源的 AWS Proton 服務，則可以使用該政策。

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

```
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton CodeBuild 佈建的服務角色

身為平台小組的成員，您可以作為系統管理員建立 AWS Proton 服務角色，並在建立環境作為環境的 CodeBuild 服務角色 (CreateEnvironmentAPI 動作的`codebuildRoleArn`參數) AWS Proton 時提供

服務角色。當環境或其中執行的任何服務執行個體使用佈建來 CodeBuild 佈建基礎結構時，此角色可 AWS Proton 讓您代表您對其他服務進行 API 呼叫。

當您使用 AWS Proton 主控台建立環境並選擇建立新角色時，請將具有系統管理員權限的原則新 AWS Proton 增至其為您建立的服務角色。當您建立自己的角色並縮減權限範圍時，請記住，`Access Denied`錯誤時會 AWS Proton 失敗。

> **Important**
>
> 請注意， AWS Proton 附加至其為您建立之角色的原則，會將管理員權限授與任何可向您帳戶註冊範本的任何人。由於我們不知道您將在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的權限。我們建議您將權限範圍縮小到將在您的環境中部署的特定資源。

AWS Proton 服務角色原則範例 CodeBuild

下列範例提供使用佈 CodeBuild 建資源的權限 AWS Cloud Development Kit (AWS CDK)。

*123456789012*用您的 AWS 帳戶 身份證替換。

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Action": [
       "logs:CreateLogStream",
       "logs:CreateLogGroup",
       "logs:PutLogEvents"
     ],
     "Resource": [
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
Shell-*",
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
Shell-*:*"
     ],
     "Effect": "Allow"
   },
   {
     "Action": "proton:NotifyResourceDeploymentStatusChange",
     "Resource": "arn:aws:proton:us-east-1:123456789012:*",
     "Effect": "Allow"
```

```
    },
    {
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::123456789012:role/cdk-*-deploy-role-*",
        "arn:aws:iam::123456789012:role/cdk-*-file-publishing-role-*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS Proton CodeBuild 信任政策

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "CodeBuildTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS Proton 管線服務角色

若要佈建服務管線， AWS Proton 需要對其他服務進行 API 呼叫的權限。必要的服務角色與您在建立環境時提供的服務角色類似。不過，建立管道的角色會在您 AWS 帳戶中的所有服務之間共用，您可以在主控台中將這些角色提供為 「帳戶」設定，或透過 UpdateAccountSettingsAPI 動作。

當您使用 AWS Proton 主控台更新帳號設定，並選擇為 AWS CloudFormation 或 CodeBuild 服務角色建立新角色時，新 AWS Proton 增至其為您建立的服務角色的政策與上一節所述的策略相同，以AWS-

受管佈建角色及CodeBuild 佈建角色。設定此原則的權限範圍時，請記住，錯誤時會 AWS Proton 失敗。`Access Denied`

> **Important**
>
> 請注意，上一節中的範例策略將管理員權限授予任何可以向您帳戶註冊範本的任何人。由於我們不知道您將在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的權限。我們建議您將權限範圍縮小到將部署在管道中的特定資源。

AWS Proton 元件角色

身為平台小組的成員，您可以作為管理員建立 AWS Proton 服務角色，並在建立環境作為環境的 CloudFormation 元件角色 (CreateEnvironmentAPI 動作的`componentRoleArn`參數) AWS Proton 時提供服務角色。此角色會縮減直接定義元件可佈建的基礎結構的範圍。如需元件的詳細資訊，請參閱元件。

下列範例政策支援建立直接定義的元件，以佈建 Amazon Simple Storage Service (Amazon S3) 儲存貯體和相關存取政策。

AWS Proton 元件角色原則範例

*123456789012*用您的 AWS 帳戶 身份證替換。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
```

```
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS Proton 元件信任原則

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
```

```
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
}
```

以條件金鑰為基礎的政策範例 AWS Proton

下列 IAM 政策範例會拒絕存 AWS Proton 取符合Condition區塊中指定範本的動作。請注意，只有在的動作、資源和條件索引鍵中列出的動作才支援這些條件索引鍵 AWS Proton。若要管理其他動作的權限，例如DeleteEnvironmentTemplate，您必須使用資源層級的存取控制。

拒絕針對特定範本執行 AWS Proton 範本動作的範例原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:EnvironmentTemplate":
 ["arn:aws:proton:region_id:123456789012:environment-template/my-environment-template"]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 ["arn:aws:proton:region_id:123456789012:service-template/my-service-template"]
                }
            }
        }
    ]
}
```

在下一個範例政策中，第一個資源層級陳述式會拒絕存取 AWS Proton 範本動作 (除`ListServiceTemplates`了符合區塊中列出的服務範本外)。`Resource`第二個陳述式會拒絕存 AWS Proton 取符合`Condition`區塊中所列範本的動作。

拒絕符合特定範本之 AWS Proton 動作的範例政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "arn:aws:region_id:123456789012:service-template/my-service-
template"
        },
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate": [
                        "arn:aws:proton:region_id:123456789012:service-template/my-
service-template"
                    ]
                }
            }
        }
    ]
}
```

最終政策範例允許開發人員 AWS Proton 動作符合`Condition`區塊中列出的特定服務範本。

允許符合特定範本的 AWS Proton 開發人員動作的範例政策：

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

```
            "Effect": "Allow",
            "Action": [
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListServices",
                "proton:ListServiceInstances",
                "proton:ListEnvironments",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetEnvironment",
                "proton:CreateService",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "proton:DeleteService",
                "codestar-connections:ListConnections"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 "arn:aws:proton:region_id:123456789012:service-template/my-service-template"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        }

    ]
}
```

AWS 受管理的政策 AWS Proton

若要新增使用者、群組和角色的權限，使用 AWS 受管理的原則比自己撰寫原則更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見的使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱 IAM 使用者指南中的AWS 受管政策。

AWS 服務 維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管理的政策移除權限，因此政策更新不會破壞您現有的權限。

此外，還 AWS 支援跨多個服務之工作職能的受管理原則。例如，ReadOnlyAccess AWS 受管理的策略提供對所有資源 AWS 服務 和資源的唯讀存取權。當服務啟動新功能時， AWS 會為新的操作和資源新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

AWS Proton 提供受管的 IAM 政策和信任關係，您可以將這些關係附加到使用者、群組或角色，以便對資源和 API 操作進行不同層級的控制。您可以直接套用這些政策，也可以使用它們開始建立您自己的政策。

每個 AWS Proton 受管理的策略都會使用下列信任關係。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ExampleTrustRelationshipWithProtonConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
      }
    }
  }
```

