開發人員指南

AWS App Runner

AWS App Runner: 開發人員指南

Table of Contents

什麼是 AWS App Runner？ 1
誰是應用程序亞軍？ 1
訪問應用運行器 1
應用程式執行程式的 2
下一步是什麼 2
設定 3
註冊一個 AWS 帳戶 3
建立具有管理權限的使用者 3
授與程式設計存取權 4
下一步是什麼 6
開始使用 7
必要條件 7
步驟 1：建立應用程式執行器服務 8
步驟 2：變更服務代碼 18
步驟 3：進行配置更改 19
步驟 4：檢視服務的記錄 21
步驟 5：清除 23
下一步是什麼 23
建築與概念 25
應用程序運行器 25
應用程序運行器支持 27
應用程序運行器 28
應用運行器資源配額 29
影像式服務 31
影像儲存庫提供 31
在您的帳戶中使用存儲在 Amazon ECR 中的 AWS 圖像 31
在不同帳戶中使用存放在 Amazon ECR 中的 AWS 映像檔 32
使用存儲在 Amazon ECR 公共圖像 32
圖像示例 34
基於代碼的服務 35
原始碼儲存庫提供者 36
從原始程式碼儲存庫提供者部署 36
來源目錄 36
應用程式執行器管理 37

託管運行時版本和應用程序運行器構建 38
更多有關應用程序運行器構建和遷移 39
Python 平台 42
Python 運行時配置 43
特定執行階段版本的編號說明 43
Python 運行時示例 44
版本資訊 49
Node.js 平台 50
Node.js 執行階段設定 51
特定執行階段版本的編號說明 53
Node.js 執行階段範例 53
版本資訊 57
爪哇平台 59
Java 運行時配置 60
Java 執行階段範例 61
版本資訊 65
網路平台 66
.NET 運行時配置 67
.NET 運行時示例 67
版本資訊 70
PHP 平台 71
PHP 運行時配置 72
相容性 72
PHP 執行階段範例 74
版本資訊 82
紅寶石平台 83
紅寶石運行配置 84
紅寶石運行時例 85
版本資訊 87
圍棋平台 88
去運行時配置 89
Go 執行階段範例 89
版本資訊 91
開發應用程式執行器 93
運行時信息 93
程式碼開發指南 94

應用程式執行器 96
整體主控台配置 96
「服務」頁面 96
服務儀表板頁面 97
[連線的帳戶] 頁面 98
自動調整規模組態頁面 98
管理您的服務 100
建立 100
必要條件 100
建立服務 101
重建失敗的服務 115
使用應用程式執行程式主控台重建失敗的應用程式執行器 116
使用應用程序運行器 API 或重建失敗的應用程序運行器服務 AWS CLI 117
部署 118
部署方法 118
手動部署 120
組態 121
使用應用程式執行器 API 設定您的服務，或 AWS CLI 122
使用應用程式執行器主控台設定服務 122
使用應用程序運行器配置文件配置服務 124
可觀測性配置 124
配置資源 125
運作狀態檢查組態 127
連線 129
管理連線 129
自動擴展 130
管理服務的 auto 調整 132
管理 auto 調整設定資源 133
自訂網域名稱 140
將自訂網域關聯 (連結) 至您的服務 141
取消關聯 (取消連結) 自訂網域 143
管理自訂網域 143
設定 Amazon 路線 53 別名記錄 151
暫停/恢復 153
暫停和刪除比較 153
當您的服務暫停時 154

暫停和恢復您的服務 154
刪除 156
暫停和刪除比較 156
應用程序亞軍刪除什麼？ 156
刪除您的服務 157
參考環境變數 159
將敏感資料參考為環境變數 159
考量事項 160
許可 161
管理環境變數 162
應用程式執行器 162
應用程序運行器 API 或 AWS CLI 164
聯網 170
術語 170
一般條款 170
設定傳出流量的特定術語 171
設定傳入流量的特定詞彙 171
傳入流量 171
標頭 172
啟用私有端點 172
為應用程式執行者的公用端點啟用 IPv6 183
傳出流量 187
VPC 連接器 188
子網路 188
安全群組 189
管理 VPC 存取 190
可觀測性 195
活動 195
追蹤應用程式執行器服務 195
記錄檔 (CloudWatch 記錄檔) 196
應用程式執行器記錄群組和串 197
在主控台中檢視應用程式執行程式 198
度量 (CloudWatch) 200
應用程式執行器 201
在主控台中檢視應用程式執行程式 202
事件處理（EventBridge） 204

建立 EventBridge 規則以對應用程式執行器事件採取行動 205
應用程序運行器事件 205
應用程序運行器事件模式 206
應用程序運行器事件 207
API 動作 (CloudTrail) 209
應用程序運行器信息 CloudTrail 209
瞭解應用程式執行器記錄檔項 210
X-Ray 追蹤 213
檢測您的追蹤應用程式 214
將 X-Ray 權限新增至您的應用程式執行個體角色 217
為您的應用程式執行器服務啟用 X-Ray 217
查看應用程序運行器服務的 X-Ray 跟踪數據 217
AWS WAF 網絡 ACL 219
傳入的 Web 請求流程 219
將 WAF 網絡 ACL 關聯到您的應用程序運行器服務 220
考量事項 220
許可 221
管理網頁 ACL 222
應用程式執行器 223
AWS CLI 226
測試和記錄 AWS WAF 網頁 ACL 230
應用程序運行器配置 232
範例 232
組態檔案範例 233
參考資料 235
結構概述 236
頂部 236
構建部分 237
執行區段 239
應用程序亞軍 API 243
使用與應用 AWS CLI 程式執行程式搭配使用 243
使用 AWS CloudShell 243
取得的 IAM 許可 AWS CloudShell 244
與應用程序運行器交互使 AWS CloudShell 244
使用驗證應用程序運行器服務 AWS CloudShell 247
疑難排解 249

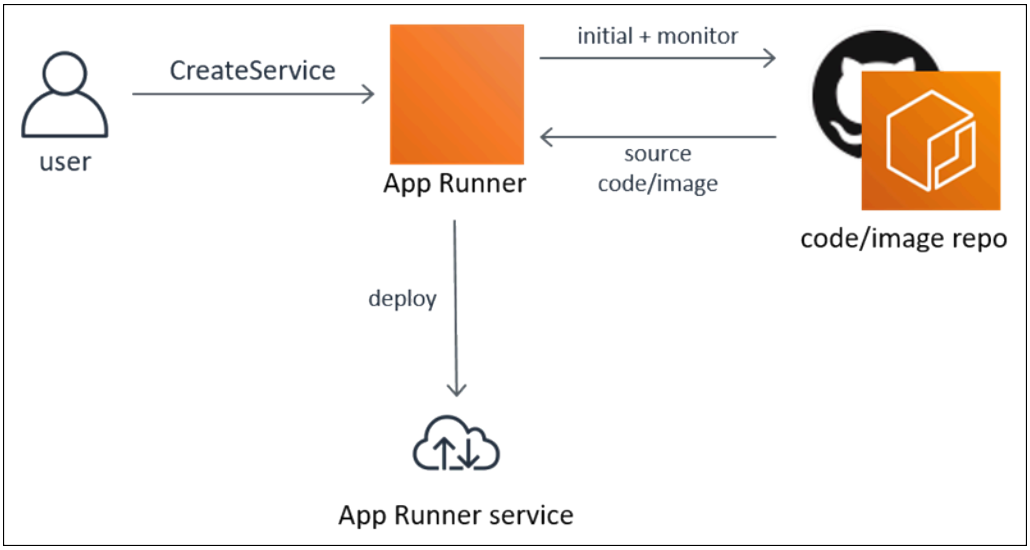
無法建立服務 249
自訂網域名稱 250
取得自訂網域的建立失敗錯誤 250
取得自訂網域的 DNS 憑證驗證擱置錯誤 251
基本的疑難排解 252
自訂網域憑證續約 252
請求路由錯誤 253
404 發送 HTTP/HTTPS 流量到應用程序運行器服務端點時未找到錯誤 254
連線失敗至 Amazon RDS 或下游服務 254
安全 257
資料保護 257
資料加密 258
網際網路隱私權 259
身分與存取管理 259
物件 260
使用身分驗證 260
使用政策管理存取權 263
應用程式執行器和 IAM 265
身分型政策範例 272
使用服務連結角色 276
AWS 受管理政策 283
故障診斷 285
日誌記錄和監控 286
法規遵循驗證 286
恢復能力 287
基礎架構安全 288
VPC 端點 288
為應用程式執行器設定 VPC 端點 289
VPC 網路隱私權考量 289
使用端點政策搭配 VPC 端點來控制存取 290
與介面端點整合 290
共同責任模式 290
安全最佳實務 290
預防性安全最佳實務 290
偵測性安全最佳實務 291
AWS 詞彙表 292

