



使用者指南

AWS Proton



AWS Proton: 使用者指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

.....	ix
什麼是 AWS Proton ?	1
平台團隊	1
開發人員	2
工作流程	2
棄用和遷移指南	3
服務狀態直到棄用	3
重要的遷移資訊	3
替代解決方案	3
遷移指引	6
FAQs	6
設定	8
使用 IAM 設定	8
註冊 AWS	8
建立 IAM 使用者	9
服務角色	10
使用 設定 AWS Proton	10
設定 Amazon S3 儲存貯體	11
設定 AWS CodeStar 連線	11
設定帳戶 CI/CD 管道設定	11
設定 AWS CLI	13
開始使用	14
先決條件	14
開始使用工作流程	14
主控台入門	16
步驟 1：開啟 AWS Proton 主控台	16
步驟 2：準備使用範例範本	16
步驟 3：建立環境範本	17
步驟 4：建立服務範本	18
步驟 5：建立環境	19
步驟 6：選用 - 建立服務並部署應用程式	19
步驟 7：清除。	21
CLI 入門	22
1. 註冊環境範本	22

2. 註冊服務範本	23
3. 部署環境	24
4. 部署服務	25
5. 清除	27
範本程式庫	28
如何 AWS Proton 運作	29
物件	30
佈建方法	33
AWS受管佈建	34
CodeBuild 佈建	36
自我管理佈建	38
AWS Proton 術語	40
範本撰寫和套件	43
範本套件	43
Parameters	45
參數類型	45
使用參數	46
環境 CloudFormation IaC 參數	49
Service CloudFormation IaC 參數	54
元件 CloudFormation IaC 參數	56
CloudFormation 參數篩選條件	59
CodeBuild 佈建參數	66
Terraform IaC 參數	67
基礎設施即程式碼檔案	68
CloudFormation IaC 檔案	69
CodeBuild 套件	121
Terraform IaC 檔案	127
結構描述檔案	134
環境結構描述要求	135
服務結構描述需求	138
資訊清單和後續處理	141
環境範本套件包裝	143
服務範本套件後續處理	144
範本套件考量事項	145
範本	146
版本	147

發布	148
發佈環境範本	148
發佈服務範本	155
檢視範本	163
更新範本	167
刪除範本	168
範本同步組態	172
推送遞交	172
同步服務範本	172
範本同步考量事項	173
建立	174
檢視	179
編輯	181
刪除	182
服務同步組態	183
AWS Proton OPS 檔案	183
建立	186
檢視	188
編輯	189
刪除	190
環境	192
IAM 角色	192
AWS Proton 服務角色	192
建立	193
在同一帳戶中建立和佈建	194
在一個帳戶中建立，並在另一個帳戶中佈建	196
自我管理佈建	200
檢視	203
更新	204
更新 AWS 受管佈建環境	205
更新自我管理的佈建環境	208
取消進行中的環境部署	212
刪除	214
帳戶連線	215
建立具有環境帳戶連線的環境	217
管理環境帳戶連線	218

客戶自管	223
使用客戶受管環境	224
CodeBuild 佈建角色建立	225
服務	229
建立	229
服務中有哪些項目？	229
服務範本	230
建立服務	230
檢視	234
編輯	236
編輯服務描述	236
新增或移除服務執行個體	238
刪除	244
檢視執行個體	245
更新執行個體	247
更新管道	252
元件	260
元件與其他資源的比較	261
AWS Proton 主控台	262
AWS Proton API 和 AWS CLI	263
元件常見問答集	263
元件狀態	264
元件 IaC 檔案	265
搭配元件使用參數	266
編寫強大的 IaC 檔案	266
元件 CloudFormation 範例	267
管理員步驟	267
開發人員步驟	270
儲存庫	273
建立儲存庫連結	274
檢視連結的儲存庫資料	275
刪除儲存庫連結	278
監控	280
AWS Proton 使用 EventBridge 自動化	280
Event types (事件類型)	280
AWS Proton 事件範例	283

EventBridgeTutorial：傳送 AWS Proton 服務狀態變更的 Amazon Simple Notification Service 提醒	284
先決條件	284
步驟 1：建立並訂閱 Amazon SNS 主題	285
步驟 2：註冊事件規則	285
步驟 3：測試您的事件規則	286
步驟 4：清理	288
AWS Proton 儀表板	289
AWS Proton 主控台	289
安全	291
身分和存取權管理	291
目標對象	292
使用身分驗證	292
使用政策管理存取權	293
AWS Proton 如何使用 IAM	294
政策範例	299
AWS 受管政策	313
使用服務連結角色	320
疑難排解	324
組態與漏洞分析	325
資料保護	326
伺服器端靜態加密	326
傳輸中加密	326
AWS Proton 加密金鑰管理	327
AWS Proton 加密內容	327
基礎設施安全性	328
VPC 端點 (AWS PrivateLink)	328
日誌記錄和監控	330
恢復能力	331
AWS Proton 備份	331
安全最佳實務	332
使用 IAM 控制存取	332
請勿在範本和範本套件中嵌入登入資料	332
使用加密來保護敏感資料	333
使用 AWS CloudTrail 來檢視和記錄 API 呼叫	333
預防跨服務混淆代理人	333

Codebuild 自訂支援	334
更新環境範本	335
標記	338
AWS 標記	338
AWS Proton 標記	339
AWS Proton AWS 受管標籤	339
將標籤傳播至佈建的資源	340
客戶受管標籤	342
使用主控台和 CLI 建立標籤	342
使用 建立標籤 AWS Proton AWS CLI	343
疑難排解	344
參考 CloudFormation 動態參數的部署錯誤	344
AWS Proton 配額	346
文件歷史紀錄	347
AWS 詞彙表	351

終止支援通知：將於 2026 年 10 月 7 日 AWS 結束對 的支援 AWS Proton。2026 年 10 月 7 日之後，您將無法再存取 AWS Proton 主控台或 AWS Proton 資源。您部署的基礎設施將保持不變。如需詳細資訊，請參閱[AWS Proton 服務棄用和遷移指南](#)。

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。

什麼是 AWS Proton ？

AWS Proton 是：

- 自動化基礎設施做為無伺服器 and 容器型應用程式的程式碼佈建和部署

AWS Proton 服務是雙管齊下的自動化架構。身為管理員，您可以建立版本控制的服務範本，為無伺服器 and 容器型應用程式定義標準化基礎設施和部署工具。身為應用程式開發人員，您可以從可用的服務範本中選取，以自動化您的應用程式或服務部署。

AWS Proton 會為您識別使用過時範本版本的所有現有服務執行個體。身為管理員，只要按一下 AWS Proton 即可請求升級。

- 標準化基礎設施

平台團隊可以使用 AWS Proton 和版本控制的基礎設施做為程式碼範本。他們可以使用這些範本來定義和管理包含架構、基礎設施資源和 CI/CD 軟體部署管道的標準應用程式堆疊。

- 與 CI/CD 整合的部署

當開發人員使用 AWS Proton 自助式界面來選取服務範本時，他們會為其程式碼部署選取標準化的應用程式堆疊定義。AWS Proton 會自動佈建資源、設定 CI/CD 管道，並將程式碼部署到定義的基礎設施。

AWS Proton 適用於平台團隊

身為管理員，您或您的平台團隊成員會建立環境範本和服務範本，其中包含基礎設施做為程式碼。環境範本定義多個應用程式或資源所使用的共用基礎設施。服務範本定義在環境中部署和維護單一應用程式或微服務所需的基礎設施類型。An AWS Proton service 是服務範本的執行個體化，通常包含數個服務執行個體和管道。AWS Proton 服務執行個體是特定環境中服務範本的執行個體。您或團隊中的其他人可以指定哪些環境範本與指定的服務範本相容。如需範本的詳細資訊，請參閱 [AWS Proton 範本](#)。

您可以使用下列基礎設施做為程式碼提供者，搭配 AWS Proton：

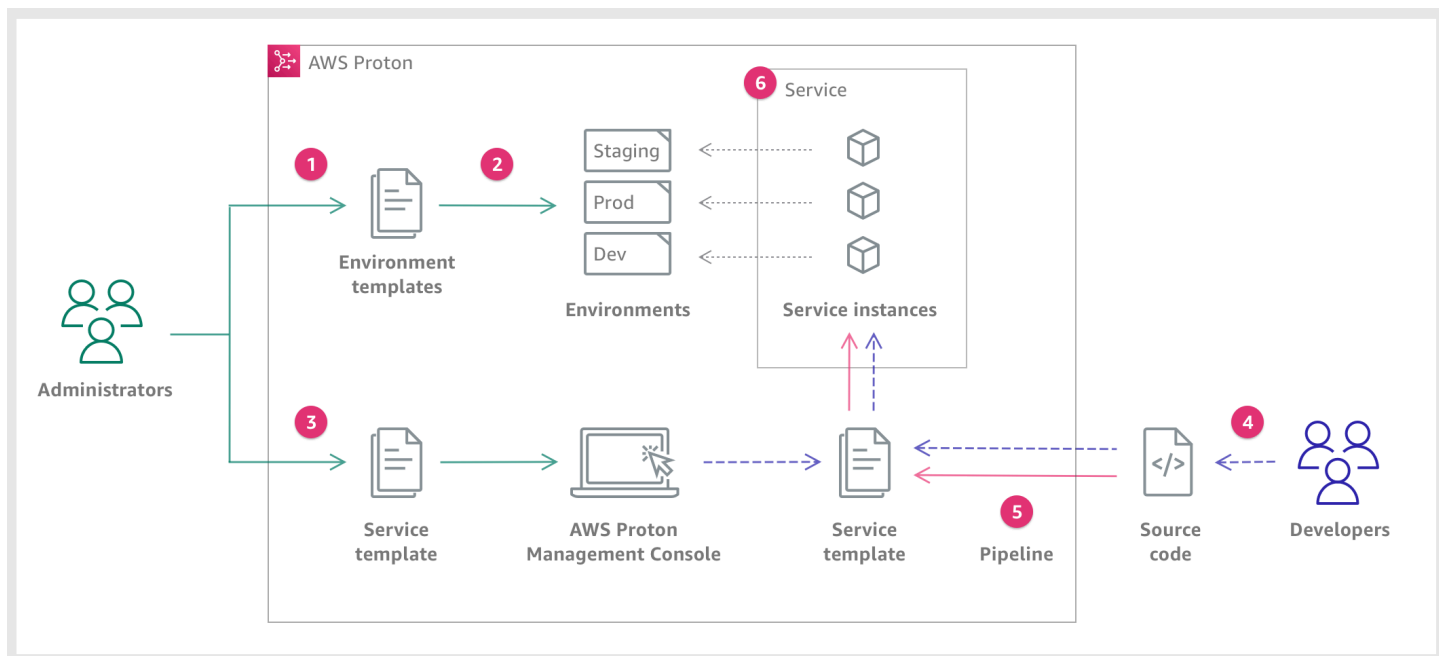
- [CloudFormation](#)
- [Terraform](#)

AWS Proton 開發人員專用

身為應用程式開發人員，您可以選擇標準化服務範本，AWS Proton 使用 來建立服務，以在服務執行個體中部署和管理應用程式。AWS Proton 服務是服務範本的執行個體化，通常包含數個服務執行個體和管道。

AWS Proton 工作流程

下圖是前段討論的主要 AWS Proton 概念的視覺化。它還提供了如何構成簡單 AWS Proton 工作流程的高階概觀。



1

身為管理員，您會使用 建立並註冊環境範本 AWS Proton，以定義共用資源。

身

2

Proton 根據環境範本部署一或多個環境。

AWS

3

身為管理員，您可以建立並註冊服務範本 AWS Proton，其中定義相關的基礎設施、監控和 CI/CD 資源，以及相容的環境範本。

4

身為開發人員，您可以選取已註冊的服務範本，並提供來源碼儲存庫的連結。

5

AWS Proton 為您的服務執行個體佈建具有 CI/CD 管道的服務。

6

AWS Proton 會佈建和管理正在執行來源碼的服務和服務執行個體，如所選服務範本中所定義。服務執行個體是管道單一階段（例如 Prod）環境中所選服務範本的執行個體。

AWS Proton 服務棄用和遷移指南

AWS 已決定停止 AWS Proton，支援將於 2026 年 10 月 7 日結束。新客戶在 2025 年 10 月 7 日之後將無法註冊，但現有客戶仍可繼續使用服務，直到 2026 年 10 月 7 日為止。

服務狀態直到棄用

在 2026 年 10 月 7 日之前，現有 AWS Proton 客戶可以繼續正常使用服務。在此期間，AWS 將：

1. 提供安全修補程式和關鍵錯誤修正
2. 維護服務可用性和效能
3. 繼續透過 AWS 支援 管道提供支援
4. 不將新功能新增至服務

重要的遷移資訊

AWS Proton 主要是用於部署基礎設施的 CI/CD 工具。AWS Proton 棄用時，您部署的 CloudFormation 堆疊及其管理的資源將保持不變並繼續運作。棄用只會影響交付管道和服務 AWS Proton 本身，不會影響已部署的基礎設施。

替代解決方案

我們已找出 的數種替代方案 AWS Proton，可協助您將基礎設施維護為程式碼和 CI/CD 功能。

CloudFormation Git 同步

最適合：使用 CloudFormation 且需要 GitOps 工作流程的團隊

Git 同步可讓平台團隊在開發團隊可以分叉的 git 儲存庫中建立 CloudFormation 範本模型。開發人員更新參數檔案、將變更推送至其分叉儲存庫，以及 Git 同步更新堆疊。

主要優點：

1. 與類似的開發人員體驗 AWS Proton
2. 利用現有的 CloudFormation 知識
3. 平台與開發人員團隊之間的明確區隔

限制:

1. 沒有環境的概念
2. 沒有進階管道功能
3. 依賴其他 Git 供應商可能無法使用的 GitHub 功能

進一步了解：[Git 同步](#)

Harmonix 開啟 AWS

最適合：需要全方位內部開發人員入口網站的企業

Harmonix 是以 Backstage.io 為基礎的 AWS Partner 解決方案，並提供 AWS 外掛程式，可讓團隊建立類似於 Proton 的範本、環境和服務。

主要優點：

1. 與類似的功能 AWS Proton
2. 以熱門的後台架構為基礎建置
3. 完整的開發人員入口網站體驗

限制:

1. 團隊未維護 AWS 服務
2. 可能需要自訂的參考實作

進一步了解：<https://harmonixonaws.io/>

AWS CodePipeline 而且 AWS CodeBuild

最適合：需要最大彈性和控制的團隊

使用 AWS 基礎 CI/CD 服務以更大的彈性和控制複製 AWS Proton 功能。

主要優點：

1. 最大彈性
2. 與 AWS 服務的深度整合
3. 作用中的維護和新功能

限制:

1. 需要更多實作工作
2. 減少out-of-box開發人員自助服務

進一步了解：

[什麼是 AWS CodePipeline](#)

[什麼是 AWS CodeBuild](#)

GitHub 動作

最適合：使用 GitHub 且希望簡化的小型團隊

>主要優點

1. 輕鬆整合 GitHub 儲存庫
2. 簡易設定 GitHub 使用者
3. 大型可重複使用動作市場

限制:

1. 繫結至 GitHub 生態系統
2. 平台團隊控制可能需要更多工作

進一步了解：

[GitHub 動作文件](#)

CI/CD 範例：[與 GitHub 動作整合 – CI/CD 管道，將 Web 應用程式部署至 Amazon EC2](#)

遷移指引

遷移程序取決於您的實作和選擇的替代方案。一般步驟：

1. 清查您的 Proton 資源：
2. 選取替代解決方案：
3. 擷取您的範本資料：
4. 實作您選擇的替代方案：
5. 遷移生產工作負載：

如需特定遷移協助，請聯絡 [AWS 支援](#) 或您的客戶團隊。

FAQs

問：為什麼 AWS 要停止 AWS Proton？答：我們找到了更好的機會來滿足客戶對基礎設施的需求，以便透過其他 AWS 和 AWS Partner 解決方案強制執行程式碼政策。

問：我的現有基礎設施是否會在棄用日期之後繼續運作？答：是。AWS Proton 主要是 CI/CD 工具。您部署的 CloudFormation 堆疊及其管理的資源將保持不變並繼續運作。棄用只會影響交付管道，不會影響已部署的基礎設施。

問：如何取得遷移的協助？答：AWS 支援 可協助遷移。請聯絡 [AWS 支援](#)，或者您可以聯絡 AWS 帳戶 經理尋求協助。

問：我應該選擇哪些替代方案？答：最佳替代方案取決於您的特定使用案例：

1. 對於簡單的 GitOps 工作流程：CloudFormation Git Sync
2. 對於需要開發人員入口網站的企業：Harmonix On AWS
3. 為了獲得最大的彈性：AWS CodePipeline 和 AWS CodeBuild
4. 對於已在 GitHub 上的團隊：GitHub 動作

問：如果我未在 2026 年 10 月 7 日之前遷移，會發生什麼情況？答：您將無法再存取 AWS Proton。您現有的基礎設施將繼續運作，但您將無法使用 AWS Proton 來管理或更新它。

問：我的資料將保留多久？ 答：截至 2026 年 10 月 7 日。在此日期之後，所有資料都會遭到刪除。

如果您有其他問題，請聯絡 AWS 支援。

設定

完成本節中的任務，以便您可以建立和註冊服務和環境範本。您需要這些項目來使用 部署環境和服務 AWS Proton。

Note

我們免費提供 AWS Proton。您可以免費建立、註冊和維護服務和環境範本。您也可以依賴 AWS Proton 自行管理自己的操作，例如儲存、安全性和部署。您使用時產生的唯一費用 AWS Proton 如下。

- 部署和使用您指示 為您 AWS Proton 部署和維護 AWS 雲端 的資源的成本。
- 維護程式碼儲存庫 AWS CodeStar 連線的成本。
- 如果您使用儲存貯體提供輸入，維護 Amazon S3 儲存貯體的成本 AWS Proton。如果您為 切換到 [the section called “範本同步組態”](#) 使用 Git 儲存庫，可以避免這些成本 [the section called “範本套件”](#)。

主題

- [使用 IAM 設定](#)
- [使用 設定 AWS Proton](#)

使用 IAM 設定

當您註冊時 AWS，您的 AWS 帳戶 會自動註冊 中的所有服務 AWS，包括 AWS Proton。您只需為所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，都必須位於相同的帳戶下。

註冊 AWS

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 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或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行 [需要根使用者存取權的任務](#)。

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	根據	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。	在 AWS Command Line Interface 使用者指南中設定 AWS CLI 以使用 來設定 AWS IAM Identity Center 程式設計存取。
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 IAM 使用者以進行緊急存取 的指示。	請依照《IAM 使用者指南》中的 管理 IAM 使用者的存取金鑰 設定以程式設計方式存取。

設定 AWS Proton 服務角色

您可能想要為 AWS Proton 解決方案的不同部分建立幾個 IAM 角色。您可以使用 IAM 主控台事先建立它們，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 Proton 以允許代表您對其他 AWS 服務，例如 CloudFormation、AWS CodeBuild 和各種運算和儲存服務進行 API 呼叫，以為您佈建資源。當環境或其中執行的任何服務執行個體使用 AWS 受管佈建時，需要受管佈建角色。[AWS](#) 當環境或其任何服務執行個體使用 CodeBuild 佈建時，需要 CodeBuild 角色。[CodeBuild](#) 若要進一步了解 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之一選擇現有角色，或為您建立具有管理權限的角色。

同樣地，建立 AWS Proton 管道角色，AWS Proton 以允許代表您對其他服務進行 API 呼叫，為您佈建 CI/CD 管道。若要進一步了解 AWS Proton 管道角色，請參閱 [the section called “管道服務角色”](#)。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 管道設定”](#)。

Note

由於我們不知道您會在 AWS Proton 範本中定義哪些資源，您使用主控台建立的角色具有廣泛的許可，並且可以用作 AWS Proton 管道服務角色、AWS Proton 和服務角色。對於生產部署，我們建議您透過為 AWS Proton 管道服務角色和 AWS Proton 環境服務角色建立自訂政策，將許可範圍縮小到要部署的特定資源。您可以使用 AWS CLI 或 IAM 來建立和自訂這些角色。如需詳細資訊，請參閱 [the section called “IAM 角色”](#)、[the section called “管道服務角色”](#)、[the section called “設定帳戶 CI/CD 管道設定”](#)、[the section called “服務角色 AWS Proton”](#) 及 [建立服務](#)。

使用 設定 AWS Proton

如果您想要使用 AWS CLI 執行 AWS Proton APIs，請確認您已安裝它。如果您尚未安裝，請參閱 [設定 AWS CLI](#)。

AWS Proton 特定組態：

- 若要建立和管理 範本：
 - 如果您使用的是 [範本同步組態](#)，請設定 [AWS CodeStar 連線](#)。
 - 否則，請設定 [Amazon S3 儲存貯體](#)。
- 若要佈建基礎設施：
 - 對於 [自我管理佈建](#)，您必須設定 [AWS CodeStar 連線](#)。
- (選用) 若要佈建管道：

- 針對 [AWS受管佈建](#) 和 [CodeBuild 型佈建](#)，設定 [管道角色](#)。
- 針對 [自我管理佈建](#)，設定 [管道儲存庫](#)。

如需佈建方法的詳細資訊，請參閱 [the section called “AWS受管佈建”](#)。

設定 Amazon S3 儲存貯體

若要設定 S3 儲存貯體，請遵循 [建立第一個 S3 儲存貯體](#) 中的指示來設定 S3 儲存貯體。將您的輸入放在 儲存貯 AWS Proton 體中，其中 AWS Proton 可以擷取它們。這些輸入稱為範本套件。您可以在本指南的其他章節中進一步了解。

設定 AWS CodeStar 連線

若要 AWS Proton 連線至儲存庫，您可以建立 AWS CodeStar 連線，在對第三方原始程式碼儲存庫進行新遞交時啟用管道。

AWS Proton 使用 連線來：

- 當您的儲存庫原始程式碼進行新的遞交時，請啟用服務管道。
- 在 基礎設施上提出提取請求做為程式碼儲存庫。
- 每當遞交推送到變更其中一個範本的範本儲存庫時，如果版本尚未存在，請建立新的範本次要或主要版本。

您可以使用 CodeConnections 連線至 Bitbucket、GitHub、GitHub Enterprise 和 GitHub Enterprise Server 儲存庫。如需詳細資訊，請參閱 AWS CodePipeline 《使用者指南》中的 [CodeConnections](#)。

設定 CodeStar 連線。

1. 開啟 [AWS Proton 主控台](#)。
2. 在導覽窗格中，選取設定，然後選取儲存庫連線，以帶您前往開發人員工具設定中的連線頁面。頁面會顯示連線清單。
3. 選擇建立連線並遵循指示。

設定帳戶 CI/CD 管道設定

AWS Proton 可以佈建 CI/CD 管道，以將應用程式程式碼部署到您的服務執行個體。您需要的管道佈建 AWS Proton 設定取決於您為管道選擇的佈建方法。

AWS受管和 CodeBuild 型佈建 - 設定管道角色

使用 [AWS受管佈建](#) 和 [CodeBuild 佈建](#)，為您 AWS Proton 佈建管道。因此，AWS Proton 需要提供佈建管道許可的服務角色。這兩種佈建方法中的每一個都使用自己的服務角色。這些角色會跨所有 AWS Proton 服務管道共用，您可以在帳戶設定中設定這些角色一次。

使用主控台建立管道服務角色

1. 開啟 [AWS Proton 主控台](#)。
2. 在導覽窗格中，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：
 - 若要讓 為您 AWS Proton 建立管道服務角色

【若要啟用管道的 AWS受管佈建】 在設定帳戶設定頁面的 AWS受管佈建管道角色區段中：

- a. 選取新增服務角色。
- b. 輸入角色的名稱，例如 **myProtonPipelineServiceRole**。
- c. 勾選核取方塊，以同意在您的帳戶中建立具有管理權限 AWS Proton 的角色。

【啟用管道的 CodeBuild 型佈建】 在設定帳戶設定頁面的 CodeBuild 管道角色區段中，選擇現有服務角色，然後選擇您在 CloudFormation 管道角色區段中建立的服務角色。或者，如果您未指派 CloudFormation 管道角色，請重複前三個步驟來建立新的服務角色。

- 選擇現有的管道服務角色

【啟用管道的 AWS受管佈建】 在設定帳戶設定頁面的 AWS受管佈建管道角色區段中，選擇現有服務角色，然後選擇您 AWS 帳戶中的服務角色。

【若要啟用管道的 CodeBuild 佈建】 在設定帳戶設定頁面的 CodeBuild 管道佈建角色區段中，選擇現有服務角色，然後選擇您 AWS 帳戶中的服務角色。

5. 選擇儲存變更。

您的新管道服務角色會顯示在帳戶設定頁面上。

自我管理佈建 - 設定管道儲存庫

透過 [自我管理佈建](#)，會將提取請求 AWS Proton (PR) 傳送至您已設定的佈建儲存庫，而您的自動化程式碼會負責佈建管道。因此，AWS Proton 不需要服務角色來佈建管道。相反地，它需要已註冊的佈建儲存庫。儲存庫中的自動化程式碼必須擔任適當的角色，提供佈建管道的許可。

使用主控台註冊管道佈建儲存庫

1. 如果您尚未建立 CI/CD 管道佈建儲存庫，請建立該儲存庫。如需自我管理佈建中管道的詳細資訊，請參閱 [the section called “自我管理佈建”](#)。
2. 在導覽窗格中，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 管道儲存庫區段中：
 - a. 選取新儲存庫，然後選擇其中一個儲存庫提供者。
 - b. 針對 CodeStar 連線，選擇其中一個連線。

Note

如果您尚未連線到相關的儲存庫提供者帳戶，請選擇新增 CodeStar 連線，完成連線建立程序，然後選擇 CodeStar 連線功能表旁的重新整理按鈕。您現在應該能夠在選單中選擇新的連線。

- c. 針對儲存庫名稱，選擇您的管道佈建儲存庫。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。

設定 AWS CLI

若要使用 AWS CLI 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 搭配 AWS Proton，請參閱 [the section called “CLI 入門”](#)。

入門 AWS Proton

開始使用之前，請先[設定](#) 以使用 ， AWS Proton 並確認您已符合[開始使用先決條件](#)。

選擇下列一或多個路徑 AWS Proton 以開始使用：

- 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](#)
- [開始使用工作流程](#)
- [開始使用 AWS 管理主控台](#)
- [開始使用 AWS CLI](#)
- [AWS Proton 範本程式庫](#)

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](#)。

- 您有具有管理員許可的 IAM 帳戶。如需詳細資訊，請參閱[使用 IAM 設定](#)。
- 您有 AWS Proton 服務角色，而 AWS Proton 管道服務角色會連接到您的帳戶。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](#)。
- 您有 AWS CodeStar 連線。如需詳細資訊，請參閱[設定 AWS CodeStar 連線](#)。
- 您熟悉建立 CloudFormation 範本和 Jinja 參數化。如需詳細資訊，請參閱 CloudFormation 《使用者指南》和 Jinja 網站中的[什麼是 CloudFormation ?](#)。 <https://palletsprojects.com/projects/jinja>
- 您具備 AWS 基礎設施服務的工作知識。
- 您已登入您的 AWS 帳戶。

開始使用工作流程

依照範例步驟和連結，了解如何建立範本套件、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 服務管道，請確認您已建立 [AWS CodeStar 連線](#) 和 [AWS Proton 管道服務角色](#)。

如需詳細資訊，請參閱 [AWS Proton 服務 API 參考](#)。

範例：工作流程入門

1. 如需 AWS Proton 輸入和輸出的高階檢視 [如何 AWS Proton 運作](#)，請參閱中的圖表。
2. [建立環境套件和服務範本套件](#)。
 - a. 識別 [輸入參數](#)。
 - b. 建立 [結構描述檔案](#)。
 - c. 建立 [基礎設施即程式碼 \(IaC\) 檔案](#)。
 - d. 若要 [包裝範本套件](#)，請建立資訊清單檔案，並在目錄中整理 IaC 檔案、資訊清單檔案和結構描述檔案。
 - e. 讓 [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 AWS CLI 建立和註冊範本時：

- 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 Proton API 參考中的 [CreateEnvironmentTemplate](#) 和 [CreateEnvironmentTemplateVersion](#)。

4. [發佈您的環境範本](#)，使其可供使用。

如需詳細資訊，請參閱 AWS Proton API 參考中的 [UpdateEnvironmentTemplateVersion](#)。

5. 若要 [建立環境](#)，請選取已發佈的環境範本版本，並提供必要輸入的值。

如需詳細資訊，請參閱 AWS Proton API 參考中的 [CreateEnvironment](#)。

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 AWS CLI 建立和註冊範本時：

- 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 Proton API 參考中的 [CreateServiceTemplate](#) 和 [CreateServiceTemplateVersion](#)。

7. [發佈您的服務範本](#) 以使其可供使用。

如需詳細資訊，請參閱 AWS Proton API 參考中的 [UpdateServiceTemplateVersion](#)。

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 Proton API 參考中的 [CreateService](#)。

開始使用 AWS 管理主控台

開始使用 AWS Proton

- 建立和檢視環境範本。
- 使用您剛建立的環境範本來建立、檢視和發佈服務範本。
- 建立環境和服務（選用）。
- 如果已建立，請刪除服務範本、環境範本、環境和服務。

步驟 1：開啟 AWS Proton 主控台

- 開啟 [AWS Proton 主控台](#)

步驟 2：準備使用範例範本

1. 建立 Codestar Connection to Github，並命名連線 my-proton-connection。
2. 導覽至 <https://github.com/aws-samples/aws-proton-cloudformation-sample-templates>。
3. 在您的 Github 帳戶中建立儲存庫的分支。

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. 在建立環境範本頁面的範本選項區段中，選擇建立範本以佈建新的環境。
3. 在範本套件來源區段中，選擇從 Git 同步範本套件。
4. 在範本定義儲存庫區段中，選取選擇連結的 Git 儲存庫。
5. 從儲存庫清單中選取 my-proton-connection。
6. 從分支清單中選取主要。
7. 在 Proton 環境範本詳細資訊區段中。
 - a. 將範本名稱輸入為 **fargate-env**。
 - b. 將環境範本顯示名稱輸入為 **My Fargate Environment**。
 - c. (選用) 輸入環境範本的描述。
8. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
9. 選擇建立環境範本。

您現在位於顯示新環境範本狀態和詳細資訊的新頁面。這些詳細資訊包括 AWS 和客戶受管標籤的清單。會在您建立 AWS Proton 資源時 AWS Proton 自動為您產生 AWS 受管標籤。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。您和具有proton>CreateEnvironment許可的其他人可以檢視和存取它。依照下一個步驟，讓範本可供其他人使用。
11. 在範本版本區段中，選擇您剛建立之範本次要版本左側的選項按鈕 (1.0)。或者，您可以在資訊提醒橫幅中選擇發佈，並略過下一個步驟。
12. 在範本版本區段中，選擇發佈。
13. 範本狀態變更為已發佈。由於它是範本的最新版本，因此它是建議版本。
14. 在導覽窗格中，選取環境範本。

新頁面會顯示您的環境範本清單，以及範本詳細資訊。

步驟 4：建立服務範本

建立服務範本。

1. 在導覽窗格中，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 同步範本套件。
4. 在範本區段中，選取選擇連結的 Git 儲存庫。
5. 從儲存庫清單中選取 my-proton-connection。
6. 從分支清單中選取主要。
7. 在 Proton 服務範本詳細資訊區段中。
 - a. 將服務範本名稱輸入為 **backend-fargate-svc**。
 - b. 將服務範本顯示名稱輸入為 **My Fargate Service**。
 - c. (選用) 輸入服務範本的描述。
8. 在相容環境範本區段中。
 - 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. 在管道定義區段中。
 - 保持選取此範本包含 CI/CD 管道按鈕。
11. 選擇建立服務範本。

您現在位於顯示新服務範本狀態和詳細資訊的新頁面，包括 AWS 和 客戶受管標籤的清單。

12. 新服務範本狀態的狀態會以草稿狀態開始。只有管理員可以檢視和存取它。若要讓 服務範本可供開發人員使用，請遵循下一個步驟。
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選項按鈕 (1.0)。或者，您可以在資訊提醒橫幅中選擇發佈，並略過下一個步驟。
14. 在範本版本區段中，選擇發佈。
15. 範本狀態變更為已發佈。

您的服務範本的第一個次要版本已發佈，可供開發人員使用。由於它是範本的最新版本，因此它是建議版本。

16. 在導覽窗格中，選擇服務範本。

新頁面會顯示您的服務範本和詳細資訊清單。

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，選取您剛建立的範本。它名為 My Fargate Environment。然後，選擇設定。
3. 在設定環境頁面的佈建區段中，選擇透過 佈建 AWS Proton。
4. 在部署帳戶區段中，選取此 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，如果您已建立 AWS Proton 服務角色，請選取現有服務角色。
 - a. 選取新服務角色以建立新的角色。
 - i. 將環境角色名稱輸入為 **MyProtonServiceRole**。
 - ii. 勾選核取方塊，以同意為您的帳戶建立具有管理權限 AWS Proton 的服務角色。
 - b. 選取現有服務角色以使用現有角色。
 - 在環境角色名稱下拉式清單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。

檢視環境詳細資訊和狀態，以及環境的 AWS 受管標籤和客戶受管標籤。

11. 在導覽窗格中，選擇 Environments (環境)。

新頁面會顯示您的環境清單，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. 在導覽窗格中，選擇服務。
3. 在服務頁面中，選擇建立服務。
4. 在選擇服務範本頁面中，選擇範本卡右上角的選項按鈕，以選取我的 Fargate 服務範本。
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. (選用) 輸入服務的描述。
8. 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. 針對儲存庫名稱，從清單中選擇來源碼儲存庫的名稱。
 - c. 針對分支名稱，從清單中選擇來源碼儲存庫分支的名稱。
9. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。然後選擇下一步。
10. 在設定自訂設定頁面的服務執行個體區段中，依照下列步驟為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 **my-fargate-environment** 服務執行個體的環境。
 - c. 保留其餘執行個體參數的預設值。
 - d. 保留管道輸入的預設值。
 - 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訊。
11. 在服務詳細資訊頁面中，選擇概觀和管道索引標籤，以檢視您的服務執行個體和管道的狀態。在這些頁面上，您也可以檢視 AWS 和客戶受管標籤。AWS Proton 會自動為您建立 AWS 受管標籤。選擇管理標籤以建立和修改客戶受管標籤。如需標記的相關資訊，請參閱 [AWS Proton 資源和標記](#)。
12. 服務處於作用中狀態後，在概觀索引標籤的服務執行個體區段中，選擇您的服務執行個體名稱 **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。

您可以在網頁中看到 AWS Proton 圖形。
14. 建立服務之後，在導覽窗格中，選擇服務以檢視您的服務清單。

步驟 7：清除。

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

- a. 在導覽窗格中，選擇服務。
- b. 在服務頁面中，選擇服務名稱 my-service。

您現在位於 my-service 的服務詳細資訊頁面。

- c. 在頁面的右上角，選擇動作，然後選擇刪除。
 - d. 模態會提示您確認刪除動作。
 - e. 遵循指示並選擇是，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選項按鈕。
 - c. 依序選擇動作和刪除。
 - d. 模態會提示您確認刪除動作。
 - e. 遵循指示並選擇是，刪除。

4. 刪除服務範本

- a. 在導覽窗格中，選擇服務範本。
- b. 在服務範本頁面中，選取服務範本 my-svc-template 左側的選項按鈕。
- c. 依序選擇動作和刪除。
- d. 模態會提示您確認刪除動作。
- e. 遵循指示並選擇是，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 my-env-template 左側的選項按鈕。
- c. 依序選擇動作和刪除。
- d. 模態會提示您確認刪除動作。
- e. 遵循指示並選擇是，刪除。這會刪除環境範本及其所有版本。

6. 刪除您的 Codestar 連線

開始使用 AWS CLI

若要開始使用 AWS Proton AWS CLI，請遵循本教學課程。本教學課程示範以為基礎的公有負載平衡 AWS Proton 服務 AWS Fargate。本教學課程也會佈建 CI/CD 管道，以部署具有顯示影像的靜態網站。

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，您會註冊範例環境範本，其中包含 Amazon Elastic Container Service (Amazon ECS) 叢集和具有兩個公有/私有子網路的 Amazon Virtual Private Cloud (Amazon VPC)。

註冊環境範本

1. 將[AWS Proton 範例 CloudFormation 範本](#)儲存庫撥入 GitHub 帳戶或組織。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，向註冊您的分叉儲存庫 AWS Proton。如需詳細資訊，請參閱[the section called “建立儲存庫連結”](#)。

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. 建立範本同步組態。

AWS Proton 設定儲存庫與環境範本之間的同步關係。然後，它會建立 DRAFT 處於狀態的範本版本 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。