```
}
```

AWS 受管理的策略： AWSProtonFullAccess

您可以附加 AWSProtonFullAccess 到 IAM 實體。 AWS Proton 也會將此原則附加至允許代表您執 AWS Proton 行動作的服務角色。

此原則會授與系統管理權限，這些權限允許完整存取 AWS Proton 動作，以及對其他相依 AWS 服務動作的有限存 AWS Proton 取權。

此原則包含下列主要動作命名空間：

- `proton`— 允許管理員完全存取 AWS Proton API。
- `iam`— 允許管理員將角色傳遞給 AWS Proton。這是必要的， AWS Proton 以便可以代表系統管理員對其他服務進行 API 呼叫。
- `kms`— 允許管理員將授權新增至客戶管理的金鑰。
- `codeconnections`— 允許管理員列出並傳遞代碼連接，以便可以使用它們。 AWS Proton

許可詳細資訊

此政策包含以下許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ProtonPermissions",
            "Effect": "Allow",
            "Action": [
                "proton:*",
                "codestar-connections:ListConnections",
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateGrantPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "proton.*.amazonaws.com"
                }
            }
        },
        {
            "Sid": "PassRolePermissions",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CreateServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
sync.proton.amazonaws.com/AWSServiceRoleForProtonSync",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "sync.proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeStarConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
                "codeconnections:PassConnection"
            ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codeconnections:PassedToService": "proton.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS 受管理的策略： AWSProtonDeveloperAccess

您可以附加 AWSProtonDeveloperAccess 到 IAM 實體。 AWS Proton 也會將此原則附加至允許代表您執 AWS Proton 行動作的服務角色。

此原則授與允許有限存取 AWS Proton 動作和其他相 AWS Proton 依 AWS 動作的權限。這些權限的範圍旨在支援建立和部署 AWS Proton 服務的開發人員角色。

此原則不會提供 AWS Proton 範本和環境建立、刪除和更新 API 的存取權。如果開發人員需要的權限甚至超過此原則所提供的限制權限，我們建議您建立一個限定範圍的自訂原則，以授與最低權限。

此原則包含下列主要動作命名空間：

- `proton`— 允許貢獻者訪問一組有限的 AWS Proton API。
- `codeconnections`— 允許貢獻者列出和傳遞代碼連接，以便他們可以被 AWS Proton使用.

許可詳細資訊

此政策包含以下許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ProtonPermissions",
            "Effect": "Allow",
            "Action": [
                "codecommit:ListRepositories",
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineExecution",
                "codepipeline:GetPipelineState",
                "codepipeline:ListPipelineExecutions",
                "codepipeline:ListPipelines",
                "codestar-connections:ListConnections",
                "codestar-connections:UseConnection",
                "proton:CancelServiceInstanceDeployment",
                "proton:CancelServicePipelineDeployment",
                "proton:CreateService",
                "proton:DeleteService",
                "proton:GetAccountRoles",
                "proton:GetAccountSettings",
                "proton:GetEnvironment",
                "proton:GetEnvironmentAccountConnection",
                "proton:GetEnvironmentTemplate",
                "proton:GetEnvironmentTemplateMajorVersion",
                "proton:GetEnvironmentTemplateMinorVersion",
                "proton:GetEnvironmentTemplateVersion",
                "proton:GetRepository",
                "proton:GetRepositorySyncStatus",
                "proton:GetResourcesSummary",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateMajorVersion",
                "proton:GetServiceTemplateMinorVersion",
                "proton:GetServiceTemplateVersion",
                "proton:GetTemplateSyncConfig",
                "proton:GetTemplateSyncStatus",
                "proton:ListEnvironmentAccountConnections",
                "proton:ListEnvironmentOutputs",
```

```
                "proton:ListEnvironmentProvisionedResources",
                "proton:ListEnvironments",
                "proton:ListEnvironmentTemplateMajorVersions",
                "proton:ListEnvironmentTemplateMinorVersions",
                "proton:ListEnvironmentTemplates",
                "proton:ListEnvironmentTemplateVersions",
                "proton:ListRepositories",
                "proton:ListRepositorySyncDefinitions",
                "proton:ListServiceInstanceOutputs",
                "proton:ListServiceInstanceProvisionedResources",
                "proton:ListServiceInstances",
                "proton:ListServicePipelineOutputs",
                "proton:ListServicePipelineProvisionedResources",
                "proton:ListServices",
                "proton:ListServiceTemplateMajorVersions",
                "proton:ListServiceTemplateMinorVersions",
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListTagsForResource",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "s3:ListAllMyBuckets",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeStarConnectionsPermissions",
            "Effect": "Allow",
            "Action": "codestar-connections:PassConnection",
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService": "proton.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
```

```
            "Action": "codeconnections:PassConnection",
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ],
            "Condition": {
                "StringEquals": {
                    "codeconnections:PassedToService": "proton.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS 受管理的策略： AWSProtonReadOnlyAccess

您可以附加 AWSProtonReadOnlyAccess 到 IAM 實體。 AWS Proton 也會將此原則附加至允許代表您執 AWS Proton 行動作的服務角色。

此原則會授與權限，這些權限允許 AWS Proton 動作的唯讀存取權，以及對其他相依 AWS 服務動作的有限唯讀存 AWS Proton 取權。

此原則包含下列主要動作命名空間：

- `proton`— 允許貢獻者對 AWS Proton API 進行唯讀訪問。

許可詳細資訊

此政策包含以下許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListPipelines",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "proton:GetAccountRoles",
        "proton:GetAccountSettings",
```

```
        "proton:GetEnvironment",
        "proton:GetEnvironmentAccountConnection",
        "proton:GetEnvironmentTemplate",
        "proton:GetEnvironmentTemplateMajorVersion",
        "proton:GetEnvironmentTemplateMinorVersion",
        "proton:GetEnvironmentTemplateVersion",
        "proton:GetRepository",
        "proton:GetRepositorySyncStatus",
        "proton:GetResourcesSummary",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetServiceTemplateVersion",
        "proton:GetTemplateSyncConfig",
        "proton:GetTemplateSyncStatus",
        "proton:ListEnvironmentAccountConnections",
        "proton:ListEnvironmentOutputs",
        "proton:ListEnvironmentProvisionedResources",
        "proton:ListEnvironments",
        "proton:ListEnvironmentTemplateMajorVersions",
        "proton:ListEnvironmentTemplateMinorVersions",
        "proton:ListEnvironmentTemplates",
        "proton:ListEnvironmentTemplateVersions",
        "proton:ListRepositories",
        "proton:ListRepositorySyncDefinitions",
        "proton:ListServiceInstanceOutputs",
        "proton:ListServiceInstanceProvisionedResources",
        "proton:ListServiceInstances",
        "proton:ListServicePipelineOutputs",
        "proton:ListServicePipelineProvisionedResources",
        "proton:ListServices",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateVersions",
        "proton:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 受管理的策略： AWSProtonSyncServiceRolePolicy

AWS Proton 將此原則附加至允許 AWS Proton 執行範本同步處理的 AWSServiceRoleForProtonSync 服務連結角色。

此原則授與允許有限存取 AWS Proton 動作和其他相 AWS Proton 依 AWS 服務動作的權限。

此原則包含下列主要動作命名空間：

- `proton`-允許對 AWS Proton API 的 AWS Proton 同步有限訪問。
- `codeconnections`-允許對 CodeConnections API 的 AWS Proton 同步有限訪問。

如需有關的權限詳細資訊 AWSProtonSyncServiceRolePolicy，請參閱的服務連結角色權限。 AWS Proton

AWS 受管理的策略： AWSProtonCodeBuildProvisioningBasicAccess

權限 CodeBuild 需要執行 AWS Proton CodeBuild 佈建的組建。您可以附加`AWSProtonCodeBuildProvisioningBasicAccess`至您的 CodeBuild 佈建角色。

此政策授予 AWS Proton CodeBuild 佈建運作的最低許可。它會授與允許產生 CodeBuild 建置記錄的權限。它還授予 Proton 許可，使基礎架構即代碼（IaC）輸出可供 AWS Proton 用戶使用。它不提供 IaC 工具管理基礎結構所需的權限。

此原則包含下列主要動作命名空間：

- `logs`-允許生 CodeBuild 成構建日誌。沒有此許可， CodeBuild 將無法啟動。
- `proton`-允許調`aws proton notify-resource-deployment-status-change`用 CodeBuild 佈建命令以更新給定 AWS Proton 資源的 IaAC 輸出。

許可詳細資訊

此政策包含以下許可。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
```

```
    "logs:CreateLogGroup",
    "logs:PutLogEvents"
   ],
   "Resource": [
    "arn:aws:logs:*:*:log-group:/aws/codebuild/AWSProton-*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "proton:NotifyResourceDeploymentStatusChange",
   "Resource": "arn:aws:proton:*:*:*"
  }
 ]
}
```

AWS 受管理的策略： AWSProtonCodeBuildProvisioningServiceRolePolicy

AWS Proton 將此原則附加至允許 AWS Proton 執行 CodeBuild基於佈建的 AWSServiceRoleForProtonCodeBuildProvisioning 服務連結角色。

此原則會授與允許有限存 AWS Proton 取相依 AWS 服務動作的權限。

此原則包含下列主要動作命名空間：

- `cloudformation`— 允許 AWS Proton CodeBuild基於佈建對 AWS CloudFormation API 的有限訪問。
- `codebuild`— 允許 AWS Proton CodeBuild基於佈建對 CodeBuild API 的有限訪問。
- `iam`— 允許管理員將角色傳遞給 AWS Proton。這是必要的， AWS Proton 以便可以代表系統管理員對其他服務進行 API 呼叫。
- `servicequotas`-允許檢 AWS Proton 查 CodeBuild 並發構建限制，以確保正確的構建隊列。