..... ccxciii

什麼是 AWS App Runner？

AWS App Runner 是一種 AWS 服務，可提供快速、簡單且具成本效益的方式，從原始程式碼或容器映像直接部署到 AWS 雲端中可擴充且安全的 Web 應用程式。您不需要學習新技術、決定要使用哪個運算服務，也不需要知道如何佈建和設定 AWS 資源。

應用程序運行器直接連接到您的代碼或圖像存儲庫。它提供自動整合和交付管道，包括全受管作業、高效能、可擴充性和安全性。



誰是應用程序亞軍？

如果您是開發人員，可以使用 App Runner 簡化部署新版程式碼或映像儲存庫的程序。

對於操作團隊，每次提交推送到代碼存儲庫或將新的容器映像版本推送到映像存儲庫時，App Runner 都會啟用自動部署。

訪問應用運行器

您可以使用下列任一介面來定義和設定 App Runner 服務部署：

- 應用程序運行器控制台-提供用於管理應用程序運行器服務的 Web 界面。
- 應用程序運行器 API-提供用於執行應用程序運行器操作的 RESTful API。如需詳細資訊，請參閱 AWS App Runner API 參考。
- AWS 命令列界面 (AWS CLI) — 為包括 Amazon VPC 在內的一組廣泛 AWS 服務提供命令，並在視窗、macOS 和 Linux 上受到支援。如需詳細資訊，請參閱 AWS Command Line Interface。

- AWS SDK — 提供特定語言的 API，並處理許多連線詳細資料，例如計算簽章、處理要求重試和錯誤處理。如需詳細資訊，請參閱 AWS 開發套件。

應用程式執行程式的

應用程序 Runner 提供了一種經濟高效的方式來運行您 您只需為應用程序運行器服務消耗的資源付費。當請求流量較低時，您的服務可以縮減到更少的運算執行個體。您可以控制延展性設定：最低和最高數量的佈建執行個體，以及執行個體處理的最高負載。

如需應用程式執行器自動縮放的詳細資訊，請參閱the section called “自動擴展”。

如需定價資訊，請參閱 AWS App Runner 定價。

下一步是什麼

在下列主題中了解如何開始使用應用程式執行器：

- 設定— 完成使用應用程序運行器的先決條件步驟。
- 開始使用— 將第一個應用程序部署到應用程序運行器

設定應用程式執行器

如果您是新 AWS 客戶，請先完成此頁面上列出的設定先決條件，然後再開始使用 AWS App Runner。

對於這些設定程序，您可以使用 AWS Identity and Access Management (IAM) 服務。如需 IAM 的完整資訊，請參閱下列參考資料：

- AWS Identity and Access Management (IAM)
- IAM 使用者指南

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

下一步是什麼

您已完成先決條件步驟。若要將您的第一個應用程式部署到應用程式執行器，開始使用

開始使用應用程式執行器

AWS App Runner 是一項 AWS 服務，可提供快速、簡單且符合成本效益的方式，將現有的容器映像檔或原始程式碼直接轉換為 AWS 雲端.

本教程介紹了如何使用 AWS App Runner 將應用程序部署到應用程序運行器服務。它逐步完成配置源代碼和部署，服務構建和服務運行時。它也會示範如何部署程式碼版本、進行設定變更，以及檢視記錄檔。最後，自學課程將展示如何在遵循自學課程的程序時清理您建立的資源。

主題

- 必要條件
- 步驟 1：建立應用程式執行器服務
- 步驟 2：變更服務代碼
- 步驟 3：進行配置更改
- 步驟 4：檢視服務的記錄
- 步驟 5：清除
- 下一步是什麼

必要條件

在開始教學課程之前，請務必執行下列動作：

1. 完成中的設定步驟設定。
2. 決定您是否要使用 GitHub 儲存庫或 Bitbucket 儲存庫。
   - 要使用 Bitbucket，請先創建一個 Bitbucket 帳戶，如果您還沒有帳戶。如果您是 Bitbucket 的新手，請參閱 Bitbucket 雲端文件中的開始使用 Bitbucket。
   - 要使用 GitHub，請創建一個GitHub帳戶（如果您還沒有帳戶）。如果您不熟悉 GitHub，請參閱「GitHub文件」 GitHub 中的「開始使用」。

> **Note**
>
> 您可以從您的帳戶建立與多個儲存庫提供者的連線。因此，如果您想逐步從 a GitHub 和 Bitbucket 存儲庫進行部署，則可以重複此過程。下次通過創建新的 App Runner 服務並為其他存儲庫提供程序創建新的帳戶連接。