當此命令傳回的結束狀態時`0`，版本註冊即完成。這在指令碼中非常有用，以確保您可以在下一個步驟中成功執行命令。

```
$ 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，您會註冊範例服務範本，其中包含在負載平衡器和使用的 CI/CD 管道後方佈建 Amazon ECS Fargate 服務所需的所有資源 AWS CodePipeline。

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. 建立範本同步組態。

AWS Proton 會在您的儲存庫和服務範本之間設定同步關係。然後，它會建立DRAFT處於狀態的範本版本 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。

當此命令傳回的結束狀態時`0`，版本註冊即完成。這在指令碼中非常有用，以確保您可以在下一個步驟中成功執行命令。

```
$ 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 環境範本執行個體化環境。

部署環境

1. 取得您所註冊環境範本的範例規格檔案。

您可以從`environment-templates/fargate-env/spec/spec.yaml`範本範例儲存庫下載檔案。或者，您可以在本機擷取整個儲存庫，並從`environment-templates/fargate-env`目錄執行`create-environment`命令。

2. 建立環境。

AWS Proton 會從您的環境規格讀取輸入值、將它們與您的環境範本合併，並使用 AWS Proton 服務角色在您的 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/AWS ProtonServiceRole" \  
  --spec "file://spec/spec.yaml"
```

3. 等待環境成功部署。

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

步驟 4：部署服務 【應用程式開發人員】

在先前的步驟中，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 環境中

部署服務

1. 取得管理員註冊之服務範本的範例規格檔案。

您可以從 `service-templates/load-balanced-fargate-svc/spec/spec.yaml` 範本範例儲存庫下載 檔案。或者，您可以在本機擷取整個儲存庫，並從 `service-templates/load-balanced-fargate-svc` 目錄執行 `create-service` 命令。

2. 將 [AWS Proton Sample Services](#) 儲存庫撥入 GitHub 帳戶或組織。此儲存庫包含我們在本教學課程中使用的應用程式原始碼。

3. 建立服務。

AWS Proton 會從服務規格讀取輸入值、將它們與您的服務範本合併，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 管道會從您在 命令中指定的儲存庫部署您的應用程式程式碼。

```
$ aws proton create-service \  
  --name "static-website" \  
  --spec "file://spec/spec.yaml"
```

```
--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. 擷取輸出並檢視您的新網站。

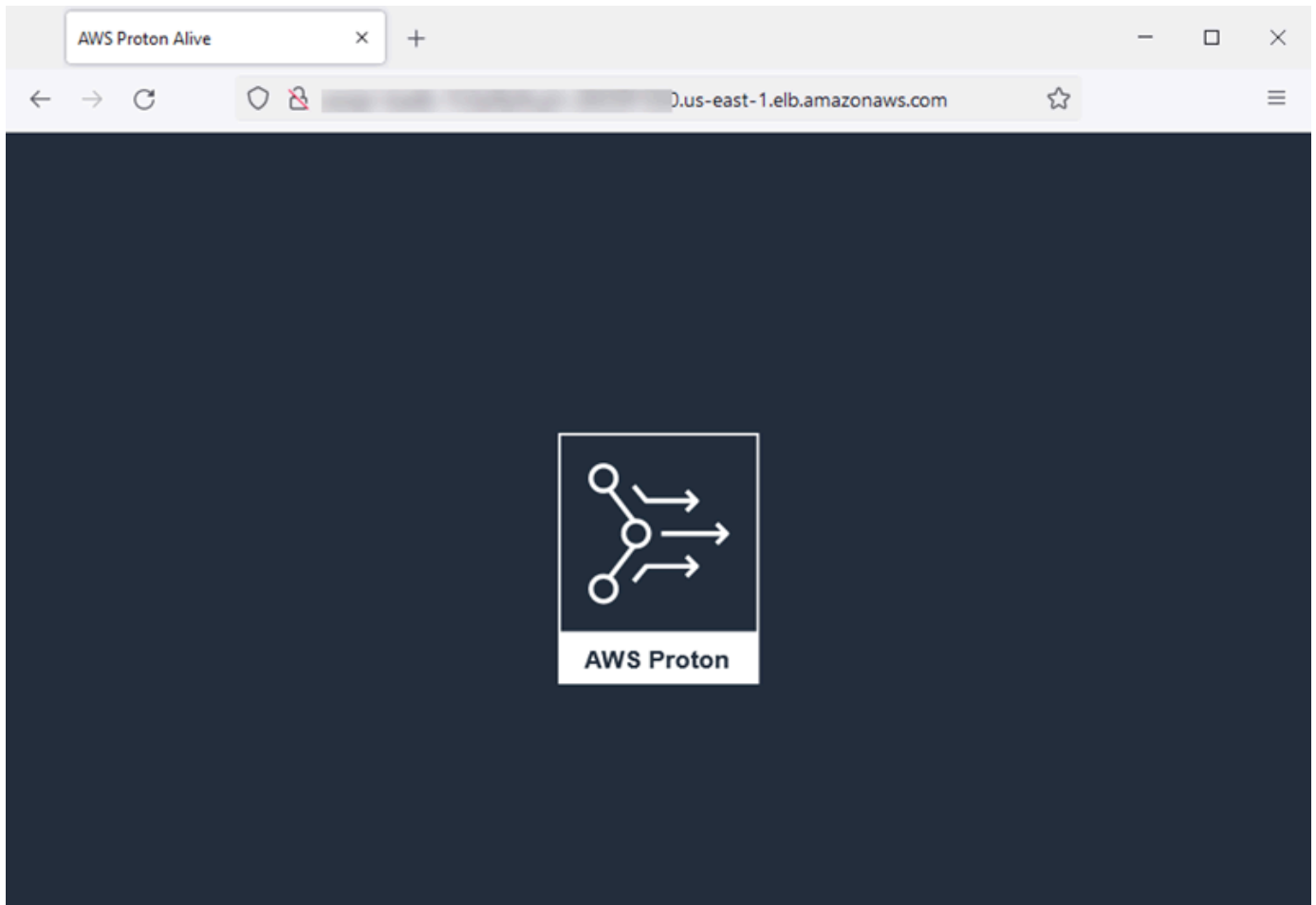
執行以下命令：

```
$ 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。使用您的瀏覽器導覽至瀏覽器。您應該會在靜態頁面上看到下圖：



步驟 5：清除（選用）

在此步驟中，當您完成探索您在本教學課程中建立 AWS 的資源，並為了節省與這些資源相關的成本時，您會將其刪除。

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，請執行下列命令：

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

AWS Proton 範本程式庫

AWS Proton 團隊會在 GitHub 上維護範本範例的程式庫。程式庫包含許多常見環境和應用程式基礎設施案例的基礎設施即程式碼 (IaC) 檔案範例。

範本程式庫存放在兩個 GitHub 儲存庫中：

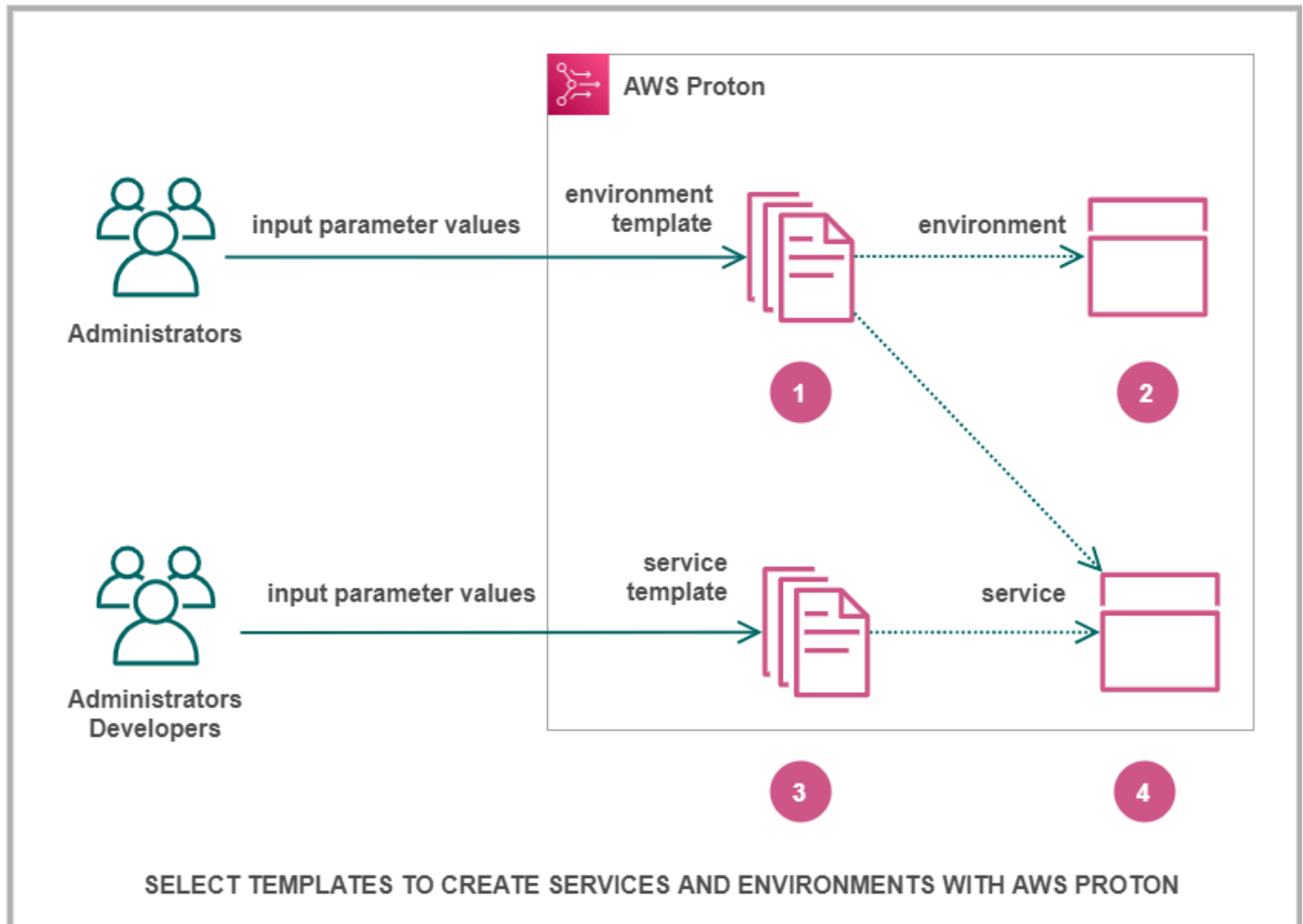
- [aws-proton-cloudformation-sample-templates](#) – AWS CloudFormation 搭配 Jinja 作為其 IaC 語言的範本套件範例。您可以針對 [AWS 受管佈建](#) 環境使用這些範例。
- [aws-proton-terraform-sample-templates](#) – 使用 Terraform 作為其 IaC 語言的範本套件範例。您可以針對 [自我管理佈建](#) 環境使用這些範例。

每個儲存庫都有一個 README 檔案，其中包含儲存庫內容和結構的完整資訊。每個範例都有範本涵蓋的使用案例、範例的架構，以及範本採用的輸入參數的相關資訊。

您可以直接使用此程式庫中的範本，方法是將其中一個程式庫的儲存庫複製到 GitHub 帳戶。或者，使用這些範例做為開發環境和服務範本的起點。

如何 AWS Proton 運作

透過 AWS Proton，您可以佈建環境，然後佈建在這些環境中執行的服務。環境和服務分別是以您在 AWS Proton 版本控制的範本程式庫中選擇的環境和服務範本為基礎。

**1**

當您以管理員身分選取具有的環境範本時 AWS Proton，您會提供必要輸入參數的值。

2

AWS Proton 使用環境範本和參數值來佈建您的環境。

3

身為開發人員或管理員，當您使用選取服務範本 AWS Proton時，您可以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 會使用 服務範本，以及您的服務和選取的環境參數值來佈建您的服務。

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

若要讓此運作，您可以建立環境或服務範本套件，並將其分別上傳至已註冊的環境或服務範本。

[範本套件](#)包含佈建環境或服務 AWS Proton 所需的一切。

建立環境或服務範本時，您會上傳範本套件，其中包含參數化基礎設施做為程式碼 (IaC) 檔案，AWS Proton 用於佈建環境或服務。

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

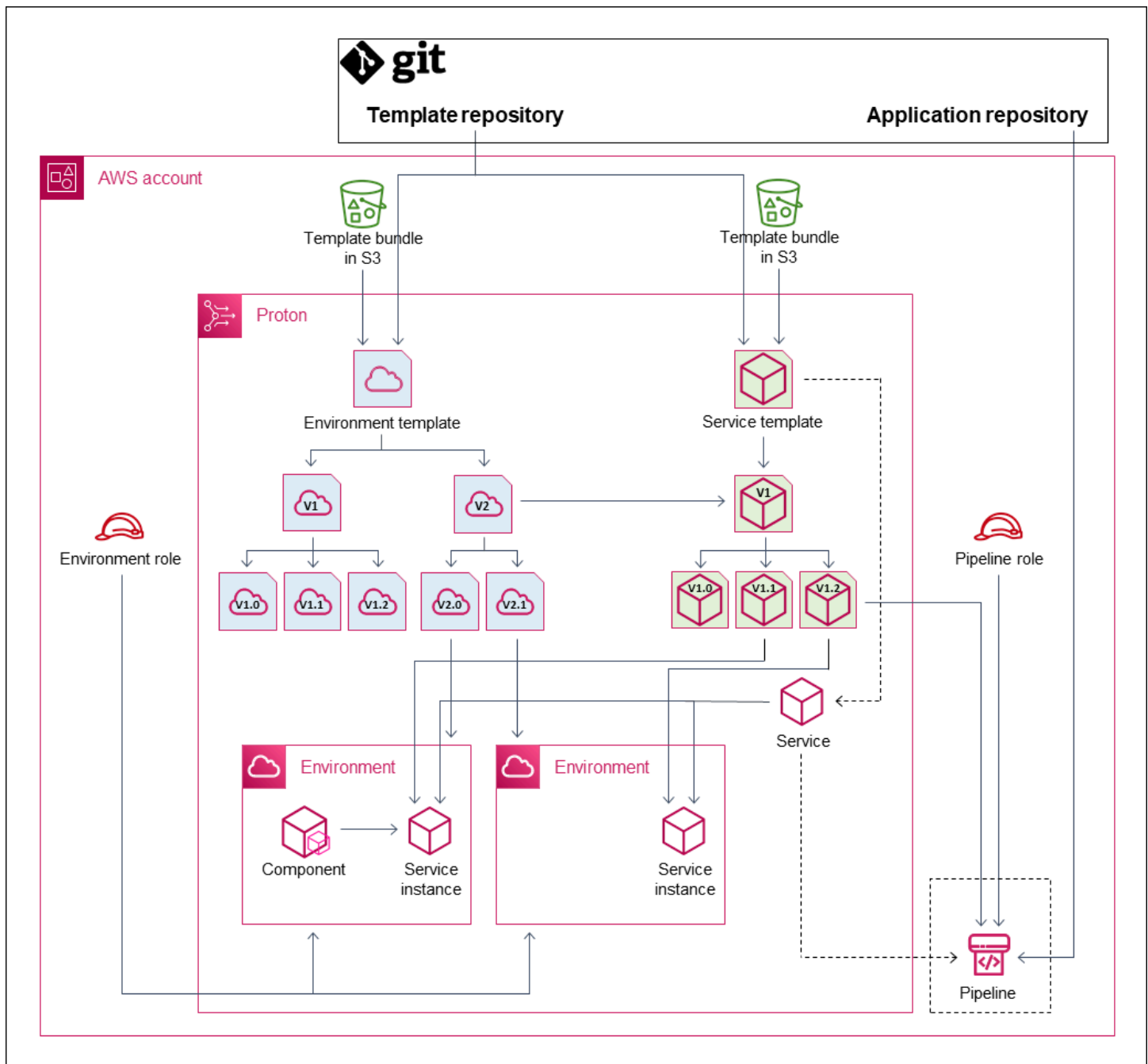
主題

- [AWS Proton 物件](#)
- [如何 AWS Proton 佈建基礎設施](#)
- [AWS Proton 術語](#)

AWS Proton 物件

下圖顯示主要 AWS Proton 物件及其與其他 AWS 和第三方物件的關係。箭頭代表資料流程的方向 (相依性的反向)。

我們遵循圖表，其中包含這些 AWS Proton 物件的簡短描述和參考連結。



- 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[範本撰寫和套件](#)及[範本](#)。

- 環境範本版本 – 環境範本的特定版本。從 S3 儲存貯體或從 Git 儲存庫取得範本套件做為輸入。套件會指定 AWS Proton 環境的基礎設施即程式碼 (IaC) 和相關輸入參數。

如需詳細資訊，請參閱[the section called “版本”](#)、[the section called “發布”](#)及[the section called “範本同步組態”](#)。

- 環境 – AWS Proton 服務部署到的一組共用 AWS 基礎設施資源和存取政策。AWS 資源是透過使用具有特定參數值叫用的環境範本版本來佈建。存取政策是在服務角色中提供。

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[範本撰寫和套件](#)及[範本](#)。

- 服務範本版本 – 服務範本的特定版本。從 S3 儲存貯體或從 Git 儲存庫取得範本套件做為輸入。套件會指定 AWS Proton 服務的基礎設施即程式碼 (IaC) 和相關輸入參數。

服務範本版本也會根據版本指定服務執行個體的這些限制：

- 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 “發布”](#)及[the section called “範本同步組態”](#)。

- 服務 – 使用服務範本中指定的資源執行應用程式的服務執行個體集合，以及選擇性地將應用程式程式碼部署到這些執行個體的 CI/CD 管道。

在圖表中，來自服務範本的虛線表示服務會將範本傳遞至服務執行個體和管道。

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 基礎設施資源。AWS 資源是透過使用具有特定參數值叫用的服務範本版本進行佈建。

如需詳細資訊，請參閱[服務](#)及[the section called “更新執行個體”](#)。

- 管道 – 選用的 CI/CD 管道，可將應用程式部署到服務的執行個體，並具有佈建此管道的存取政策。存取政策是在服務角色中提供。服務不一定有相關聯的 AWS Proton 管道，您可以選擇在外部管理您的應用程式程式碼部署 AWS Proton。

在圖表中，來自 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，可能就無法建立 AWS Proton 管道，而且您自己的管道可能不在 AWS 您的帳戶內。

如需詳細資訊，請參閱[服務](#)及[the section called “更新管道”](#)。

- 元件 – 服務執行個體的開發人員定義延伸。除了環境和服務執行個體提供的資源之外，指定特定應用程式可能需要的其他 AWS 基礎設施資源。平台團隊透過將元件角色連接至環境來控制元件可以佈建的基礎設施。

如需詳細資訊，請參閱[元件](#)。

如何 AWS Proton 佈建基礎設施

AWS Proton 可以透過下列其中一種方式佈建基礎設施：

- **AWS 受管佈建** – 代表您 AWS Proton 呼叫佈建引擎。此方法僅支援 AWS CloudFormation 範本套件。如需詳細資訊，請參閱[the section called “CloudFormation IaC 檔案”](#)。
- **CodeBuild 佈建** – AWS Proton 用來 AWS CodeBuild 執行您提供的 shell 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設施和產生輸出值。此方法的範本套件包含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。

使用 CodeBuild 佈建的範例包括使用 AWS Cloud Development Kit (AWS CDK) 佈建 AWS 資源的程式碼，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 套件”](#)。

Note

您可以搭配環境和服務使用 CodeBuild 佈建。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 發出提取請求 (PR)，其中您自己的基礎設施部署系統會在其中執行佈建程序。此方法僅支援 Terraform 範本套件。如需詳細資訊，請參閱[the section called “Terraform IaC 檔案”](#)。

AWS Proton 會分別決定和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 檢查您提供的範本套件，並決定範本套件指示的佈建方法。在環境層級，您可以提供環境及其潛在服務為其佈建方法所需的參數：AWS Identity and Access Management (IAM) 角色、環境帳戶連線或基礎設施儲存庫。

無論佈建方法為何，使用 AWS Proton 佈建服務的開發人員都有相同的體驗。開發人員不需要知道佈建方法，也不需要服務佈建程序中進行任何變更。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，以提供服務執行個體佈建所需的參數。

下圖摘要說明不同佈建方法的一些主要特徵。表格後面的區段提供每個方法的詳細資訊。

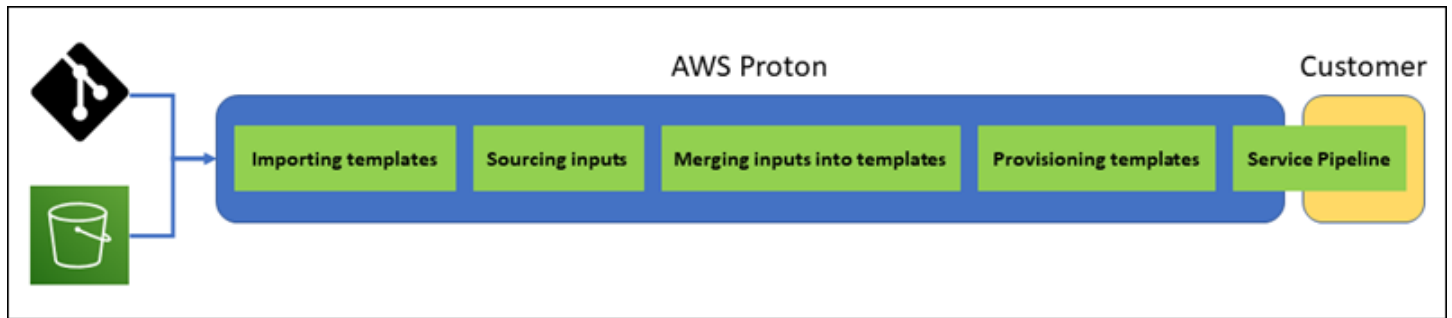
佈建方法	範本	佈建者	追蹤的狀態
AWS受管	資訊清單、結構描述、IaC 檔案 (CloudFormation)	AWS Proton (透過 CloudFormation)	AWS Proton (透過 CloudFormation)
CodeBuild :	manifest (使用 命令)、結構描述、命令相依性 (例如 AWS CDK code)	AWS Proton (透過 CodeBuild)	AWS Proton (您的命令會透過 CodeBuild 傳回狀態)
自我管理	manifest、結構描述、IaC 檔案 (Terraform)	您的程式碼 (透過 Git 動作)	您的程式碼 (AWS 透過 API 呼叫傳遞至)

AWS受管佈建的運作方式

當環境或服務使用 AWS受管佈建時，基礎設施的佈建如下所示：

1. AWS Proton 客戶 (管理員或開發人員) 會建立 AWS Proton 資源 (環境或服務)。客戶會選取資源的範本，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 轉譯用於佈建資源的完整 CloudFormation 範本。
3. AWS Proton 呼叫 CloudFormation 以使用轉譯範本開始佈建。
4. AWS Proton 會持續監控 CloudFormation 部署。
5. 佈建完成時，會在失敗時 AWS Proton 回報錯誤，並在成功時擷取佈建輸出，例如 Amazon VPC ID。

下圖顯示 會直接 AWS Proton 處理大部分的步驟。



AWS受管佈建的考量

- 基礎設施佈建角色 – 當環境或其中執行的任何服務執行個體可能使用 AWS受管佈建時，管理員需要設定 IAM 角色（直接或做為 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設施。角色應該具有使用的許可 CloudFormation，以建立這些資源範本包含的所有資源。

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 AWS受管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設施。開發人員看不到此角色，也無法變更。
- 具有管道的服務 – 使用 AWS受管佈建的服務範本可能包含在 CloudFormation YAML 結構描述中寫入的管道定義。AWS Proton 也會透過呼叫 來建立管道 CloudFormation。AWS Proton 用來建立管道的角色與每個個別環境的角色不同。此角色單獨提供給 AWS Proton，僅在 AWS 帳戶層級提供一次，用於佈建和管理所有 AWS受管管道。此角色應具有建立管道和管道所需其他資源的許可。

下列程序說明如何提供管道角色 AWS Proton。

AWS Proton console

提供管道角色

1. 在[AWS Proton 主控台](#)的導覽窗格中，選擇設定 > 帳戶設定，然後選擇設定。
2. 使用管道 AWS受管角色區段，為 AWS受管佈建設定新的或現有的管道角色。

AWS Proton API

提供管道角色

1. 使用 [UpdateAccountSettings](#) API 動作。
2. 在 pipelineServiceRoleArn 參數中提供管道服務角色的 Amazon Resource Name (ARN)。

AWS CLI

提供管道角色

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

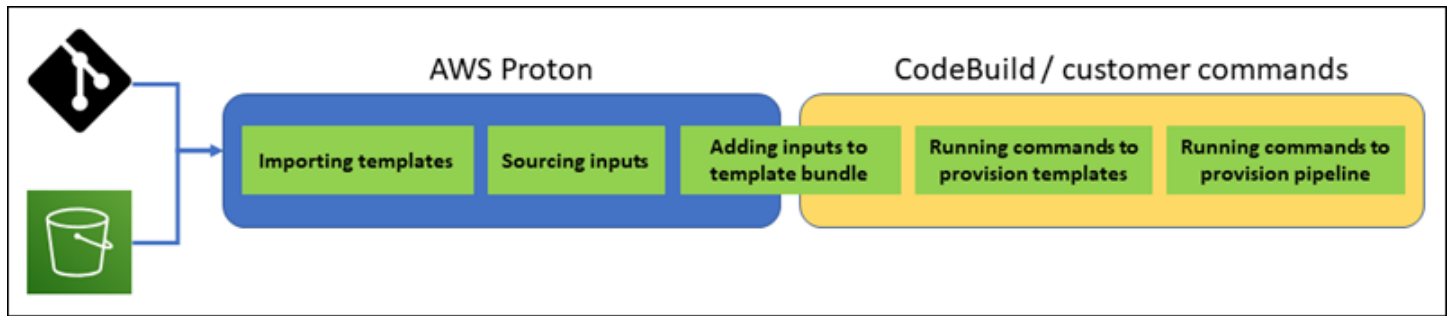
當環境或服務使用 CodeBuild 佈建時，基礎設施的佈建如下所示：

1. AWS Proton 客戶（管理員或開發人員）會建立 AWS Proton 資源（環境或服務）。客戶為資源選取範本，並提供必要的參數。如需詳細資訊，請參閱下一節：[the section called “CodeBuild 佈建的考量事項”](#)。
2. AWS Proton 會使用用於佈建資源的輸入參數值來轉譯輸入檔案。
3. AWS Proton 呼叫 CodeBuild 來啟動任務。CodeBuild 任務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設施，同時選擇性地讀取輸入值。
4. 佈建完成時，最終客戶命令會將佈建狀態傳回 CodeBuild，並呼叫 [NotifyResourceDeploymentStatusChange](#) AWS Proton API 動作，以提供輸出，例如 Amazon VPC ID。

Important

請確定您的命令正確地將佈建狀態傳回 CodeBuild，並提供輸出。如果沒有，AWS Proton 將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 AWS Proton 執行的步驟，以及命令在 CodeBuild 任務中執行的步驟。



CodeBuild 佈建的考量事項

- 基礎設施佈建角色 – 當環境或其中執行的任何服務執行個體可能使用 CodeBuild 型佈建時，管理員需要設定 IAM 角色（直接或做為 AWS Proton 環境帳戶連線的一部分）。AWS Proton 使用此角色來佈建這些 CodeBuild 佈建資源的基礎設施。角色應具有使用 CodeBuild 的許可，以便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 – 當開發人員部署使用 CodeBuild 佈建的服務執行個體到環境時，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設施。開發人員看不到此角色，也無法變更。
- 具有管道的服務 – 使用 CodeBuild 佈建的服務範本可能包含佈建管道的命令。AWS Proton 也會呼叫 CodeBuild 來建立管道。AWS Proton 用來建立管道的角色與每個個別環境的角色不同。此角色單獨提供給 AWS Proton，僅在 AWS 帳戶層級提供一次，用於佈建和管理所有 CodeBuild 型管道。此角色應具有建立管道和管道所需其他資源的許可。

下列程序說明如何提供管道角色 AWS Proton。

AWS Proton console

提供管道角色

1. 在[AWS Proton 主控台](#)的導覽窗格中，選擇設定 > 帳戶設定，然後選擇設定。
2. 使用 Codebuild 管道佈建角色區段，為 CodeBuild 佈建設定新的或現有的管道角色。

AWS Proton API

提供管道角色

1. 使用 [UpdateAccountSettings](#) API 動作。
2. 在 pipelineCodebuildRoleArn 參數中提供管道服務角色的 Amazon Resource Name (ARN)。

AWS CLI

提供管道角色

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，基礎設施的佈建如下所示：

1. AWS Proton 客戶（管理員或開發人員）會建立 AWS Proton 資源（環境或服務）。客戶為資源選取範本，並提供必要的參數。對於環境，客戶也提供連結的基礎設施儲存庫。如需詳細資訊，請參閱下一節：[the section called “自我管理佈建的考量事項”](#)。
2. AWS Proton 轉譯完整的 Terraform 範本。它包含一或多個 Terraform 檔案，可能位於多個資料夾中，以及 .tfvars 變數檔案。會將資源建立呼叫上提供的參數值 AWS Proton 寫入此變數檔案中。
3. AWS Proton 使用轉譯的 Terraform 範本將 PR 提交至基礎設施儲存庫。
4. 當客戶（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，以使用合併的範本開始佈建基礎設施。

Note

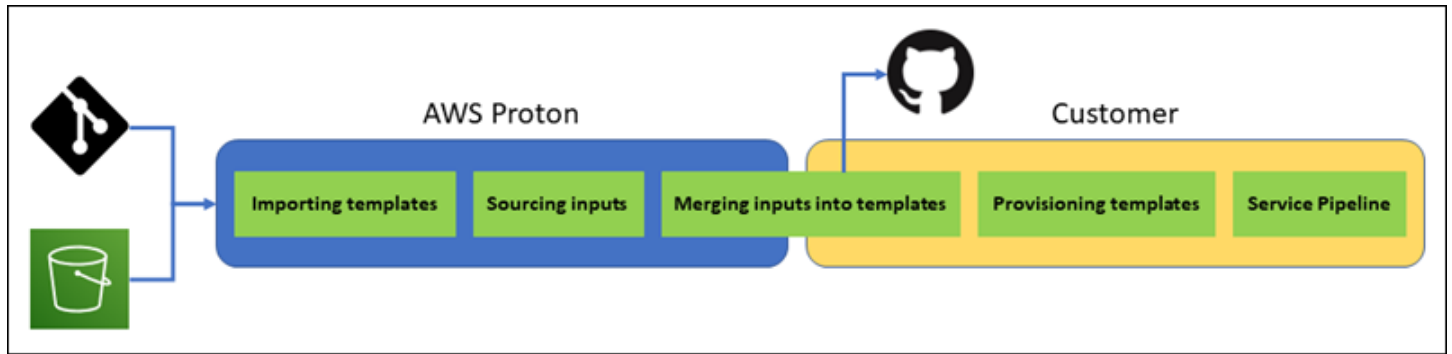
如果客戶（管理員或開發人員）關閉 PR，會將 PR AWS Proton 識別為關閉，並將部署標記為已取消。

5. 佈建完成時，客戶的自動化會呼叫 [NotifyResourceDeploymentStatusChange](#) AWS Proton API 動作，指出完成、提供狀態（成功或失敗），並在存在時提供輸出，例如 Amazon VPC ID。

Important

請確定您的自動化程式碼 AWS Proton 以佈建狀態和輸出回撥。如果沒有，AWS Proton 可能會將佈建視為等待超過預期的時間，並繼續顯示進行中狀態。

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事項

- **基礎設施儲存庫** – 當管理員設定自我管理佈建的環境時，他們需要提供連結的基礎設施儲存庫。會將 PRs AWS Proton 提交至此儲存庫，以佈建環境的基礎設施和部署至該儲存庫的所有服務執行個體。儲存庫中的客戶擁有的自動化動作應擔任具有許可的 IAM 角色，以建立您環境和服務範本包含的所有資源，以及反映目的地 AWS 帳戶的身分。如需擔任角色的 GitHub 動作範例，請參閱「設定 AWS 登入資料」動作 GitHub 動作文件中[的擔任角色](#)。
- **許可** – 您的佈建程式碼必須視需要向 帳戶進行身分驗證（例如，向 AWS 帳戶進行身分驗證），並提供資源佈建授權（例如，提供角色）。
- **服務佈建** – 當開發人員部署使用自我管理佈建的服務執行個體至環境時，會將 PR AWS Proton 提交至與環境相關聯的儲存庫，以佈建服務執行個體的基礎設施。開發人員看不到儲存庫，也無法變更儲存庫。

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，並從中抽象化差異。不過，使用自我管理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（可能不是自己）合併基礎設施儲存庫中的 PR，才能開始佈建。

- **具有管道的服務** – 具有自我管理佈建之環境的服務範本可能包含以 Terraform HCL 撰寫的管道定義（例如，AWS CodePipeline 管道）。若要讓 AWS Proton 佈建這些管道，管理員會提供連結的管道儲存庫給 AWS Proton。佈建管道時，儲存庫中的客戶擁有的自動化動作應擔任具有佈建管道許可的 IAM 角色，以及反映目的地 AWS 帳戶的身分。管道儲存庫和角色與用於每個個別環境的儲存庫和角色不同。連結的儲存庫會 AWS Proton 分別提供給，僅在 AWS 帳戶層級提供一次，用於佈建和管理所有管道。角色應具有建立管道和管道所需其他資源的許可。

下列程序說明如何提供管道儲存庫和角色 AWS Proton。

AWS Proton console

提供管道角色

1. 在 [AWS Proton 主控台](#) 的導覽窗格中，選擇設定 > 帳戶設定，然後選擇設定。
2. 使用 CI/CD 管道儲存庫區段來設定新的或現有的儲存庫連結。

AWS Proton API

提供管道角色

1. 使用 [UpdateAccountSettings](#) API 動作。
2. 在 `pipelineProvisioningRepository` 參數中提供管道儲存庫的提供者、名稱和分支。

AWS CLI

提供管道角色

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 操作所需的組態元素。因此，AWS Proton 無法刪除環境或服務執行個體的所有 Terraform 檔案。反之，會 AWS Proton 標記要刪除的檔案，並更新 PR 中繼資料中的旗標。您的自動化可以讀取該旗標，並使用它來觸發 terraform 銷毀命令。

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設施，例如 VPC 或叢集。

環境範本套件

您上傳以建立和註冊環境範本的檔案集合 AWS Proton。環境範本套件包含下列項目：

1. 將基礎設施定義為程式碼輸入參數的結構描述檔案。
2. 定義共用基礎設施的基礎設施即程式碼 (IaC) 檔案，例如 VPC 或叢集，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

佈建的共用基礎設施，例如 VPC 或叢集，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設施類型。

服務範本套件

您上傳以建立和註冊服務範本的檔案集合 AWS Proton。服務範本套件包含下列項目：

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

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設施。

服務管道

支援管道的佈建基礎設施。

範本版本

範本的主要或次要版本。如需詳細資訊，請參閱[版本化範本](#)。

輸入參數

在結構描述檔案中定義，並在基礎設施中用作程式碼 (IaC) 檔案，以便重複使用 IaC 檔案，並用於各種使用案例。

結構描述檔案

將基礎設施定義為程式碼檔案輸入參數。

規格檔案

將基礎設施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。

清單檔案

將基礎設施列為程式碼檔案。

為編寫範本和建立套件 AWS Proton

AWS Proton 會根據基礎設施即程式碼 (IaC) 檔案為您佈建資源。您可以在可重複使用的 IaC 檔案中描述基礎設施。若要讓檔案可供不同環境和應用程式重複使用，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，將輸入值與範本結合，以建立準備好佈建的 IaC 檔案。

管理員會將大多數範本撰寫為範本套件，然後將它們上傳並註冊到其中 AWS Proton。此頁面的其餘部分討論了這些 AWS Proton 範本套件。直接定義的元件是例外狀況：開發人員會建立元件，並直接提供 IaC 範本檔案。如需元件的詳細資訊，請參閱 [元件](#)。

主題

- [範本套件](#)
- [AWS Proton 參數](#)
- [AWS Proton 基礎設施即程式碼檔案](#)
- [結構描述檔案](#)
- [後續處理的範本檔案 AWS Proton](#)
- [範本套件考量事項](#)

範本套件

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 佈建和部署服務執行個體至選取的環境。如需詳細資訊，請參閱 [AWS Proton 適用於平台團隊](#)。

若要在 中建立和註冊範本 AWS Proton，您可以上傳範本套件，其中包含 AWS Proton 需要佈建 和環境或服務的基礎設施即程式碼 (IaC) 檔案。

範本套件包含下列項目：

- [基礎設施即程式碼 \(IaC\) 檔案](#)，其中包含列出 IaC [檔案的資訊清單 YAML](#) 檔案。IaC
- IaC [檔案輸入參數定義的結構描述 YAML](#) 檔案。

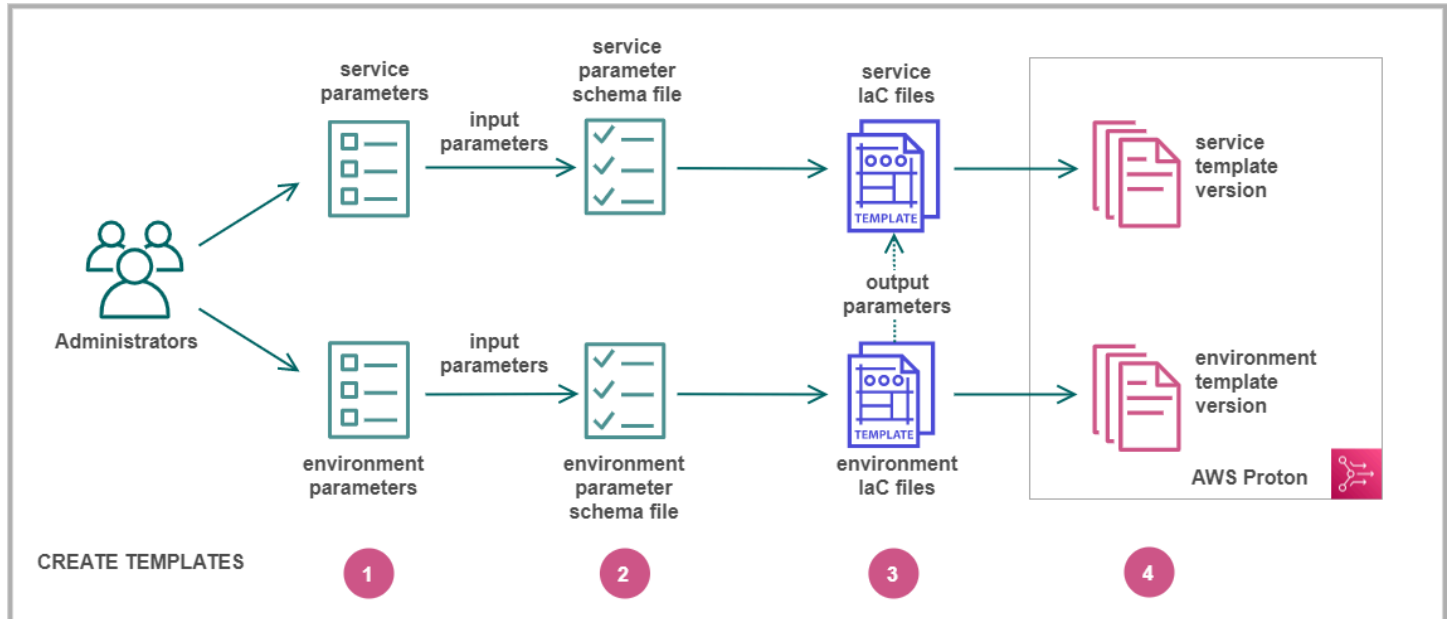
CloudFormation 環境範本套件包含一個 IaC 檔案。

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，以及另一個用於管道定義的選用 IaC 檔案。

Terraform 環境和服務範本套件可以各自包含多個 IaC 檔案。

AWS Proton 需要輸入參數結構描述檔案。當您使用 AWS CloudFormation 建立 IaC 檔案時，您可以使用 [Jinja](#) 語法來參考輸入參數。AWS Proton 提供參數命名空間，可用來參考 IaC 檔案中的 [參數](#)。

下圖顯示您可以用來建立範本的步驟範例 AWS Proton。



1
識別 [輸入參數](#)。

2
建立 [結構描述檔案](#) 以定義您的輸入參數。

3
建立參考輸入參數的 [IaC 檔案](#)。您可以參考環境 IaC 檔案輸出做為服務 IaC 檔案的輸入。

4
向 [註冊範本版本](#)，AWS Proton 並上傳您的範本套件。

AWS Proton 參數

您可以定義並使用基礎設施中的參數做為程式碼 (IaC) 檔案，讓它們具有彈性且可重複使用。您可以參考參數 AWS Proton 命名空間中的參數名稱來讀取 IaC 檔案中的參數值。會將參數值 AWS Proton 插入其在資源佈建期間產生的轉譯 IaC 檔案中。若要處理 AWS CloudFormation IaC 參數，AWS Proton 請使用 [Jinja](#)。若要處理 Terraform IaC 參數，AWS Proton 會產生 Terraform 參數值檔案，並依賴內建於 HCL 的參數化功能。

使用 [CodeBuild 佈建](#)，AWS Proton 產生您的程式碼可以匯入的輸入檔案。檔案是 JSON 或 HCL 檔案，取決於範本資訊清單中的屬性。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。
- 合併的參數命名空間和資源名稱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

您可以在 AWS Proton IaC 檔案中參考下列參數類型：

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的 [結構描述檔案中](#) 定義的輸入參數。您可以在資源的 IaC 檔案中參考資源的輸入參數。元件 IaC 檔案可以參考元件所連接之服務執行個體的輸入參數。

AWS Proton 會對照您的結構描述檔案檢查輸入參數名稱，並將其與 IaC 檔案中參考的參數相符，以在資源佈建期間注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 檔案中定義輸出。輸出可以是範本佈建之其中一個資源的名稱、ID 或 ARN，也可以是傳遞其中一個範本輸入的方式。您可以在其他資源的 IaC 檔案中參考這些輸出。

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

資源參數

AWS Proton 會自動建立 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 檔案中使用 AWS Proton 參數

若要讀取 IaC 檔案中的參數值，請參閱參數命名空間中的 AWS Proton 參數名稱。對於 AWS CloudFormation IaC 檔案，您可以使用 Jinja 語法，並將參數包圍成對大括號和引號。

下表顯示每個支援範本語言的參考語法，其中包含範例。

範本語言	語法	範例：名為 "VPC" 的環境輸入
CloudFormation	"{{ <i>parameter-name</i> }}"	"{{ environment.inputs.VPC }}"
Terraform	var. <i>parameter-name</i>	var.environment.inputs.VPC 產生的 Terraform 變數定義

Note

如果您在 IaC 檔案中使用 [CloudFormation 動態參數](#)，您必須[逸出這些](#)參數，以防止 Jinja 錯誤解譯錯誤。如需詳細資訊，請參閱[故障診斷 AWS Proton](#)

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參數命名空間的不同子集。

範本檔案	參數類型	參數名稱	Description
Environment	資源	environment. name	環境名稱
	input	environment.inputs. <i>input-name</i>	結構描述定義的環境輸入

範本檔案	參數類型	參數名稱	Description
服務	資源	<code>environment.name</code>	環境名稱和 AWS 帳戶 ID
		<code>environment.account_id</code>	
	output	<code>environment.outputs.output-name</code>	環境 IaC 檔案輸出
	資源	<code>service.branch_name</code>	服務名稱和程式碼儲存庫
		<code>service.name</code>	
		<code>service.repository_connection_arn</code>	
		<code>service.repository_id</code>	
	資源	<code>service_instance.name</code>	服務執行個體名稱
input	<code>service_instance.inputs.input-name</code>	結構描述定義的服務執行個體輸入	
資源	<code>service_instance.components.default.name</code>	連接的預設元件名稱	
output	<code>service_instance.components.default.outputs.output-name</code>	連接的預設元件 IaC 檔案輸出	
管道	資源	<code>service_instance.environment.name</code>	服務執行個體環境名稱和 AWS 帳戶 ID
		<code>service_instance.environment.account_id</code>	
	output	<code>service_instance.environment.outputs.output-name</code>	服務執行個體環境 IaC 檔案輸出
	input	<code>pipeline.inputs.input-name</code>	結構描述定義的管道輸入

範本檔案	參數類型	參數名稱	Description
	資源	service.branch_name service.name service.repository_connection_arn service.repository_id	服務名稱和程式碼儲存庫
	input	service_instance.inputs. <i>input-name</i>	結構描述定義的服務執行個體輸入
	收集	{% for service_instance in service_instances %}...{% endfor %}	您可以循環的服務執行個體集合
元件	資源	environment.name environment.account_id	環境名稱和 AWS 帳戶帳戶 ID
	output	environment.outputs. <i>output-name</i>	環境 IaC 檔案輸出
	資源	service.branch_name service.name service.repository_connection_arn service.repository_id	服務名稱和程式碼儲存庫 (連接的元件)
	資源	service_instance. name	服務執行個體名稱 (連接的元件)
	input	service_instance.inputs. <i>input-name</i>	結構描述定義的服務執行個體輸入 (連接的元件)

範本檔案	參數類型	參數名稱	Description
	資源	<code>component.name</code>	元件名稱

如需詳細資訊和範例，請參閱不同資源類型和範本語言的 IaC 範本檔案中參數的相關子主題。

主題

- [環境 CloudFormation IaC 檔案參數詳細資訊和範例](#)
- [Service CloudFormation IaC 檔案參數詳細資訊和範例](#)
- [元件 CloudFormation IaC 檔案參數詳細資訊和範例](#)
- [CloudFormation IaC 檔案的參數篩選條件](#)
- [CodeBuild 佈建參數詳細資訊和範例](#)
- [Terraform 基礎設施即程式碼 \(IaC\) 檔案參數詳細資訊和範例](#)

環境 CloudFormation IaC 檔案參數詳細資訊和範例

您可以在環境基礎設施中將參數定義為程式碼 (IaC) 檔案並加以參考。如需 AWS Proton 參數、參數類型、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 "Parameters"](#)。

定義環境參數

您可以同時定義環境 IaC 檔案的輸入和輸出參數。

- 輸入參數 – 在[結構描述檔案中](#)定義環境輸入參數。

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
- 運作狀態檢查逾時

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 AWS Proton 提供的結構描述型表單。

- 使用 CLI 提供包含值的規格。
- 輸出參數 – 在環境 IaC 檔案中定義環境輸出。然後，您可以在其他資源的 IaC 檔案中參考這些輸出。

讀取環境 IaC 檔案中的參數值

您可以在環境 IaC 檔案中讀取與環境相關的參數。您可以透過在參數命名空間中參考參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 – 參考 讀取環境輸入值 `environment.inputs.input-name`。
- 資源參數 – 透過參考 等名稱讀取 AWS Proton 資源參數 `environment.name`。

Note

環境 IaC 檔案沒有其他資源的輸出參數。

具有參數的環境和服務 IaC 檔案範例

下列範例示範環境 IaC 檔案中的參數定義和參考。此範例會示範如何在服務 IaC 檔案中參考環境 IaC 檔案中定義的環境輸出參數。

Example環境 CloudFormation IaC 檔案

在此範例中，請注意下列事項：

- `environment.inputs` 命名空間是指環境輸入參數。
- Amazon EC2 Systems Manager (SSM) 參數 `StoreInputValue` 會串連環境輸入。
- `MyEnvParameterValue` 輸出會公開與輸出參數相同的輸入參數串連。三個額外的輸出參數也會個別公開輸入參數。
- 六個額外的輸出參數會公開環境佈建的資源。

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 Service CloudFormation IaC 檔案

`environment.outputs`. 命名空間是指來自環境 IaC 檔案的環境輸出。例如，名稱 `environment.outputs.ClusterName` 讀取 `ClusterName` 環境輸出參數的值。