許可詳細資訊

此政策包含以下許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
          "servicequotas:GetServiceQuota"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

AWS 受管理的策略： AwsProtonServiceGitSyncServiceRolePolicy

AWS Proton 將此原則附加至允許 AWS Proton 執行服務同步處理的服務 AwsProtonServiceGitSyncServiceRolePolicy 連結角色。

此原則授與允許有限存取 AWS Proton 動作和其他相 AWS Proton 依 AWS 服務動作的權限。

此原則包含下列主要動作命名空間：

- `proton`— 允許 AWS Proton 同步有限存取 AWS Proton API。

如需有關的權限詳細資訊 AwsProtonServiceGitSyncServiceRolePolicy，請參閱的服務連結角色權限。AWS Proton

AWS ProtonAWS 受管理策略的更新

檢視 AWS Proton 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「 AWS Proton 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
AWSProtonFullAccess – 更新現有政策	服務連結角色與 Git 儲存庫搭配使用 Git sync 的受管理原則已針對具有兩個服務前置詞的資源更新。如需詳細資訊，請參閱針對 AWS 使用服務連結角色 CodeConnections和受管政策。	2024年4月25日
AWSProtonDeveloperAccess – 更新現有政策	服務連結角色與 Git 儲存庫搭配使用 Git sync 的受管理原則已針對具有兩個服務前置詞的資源更新。如需詳細資訊，請參閱針對 AWS 使用服務連結角色 CodeConnections和受管政策。	2024年4月25日

變更	描述	日期
AWSProtonSyncServiceRolePolicy – 更新現有政策	服務連結角色與 Git 儲存庫搭配使用 Git sync 的受管理原則已針對具有兩個服務前置詞的資源更新。如需詳細資訊，請參閱針對 AWS 使用服務連結角色 CodeConnections和受管政策。	2024年4月25日
AWSProtonCodeBuildProvisioningServiceRolePolicy – 更新現有政策	AWS Proton 已更新此原則以新增權限，以確保帳戶具有必要的 CodeBuild 並行建置限制，以便使用 CodeBuild 佈建。	2023 年 5 月 12 日
AwsProtonServiceGitSyncServiceRolePolicy – 新政策	AWS Proton 添加了新策略以允許執 AWS Proton 行服務同步。該策略用於AWSServiceRoleForProtonServiceSync服務連結角色。	2023 年 3 月 31 日
AWSProtonDeveloperAccess – 更新現有政策	AWS Proton 已新增新GetResourcesSummary 動作，可讓您檢視範本摘要、已部署的範本資源和過期資源。	2022 年 11 月 18 日
AWSProtonReadOnlyAccess – 更新現有政策	AWS Proton 已新增新GetResourcesSummary 動作，可讓您檢視範本摘要、已部署的範本資源和過期資源。	2022 年 11 月 18 日
AWSProtonCodeBuildProvisioningBasicAccess – 新政策	AWS Proton 添加了一個新的策略，提供了運行 AWS Proton CodeBuild 佈建構 CodeBuild 所需的權限。	2022 年 11 月 16 日

變更	描述	日期
AWSProtonCodeBuildProvisioningServiceRolePolicy – 新政策	AWS Proton 新增政策以允許執 AWS Proton 行與 CodeBuild基礎佈建相關的作業。該策略用於AWSServiceRoleForProtonCodeBuildProvisioning服務連結角色。	2022年9月02 日
AWSProtonFullAccess – 更新現有政策	AWS Proton 已更新此原則，以提供新 AWS Proton API 作業的存取權，並修正部分 AWS Proton 主控台作業的權限問題。	2022 年 3 月 30 日
AWSProtonDeveloperAccess – 更新現有政策	AWS Proton 更新此原則以提供新 AWS Proton API 作業的存取權，並修正某些 AWS Proton 主控台作業的權限問題。	2022 年 3 月 30 日
AWSProtonReadOnlyAccess – 更新現有政策	AWS Proton 更新此原則以提供新 AWS Proton API 作業的存取權，並修正某些 AWS Proton 主控台作業的權限問題。	2022 年 3 月 30 日
AWSProtonSyncServiceRolePolicy – 新政策	AWS Proton 新增政策以允許執 AWS Proton 行與範本同步相關的作業。該策略用於AWSServiceRoleForProtonSync服務連結角色。	2021 年 11 月 23 日
AWSProtonFullAccess – 新政策	AWS Proton 已新增新原則，以提供 AWS Proton API 作業和 AWS Proton 主控台的管理角色存取權。	2021年6月09 日

變更	描述	日期
AWSProtonDeveloperAccess – 新政策	AWS Proton 新增政策以提供 AWS Proton API 作業和 AWS Proton 主控台的開發人員角色存取權。	2021年6月09 日
AWSProtonReadOnlyAccess – 新政策	AWS Proton 新增政策以提供 AWS Proton API 作業和 AWS Proton 主控台的唯讀存取權。	2021年6月09 日
AWS Proton 開始追蹤變更。	AWS Proton 開始追蹤其 AWS 受管理策略的變更。	2021年6月09 日

使用 AWS Proton的服務連結角色

AWS Proton 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 AWS Proton的唯一 IAM 角色類型。服務連結角色由預先定義， AWS Proton 並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

主題

- 使用角色進行 AWS Proton 同步
- 使用角色進行 CodeBuild基於佈建

使用角色進行 AWS Proton 同步

AWS Proton 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 AWS Proton的唯一 IAM 角色類型。服務連結角色由預先定義， AWS Proton 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 AWS Proton 更輕鬆地設定，因為您不需要手動新增必要的權限。 AWS Proton 定義其服務連結角色的權限，除非另有定義，否則只 AWS Proton 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 AWS Proton 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 AWS Proton

AWS Proton 使用兩個名為AWSServiceRoleForProtonSync和AWSServiceRoleForProtonServiceSync的服務連結角色。

服 AWSServiceRoleForProtonSync 務連結角色會信任下列服務擔任該角色：

- `sync.proton.amazonaws.com`

名為的角色權限原則 AWSProtonSyncServiceRolePolicy AWS Proton 允許對指定的資源完成下列動作：

- 動作：建立、管理及閱讀範本和AWS Proton 範本版本
- 動作：使用連線 CodeConnections

AWSProtonSyncServiceRolePolicy

此政策包含以下許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SyncToProton",
            "Effect": "Allow",
            "Action": [
                "proton:UpdateServiceTemplateVersion",
                "proton:UpdateServiceTemplate",
                "proton:UpdateEnvironmentTemplateVersion",
                "proton:UpdateEnvironmentTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetServiceTemplate",
                "proton:GetEnvironmentTemplateVersion",
                "proton:GetEnvironmentTemplate",
                "proton:DeleteServiceTemplateVersion",
                "proton:DeleteEnvironmentTemplateVersion",
                "proton:CreateServiceTemplateVersion",
                "proton:CreateServiceTemplate",
```

```
                "proton:CreateEnvironmentTemplateVersion",
                "proton:CreateEnvironmentTemplate",
                "proton:ListEnvironmentTemplateVersions",
                "proton:ListServiceTemplateVersions",
                "proton:CreateEnvironmentTemplateMajorVersion",
                "proton:CreateServiceTemplateMajorVersion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AccessGitRepos",
            "Effect": "Allow",
            "Action": [
                "codestar-connections:UseConnection",
                "codeconnections:UseConnection"
             ],
            "Resource": [
                "arn:aws:codestar-connections:*:*:connection/*",
                "arn:aws:codeconnections:*:*:connection/*"
            ]
        }
    ]
}
```

如需有關的資訊 AWSProtonSyncServiceRolePolicy，請參閱AWS 受管理的策略：AWSProtonSyncServiceRolePolicy。

服 AWSServiceRoleForProtonServiceSync 務連結角色會信任下列服務擔任該角色：

- `service-sync.proton.amazonaws.com`

名為的角色權限原則 AWSServiceRoleForProtonServiceSync AWS Proton 允許對指定的資源完成下列動作：

- 動作：建立、管理及讀取 AWS Proton 服務和服務執行個體

AwsProtonServiceGitSyncServiceRolePolicy

此政策包含以下許可：

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