3. 在儲存庫提供者帳戶中建立儲存庫。本教學課程使用儲存庫名稱python-hello。使用下列範例中指定的名稱和內容，在儲存庫的根目錄中建立檔案。

python-hello範例儲存庫的檔案

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
    server.serve_forever()
```

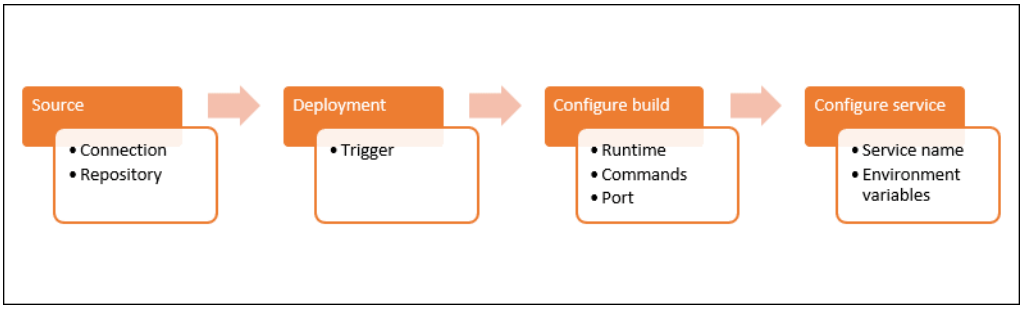
步驟 1：建立應用程式執行器服務

在此步驟中，您會根據您建立的範例原始程式碼儲存庫 GitHub 或 Bitbucket 做為其一部分，建立應用程式執行器服務。the section called “必要條件”這個例子包含一個簡單的 Python 網站。以下是您建立服務所採取的主要步驟：

1. 設定您的原始程式碼。
2. 設定來源部署。

3. 設定應用程式組建。
4. 設定您的服務。
5. 檢閱並確認。

下圖概述了創建應用程序運行器服務的步驟：

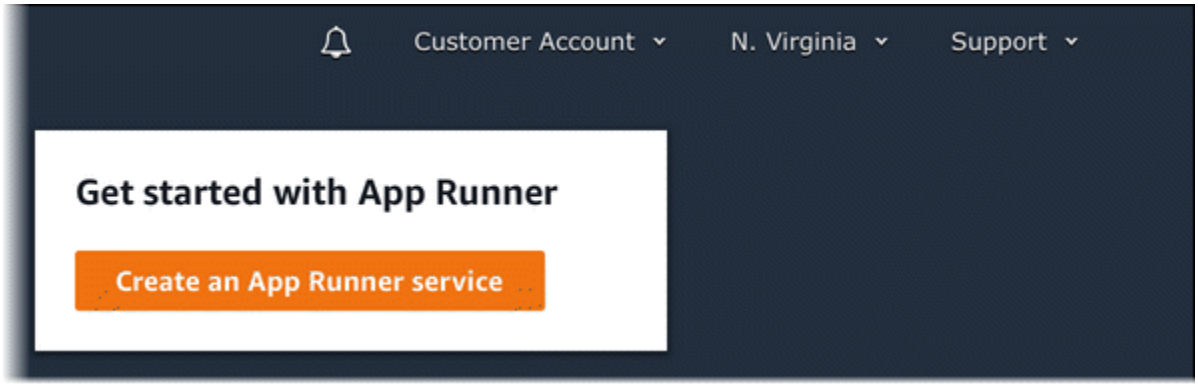


若要根據原始程式碼儲存庫建立應用程式執行器服務

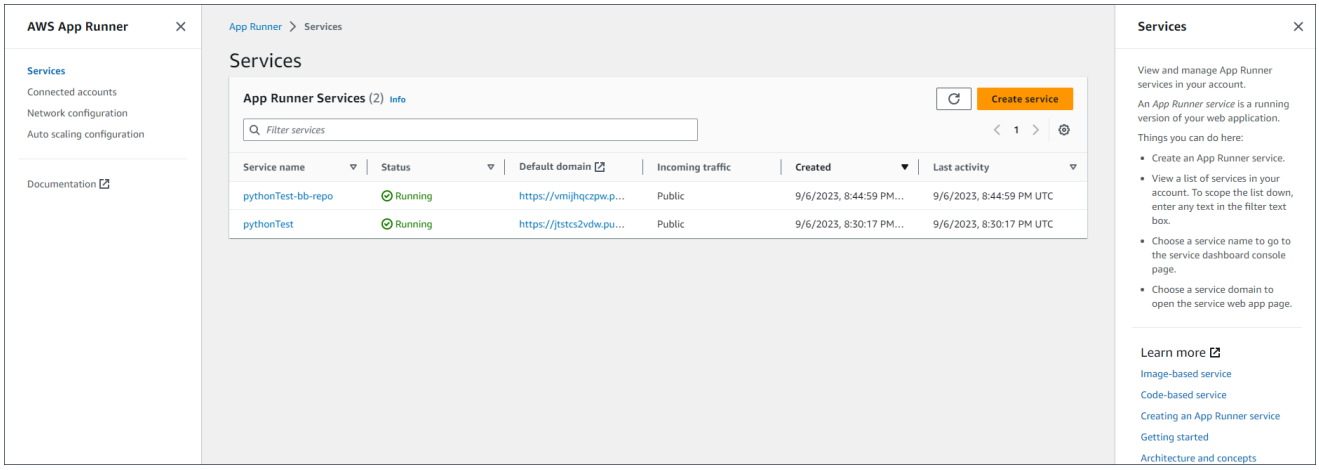
1. 設定您的原始程式碼。

   a. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.

   b. 如果還 AWS 帳戶 沒有任何應用程序運行器服務，則顯示控制台主頁。選擇創建應用程序運行器服務。



   如果 AWS 帳戶 已有服務，則會顯示含有您服務清單的「服務」頁面。選擇 Create service (建立服務)。



c. 在「來源與部署」頁面的「來源」段落中，選擇「來源程式碼儲存區域」做為「儲存區域」類型。

d. 選取提供者類型。選擇任一GitHub或比特桶。

e. 接下來選擇添加新的。如果出現提示，請提供您的 GitHub 或 Bitbucket 認證。

f. 根據您之前選取的提供者類型，選擇下一組步驟。

> **Note**
>
> 下列步驟將 AWS 連接器安裝 GitHub 到您的 GitHub 帳戶是一次性步驟。您可以重複使用連線，以根據此帳戶中的儲存庫建立多個 App Runner 服務。當您擁有現有的連線時，請選擇該連線，然後跳至存放庫選取。
> 這同樣適用於您的 Bitbucket 帳戶的 AWS 連接器。如果您同時使用 GitHub 和 Bitbucket 做為應用程式執行器服務的原始程式碼儲存庫，則需要為每個供應商安裝一個 AWS 連接器。然後，您可以重複使用每個連接器來創建更多應用程序運行器

- 對於 GitHub，請按照下列步驟操作。

  i. 在下一個畫面中，輸入連線名稱。

  ii. 如果這是您第一次使用應 GitHub 用程式執行器，請選取 [安裝其他]。

  iii. 在AWS 連接器對 GitHub話方塊中，如果出現提示，請選擇您的 GitHub 帳戶名稱。

  iv. 如果系統提示您授權連 AWS 接器 GitHub，請選擇授權 AWS 連線。

  v. 在「安裝 AWS 連接器 GitHub」對話方塊中，選擇「安裝」。

  您的帳戶名稱會顯示為選取的GitHub 帳戶/組織。現在，您可以在帳戶中選擇一個存儲庫。