```

AWS::CloudFormation::Template
  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'
[...]
```

Service CloudFormation IaC 檔案參數詳細資訊和範例

您可以在服務和管道基礎設施中將參數定義為程式碼 (IaC) 檔案並加以參考。如需參數、參數類型、參數命名空間，以及如何在 IaC 檔案中使用參數的 AWS Proton 詳細說明，請參閱 [the section called “Parameters”](#)。

定義服務參數

您可以定義服務 IaC 檔案的輸入和輸出參數。

- 輸入參數 – 在[結構描述檔案中](#)定義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。

- 站點
- 任務大小
- 影像
- 所需計數
- Docker 檔案
- 單位測試命令

[您在建立服務](#)時提供輸入參數的值：

- 使用 主控台填寫 AWS Proton 提供的結構描述型表單。
- 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 檔案中參考這些輸出。

讀取服務 IaC 檔案中的參數值

您可以讀取與服務 IaC 檔案中其他資源相關的參數。您可以在參數命名空間中參考參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 – 參考 讀取服務執行個體輸入值 `service_instance.inputs.input-name`。

- 資源參數 – 透過參考 `service.name`、`service_instance.name`和 等名稱讀取 AWS Proton 資源參數`environment.name`。
- 輸出參數 – 透過參考 `environment.outputs.output-name`或 讀取其他資源的輸出`service_instance.components.default.outputs.output-name`。

具有參數的服務 IaC 檔案範例

下列範例是來自服務 CloudFormation IaC 檔案的程式碼片段。 `environment.outputs`。命名空間是指來自環境 IaC 檔案的輸出。 `service_instance.inputs`。命名空間是指服務執行個體輸入參數。 `service_instance.name` 屬性是指 AWS Proton 資源參數。

```
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
```

元件 CloudFormation IaC 檔案參數詳細資訊和範例

您可以在元件基礎設施中將參數定義為程式碼 (IaC) 檔案並加以參考。如需 AWS Proton 參數、參數類型、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 "Parameters"](#)。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 檔案中參考這些輸出。

Note

您無法定義元件 IaC 檔案的輸入。連接的元件可以從其連接的服務執行個體取得輸入。分離的元件沒有輸入。

讀取元件 IaC 檔案中的參數值

您可以讀取與元件和元件 IaC 檔案中其他資源相關的參數。您可以在參數命名空間中參考參數的名稱來讀取 AWS Proton 參數值。

- 輸入參數 – 透過參考 讀取附加的服務執行個體輸入值 `service_instance.inputs.input-name`。
- 資源參數 – 透過參考 `component.name`、`service_instance.name`、`service.name` 和 等名稱讀取 AWS Proton 資源參數 `environment.name`。
- 輸出參數 – 參考 讀取環境輸出 `environment.outputs.output-name`。

具有參數的元件和服務 IaC 檔案範例

下列範例顯示佈建 Amazon Simple Storage Service (Amazon S3) 儲存貯體和相關存取政策的元件，並將這兩個資源的 Amazon Resource Name (ARNs) 公開為元件輸出。服務 IaC 範本會將元件輸出新增為 Amazon Elastic Container Service (Amazon ECS) 任務的容器環境變數，讓輸出可供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。儲存貯體名稱是以環境、服務、服務執行個體和元件的名稱為基礎，這表示儲存貯體會與延伸特定服務執行個體之元件範本的特定執行個體結合。

開發人員可以根據此元件範本建立多個自訂元件，為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 `{{ ... }}` 語法來參考服務 IaC 檔案中的元件和其他資源參數。您只能在元件連接至服務執行個體時，使用 `{% if ... %}` 陳述式來新增陳述式區塊。`proton_cfn_*` 關鍵字是可用來淨化和格式化輸出參數值的篩選條件。如需篩選條件的詳細資訊，請參閱[the section called “CloudFormation 參數篩選條件”](#)。

身為管理員，您會撰寫服務 IaC 範本檔案。

Example 使用元件的服務 CloudFormation IaC 檔案

```
# 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 元件 CloudFormation IaC 檔案

```
# 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
```

當 AWS Proton 轉譯服務執行個體的 CloudFormation 範本，並將所有參數取代為實際值時，範本可能看起來像下列檔案。

Example 服務執行個體 CloudFormation 轉譯 IaC 檔案

```
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 檔案的參數篩選條件

當您參考 AWS CloudFormation IaC 檔案中的 [AWS Proton 參數](#) 時，您可以使用稱為篩選條件的 Jinja 修飾詞，在參數值插入轉譯範本之前對其進行驗證、篩選和格式化。參考 [元件輸出參數](#) 時，篩選條件驗證特別有用，因為元件建立和連接是由開發人員完成，而且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和有效性。不過，您可以在任何 Jinja IaC 檔案中使用篩選條件。

下列各節說明和定義可用的參數篩選條件，並提供範例。AWS Proton 定義這些篩選條件的大部分。default 篩選條件是 Jinja 內建篩選條件。

為 Amazon ECS 任務格式化環境屬性

宣告

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

Description

此篩選條件會在 Amazon Elastic Container Service (Amazon ECS) 任務定義的 ContainerDefinition 區段中，格式化要在 [環境屬性](#) 中使用的輸出清單。

raw 設定為 False 以驗證參數值。在此情況下，需要值才能符合規則表達式 `^[a-zA-Z0-9_-]*$`。如果值未通過此驗證，範本轉譯會失敗。

範例

使用下列自訂元件範本：

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

以及下列服務範本：

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

轉譯的服務範本如下所示：

```
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

宣告

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

Description

此篩選條件會格式化要在 AWS Lambda 函數定義的 Properties 區段中的 [環境屬性](#) 中使用的輸出清單。

raw 設定為 False 以驗證參數值。在此情況下，需要值才能符合規則表達式 `^[a-zA-Z0-9_-]*$`。如果值未通過此驗證，範本轉譯會失敗。

範例

使用下列自訂元件範本：

```

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

```

以及下列服務範本：

```

Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:

```

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

轉譯的服務範本如下所示：

```
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 ARNs

宣告

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

Description

此篩選條件會格式化 AWS Identity and Access Management (IAM) 角色定義 `Properties` 區段中要在 [ManagedPolicyArns 屬性](#) 中使用的輸出清單。篩選條件使用規則表達 `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` 式，從輸出參數清單中擷取有效的 IAM 政策 ARNs。您可以使用此篩選條件，將輸出參數值中的政策附加至服務範本中的 IAM 角色定義。

範例

使用下列自訂元件範本：

```
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

```

以及下列服務範本：

```

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:
    # ...

```

轉譯的服務範本如下所示：

```

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

宣告

```
string # proton_cfn_sanitize # string
```

Description

這是一般用途篩選條件。使用它來驗證參數值的安全性。篩選條件會驗證該值是否符合規則表達式，`^[a-zA-Z0-9_-]*$`或是有效的 Amazon Resource Name (ARN)。如果值未通過此驗證，範本轉譯會失敗。

範例

使用下列自訂元件範本：

```
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：

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

轉譯方式如下：

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

- 服務範本中的下列參考：

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

具有下列轉譯錯誤的結果：

```
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：

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

轉譯方式如下：

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

提供不存在參考的預設值

Description

當命名空間參考不存在時，`default`篩選條件會提供預設值。使用它來撰寫強大的範本，即使在您參考的參數遺失時，也能在沒有失敗的情況下轉譯。

範例

如果服務執行個體沒有直接定義的（預設）元件，或者連接的元件沒有名為的輸出，則服務範本中的下列參考會導致範本轉譯失敗`test`。

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

若要避免此問題，請新增`default`篩選條件。

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

CodeBuild 佈建參數詳細資訊和範例

您可以在 CodeBuild 型 AWS Proton 資源的範本中定義參數，並在佈建程式碼中參考這些參數。如需 AWS Proton 參數、參數類型、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 “Parameters”](#)。

Note

您可以搭配環境和服務使用 CodeBuild 佈建。目前您無法以這種方式佈建元件。

輸入參數

當您建立 AWS Proton 資源時，例如環境或服務，您可以為範本 [結構描述檔案](#) 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 CodeBuild 範本的範例，請參閱 [the section called “CodeBuild 套件”](#)。如需資訊清單檔案的相關資訊，請參閱 [the section called “資訊清單和後續處理”](#)。

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

範例：AWS CDK 搭配 CodeBuild 佈建使用

```
{
```

```
"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給 AWS Proton，您的佈建程式碼可以產生名為 `proton-outputs.json` 的 JSON 檔案，其中包含範本 [結構描述檔案](#) 中定義的輸出參數值。例如，`cdk deploy` 命令具有 `--outputs-file` 引數，指示使用佈建輸出 AWS CDK 產生 JSON 檔案。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 會尋找此 JSON 檔案。如果檔案在您的佈建程式碼成功完成之後存在，會從中 AWS Proton 讀取輸出參數值。

Terraform 基礎設施即程式碼 (IaC) 檔案參數詳細資訊和範例

您可以在範本套件的 `variable.tf` 檔案中包含 Terraform 輸入變數。您也可以建立結構描述來建立 AWS Proton 受管變數。`.tf` files 從結構描述檔案 AWS Proton 建立變數。如需詳細資訊，請參閱 [the section called “Terraform IaC 檔案”](#)。

若要在基礎設施 中參考結構描述定義的 AWS Proton 變數 `.tf` files，您可以使用 Terraform IaC 資料表的參數和命名 AWS Proton 空間中顯示的命名空間。IaC 例如，您可以使用 `var.environment.inputs.vpc_cidr`。在引號內，以單一括號括住這些變數，並在第一個括號前面加上貨幣符號（例如 `"${var.environment.inputs.vpc_cidr}"`）。

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

```
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 基礎設施即程式碼檔案

範本套件的主要部分是基礎設施即程式碼 (IaC) 檔案，這些檔案會定義您要佈建的基礎設施資源和屬性。AWS CloudFormation 其他基礎設施即程式碼引擎則使用這些類型的檔案來佈建基礎設施資源。

Note

laC 檔案也可以獨立於範本套件使用，做為直接定義元件的直接輸入。如需元件的詳細資訊，請參閱 [元件](#)。

AWS Proton 目前支援兩種 laC 檔案類型：

- [CloudFormation](#) 檔案 – 用於 AWS 受管佈建。在 CloudFormation 範本檔案格式的頂端 AWS Proton 使用 Jinja 進行參數化。
- [Terraform HCL](#) 檔案 – 用於自我管理佈建。HCL 原生支援參數化。

您無法使用佈建方法的組合來佈建 AWS Proton 資源。您必須使用其中一個。您無法將 AWS 受管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](#)。

CloudFormation laC 檔案

了解如何搭配使用 AWS CloudFormation 基礎設施做為程式碼檔案 AWS Proton。CloudFormation 是基礎設施做為程式碼 (laC) 服務，可協助您建立和設定 AWS 資源的模型。您可以在範本中使用 Jinja 在 parametrization. AWS Proton expands 參數的 CloudFormation 範本檔案格式上定義基礎設施資源，並轉譯完整的 CloudFormation 範本。CloudFormation 會將定義的資源佈建為 CloudFormation 堆疊。如需詳細資訊，請參閱《使用者指南》CloudFormation 中的 [什麼是 CloudFormation](#)。

AWS Proton 支援 CloudFormation laC [AWS 的受管佈建](#)。

從您現有的基礎設施開始，做為程式碼檔案

您可以調整自己的現有基礎設施做為程式碼 (laC) 檔案，以搭配使用 AWS Proton。

下列 CloudFormation 範例 [範例 1](#) 和 [範例 2](#) 代表您現有的 CloudFormation laC 檔案。CloudFormation 可以使用這些檔案來建立兩個不同的 CloudFormation 堆疊。

在 [範例 1](#) 中，CloudFormation laC 檔案設定為佈建要在容器應用程式之間共用的基礎設施。在此範例中，會新增輸入參數，以便您可以使用相同的 laC 檔案來建立多組佈建基礎設施。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。身為管理員或開發人員，當您使用 laC 檔案搭配 CloudFormation 佈建基礎設施資源時，您可以提供這些參數的值。為了方便起見，這些輸入參數會以註解標記，並在範例中多次參考。輸出會在範本結尾處定義。它們可以在其他 CloudFormation laC 檔案中參考。

在範例 2 中，CloudFormation IaC 檔案設定為將應用程式部署至從範例 1 佈建的基礎設施。為了您的方便，參數會加上註解。

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::VPCGatewayAttachement
  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搭配 使用 AWS Proton。

將您的基礎設施做為程式碼帶到 AWS Proton

只要稍微修改，您就可以使用[範例 1](#) 做為環境範本套件的基礎設施做為程式碼 (IaC) 檔案，該套件 AWS Proton 使用 來部署環境（如[範例 3](#) 所示）。

您可以使用 [Jinja](#) 語法來參考您在 [Open API 型結構描述檔案中](#) 定義的參數，而不是使用 CloudFormation 參數。為了方便起見，系統會對這些輸入參數進行註解，並在 IaC 檔案中多次參考。如此一來，AWS Proton 就可以稽核和檢查參數值。它也可以將一個 IaC 檔案中的輸出參數值比對並插入另一個 IaC 檔案中的參數。

身為管理員，您可以將命名空間新增至 AWS Proton `environment.inputs` 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，用大括號和引號括住它們。

透過這些修改，您的 CloudFormation IaC 檔案可供使用 AWS Proton。

範例 3：AWS Proton 環境基礎設施做為程式碼檔案

```
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 檔案會產生略有不同的 CloudFormation 堆疊。[???](#) 參數會以不同的方式顯示在堆疊範本檔案中。[範例 1](#) CloudFormation 堆疊範本檔案會在堆疊範本檢視中顯示參數標籤（索引鍵）。[範例 3](#) AWS Proton CloudFormation 基礎設施堆疊範本檔案會顯示參數值。AWS Proton 輸入參數不會顯示在主控台 CloudFormation 堆疊參數檢視中。

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 管道 IaC 檔案佈建管道基礎設施，以支援[範例 4](#) 佈建的服務執行個體。

範例 5：AWS Proton 服務管道 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

使用 CodeBuild 佈建，而不是使用 IaC 範本轉譯 IaC 檔案並使用 IaC 佈建引擎執行它們，AWS Proton 只需執行您的 shell 命令。若要這樣做，會在環境帳戶中為環境 AWS Proton 建立 AWS

CodeBuild 專案，並啟動任務，以針對每個 AWS Proton 資源建立或更新執行您的命令。當您撰寫範本套件時，您會提供資訊清單，指定基礎設施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設施和產生輸出值。

資訊清單也會指定 AWS Proton 應如何轉譯您的程式碼可輸入並從中取得輸入值的輸入檔案。它可以轉譯為 JSON 或 HCL。如需有關輸入參數的詳細資訊，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 “資訊清單和後續處理”](#)。

Note

您可以搭配環境和服務使用 CodeBuild 佈建。目前您無法以這種方式佈建元件。

範例：AWS CDK 搭配 CodeBuild 佈建使用

使用 CodeBuild 佈建的範例包括使用 AWS Cloud Development Kit (AWS CDK) 佈建 (部署) 和取消佈建 (destroy) AWS 資源的程式碼，以及安裝 CDK 並執行 CDK 程式碼的資訊清單。

下列各節列出您可以包含在 CodeBuild 佈建範本套件中的範例檔案，該套件使用 佈建環境 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 AWS CDK、輸出檔案處理和回報輸出所需的命令 AWS Proton。

Example infrastructure/manifest.yaml

```
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 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 schema/schema.yaml

```

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 檔案

下列檔案是 Node.js CDK 專案的範例。

Example infrastructure/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironment": "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 infrastructure/tsconfig.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 infrastructure/cdk.json

```
{
  "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 infrastructure/bin/ProtonEnvironment.ts

```

#!/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 infrastructure/lib/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`
});
}
```

轉譯的輸入檔案

當您使用 CodeBuild 型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。下列檔案是轉譯輸入檔案的範例。

Example infrastructure/proton-inputs.json

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

Terraform IaC 檔案

了解如何搭配使用 Terraform 基礎設施做為程式碼 (IaC) 檔案 AWS Proton。 [Terraform](#) 是由 [HashiCorp](#) 開發的廣泛使用開放原始碼 IaC 引擎。Terraform 模組是以 HashiCorp 的 HCL 語言開發，並支援數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](#)。

如需回應提取請求並實作基礎設施佈建的佈建儲存庫完整範例，請參閱 [GitHub 上適用於的 Terraform OpenSource GitHub Actions 自動化範本 AWS Proton](#)。GitHub

自我管理佈建如何與 Terraform IaC 範本套件檔案搭配使用：

1. 當您從 Terraform 範本套件 [建立環境](#) 時，會使用主控台或 spec file 輸入參數 AWS Proton 編譯您的 .tf 檔案。
2. 它提出提取請求，將編譯的 IaC 檔案合併到 [您已向註冊的儲存庫 AWS Proton](#)。

3. 如果請求獲得核准，會 AWS Proton 等待您提供的佈建狀態。
4. 如果請求遭到拒絕，則會取消環境建立。
5. 如果提取請求逾時，則環境建立未完成。

AWS Proton 搭配 Terraform IaC 考量：

- AWS Proton 不會管理您的 Terraform 佈建。
- 您必須在此[儲存庫上向 .makes 提取請求註冊佈建](#)儲存庫。AWS Proton AWS Proton
- 您必須[建立 CodeStar 連線](#)，才能 AWS Proton 與您的佈建儲存庫連線。
- 若要從 AWS Proton 編譯的 IaC 檔案佈建，您必須回應 AWS Proton pull request. AWS Proton makes 在環境和服務建立和更新動作之後提取請求。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 檔案佈建管道，您必須[建立 CI/CD 管道儲存庫](#)。
- 您的提取請求型佈建自動化必須包含通知 AWS Proton 任何佈建 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 CloudFormation IaC 檔案建立的服務、管道和元件部署至從 Terraform IaC 檔案建立的環境。
- 您無法將從 Terraform IaC 檔案建立的服務、管道和元件部署至從 CloudFormation IaC 檔案建立的環境。

為準備 Terraform IaC 檔案時 AWS Proton，您可以將命名空間連接至輸入變數，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 Terraform 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
}

```

編譯的基礎設施做為程式碼

當您建立環境或服務時，會使用主控台或spec file輸入將基礎設施 AWS Proton 編譯為程式碼檔案。它會為您的輸入建立 `proton.resource-type.variables.tf` 和 `proton.auto.tfvars.json` 檔案，供 Terraform 使用，如下列範例所示。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指定的儲存庫中。

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 標籤傳播到佈建的資源。如需詳細資訊，請參閱[the section called “將標籤傳播至佈建的資源”](#)。

範例 2：為名為 "dev" 的環境編譯 IaC 檔案。

dev/environment.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
}
```

dev/proton.environment.variables.tf :

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

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

dev/proton.auto.tfvars.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建立動作的主控制台或規格輸入來尋找儲存庫，以及尋找編譯 IaC 檔案的路徑。輸入值會傳遞至[命名空間輸入參數](#)。

AWS Proton 支援兩種儲存庫路徑配置。在下列範例中，路徑會以來自兩個環境的命名空間資源參數命名。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 服務執行個體具有直接定義的元件。

Resource Type (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"

Resource Type (資源類型)	名稱參數	=	資源名稱
服務執行個體	service_instance.name		"instance -two- prod"
元件	service_instance.components.default.name		"componen t-prod"
服務執行個體	service_instance.name		"instance -two- staged"
元件	service_instance.components.default.name		"componen t- staged"

Layout 1

如果 AWS Proton 找到具有 environments 資料夾的指定儲存庫，它會建立包含編譯的 IaC 檔案並以命名的資料夾environment.name。

如果 AWS Proton 找到具有資料夾environments的指定儲存庫，其中包含與服務執行個體相容環境名稱相符的資料夾名稱，則會建立資料夾，其中包含編譯的執行個體 IaC 檔案，並以命名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` 資料夾的指定儲存庫，它會建立資料夾 `environment.name`，找到編譯的環境 IaC 檔案。

如果 AWS Proton 找到具有與服務執行個體相容環境名稱相符之資料夾名稱的指定儲存庫，它會建立 `service_instance.name` 資料夾來尋找編譯的執行個體 IaC 檔案。

```

/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
```

結構描述檔案

身為管理員，當您使用 Open API [Data Models \(結構描述\)](#) 區段來定義範本套件的參數結構描述 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。

如需格式和可用關鍵字詳細資訊，請參閱 OpenAPI 的 [結構描述物件](#) 區段。

環境範本套件的結構描述需求

您的結構描述必須遵循 YAML 格式 OpenAPI [的資料模型（結構描述）區段](#)。它也必須是環境範本套件的一部分。

對於您的環境結構描述，您必須包含格式化標頭，以確定您使用的是 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。

`environment_input_type` 必須包含並定義您提供的名稱。在下列範例中，這在第 5 行定義。透過定義此參數，您可以將其與 AWS Proton 環境資源建立關聯。

若要遵循 Open API 結構描述模型，您必須包含 `types`。在下列範例中，這是第 6 行。

在之後 `types`，您必須定義 `environment_input_type` 類型。您可以將環境的輸入參數定義為的屬性 `environment_input_type`。您必須包含至少一個名稱與環境基礎設施中列出的至少一個參數相符的屬性，做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、驗證和插入至相關聯 CloudFormation IaC 檔案中的大括號參數。針對每個屬性（參數），提供 `name` 和 `type`。或者，您也可以提供 `description`、`default` 和 `pattern`。

下列範例標準環境範本結構描述的定義參數包括 `vpc_cidr`、`subnet_one_cidr`、`subnet_two_cidr` 以及 `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自己的值。如果參數沒有預設值並列為 `required` 屬性（參數），您必須在建立環境時為其提供值。

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範本建立結構描述。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，使用描述和輸入屬性定義環境輸入類型。此結構描述範例可與 [範例 3](#) 中顯示的 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

```

- 客戶受管環境範本

在下列範例中，結構描述只會包含輸出清單，這些輸出會複寫您用來佈建客戶受管基礎設施的 IaC 輸出。您只需要將輸出值類型定義為字串 (而非清單、陣列或其他類型)。例如，下一個程式碼片段會顯示外部 CloudFormation 範本的輸出區段。這是來自[範例 1](#) 中顯示的範本。它可用於為從[範例 4](#) 建立的 AWS Proton Fargate 服務建立外部客戶受管基礎設施。

Important

身為管理員，您必須確保佈建和管理的基礎設施和所有輸出參數都與相關聯的客戶受管環境範本相容。AWS Proton 無法代表您考慮變更，因為這些變更不可見 AWS Proton。不一致會導致失敗。

客戶受管環境範本的範例 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'
[...]