```
 "Statement": [
  {
   "Sid": "ProtonServiceSync",
   "Effect": "Allow",
   "Action": [
    "proton:GetService",
    "proton:UpdateService",
    "proton:UpdateServicePipeline",
    "proton:CreateServiceInstance",
    "proton:GetServiceInstance",
    "proton:UpdateServiceInstance",
    "proton:ListServiceInstances",
    "proton:GetComponent",
    "proton:CreateComponent",
    "proton:ListComponents",
    "proton:UpdateComponent",
    "proton:GetEnvironment",
    "proton:CreateEnvironment",
    "proton:ListEnvironments",
    "proton:UpdateEnvironment"
   ],
   "Resource": "*"
  }
 ]
}
```

如需有關的資訊 AwsProtonServiceSyncServiceRolePolicy，請參閱AWS 受管理的策略：AwsProtonServiceSyncServiceRolePolicy。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、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。

為 AWS Proton建立服務連結角色

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 中設定要同步 AWS Proton 的存放庫或服務時 AWS CLI，會為您 AWS Proton 建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您設定要同步處理的存放庫或服務時 AWS Proton， AWS Proton 會再次為您建立服務連結角色。

若要重新建立 AWSServiceRoleForProtonSync 服務連結角色，您需要設定儲存庫以進行同步處理，然後重新建立 AWSServiceRoleForProtonServiceSync，您需要設定服務以進行同步處理。

為 AWS Proton編輯服務連結角色

AWS Proton 不允許您編輯 AWSServiceRoleForProtonSync 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。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。

為 AWS Proton刪除服務連結角色

您不需要手動刪除 AWSServiceRoleForProtonSync 角色。當您刪除 AWS Management Console、或 AWS API 中存放庫同步的所有 AWS Proton 連結存放庫時，會為您 AWS Proton 清除資源並刪除服務連結角色。 AWS CLI

AWS Proton 服務連結角色的支援區域

AWS Proton 支援在所有可用服務的 AWS 區域 地方使用服務連結角色。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS Proton 端點和配額。

使用角色進行 CodeBuild基於佈建

AWS Proton 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 AWS Proton的唯一 IAM 角色類型。服務連結角色由預先定義， AWS Proton 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 AWS Proton 更輕鬆地設定，因為您不需要手動新增必要的權限。 AWS Proton 定義其服務連結角色的權限，除非另有定義，否則只 AWS Proton 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 AWS Proton 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 AWS Proton

AWS Proton 使用名為 AWSServiceRoleForProtonCodeBuildProvisioning— 服務連結角色的服務連結角色進行 AWS Proton CodeBuild 佈建。

服 AWSServiceRoleForProtonCodeBuildProvisioning 務連結角色會信任下列服務擔任該角色：

- `codebuild.proton.amazonaws.com`

名為的角色權限原則 AWSProtonCodeBuildProvisioningServiceRolePolicy AWS Proton 允許對指定的資源完成下列動作：

- 動作：建立、管理和閱讀AWS CloudFormation 堆疊和轉換
- 動作：建立、管理和閱讀CodeBuild 專案和組建

AWSProtonCodeBuildProvisioningServiceRolePolicy

此政策包含以下許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
          "servicequotas:GetServiceQuota"
      ],
      "Resource": "*"
    }
  ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、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。

為 AWS Proton建立服務連結角色

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 中建立使用 CodeBuild基 AWS Proton 於佈建的環境時 AWS CLI， AWS Proton 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您在中建立使用 CodeBuild基於佈建的環境時 AWS Proton，請再次為您建 AWS Proton 立服務連結角色。

為 AWS Proton編輯服務連結角色

AWS Proton 不允許您編輯 AWSServiceRoleForProtonCodeBuildProvisioning 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。

為 AWS Proton刪除服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。但是，您必須先刪除中使用 CodeBuild基於佈建的所有環境和服務 (執行個體和管道)， AWS Proton 才能手動刪除它。

手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForProtonCodeBuildProvisioning服務連結角色。AWS CLI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。

AWS Proton 服務連結角色的支援區域

AWS Proton 支援在所有可用服務的 AWS 區域 地方使用服務連結角色。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS Proton 端點和配額。

疑難排解 AWS Proton 身分和存取

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 AWS Proton 常見問題。

主題

- 我沒有執行操作的授權 AWS Proton
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪 AWS 帳戶 問我的 AWS Proton 資源

我沒有執行操作的授權 AWS Proton

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。您的管理員是為您提供簽署憑證的人員。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 proton:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *my-example-widget* 動作存取 proton:*GetWidget* 資源。

我沒有授權執行 iam：PassRole

如果您收到錯誤，告知您未獲授權執行 iam:PassRole 動作，您的政策必須更新，允許您將角色傳遞給 AWS Proton。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 AWS Proton中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 AWS Proton 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解是否 AWS Proton 支援這些功能，請參閱如何與 IAM AWS Proton 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 IAM 使用者指南中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

AWS Proton 中的組態與漏洞分析

AWS Proton不會為客戶提供的代碼提供修補程序或更新。客戶負責更新修補程序並將修補程序應用於自己的代碼，包括其服務和應用程序的源代碼，這些代碼在AWS Proton以及其服務和環境模板包中提供的代碼。

客戶負責在其環境和服務中更新和修補基礎架構資源。AWS Proton不會自動更新或修補任何資源。客戶應查閱相關文檔，瞭解其體繫結構中的資源，以瞭解各自的修補策略。

除了為次要版本的服務和環境模板提供客戶請求的環境和服務更新外，AWS Proton不會為客戶在其服務和環境模板和模板捆綁包中定義的資源提供修補程序或更新。

如需詳細資訊，請參閱以下 資源：

- 共同的責任模型
- Amazon Web Services：安全流程概觀

AWS Proton 中的資料保護

AWS Proton符合共享責任模型AWS共享責，包含適於資料保護的法規和指導方針。 AWS負責保護執行所有的全球基礎設施AWS 服務。 AWS維護控制在此基礎設施上託管的資料，包括處理客戶內容和個人資料的安全組態。 AWS客戶和 APN 合作夥伴，做為資料控制器或資料處理器，負責它們放置的任何個人資料AWS 雲端

基於資料保護目的，我們建議您保護AWS 帳戶憑證，並使用者帳戶，以便每個使用者都只獲得完成其任務所需的許可。AWS Identity and Access Management我們也建議您採用下列方式保護資料：

- 每個帳戶都使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。建議使用 TLS 1.2 或更新版本。
- 使用 AWS CloudTrail 設定 API 和使用者活動記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格的文字欄位。這包括當您使用 AWS Proton 或使用主控台、API、AWS CLI 或 AWS 開發套件的其他 AWS 服務。您在資源識別碼或與資源管理相關的任何AWS資料都可能選入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

伺服器端靜態加密

如果您選擇在存放範本服務包的 S3 儲存貯體中靜態加密範本服務包中的敏感資料，則必須使用 SSE-S3 或 SSE-KMS 金鑰AWS Proton來允許擷取範本服務包，以便將它們附加至已註冊的AWS Proton範本。

傳輸中加密

服務之間所有通訊在途中都使用 SSL/TLS 進行加密。

AWS Proton加密金鑰管理

在中AWS Proton，預設情況下，所有客戶資料都會使用AWS Proton擁有的金鑰加密。如果您提供客戶擁有且受管理的AWS KMS金鑰，則所有客戶資料都會使用客戶提供的金鑰加密，如以下段落所述。

建立AWS Proton範本時，您可以指定金鑰並AWS Proton使用您的認證來建立允許使AWS Proton用金鑰的授權。

如果您手動淘汰授權，或停用或刪除指定的金鑰，AWS Proton則無法讀取由指定金鑰加密的資料並擲回`ValidationException`。

AWS Proton 加密內容

AWS Proton支援加密內容標頭。加密內容是一組選用的金鑰值對，可以包含資料的其他相關內容資訊。如需有關加密內容的更多資訊，請參閱《AWS Key Management Service 開發人員指南》中的AWS Key Management Service 概念 – 加密內容。

加密密密密密密密密密密密密密密密密密資料。在加密密資料。AWS KMS若要解密資料，您必須傳遞相同的加密內容。

客戶可以使用加密密密密密密密密密密密密密密密金鑰的情況。它還會在日誌中以純文字顯示，例如AWS CloudTrail和 Amazon CloudWatch Logs。

AWS Proton不會接受任何客戶指定或外部指定的加密內容。

AWS Proton添加密密密密密密密密密密密密

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

第一個加密內容會識別與資源相關聯的AWS Proton範本，也可做為客戶管理金鑰權限和授與的限制。

第二個加密內容會識別已加密的AWS Proton資源。

下列範例顯示AWS Proton加密內容的使用方式。

創建服務實例的開發人員。