vi. 對於存放庫，請選擇您建立的範例存放庫`python-hello`。對於「分支」，請選擇儲存庫的預設分支名稱 (例如 main)。

vii. 將來源目錄保留為預設值。目錄預設為儲存庫根目錄。您將原始程式碼儲存在先前的先決條件步驟中的儲存庫根目錄中。

- 對於比特桶，請按照下列步驟操作。

i. 在下一個畫面中，輸入連線名稱。

ii. 如果這是您第一次搭配應用程式執行器使用 Bitbucket，請選取 [安裝另一個]。

iii. 在AWS CodeStar 請求存取對話方塊中，您可以選取您的工作區，並授與 Bitbucket 整合的存取權。 AWS CodeStar 選取您的工作區，然後選取授與存取權。

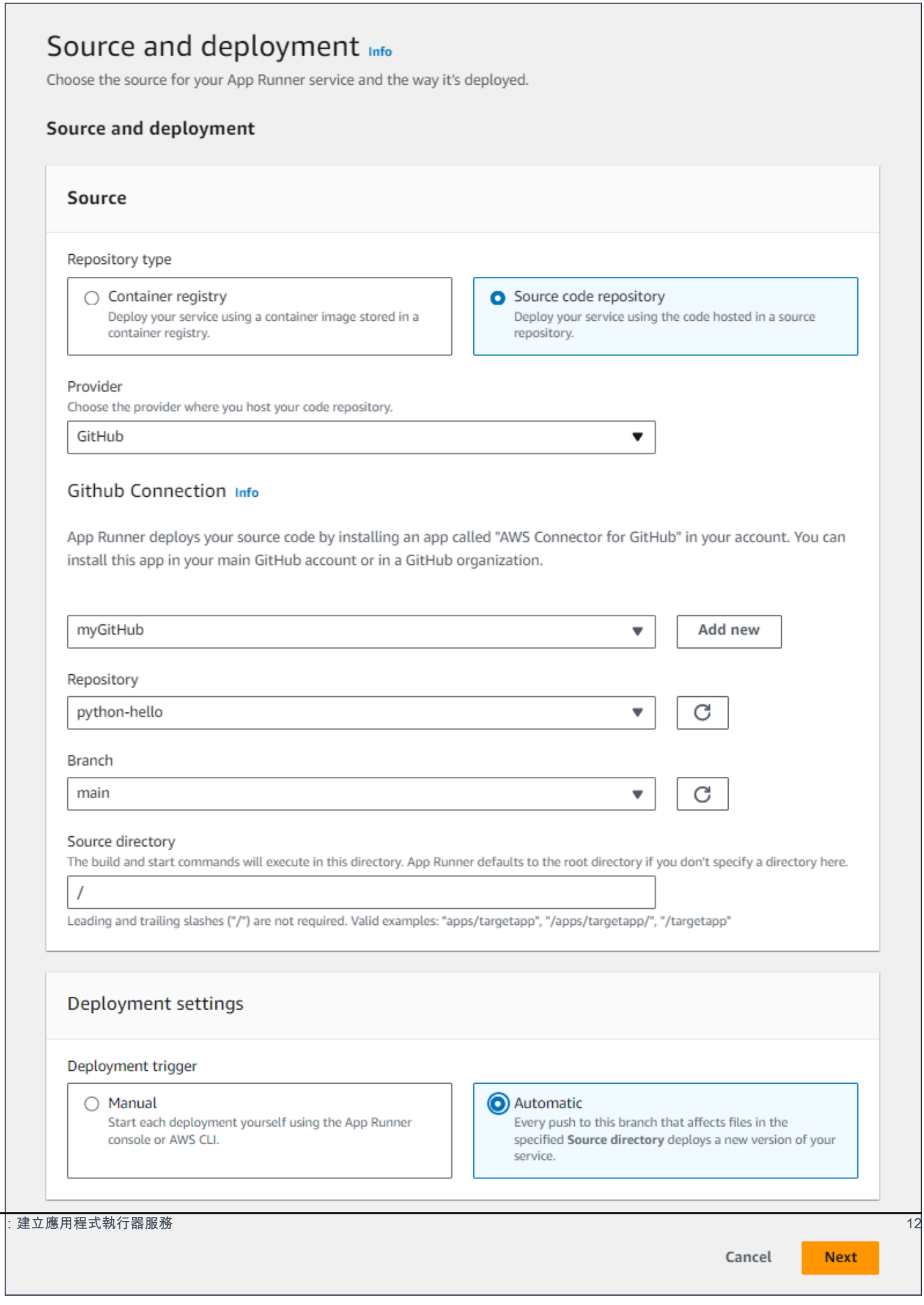
iv. 接下來，您將被重定向到 AWS 控制台。確認 Bitbucket 應用程式已設定為正確的 Bitbucket 工作區，然後選取 [下一步]。

v. 對於存放庫，請選擇您建立的範例存放庫`python-hello`。對於「分支」，請選擇儲存庫的預設分支名稱 (例如 main)。

vi. 將來源目錄保留為預設值。目錄預設為儲存庫根目錄。您將原始程式碼儲存在先前的先決條件步驟中的儲存庫根目錄中。

2. 設定您的部署：在「部署設定」區段中，選擇「自動」，然後選擇「下一步」。

> **Note**
>
> 透過自動部署，每次對儲存庫來源目錄的新認可都會自動部署服務的新版本。


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source and deployment
Source
Repository type
Container registry
Deploy your service using a container image stored in a container registry.
Source code repository
Deploy your service using the code hosted in a source repository.
Provider
Choose the provider where you host your code repository.
GitHub
Github Connection Info
App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.
myGitHub
Add new
Repository
python-hello
Branch
main
Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.
/
Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
Every push to this branch that affects files in the specified Source directory deploys a new version of your service.
Cancel
Next

3. 設定應用程式組建。

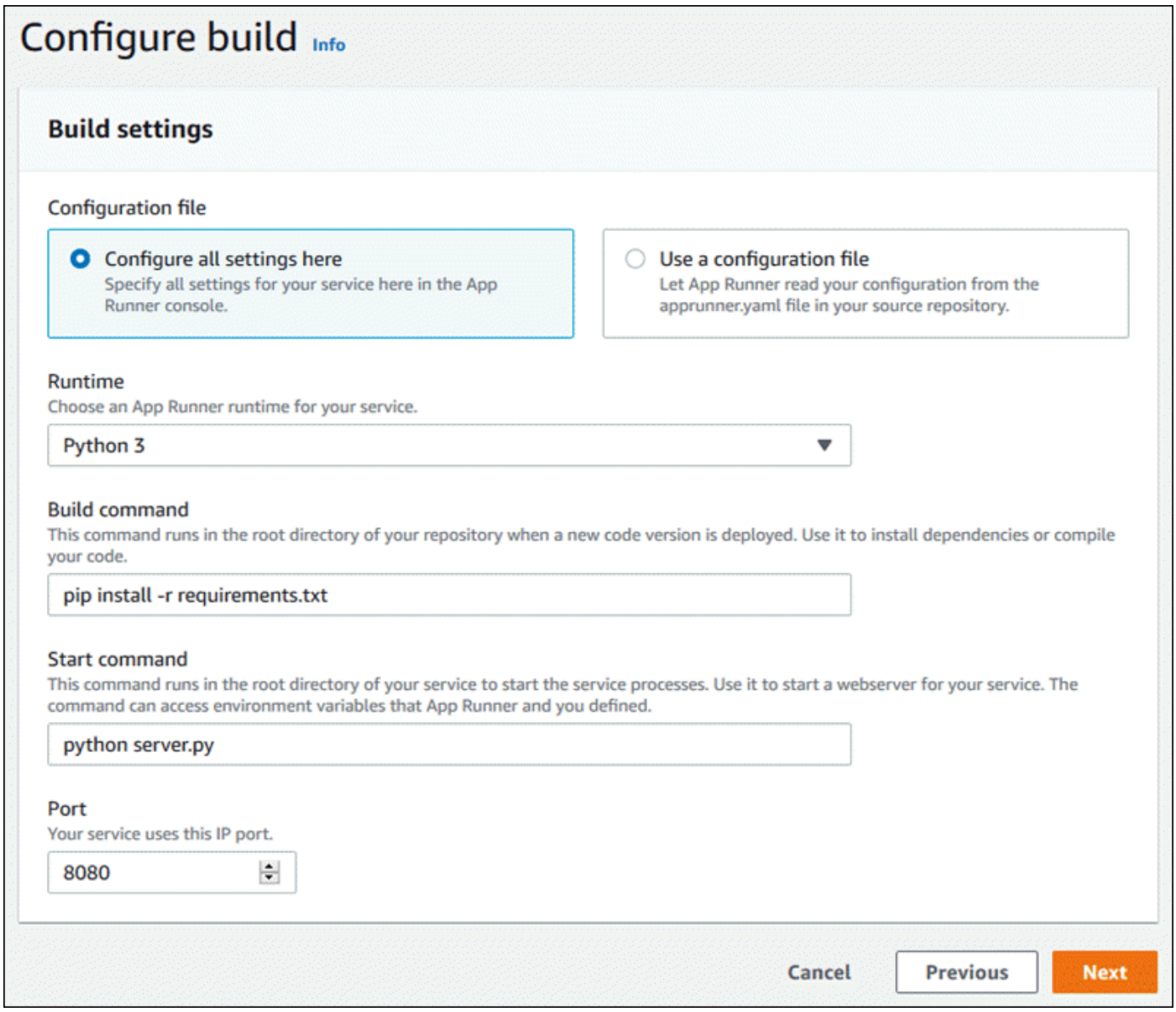
   a. 在 [設定組建] 頁面上，對於 [組態檔案]，選擇 [此處設定所有設定]。