```

對應 AWS Proton 客戶受管環境範本套件的結構描述如下列範例所示。每個輸出值都定義為字串。

客戶受管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"
```

[...]

服務範本套件的結構描述需求

您的結構描述必須遵循 YAML 格式 OpenAPI [的資料模型（結構描述）區段](#)，如下列範例所示。您必須在服務範本套件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。在下列範例中，這位於前三行。這是為了確認您使用的是 Open API 的資料模型（結構描述）區段。

`service_input_type` 必須包含並定義您提供的名稱。在下列範例中，這是第 5 行。這會將參數與服務 AWS Proton 資源建立關聯。

當您使用主控台或 CLI 建立 AWS Proton 服務時，預設會包含服務管道。當您包含服務的服務管道時，您必須 `pipeline_input_type` 包含您提供的名稱。在下列範例中，這是第 7 行。如果您未包含 AWS Proton 服務管道，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，您必須包含 `types` 下列範例中的第 9 行。

在之後 `types`，您必須定義 `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屬性，其名稱至少符合服務基礎設施中列出的至少一個參數做為與結構描述相關聯的程式碼 (IaC) 檔案。

若要定義服務管道，請在 `service_input_type` 定義下方定義 `pipeline_input_type`。如上所述，您必須包含至少一個屬性，其名稱至少符合與結構描述相關聯的管道 IaC 檔案中列出的至少一個參數。如果您未包含 AWS Proton 服務管道，請勿包含此定義。

身為管理員或開發人員，當您建立服務並提供自訂參數值時，AWS Proton 會使用結構描述檔案來比對、驗證和插入相關聯的 CloudFormation IaC 檔案的大括號參數。針對每個屬性（參數），提供 `name` 和 `type`。或者，您也可以提供 `description`、`default` 和 `pattern`。

範例結構描述的定義參數包括 `port`、`task_size` 和 `desired_count`，`image` 以及 `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自己的值。參數 `unique_name` 也包含在範例中，而且沒有預設值。它被列為 `required` 屬性（參數）。身為管理員或開發人員，您必須在建立服務時提供 `required` 參數的值。

如果您想要使用服務管道建立服務範本，請在結構描述 `pipeline_input_type` 中包含。

包含服務 AWS Proton 管道之服務的結構描述檔案範例。

此結構描述範例可與 [範例 4](#) 和 [範例 5](#) 中顯示的 AWS Proton IaC 檔案搭配使用。包含服務管道。

```

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

```

如果您想要在沒有服務管道的情況下建立服務範本，請不要在結構描述 `pipeline_input_type` 中包含，如下列範例所示。

不包含服務 AWS Proton 管道之服務的服務結構描述檔案範例

```

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"

```

後續處理 的範本檔案 AWS Proton

準備環境和服務基礎設施做為程式碼 (IaC) 檔案及其個別結構描述檔案之後，您必須在目錄中組織它們。您還必須建立資訊清單 YAML 檔案。資訊清單檔案會列出目錄中的 IaC 檔案、轉譯引擎，以及用於在此範本中開發 IaC 的範本語言。

Note

資訊清單檔案也可以獨立於範本套件使用，做為直接定義元件的直接輸入。在此情況下，它會一律為 CloudFormation 和 Terraform 指定單一 IaC 範本檔案。如需元件的詳細資訊，請參閱 [元件](#)。

資訊清單檔案需要遵循下列範例中顯示的格式和內容。


CloudFormation 資訊清單檔案格式：

使用 CloudFormation，您可以列出單一檔案。

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

Terraform 資訊清單檔案格式：

使用 terraform，您可以明確列出單一檔案，或使用萬用字元*列出目錄中的每個檔案。


 Note

萬用字元只包含名稱結尾為 `.tf` 的檔案。會忽略其他檔案。

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

CodeBuild 型佈建資訊清單檔案格式：

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

 Note

除了資訊清單之外，您的套件還應該包含命令所依賴的任何檔案。

下列範例資訊清單使用 CodeBuild 型佈建來使用 `proton build` 佈建 AWS Cloud Development Kit (AWS CDK) (部署) 和取消佈建 (還原) 資源 AWS CDK。範本套件也應包含 CDK 程式碼。

在佈建期間，AWS Proton 會建立輸入檔案，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 }}"

```

為您的環境或服務範本套件設定目錄和資訊清單檔案後，您可以將目錄壓縮為 tar 球，並將其上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體，其中 AWS Proton 可以擷取它們，或[範本同步步 Git 儲存庫](#)。

當您建立環境的次要版本或註冊的服務範本時 AWS Proton，您會提供位於 S3 儲存貯體中環境或服務範本套件 tar 球的路徑。AWS Proton 會使用新的範本次要版本儲存它。您可以選擇新的範本次要版本來建立或更新環境或服務 AWS Proton。

環境範本套件包裝

您建立的環境範本套件有兩種類型 AWS Proton。

- 若要為標準環境範本建立環境範本套件，請在目錄中組織結構描述、基礎設施即程式碼 (IaC) 檔案和資訊清單檔案，如下列環境範本套件目錄結構所示。
- 若要為客戶受管環境範本建立環境範本套件，請僅提供結構描述檔案和目錄。請勿包含基礎設施目錄和檔案。如果包含基礎設施目錄和檔案，會 AWS Proton 擲回錯誤。

如需詳細資訊，請參閱[註冊和發佈範本](#)。

CloudFormation 環境範本套件目錄結構：

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

Terraform 環境範本套件目錄結構：

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

服務範本套件後續處理

若要建立服務範本套件，您必須將結構描述、基礎設施即程式碼 (IaC) 檔案和資訊清單檔案組織到目錄中，如服務範本套件目錄結構範例所示。

如果您未在範本套件中包含服務管道，請勿在建立要與此範本套件相關聯的服務範本"pipelineProvisioning": "CUSTOMER_MANAGED"時包含管道目錄和檔案並設定。

Note

建立服務範本pipelineProvisioning後，您無法修改。

如需詳細資訊，請參閱[註冊和發佈範本](#)。

CloudFormation 服務範本套件目錄結構：

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

```
cloudformation.yaml
```

Terraform 服務範本套件目錄結構：

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

範本套件考量事項

- 基礎設施即程式碼 (IaC) 檔案

AWS Proton 稽核範本的檔案格式是否正確。不過，AWS Proton 不會檢查範本開發、相依性和邏輯錯誤。例如，假設您已指定在 CloudFormation IaC 檔案中建立 Amazon S3 儲存貯體，做為服務或環境範本的一部分。服務會根據這些範本建立。現在，假設您想要在某個時間點刪除服務。如果指定的 S3 儲存貯體不是空的，且 CloudFormation IaC 檔案未在 Retain 中將其標記為 DeletionPolicy，則服務刪除操作會 AWS Proton 失敗。

- 套件檔案大小限制和格式

- 套件檔案大小、計數和名稱大小限制可在 [找到 AWS Proton 配額](#)。

- 檔案的範本套件目錄會壓縮為 tar 球，並位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。

- 套件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套件中的敏感資料，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

若要將範本套件新增至 AWS Proton 範本程式庫，請建立範本次要版本並將其註冊 AWS Proton。建立範本時，請提供範本套件的 Amazon S3 儲存貯體名稱和路徑。發佈範本之後，平台團隊成員和開發人員即可選取範本。選取之後，AWS Proton 會使用 範本來建立和佈建基礎設施和應用程式。

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」、「預備」和「生產」環境。「開發」環境可能包含具有私有子網路的 VPC，以及對所有 資源的限制存取政策。環境輸出可以用作 服務的輸入。

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

- 註冊並發佈使用的標準環境範本 AWS Proton，以建立佈建和管理環境基礎設施的標準環境。
- 註冊並發佈客戶受管環境範本，該範本 AWS Proton 使用 來建立連線至您現有佈建基礎設施的客戶受管環境。AWS Proton 不會管理您現有的佈建基礎設施。

您可以使用 建立並註冊服務範本 AWS Proton，以將服務部署至環境。必須先建立 AWS Proton 環境，才能將服務部署到其中。

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 IAM AWS Proton 服務角色的存取。如需詳細資訊，請參閱[the section called “IAM 角色”](#)。
- 編寫範本套件。如需詳細資訊，請參閱[範本套件](#)。
- 在範本套件編寫、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 主控台或使用 來執行此操作 AWS CLI。
- 測試並使用 範本在註冊後建立和管理 AWS Proton 佈建的資源 AWS Proton。
- 在範本的整個生命週期中，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組態來管理範本版本：

- 使用 AWS Proton 主控台和 AWS CLI 建立新的次要或主要版本。
- [建立範本同步組態](#)，讓 在您定義的儲存庫中偵測到範本套件變更時，AWS Proton 會自動建立新的次要或主要版本。

如需詳細資訊，請參閱 [AWS Proton 服務 API 參考](#)。

主題

- [版本化範本](#)
- [註冊和發佈範本](#)
- [檢視範本資料](#)
- [更新範本](#)
- [刪除範本](#)
- [範本同步組態](#)
- [服務同步組態](#)

版本化範本

身為管理員或平台團隊的成員，您可以定義、建立和管理用於佈建基礎設施資源的版本化範本程式庫。範本版本有兩種類型：次要版本和主要版本。

- 次要版本 – 變更具有回溯相容結構描述的範本。這些變更不需要開發人員在更新至新範本版本時提供新資訊。

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先前次要版本的範本回溯相容。如果新的結構描述無法回溯相容，會 AWS Proton 失敗註冊新的次要版本。

Note

相容性僅根據 schema 決定。AWS Proton 不會檢查範本套件基礎設施作為程式碼 (IaC) 檔案是否與先前的次要版本回溯相容。例如，AWS Proton 不會檢查新的 IaC 檔案是否對在由先前範本次要版本佈建的基礎設施上執行的應用程式造成中斷變更。

- 主要版本 – 對範本所做的變更可能無法回溯相容。這些變更通常需要開發人員的新輸入，而且通常涉及範本結構描述變更。

有時，您可以根據團隊的操作模型，選擇將回溯相容的變更指定為主要版本。

AWS Proton 判斷範本版本請求是針對次要或主要版本的方式，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請求時，您可以透過指定主要版本編號來請求主要版本，並且透過不指定主要版本編號來請求次要版本。

- 當您使用 [範本同步](#)（因此不提出明確的範本版本請求）時，AWS Proton 會嘗試為現有 YAML 檔案中發生的範本變更建立新的次要版本。當您為新範本變更建立新目錄（例如，從 v1 移至 v2）時，會 AWS Proton 建立主要版本。

Note

如果 AWS Proton 判斷變更不向後相容，則根據範本同步的新次要版本註冊仍會失敗。

當您發佈範本的新版本時，如果範本是最高主要和次要版本，則會成為建議版本。使用新的建議版本建立新 AWS Proton 資源，並 AWS Proton 提示管理員使用新版本，以及更新使用過時版本的現有 AWS Proton 資源。

註冊和發佈範本

您可以使用 註冊和發佈環境和服務範本 AWS Proton，如以下各節所述。

您可以使用 主控台或 建立新的範本版本 AWS CLI。

或者，您可以使用 主控台或 AWS CLI 來建立範本，並為其 [設定範本同步](#)。此組態可讓您從位於您已定義之已註冊 git 儲存庫中的範本套件進行 AWS Proton 同步。每當遞交推送到變更其中一個範本套件的儲存庫時，如果該版本不存在，就會建立新的範本次要或主要版本。若要進一步了解範本同步組態先決條件和需求，請參閱 [範本同步組態](#)。

註冊和發佈環境範本

您可以註冊和發佈以下類型的環境範本。

- 註冊並發佈標準環境範本，AWS Proton 使用 來部署和管理環境基礎設施。
- 註冊並發佈客戶受管環境範本，該範本 AWS Proton 使用 連線到您管理的現有佈建基礎設施。AWS Proton 不會管理現有的佈建基礎設施。

Important

作為管理員，請確保您的佈建和管理的基礎設施和所有輸出參數與關聯的客戶受管環境範本相容。AWS Proton 無法代表您考慮變更，因為這些變更不可見 AWS Proton。不一致會導致失敗。

您可以使用 主控台或 AWS CLI 來註冊和發佈環境範本。

AWS 管理主控台

使用 主控台來註冊和發佈新的環境範本。

1. 在 [AWS Proton 主控台](#) 中，選擇環境範本。
2. 選擇建立環境範本。
3. 在建立環境範本頁面的範本選項區段中，選擇兩個可用範本選項之一。
 - 建立範本以佈建新的環境。
 - 建立範本以使用您管理的佈建基礎設施。
4. 如果您選擇建立用於佈建新環境的範本，請在範本套件來源區段中，選擇三個可用範本套件來源選項的其中一個。若要進一步了解同步範本的需求和先決條件，請參閱 [範本同步組態](#)。
 - 使用其中一個範例範本套件。
 - 使用您自己的範本套件。
 - [從 Git 同步範本](#)。
5. 提供範本套件的路徑。
 - a. 如果您選擇使用其中一個範例範本套件：

在範本套件區段中，選取範本套件範例。
 - b. 如果您從 Git 選擇同步範本，請在來源碼區段中：
 - i. 選取範本同步組態的儲存庫。
 - ii. 輸入要從中同步的儲存庫分支名稱。
 - iii. （選用）輸入目錄名稱以限制範本套件的搜尋。
 - c. 否則，在 S3 套件位置區段中，提供範本套件的路徑。
6. 在範本詳細資訊區段中。
 - a. 輸入範本名稱。
 - b. （選用）輸入範本顯示名稱。
 - c. （選用）輸入環境範本的範本描述。
7. （選用）勾選加密設定區段中自訂加密設定的核取方塊（進階），以提供您自己的加密金鑰。

8. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
9. 選擇建立環境範本。

您現在位於顯示新環境範本狀態和詳細資訊的新頁面。這些詳細資訊包括 AWS 和客戶受管標籤的清單。當您建立 AWS Proton 資源時，AWS Proton 會自動為您產生 AWS 受管標籤。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。您和具有`proton:CreateEnvironment`許可的其他人可以檢視和存取它。依照下一個步驟讓範本可供其他人使用。
11. 在範本版本區段中，選擇您剛建立之範本次要版本左側的選項按鈕 (1.0)。或者，您可以在資訊提醒中選擇發佈，並略過下一個步驟。
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。由於它是範本的最新版本，因此它是建議版本。
14. 在導覽窗格中，選取環境範本以檢視環境範本和詳細資訊的清單。

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[版本化範本](#)。

1. 在 [AWS Proton 主控台](#) 中，選擇環境範本。
2. 在環境範本清單中，選擇您要為其建立主要或次要版本的環境範本名稱。
3. 在環境範本詳細資訊檢視中，選擇範本版本區段中的建立新版本。
4. 在建立新的環境範本版本頁面的範本套件來源區段中，選擇兩個可用範本套件來源選項之一。
 - 使用其中一個範例範本套件。
 - 使用您自己的範本套件。
5. 提供所選範本套件的路徑。
 - 如果您選擇使用其中一個範例範本套件，請在範例範本套件區段中，選取範例範本套件。
 - 如果您選擇使用自己的範本套件，請在 S3 套件位置區段中，選擇範本套件的路徑。
6. 在範本詳細資訊區段中。
 - a. (選用) 輸入範本顯示名稱。
 - b. (選用) 輸入服務範本的範本描述。
7. 在範本詳細資訊區段中，選擇下列其中一個選項。
 - 若要建立次要版本，請保留核取方塊 核取以建立新的主要版本。

- 若要建立主要版本，請勾選核取方塊 核取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 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（選用）和描述（選用）來建立標準 OR 客戶受管環境範本。
 - 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. 透過新增值為的 provisioning 參數來建立客戶受管環境範本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"
```

```
--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

對於標準和客戶受管環境範本，此步驟和其餘步驟都相同。

包含範本名稱、主要版本，以及包含您環境範本套件之儲存貯體的 S3 儲存貯體名稱和金鑰。

執行以下命令：

```
$ aws proton create-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. 透過提供範本名稱和主要和次要版本，發佈環境範本主要版本 1 的次要版本 0。此版本是 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 和客戶受管標籤的清單。AWS Proton 會自動為您產生 AWS 受管標籤。您也可以使用 修改和建立客戶受管標籤 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。如此一來，當開發人員選取服務範本時，他們可以選擇將服務部署到哪個環境。

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

如果服務部署到從移除的相容環境範本建置的環境，則無法將其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，您可以建立新的主要版本。

您可以使用 主控台或 AWS CLI 來註冊和發佈服務範本。

AWS 管理主控台

使用 主控台來註冊和發佈新的服務範本。

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件來源區段中，選擇其中一個可用的範本選項。
 - 使用您自己的範本套件。
 - 從 Git 同步範本。
4. 提供範本套件的路徑。
 - a. 如果您從 Git 選擇同步範本，請在來源碼儲存庫區段中：
 - i. 選取範本同步組態的儲存庫。
 - ii. 輸入要從中同步的儲存庫分支名稱。
 - iii. (選用) 輸入目錄名稱以限制範本套件的搜尋。
 - b. 否則，在 S3 套件位置區段中，提供範本套件的路徑。
5. 在範本詳細資訊區段中。
 - a. 輸入範本名稱。
 - b. (選用) 輸入範本顯示名稱。

- c. (選用) 輸入服務範本的範本描述。
6. 在相容環境範本區段中，從相容環境範本清單中選擇。
7. (選用) 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) 在管道區段中：

如果您未在服務範本中包含服務管道定義，請取消勾選頁面底部的管道 - 選用核取方塊。建立服務範本後，您無法變更此設定。如需詳細資訊，請參閱[範本套件](#)。

9. (選用) 在支援的元件來源區段中，針對元件來源選擇直接定義，以啟用直接定義的元件連接至您的服務執行個體。
10. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
11. 選擇建立服務範本。

您現在位於顯示新服務範本狀態和詳細資訊的新頁面。這些詳細資訊包括 AWS 和客戶受管標籤的清單。當您建立 AWS Proton 資源時，AWS Proton 會自動為您產生 AWS 受管標籤。如需詳細資訊，請參閱[AWS Proton 資源和標記](#)。

12. 新服務範本狀態的狀態會以草稿狀態開始。您和具有proton:CreateService許可的其他人可以檢視和存取它。依照下一個步驟讓範本可供其他人使用。
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選項按鈕 (1.0)。或者，您可以在資訊提醒中選擇發佈，並略過下一個步驟。
14. 在範本版本區段中，選擇發佈。
15. 範本狀態變更為已發佈。因為這是範本的最新版本，所以它是建議版本。
16. 在導覽窗格中，選取服務範本以檢視您的服務範本和詳細資訊清單。

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[版本化範本](#)。

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. 在服務範本詳細資訊檢視中，選擇範本版本區段中的建立新版本。
4. 在建立新的服務範本版本頁面的套件來源區段中，選取使用您自己的範本套件。
5. 在 S3 套件位置區段中，選擇範本套件的路徑。
6. 在範本詳細資訊區段中。

- a. (選用) 輸入範本顯示名稱。
 - b. (選用) 輸入服務範本的範本描述。
7. 在範本詳細資訊區段中，選擇下列其中一個選項。
- 若要建立次要版本，請保留核取方塊 `核取以建立新的主要版本`。
 - 若要建立主要版本，請勾選核取方塊 `核取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道的情況下部署服務的服務範本，請將參數和值新增至 `--pipeline-provisioning "CUSTOMER_MANAGED" create-service-template` 命令。如 [範本套件](#) 建立和 [中所述](#) 設定您的範本套件 [服務範本套件的結構描述需求](#)。

Note

建立服務範本 `pipelineProvisioning` 後，您無法修改。

1. 使用 CLI 註冊和發佈新的服務範本，無論是否有服務管道，如下列步驟所示。
 - a. 使用 CLI 建立具有服務管道的服務範本。

指定名稱、顯示名稱 (選用) 和描述 (選用)。

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. 建立不含服務管道的服務範本。

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、相容的環境範本、主要版本，以及包含服務範本套件之儲存貯體的 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 命令來檢查註冊狀態。

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. 使用更新命令將狀態變更為 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\n pipeline_input_type: \"MyPipelineInputType\"\n service_input_type:  
\"MyServiceInstanceInputType\"\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 MyServiceInstanceInputType:  
\n type: object\n description: \"Service instance input properties\"\n\n\n required:\n - my_sample_service_instance_required_input\n\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"  
  }  
}
```

5. 使用 get 命令擷取服務範本詳細資訊，檢查 AWS Proton 是否已發佈 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 檢視具有詳細資訊的範本清單，以及檢視具有詳細資訊的個別範本 AWS CLI。

客戶受管環境範本資料包含 值為 的 `provisioned` 參數 `CUSTOMER_MANAGED`。

如果服務範本不包含服務管道，則服務範本資料會包含值為 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 管理主控台

使用 主控台 列出和檢視範本。

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

檢視範本的詳細資訊、範本主要和次要版本的清單、使用範本版本和範本標籤部署 AWS Proton 的資源清單。

建議的主要版本和次要版本標記為建議。

AWS CLI

使用 AWS CLI 來列出和檢視範本。

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:\n    \"MyServiceInstanceInputType\"\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:\n            \"Another sample input\"\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"
    }
  }
}

```

檢視沒有服務管道的服務範本，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

您可以更新範本，如下列清單所述。

- 當您使用 主控台description或 時display name，編輯範本的 或 AWS CLI。您無法編輯範本name的。
- 當您使用 主控台或 時，更新範本次要版本的狀態 AWS CLI。您只能將狀態從 變更為 DRAFT PUBLISHED。
- 當您使用 時，編輯範本次要或主要版本的顯示名稱和描述 AWS CLI。

AWS 管理主控台

使用主控台編輯範本描述和顯示名稱，如下列步驟所述。

在 範本清單中。

1. 在 [AWS Proton 主控台](#) 中，選擇（環境或服務）範本。
2. 在範本清單中，選擇您要更新其描述或顯示名稱之範本左側的選項按鈕。
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訊區段中，在表單中輸入編輯，然後選擇儲存變更。

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

在（環境或服務）範本詳細資訊頁面中。

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 Draft 更新為 Published 的範本名稱。
3. 在（環境或服務）範本詳細資訊頁面的範本版本區段中，選取您要發佈之次要版本左側的選項按鈕。
4. 在範本版本區段中選擇發佈。狀態會從 Draft 變更為 Published。

AWS CLI

下列範例命令和回應顯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。如需更新服務範本次要版本狀態的範例，請參閱[註冊和發佈服務範本](#)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，您可以刪除環境範本的次要版本。

如果沒有服務執行個體或管道部署至該版本，您可以刪除服務範本的次要版本。您的管道可以部署到與服務執行個體不同的範本版本。例如，如果您的服務執行個體從 1.0 更新至 1.1 版，且您的管道仍部署至 1.0 版，則無法刪除服務範本 1.0。

AWS 管理主控台

您可以使用 主控台來刪除整個範本或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時。

- 當您刪除整個範本時，也會刪除範本的主要和次要版本。

在（環境或服務）範本清單中。

1. 在 [AWS Proton 主控台](#) 中，選擇（環境或服務）範本。
2. 在範本清單中，選取您要刪除之範本左側的選項按鈕。

只有在沒有 AWS Proton 資源部署到其版本時，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. 模態會提示您確認刪除動作。
5. 遵循指示並選擇是，刪除。

在（環境或服務）範本詳細資訊頁面中。

1. 在 [AWS Proton 主控台](#) 中，選擇（環境或服務）範本。
2. 在範本清單中，選擇您要完全刪除或刪除其個別主要或次要版本的範本名稱。
3. 刪除整個範本。

只有在沒有 AWS Proton 資源部署到其版本時，您才能刪除整個範本。

- a. 選擇刪除，頁面右上角。
- b. 模態會提示您確認刪除動作。
- c. 遵循指示並選擇是，刪除。

4. 刪除範本的主要或次要版本。

只有在沒有 AWS Proton 資源部署到該版本時，您才能刪除範本的次要版本。

- a. 在範本版本區段中，選取您要刪除之版本左側的選項按鈕。
- b. 在範本版本區段中選擇刪除。
- c. 模態會提示您確認刪除動作。
- d. 遵循指示並選擇是，刪除。

AWS CLI

AWS CLI 範本刪除操作不包含刪除範本的其他版本。使用時 AWS CLI，請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。
- 當您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 AWS Proton 資源部署到該版本，請刪除範本的次要版本。
- 如果範本沒有其他次要版本，而且沒有 AWS Proton 資源部署到該版本，請刪除建議的範本次要版本。

下列範例命令和回應示範如何使用 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"
}
}
```

範本同步組態

了解如何設定範本，以便從您定義的已註冊 git 儲存庫中的範本套件進行 AWS Proton 同步。將遞交推送至儲存庫時，會 AWS Proton 檢查儲存庫範本套件的變更。如果偵測到範本套件變更，則會建立其範本的新次要或主要版本，如果版本尚不存在。AWS Proton 目前支援 GitHub、GitHub Enterprise 和 BitBucket。

將遞交推送至同步範本套件

當您將遞交推送至其中一個範本所追蹤的分支時，會 AWS Proton 複製您的儲存庫，並決定需要同步哪些範本。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 確定哪些範本和主要版本與您的儲存庫和分支相關聯後，它會開始嘗試平行同步所有這些範本。

每次同步到特定範本時，AWS Proton 首先檢查範本目錄的內容自上次成功同步以來是否變更。如果內容未變更，AWS Proton 請略過註冊重複的套件。這可確保在範本套件的內容變更時建立新的範本次要版本。如果範本套件的內容已變更，則會向註冊套件 AWS Proton。

註冊範本套件之後，會 AWS Proton 監控註冊狀態，直到註冊完成為止。

在單一指定時間，特定範本次要和主要版本只能發生一次同步。同步進行期間可能已推送的任何遞交都會批次處理。批次遞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 儲存庫同步環境和服務範本。若要同步您的服務範本，請將名為的額外檔案新增至範本套件中的 .template-registration.yaml 每個主要版本目錄。此檔案包含在遞交後為您建立服務範本版本時 AWS Proton 所需的其他詳細資訊：相容的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”](#)。

範本同步組態考量事項

檢閱下列使用範本同步組態的考量事項。

- 儲存庫不得超過 250 MB。
- 若要設定範本同步，請先將儲存庫連結至 AWS Proton。如需詳細資訊，請參閱[the section called “建立儲存庫連結”](#)。
- 從同步範本建立新範本版本時，該版本處於 DRAFT 狀態。
- 如果下列其中一項為 true，則會建立新的範本次要版本：
 - 範本套件內容與上次同步範本次要版本的內容不同。
 - 已刪除上次同步的範本次要版本。
- 同步無法暫停。
- 新的次要或主要版本都會自動同步。
- 範本同步組態無法建立新的頂層範本。
- 您無法從具有範本同步組態的多個儲存庫同步至一個範本。
- 您無法使用標籤而非分支。
- 當您[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，並將其新增為相同遞交中服務範本的相容環境。
- 同步至單一範本主要版本會一次執行一個。在同步期間，如果偵測到任何新的遞交，則會在作用中同步結束時批次處理並套用。同步到不同的範本主要版本會平行進行。
- 如果您變更範本要同步的分支，則舊分支中任何持續的同步都會先完成。然後，同步會從新的分支開始。
- 如果您變更範本同步的儲存庫，舊儲存庫中任何進行中的同步都可能會失敗或執行至完成。這取決於他們所在的同步階段。

如需詳細資訊，請參閱[AWS Proton 服務 API 參考](#)。

主題

- [建立範本同步組態](#)
- [檢視範本同步組態詳細資訊](#)
- [編輯範本同步組態](#)

- [刪除範本同步組態](#)

建立範本同步組態

了解如何使用 建立範本同步組態 AWS Proton。

建立範本同步組態先決條件：

- 您已將[儲存庫與連結](#) AWS Proton。
- [範本套件](#)位於您的儲存庫中。

儲存庫連結包含下列項目：

- CodeConnections 連線提供存取儲存庫和訂閱其通知的 AWS Proton 許可。
- [服務連結角色](#)。當您連結儲存庫時，系統會為您建立服務連結角色。

建立第一個範本同步組態之前，請將範本套件推送至您的儲存庫，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/
```

建立第一個範本同步組態之後，當您推送在新版本（例如，在下）下新增更新範本套件的遞交時，系統會自動建立新的範本版本/my-env-template/v2/。

```
/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/
```

您可以在單一遞交中包含一或多個同步設定範本的新範本套件版本。會為每個包含在遞交中的新範本套件版本 AWS Proton 建立新的範本版本。

建立範本同步組態之後，您仍然可以從 S3 儲存貯體上傳範本套件，AWS CLI 在 主控台 或使用 手動 建立新版本的範本。範本同步只能以一個方向運作：從儲存庫到 AWS Proton。手動建立的範本版本不會同步。

設定範本同步組態之後，AWS Proton 請列出儲存庫的變更。每當推送變更時，就會尋找與範本同名的任何目錄。然後，它會在該目錄中尋找任何看起來像主要版本的目錄。會將範本套件 AWS Proton 註冊到對應的範本主要版本。新版本一律處於 DRAFT 狀態。您可以使用 主控台 或 [發佈新版本](#) AWS CLI。

例如，假設您有一個名為 的範本，my-env-template 其設定為 main 使用以下配置在分支 my-repo/templates 上從 同步。

```
/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 會將 的內容同步 /templates/my-env-template/v1/ 至 my-env-template:1 並將 的內容同步 /templates/my-env-template/v2/ 至 my-env-template:2。如果它們尚未存在，則會建立這些主要版本。

AWS Proton 找到第一個符合範本名稱的目錄。您可以在建立或編輯範本同步組態 subdirectoryPath 時指定 ，以限制目錄 AWS Proton 搜尋。例如，您可以 /production-templates/ 為 指定 subdirectoryPath。

您可以使用 主控台 或 CLI 建立範本同步組態。

AWS 管理主控台

使用 主控台 建立範本和範本同步組態。

1. 在 [AWS Proton 主控台](#) 中，選擇環境範本。
2. 選擇建立環境範本。

3. 在建立環境範本頁面的範本選項區段中，選擇建立範本以佈建新的環境。
4. 在範本套件來源區段中，從 Git 選擇同步範本。
5. 在來源碼儲存庫區段中：
 - a. 針對儲存庫，選取包含範本套件的連結儲存庫。
 - b. 針對分支，選取要從中同步的儲存庫分支。
 - c. (選用) 針對範本套件目錄，輸入要縮小範本套件搜尋範圍的目錄名稱。
6. 在範本詳細資訊區段中。
 - a. 輸入範本名稱。
 - b. (選用) 輸入範本顯示名稱。
 - c. (選用) 輸入環境範本的範本描述。
7. (選用) 勾選加密設定區段中自訂加密設定的核取方塊 (進階)，以提供您自己的加密金鑰。
8. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
9. 選擇建立環境範本。

您現在位於顯示新環境範本狀態和詳細資訊的新頁面。這些詳細資訊包括 AWS 受管和客戶受管標籤的清單。當您建立 AWS Proton 資源時，AWS Proton 會自動為您產生 AWS 受管標籤。如需詳細資訊，請參閱[AWS Proton 資源和標記](#)。

10. 在範本詳細資訊頁面中，選擇同步索引標籤以檢視範本同步組態詳細資訊。
11. 選擇範本版本索引標籤，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。您和具有`proton:CreateEnvironment`許可的其他人可以檢視和存取它。依照下一個步驟，讓範本可供其他人使用。
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選項按鈕 (1.0)。或者，您可以在資訊提醒中選擇發佈，並略過下一個步驟。
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。這是範本的最新和建議版本。
16. 在導覽窗格中，選取環境範本以檢視環境範本和詳細資訊的清單。

建立服務範本和範本同步組態的程序類似。

AWS CLI

使用 建立範本和範本同步組態 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"
```

回應如下所示。

```
{  
  "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. 透過提供下列項目 AWS CLI，使用 建立範本同步組態：
 - 您要同步的範本。建立範本同步組態之後，您仍然可以在 主控台或使用 手動建立新版本 AWS CLI。
 - 範本名稱。
 - 範本類型。
 - 您要從中同步的連結儲存庫。
 - 連結的儲存庫提供者。
 - 範本套件所在的分支。
 - (選用) 包含範本套件的目錄路徑。根據預設，AWS Proton 尋找符合您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/"
```

回應如下所示。

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

- 若要發佈範本版本，請參閱 [註冊和發佈範本](#)。

同步服務範本

上述範例示範如何同步環境範本。服務範本類似。若要同步服務範本，請將名為 的額外檔案新增至範本套件中的 `.template-registration.yaml` 每個主要版本目錄。此檔案包含在遞交後為您建立服務範本版本時 AWS Proton 所需的其他詳細資訊。當您使用 AWS Proton 主控台或 API 明確建立服務範本版本時，您會提供這些詳細資訊做為輸入，而此檔案會取代這些輸入以進行範本同步。

```
./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/
```

`.template-registration.yaml` 檔案包含下列詳細資訊：

- 相容的環境 **【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基於此服務範本版本的服務相容。如果未指定，則無法將元件連接至這些服務。如需元件的詳細資訊，請參閱 [元件](#)。

檔案的 YAML 語法如下：

```
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` 是選用的，唯一支援的值是 `DIRECTLY_DEFINED`。

Example.template-registration.yaml

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` 環境範本的主要版本 1 和 3 相容。檔案未指定 `supported_component_sources`，因此無法根據此服務範本版本將元件連接至服務。

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

Note

先前，為指定相容的環境 AWS Proton 定義了不同的檔案 `.compatible-envs`。AWS Proton 支援該檔案及其格式，以實現回溯相容性。我們不建議使用它，因為它無法擴展，也無法支援較新的功能，例如元件。

檢視範本同步組態詳細資訊

使用主控台或 CLI 檢視範本同步組態詳細資訊。

AWS 管理主控台

使用 主控台檢視範本同步組態詳細資訊。

1. 在導覽窗格中，選擇（環境或服務）範本。
2. 若要檢視詳細資訊，請選擇您為其建立範本同步組態的範本名稱。
3. 在範本的詳細資訊頁面中，選取同步索引標籤以檢視範本同步組態詳細資訊。

AWS CLI

使用 AWS CLI 檢視同步範本。

執行下列命令。

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

回應如下所示。

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

使用 AWS CLI 取得範本同步狀態。

針對 `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 "1.0.0"
```

```
--template-version "1"
```

編輯範本同步組態

您可以編輯 `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 編輯範本同步組態。

AWS 管理主控台

使用 主控台編輯範本同步組態分支。

在 範本清單中。

1. 在 [AWS Proton 主控台](#) 中，選擇（環境或服務）範本。
2. 在範本清單中，選擇範本名稱與您要編輯的範本同步組態。
3. 在範本詳細資訊頁面中，選擇範本同步索引標籤。
4. 在範本同步詳細資訊區段中，選擇編輯。
5. 在編輯頁面的來源碼儲存庫區段中，針對分支選取分支，然後選擇儲存組態。

AWS CLI

下列範例命令和回應顯示如何使用 `branchCLI` 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"
```

回應如下所示。

```
{  
  "templateSyncConfigDetails": {
```

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

您同樣可以使用 AWS CLI 來更新同步的服務範本。

刪除範本同步組態

使用主控台或 CLI 刪除範本同步組態。

AWS 管理主控台

使用主控台刪除範本同步組態。

1. 在範本詳細資訊頁面中，選擇同步索引標籤。
2. 在同步詳細資訊區段中，選擇中斷連線。

AWS CLI

下列範例命令和回應示範如何使用 AWS CLI 刪除同步的範本組態。

執行下列命令。

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

回應如下所示。

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

服務同步組態

透過服務同步，您可以使用 Git 來設定和部署 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請求等功能來設定、管理和部署您的服務。服務同步結合了 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設施。它可管理 Git 儲存庫中的服務定義，並減少工具切換。相較於單獨使用 Git，中的 AWS Proton 範本和部署標準化可協助您減少管理基礎設施的時間。AWS Proton 也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 OPS 檔案

`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。它還定義了在中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提升到另一個執行個體。

`proton-ops` 檔案支援使用規格檔案或連結儲存庫中的多個規格檔案來同步服務執行個體。您可以在 `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 `./configuration/proton-ops.yaml`：

```
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，而 `alpha`、`prod-two`、`beta` 和 `prod-one`、`prod-three`是執行個體。

`spec` 檔案可以是檔案內定義的所有執行個體或執行個體子集`proton-ops`。不過，它至少必須在分支中定義執行個體，以及要同步的規格。如果未在 `proton-ops` 檔案中定義執行個體，且具有特定分支和`spec`檔案位置，則服務同步不會建立或更新這些執行個體。

下列範例顯示`spec`檔案的外觀。請記住，`proton-ops`檔案會從這些`spec`檔案同步。

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"
```

如果執行個體未同步，且嘗試同步時持續發生問題，呼叫 [GetServiceInstanceSyncStatus](#) API 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。

封鎖程式

透過使用服務同步來 AWS Proton 同步服務，您可以更新您的服務規格，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 管理主控台 AWS CLI。

AWS Proton 有助於避免覆寫您透過 AWS 管理主控台 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。為了達成此目的，AWS Proton 會在偵測到手動變更時停用服務同步，以自動建立服務同步封鎖程式。

若要取得與服務相關聯的所有封鎖程式，您必須針對與服務 `serviceInstance` 相關聯的每個執行下列動作：

- 僅使用 呼叫 `getServiceSyncBlockerSummary API` `serviceName`。
- 使用 `serviceName` 和 呼叫 `getServiceSyncBlockerSummary API` `serviceInstanceName`。

這會傳回最新的封鎖程式清單，以及與其相關聯的狀態。如果有任何封鎖程式標示為 `ACTIVE`，您必須呼叫 `UpdateServiceSyncBlocker API` 來解決這些問題，`resolvedReason` 並針對每個封鎖程式呼叫 `blockerId` 和 `blockerId`。

如果您手動更新或建立服務執行個體，會在服務執行個體上 AWS Proton 建立服務同步封鎖程式。AWS Proton 會繼續同步所有其他服務執行個體，但會停用此服務執行個體的同步，直到封鎖程式解決為止。如果您從服務中刪除服務執行個體，會在服務上 AWS Proton 建立服務同步封鎖程式。這 AWS Proton 可防止同步任何服務執行個體，直到封鎖程式解決為止。

在您擁有所有作用中的封鎖程式之後，您必須透過呼叫 `UpdateServiceSyncBlocker API` 搭配 `blockerId` 和每個作用中 `resolvedReason` 的封鎖程式來解決這些問題。

使用 AWS 管理主控台，您可以透過導覽並選擇 AWS Proton 服務同步索引標籤來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，則會顯示啟用按鈕。若要啟用服務同步，請選擇啟用。

主題

- [建立服務同步組態](#)
- [檢視服務同步的組態詳細資訊](#)
- [編輯服務同步組態](#)
- [刪除服務同步組態](#)

建立服務同步組態

您可以使用 主控台 或 建立服務同步組態 AWS CLI。

AWS 管理主控台

1. 在選擇服務範本頁面上，選取範本，然後選擇設定。
2. 在設定服務頁面上的服務詳細資訊區段中，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，請選擇連結另一個 Git 儲存庫，並遵循[建立儲存庫連結](#)中的指示。
5. 對於儲存庫，從清單中選擇來源碼儲存庫的名稱。
6. 針對分支，從清單中選擇來源碼的儲存庫分支名稱。
7. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
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，如果您想要 AWS Proton 建立proton-ops檔案，請選取我想要 AWS Proton 建立檔案。使用此選項，會在您指定的位置 AWS Proton 建立 spec和 proton-ops 檔案。選取我提供自己的檔案，以建立您自己的 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. 針對儲存庫名稱，從清單中選擇來源碼儲存庫的名稱。
13. 對於proton-ops檔案分支，請從 AWS Proton 將放置 OPS 和規格檔案的清單中選擇分支的名稱。
14. 在服務執行個體區段中，每個欄位都會根據proton-ops檔案中的值自動填入。
15. 選擇下一步並檢閱您的輸入。
16. 選擇建立。

AWS CLI

使用 建立服務同步組態 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)
```

回應如下所示。

```
{
  "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訊資料](#) AWS CLI。

AWS 管理主控台

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，選擇服務。
2. 若要檢視詳細資訊，請選擇您為其建立服務同步組態的服務名稱。
3. 在服務的詳細資訊頁面中，選取服務同步索引標籤以檢視服務同步的組態詳細資訊資料。