```
{
```

```
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service/my-service/
service-instance/my-service-instance"
}
```

建立範本的管理員。

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service-template/my-
template"
}
```

基礎結構安全 AWS Proton

作為託管服務， AWS Proton 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透 AWS Proton 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，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

若要改善網路隔離，您可 AWS PrivateLink 以使用下一節所述。

AWS Proton 和介面 VPC 端端點 ()AWS PrivateLink

您可以在 VPC 和 AWS Proton 建立介面 VPC 端點之間建立私人連線。介面端點採用這項技術 AWS PrivateLink，可讓您在沒有網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線的情況下私密存取 AWS Proton API。VPC 中的執行個體不需要公有 IP 位址即可與 AWS Proton API 通訊。您的 VPC 和 AWS Proton 不離開 Amazon 網絡之間的流量。

每個介面端點都是由您子網路中的一或多個彈性網路介面表示。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的介面 VPC 端點 (AWS PrivateLink)。

AWS Proton VPC 端點的考量

在為其設定介面 VPC 端點之前 AWS Proton，請務必先查看 Amazon VPC 使用者指南中的界面端點屬性和限制。

AWS Proton 支援從您的 VPC 呼叫其所有 API 動作。

支援的 AWS Proton VPC 端點原則。依預設，允許透過端點進行完整存取。 AWS Proton 如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制對服務的存取。

為建立介面 VPC 端點 AWS Proton

您可以使用 Amazon VPC 主控台或 AWS Command Line Interface ()AWS CLI建立 AWS Proton 服務的 VPC 端點。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

建立 VPC 端點以 AWS Proton 使用下列服務名稱：

- com.amazonaws.*region*.proton

如果您為端點啟用私有 DNS，則可以 AWS Proton 使用該區域的預設 DNS 名稱發出 API 要求，例如`proton.`*region*`.amazonaws.com`。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的透過介面端點存取服務。

建立 VPC 端點原則 AWS Proton

您可以將端點政策連接至控制 AWS Proton存取權限的 VPC 端點。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：用於動作的 VPC 端點原則 AWS Proton

以下是的端點策略範例 AWS Proton。連接至端點時，此策略會授與所有資源上所有主參與者所列 AWS Proton 動作的存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServices",
        "proton:ListServiceInstances",
        "proton:ListEnvironments",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetEnvironment",
        "proton:CreateService",
        "proton:UpdateService",
        "proton:UpdateServiceInstance",
        "proton:UpdateServicePipeline",
        "proton:DeleteService"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

AWS Proton 中的記錄和監控

監控是維護可靠性、可用性和效能的重要環節。AWS Proton和您的其他AWS解決方案。AWS提供了以下監控工具來監視您在AWS Proton，在發現錯誤時回報，並適時自動採取動作。

在這個時候,AWS Proton本身未與 Amazon CloudWatch Logs 整合，或AWS Trusted Advisor。管理員可以配置和使用 CloudWatch 來監控其他AWS 服務，如其服務和環境模板中定義的。AWS Proton已與整合AWS CloudTrail。

- Amazon CloudWatch 會即時監控您的 AWS 資源，以及您在 AWS 上執行的應用程式。您可以收集和追蹤指標、建立自訂儀表板，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。

例如，您可以讓 CloudWatch 追蹤 CPU 使用量或其他 Amazon EC2 執行個體指標，並在需要時自動啟動新的執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

- Amazon CloudWatch Logs可讓您從 Amazon EC2 執行個體、CloudTrail 及其他來源監控、存放及存取日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- AWS CloudTrail 擷取您 AWS 帳戶 發出或代表發出的 API 呼叫和相關事件，並傳送記錄檔案至您指定的 Amazon S3 儲存貯體。您可以找出哪些使用者和帳戶呼叫 AWS、發出呼叫的來源 IP 地址，以及呼叫的發生時間。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》。
- Amazon EventBridge是無伺服器 EventBridge 服務，可讓您輕鬆將應用程式與來自各種來源的資料互相連線。EventBridge 可從自己的應用程式、軟體即服務 (SaaS) 應用程式和AWS 服務並將該數據路由到目標（如 Lambda）。這可讓您監控在服務中發生的事件，並建置事件導向的架構。如需詳細資訊，請參閱「」AWS Proton 使用自動化 EventBridge與EventBridge 使用者指南。

AWS Proton 中的恢復能力

所以此AWS全球基礎設施是以AWS 區域和可用區域。AWS 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聯網功能相互連結。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 全球基礎設施。

除了 AWS 全球基礎設施外，AWS Proton 還提供支援資料復原和備份需求的多項功能。

AWS Proton備份

AWS Proton維護所有客户數據的備份。如果發生全部停機，您可以使用此備份來還原AWS Proton以及來自以前有效狀態的客户資料。

AWS Proton 的安全最佳實務

AWS Proton在您開發與實作自己的安全政策時，可考慮使用提供的安全功能。以下最佳實務為一般準則，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

- 使用 IAM 控制存取
- 請勿在您的範本和範本包中內嵌登入資料
- 使用加密保護敏感數據
- 使用AWS CloudTrail來查看和記錄 API 呼叫

使用 IAM 控制存取

IAM 是一種AWS 服務中管理使用者及其許可AWS。您可以對使用 IAMAWS Proton來指定哪個AWS Proton管理員和開發人員可以執行的操作，例如管理模板、環境或服務。您可以使用 IAM 服務角色允許AWS Proton代表您呼叫其他服務。

如需詳細資訊AWS Proton和 IAM 角色，請參閱的 Identity and Access Management AWS Proton。

實作最低權限存取。如需詳細資訊，請參閱「」IAM 中的政策和許可中的AWS Identity and Access Management使用者指南。

請勿在您的範本和範本包中內嵌登入資料

而不是將敏感信息嵌入AWS CloudFormation模板和模板捆綁包，我們建議您使用動態參考在堆棧模板中。

動態參考提供簡潔、強大的方式，讓您參考在其他服務 (例如AWS Systems Manager參數存放區或 AWS Secrets Manager。當您使用動態參考時，在堆疊和變更集操作期間，CloudFormation 會在必要時擷取指定參考的值，並將值傳遞至適當的資源。不過，CloudFormation 絕不會存放實際參考值。如需詳細資訊，請參閱「」使用動態參考來指定範本值中的AWS CloudFormation使用者指南。

AWS Secrets Manager 可協助您安全地加密、存放與擷取資料庫及其他服務的登入資料。所以此AWS Systems Manager參數存放區會提供安全的階層式儲存空間，進行組態資料管理。

如需定義範本參數的詳細資訊，請參閱https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html中的AWS CloudFormation使用者指南。

使用加密保護敏感數據

WHINAWS Proton，默認情況下，所有客户數據都會使用AWS Proton擁有的密鑰。

作為平台團隊的成員，您可以提供客户管理的密鑰來AWS Proton來加密和保護您的敏感數據。對 S3 存儲桶中靜態敏感數據進行加密。如需詳細資訊，請參閱 AWS Proton 中的資料保護。

使用AWS CloudTrail來查看和記錄 API 呼叫

AWS CloudTrail會追蹤在您的AWS 帳戶。每當任何人使用AWS ProtonAPI，AWS Proton主控台或AWS Proton AWS CLI命令。啟用記錄日誌，然後指定 Amazon S3 儲存貯體來存放日誌。如此一來，如果您需要，即可稽核誰做AWS Proton在您的帳户中調用。如需詳細資訊，請參閱AWS Proton 中的記錄和監控。

預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

若要限制 AWS Proton 為資源提供另一項服務的許可，我們建議在資源政策中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵。如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Simple Storage Service (Amazon S3) 儲存貯體 ARN)，您必須使用這兩個全域條件內容索引鍵來限制許可。如果同時使用這兩個全域條件內容索引鍵，且 `aws:SourceArn` 值包含帳戶 ID，則在相同政策陳述式中使用 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，則請使用 `aws:SourceAccount`。