   b. 提供下列組建設定：

      - 執行階段 — 選擇 Python 3。
      - 建置命令 — 輸入**`pip install -r requirements.txt`**。
      - 啟動指令 — 輸入**`python server.py`**。
      - 連接埠 — 輸入**`8080`**。

   c. 選擇下一步。

> **Note**
>
> Python 3 運行時使用基本 Python 3 圖像和您的示例 Python 代碼構建一個碼頭圖像。然後，它會啟動執行此映像檔之容器執行個體的服務。



4. 設定您的服務。

   a. 在 [設定服務] 頁面的 [服務設定] 區段中，輸入服務名稱。

   b. 在環境變數下，選取新增環境變數。為環境變數提供下列值。

      - 來源 — 選擇純文字
      - 環境變數名稱 — **NAME**
      - 環境變數值 — 任何名稱 (例如，您的名字)。

> **Note**
>
> 範例應用程式會讀取您在此環境變數中設定的名稱，並在其網頁上顯示名稱。

c. 選擇下一步。

Configure service Info

Service settings

Service name

python-test

Virtual CPU & memory

1 vCPU | 2 GB

Environment variables — *optional* Info

Add environment variables in plain text or reference them from Secrets Manager and SSM Parameter Store. Update IAM Policies using the IAM Policy template given below to securely reference secrets and configurations as environment variables.

No environment variables have been configured.

Add environment variable

You can add up to 50 more items.

▶ IAM policy templates

▶ Auto scaling Info

Configure automatic scaling behavior.

▶ Health check Info

Configure load balancer health checks.

▶ Security Info

Specify an Instance role and an AWS KMS encryption key

▶ Networking Info

Configure the way your service communicates with other applications, services, and resources.

▶ Observability

Configure observability tooling.

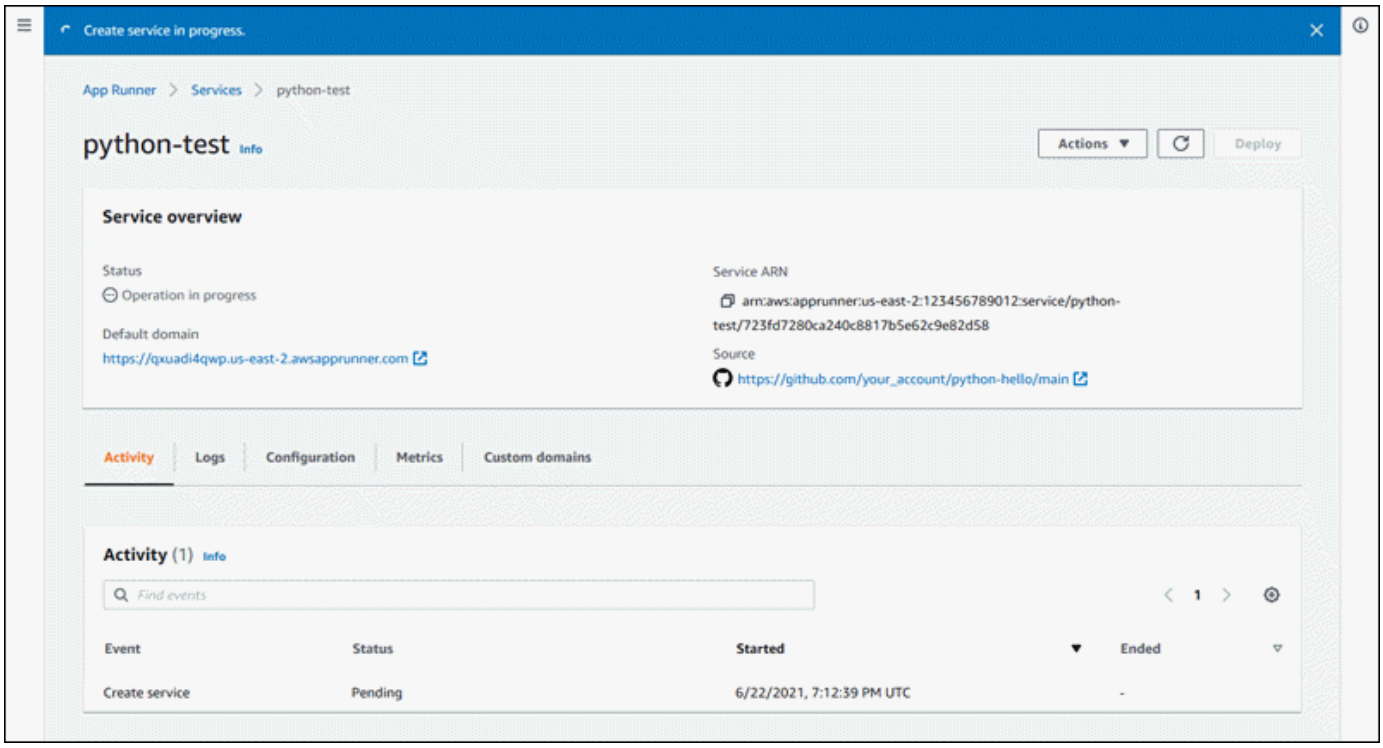
▼ Tags Info