AWS CLI

使用 AWS CLI 取得同步服務。

執行下列命令。

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

回應如下所示。

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

```
    "serviceName": "service name"
  }
}
```

使用 AWS CLI 取得服務同步狀態。

執行下列命令。

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

編輯服務同步組態

您可以使用 主控台 或 編輯服務同步組態 AWS CLI。

AWS 管理主控台

使用主控台編輯服務同步組態。

1. 在導覽窗格中，選擇服務。
2. 若要檢視詳細資訊，請選擇您為其建立服務同步組態的服務名稱。
3. 在服務詳細資訊頁面上，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 編輯服務同步組態 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"
```

回應如下所示。

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

刪除服務同步組態

您可以使用 主控台 或 刪除服務同步組態 AWS CLI。

AWS 管理主控台

使用主控台刪除服務同步組態

1. 在服務詳細資訊頁面上，選擇服務同步索引標籤。
2. 在服務同步詳細資訊區段中，選擇中斷連線以中斷儲存庫的連線。您的儲存庫中斷連線後，我們就不會再同步該儲存庫的服務。

AWS CLI

下列範例命令和回應示範如何使用 AWS CLI 刪除服務同步組態。


執行下列命令。

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

回應如下所示。

```
{
  "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執行個體。它只會刪除組態。

AWS Proton 環境

對於 AWS Proton，環境代表[服務](#)部署所在的一組共用資源和政策 AWS Proton。它們可以包含預期在 AWS Proton 服務執行個體之間共用的任何資源。這些資源可能包括 VPCs、叢集和共用負載平衡器或 API Gateway。必須先建立 AWS Proton 環境，才能將服務部署到其中。

本節說明如何使用建立、檢視、更新和刪除操作來管理環境。如需 >其他資訊，請參閱[AWS Proton 服務 API 參考](#)。

主題

- [IAM 角色](#)
- [建立環境](#)
- [檢視環境資料](#)
- [更新環境](#)
- [刪除環境](#)
- [環境帳戶連線](#)
- [客戶受管環境](#)
- [CodeBuild 佈建角色建立](#)

IAM 角色

透過 AWS Proton，您可以為您擁有和管理 AWS 的資源提供 IAM 角色和 AWS KMS 金鑰。這些稍後會套用到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。

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的所有佈建基礎設施所需的所有許可。如需角色範例，請參閱 [AWS Proton 使用 佈建的 服務角色 CloudFormation](#)。如果您使用環境帳戶連線和環境帳戶，您可以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](#)。

您提供此服務角色的方式，以及擔任該角色的人員，取決於您環境的佈建方法。

- AWS 受管佈建 – 您可以在建立環境時直接提供角色給 AWS Proton，或透過帳戶連線間接提供角色。AWS Proton 會擔任相關帳戶中的角色來佈建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設定為使用適當的登入資料擔任適當的角色。如需擔任角色的 GitHub 動作範例，請參閱「設定 AWS 登入資料」動作 GitHub 動作文件中[的擔任角色](#)。

如需佈建方法的詳細資訊，請參閱 [the section called “佈建方法”](#)。

建立環境

了解如何建立 AWS Proton 環境。

您可以透過下列兩種方式之一建立 AWS Proton 環境：

- 使用標準環境範本建立、管理和佈建標準環境。為您的環境 AWS Proton 佈建基礎設施。
- 使用客戶受管環境範本 AWS Proton 連線至客戶受管基礎設施。您可以在之外佈建自己的共用資源 AWS Proton，然後提供 AWS Proton 可使用的佈建輸出。

建立環境時，您可以選擇多種佈建方法之一。

- AWS 受管佈建 – 在單一帳戶中建立、管理和佈建環境。AWS Proton 佈建您的環境。

此方法僅支援 CloudFormation 基礎設施程式碼 (IaC) 範本。

- AWS 受管佈建至另一個帳戶 – 在單一管理帳戶中，建立和管理在具有環境帳戶連線的另一個帳戶中佈建的環境。在另一個帳戶中 AWS Proton 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](#)。

此方法僅支援 CloudFormation IaC 範本。

- 自我管理佈建 – 使用您自己的佈建基礎設施，將佈建提取請求 AWS Proton 提交至連結的儲存庫。

此方法僅支援 Terraform IaC 範本。

- CodeBuild 佈建 – AWS Proton 用來 AWS CodeBuild 執行您提供的 shell 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設施和產生輸出值。此方法的範本套件包含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。

使用 CodeBuild 佈建的範例包括使用 AWS Cloud Development Kit (AWS CDK) 佈建 AWS 資源的程式碼，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 套件”](#)。

Note

您可以搭配環境和服務使用 CodeBuild 佈建。目前您無法以這種方式佈建元件。

使用 AWS 受管佈建（在相同帳戶和另一個帳戶中），AWS Proton 會直接呼叫來佈建您的資源。

透過自我管理佈建，AWS Proton 會提取請求以提供編譯的 IaC 檔案，供 IaC 引擎用來佈建資源。

如需詳細資訊，請參閱[the section called “佈建方法”](#)、[the section called “範本套件”](#)及[the section called “環境結構描述要求”](#)。

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 主控台或 AWS CLI 在單一帳戶中建立和佈建環境。佈建由 管理 AWS。

AWS 管理主控台

使用 主控台在單一帳戶中建立和佈建環境

1. 在 [AWS Proton 主控台](#) 中，選擇環境。
2. 選擇 Create environment (建立環境)。
3. 在選擇環境範本頁面中，選取範本，然後選擇設定。
4. 在設定環境頁面的佈建區段中，選擇AWS 受管佈建。
5. 在部署帳戶區段中，選擇此 AWS 帳戶。
6. 在設定環境頁面的環境設定區段中，輸入環境名稱。
7. （選用）輸入環境的描述。
8. 在環境角色區段中 AWS Proton，選取您建立做為一部分的服務角色[設定 AWS Proton 服務角色](#)。
9. （選用）在元件角色區段中，選取服務角色，讓直接定義的元件能夠在環境中執行，並縮小可佈建的資源範圍。如需詳細資訊，請參閱[元件](#)。

10. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
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參數的值。您可以輸入optional參數的值，或在指定時使用預設值。
13. 選擇下一步並檢閱您的輸入。
14. 選擇建立。

檢視環境詳細資訊和狀態，以及環境的 AWS 受管標籤和客戶受管標籤。

15. 在導覽窗格中，選擇 Environments (環境)。

新頁面會顯示您的環境清單，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，您可以指定[AWS Proton 服務角色](#) ARN、規格檔案的路徑、環境名稱、環境範本 ARN、主要和次要版本，以及描述 (選用)。

下一個範例顯示YAML格式化的規格檔案，指定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` 命令來檢視環境範本結構描述。

```
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/AWS ProtonServiceRole" \
  --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"
  }
}
```

建立新環境後，您可以檢視 AWS 和客戶受管標籤的清單，如下列範例 command. AWS Proton automatic 為您產生 AWS 受管標籤所示。您也可以使用 修改和建立客戶受管標籤 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 主控台或 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設施。佈建由 管理 AWS。

使用主控台或 CLI 之前，請完成下列步驟。


1. 識別管理和環境帳戶的 AWS 帳戶 IDs，並複製以供日後使用。
2. 在環境帳戶中，建立具有要建立之環境最低許可 AWS Proton 的服務角色。如需詳細資訊，請參閱[AWS Proton 使用 佈建的 服務角色 CloudFormation](#)。

AWS 管理主控台

使用 主控台在一個帳戶中建立環境，並在另一個帳戶中佈建。


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請求。
 - a. 在[AWS Proton 主控台](#)的導覽窗格中，選擇環境帳戶連線。

- b. 在環境帳戶連線頁面中，選擇請求連線。

 Note

確認環境帳戶連線頁面標題中列出的帳戶 ID 符合您預先識別的環境帳戶 ID。

- c. 在請求連線頁面的環境角色區段中，選取現有服務角色和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 AWS Proton 您環境的管理帳戶 ID 和環境名稱。複製名稱以供日後使用。
 - e. 選擇頁面右下角的請求連線。
 - f. 您的請求會在傳送至管理帳戶資料表的環境連線中顯示為待定，而模式顯示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請求。
 - a. 登入您的管理帳戶，然後在 AWS Proton 主控台中選擇環境帳戶連線。
 - b. 在環境帳戶連線頁面的環境帳戶連線請求表中，選取環境帳戶連線與環境帳戶 ID 相符的環境帳戶 ID。

 Note

確認環境帳戶連線頁面標題中列出的帳戶 ID 符合您預先識別的管理帳戶 ID。

- c. 選擇 Accept (接受)。狀態會從待定變更為已連線。
3. 在管理帳戶中，建立環境。
 - a. 在導覽窗格中，選擇環境範本。
 - b. 在環境範本頁面中，選擇建立環境範本。
 - c. 在選擇環境範本頁面中，選擇環境範本。
 - d. 在設定環境頁面的佈建區段中，選擇AWS 受管佈建。
 - e. 在部署帳戶區段中，選擇另一個 AWS 帳戶；。
 - f. 在環境詳細資訊區段中，選取您的環境帳戶連線和環境名稱。
 - g. 選擇下一步。
 - h. 填寫表單並選擇下一步，直到您到達檢閱和建立頁面為止。
 - i. 檢閱並選擇建立環境。

AWS CLI

使用 AWS CLI 在一個帳戶中建立環境，並在另一個帳戶中佈建。

在環境帳戶中，建立環境帳戶連線，並執行下列命令請求連線。

```
$ 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請求。

```
$ 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",
    "environmentAccountId": "222222222222",
    "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "environmentId": "222222222222",
    "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
    "name": "simple-env-connected",
    "templateName": "simple-env-template"
  }
}
```

使用自我管理佈建建立和佈建環境

當您使用自我管理佈建時，會使用您自己的佈建基礎設施，將佈建提取請求 AWS Proton 提交至連結的儲存庫。提取請求會啟動您自己的工作流，呼叫 AWS 服務來佈建基礎設施。

自我管理的佈建考量：

- 建立環境之前，請設定儲存庫資源目錄以進行自我管理佈建。如需詳細資訊，請參閱[AWS Proton 基礎設施即程式碼檔案](#)。
- 建立環境後，會 AWS Proton 等待接收有關基礎設施佈建狀態的非同步通知。您的佈建程式碼必須使用 API `AWS Proton NotifyResourceStateChange` 來傳送這些非同步通知 AWS Proton。


您可以在 主控台或 中使用自我管理佈建 AWS CLI。下列範例示範如何搭配 Terraform 使用自我管理佈建。

AWS 管理主控台

使用 主控台，使用自我管理佈建建立 Terraform 環境。

1. 在 [AWS Proton 主控台](#) 中，選擇環境。
2. 選擇 Create environment (建立環境)。

3. 在選擇環境範本頁面中，選取 Terraform 範本，然後選擇設定。
4. 在設定環境頁面的佈建區段中，選擇自我管理佈建。
5. 在佈建儲存庫詳細資訊區段中：
 - a. 如果您尚未將[佈建儲存庫連結至 AWS Proton](#)，請選擇新儲存庫，選擇其中一個儲存庫提供者，然後針對 CodeStar 連線，選擇其中一個連線。

 Note

如果您尚未連線到相關的儲存庫提供者帳戶，請選擇新增 CodeStar 連線。然後，建立連線，然後選擇 CodeStar 連線功能表旁的重新整理按鈕。您現在應該能夠在選單中選擇新的連線。

如果您已將儲存庫連結至 AWS Proton，請選擇現有的儲存庫。

- b. 針對儲存庫名稱，選擇儲存庫。下拉式功能表會顯示現有儲存庫的連結儲存庫，或新儲存庫提供者帳戶中的儲存庫清單。
 - c. 針對分支名稱，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. (選用) 輸入環境的描述。
8. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
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參數的值。您可以輸入optional參數的值，或在指定時使用預設值。
11. 選擇下一步並檢閱您的輸入。
12. 選擇建立以傳送提取請求。
 - 如果您核准提取請求，則部署正在進行中。
 - 如果您拒絕提取請求，環境建立會取消。
 - 如果提取請求逾時，則環境建立未完成。
13. 檢視環境詳細資訊和狀態，以及環境的 AWS 受管標籤和客戶受管標籤。
14. 在導覽窗格中，選擇 Environments (環境)。

新頁面會顯示您的環境清單，以及狀態和其他環境詳細資訊。

AWS CLI

當您使用自我管理佈建建立環境時，您可以新增 `provisioningRepository` 參數並省略 `ProtonServiceRoleArn` 和 `environmentAccountIdConnectionId` 參數。

使用 AWS CLI 建立具有自我管理佈建的 Terraform 環境。

1. 建立環境，並將提取請求傳送至儲存庫以供檢閱和核准。

下一個範例顯示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` 命令來檢視環境範本結構描述。

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：>

```
{
  "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，則佈建正在進行中。
- 如果您拒絕請求，環境建立會取消。
- 如果提取請求逾時，環境建立不會完成。

3. 以非同步方式提供佈建狀態給 AWS Proton。下列範例會通知 佈建 AWS Proton 成功。

```
$ 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 檢視環境詳細資訊 AWS CLI。

AWS 管理主控台

您可以使用 [AWS Proton 主控台](#) 檢視具有詳細資訊的環境清單，以及具有詳細資訊的個別環境清單。

1. 若要檢視您的環境清單，請在導覽窗格中選擇環境。
2. 若要檢視詳細資訊，請選擇環境的名稱。

檢視您的環境詳細資訊。

AWS CLI

使用 AWS CLI 取得或列出環境詳細資訊。

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 環境與環境帳戶連線相關聯，請勿更新或包含 `protonServiceRoleArn` 參數來更新或連線至環境帳戶連線。

只有在下列兩者都成立時，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的環境相關聯。

如果環境未與環境帳戶連線相關聯，請勿更新或包含 `environmentAccountId` 參數。

您可以更新 `environmentAccountId` 或 `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，請勿更新 `provisioning-repository` 參數，並省略 `environmentAccountId` 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，`deployment-type` 欄位會定義模式。使用主控台時，這些模式會對應至從動作下拉式清單的編輯、更新、更新次要和更新主要動作。

NONE

在此模式中，不會發生部署。只會更新請求的中繼資料參數。

CURRENT_VERSION

在此模式中，會使用您提供的新規格來部署和更新環境。只會更新請求的參數。使用此時，請勿包含次要或主要版本參數`deployment-type`。

MINOR_VERSION

在此模式中，環境會依預設使用中目前主要版本的已發佈、建議（最新）次要版本進行部署和更新。您也可以指定目前使用中主要版本的不同次要版本。

MAJOR_VERSION

在此模式中，預設會使用發佈、建議（最新）主要和次要版本的目前範本來部署和更新環境。您也可以指定高於使用中主要版本的不同主要版本，以及次要版本（選用）。

主題

- [更新 AWS 受管佈建環境](#)
- [更新自我管理的佈建環境](#)
- [取消進行中的環境部署](#)

更新 AWS 受管佈建環境

只有使用佈建的環境才支援標準佈建 CloudFormation。

使用主控台或 AWS CLI 來更新您的環境。

AWS 管理主控台

使用 主控台更新環境，如下列步驟所示。

1. 選擇下列 2 個步驟中的 1 個。
 - a. 在環境清單中。
 - i. 在 [AWS Proton 主控台](#) 中，選擇環境。
 - ii. 在環境清單中，選擇您要更新之環境左側的選項按鈕。
 - b. 在主控台環境詳細資訊頁面中。
 - i. 在 [AWS Proton 主控台](#) 中，選擇環境。
 - ii. 在環境清單中，選擇您要更新的環境名稱。
2. 選擇接下來 4 個步驟中的 1 個來更新您的環境。
 - a. 進行不需要環境部署的編輯。
 - i. 例如，變更描述。
選擇編輯。
 - ii. 填寫表單並選擇下一步。
 - iii. 檢閱您的編輯，然後選擇更新。
 - b. 僅更新中繼資料輸入。
 - i. 選擇動作，然後選擇更新。
 - ii. 填寫表單，然後選擇編輯。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。
 - c. 更新其環境範本的新次要版本。
 - i. 選擇動作，然後選擇更新次要。
 - ii. 填寫表單，然後選擇下一步。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。

- d. 更新其環境範本的新主要版本。
 - i. 選擇動作，然後選擇更新主要。
 - ii. 填寫表單，然後選擇下一步。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。

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佈建。

使用 主控台或 AWS CLI 來更新您的環境。

AWS 管理主控台

使用 主控台更新環境，如下列步驟所示。

1. 選擇下列 2 個步驟中的 1 個。
 - a. 在環境清單中。
 - i. 在 [AWS Proton 主控台](#) 中，選擇環境。
 - ii. 在環境清單中，選擇您要更新的環境範本左側的選項按鈕。

- b. 在主控制台環境詳細資訊頁面中。
 - i. 在 [AWS Proton 主控台](#) 中，選擇環境。
 - ii. 在環境清單中，選擇您要更新的環境名稱。
2. 選擇接下來 4 個步驟中的 1 個來更新您的環境。
 - a. 進行不需要環境部署的編輯。
 - i. 例如，變更描述。
選擇編輯。
 - ii. 填寫表單，然後選擇下一步。
 - iii. 檢閱您的編輯，然後選擇更新。
 - b. 僅更新中繼資料輸入。
 - i. 選擇動作，然後選擇更新。
 - ii. 填寫表單，然後選擇編輯。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。
 - c. 更新其環境範本的新次要版本。
 - i. 選擇動作，然後選擇更新次要。
 - ii. 填寫表單，然後選擇下一步。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。
 - d. 更新其環境範本的新主要版本。
 - i. 選擇動作，然後選擇更新主要。
 - ii. 填寫表單，然後選擇下一步。
 - iii. 填寫表單並選擇下一步，直到您到達檢閱頁面為止。
 - iv. 檢閱您的更新，然後選擇更新。

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\n ssm_another_parameter_value: \"update\"\n\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "1",
    "templateName": "pr-env-template"
  }
}
```

3. 檢閱 傳送的提取請求 AWS Proton。

- 如果您核准請求，則佈建正在進行中。
- 如果您拒絕請求，環境建立會取消。
- 如果提取請求逾時，環境建立不會完成。

4. 提供佈建狀態給 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

如果 deploymentStatus 為 IN_PROGRESS AWS Proton attempts 以取消部署，您可以嘗試取消環境更新部署。不保證成功取消。

當您取消更新部署時，會 AWS Proton 嘗試取消部署，如下列步驟所列。

使用 AWS 受管佈建時，會 AWS Proton 執行下列動作：

- 將部署狀態設定為 CANCELLING。
- 停止進行中的部署，並在 時刪除部署建立的任何新資源 IN_PROGRESS。
- 將部署狀態設定為 CANCELLED。
- 將資源的狀態還原為部署開始之前的狀態。

使用自我管理佈建時，會 AWS Proton 執行下列動作：

- 嘗試關閉提取請求，以防止合併對儲存庫的變更。
- CANCELLED 如果提取請求已成功關閉，則將部署狀態設定為 。

如需如何取消環境部署的指示，請參閱 AWS Proton API 參考中的 [CancelEnvironmentDeployment](#)。

您可以使用 主控台或 CLI 來取消正在進行的環境。

AWS 管理主控台

使用 主控台取消環境更新部署，如下列步驟所示。

1. 在 [AWS Proton 主控台](#) 的導覽窗格中，選擇環境。
2. 在環境清單中，選擇具有您要取消之部署更新的環境名稱。
3. 如果您的更新部署狀態為進行中，請在環境詳細資訊頁面中選擇動作，然後選擇取消部署。
4. 模態會提示您確認是否要取消。選擇取消部署。
5. 您的更新部署狀態設定為取消，然後取消以完成取消。

AWS CLI

使用 AWS Proton AWS CLI 取消 IN_PROGRESS 環境更新部署至新的次要版本 2。

用於此範例的範本中包含等待條件，因此取消會在更新部署成功之前開始。

執行下列命令來取消更新：

```
$ 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

您可以使用 AWS Proton 主控台或刪除 AWS Proton 環境 AWS CLI。

Note

您無法刪除具有任何相關聯元件的環境。若要刪除這類環境，您應該先刪除環境中執行的所有元件。如需元件的詳細資訊，請參閱 [元件](#)。

AWS 管理主控台

使用主控台刪除環境，如以下兩個選項所述。

在環境清單中。

1. 在 [AWS Proton 主控台](#) 中，選擇環境。
2. 在環境清單中，選取您要刪除之環境左側的選項按鈕。
3. 選擇動作，然後選擇刪除。
4. 模態會提示您確認刪除動作。
5. 遵循指示並選擇是，刪除。

在環境詳細資訊頁面中。

1. 在 [AWS Proton 主控台](#) 中，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，然後選擇刪除。
4. 模態會提示您確認要刪除。
5. 遵循指示並選擇是，刪除。

AWS CLI

使用 AWS CLI 刪除環境。

如果 服務或 服務執行個體部署到環境，請勿刪除環境。

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

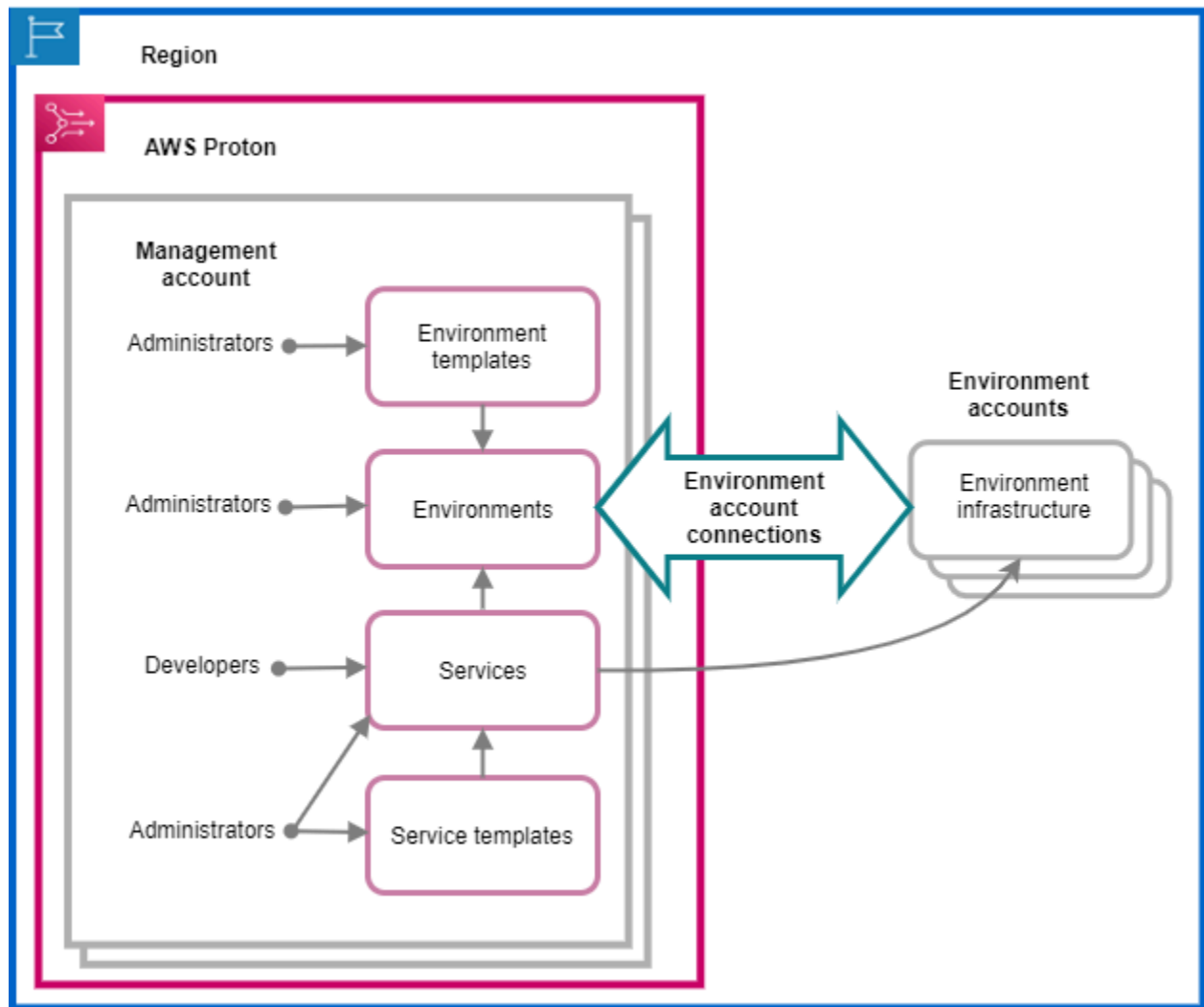
概觀

了解如何在一個帳戶中建立和管理 AWS Proton 環境，並在另一個帳戶中佈建其基礎設施資源。這有助於大規模提高可見性和效率。環境帳戶連線僅支援使用 CloudFormation 基礎設施做為程式碼的標準佈建。

Note

本主題中的資訊與使用AWS 受管佈建設定的環境有關。透過使用自我管理佈建設定的環境，AWS Proton 不會直接佈建您的基礎設施。反之，它會傳送提取請求 (PRs) 到您的儲存庫以進行佈建。您有責任確保您的自動化程式碼擔任正確的身分和角色。如需佈建方法的詳細資訊，請參閱 [the section called “佈建方法”](#)。

術語



透過 AWS Proton 環境帳戶連線，您可以從一個帳戶建立 AWS Proton 環境，並在另一個帳戶中佈建其基礎設施。

管理帳戶

您身為管理員的單一帳戶，會建立在另一個 AWS Proton 環境帳戶中佈建基礎設施資源的環境。

環境帳戶

當您在另一個帳戶中建立 AWS Proton 環境時，環境基礎設施佈建所在的帳戶。

環境帳戶連線

管理帳戶與環境帳戶之間的安全雙向連線。它會維護授權和許可，如以下各節所述。

當您在特定區域中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才能查看和使用環境帳戶連線。這表示在管理帳戶中建立 AWS Proton 的環境和在環境帳戶中佈建的環境基礎設施必須位於相同的區域。

環境帳戶連線考量事項

- 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 來檢視和管理環境帳戶連線客戶受管標籤。環境帳戶連線不會產生受 AWS 管標籤。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設施

若要從單一管理帳戶建立和佈建環境，請為您計劃建立的環境設定環境帳戶。

從環境帳戶開始並建立連線。

在環境帳戶中，建立範圍縮小為僅佈建環境基礎設施資源所需的許可 AWS Proton 的服務角色。如需詳細資訊，請參閱 [AWS Proton 使用 佈建的 服務角色 CloudFormation](#)。

然後，建立環境帳戶連線請求並將其傳送至您的管理帳戶。接受請求時，AWS Proton 可以使用在相關聯環境帳戶中允許環境資源佈建的相關聯 IAM 角色。

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請求。您無法從管理帳戶刪除環境帳戶連線。

如果您接受請求，AWS Proton 可以使用相關環境帳戶中允許資源佈建的相關聯 IAM 角色。

環境基礎設施資源會在相關聯的環境帳戶中佈建。您只能使用 AWS Proton APIs 從管理帳戶存取和管理環境及其基礎設施資源。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](#)。

拒絕請求之後，您就無法接受或使用遭拒絕的環境帳戶連線。

Note

您無法拒絕連線至環境的環境帳戶連線。若要拒絕環境帳戶連線，您必須先刪除相關聯的環境。

在環境帳戶中，存取佈建的基礎設施資源。

在環境帳戶中，您可以檢視和存取佈建的基礎設施資源。例如，您可以視需要使用 CloudFormation API 動作來監控和清除堆疊。您無法使用 AWS Proton API 動作來存取或管理用來佈建基礎設施資源 AWS Proton 的環境。

在環境帳戶中，您可以刪除在環境帳戶中建立的環境帳戶連線。您無法接受或拒絕它們。如果您刪除環境正在使用的環境帳戶連線 AWS Proton，AWS Proton 將無法管理環境基礎設施資源，直到環境帳戶和具名環境接受新的環境連線。您有責任清除在沒有環境連線的情況下保留的佈建資源。

使用 主控台或 CLI 來管理環境帳戶連線

您可以使用 主控台或 CLI 來建立和管理環境帳戶連線。

AWS 管理主控台

使用 主控台建立環境帳戶連線，並將請求傳送至管理帳戶，如後續步驟所示。

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. 在環境帳戶中的 [AWS Proton 主控台](#) 中，選擇導覽窗格中的環境帳戶連線。
3. 在環境帳戶連線頁面中，選擇請求連線。

Note


驗證環境帳戶連線頁面標題中列出的帳戶 ID。請確定它與您希望具名環境佈建的環境帳戶的帳戶 ID 相符。

4. 在請求連線頁面中：
 - a. 在連線至管理帳戶區段中，輸入您在步驟 1 中輸入的管理帳戶 ID 和環境名稱。
 - b. 在環境角色區段中，選擇新服務角色，並 AWS Proton 自動為您建立新的角色。或者，選取現有服務角色和您先前建立的服務角色名稱。
-  **Note**

AWS Proton 自動為您建立的角色具有廣泛的許可。建議您將角色範圍縮小為佈建環境基礎設施資源所需的許可。如需詳細資訊，請參閱[AWS Proton 使用 佈建的服務角色 CloudFormation](#)。
- c. (選用) 在標籤區段中，選擇新增標籤，為您的環境帳戶連線建立客戶受管標籤。
 - d. 選擇請求連線。
 5. 您的請求會在傳送至管理帳戶資料表的環境連線中顯示為待定，而模態可讓您了解如何接受來自管理帳戶的請求。

接受或拒絕環境帳戶連線請求。

1. 在管理帳戶中的 [AWS Proton 主控台](#) 中，選擇導覽窗格中的環境帳戶連線。
2. 在環境帳戶連線頁面的環境帳戶連線請求表中，選擇要接受或拒絕的環境連線請求。

 **Note**

驗證環境帳戶連線頁面標題中列出的帳戶 ID。請確定它符合與要拒絕的環境帳戶連線相關聯的管理帳戶的帳戶 ID。拒絕此環境帳戶連線後，您就無法接受或使用遭拒絕的環境帳戶連線。

3. 選擇拒絕或接受。
 - 如果您選取拒絕，狀態會從待定變更為拒絕。
 - 如果您選取接受，狀態會從待定變更為已連線。

刪除環境帳戶連線。

1. 在環境帳戶中的 [AWS Proton 主控台](#) 中，選擇導覽窗格中的環境帳戶連線。

Note

驗證環境帳戶連線頁面標題中列出的帳戶 ID。請確定它符合與要拒絕的環境帳戶連線相關聯的管理帳戶的帳戶 ID。刪除此環境帳戶連線後，AWS Proton 無法管理環境帳戶中的環境基礎設施資源。只有在環境帳戶的新環境帳戶連線，且管理帳戶接受具名環境之後，才能管理它。

2. 在環境帳戶連線頁面的傳送連線至管理帳戶請求區段中，選擇刪除。
3. 模態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線，如下列命令和回應所示。

Note

如果您拒絕此環境帳戶連線，您將無法接受或使用拒絕的環境帳戶連線。

如果您指定拒絕，狀態會從待定變更為拒絕。

如果您指定接受，狀態會從待定變更為已連線。

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。您可以取得或列出環境帳戶連線。

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已接受新的環境連線之前，將無法管理環境帳戶中的環境基礎設施資源。您有責任清除在沒有環境連線的情況下保留的佈建資源。

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"  
  }  
}
```

客戶受管環境

透過客戶受管環境，您可以使用已部署為 AWS Proton 環境的現有基礎設施，例如 VPC。使用客戶受管環境時，您可以在外部佈建自己的共用資源 AWS Proton。不過，您仍然可以允許在部署服務時 AWS Proton，使用相關的佈建輸出做為 AWS Proton 服務的輸入。如果輸出可以變更，AWS Proton 則可以接受更新。但是，由於佈建是在外部管理，因此 AWS Proton 無法直接變更環境 AWS Proton。

建立環境之後，您需負責提供相同的輸出給 AWS Proton，如果 AWS Proton 已建立環境，例如 Amazon ECS 叢集名稱或 Amazon VPC IDs。

使用此功能，您可以將 AWS Proton 服務資源從 AWS Proton 服務範本部署和更新至此環境。不過，環境本身不會透過 中的範本更新進行修改 AWS Proton。您有責任執行環境的更新，並在其中更新這些輸出 AWS Proton。

您可以在單一帳戶中擁有多個混合 AWS Proton 受管環境和客戶受管環境的環境。您也可以連結第二個帳戶，並使用主要帳戶中的 AWS Proton 範本，對該第二個連結帳戶中的環境和服務執行部署和更新。

如何使用客戶受管環境

管理員需要做的第一件事是註冊匯入的客戶受管環境範本。請勿在範本套件中提供資訊清單或基礎設施檔案。僅提供結構描述。

下面的結構描述概述了使用開放 API 格式的輸出清單，並從 CloudFormation 範本複寫輸出。

Important

輸出只允許輸入字串。

下列範例是對應 Fargate 範本之 CloudFormation 範本輸出區段的程式碼片段。

```
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 AWS Proton 匯入環境的結構描述類似如下。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"
  [...]
```

在註冊範本時，您表示此範本已匯入，並提供 bundle 的 Amazon S3 儲存貯體位置。在將 CloudFormation 範本放入草稿之前，會 AWS Proton 驗證結構描述僅包含 environment_input_type 且沒有範本參數。

您提供下列項目來建立匯入的環境。

- 部署時要使用的 IAM 角色。
- 具有所需輸出值的規格。

您可以使用與一般環境部署類似的 AWS CLI 程序，透過 主控台 或 提供兩者。

CodeBuild 佈建角色建立

CloudFormation 和 Terraform 等基礎設施即程式碼 (IaC) 工具需要許多不同類型的 AWS 資源的許可。例如，如果 IaC 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。將角色限制在所需的最低許可，是視為安全最佳實務。鑑於 AWS 資源的廣度，為 IaC 範本建立最低權限政策極具挑戰性，尤其是當這些範本管理的資源稍後可能會變更時。例如，在對受管範本的最新編輯中 AWS Proton，您可以新增 RDS 資料庫資源。

設定正確的許可有助於讓 IaC 部署順暢。AWS Proton CodeBuild Provisioning 會在位於客戶帳戶中的 CodeBuild 專案中執行任意客戶提供的 CLI 命令。一般而言，這些命令會使用 Infrastructure as Code (IaC) 工具建立和刪除基礎設施，例如 AWS CDK。當 AWS 資源部署其範本使用

CodeBuild Provisioning 時，AWS 將在由管理的 CodeBuild 專案中啟動組建 AWS。角色會傳遞至 CodeBuild，CodeBuild 會擔任該角色來執行命令。此角色稱為 CodeBuild 佈建角色，由客戶提供，並包含佈建基礎設施所需的許可。它只能由 CodeBuild 擔任，甚至 AWS Proton 無法擔任。

建立角色

CodeBuild 佈建角色可以在 IAM 主控台或中建立 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
```

這也會連接 `AWSProtonCodeBuildProvisioningBasicAccess`，其中包含 CodeBuild 服務執行組建所需的最低許可。

如果您偏好使用 主控台，請在建立角色時確保下列事項：

1. 對於信任的實體，選取 AWS 服務，然後選取 CodeBuild。
2. 在新增許可步驟中，選取您要連接 `AWSProtonCodeBuildProvisioningBasicAccess` 的任何其他政策。

管理員存取

如果您將 `AdministratorAccess` 政策連接到 CodeBuild 佈建角色，它將保證任何 IaC 範本不會因為缺少許可而失敗。這也表示可以建立環境範本或服務範本的任何人都可以執行管理員層級動作，即使該使用者不是管理員。AWS Proton 不建議 `AdministratorAccess` 搭配 CodeBuild 佈建角色使用。如果您決定 `AdministratorAccess` 搭配 CodeBuild 佈建角色使用，請在沙盒環境中執行此操作。

您可以在 IAM 主控台 `AdministratorAccess` 中使用 建立角色，或執行此命令：

```
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
```

建立最小範圍的角色

如果您想要建立具有最低許可的角色，有多種方法：

- 使用管理員許可進行部署，然後縮小角色的範圍。建議使用 [IAM Access Analyzer](#)。
- 使用 受管政策可讓您存取您計劃使用的服務。

AWS CDK

如果您使用 AWS CDK 搭配 AWS Proton，且已在每個環境帳戶/區域 cdk bootstrap 上執行，則已存在角色 cdk deploy。在此情況下，請將下列政策連接至 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 VPC

如果您決定在 [自訂 VPC](#) 中執行 CodeBuild，則需要 CodeBuild 角色的下列許可：