的值`aws:SourceArn`必須是AWS Proton存放區。

防範混淆代理人問題的最有效方法是使用 `aws:SourceArn` 全域條件內容金鑰，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 `aws:SourceArn` 全域條件內容金鑰，同時使用萬用字元 (*) 表示 ARN 的未知部分。例如：`arn:aws::proton:*:123456789012:environment/*`。

下列範例示範如何使用 AWS Proton 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容金鑰，來預防混淆代理人問題。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "ExampleProtonConfusedDeputyPreventionPolicy",
        "Effect": "Allow",
```

```
        "Principal": {"Service": "proton.amazonaws.com"},
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws::proton:*:123456789012:environment/*"
            }
        }
    }
}
```

CodeBuild 佈建自訂亞馬遜 VPC 支援

AWS Proton CodeBuild 佈建會在位於AWS Proton環境帳戶的 CodeBuild 專案中，執行客戶提供的任意 CLI 命令。這些命令通常管理資源使用基礎結構作為代碼（IaC）工具，如 CDK。如果您在 Amazon VPC 中有資源， CodeBuild 可能無法存取這些資源。若要啟用此功能，請 CodeBuild 支援在特定 Amazon VPC 內執行的功能。一些示例用例包括：

- 從自託管的內部成品存儲庫中檢索依賴關係，PyPIMaven例如用npm於 Python，Java 和 Node.js
- CodeBuild 需要訪問詹金斯服務器在一個特定的亞馬遜 VPC 註冊一個管道。
- 存取 Amazon S3 儲存貯體中的物件，該儲存貯體設定為只允許透過 Amazon VPC 端點進行存取。
- 針對隔離在私有子網路上的 Amazon RDS 資料庫中的資料，從您的組建執行整合測試。

如需詳細資訊，請參閱CodeBuild 和 VPC 文件。

如果您希望 CodeBuild 佈建在自訂 VPC 中執行，請AWS Proton提供直接的解決方案。首先，您必須將 VPC ID、子網路和安全群組新增至環境範本。接下來，您將這些值輸入到環境規格中。這將導致為您建立以指定 VPC 為目標的 CodeBuild 專案。

更新環境範本

結構描述

需要將 VPC ID、子網路和安全群組新增至範本結構描述，以便它們存在於環境規格中。

一個例子schema.yaml：

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentInputType"
  types:
    EnvironmentInputType:
      type: object
      properties:
        codebuild_vpc_id:
          type: string
        codebuild_subnets:
          type: array
          items:
            type: string
        codebuild_security_groups:
          type: array
          items:
            type: string
```

這將添加三個將由清單使用的新屬性：

- codebuild_vpc_id
- codebuild_subnets
- codebuild_security_groups

清單檔案

若要在中設定 Amazon VPC 設定 CodeBuild，範本資訊清單中project_properties提供名為的選用屬性。的內容project_properties會加入至建立 CodeBuild 專案的AWS CloudFormation堆疊。這使得不僅可以添加 Amazon VPCAWS CloudFormation 屬性，還可以添加任何支持的CodeBuild CloudFormation 屬性，例如構建超時。提供給的相同proton-inputs.json資料可供的值使用project_properties。

將此部分添加到您的manifest.yaml：

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

以下是結果manifest.yaml可能是什麼樣子：

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy -- --force
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

建立環境

使用已啟用 CodeBuild 佈建 vPC 的範本建立環境時，必須提供 Amazon VPC ID、子網路和安全群組。

若要取得區域中所有 Amazon VPC ID 的清單，請執行以命令：

```
aws ec2 describe-vpcs
```

若要取得所有子網路 ID 的清單，請執行以下命令：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> 僅包含私有子網路。 CodeBuild 如果您提供公共子網路，將會失敗。公用子網路具有通往網際網路閘道的預設路由，而私有子網路則沒有。

執行命令以命令以命令取得安全群組 ID。這些 ID 也可以透過以下方式取得AWS Management Console：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