Use tags to search and filter your resources, track your AWS costs, and control access permissions.

Tags — *optional*

5. 在 [檢閱並建立] 頁面上，確認您輸入的所有詳細資料，然後選擇 [建立並部署]。

   如果成功建立服務，則主控台會顯示服務儀表板，以及新服務的服務概觀。



6. 確認您的服務正在執行。

   a. 在服務儀表板頁面上，等待服務狀態為「執行中」。

   b. 選擇預設網域值 — 這是您服務網站的 URL。

> **Note**
>
> 為了增強應用程式執行器應用程式的安全性，*.awsapprunner.com 網域註冊在公用尾碼清單 (PSL) 中。為了進一步的安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

   顯示一個網頁：你好，**####**！

步驟 2：變更服務代碼

在此步驟中，您會變更儲存庫來源目錄中的程式碼。應用程式執行器 CI/CD 功能會自動建置和部署變更至您的服務。

變更您的服務代碼

1. 導航到您的示例存儲庫。
2. 編輯名為的檔案`server.py`。
3. 在指派給變數的運算式中`message`，`Hello`將文字變更為`Good morning`。
4. 儲存您的變更並提交至儲存庫。
5. 下列步驟說明變更 GitHub 儲存庫中的服務程式碼。

    a. 導航到您的示例 GitHub 存儲庫。

    b. 選擇要導覽`server.py`至該檔案的檔案名稱。

    c. 選擇 [編輯此檔案] (鉛筆圖示)。

    d. 在指派給變數的運算式中`message`，`Hello`將文字變更為`Good morning`。

e. 選擇 Commit changes (遞交變更)。

6. 新的提交開始為您的應用程序運行器服務部署。在服務儀表板頁面上，服務狀態會變更為 [作業進行中]。

   等待部署結束。在服務儀表板頁面上，服務狀態應變回執行中。

7. 確認部署是否成功：重新整理顯示服務網頁的瀏覽器索引標籤。

   該頁面現在顯示修改後的消息：早上好，**####**！

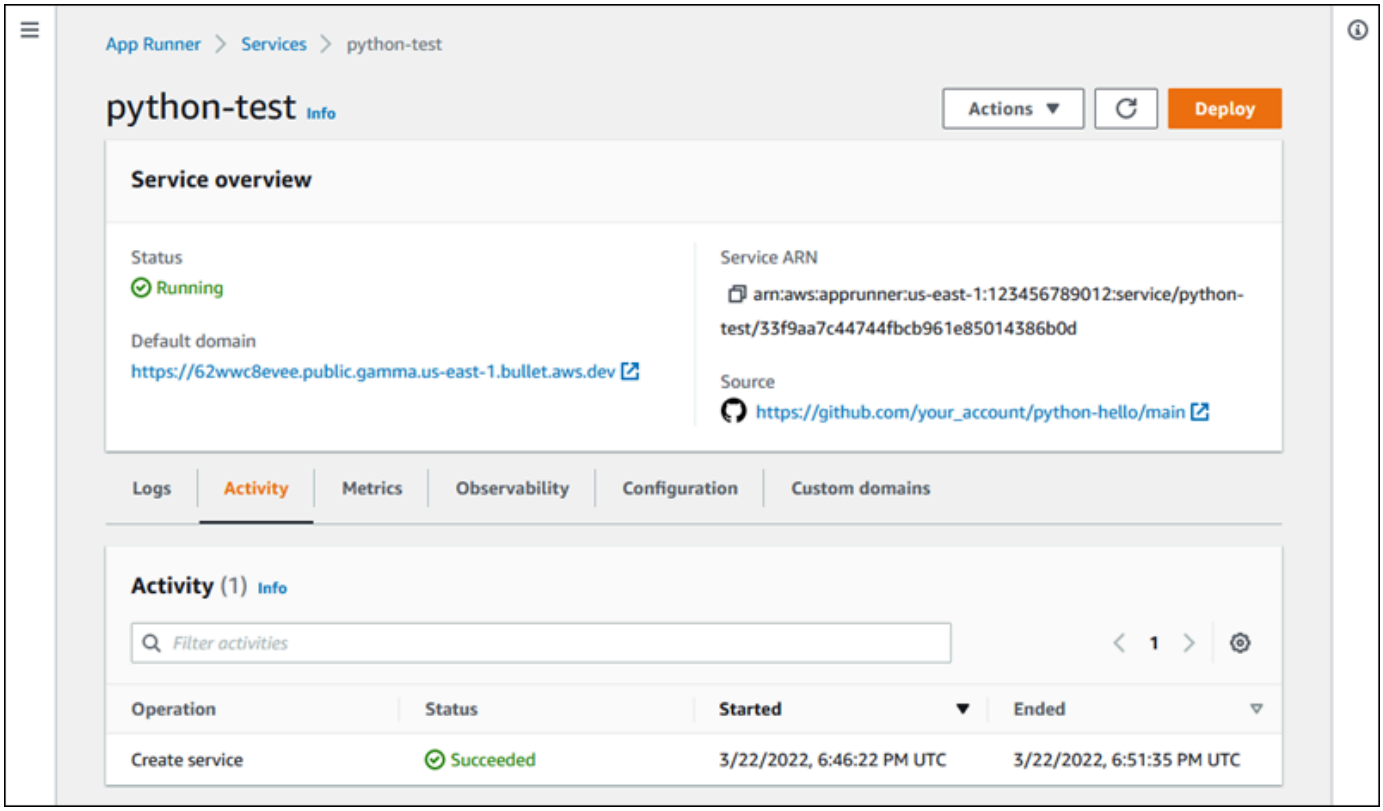
步驟 3：進行配置更改

在此步驟中，您會變更**NAME**環境變數值，以示範服務組態變更。

若要變更環境變數值

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

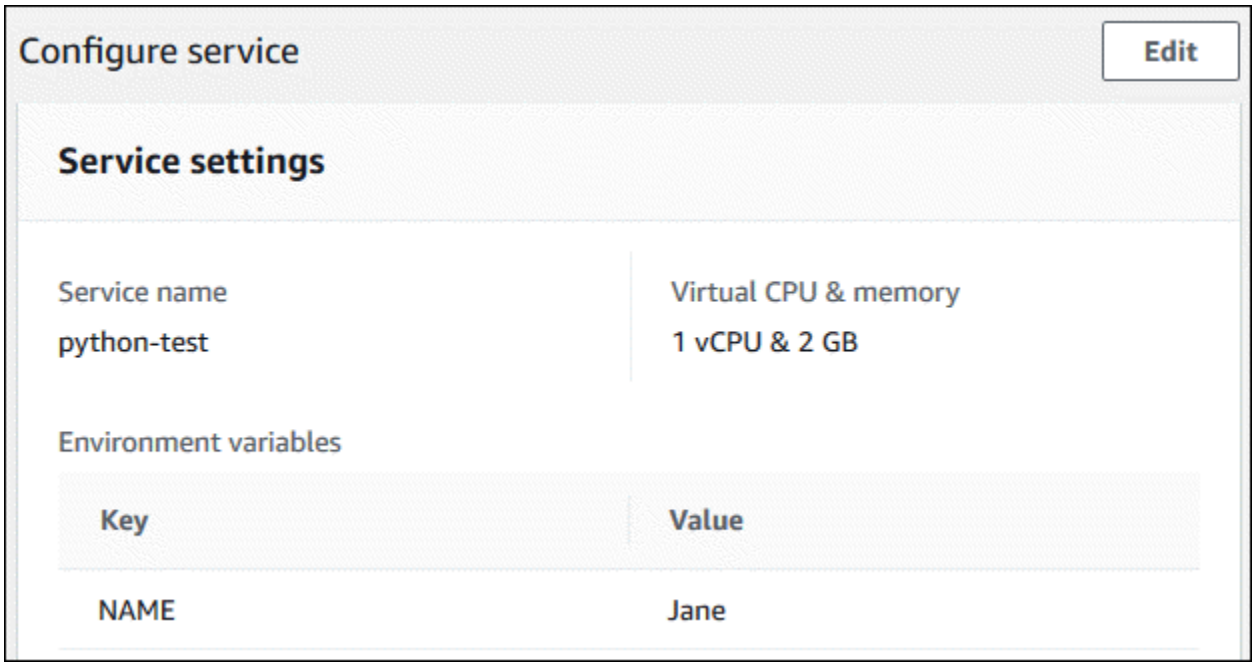
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇 [組態] 索引標籤。

   主控台會在數個區段中顯示您的服務組態設定。

4. 在「設定服務」區段中，選擇 「編輯」。



5. 對於具有索引鍵的環境變數**NAME**，請將值變更為其他名稱。

6. 選擇 Apply changes (套用變更)。

   應用程序運行器啟動更新過程。在服務儀表板頁面上，服務狀態會變更為 [作業進行中]。

7. 等待更新結束。在服務儀表板頁面上，服務狀態應變回執行中。

8. 確認更新是否成功：重新整理顯示服務網頁的瀏覽器索引標籤。