```
{
  "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](#) 受管政策，但其中包含您可能不需要的許可。若要使用 CLI 連接受管政策：

```

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 服務是服務範本的實例化，通常包括數個服務執行個體和管道。AWS Proton 服務執行個體是特定[環境中](#)服務範本的執行個體。服務範本是 AWS Proton 服務基礎設施和選用服務管道的完整定義。

部署服務執行個體之後，您可以藉由來源碼推送來提示 CI/CD 管道，或透過將服務更新為其服務範本的新版本來更新它們。當其服務範本的新版本可用時，會 AWS Proton 提示您，以便您可以將服務更新為新版本。當您的服務更新時，會 AWS Proton 重新部署服務和服務執行個體。

本章說明如何使用建立、檢視、更新和刪除操作來管理服務。如需詳細資訊，請參閱[AWS Proton 服務 API 參考](#)。

主題

- [建立服務](#)
- [檢視服務資料](#)
- [編輯服務](#)
- [刪除服務](#)
- [檢視服務執行個體資料](#)
- [更新服務執行個體](#)
- [更新服務管道](#)

建立服務

若要以開發人員身分部署應用程式 AWS Proton，您可以建立服務並提供下列輸入。

1. 平台團隊發佈 AWS Proton 的服務範本名稱。
2. 服務的名稱。
3. 您要部署的服務執行個體數量。
4. 您想要使用的環境選項。
5. 如果您使用的是包含服務管道的服務範本，則為程式碼儲存庫的連線（選用）。

服務中有哪些項目？

建立 AWS Proton 服務時，您可以從兩種不同類型的服務範本中選擇：

- 包含服務管道的服務範本（預設）。
- 不包含服務管道的服務範本。

建立服務時，必須至少建立一個服務執行個體。

服務執行個體和選用管道會與服務建立關聯。您只能在服務建立和刪除動作的情況下建立或刪除管道。若要了解如何從服務新增和移除執行個體，請參閱 [編輯服務](#)。

Note

您的環境是針對環境中 AWS 的 或自我管理的 provisioning. AWS Proton provisions 服務，使用與環境相同的佈建方法。建立或更新服務執行個體的開發人員看不到差異，而且兩者的體驗都相同。

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

服務範本的主要和次要版本皆可使用。使用主控台時，您可以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，您可以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。

- 範本的新版本 Recommended 會在平台團隊成員核准後立即變成。這表示使用該版本建立新服務，系統會提示您將現有服務更新為新版本。
- 透過 AWS Proton，平台團隊可以自動將服務執行個體更新為服務範本的新次要版本。次要版本必須回溯相容。
- 由於主要版本需要您在更新程序中提供新的輸入，因此您需要將服務更新為其服務範本的主要版本。主要版本無法回溯相容。

建立服務

下列程序說明如何使用 AWS Proton 主控台或 AWS CLI 來建立有或沒有服務管道的服務。

AWS 管理主控台

建立服務，如下列主控台步驟所示。

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，然後選擇設定。

當您不想使用已啟用的管道時，請選擇標記 排除您服務的管道的範本。

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. 針對 CodeStar Connection，從清單中選擇您的連線。
 - b. 針對儲存庫 ID，從清單中選擇來源碼儲存庫的名稱。
 - c. 針對分支名稱，從清單中選擇來源碼儲存庫分支的名稱。
7. (選用) 在標籤區段中，選擇新增標籤，然後輸入索引鍵和值以建立客戶受管標籤。
8. 選擇下一步。
9. 在設定自訂設定頁面的 服務執行個體 區段的新執行個體 區段中。您必須輸入required參數的值。您可以輸入optional參數的值，或在指定時使用預設值。
10. 在管道輸入區段中，您必須輸入required參數的值。您可以輸入optional參數的值，或在指定時使用預設值。
11. 選擇下一步並檢閱您的輸入。
12. 選擇建立。

檢視服務詳細資訊和狀態，以及服務的 AWS 受管標籤和客戶受管標籤。

13. 在導覽窗格中，選擇服務。

新頁面會顯示您的服務清單，以及狀態和其他服務詳細資訊。

AWS CLI

當您使用時 AWS CLI，您可以在 YAML 格式spec的檔案中指定服務輸入，`.aws-proton/service.yaml`位於您的來源碼目錄中。

您可以使用 CLI `get-service-template-minor-version` 命令來檢視您在規格檔案中為提供值所需的結構描述和選用參數。

如果您想要使用具有的服務範本 `pipelineProvisioning: "CUSTOMER_MANAGED"`，請不要在規格中包含 `pipeline:區段-repository-id`，也不要包含 `create-service` 命令中的 `-repository-connection-arn`、`-branch-name` 參數。

使用服務管道建立服務，如下列 CLI 步驟所示。

1. 設定管道的[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/AWS  
ProtonServiceRole"
```

2. 下列清單顯示以服務範本結構描述為基礎的範例規格，其中包含服務管道和執行個體輸入。

規格：

```
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"
```

使用管道建立服務，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"
  }
}
```

建立沒有服務管道的服務，如下列 CLI 範例命令和回應所示。

以下顯示不包含服務管道輸入的範例規格。

規格：

```
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道的情況下建立服務，請提供的路徑，`spec.yaml`而且不會包含儲存庫參數，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 管理主控台

使用 [AWS Proton 主控台](#) 列出和檢視服務詳細資訊，如下列步驟所示。

1. 若要檢視服務清單，請在導覽窗格中選擇服務。
2. 若要檢視詳細資訊，請選擇服務的名稱。

檢視您的服務詳細資訊。

AWS CLI

檢視具有服務管道的服務詳細資訊，如下列 CLI 範例命令和回應所示。

命令：

```
$ aws proton get-service \  

```

```
--name "simple-svc"
```

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"
  }
}
```

檢視沒有服務管道的服務詳細資訊，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\nenvironment: 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 服務進行下列編輯。

- 編輯服務描述。
- 透過新增和移除服務執行個體來編輯服務。

編輯服務描述

您可以使用 主控台 或 AWS CLI 編輯服務描述。

AWS 管理主控台

使用主控台編輯服務，如下列步驟所述。

在 服務清單 中。

1. 在 [AWS Proton 主控台](#) 中，選擇 服務。
2. 在服務清單中，選擇您要更新之服務左側的選項按鈕。

3. 選擇編輯。
4. 在設定服務頁面中，填寫表單並選擇下一步。
5. 在設定自訂設定頁面中，選擇下一步。
6. 檢閱您的編輯，然後選擇儲存變更。

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，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，填寫表單並選擇下一步。
5. 在設定自訂設定頁面中，填寫表單並選擇下一步。
6. 檢閱您的編輯，然後選擇儲存變更。

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，您可以透過提交編輯的規格來新增或刪除服務執行個體。成功請求必須符合下列條件：

- 當您提交編輯請求時，您的服務和管道尚未進行編輯或刪除。
- 您編輯的規格不包含修改服務管道的編輯，或編輯尚未刪除的現有服務執行個體。
- 您編輯的規格不會移除任何具有連接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，將其從服務執行個體中分離。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 狀態的服務執行個體。當您請求服務編輯時，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，即使這些資源無法從主控台或 `aws` 中看見 AWS CLI。檢查您的刪除失敗的執行個體基礎設施資源並進行清除，讓 AWS Proton 可以為您移除它們。

如需服務的執行個體配額，請參閱 [AWS Proton 配額](#)。建立服務之後，您也必須為服務維護至少 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和要新增或移除之執行個體的計數。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 AWS CLI 來新增或移除服務執行個體

AWS 管理主控台

編輯您的服務，使用主控台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，選擇服務。
2. 選取您要編輯的服務。
3. 選擇編輯。
4. (選用) 在設定服務頁面上，編輯服務名稱或描述，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. 選擇下一步。
7. 檢閱您的更新，然後選擇儲存變更。
8. 模式會要求您驗證服務執行個體的刪除。遵循指示並選擇是，刪除。
9. 在服務詳細資訊頁面上，檢視服務的狀態詳細資訊。

AWS CLI

新增和刪除已編輯的服務執行個體 **spec**，如下列 AWS CLI 範例命令和回應所示。

當您使用 CLI 時，您的 **spec** 必須排除要刪除的服務執行個體，並同時包含要新增的服務執行個體，以及您尚未標記為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。在上一個範例中使用此規格來編輯服務描述。

規格：

```
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 CLI `list-service-instances` 命令和回應會在新增或刪除服務執行個體之前顯示作用中的執行個體。

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"
    }
  ]
}
```

下列清單顯示已編輯spec用來刪除和新增執行個體的範例。my-instance 已移除名為 的現有執行個體，並新增名為 yet-another-instance 的新執行個體。

規格：

```
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 `spec` 中，您可以使用 `"${Proton::CURRENT_VAL}"` 來指示要從原始 保留哪些參數值 `spec`。 `get-service` 使用 檢視 `spec` 服務的原始 ，如中所述 [檢視服務資料](#)。

下列清單顯示如何使用 來 `"${Proton::CURRENT_VAL}"` 確保您的 `spec` 不包含現有服務執行個體要保留的參數值變更。

規格：

```

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"

```

下一個清單顯示編輯服務的 CLI 命令和回應。

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` 命令和回應會確認 `my-instance` 已移除名為 `my-instance` 的現有執行個體，並新增名為 `yet-another-instance` 的新執行個體。

命令：

```
$ aws proton list-service-instances \
  --service-name "MySimpleService"
```

回應：

```
{
  "serviceInstances": [
    {
      "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/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"
    }
]
}

```

當您新增或移除服務執行個體時會發生什麼情況

在您提交服務編輯以刪除和新增服務執行個體之後，會 AWS Proton 採取下列動作。

- 將服務設定為 UPDATE_IN_PROGRESS。
- 如果服務有管道，會將其狀態設定為 IN_PROGRESS 並封鎖管道動作。
- 將要刪除的任何服務執行個體設定為 DELETE_IN_PROGRESS。
- 封鎖服務動作。
- 在標記為刪除的服務執行個體上封鎖動作。
- 建立新的服務執行個體。
- 刪除您列出要刪除的執行個體。
- 嘗試移除刪除失敗的執行個體。
- 新增和刪除完成後，會重新佈建服務管道（如果有的話），將您的服務設定為 ACTIVE 並啟用服務和管道動作。

AWS Proton 會嘗試修復故障模式，如下所示。

- 如果一或多個服務執行個體無法建立，會 AWS Proton 嘗試取消佈建所有新建立的服務執行個體，並將還原spec為先前的狀態。它不會刪除任何服務執行個體，也不會以任何方式修改管道。
- 如果一個或多個服務執行個體無法刪除，請 AWS Proton 重新佈建管道，而沒有已刪除的執行個體。spec 已更新以包含新增的執行個體，並排除標記為刪除的執行個體。
- 如果管道佈建失敗，則不會嘗試回復，並且服務和管道都會反映失敗的更新狀態。

標記和服務編輯

當您新增服務執行個體做為服務編輯的一部分時，AWS 受管標籤會傳播至，並針對新執行個體和佈建的資源自動建立。如果您建立新的標籤，這些標籤只會套用至新的執行個體。現有的服務客戶受管標籤也會傳播到新的執行個體。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，刪除具有其執行個體和管道 AWS Proton 的服務 AWS CLI。

您無法刪除具有任何具有已連接元件之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接的元件，以將其從服務執行個體中分離。如需元件的詳細資訊，請參閱[元件](#)。

AWS 管理主控台

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，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. 模式會提示您確認刪除動作。
5. 遵循指示並選擇是，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"  
  }  
}
```

檢視服務執行個體資料

了解如何檢視 AWS Proton 服務執行個體詳細資訊。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](#) 和刪除動作的內容中 [建立](#) 或 [刪除](#) 執行個體。若要了解如何從服務新增和移除執行個體，請參閱 [編輯服務](#)。

AWS 管理主控台

使用 [AWS Proton 主控台](#) 列出和檢視服務執行個體詳細資訊，如下列步驟所示。

1. 若要檢視服務執行個體的清單，請在導覽窗格中選擇服務執行個體。
2. 若要檢視詳細資訊，請選擇服務執行個體的名稱。

檢視您的服務執行個體詳細資訊。

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:
0la\n   my_sample_service_instance_required_input: Ciao\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "0",
    "templateName": "svc-simple"
  }
}
```

更新服務執行個體

了解如何更新 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](#)和刪除動作的內容中建立[或刪除](#)執行個體。若要了解如何從服務新增和移除執行個體，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「更新至最新的次要版本」和「更新至最新的主要版本」動作，這些動作會從服務執行個體詳細資訊頁面的動作中下拉。

NONE

在此模式中，不會發生部署。只會更新請求的中繼資料參數。

CURRENT_VERSION

在此模式中，會使用您提供的新規格來部署和更新服務執行個體。只會更新請求的參數。使用此時，請勿包含次要或主要版本參數`deployment-type`。

MINOR_VERSION

在此模式中，服務執行個體會依預設使用中目前主要版本的已發佈建議（最新）次要版本進行部署和更新。您也可以指定目前使用中主要版本的不同次要版本。

MAJOR_VERSION

在此模式中，預設會使用已發佈、建議（最新）的目前範本主要和次要版本來部署和更新服務執行個體。您也可以指定高於使用中主要版本的不同主要版本，以及次要版本（選用）。

如果 `deploymentStatus` 是 `.IN_PROGRESS`，AWS Proton 嘗試以取消部署，您可以嘗試取消服務執行個體更新部署。不保證成功取消。

當您取消更新部署時，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`。
- 停止進行中的部署，並在時刪除部署建立的任何新資源 `IN_PROGRESS`。
- 將部署狀態設定為 `CANCELLED`。
- 將資源的狀態還原為部署開始之前的狀態。

如需取消服務執行個體部署的詳細資訊，請參閱 AWS Proton API 參考中的 [CancelServiceInstanceDeployment](#)。

使用 主控台 或 AWS CLI 進行更新或取消更新部署。

AWS 管理主控台

請依照下列步驟，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. 在服務執行個體清單中，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、編輯以更新規格或動作，然後更新至最新的次要版本，或更新至最新的主要版本。
4. 填寫每個表單，然後選擇下一步，直到您到達檢閱頁面為止。
5. 檢閱您的編輯，然後選擇更新。

AWS CLI

將服務執行個體更新為新的次要版本，如 CLI 範例命令和回應所示。

當您使用修改過的更新服務執行個體時 `spec`，如果值存在於 `spec` 中，您可以使用 `"${Proton::CURRENT_VAL}"` 來指出要從原始保留的參數值 `spec`。使用 `get-service` 檢視服務執行個體 `spec` 的原始，如中所述 [檢視服務資料](#)。

下列範例顯示如何在 `"${Proton::CURRENT_VAL}"` 中使用 `spec`。

規格：

```

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 管理主控台

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. 在服務執行個體清單中，選擇具有您要取消之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. 模態會要求您確認取消。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。

用於此範例的範本中包含等待條件，因此取消會在更新部署成功之前開始。

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:\nmy_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\nmy_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。

服務管道屬於服務。您只能在服務建立和刪除動作的情況下[建立或刪除](#)管道。

更新服務管道有四種模式，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道和更新至建議的版本。

NONE

在此模式中，不會發生部署。只會更新請求的中繼資料參數。

CURRENT_VERSION

在此模式中，會使用您提供的新規格來部署和更新服務管道。只會更新請求的參數。使用此時，請勿包含次要或主要版本參數`deployment-type`。

MINOR_VERSION

在此模式中，服務管道會依預設使用中目前主要版本的已發佈、建議（最新）次要版本進行部署和更新。您也可以指定目前使用中主要版本的不同次要版本。

MAJOR_VERSION

在此模式中，服務管道預設會部署並更新目前範本的已發佈、建議（最新）主要和次要版本。您也可以指定高於使用中主要版本的不同主要版本，以及次要版本（選用）。

如果 `deploymentStatus` 是 `.IN_PROGRESS` AWS Proton attempts 以取消部署，您可以嘗試取消服務管道更新部署。不保證成功取消。

當您取消更新部署時，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`。
- 停止進行中的部署，並在時刪除部署建立的任何新資源 `IN_PROGRESS`。
- 將部署狀態設定為 `CANCELLED`。
- 將資源的狀態還原為部署開始之前的狀態。

如需取消服務管道部署的詳細資訊，請參閱 AWS Proton API 參考中的 [CancelServicePipelineDeployment](#)。

使用 主控台或 AWS CLI 進行更新或取消更新部署。

AWS 管理主控台

使用主控台更新服務管道，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，選擇您要更新管道的服務名稱。
3. 服務詳細資訊頁面上有兩個索引標籤：概觀和管道。選擇管道。
4. 如果您想要更新規格，請選擇編輯管道並填寫每個表單，然後選擇下一步，直到您完成最終表單，然後選擇更新管道。

如果您想要更新至新版本，且有資訊圖示指出管道範本有新版本可用，請選擇新範本版本的名稱。

- a. 選擇更新為建議的版本。
- b. 填寫每個表單，然後選擇下一步，直到您完成最終表單，然後選擇更新。

AWS CLI

將服務管道更新為新的次要版本，如下列 CLI 範例命令和回應所示。

當您使用修改過的 更新服務管道時spec，如果值存在於 中spec，您可以使用 `"${Proton::CURRENT_VAL}"` 來指出要從原始 保留的參數值spec。get-service 使用 檢視服務管道spec的原始，如中所述[檢視服務資料](#)。

下列範例顯示如何在 `"${Proton::CURRENT_VAL}"` 中使用 spec。

規格：

```
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:
\n\"123\"\n\ninstances:\n - name: \"my-instance\"\n   environment: \"MySimpleEnv
\n\n   spec:\n     my_sample_service_instance_optional_input: \"def
\n\n     my_sample_service_instance_required_input: \"456\"\n   - name:
\n\"my-other-instance\"\n   environment: \"MySimpleEnv\"\n   spec:\n
my_sample_service_instance_required_input: \"789\"\n\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:
\n\"123\"\n\ninstances:\n - name: \"instance-one\"\n  environment: \"simple-
env\"\n  spec:\n    my_sample_service_instance_optional_input: \"def
\n\n my_sample_service_instance_required_input: \"456\"\n - name:
\n\"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:
\n\"123\"\n\ninstances:\n - name: \"instance-one\"\n  environment: \"simple-
env\"\n  spec:\n    my_sample_service_instance_optional_input: \"def
\n\n my_sample_service_instance_required_input: \"456\"\n - name:
\n\"my-other-instance\"\n  environment: \"simple-env\"\n  spec:\n
my_sample_service_instance_required_input: \"789\"\n",
    "status": "ACTIVE",
    "templateName": "svc-simple"
  }
}
```

AWS 管理主控台

使用主控台取消服務管道部署，如下列步驟所示。

1. 在 [AWS Proton 主控台](#) 的導覽窗格中，選擇服務。
2. 在服務清單中，選擇具有您要取消之部署更新管道的服務名稱。
3. 在服務詳細資訊頁面中，選擇管道索引標籤。
4. 如果您的更新部署狀態為進行中，請在服務管道詳細資訊頁面中選擇取消部署。
5. 模態會要求您確認取消。選擇取消部署。
6. 您的更新部署狀態設定為取消，然後取消以完成取消。

AWS CLI

取消次要版本 2 的 IN_PROGRESS 服務管道部署更新，如下列 CLI 範例命令和回應所示。

用於此範例的範本中包含等待條件，因此取消會在更新部署成功之前開始。

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

元件是一種 AWS Proton 資源。它們可為服務範本增加靈活性。元件為平台團隊提供擴展核心基礎設施模式的機制，並定義保護機制，讓開發人員能夠管理其應用程式基礎設施的各個層面。

在 AWS Proton 管理員中，定義用於開發團隊和應用程式的標準基礎設施。不過，開發團隊可能需要針對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 資料表。這些應用程式特定的資源可能會經常變更，特別是在應用程式的早期開發期間。在管理員撰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範本，而不會增加真正的管理員值。讓應用程式開發人員為其應用程式編寫範本的替代方案也不理想，因為它會消除管理員標準化主要架構元件的能力，例如 AWS Fargate 任務。這是元件進來的位置。

透過 元件，開發人員可以將補充資源新增至其應用程式，超越環境和服務範本中定義的管理員。然後，開發人員將元件連接到服務執行個體。AWS Proton 佈建元件定義的基礎設施資源，就像為環境和服務執行個體佈建資源一樣。

元件可以讀取服務執行個體輸入，並將輸出提供給服務執行個體，以獲得完全整合的體驗。例如，如果元件新增供服務執行個體使用的 Amazon Simple Storage Service (Amazon S3) 儲存貯體，則元件範本可以將環境和服務執行個體名稱納入命名儲存貯體的考量。當 AWS Proton 轉譯服務範本以佈建服務執行個體時，服務執行個體可以參考儲存貯體並使用它。

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設施的基礎設施即程式碼 (IaC) 檔案直接傳遞至 AWS Proton API 或主控台。這與您在範本套件中定義 IaC 並將套件註冊為範本資源的環境或服務不同，然後使用範本資源來建立環境或服務。

Note

直接定義的元件可讓開發人員定義額外的基礎設施並進行佈建。使用相同的 AWS Identity and Access Management (IAM) 角色 AWS Proton 佈建在相同環境中執行的所有直接定義的元件。

管理員可以透過兩種方式控制開發人員可以使用元件執行的操作：

- 支援的元件來源 – 管理員可以允許根據服務範本版本的屬性，將元件連接至 AWS Proton 服務執行個體。根據預設，開發人員無法將元件連接至服務執行個體。

如需此屬性的詳細資訊，請參閱《API AWS Proton 參考》中 [CreateServiceTemplateVersion](#) API 動作的 [supportedComponentSources](#) 參數。

Note

當您使用範本同步時，當您在儲存庫中遞交服務範本套件的變更時，會隱含地 AWS Proton 建立服務範本版本。在此情況下，您可以在與每個服務範本主要版本相關聯的檔案中指定此屬性，而不是在建立服務範本版本期間指定支援的元件來源。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。當 佈建由環境中直接定義的元件所定義的基礎設施時，會 AWS Proton 擔任此角色。因此，元件角色會縮小 基礎設施的範圍，開發人員可以使用環境中直接定義的元件來新增這些基礎設施。如果沒有元件角色，開發人員無法在環境中建立直接定義的元件。

如需指派元件角色的詳細資訊，請參閱《API AWS Proton 參考》中 [CreateEnvironment](#) API 動作的 [componentRoleArn](#) 參數。

Note

[自我管理佈建](#) 環境不會使用元件角色。

主題

- [元件與其他 AWS Proton 資源相比如何？](#)
- [AWS Proton 主控台](#)中的元件
- [AWS Proton API 和 中的元件 AWS CLI](#)
- [元件常見問答集](#)
- [元件狀態](#)
- [元件基礎設施做為程式碼檔案](#)
- [元件 CloudFormation 範例](#)

元件與其他 AWS Proton 資源相比如何？

在許多方面，元件與其他 AWS Proton 資源類似。其基礎設施在 [IaC 範本檔案](#)中定義，以 CloudFormation YAML 或 Terraform HCL 格式撰寫。AWS Proton 可以使用 [AWS受管佈建](#)或 [自我管理佈建](#)來佈建元件基礎設施。

不過，元件與其他 AWS Proton 資源有幾種不同：

- 分離狀態 – 元件的設計是要連接到服務執行個體並擴展其基礎設施，但也可以處於分離狀態，其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”](#)。
- 無結構描述 – 元件沒有與[範本套件](#)類似的關聯結構描述。元件輸入由服務定義。當元件連接到服務執行個體時，可能會耗用輸入。
- 沒有客戶受管元件 – AWS Proton 一律為您佈建元件基礎設施。沒有自己的資源版本的元件。如需客戶受管環境的詳細資訊，請參閱 [the section called “建立”](#)。
- 無範本資源 – 直接定義的元件沒有與環境和服務範本類似的關聯範本資源。您可以將 IaC 範本檔案直接提供給元件。同樣地，您直接提供資訊清單來定義範本語言和轉譯引擎，以佈建元件的基礎設施。您編寫範本檔案和資訊清單的方式類似於編寫[範本套件](#)。不過，透過直接定義的元件，您不需要在特定位置將 IaC 檔案儲存為套件，而且您不需要在 IaC 檔案 AWS Proton 外的 中建立範本資源。
- 無 CodeBuild 型佈建 – 您無法使用自己的自訂佈建指令碼佈建直接定義的元件，稱為 CodeBuild 型佈建。如需詳細資訊，請參閱[the section called “CodeBuild 佈建”](#)。

AWS Proton 主控台下的元件

使用 AWS Proton 主控台來建立、更新、檢視和使用 AWS Proton 元件。

下列主控台頁面與元件相關。我們包含頂層主控台頁面的直接連結。

- [元件](#) – 檢視您 AWS 帳戶中的元件清單。您可以建立新的元件，並更新或刪除現有的元件。在清單上選擇元件名稱以檢視其詳細資訊頁面。

環境詳細資訊和服務執行個體詳細資訊頁面上也存在類似的清單。這些清單只會顯示與正在檢視的資源相關聯的元件。當您從其中一個清單建立元件時，請在建立元件頁面上 AWS Proton 預先選取相關聯的環境。

- 元件詳細資訊 – 若要檢視元件詳細資訊頁面，請在元件清單上選擇[元件](#)名稱。

在詳細資訊頁面上，檢視元件詳細資訊和狀態，並更新或刪除元件。檢視和管理輸出清單（例如，佈建的資源 ARNs）、佈建的 CloudFormation 堆疊和指派的標籤。

- [建立元件](#) – 建立元件。輸入元件名稱和描述、選擇相關聯的資源、指定元件來源 IaC 檔案，以及指派標籤。
- 更新元件 – 若要更新元件，請在元件清單中選取[元件](#)，然後在動作功能表中選擇更新元件。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訊。您無法更新元件名稱。您也可以選擇是否在成功更新後重新部署元件。

- 設定環境 – 當您建立或更新環境時，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，並提供佈建元件的許可。
- 建立新的服務範本版本 – 當您建立服務範本版本時，您可以為範本版本指定支援的元件來源。這可控制根據此範本版本將元件連接至 服務執行個體的功能。

AWS Proton API 和 中的元件 AWS CLI

使用 AWS Proton API 或 AWS CLI 來建立、更新、檢視和使用 AWS Proton 元件。

下列 API 動作會直接管理 AWS Proton 元件資源。

- [CreateComponent](#) – 建立 AWS Proton 元件。
- [DeleteComponent](#) – 刪除元件 AWS Proton 。
- [GetComponent](#) – 取得元件的詳細資訊。
- [ListComponentOutputs](#) – 取得元件基礎設施做為程式碼 (IaC) 輸出的清單。
- [ListComponentProvisionedResources](#) – 列出具有詳細資訊之元件的佈建資源。
- [ListComponents](#) – 列出包含摘要資料的元件。您可以依環境、服務或單一服務執行個體篩選結果清單。

下列其他 AWS Proton 資源的 API 動作有一些與元件相關的功能。

- [CreateEnvironment](#)、[UpdateEnvironment](#) – `componentRoleArn`用於指定在此環境中佈建直接定義元件 AWS Proton 時使用的 IAM 服務角色的 Amazon Resource Name (ARN)。它決定直接定義的元件可以佈建的基礎設施範圍。
- [CreateServiceTemplateVersion](#) – `supportedComponentSources`用來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連接到服務執行個體。

元件常見問答集

元件的生命週期為何？

元件可以處於連接或分離狀態。它們旨在連接到服務執行個體，並在大部分時間增強其基礎設施。分離的元件處於轉換狀態，可讓您刪除元件或以受控制且安全的方式將其連接至另一個服務執行個體。如需詳細資訊，請參閱[the section called “元件狀態”](#)。

為什麼我無法刪除連接的元件？

解決方案：若要刪除連接的元件，請更新元件以將其從服務執行個體分離、驗證服務執行個體穩定性，然後刪除元件。

為什麼需要此項目？連接的元件提供應用程式執行其執行時間函數所需的額外基礎設施。服務執行個體可能正在使用元件輸出來偵測和使用此基礎設施的資源。刪除元件，進而移除其基礎設施資源，可能會對連接的服務執行個體造成干擾。

作為額外的安全措施，AWS Proton 會要求您更新元件並將其從服務執行個體中分離，然後才能將其刪除。然後，您可以驗證您的服務執行個體，以確保其繼續部署並正常運作。如果您偵測到問題，可以快速將元件重新連接到服務執行個體，然後努力修正問題。當您確信您的服務執行個體沒有對元件的任何相依性時，您可以安全地刪除元件。

為什麼我無法直接變更元件連接的服務執行個體？

解決方案：若要變更附件，請更新元件以將其從服務執行個體分離、驗證元件和服務執行個體穩定性，然後將元件連接至新的服務執行個體。

為什麼需要此項目？元件旨在連接至服務執行個體。您的元件可能會使用服務執行個體輸入來進行基礎設施資源命名和組態。變更連接的服務執行個體可能會對元件造成干擾（除了可能中斷服務執行個體之外，如先前常見問答集所述，[為什麼我無法刪除連接的元件？](#)）。例如，這可能會導致重新命名，甚至取代元件 IaC 範本中定義的資源。

作為額外的安全措施，AWS Proton 會要求您更新元件並將其從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，您可以在將元件連接到新的服務執行個體之前，驗證元件和服務執行個體的穩定性。

元件狀態

AWS Proton 元件可以處於兩種基本上不同的狀態：

- 已連接 – 元件已連接至服務執行個體。它定義了支援服務執行個體執行時間功能的基礎設施。元件使用開發人員定義的基礎設施，擴展環境和服務範本中定義的基礎設施。

典型元件在其生命週期的大部分有用部分都處於連接狀態。

- 分離 – 元件與 AWS Proton 環境相關聯，不會連接到環境中的任何服務執行個體。

這是將元件生命週期延長到超過單一服務執行個體轉換狀態。

下表提供不同元件狀態的頂層比較。

	Attached	Detached
狀態的主要用途	擴展服務執行個體的基礎設施。	在服務執行個體附件之間維護元件的基礎設施。
與相關聯	服務執行個體和環境	環境
關鍵特定屬性	<ul style="list-style-type: none"> • 服務名稱 • 服務執行個體名稱 • Spec (規格) 	<ul style="list-style-type: none"> • 環境名稱
可以刪除	× 否	✓ 是
可以更新到另一個服務執行個體	× 否	✓ 是
可以讀取輸入	✓ 是	× 否

元件的主要目的是連接到服務執行個體，並使用其他資源擴展其基礎設施。連接的元件可以根據規格從服務執行個體讀取輸入。您無法直接刪除元件，或將其連接至不同的服務執行個體。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，將其從服務執行個體中分離。

若要維護元件的基礎設施超過單一服務執行個體的生命週期，您可以更新元件，並透過移除服務和服務執行個體名稱將其從服務執行個體分離。此分離狀態為轉換狀態。元件沒有輸入。其基礎設施會保持佈建狀態，您可以進行更新。您可以刪除元件在連接時與其相關聯的資源（服務執行個體、服務）。您可以刪除元件，或將其更新為再次連接到服務執行個體。

元件基礎設施做為程式碼檔案

元件基礎設施即程式碼 (IaC) 檔案類似於其他 AWS Proton 資源的元件基礎設施。在此處了解元件特有的一些詳細資訊。如需為撰寫 IaC 檔案的完整資訊 AWS Proton，請參閱 [範本撰寫和套件](#)。

搭配元件使用參數

AWS Proton 參數命名空間包含一些參數，服務 IaC 檔案可以參考這些參數，以取得相關聯的元件名稱和輸出。命名空間也包含元件 IaC 檔案可以參考的參數，以便從與元件相關聯的環境、服務和服務執行個體取得輸入、輸出和資源值。

元件沒有自己的輸入，它會從其連接的服務執行個體取得其輸入。元件也可以讀取環境輸出。

如需在元件和相關聯的服務 IaC 檔案中使用參數的詳細資訊，請參閱 [the section called “元件 CloudFormation IaC 參數”](#)。如需 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 “Parameters”](#)。

編寫強大的 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具有連接的元件。請參閱 API AWS Proton 參考中 [CreateServiceTemplateVersion](#) API 動作的 [supportedComponentSources](#) 參數。不過，對於任何未來的服務執行個體，建立執行個體的人員會決定是否將元件連接至執行個體，以及（如果是直接定義的元件）撰寫元件 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，您無法保證元件會連接到服務執行個體。您也無法保證特定元件輸出名稱的存在，或這些輸出值的有效性和安全性。

AWS Proton 和 Jinja 語法可協助您解決這些問題，並撰寫強大的服務範本，以下列方式進行轉譯而不會失敗：

- AWS Proton 參數篩選條件 – 當您參考元件輸出屬性時，您可以使用參數篩選條件，也就是驗證、篩選和格式化參數值的修飾詞。如需詳細資訊和範例，請參閱 [the section called “CloudFormation 參數篩選條件”](#)。
- 單一屬性預設值 – 當您參考元件的單一資源或輸出屬性時，您可以保證使用 default 篩選條件來轉譯您的服務範本不會失敗，無論是否有預設值。如果參考的元件或特定輸出參數不存在，則會改為轉譯預設值（或空白字串，如果您尚未指定預設值），且轉譯成功。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元件的命名空間 `.default` 部分與 `default` 篩選條件混淆，該篩選條件會在參考屬性不存在時提供預設值。

- 整個物件參考 – 當您參考整個元件或元件輸出的集合時，會 AWS Proton 傳回空物件 `{}`，因此保證轉譯您的服務範本不會失敗。您不需要使用任何篩選條件。請務必在可以取得空物件的內容中進行參考，或使用 `{{ if .. }}` 條件來測試空物件。

範例：

- `{{ service_instance.components.default }}`
- `{{ service_instance.components.default.outputs }}`

元件 CloudFormation 範例

以下是 AWS Proton 直接定義元件的完整範例，以及如何在 AWS Proton 服務中使用它。元件會佈建 Amazon Simple Storage Service (Amazon S3) 儲存貯體和相關的存取政策。服務執行個體可以參考此儲存貯體並使用它。儲存貯體名稱是以環境、服務、服務執行個體和元件的名稱為基礎，這表示儲存貯體會與延伸特定服務執行個體之元件範本的特定執行個體結合。開發人員可以根據此元件範本建立多個元件，為不同的服務執行個體和功能需求佈建 Amazon S3 儲存貯體。

此範例涵蓋撰寫各種必要的 CloudFormation 基礎設施做為程式碼 (IaC) 檔案，以及建立必要的 AWS Identity and Access Management (IAM) 角色。此範例透過擁有人員角色來將步驟分組。

管理員步驟

讓開發人員搭配 服務使用元件

1. 建立 AWS Identity and Access Management (IAM) 角色，以縮小直接定義在環境中執行的元件可以佈建的資源範圍。稍後會 AWS Proton 擔任此角色，以在環境中佈建直接定義的元件。

在此範例中，請使用下列政策：

Example 直接定義的元件角色

JSON

```
{
  "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"
    }
  }
}
]
}

```

- 當您建立或更新環境時，請提供您在上一個步驟中建立的角色。在 AWS Proton 主控台中，於設定環境頁面上指定元件角色。如果您使用的是 AWS Proton API 或 AWS CLI，請指定 [CreateEnvironment](#) 或 [UpdateEnvironment](#) API 動作 `componentRoleArn` 的。
- 建立服務範本，其參照連接至服務執行個體的直接定義元件。

此範例說明如何撰寫強大的服務範本，如果元件未連接到服務執行個體，則不會中斷。

Example 使用元件的服務 CloudFormation IaC 檔案

```

# 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. 建立新的服務範本次要版本，將直接定義的元件宣告為支援的元件。
 - Amazon S3 中的範本套件 – 在 AWS Proton 主控台中，當您建立服務範本版本時，針對支援的元件來源，選擇直接定義。如果您使用的是 AWS Proton API 或 AWS CLI，請在 [CreateServiceTemplateVersion](#) 或 [UpdateServiceTemplateVersion](#) API 動作的 `supportedComponentSources` 參數 `DIRECTLY_DEFINED` 中指定。
 - 範本同步 – 將變更遞交至您的服務範本套件儲存庫，您可以在主要版本目錄中 `supported_component_sources:` 的 `.template-registration.yaml` 檔案中將指定 `DIRECTLY_DEFINED` 為項目。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. 發佈新的服務範本次要版本。如需詳細資訊，請參閱 [the section called “發布”](#)。
6. 請務必允許使用此服務範本之開發人員的 IAM 角色 `proton:CreateComponent` 中的。

開發人員步驟

將直接定義的元件與服務執行個體搭配使用

1. 建立使用管理員使用元件支援建立之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 元件 CloudFormation IaC 檔案

```
# 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 AWS Proton API 或 AWS CLI，請撰寫元件的資訊清單檔案。

Example 直接定義的元件資訊清單