這些值將類似於：

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

確保 CodeBuild 權限

Amazon VPC 支援需要特定許可，例如建立彈性網路界面的能力。

如果要在主控台中建立環境，請在環境建立精靈期間輸入這些值。如果您想以編程方式創建環境，則spec.yaml如下所示：

```

proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton資源和標記

AWS Proton指派 Amazon 資源名稱 (ARN) 的資源包括環境範本及其主要和次要版本、服務範本及其主要和次要版本、環境、服務執行個體、元件和存放庫。您可以對這些資源進行標記，以協助您對其進行整理和識別它們。您可使用標籤來依照用途、擁有者、環境或其他條件分類資源。如需詳細資訊，請參閱 標記策略。若要追蹤和管理您的AWS Proton資源時，您可以使用下列各節所述的標記功能。

AWS標記

您可以用標籤的形式將中繼資料指派給 AWS 資源。每個標籤皆包含客戶定義金鑰和選用值。標籤可協助您管理、識別、組織、搜尋及篩選資源。

> **Important**
>
> 請勿在標籤中加入個人識別資訊 (PII) 或其他機密或敏感資訊。許多 AWS 服務 都可以存取標籤，包括帳單。標籤不適用於私人或敏感資料。

每個 標籤都有兩個部分。

- 標籤鍵 (例如 `CostCenter`、`Environment` 或 `Project`)。標籤鍵會區分大小寫。
- 標記值 (選用) (例如`111122223333`或者`Production`。與標籤鍵相同，標籤值會區分大小寫。

下列基本命名和使用需求適用於標籤。

- 每個資源最多可以有 50 個使用者建立的標籤。

> **Note**
>
> 系統建立的標籤，開頭為`aws:`前綴保留AWS使用，且不會計入此限制之內。您無法編輯或刪除以 `aws:` 字首開頭的標籤。

- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。
- 在 UTF-8 中，標籤金鑰必須至少為 1 且最多為 128 個 Unicode 字元。
- 在 UTF-8 中，標籤值必須至少為 1 且最多為 256 個 Unicode 字元。
- 標籤中允許的字元包括可用 UTF-8 表示的英文字母、數字和空格，還有以下特殊字元：

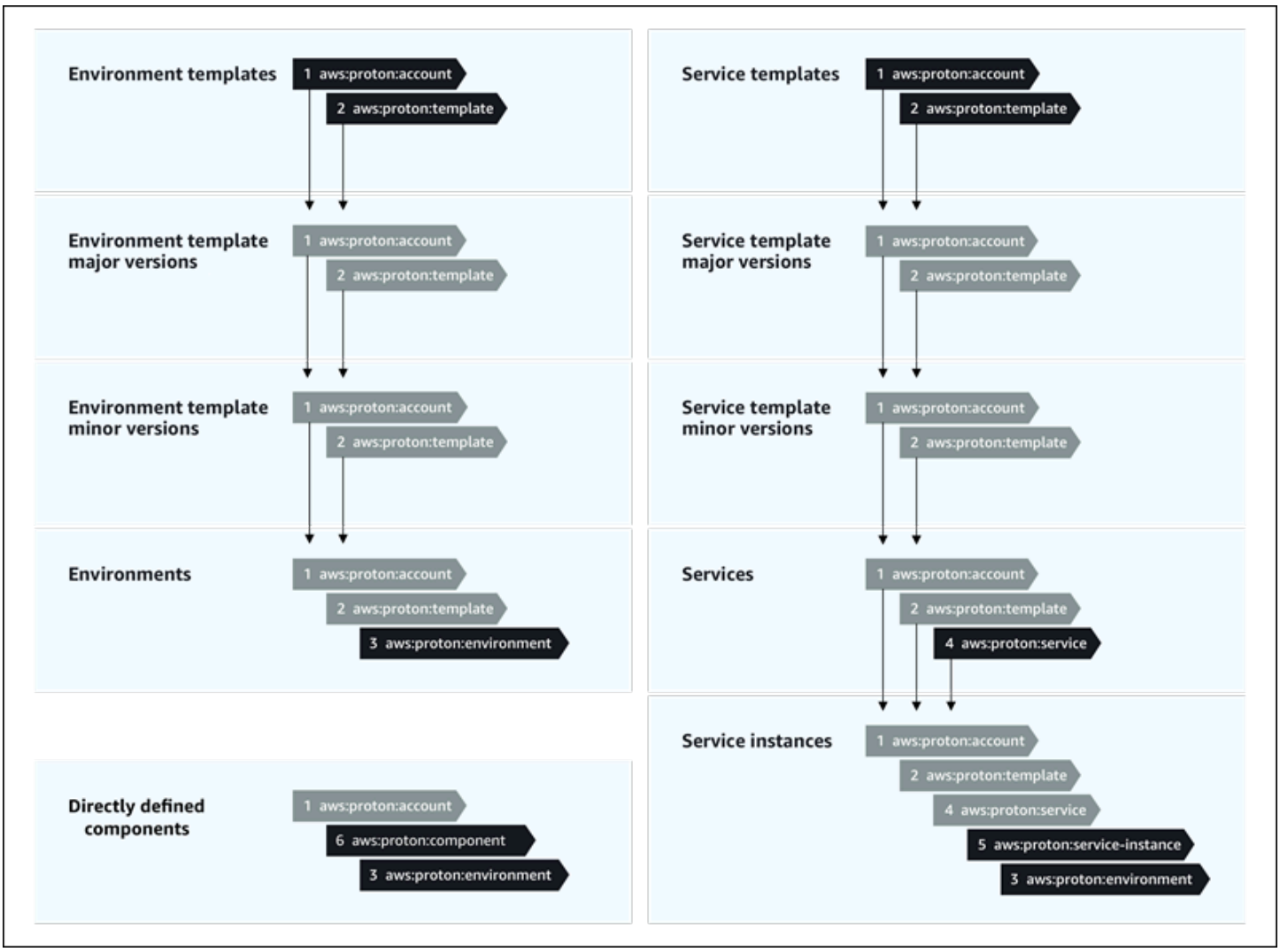
AWS Proton標記

搭配AWS Proton，您可以同時使用您建立的標籤以及標籤AWS Proton會自動為您產生。

AWS Proton AWS受管標記

建立時AWS Proton資源，AWS Proton自動產生AWS管理標記您的新資源，如下圖所示。AWS受管理的標籤稍後會傳播至其他AWS Proton以新資源為基礎的資源。例如，環境範本中的受管理標籤會傳播至其版本，而受管理標籤則會從服務傳播至其服務執行個體。

> **Note**
>
> AWS受管標記不是為環境帳戶連接生成。如需詳細資訊，請參閱 the section called “帳戶連線”。



標籤傳播至已佈建資源

如果佈建的資源 (例如在服務和環境範本中定義的資源)，則支援AWS標記,AWS受管理標籤會以客戶管理標籤的形式傳播至佈建的資源。這些標籤不會傳播到不支援的已佈建資源AWS標記。

AWS Proton將標籤套用至您的資源AWS Proton帳戶、註冊範本和已部署的環境，以及服務和服務執行個體，如下表所述。您可以使用AWS用於查看和管理您的託管標籤AWS Proton資源，但無法修改它們。

AWS受管標記金鑰	傳播客戶受管金鑰	描述
`aws:proton:account`	`proton:account`	所以此AWS建立和部署的帳戶AWS Proton的費用。

AWS受管標記金鑰	傳播客戶受管金鑰	描述
aws:proton:template	proton:template	所選範本的 ARN。
aws:proton:environment	proton:environment	所選環境的 ARN。
aws:proton:service	proton:service	所選服務的 ARN。
aws:proton:service-instance	proton:service-instance	所選服務執行處理的 ARN。
aws:proton:component	proton:component	所選元件的 ARN。

以下是的範例AWS受管理的標籤AWS Proton資源。

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

以下是套用至已佈建資源的客戶受管標籤的範例，AWS受管標記。

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

搭配AWS-管理佈建、AWS Proton將傳播的標籤直接套用至已佈建的資源。

搭配自我管理佈建、AWS Proton使傳輸的標籤與它在佈建提取要求 (PR) 中提交的轉譯 IaC 檔案一起使用。標籤在字符串映射變量中提供proton_tags。我們建議您在 Terraform 組態中參照此變數，以包含AWS Proton標籤default_tags。這傳播AWS Proton所有佈建資源的標籤。

下列範例會示範環境 Terraform 範本中標籤傳播的這種方法。

以下是proton_tags變數定義：

質子環境變量 .tf：

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
```

```
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

以下是標籤值指派給此變數的方式：

質子. 自動. tfvar.json:

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

以下是您可以如何新增AWS Proton標籤到您的 Terraform 配置，以便將它們添加到佈建的資源中：

```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

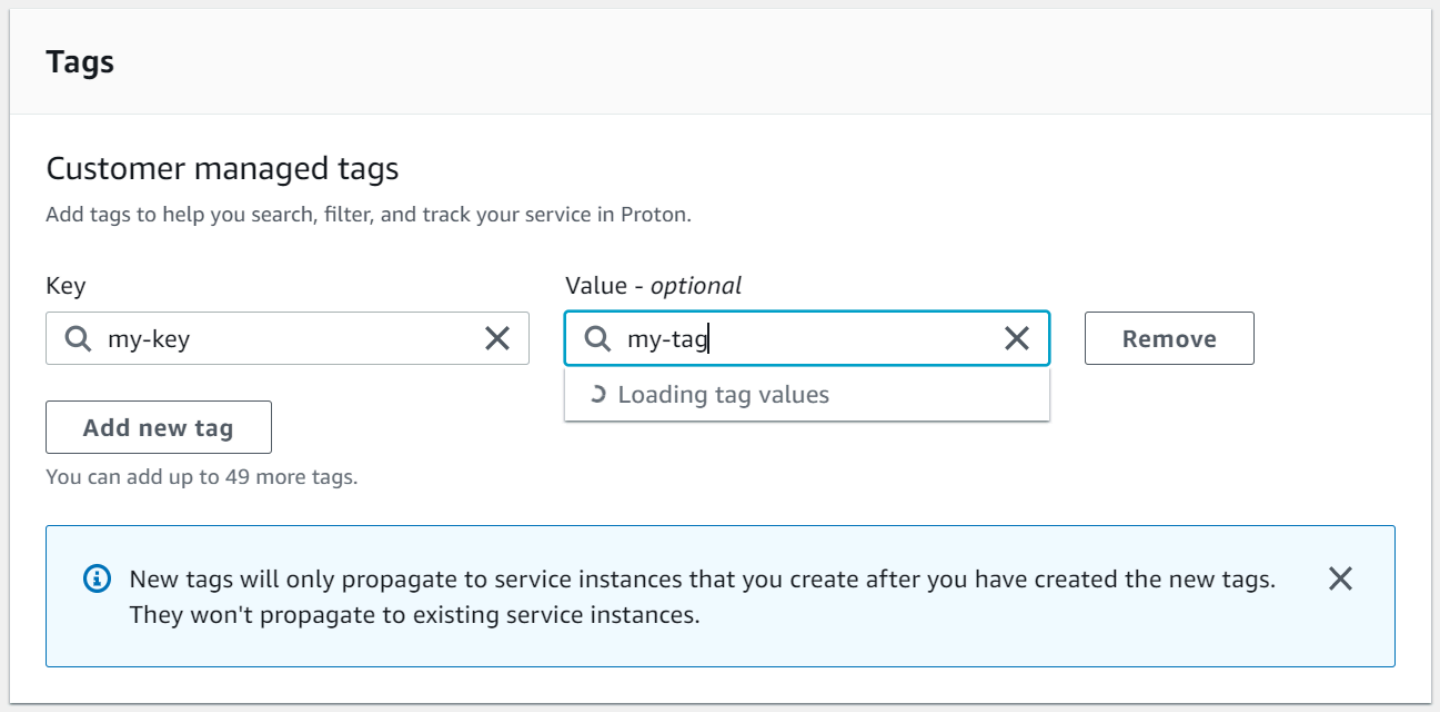
客戶受管標

EACHAWS Proton資源的配額上限為 50 個客戶管理的標籤。客戶管理的標籤會傳播至子系AWS Proton以相同的方式AWS託管標籤可以，除非它們不傳播到現有標籤AWS Proton資源或佈建的資源。