   該頁面現在顯示修改後的名稱：早上好，***###***！

步驟 4：檢視服務的記錄

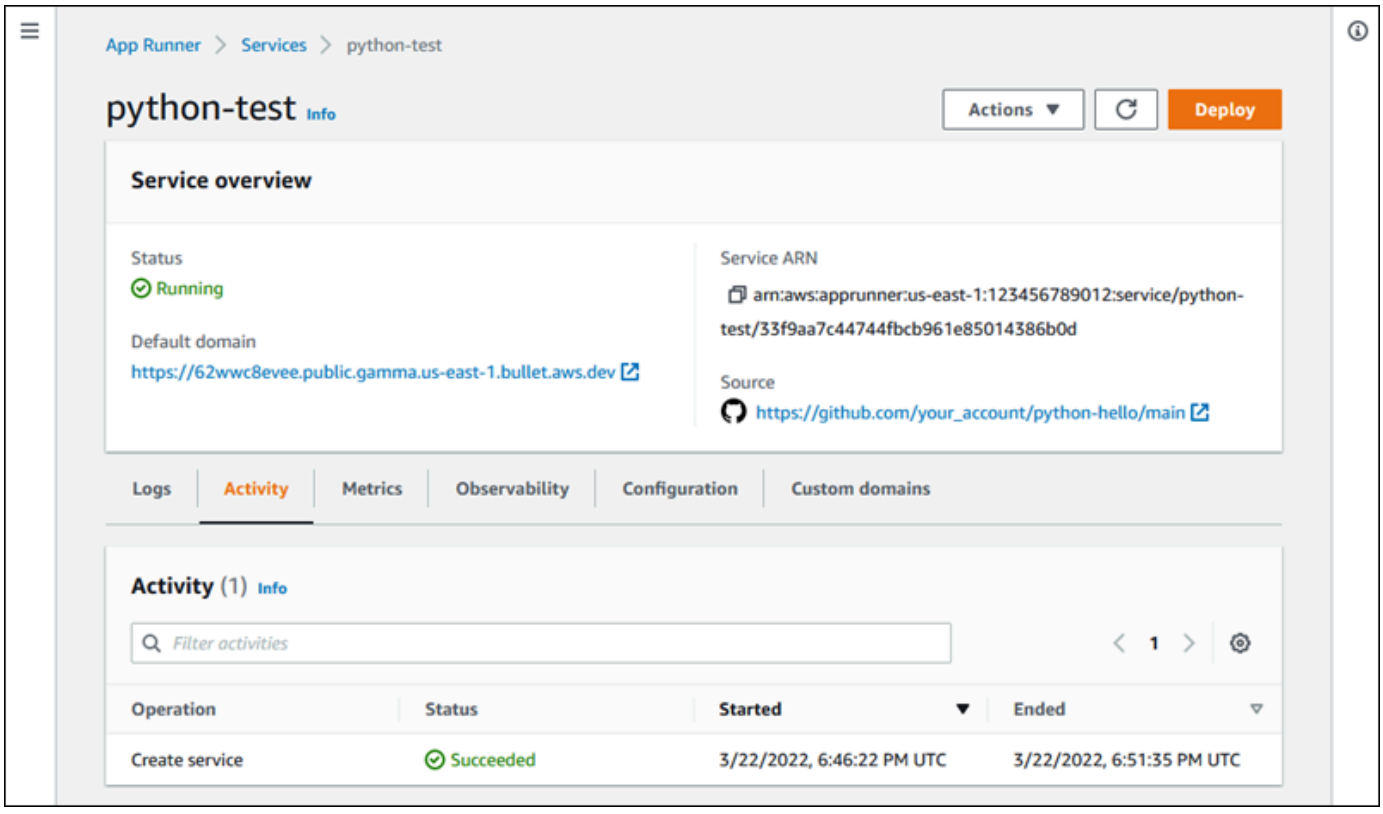
在此步驟中，您可以使用應用程式執行器主控台來檢視應用程式執行器服務的記錄檔。應用程式執行器會將日 CloudWatch 誌串流到 Amazon CloudWatch 日誌 (日誌)，並將其顯示在服務的儀表板上。如需有關應用程式執行程式記錄檔的資訊the section called “記錄檔 (CloudWatch 記錄檔)”，

若要檢視服務的記錄

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.

2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

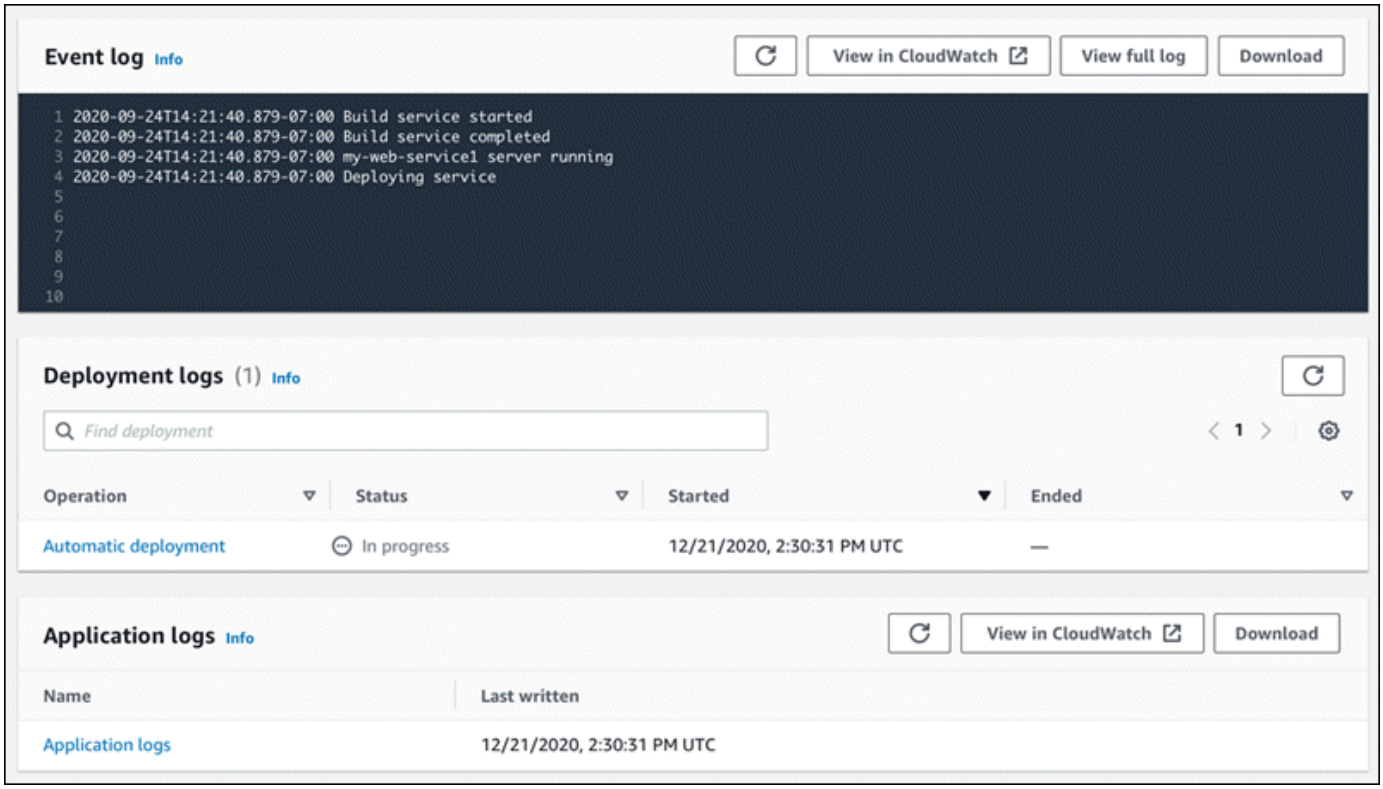
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇記錄檔索引標籤。

   主控台會在數個區段中顯示幾種類型的記錄檔：

   - 事件日誌 — 應用程序運行器服務生命週期中的活動。主控台會顯示最新的事件。
   - 部署記錄 — 來源儲存庫部署到您的應用程式執行器服務。主控台會針對每個部署顯示個別的記錄資料流。
   - 應用程式記錄檔 — 部署到應用程式執行器服務的 Web 應用程式輸出。主控台會將所有執行中執行個體的輸出結合到單一記錄串流中。



4. 若要尋找特定部署，請輸入搜尋詞彙，以縮小部署記錄清單的範圍。您可以搜尋表格中顯示的任何值。
5. 若要檢視記錄檔的內容，請選擇 [檢視完整記錄檔 (事件記錄檔)] 或記錄資料流名稱 (部署和應用程式記錄檔)。
6. 選擇 [下載] 以下載記錄檔。對於部署記錄串流，請先選取記錄資料流。
7. 選擇 [檢視 CloudWatch於] 開啟 CloudWatch 主控台，並使用其完整功能瀏覽您的 App Runner 服務記錄檔。對於部署記錄串流，請先選取記錄資料流。

> **Note**
>
> 如果您想要檢視特定執行個體的應用程式記錄檔，而不是合併的應用程式記錄檔，CloudWatch 主控台特別有用。

步驟 5：清除

您現在已經學習瞭如何創建應用程序運行器服務，查看日誌並進行一些更改。在此步驟中，您會刪除該服務以移除不再需要的資源。

若要刪除您的服務

1. 在服務儀表板頁面上，選擇 [動作]，然後選擇 [刪除服務]。
2. 在確認對話方塊中，輸入要求的文字，然後選擇 [刪除]。

   結果：主控台會導覽至「服務」頁面。您剛剛刪除的服務會顯示「刪除」的狀態。很短的時間後，它會從列表中消失。

也請考慮刪除您在本教學課程中建立的 GitHub 和 Bitbucket 連線。如需詳細資訊，請參閱 the section called “連線”。

下一步是什麼

現在您已經部署了第一個 App Runner 服務，請在下列主題中進一步了解：

- 建築與概念— 與應用程序運行器相關的體系結構，主要概念和 AWS 資源。
- 影像式服務和 基於代碼的服務 — 應用程式執行器可以部署的兩種應用程式來源類型。
- 開發應用程式執行器— 開發或移轉應用程式程式碼以部署至 App Runner 時應該知道的事項。