```

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

```


4. 建立直接定義的 component. AWS Proton assume 元件角色，由管理員定義來佈建元件。

在 AWS Proton 主控台的 [元件](#) 頁面上，選擇建立元件。在元件設定中，輸入元件名稱和選用的元件描述。針對元件連接，選擇將元件連接至服務執行個體。選取您的環境、服務和服務執行個體。針對元件來源，選擇 CloudFormation，然後選擇元件 IaC 檔案。

Note

您不需要提供資訊清單，主控台會為您建立一個資訊清單。

如果您使用的是 AWS Proton API 或 AWS CLI，請使用 [CreateComponent](#) API 動作。設定 `elementName` 和選用的 `description`。設定 `environmentName`、`serviceName` 和 `serviceInstanceName`。將 `manifestTemplateSource` 設定為您所建立檔案的路徑。

 Note

當您指定服務和服務執行個體名稱時，可選擇性指定環境名稱。這兩者的組合在 AWS 您的帳戶中是唯一的，並且 AWS Proton 可以從服務執行個體判斷環境。

5. 更新您的服務執行個體以重新部署它。AWS Proton 會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元件佈建的 Amazon S3 儲存貯體。

搭配使用 git 儲存庫 AWS Proton

AWS Proton 會將 git 儲存庫用於各種用途。下列清單會分類與 AWS Proton 資源相關聯的儲存庫類型。對於重複連接到儲存庫以推送內容或從中提取內容 AWS Proton 的功能，您必須在 AWS Proton AWS 帳戶中向註冊儲存庫連結。儲存庫連結是一組屬性，AWS Proton 可在連線至儲存庫時使用。AWS Proton 目前支援 GitHub、GitHub Enterprise 和 BitBucket。

開發人員儲存庫

程式碼儲存庫 – 開發人員用來存放應用程式程式碼的儲存庫。用於程式碼部署。AWS Proton 不會直接與此儲存庫互動。當開發人員佈建包含管道的服務時，他們會提供儲存庫名稱和分支，以便從中讀取其應用程式碼。會將此資訊 AWS Proton 傳遞給其佈建的管道。

如需詳細資訊，請參閱[the section called “建立”](#)。

管理員儲存庫

範本儲存庫 – 管理員存放 AWS Proton 範本套件的儲存庫。用於範本同步。當管理員在 中建立範本時 AWS Proton，他們可以指向範本儲存庫，並讓新範本 AWS Proton 與其保持同步。當管理員更新儲存庫中的範本套件時，AWS Proton 會自動建立新的範本版本。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 “範本同步組態”](#)。

Note

如果您繼續將範本上傳至 Amazon Simple Storage Service (Amazon S3) 並呼叫 AWS Proton 範本管理 APIs 來建立新的範本或範本版本，則不需要範本儲存庫。

自我管理的佈建儲存庫

基礎設施儲存庫 – 託管轉譯基礎設施範本的儲存庫。用於資源基礎設施的自我管理佈建。當管理員建立自我管理佈建的環境時，他們會提供儲存庫。向此儲存庫 AWS Proton 提交提取請求 (PRs)，以建立環境的基礎設施，以及部署到環境的任何服務執行個體。將 基礎設施儲存庫連結至 ，AWS Proton 然後才能將其用於自我管理的基礎設施佈建。

管道儲存庫 – 用來建立管道的儲存庫。用於管道的自我管理佈建。使用額外的儲存庫佈建管道 AWS Proton，可讓 獨立於任何個別環境或服務存放管道組態。您只需要為所有自我管理佈建服務提供單一管道儲存庫。將管道儲存庫連結至 ，AWS Proton 然後才能將其用於自我管理管道佈建。


```
--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` 是選用的。

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 AWS 和客戶受管標籤的清單，如下列範例 command. AWS Proton automatic 為您產生 AWS 受管標籤所示。您也可以使用 修改和建立客戶受管標籤 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"
```

檢視連結的儲存庫資料

您可以使用 主控台 或 列出和檢視連結的儲存庫詳細資訊 AWS CLI。對於用於與 同步 git 儲存庫的儲存庫連結 AWS Proton，您可以使用 擷取儲存庫同步定義和狀態 AWS CLI。

AWS 管理主控台

使用 [AWS Proton 主控台](#) 列出和檢視連結的儲存庫詳細資訊。

1. 若要列出連結的儲存庫，請在導覽窗格中選擇儲存庫。
2. 若要檢視詳細資訊，請選擇儲存庫的名稱。

AWS CLI

列出您的連結儲存庫。

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"
}
}
```

列出您的同步儲存庫。

下列範例列出您為範本同步設定的儲存庫。

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。

下列範例會擷取範本同步儲存庫的同步狀態。

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"
}
```

刪除儲存庫連結

您可以使用 [主控台](#) 或 [刪除儲存庫連結 AWS CLI](#)。

Note

刪除儲存庫連結只會移除帳戶中 AWS Proton AWS 具有的已註冊連結。它不會從您的儲存庫刪除任何資訊。

AWS 管理主控台

使用 [主控台](#) 刪除儲存庫連結。

在儲存庫詳細資訊頁面中。

1. 在 [AWS Proton 主控台](#) 中，選擇儲存庫。
2. 在儲存庫清單中，選擇您要刪除之儲存庫左側的選項按鈕。
3. 選擇 刪除。
4. 模態會提示您確認刪除動作。
5. 遵循指示並選擇是，刪除。

AWS CLI

刪除儲存庫連結。

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

監控是維護和 AWS 解決方案的可靠性、可用性 AWS Proton 和效能的重要部分。下一節說明可與搭配使用的監控工具 AWS Proton。

AWS Proton 使用 EventBridge 自動化

您可以在 Amazon EventBridge 中監控 AWS Proton 事件。EventBridge 可從您自己的應用程式、software-as-a-service(SaaS) 應用程式和提供即時資料串流 AWS 服務。您可以設定事件以回應 AWS 資源狀態變更。EventBridge 會接著將此資料路由至目標服務，例如 AWS Lambda 和 Amazon Simple Notification Service。這些事件與 Amazon CloudWatch Events 中出現的事件相同。CloudWatch Events 提供近乎即時的系統事件串流，描述 AWS 資源的變更。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[什麼是 Amazon EventBridge？](#)。

使用 EventBridge 接收 AWS Proton 佈建工作流程中狀態變更的通知。

Event types (事件類型)

事件由包含事件模式和目標的規則組成。您可以透過選擇事件模式和目標物件來設定規則：

事件模式

每個規則會以事件模式表示，其中包含要監控的事件來源和類型，以及事件目標。若要監控事件，您可以使用要監控為事件來源的服務來建立規則。例如，您可以建立具有事件模式的規則，使用 AWS Proton 做為事件來源，在部署狀態發生變更時觸發規則。

目標

規則會收到所選服務做為事件目標。您可以設定目標服務來傳送通知、擷取狀態資訊、採取修正動作、啟動事件或採取其他動作。

事件物件包含 ID、帳戶 AWS 區域、Detail-type、來源、版本、資源、時間（選用）的標準欄位。詳細資訊欄位是巢狀物件，其中包含事件的自訂欄位。

AWS Proton 事件會盡最大努力發出。全力傳遞意味著服務會嘗試將所有事件傳送至 EventBridge，但在某些罕見的情況下，事件可能無法傳遞。

對於可以發出事件的每個 AWS Proton 資源，下表列出詳細資訊類型值、詳細資訊欄位，以及（如果可用）status和previousStatus詳細資訊欄位值清單的參考。刪除資源時，status詳細資訊欄位值為 DELETED。

資源	Detail-type 值	詳細資訊欄位
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	Detail-type 值	詳細資訊欄位
		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 服務執行個體狀態變更	name serviceName status previousStatus
ServicePipeline	AWS Proton 服務管道狀態變更	serviceName status previousStatus

資源	Detail-type 值	詳細資訊欄位
EnvironmentAccount Connection	AWS Proton 環境帳戶連線狀態變更	id status previousS tatus 狀態值
Component	AWS Proton 元件狀態變更	name status previousS tatus

AWS Proton 事件範例

下列範例顯示 AWS Proton 可將事件傳送至 EventBridge 的方式。

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 : 傳送 AWS Proton 服務狀態變更的 Amazon Simple Notification Service 提醒

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 會將狀態變更傳送至 Amazon SNS 主題。您訂閱主題，Amazon SNS 會傳送您 AWS Proton 服務的狀態變更電子郵件給您。

先決條件

您現有的 AWS Proton 服務具有 Active 狀態。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 [開始使用](#)。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 中建立之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. 針對名稱，輸入 **tutorial-service-status-change** 並選擇建立主題。
4. 在tutorial-service-status-change詳細資訊頁面中，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。當您收到它時，請開啟它，然後選擇確認訂閱。

步驟 2：註冊事件規則

註冊擷取 AWS Proton 服務狀態變更的事件規則。如需詳細資訊，請參閱 [先決條件](#)。

建立事件規則。

1. 開啟 [Amazon EventBridge 主控台](#)。
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. 在名稱和描述區段中，針對名稱輸入 **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. 針對事件類型，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 AWS Proton 服務的名稱。
- viii. 在服務詳細資訊頁面中，複製服務 Amazon Resource Name (ARN)。
- ix. 導覽回 EventBridge 主控台和您的教學課程規則，然後在文字編輯器中選擇編輯。
- x. 在文字編輯器中，針對 "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. 針對主題，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 您的服務，以驗證您的事件規則是否正常運作。

1. 切換到 [AWS Proton 主控台](#)。
2. 在導覽窗格中，選擇服務。
3. 在服務頁面中，選擇服務的名稱。

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. 為您的新執行個體填寫表單：
 - 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同相容環境。
 - c. 輸入所需輸入的值。
 - d. 選擇下一步。
8. 檢閱您的輸入，然後選擇更新。
9. 服務狀態為 之後Active，請檢查您的電子郵件，以確認您已收到提供狀態更新的 AWS 通知。

```
{
  "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. 模態會提示您驗證刪除。依照指示操作，然後選擇刪除。

刪除您的 EventBridge 規則。

1. 導覽至 [Amazon EventBridge 主控台](#)。
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. 模態會提示您驗證刪除。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，選擇服務的名稱。
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，選擇刪除您在本教學課程中建立的服務執行個體，然後選擇下一步。

6. 檢閱您的輸入，然後選擇更新。
7. 模態會提示您驗證刪除。遵循指示並選擇是，刪除。

使用 AWS Proton 儀表板讓基礎設施保持在最新狀態

AWS Proton 儀表板提供 AWS 帳戶中 AWS Proton 資源的摘要，特別著重於過時，即更新的部署資源。使用其相關聯範本的建議版本時，部署的資源是最新的。部署out-of-date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。

Resources

The screenshot displays the AWS Proton Dashboard Resources section. It includes a summary of resources and a table of service instances.

Resource type	Up to date	Failed	Minor update pending	Major update pending
Services	1	0	0	0
Service instances	2	0	0	0
Environments	1	0	0	0
Components	0	0	0	0

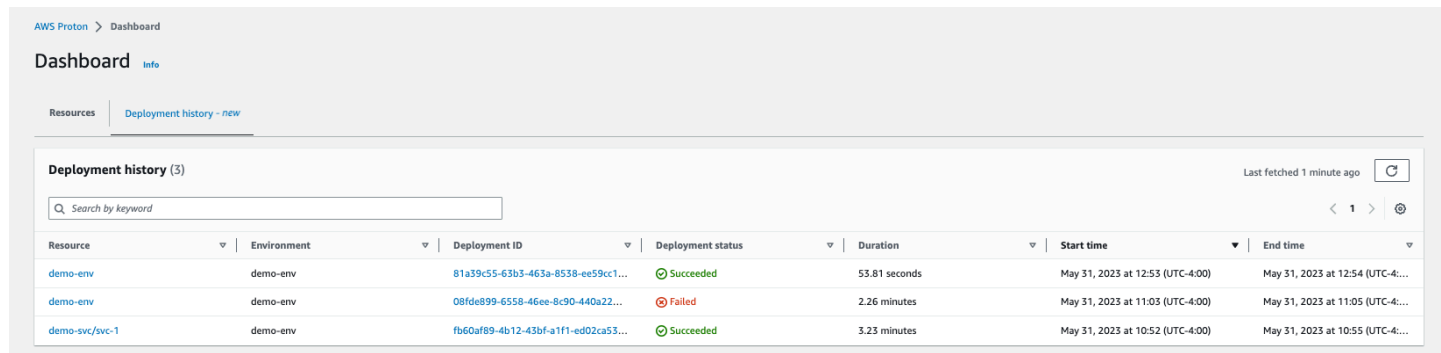
Name	Deployment status	Service template	Service	Environment	Last successful deployment	Created
demo-inst-2	Succeeded	demo-svc-temp-lambda	demo-svc-lambda	demo-env-lambda	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:52 (UTC-4:00)
demo-inst-1	Succeeded	demo-svc-temp-lambda	demo-svc-lambda	demo-env-lambda	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:52 (UTC-4:00)

儀表板的第一個索引標籤會顯示您帳戶中所有資源的計數。資源索引標籤會顯示您的服務執行個體、服務、環境和元件的數量，以及您的資源範本。它也會根據該類型的資源狀態來細分每個已部署資源類型的資源計數。服務執行個體資料表會顯示每個服務執行個體的詳細資訊，包括部署狀態、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篩選服務執行個體清單。例如，您可以篩選以查看在特定時段內部署的服務執行個體，或相對於主要或次要版本建議的過期服務執行個體。

選擇服務執行個體名稱以導覽至服務執行個體詳細資訊頁面，您可以在其中採取行動進行適當的版本更新。選擇任何其他 AWS Proton 資源名稱以導覽至其詳細資訊頁面，或選擇資源類型以導覽至個別資源清單。

部署歷史記錄



The screenshot shows the AWS Proton Dashboard with the 'Deployment history' tab selected. The table displays three deployment records for different resources in a 'demo-env' environment. The first record for 'demo-env' succeeded in 53.81 seconds. The second record for 'demo-env' failed after 2.26 minutes. The third record for 'demo-svc-1' succeeded in 3.23 minutes. The table includes columns for Resource, Environment, Deployment ID, Deployment status, Duration, Start time, and End time.

Resource	Environment	Deployment ID	Deployment status	Duration	Start time	End time
demo-env	demo-env	81a39c55-63b3-463a-8538-ee59cc1...	Succeeded	53.81 seconds	May 31, 2023 at 12:53 (UTC-4:00)	May 31, 2023 at 12:54 (UTC-4:00)
demo-env	demo-env	08fde899-6558-46ee-8c90-440a22...	Failed	2.26 minutes	May 31, 2023 at 11:03 (UTC-4:00)	May 31, 2023 at 11:05 (UTC-4:00)
demo-svc-1	demo-env	fb60af69-4b12-43bf-a1f1-ed02ca53...	Succeeded	3.23 minutes	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:55 (UTC-4:00)

部署歷史記錄索引標籤可讓您查看部署的詳細資訊。在部署歷史記錄表中，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 以查看更多詳細資訊，例如部署狀態訊息和資源輸出。資料表也可讓您篩選任何資料表屬性。

中的安全性 AWS Proton

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模式](#)將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS 服務 中執行的基礎設施 AWS 雲端。AWS 也為您提供可安全使用的服務。在[AWS 合規計劃](#)中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用的合規計劃 AWS Proton，請參閱[AWS 服務 合規計劃範圍](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 服務 的。您也必須對其他因素負責，包括資料的機密性、您的公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 時套用共同責任模型 AWS Proton。下列主題說明如何將 AWS Proton 設定為達到您的安全及法規遵循目標。您也會了解如何使用其他 AWS 服務 來協助您監控和保護 AWS Proton 資源。

主題

- [的 Identity and Access Management AWS Proton](#)
- [中的組態和漏洞分析 AWS Proton](#)
- [中的資料保護 AWS Proton](#)
- [中的基礎設施安全性 AWS Proton](#)
- [在 中記錄和監控 AWS Proton](#)
- [中的彈性 AWS Proton](#)
- [的安全最佳實務 AWS Proton](#)
- [預防跨服務混淆代理人](#)
- [CodeBuild 佈建自訂 Amazon VPC 支援](#)

的 Identity and Access Management AWS Proton

AWS Identity and Access Management (IAM) 是 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行驗證（登入）和授權（具有許可）來使用 AWS Proton 資源。IAM 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [AWS Proton 如何使用 IAM](#)
- [的政策範例 AWS Proton](#)
- [AWS 的 受管政策 AWS Proton](#)
- [使用的服務連結角色 AWS Proton](#)
- [對 AWS Proton 身分和存取進行故障診斷](#)

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 [對 AWS Proton 身分和存取進行故障診斷](#))
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 [AWS Proton 如何使用 IAM](#))
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 [的身分型政策範例 AWS Proton](#))

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的[如何登入您的 AWS 帳戶](#)。

對於程式設計存取，AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的[API 請求的AWS 第 4 版簽署程序](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務和資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務 使用臨時憑證存取。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用來自身分來源的 AWS 服務 憑證存取 Directory Service。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，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 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html 是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的 [要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取](#)。

[IAM 群組](#) 會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 使用者的使用案例](#)。

IAM 角色

IAM 角色 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html 的身分具有特定許可權，其可以提供臨時憑證。您可以透過 [從使用者切換到 IAM 角色（主控台）](#) 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 [擔任角色的方法](#)。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的快帳戶資源存取](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS，會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 [JSON 政策概觀](#)。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的[在受管政策與內嵌政策之間選擇](#)。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中[指定主體](#)。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 實體許可界限](#)。
- 服務控制政策 (SCP) — 為 AWS Organizations 中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的[服務控制政策](#)。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的[資源控制政策 \(RCP\)](#)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[工作階段政策](#)。

多種政策類型

當多種類型的政策套用到請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

AWS Proton 如何使用 IAM

在您使用 IAM 管理對的存取之前 AWS Proton，請先了解可使用哪些 IAM 功能 AWS Proton。

您可以搭配使用的 IAM 功能 AWS Proton

IAM 功能	AWS Proton 支援
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
主體許可	是
服務角色	是
服務連結角色	是

若要全面了解 AWS Proton 和其他如何與大多數 IAM 功能 AWS 服務 搭配使用，請參閱 [《AWS 服務 IAM 使用者指南》](#) 中的 [與 IAM 搭配使用](#)。

的身分型政策 AWS Proton

支援身分型政策：是

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。如需了解如何建立身分型政策，請參閱 [《IAM 使用者指南》](#) 中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

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。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](#)。

的身分型政策範例 AWS Proton

若要檢視 AWS Proton 身分型政策的範例，請參閱 [的身分型政策範例 AWS Proton](#)。

內的資源型政策 AWS Proton

支援資源型政策：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](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的快帳戶資源存取](#)。

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。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》中的 [定義的動作 AWS Proton](#)。

中的政策動作在動作之前 AWS Proton 使用以下字首：

```
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 AWS Proton 身分型政策的範例，請參閱 [的身分型政策範例 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。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 AWS Proton 資源類型及其 ARNs，請參閱《服務授權參考》中的 [定義的資源 AWS Proton](#)。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 AWS Proton 身分型政策的範例，請參閱 [的身分型政策範例 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 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

若要查看 AWS Proton 條件索引鍵的清單，請參閱《服務授權參考》中的 [的條件索引鍵 AWS Proton](#)。若要了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的範例 condition-key-based 政策，請參閱 [的條件索引鍵型政策範例 AWS Proton](#)。

中的存取控制清單 ACLs) 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。

使用的屬性型存取控制 (ABAC) 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，依據稱為標籤的屬性來定義許可。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，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 [使用 ABAC 授權定義許可](#)。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 資源的詳細資訊，請參閱 [AWS Proton 資源和標記](#)。

搭配使用暫時登入資料 AWS Proton

支援臨時憑證：是

臨時登入資料提供 AWS 資源的短期存取權，當您使用聯合或切換角色時，會自動建立。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 使用者指南》中的 [IAM 中的臨時安全憑證與可與 IAM 搭配運作的 AWS 服務](#)。

的跨服務主體許可 AWS Proton

支援轉寄存取工作階段 (FAS)：是

轉送存取工作階段 (FAS) 使用呼叫的委託人許可 AWS 服務，並結合 AWS 服務請求向下游服務提出請求。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。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可給 AWS 服務](#)。

如需詳細資訊，請參閱[AWS Proton IAM 服務角色政策範例](#)。

Warning

變更服務角色的許可可能會中斷 AWS Proton 功能。只有在 AWS Proton 提供指引時，才能編輯服務角色。

的服務連結角色 AWS Proton

支援服務連結角色：是

服務連結角色是連結至的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需詳細資訊，請參閱[使用的服務連結角色 AWS Proton](#)。

的政策範例 AWS Proton

在下列各節中尋找 AWS Proton IAM 政策範例。

主題

- [的身分型政策範例 AWS Proton](#)
- [AWS Proton IAM 服務角色政策範例](#)
- [的條件索引鍵型政策範例 AWS Proton](#)

的身分型政策範例 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。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。

如需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 \(主控台\)](#)。

如需定義的動作和資源類型的詳細資訊 AWS Proton，包括每種資源類型的 ARNs 格式，請參閱《服務授權參考》中的[的動作、資源和條件索引鍵 AWS Proton](#)。

主題

- [政策最佳實務](#)
- [的身分型政策範例連結 AWS Proton](#)

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的 [AWS 受管政策](#) 或 [任務職能的 AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。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。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素：條件](#)。
- 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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [透過 MFA 的安全 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](#)。

的身分型政策範例連結 AWS Proton

的身分型政策範例連結 AWS Proton

- [AWS 的 受管政策 AWS Proton](#)
- [AWS Proton IAM 服務角色政策範例](#)

- [的條件索引鍵型政策範例 AWS Proton](#)

AWS Proton IAM 服務角色政策範例

管理員擁有和管理由環境和服務範本定義的 AWS Proton 建立的資源。他們會將 IAM 服務角色連接到其帳戶，AWS Proton 以允許代表他們建立資源。當將他們的應用程式部署為 AWS Proton 環境中 AWS Proton 的服務時，管理員會為開發人員稍後擁有和管理的資源提供 IAM AWS Proton 角色和 AWS Key Management Service 金鑰。如需 AWS KMS 和資料加密的詳細資訊，請參閱 [中的資料保護 AWS Proton](#)。

服務角色是 Amazon Web Services (IAM) 角色，AWS Proton 允許代表您呼叫資源。如果您指定服務角色，AWS Proton 就會使用角色的憑證。使用服務角色明確指定 AWS Proton 可執行的動作。

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 [建立角色以委派許可給 AWS 服務](#)。

AWS Proton 使用 佈建的 服務角色 CloudFormation

身為平台團隊的成員，您可以在建立環境做為環境的 CloudFormation 服務角色 ([CreateEnvironment](#) API 動作的 `protonServiceRoleArn` 參數) AWS Proton 時，以管理員身分建立 AWS Proton 服務角色並將其提供給。當環境或其中執行的任何服務執行個體使用受管佈建和 AWS CloudFormation 佈建基礎設施時，此角色允許代表您對其他服務 AWS Proton 進行 API AWS 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 AWS Proton 主控台建立環境並選擇建立新角色時，這是 AWS Proton 新增至其為您建立之服務角色的政策。縮小此政策的許可範圍時，請記住會在 Access Denied 發生錯誤時 AWS Proton 失敗。

Important

請注意，下列範例中顯示的政策會將管理員權限授予任何可以向您的帳戶註冊範本的人。由於我們不知道您會在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的許可。我們建議您將許可範圍縮小為將部署在您的環境中的特定資源。

AWS Proton 的服務角色政策範例 CloudFormation

123456789012 以您的 AWS 帳戶 ID 取代。

JSON

```

{
  "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

JSON

```

{
  "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/*"
        }
      }
    }
  ]
}

```

範圍下 AWS 管佈建服務角色政策

以下是範圍縮小 AWS Proton 服務角色政策的範例，如果您只需要 AWS Proton 服務來佈建 S3 資源，您可以使用該政策。

JSON

```
{
  "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

身為平台團隊的成員，您可以在建立環境做為環境的 CodeBuild AWS Proton 服務角色 ([CreateEnvironment](#) API 動作的 `codebuildRoleArn` 參數) AWS Proton 時，以管理員身分建立服務角色並將其提供給。當環境或其中執行的任何服務執行個體使用 CodeBuild 佈建來佈建基礎設施時，此角色允許代表您對其他服務 AWS Proton 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，會將具有管理員權限的政策 AWS Proton 新增至其為您建立的服務角色。當您建立自己的角色和縮小範圍許可時，請記住會在 Access Denied 發生錯誤時 AWS Proton 失敗。

Important

請注意，AWS Proton 連接到其為您建立的角色政策會將管理員權限授予任何可以向您的帳戶註冊範本的人。由於我們不知道您會在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的許可。我們建議您將許可範圍縮小為將部署在您的環境中的特定資源。

AWS Proton CodeBuild 的服務角色政策範例

下列範例提供 CodeBuild 使用 佈建資源的許可 AWS Cloud Development Kit (AWS CDK)。

123456789012 以您的 AWS 帳戶 ID 取代。

JSON

```

{
  "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

JSON

```

{
  "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 管道服務角色

若要佈建服務管道，AWS Proton 需要對其他服務進行 API 呼叫的許可。所需的服務角色類似於您在建立環境時提供的服務角色。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，而且您可以在主控台中或透過 [UpdateAccountSettings](#) API 動作提供這些角色做為帳戶設定。

當您使用 AWS Proton 主控台更新帳戶設定，並選擇為 CloudFormation 或 CodeBuild 服務角色建立新角色時，AWS Proton 新增至其為您建立之服務角色的政策會與先前章節中所述的政策相同，[AWS 受管佈建角色](#) 以及 [CodeBuild 佈建角色](#)。縮小此政策的許可範圍時，請記住會在 Access Denied 發生錯誤時 AWS Proton 失敗。

Important

請注意，前幾節中的範例政策會將管理員權限授予任何可以向您的帳戶註冊範本的人。由於我們不知道您會在 AWS Proton 範本中定義哪些資源，因此這些政策具有廣泛的許可。建議您將許可範圍縮小為將在管道中部署的特定資源。

AWS Proton 元件角色

身為平台團隊的成員，您可以在建立環境做為環境的 CloudFormation 元件角色 ([CreateEnvironment](#) API 動作的 `componentRoleArn` 參數) AWS Proton 時，以管理員身分建立 AWS Proton 服務角色並將其提供給。此角色會縮小直接定義元件可以佈建的基礎設施範圍。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) 儲存貯體和相關的存取政策。

AWS Proton 元件角色政策範例

123456789012 以您的 AWS 帳戶 ID 取代。

JSON

```
{
  "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 元件信任政策

JSON

```

{
  "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/*"
        }
      }
    }
  ]
}

```

的條件索引鍵型政策範例 AWS Proton

下列範例 IAM 政策拒絕存取符合 Condition 區塊中指定範本 AWS Proton 的動作。請注意，這些條件索引鍵僅支援 [動作、資源和條件索引鍵 AWS Proton](#) 中列出的動作。若要管理其他動作的許可，例如 DeleteEnvironmentTemplate，您必須使用資源層級存取控制。

拒絕特定 AWS Proton 範本上範本動作的政策範例：

JSON

```
{
  "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符合 Resource 區塊中所列服務範本的 AWS Proton 範本動作 `ListServiceTemplates`，但除外。第二個陳述式拒絕存取符合 Condition 區塊中所列範本 AWS Proton 的動作。

拒絕符合特定範本 AWS Proton 之動作的政策範例：

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "Action": [
      "proton:*"
    ],
    "Resource": "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
  },
  {
    "Effect": "Deny",
    "Action": [
      "proton:*"
    ],
    "Resource": "*",
    "Condition": {
      "StringEqualsIfExists": {
        "proton:ServiceTemplate": [
          "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
        ]
      }
    }
  }
]
}

```

最終政策範例允許符合 Condition 區塊中所列特定服務範本的開發人員 AWS Proton 動作。

允許符合特定範本的 AWS Proton 開發人員動作的範例政策：

JSON

```

{
  "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 的 受管政策 AWS Proton

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自行撰寫政策更容易。建立 [IAM 客戶受管政策](#) 需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 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 受管政策。您無法變更 AWS 受管政策中的許可。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管政策移除許可，因此政策更新不會破壞您現有的許可。

此外，AWS 支援跨多個 服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 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 IAM 政策和信任關係，您可以連接到使用者、群組或角色，以允許對資源和 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。

下列信任關係會用於每個 AWS Proton 受管政策。

JSON

```
{
  "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 受管政策：AWSProtonFullAccess

您可以將 `AWSProtonFullAccess` 連接到您的 IAM 實體。AWS Proton 也會將此政策連接到允許代表您 AWS Proton 執行動作的服務角色。

此政策會授予管理許可，允許完整存取 AWS Proton 動作，以及有限存取 AWS Proton 其他依賴 AWS 的服務動作。

此政策包含下列關鍵動作命名空間：

- `proton` – 允許管理員完整存取 AWS Proton APIs。
- `iam` – 允許管理員將角色傳遞至 AWS Proton。這是必要的，以便 AWS Proton 可以代表管理員對其他服務進行 API 呼叫。
- `kms` – 允許管理員將授予新增至客戶受管金鑰。
- `codeconnections` – 允許管理員列出和傳遞程式碼連線，以便供使用 AWS Proton。

如需詳細資訊，請參閱 [AWSProtonFullAccess](#)。

AWS 受管政策：AWSProtonDeveloperAccess

您可以將 `AWSProtonDeveloperAccess` 連接到您的 IAM 實體。AWS Proton 也會將此政策連接到允許代表您 AWS Proton 執行動作的服務角色。

此政策會授予許可，允許有限存取 AWS Proton 動作和其他 AWS Proton 相依 AWS 的動作。這些許可的範圍旨在支援建立和部署 AWS Proton 服務的開發人員角色。

此政策不提供 AWS Proton 範本和環境建立、刪除和更新 APIs 存取權。如果開發人員需要比此政策提供的更有限的許可，我們建議您建立範圍縮小的自訂政策，以授予 [最低權限](#)。

此政策包含下列關鍵動作命名空間：

- `proton` – 允許參與者存取一組 AWS Proton 有限的 APIs。
- `codeconnections` – 允許參與者列出和傳遞程式碼連線，以便供使用 AWS Proton。

如需詳細資訊，請參閱 [AWSProtonDeveloperAccess](#)。

AWS 受管政策：AWSProtonReadOnlyAccess

您可以將 `AWSProtonReadOnlyAccess` 連接到您的 IAM 實體。AWS Proton 也會將此政策連接到允許代表您 AWS Proton 執行動作的服務角色。

此政策授予許可，允許唯讀存取 AWS Proton 動作，以及有限唯讀存取 AWS Proton 其他依賴 AWS 的服務動作。

此政策包含下列關鍵動作命名空間：

- `proton` – 允許參與者唯讀 AWS Proton 存取 APIs。

如需詳細資訊，請參閱 [AWSProtonReadOnlyAccess](#)。

AWS 受管政策：AWSProtonSyncServiceRolePolicy

AWS Proton 會將此政策連接至 [AWSServiceRoleForProtonSync](#) 服務連結角色，AWS Proton 允許執行範本同步。

此政策授予許可，允許有限存取 AWS Proton 動作和 AWS Proton 依賴的其他 AWS 服務動作。

此政策包含下列關鍵動作命名空間：

- `proton` – 允許 AWS Proton 同步有限存取 AWS Proton APIs。
- `codeconnections` – 允許 AWS Proton 同步有限存取 CodeConnections APIs。

如需詳細資訊，請參閱 [AWSProtonSyncServiceRolePolicy](#)。

AWS 受管政策：AWSProtonCodeBuildProvisioningBasicAccess

Permissions CodeBuild 需要執行建置 for AWS Proton CodeBuild Provisioning。您可以 `AWSProtonCodeBuildProvisioningBasicAccess` 連接到 CodeBuild 佈建角色。

此政策授予 AWS Proton CodeBuild Provisioning 運作的最低許可。它授予許可，允許 CodeBuild 產生建置日誌。它還授予 Proton 許可，讓 Infrastructure as Code (IaC) 輸出可供 AWS Proton 使用者使用。它不提供 IaC 工具管理基礎設施所需的許可。

此政策包含下列關鍵動作命名空間：

- `logs` - 允許 CodeBuild 產生建置日誌。如果沒有此許可，CodeBuild 將無法啟動。

- `proton` - 允許 CodeBuild Provisioning 命令呼叫 `aws proton notify-resource-deployment-status-change` 來更新指定 AWS Proton 資源的 IaC 輸出。

如需詳細資訊，請參閱 [AWSProtonCodeBuildProvisioningBasicAccess](#)。

AWS 受管政策：AWSProtonCodeBuildProvisioningServiceRolePolicy

AWS Proton 將此政策連接至 [AWSServiceRoleForProtonCodeBuildProvisioning](#) 服務連結角色，AWS Proton 允許執行 CodeBuild 型佈建。

此政策授予許可，允許有限存取 AWS Proton 依賴 AWS 的服務動作。

此政策包含下列關鍵動作命名空間：

- `cloudformation` - Allow AWS Proton CodeBuild 型佈建對 CloudFormation APIs 有限存取。
- `codebuild` - Allow AWS Proton CodeBuild 型佈建對 CodeBuild APIs 有限存取。
- `iam` - 允許管理員將角色傳遞至 AWS Proton。這是必要的，以便 AWS Proton 可以代表管理員對其他服務進行 API 呼叫。
- `servicequotas` - 允許 AWS Proton 檢查 CodeBuild 並行建置限制，以確保適當的建置佇列。

如需詳細資訊，請參閱 [AWSProtonCodeBuildProvisioningServiceRolePolicy](#)。

AWS 受管政策：AWSProtonServiceGitSyncServiceRolePolicy

AWS Proton 會將此政策連接至允許 AWS Proton 執行服務同步的 [AWSServiceRoleForProtonServiceSync](#) 服務連結角色。

此政策授予許可，允許有限存取 AWS Proton 動作和 AWS Proton 依賴的其他 AWS 服務動作。

此政策包含下列關鍵動作命名空間：

- `proton` - 允許 AWS Proton 同步有限存取 AWS Proton APIs。

如需詳細資訊，請參閱 [AWSProtonServiceGitSyncServiceRolePolicy](#)。

AWS Proton AWS 受管政策的更新

檢視自此服務開始追蹤這些變更 AWS Proton 以來，AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱 AWS Proton 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	Date
AWSProtonCodeBuildProvisioningServiceRolePolicy – 更新現有政策	允許 AWS Proton 執行 CodeBuild 型佈建的服務連結角色受管政策現在會授予呼叫 CloudFormation TagResource 和 UntagResource API 動作的許可。需要這些許可才能對資源執行標記操作。	2024 年 6 月 15 日
AWSProtonFullAccess – 更新現有政策	服務連結角色搭配 Git 儲存庫使用 Git 同步的受管政策已更新，適用於具有兩個服務字首的資源。如需詳細資訊，請參閱 針對 AWS CodeConnections 和 受管政策使用服務連結角色 。 https://docs.aws.amazon.com/dtconsole/latest/userguide/security-iam-awsmanpol.html	2024 年 4 月 25 日
AWSProtonDeveloperAccess – 更新現有政策	服務連結角色搭配 Git 儲存庫使用 Git 同步的受管政策已更新，適用於具有兩個服務字首的資源。如需詳細資訊，請參閱 針對 AWS CodeConnections 和 受管政策使用服務連結角色 。 https://docs.aws.amazon.com/dtconsole/latest/userguide/security-iam-awsmanpol.html	2024 年 4 月 25 日
AWSProtonSyncServiceRolePolicy – 更新現有政策	服務連結角色搭配 Git 儲存庫使用 Git 同步的受管政策已更新，適用於具有兩個服務字首的資源。如需詳細資訊，請參閱 針對 AWS CodeConne	2024 年 4 月 25 日

變更	描述	Date
	ctions 和 受管政策使用服務連結角色。 https://docs.aws.amazon.com/dtconsole/latest/userguide/security-iam-awsmanpol.html	
AWSProtonCodeBuildProvisioningServiceRolePolicy – 更新現有政策	AWS Proton 已更新此政策來新增許可，以確保帳戶具有必要的 CodeBuild 並行建置限制，以便使用 CodeBuild Provisioning。	2023 年 5 月 12 日
AWSProtonServiceGitSyncServiceRolePolicy – 新政策	AWS Proton 新增了新的政策，AWS Proton 以允許執行服務同步。此政策用於 AWSServiceRoleForProtonServiceSync 服務連結角色。	2023 年 3 月 31 日
AWSProtonDeveloperAccess – 更新現有政策	AWS Proton 新增了 GetResourcesSummary 動作，可讓您檢視範本、部署的範本資源和過時資源的摘要。	2022 年 11 月 18 日
AWSProtonReadOnlyAccess – 更新現有政策	AWS Proton 新增了 GetResourcesSummary 動作，可讓您檢視範本、部署的範本資源和過時資源的摘要。	2022 年 11 月 18 日
AWSProtonCodeBuildProvisioningBasicAccess – 新政策	AWS Proton 新增了新的政策，提供 CodeBuild 執行建置 for AWS Proton CodeBuild Provisioning 所需的許可。	2022 年 11 月 16 日