如果您將新標籤套用至AWS Proton具有現有子資源的資源，並且您希望現有的子資源用新標籤標記，您需要使用控制台或手動標記每個現有的子資源AWS CLI。

透過主控台和 CLI 建立標記

建立時AWS Proton使用主控台的資源，您可以在建立程序的第一頁或第二頁上建立客戶管理的標籤，如下列主控台快照集所示。選擇新增標籤」下，進入金鑰和值，然後繼續進行。



建立新的資源之後AWS Proton控制台，您可以查看其列表AWS詳細資訊頁面中的受管理和客戶管理標籤。

建立或編輯標記

1. 在 中AWS Proton安慰，開啟AWS Proton資源詳細信息頁面，您可以在其中查看標籤列表。
2. 選擇 Manage tags (管理標籤)。
3. 在 中管理標籤頁面，您可以查看、建立、移除和編輯標籤。您無法修改AWS列在頂端的受管理標籤。不過，您可以使用編輯欄位新增和修改客戶管理的標籤，列在AWS受管標記。

   選擇新增標籤建立新的標籤。
4. 為新標籤輸入金鑰和值。
5. 若要編輯標籤，請在所選金鑰的標籤值欄位中輸入值。
6. 刪除標記請選擇Remove (移除)用於選取的標籤。

7. 完成變更後，請選擇儲存變更。

使用標記AWS Proton AWS CLI

您可以使用檢視、建立、移除和編輯標籤AWS Proton AWS CLI。

您可以建立或編輯資源的標籤，如下列範例所示。

```
$ aws proton tag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tags '[{"key":"mykey1","value":"myval1"},{"key":"mykey2","value":"myval2"}]'
```

您可以移除資源的標籤，如下一個範例所示。

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

您可以列出資源的標籤，如最後一個範例所示。

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

AWS Proton 疑難排解

了解如何對AWS Proton.

主題

參考AWS CloudFormation動態參數的部署錯誤

如果您看到參考CloudFormation 動態變數的部署錯誤，請確認它們是否已逸出 Jinja。這些錯誤可能是由於 Jinja 對動態變量的誤解造成的。動 CloudFormation 態參數語法與您搭配AWS Proton參數使用的 Jinja 語法非常相似。

CloudFormation 動態變數語法範例：

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

示例AWS Proton參數神社語法：

`'{{ service_instance.environment.outputs.env-outputs }}'`.

為了避免這些誤解錯誤，Jinja 逸出您的 CloudFormation 動態參數，如下列範例所示。

此範例來自《AWS CloudFormation使用者指南》。AWS Secrets Manager秘密名稱和 JSON 鍵段可用於檢索存儲在密鑰中的登錄憑據。

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
```

要轉義 CloudFormation 動態參數，您可以使用兩種不同的方法：

- 在下列項{% raw %} and {% endraw %}目之間括住圖塊：

```
'{% raw %}'
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
'{% endraw %}'
```

- 在下列項"{{ }}"目之間括住參數：

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}' }}"
      MasterUserPassword:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}' }}"
```

有關信息，請參見金剛逃脫。

AWS Proton 配額

下表列出AWS Proton配額。所有值均為AWS帳戶，每個支援AWS區域。

資源配額	預設限制	是否可調整？
範本組合的大小上限	10 MB	X 否
範本資訊清單檔案的大小上限	2 MB	X 否
範本結構描述檔案的大小上限	2 MB	X 否
每個範本檔案的大小上限	2 MB	X 否
每個範本名稱的最大長度	100 個字元	X 否
最大數量CloudFormation每個套裝軟體範本檔	1	X 否
每個帳戶，服務和環境模板的最大註冊模板數量合併	1000	✓ 是
每個範本註冊的範本版本數目上限	1000	✓ 是
每個文件的最大數量CodeBuild佈建服務包	500	X 否
每個帳戶的最大環境數	1000	✓ 是
每個帳戶的服務數目上限	1000	✓ 是
每個服務的服務執行個體數目上限	20	✓ 是
每個帳戶的最大元件數	1000	✓ 是
每個環境帳戶的環境帳戶連線數目上限	1000	✓ 是

文件歷史記錄

下表說明文件中與最新版本 AWS Proton 及客戶意見反應相關的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- 應用程式介面版本:

變更	描述	日期
受管政策更新	AWSProtonDeveloperAccess 政策已更新。	2024年4月25日
受管政策更新	AWSProtonFullAccess 政策已更新。	2024年4月25日
受管政策更新	AWSProtonSyncServiceRolePolicy 政策已更新。	2024年4月25日
受管政策更新	AWSProtonCodeBuildProvisioningServiceRolePolicy 政策已更新。	2023 年 5 月 12 日
服務同步配置。	AWS Proton 添加了對服務同步配置的支持。	2023 年 3 月 31 日
CodeBuild	AWS Proton 增加了對CodeBuild佈建的支援。	2022 年 11 月 16 日
受管政策更新	已新增AWSProtonCodeBuildProvisioningBasicAccess 原則，CodeBuild 提供執行組建進行 AWS Proton CodeBuild 佈建所需的權限。	2022 年 11 月 11 日

地形標籤傳播	在標記章節中新增地形標籤傳播。	2022 年 9 月 16 日
API 移轉指南	移除了 GA 之前的 API 遷移指南。	2022 年 8 月 12 日
AWS Proton 物件	已新增有關 AWS Proton 物件及其與其 AWS 他及第三方物件之間關係的主題。請參閱 AWS Proton 物件。	2022 年 7 月 29 日
鏈接存儲庫澄清	澄清了鏈接（註冊）存儲庫的目的及其在指南中的用法。	2022 年 7 月 18 日
導引合併	將兩份獨立的管理員和使用者指南合併為單一指南，即「AWS Proton 使用者指南」。	2022 年 6 月 30 日
受管政策更新	更新受管原則，可提供新 AWS Proton API 作業的存取權，並修正部分 AWS Proton 主控台作業的權限問題。請參閱的AWS 受管政策 AWS Proton。	2022 年 6 月 20 日
開始使用 CLI	已更新使用新範本資源庫的新自學課程開始使用。 AWS CLI	2022 年 6 月 14 日
直接定義的元件	已新增「元件」章節，並在整個指南中進行了相關修改。	2022 年 6 月 1 日
AWS Proton 範本資源庫	添加了 AWS Proton 模板庫主題。	2022 年 5 月 6 日
地形一般可用性 (GA)	將提取要求佈建重新命名為自我管理佈建。新增佈建方法主題。	2022 年 3 月 23 日

儲存庫標記	新增標記儲存庫資源的支援。請參閱建立存放庫的連結。	2022 年 3 月 23 日
文件更新	添加了環境帳戶連接標記。	2021 年 11 月 26 日
範本同步和地形預覽	添加了具有模板同步功能的自動化模板版本控制以實現正式可用性，並在預覽中使用 Terraform 進行提取請求 API 遷移指南回到.	2021 年 11 月 24 日
文件更新	新增EventBridge教學課程、入門工作流程、AWS Proton 運作方式和範本套件區段增強功能。	2021 年 9 月 17 日
AWS Proton 主控台說明面板版本	說明面板已新增至主控台。主控台範本版本刪除不會再刪除較低版本。API 移轉指南已移除。	2021 年 9 月 8 日
AWS Proton 正式上市 (GA) 版本	新增跨帳戶環境、EventBridge監控、IAM 條件金鑰、冪等性支援，以及增加配額。	2021 年 6 月 9 日
新增和刪除服務的服務執行個體，並將現有的外部基礎結構用於下列環境 AWS Proton	此公開預覽版本包含的更新可讓您從服務新增和刪除服務執行個體、在 AWS Proton 環境中使用現有的外部基礎結構，以及取消環境、服務執行個體和管道部署。 AWS Proton 現在支持PrivateLink。已新增額外的刪除驗證，以防止在資源使用次要版本時錯誤地刪除次要版本。	2021 年 4 月 27 日

標記使用 AWS Proton	公開預覽版本 2 包括 AWS Proton 標記和在沒有服務管線的情況下啟動服務的能力。	2021 年 3 月 5 日
初始版本	公開預覽版現已在特定地區推出。	2020 年 12 月 1 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。