變更	描述	Date
AWSProtonSyncServiceRolePolicy – 新政策	AWS Proton 新增了新的政策，AWS Proton 以允許執行與 CodeBuild 型佈建相關的操作。此政策用於 AWSServiceRoleForProtonCodeBuildProvisioning 服務連結角色。	2022 年 9 月 2 日
AWSProtonFullAccess – 更新現有政策	AWS Proton 已更新此政策，以提供對新 AWS Proton API 操作的存取權，並修正某些 AWS Proton 主控台操作的許可問題。	2022 年 3 月 30 日
AWSProtonDeveloperAccess – 更新現有政策	AWS Proton 更新此政策，以提供對新 AWS Proton API 操作的存取權，並修正某些 AWS Proton 主控台操作的許可問題。	2022 年 3 月 30 日
AWSProtonReadOnlyAccess – 更新現有政策	AWS Proton 更新此政策，以提供對新 AWS Proton API 操作的存取權，並修正某些 AWS Proton 主控台操作的許可問題。	2022 年 3 月 30 日
AWSProtonSyncServiceRolePolicy – 新政策	AWS Proton 新增了新的政策，AWS Proton 以允許執行與範本同步相關的操作。此政策用於 AWSServiceRoleForProtonSync 服務連結角色。	2021 年 11 月 23 日
AWSProtonFullAccess – 新政策	AWS Proton 新增了新的政策，以提供 AWS Proton API 操作和 AWS Proton 主控台的管理角色存取權。	2021 年 6 月 9 日

變更	描述	Date
AWSProtonDeveloperAccess – 新政策	AWS Proton 已新增新政策，以提供開發人員角色對 AWS Proton API 操作和 AWS Proton 主控台的存取權。	2 021 年 6 月 9 日
AWSProtonReadOnlyAccess – 新政策	AWS Proton 新增了新的政策，以提供 AWS Proton API 操作和 AWS Proton 主控台的唯讀存取權。	2 021 年 6 月 9 日
AWS Proton 已開始追蹤變更。	AWS Proton 已開始追蹤其 AWS 受管政策的變更。	2 021 年 6 月 9 日

使用的服務連結角色 AWS Proton

AWS Proton use AWS Identity and Access Management (IAM) [服務連結角色](#)。服務連結角色是直接連結至的唯一 IAM 角色類型 AWS Proton。服務連結角色由預先定義，AWS Proton 並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

主題

- [使用角色進行 AWS Proton 同步](#)
- [使用 CodeBuild 型佈建的角色](#)

使用角色進行 AWS Proton 同步

AWS Proton use AWS Identity and Access Management (IAM) [服務連結角色](#)。服務連結角色是直接連結至的唯一 IAM 角色類型 AWS Proton。服務連結角色由預先定義，AWS Proton 並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更 AWS Proton 輕鬆地設定，因為您不必手動新增必要的許可。AWS Proton 會定義其服務連結角色的許可，除非另有定義，否則只能 AWS Proton 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。這可保護您的 AWS Proton 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

的服務連結角色許可 AWS Proton

AWS Proton 使用兩個名為 `AWSServiceRoleForProtonSync` 和 `AWSServiceRoleForProtonServiceSync` 的服務連結角色。

`AWSServiceRoleForProtonSync` 服務連結角色信任下列服務擔任該角色：

- `sync.proton.amazonaws.com`

名為的角色許可政策 `AWSProtonSyncServiceRolePolicy` 允許對指定的資源 AWS Proton 完成下列動作：

- 動作：建立、管理和讀取 AWS Proton 範本和範本版本
- 動作：在 CodeConnections 上使用連線 CodeConnections

如需此政策的詳細資訊，請參閱 [AWS 受管政策：AWSProtonSyncServiceRolePolicy](#)。

`AWSServiceRoleForProtonServiceSync` 服務連結角色信任下列服務擔任該角色：

- `service-sync.proton.amazonaws.com`

名為的角色許可政策 `AWSProtonServiceGitSyncServiceRolePolicy` 允許對指定的資源 AWS Proton 完成下列動作：

- 動作：建立、管理和讀取 AWS Proton 服務和服務執行個體

如需此政策的詳細資訊，請參閱 [AWS 受管政策：AWSProtonServiceGitSyncServiceRolePolicy](#)。

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 建立服務連結角色 AWS Proton

您不需要手動建立服務連結角色，當您在 AWS CLI、或 AWS API AWS 管理主控台 AWS Proton 中設定要同步的儲存庫或服務時，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。當您設定儲存庫或服務進行同步時 AWS Proton，會再次為您 AWS Proton 建立服務連結角色。

若要重新建立 `AWSServiceRoleForProtonSync` 服務連結角色，建議您設定要同步的儲存庫，若要重新建立 `AWSServiceRoleForProtonServiceSync`，建議您設定要同步的服務。

編輯的服務連結角色 AWS Proton

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](#)。

刪除的服務連結角色 AWS Proton

您不需要手動刪除 `AWSServiceRoleForProtonSync` 角色。當您在 AWS CLI、或 AWS API AWS 管理主控台中刪除所有用於儲存庫同步 AWS Proton 的連結儲存庫時，AWS Proton 會清除資源，並為您刪除服務連結角色。

AWS Proton 服務連結角色支援的區域

AWS Proton 支援在所有提供服務 AWS 區域的 中使用服務連結角色。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 CodeBuild 型佈建的角色

AWS Proton use AWS Identity and Access Management (IAM) [服務連結角色](#)。服務連結角色是直接連結至的唯一 IAM 角色類型 AWS Proton。服務連結角色由預先定義，AWS Proton 並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更 AWS Proton 輕鬆地設定，因為您不必手動新增必要的許可。AWS Proton 會定義其服務連結角色的許可，除非另有定義，否則只能 AWS Proton 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。這可保護您的 AWS Proton 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

的服務連結角色許可 AWS Proton

AWS Proton 使用名為 `AWSServiceRoleForProtonCodeBuildProvisioning` 的服務連結角色：AWS Proton CodeBuild 佈建的服務連結角色。

AWSServiceRoleForProtonCodeBuildProvisioning 服務連結角色信任下列服務擔任該角色：

- `codebuild.proton.amazonaws.com`

名為的角色許可政策AWSProtonCodeBuildProvisioningServiceRolePolicy允許對指定的資源 AWS Proton 完成下列動作：

- 動作：在CloudFormation 堆疊和轉換上建立、管理和讀取
- 動作：在 CodeBuild 專案和組建上建立、管理和讀取

如需此政策的詳細資訊，請參閱 [AWS 受管政策：AWSProtonCodeBuildProvisioningServiceRolePolicy](#)。

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 建立服務連結角色 AWS Proton

您不需要手動建立服務連結角色，當您在 AWS 管理主控台 AWS CLI、或 AWS API AWS Proton 中建立使用 CodeBuild 型佈建的環境時，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 型佈建的環境時 AWS Proton，會再次為您 AWS Proton 建立服務連結角色。

編輯的服務連結角色 AWS Proton

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](#)。

刪除的服務連結角色 AWS Proton

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。不過，您必須先刪除 中使用 CodeBuild 型佈建的所有環境和服務（執行個體和管道），AWS Proton 才能手動將其刪除。

手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 來刪除 AWSServiceRoleForProtonCodeBuildProvisioning 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 [刪除服務連結角色](#)。

AWS Proton 服務連結角色支援的區域

AWS Proton 支援在所有提供服務 AWS 區域 的 中使用服務連結角色。如需詳細資訊，請參閱 AWS 一般參考 中的 [AWS Proton 端點和配額](#)。

對 AWS Proton 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 AWS Proton 和 IAM 時可能遇到的常見問題。

主題

- [我無權在 中執行動作 AWS Proton](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許 以外的人員 AWS 帳戶 存取我的 AWS Proton 資源](#)

我無權在 中執行動作 AWS Proton

如果 AWS 管理主控台 告知您無權執行 動作，則必須聯絡您的管理員尋求協助。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。

我未獲得執行 iam:PassRole 的授權

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 可讓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 沒有將角色傳遞給服務的許可。

```
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 管理員。您的管理員提供您的簽署憑證。

我想要允許以外的人員 AWS 帳戶 存取我的 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：

- 若要了解 是否 AWS Proton 支援這些功能，請參閱 [AWS Proton 如何使用 IAM](#)。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱 [《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱 [《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有](#)。
- 如需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 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 [《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取](#)。

中的組態和漏洞分析 AWS Proton

AWS Proton 不會為客戶提供的程式碼提供修補程式或更新。客戶負責更新修補程式並將其套用至自己的程式碼，包括其上執行之服務和應用程式的原始碼，AWS Proton 以及其服務和環境範本套件中提供的程式碼。

客戶負責更新和修補其環境和服務中的基礎設施資源。AWS Proton 不會自動更新或修補任何資源。客戶應參閱其架構中資源的文件，以了解其各自的修補政策。

除了提供客戶請求的環境和服務更新給次要版本的服務和環境範本之外，AWS Proton 不會提供修補程式或更新給客戶在服務和環境範本和範本套件中定義的資源。

如需詳細資訊，請參閱以下 資源：

- [共同的責任模型](#)
- [Amazon Web Services : 安全程序概觀](#)

中的資料保護 AWS Proton

AWS Proton 符合 AWS [共同責任模型](#)，其中包括資料保護的法規和指導方針。AWS 負責保護執行所有的全域基礎設施 AWS 服務。會 AWS 維護對此基礎設施上託管資料的控制。包括處理客戶內容和個人資料的安全組態控制。AWS 客戶和 APN 合作夥伴、做為資料控制者或資料處理者，對其放入中的任何個人資料負責 AWS 雲端

基於資料保護目的，建議您保護 AWS 帳戶 登入資料並使用 AWS Identity and Access Management (IAM) 設定個別使用者帳戶，以便每個使用者只獲得完成其任務所需的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (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 設定 API 和使用活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 中的所有預設安全控制 AWS 服務。

我們強烈建議您絕對不要將敏感的識別資訊，例如客戶的帳戶號碼，放入任意格式的文字欄位中，例如名稱欄位。這包括當您 AWS 服務 使用 AWS Proton 或使用主控台、API AWS CLI 或其他 AWS SDKs 時。您輸入資源識別符自由格式文字欄位或與 AWS 資源管理相關的類似項目的任何資料，都可能被選入診斷日誌中。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套件的 S3 儲存貯體中加密範本套件中的靜態敏感資料，則必須使用 SSE-S3 或 SSE-KMS 金鑰來允許 AWS Proton 擷取範本套件，以便將其連接到已註冊的 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

在其中 AWS Proton，預設會使用 AWS Proton 擁有的金鑰來加密所有客戶資料。如果您提供客戶擁有和管理的 AWS KMS 金鑰，所有客戶資料都會使用客戶提供的金鑰進行加密，如以下段落所述。

當您建立 AWS Proton 範本時，您可以指定金鑰，AWS Proton 並使用 登入資料來建立授予，AWS Proton 允許 使用您的金鑰。

如果您手動淘汰授予，或停用或刪除您指定的金鑰，則 AWS Proton 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](#)。

加密內容是一組金鑰/值對，其中包含任意的非秘密資料。在加密資料的請求中包含加密內容時，AWS KMS 會以密碼編譯方式將加密內容繫結至加密的資料。若要解密資料，您必須傳遞相同的加密內容。

客戶可以使用加密內容來識別在稽核記錄和日誌中使用其客戶受管金鑰。它也會以純文字顯示在日誌中，例如 AWS CloudTrail 和 Amazon CloudWatch Logs。

AWS Proton 不會採用任何客戶指定或外部指定的加密內容。

AWS Proton 新增下列加密內容。

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

第一個加密內容會識別資源相關聯的 AWS Proton 範本，並做為客戶受管金鑰許可和授權的限制。

第二個加密內容會識別已加密 AWS Proton 的資源。

下列範例顯示 AWS Proton 加密內容的使用方式。

開發人員建立服務執行個體。

```
{
  "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"
}
```

中的基礎設施安全性 AWS Proton

作為受管服務，AWS Proton 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全的最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 已發佈的 API 呼叫，AWS Proton 透過網路存取。使用者端必須支援下列專案：

- 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。

若要改善網路隔離，您可以使用 AWS PrivateLink，如下節所述。

AWS Proton 和介面 VPC 端點 (AWS PrivateLink)

您可以在 VPC 和 之間建立私有連線，AWS Proton 方法是建立介面 VPC 端點。介面端點採用 [AWS PrivateLink](#)技術，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，私下存取 AWS Proton APIs。VPC 中的執行個體不需要公有 IP 地址即可與 AWS Proton APIs通訊。VPC 與 之間的流量 AWS Proton 不會離開 Amazon 網路。

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 端點的考量事項

設定介面 VPC 端點之前 AWS Proton，請務必檢閱《Amazon VPC 使用者指南》中的[介面端點屬性和限制](#)。

AWS Proton 支援從您的 VPC 呼叫其所有 API 動作。

支援 VPC 端點政策 AWS Proton。根據預設，AWS Proton 允許透過端點完整存取。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](#)。

建立的介面 VPC 端點 AWS Proton

您可以使用 Amazon VPC 主控台或 AWS Command Line Interface () 來建立 AWS Proton 服務的 VPC 端點 AWS CLI。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[建立介面端點](#)。

AWS Proton 使用下列服務名稱建立的 VPC 端點：

- `com.amazonaws.region.proton`

如果您為端點啟用私有 DNS，則可以 AWS Proton 使用區域的預設 DNS 名稱向 提出 API 請求，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](#)。

為 建立 VPC 端點政策 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：AWS Proton 動作的 VPC 端點政策

以下是 端點政策的範例 AWS Proton。連接到端點時，此政策會授予所有資源上所有委託人的所列 AWS Proton 動作的存取權。

JSON

```
{
  "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": "*"
    }
  ]
}
```

在 中記錄和監控 AWS Proton

監控是維護 AWS Proton 和其他 AWS 解決方案的可靠性、可用性和效能的重要部分。AWS 提供下列監控工具，可讓您監看在 中執行的執行個體 AWS Proton、在發生錯誤時回報，以及適時採取自動動作。

目前，AWS Proton 它本身未與 Amazon CloudWatch Logs 或 整合 AWS Trusted Advisor。管理員可以設定和使用 CloudWatch 來監控其服務和環境範本中 AWS 服務 定義的其他。AWS Proton 已與 整合 AWS CloudTrail。

- Amazon CloudWatch AWS 會即時監控您的 AWS 資源和您在 上執行的應用程式。您可以收集和追蹤指標、建立自訂儀板表，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 可讓您監控、存放和存取來自 Amazon EC2 執行個體、CloudTrail 及其他來源的日誌檔案。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 會擷取由 發出或代表發出的 API 呼叫和相關事件，AWS 帳戶 並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱 [「AWS CloudTrail 使用者指南」](#)。
- Amazon EventBridge 為無伺服器事件匯流排服務，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 會從您自己的應用程式、Software-as-a-Service (SaaS) 應用程式提供即時資料串流，AWS 服務 並將該資料路由至 Lambda 等目標。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 EventBridge 自動化](#)和 [EventBridge 使用者指南](#)。

中的彈性 AWS Proton

AWS 全球基礎設施是以 AWS 區域 和 可用區域為基礎建置。AWS 區域提供多個實體隔離和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

除了 AWS 全球基礎設施之外，AWS Proton 還提供 功能，以協助支援您的資料彈性和備份需求。

AWS Proton 備份

AWS Proton 會維護所有客戶資料的備份。在總中斷的情況下，此備份可用來從先前的有效狀態還原 AWS Proton 和客戶資料。

的安全最佳實務 AWS Proton

AWS Proton 提供安全功能，供您在開發和實作自己的安全政策時考慮。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](#)
- [請勿在範本和範本套件中嵌入登入資料](#)
- [使用加密來保護敏感資料](#)
- [使用 AWS CloudTrail 來檢視和記錄 API 呼叫](#)

使用 IAM 控制存取

IAM 是 AWS 服務，可用來管理中的使用者及其許可 AWS。您可以使用 IAM 搭配 AWS Proton 來指定管理員和開發人員可執行 AWS Proton 的動作，例如管理範本、環境或服務。您可以使用 IAM 服務角色，AWS Proton 允許代表您呼叫其他服務。

如需 AWS Proton 和 IAM 角色的詳細資訊，請參閱 [Identity and Access Management AWS Proton](#)。

實作最低權限存取。如需詳細資訊，請參閱 AWS Identity and Access Management 《使用者指南》[中的 IAM 中的政策和許可](#)。

請勿在範本和範本套件中嵌入登入資料

我們建議您在堆疊範本中使用動態參考，而不是在 CloudFormation 範本和範本套件中內嵌敏感資訊。

動態參考提供精簡且強大的方式，讓您參考儲存在其他服務中的外部值，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 CloudFormation 《使用者指南》中的 [使用動態參考指定範本值](#)。

[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。[AWS Systems Manager 參數存放區](#) 為組態資料管理提供安全的階層式儲存。

如需定義範本參數的詳細資訊，請參閱 CloudFormation 《使用者指南 <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html>》中的。

使用加密來保護敏感資料

在其中 AWS Proton，預設會使用 AWS Proton 擁有的金鑰來加密所有客戶資料。

身為平台團隊的成員，您可以提供客戶受管金鑰給 `aws:Proton`，AWS Proton 以加密和保護您的敏感資料。加密 S3 儲存貯體中的靜態敏感資料。如需詳細資訊，請參閱 [中的資料保護 AWS Proton](#)。

使用 AWS CloudTrail 來檢視和記錄 API 呼叫

AWS CloudTrail 會追蹤在您的 `aws:Proton` 中進行 API 呼叫的任何人 AWS 帳戶。每當任何人使用 API、主控台或 AWS Proton AWS CLI 命令時，都會記錄 AWS Proton API AWS Proton 呼叫。啟用記錄，然後指定 Amazon S3 儲存貯體來存放日誌。如此一來，如果您需要，您可以稽核在您帳戶中進行 AWS Proton 呼叫的人員。如需詳細資訊，請參閱 [在中記錄和監控 AWS Proton](#)。

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，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:Proton` 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。為了防止這種情況，AWS 提供工具，協助您保護所有服務的資料，讓服務主體能夠存取您帳戶中的資源。

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，以限制將另一個服務 AWS Proton 提供給資源的許可。如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容金鑰來限制許可。如果同時使用這兩個全域條件內容金鑰，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，請使用 `aws:SourceAccount`。

的值 `aws:SourceArn` 必須是 AWS Proton 存放的資源。

防範混淆代理人問題最有效的方法，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/*`。

下列範例示範如何在 `aws:Proton` 中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵 AWS Proton，以防止混淆代理人問題。

JSON

```
{
  "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 Amazon VPC 支援

AWS Proton CodeBuild Provisioning 會在位於 AWS Proton 環境帳戶中的 CodeBuild 專案中執行任意客戶提供的 CLI 命令。這些命令通常會使用 Infrastructure as Code (IaC) 工具來管理資源，例如 CDK。如果您在 Amazon VPC 中有資源，CodeBuild 可能無法存取它們。為了啟用此功能，CodeBuild 支援在特定 Amazon VPC 內執行的能力。幾個範例使用案例包括：

- 從自我託管的內部成品儲存庫擷取相依性，例如 PyPI 適用於 Python、Maven 適用於 Java 和 npm 適用於 Node.js 的
- CodeBuild 需要存取特定 Amazon VPC 中的 Jenkins 伺服器，才能註冊管道。
- 在設定為僅允許透過 Amazon VPC 端點存取的 Amazon S3 儲存貯體中存取物件。
- 針對私有子網路上隔離的 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 Provisioning 在自訂 VPC 中執行，AWS Proton 會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。接著，將這些值輸入環境規格。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

VPC ID、子網路和安全群組需要新增至範本結構描述，以便它們可以存在於環境規格中。

範例 `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
```

這會新增三個新屬性，供資訊清單使用：

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

清單檔案

若要在 CodeBuild 中設定 Amazon VPC 設定，範本資訊清單中 `project_properties` 會提供名為 `project_properties` 的選用屬性。的內容 `project_properties` 會新增至建立 CodeBuild 專案的 CloudFormation 堆疊。這不僅可以新增 [Amazon VPC CloudFormation 屬性](#)，還可以新增任何支援的 [CodeBuild CloudFormation 屬性](#)，例如建置逾時。提供給 `proton-inputs.json` 的相同資料可供 `project_properties` 使用。

將此區段新增至您的 `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` 可能看起來像這樣 :

```
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

當您使用已啟用 CodeBuild 佈建 VPC 的範本建立環境時，您必須提供 Amazon VPC ID、子網路和安全群組。

若要取得您區域中所有 Amazon VPC IDs 的清單，請執行下列命令：

```
aws ec2 describe-vpcs
```

若要取得所有子網路 IDs 的清單，請執行：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

⚠ Important

僅包含私有子網路。如果您提供公有子網路，CodeBuild 將會失敗。公有子網路具有網際網路 [閘道](#) 的預設路由，而私有子網路則不會。

執行下列命令以取得安全群組 IDs。這些 IDs 也可以透過 取得 AWS 管理主控台：

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

這些值會類似：

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
security-groups:
  - sg-03bc4c4ce32d67e8d
```

確保 CodeBuild 許可

Amazon VPC 支援需要特定許可，例如建立彈性網路界面的能力。

如果環境是在 主控台中建立，請在環境建立精靈期間輸入這些值。如果您想要以程式設計方式建立環境，您的 `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 獲指派 Amazon Resource Name (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、服務執行個體、元件和儲存庫。您可以標記這些資源，以協助您整理和識別這些資源。您可以使用標籤，依用途、擁有者、環境或其他條件來分類資源。如需詳細資訊，請參閱 [標記策略](#)。若要追蹤和管理資源 AWS Proton，您可以使用下列各節所述的標記功能。

AWS 標記

您可以將中繼資料以標籤形式指派給 AWS 資源。每個標籤都包含客戶定義的金鑰和選用值。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身分識別資訊 (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。標籤不適用於私有或敏感資料。

每個標籤都有兩個部分。

- 標籤鍵 (例如 CostCenter、Environment 或 Project)。標籤金鑰會區分大小寫。
- 標籤值 (選用) (例如 111122223333 或 Production)。與標籤金鑰相同，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

系統會保留以字aws:首開頭的系統建立標籤以供 AWS 使用，且不計入此限制。您無法編輯或刪除以 aws: 字首開頭的標籤。

- 針對每一個資源，每個標籤鍵必須是唯一的，且每個標籤鍵只能有一個值。
- 標籤索引鍵在 UTF-8 中必須至少為 1，最多為 128 個 Unicode 字元。
- 在 UTF-8 中，標籤值必須至少為 1，最多為 256 個 Unicode 字元。
- 標籤中允許的字元為以 UTF-8 表示的字母、數字、空格，以及下列字元：* _ . : / = + - @。

AWS Proton 標記

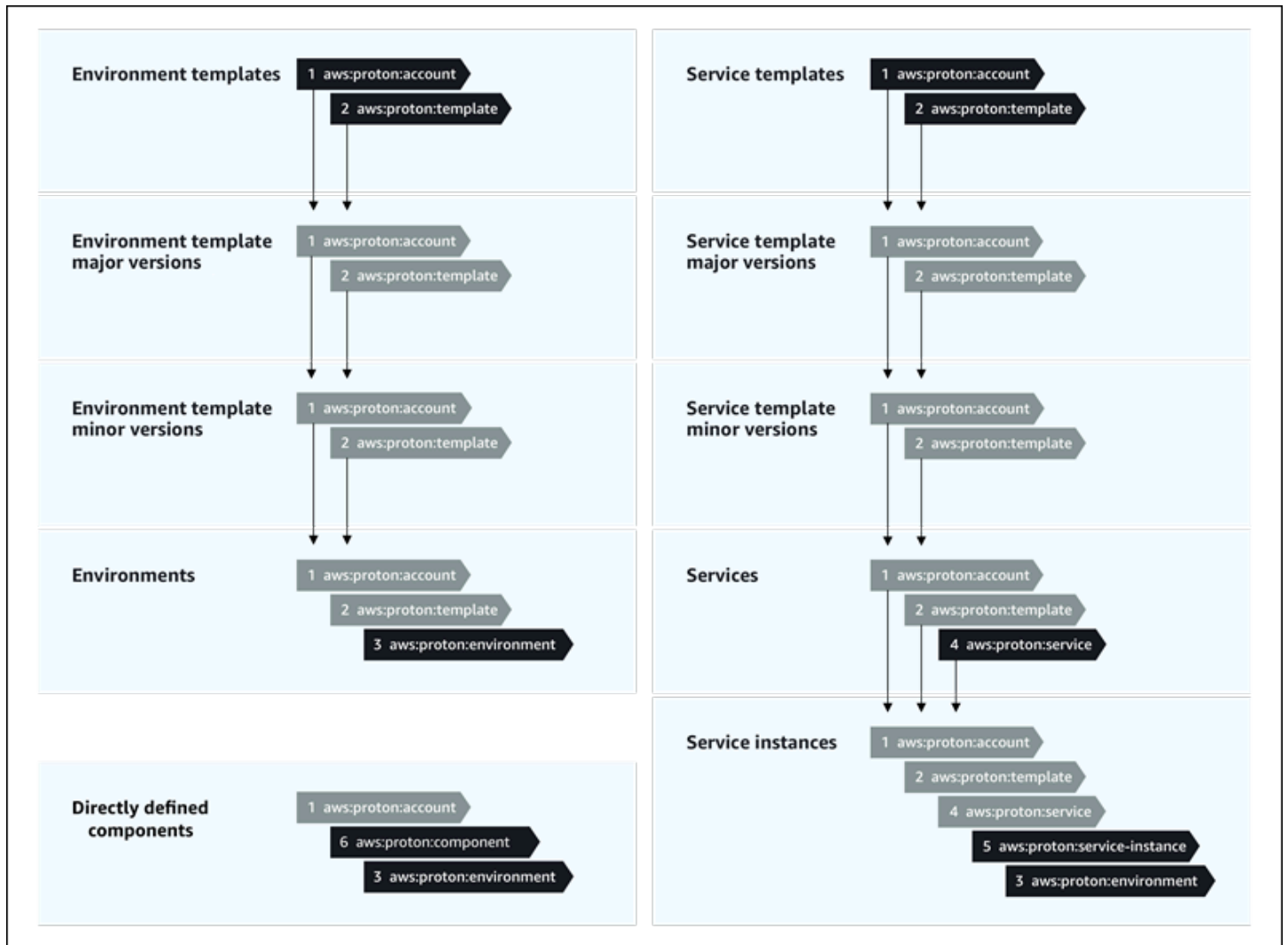
透過 AWS Proton，您可以使用您建立的標記以及 AWS Proton 自動為您產生的標記。

AWS Proton AWS 受管標記

當您建立 AWS Proton 資源時，AWS Proton 會自動為您的新資源產生 AWS 受管標記，如下圖所示。AWS 受管標記稍後會傳播到以新 AWS Proton 資源為基礎的其他資源。例如，從環境範本傳播到其版本的受管標記，以及從服務傳播到其服務執行個體的受管標記。

Note

AWS 環境帳戶連線不會產生受管標記。如需詳細資訊，請參閱 [the section called “帳戶連線”](#)。



將標籤傳播至佈建的資源

如果佈建的資源，例如服務和環境範本中定義的資源，支援 AWS 標記，AWS 受管標籤會傳播為客戶受管標籤到佈建的資源。這些標籤不會傳播到不支援 AWS 標記的佈建資源。

AWS Proton 會依 AWS Proton 帳戶、已註冊的範本和部署的環境，以及服務和服務執行個體，將標籤套用至您的資源，如下表所述。您可以使用 AWS 受管標籤來檢視和管理 AWS Proton 資源，但無法修改它們。

AWS 受管標籤金鑰	傳播的客戶受管金鑰	說明
aws:proton:account	proton:account	建立和部署 AWS Proton 資源 AWS 的帳戶。
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	所選服務執行個體的 ARN。
aws:proton:component	proton:component	所選元件的 ARN。

以下是 AWS Proton 資源的 AWS 受管標籤範例。

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

以下是套用至佈建資源的客戶受管標籤範例，該資源是從 AWS 受管標籤傳播而來。

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

透過 [AWS 受管佈建](#)，會將傳播的標籤直接 AWS Proton 套用至佈建的資源。

使用 [自我管理佈建](#)，AWS Proton 讓傳播的標籤與其在佈建提取請求 (PR) 中提交的轉譯 IaC 檔案一起可用。標籤會在字串映射變數 中提供 `proton_tags`。我們建議您在 Terraform 組態中參考此變數，以在中包含 AWS Proton 標籤 `default_tags`。這會將 AWS Proton 標籤傳播到所有佈建的資源。

下列範例顯示在環境 Terraform 範本中的此標籤傳播方法。

以下是 `proton_tags` 變數定義：

`https://proton.environment.variables.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：

`proton.auto.tfvars.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"
  }
}
```

以下是如何將 AWS Proton 標籤新增至 Terraform 組態，以便將標籤新增至佈建的資源：

```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

客戶受管標籤

每個 AWS Proton 資源的配額上限為 50 個客戶受管標籤。客戶受管標籤傳播至子 AWS Proton 資源的方式與 AWS 受管標籤相同，但不傳播至現有 AWS Proton 資源或佈建的資源。如果您將新標籤套用到 AWS Proton 具有現有子資源的資源，並且希望使用新標籤來標記現有的子資源，則需要使用 主控台 或 手動標記每個現有的子資源 AWS CLI。

使用主控台和 CLI 建立標籤

當您使用主控台建立 AWS Proton 資源時，您有機會在建立程序的第一或第二頁建立客戶受管標籤，如下列主控台快照所示。選擇新增標籤，輸入索引鍵和值並繼續。

The screenshot shows the 'Tags' section in the AWS Proton console. It features a heading 'Customer managed tags' with a sub-heading 'Add tags to help you search, filter, and track your service in Proton.' Below this, there are two input fields for 'Key' and 'Value - optional'. The 'Key' field contains 'my-key' and the 'Value' field contains 'my-tag'. A 'Remove' button is located to the right of the 'Value' field. Below the input fields is an 'Add new tag' button. A message below the button states 'You can add up to 49 more tags.' At the bottom of the screenshot, there is a blue information box with a close button (X) that reads: 'New tags will only propagate to service instances that you create after you have created the new tags. They won't propagate to existing service instances.'

使用 AWS Proton 主控台建立新資源後，您可以從詳細資訊頁面檢視其 AWS 受管和客戶受管標籤清單。

建立或編輯標籤

1. 在 [AWS Proton 主控台](#) 中，開啟 AWS Proton 資源詳細資訊頁面，您可以在其中看到標籤清單。
2. 選擇管理標籤。
3. 在管理標籤頁面中，您可以檢視、建立、移除和編輯標籤。您無法修改列在頂端的 AWS 受管標籤。不過，您可以使用編輯欄位來新增和修改客戶受管標籤，這些欄位列在 AWS 受管標籤之後。

選擇新增標籤以建立新的標籤。

4. 輸入新標籤的索引鍵和值。
5. 若要編輯標籤，請在所選索引鍵的標籤值欄位中輸入值。
6. 若要刪除標籤，請選擇移除所選標籤。
7. 完成變更後，請選擇儲存變更。

使用 建立標籤 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，如最終範例所示。

```
$ aws proton list-tags-for-resource \  
  --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

故障診斷 AWS Proton

了解如何疑難排解 的問題 AWS Proton。

主題

- [參考 CloudFormation 動態參數的部署錯誤](#)

參考 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 [Jinja 逸出](#) 的。這些錯誤可能是由於 Jinja 錯誤解譯您的動態變數所造成。CloudFormation 動態參數語法與搭配 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}}'
```

範例 AWS Proton 參數 Jinja 語法：

```
'{{ service_instance.environment.outputs.env-outputs }}'
```

為了避免這些錯誤解譯錯誤，Jinja 會逸出 CloudFormation 動態參數，如下列範例所示。

此範例來自 CloudFormation 使用者指南。AWS Secrets Manager 秘密名稱和 json 金鑰區段可用來擷取存放在秘密中的登入憑證。

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
    AllocatedStorage: '20'
    DBInstanceClass: db.t2.micro
    Engine: mysql
    MasterUsername: '{{resolve:secretsmanager:MyRDSecret:SecretString:username}}'
    MasterUserPassword:
      '{{resolve:secretsmanager:MyRDSecret:SecretString:password}}'
```

若要逸出 CloudFormation 動態參數，您可以使用兩種不同的方法：

- 在之間括住區塊`{% raw %}` and `{% endraw %}` :

```
'{% 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 %}'
```

- 將參數括在之間`"{{ }}"` :

```
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}}' }}'
```

如需詳細資訊，請參閱 [Jinja 逸出](#)。

AWS Proton 配額

下表列出 AWS Proton 配額。所有值都是每個 AWS 帳戶、每個支援 AWS 的區域。

資源配額	預設值限制	是否可調整？
範本套件的大小上限	10 MB	× 否
範本資訊清單檔案的大小上限	2 MB	× 否
範本結構描述檔案的大小上限	2 MB	× 否
每個範本檔案的大小上限	2 MB	× 否
每個範本名稱的長度上限	100 個字元	× 否
每個套件的 CloudFormation 範本檔案數目上限	1	× 否
每個帳戶、服務和環境範本合併的已註冊範本數量上限	1000	✓ 是
每個範本註冊的範本版本數量上限	1000	✓ 是
每個 CodeBuild 佈建套件的檔案數量上限	500	× 否
每個帳戶的環境數量上限	1000	✓ 是
每個帳戶的服務數量上限	1000	✓ 是
每個服務的服務執行個體數量上限	20	✓ 是
每個帳戶的元件數量上限	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

下表說明與最新版本 AWS Proton 和客戶意見回饋相關的文件的**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。

- API 版本：2020-07-20

變更	描述	日期
終止支援通知	AWS 將於 2026 年 10 月 7 日結束對的支援 AWS Proton。	2025 年 10 月 7 日
受管政策更新	AWSProtonCodeBuildProvisioningServiceRolePolicy 政策已更新。	2024 年 6 月 15 日
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
服務同步組態。	AWS Proton 新增對 服務同步組態 的支援。	2023 年 3 月 31 日
CodeBuild ：	AWS Proton 新增 CodeBuild 佈建 的支援。	2022 年 11 月 16 日

受管政策更新	新增 AWS Proton CodeBuild Provisioning Basic Access 的政策，提供 CodeBuild 執行建置 for AWS Proton CodeBuild Provisioning 所需的許可。	2022 年 11 月 11 日
Terraform 標籤傳播	已將 Terraform 標籤傳播新增至 標記 章節。	2022 年 9 月 16 日
API 遷移指南	已移除 GA 前 API 遷移指南。	2022 年 8 月 12 日
AWS Proton 物件	新增有關 AWS Proton 物件及其與其他 AWS 和第三方物件關係的主題。請參閱 AWS Proton 物件 。	2022 年 7 月 29 日
連結的儲存庫說明	釐清連結（已註冊）儲存庫的目的及其在指南中的用量。	2022 年 7 月 18 日
指南合併	使用者指南將兩個單獨的管理員和使用者指南合併為單一指南 AWS Proton。	2022 年 6 月 30 日
受管政策更新	更新 受管政策，以提供對新 AWS Proton API 操作的存取權，並修正某些 AWS Proton 主控台操作的許可問題。請參閱 AWS 的 受管政策 AWS Proton 。	2022 年 6 月 20 日
CLI 入門	使用使用新範本程式庫的新教學課程更新 入門 AWS CLI 。	2022 年 6 月 14 日
直接定義的元件	新增 元件 章節，並在本指南中進行相關修改。	2022 年 6 月 1 日
AWS Proton 範本程式庫	新增 AWS Proton 範本程式庫 主題。	2022 年 5 月 6 日

Terraform 一般可用性 (GA)	將提取請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 建立儲存庫的連結 。	2022 年 3 月 23 日
文件更新	新增環境帳戶連線標記。	2021 年 11 月 26 日
範本同步和 Terraform 預覽	使用 範本同步 功能新增自動化範本版本控制以獲得一般可用性，並使用預覽版中的 Terraform 提取請求佈建 。返回 API 遷移指南 。	2021 年 11 月 24 日
文件更新	新增了 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	新增至主控台的說明面板。主控台範本版本刪除不會再刪除較低版本。已移除 API 遷移指南 。	2021 年 9 月 8 日
AWS Proton 一般可用性 (GA) 版本	新增 跨帳戶環境 、 EventBridge 監控 、 IAM 條件金鑰 、 冪等 支援和 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設施用於具有的環境 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 、 使用 AWS Proton 環境中現有的外部基礎設施 ，以及取消環境、服務執行個體和管道部署。AWS Proton 現在支援 PrivateLink 。已新增額外的刪除驗證，以防止次要版本在使用資源時被錯誤刪除。	2021 年 4 月 27 日

[使用 標記 AWS Proton](#)

公有預覽版本 2 包含 AWS Proton [標記](#)和在沒有服務[管道](#)的情況下啟動服務的能力。

2021 年 3 月 5 日

[初始版本](#)

公有預覽版本現在可在所選區域中使用。

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](#)。