- 應用程式執行器— 使用應用程序運行器控制台管理和監視您的服務。
- 管理您的服務— 管理您的應用程式執行器服務的生命週期。
- 可觀測性— 通過監視指標，讀取日誌，處理事件，跟踪服務操作調用以及跟踪 HTTP 調用等應用程序事件，獲得 App Runner 服務操作的可見性。
- 應用程序運行器配置— 一種基於配置的方法，用於指定應用程序 Runner 服務的構建和運行時行為的選項。

- 應用程序亞軍 API— 使用應用程序運行器應用程序編程界面（API）創建，讀取，更新和刪除應用程序運行器資源。
- 安全— 在使用應用程序 Runner AWS 和其他服務時確保雲安全性的不同方式。

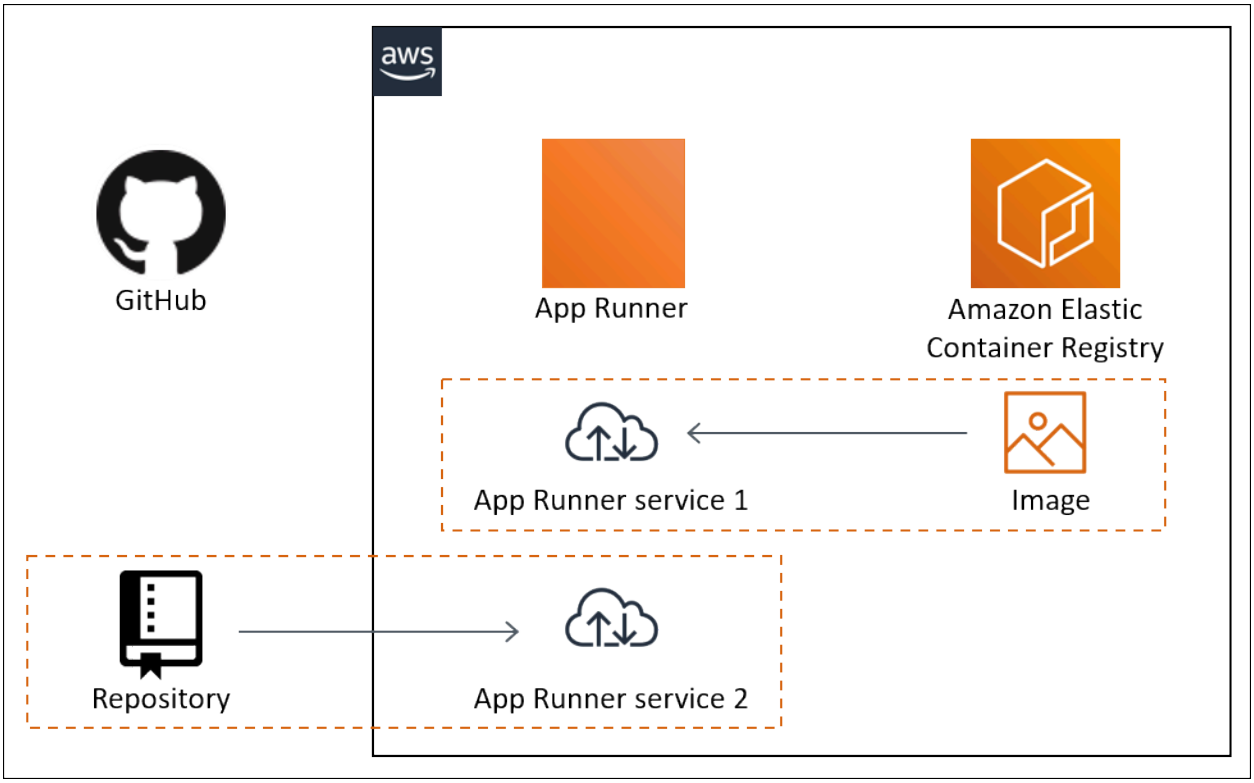
應用程序運行架構和概念

AWS App Runner 從儲存庫中取得原始程式碼或來源映像檔，並在中為您建立和維護執行中的 Web 服務 AWS 雲端。通常情況下，您只需要調用一個應用程序運行器操作 CreateService，以創建您的服務。

透過來源映像檔儲存庫，您可以提供 ready-to-use 容器映像檔，讓 App Runner 可以部署以執行 Web 服務。使用原始程式碼儲存庫，您可以提供建置和執行 Web 服務的程式碼和指示，並以特定的執行階段環境為目標。App Runner 支持多種編程平台，每個平台主要版本都具有一個或多個託管運行時。

此時，應用程式執行程式可以從 Bit GitHubbucket 或儲存庫擷取您的原始 Amazon 式碼，也可以從您的. AWS 帳戶

下圖顯示了應用程序運行器服務體系結構的概述。在圖中，有兩個範例服務：一個是從部署原始程式碼 GitHub，另一個從 Amazon ECR 部署來源映像。相同的流程適用於比特桶存儲庫。



應用程序運行器

以下是與在 App Runner 中運行的 Web 服務相關的關鍵概念：

- 應用程式執行器服務 — App Runner 使用的 AWS 資源，根據其原始程式碼儲存庫或容器映像來部署和管理您的應用程式。應用程序運行服務是應用程序的運行版本。如需建立服務的詳細資訊，請參閱the section called “建立”。
- 來源類型 — 您為部署 App Runner 服務所提供的來源存放庫類型：原始程式碼或來源映像檔。
- 儲存庫提供者 — 包含應用程式來源 (例如 GitHub，Bitbucket 或 Amazon ECR) 的儲存庫服務。
- 應用程式執行器連線 — 可讓 App Runner 存取儲存庫提供者帳戶 (例如， GitHub 帳戶或組織) 的 AWS 資源。如需連線的相關資訊，請參閱the section called “連線”。
- 執行階段 — 用於部署原始程式碼儲存庫的基本影像。App Runner 為不同的編程平台和版本提供了各種託管運行時。如需詳細資訊，請參閱 基於代碼的服務。
- 部署 — 將原始碼儲存庫 (程式碼或影像) 的版本套用至 App Runner 服務的動作。服務的第一次部署會在建立服務的過程中進行。稍後的部署可以透過下列兩種方式之一進行：
  - 自動部署 — CI/CD 功能。您可以將 App Runner 服務設定為自動建置 (針對原始程式碼)，並在儲存庫中顯示的每個應用程式版本部署。這可以是源代碼存儲庫中的新提交，也可以是源圖像存儲庫中的新映像版本。
  - 手動部署 — 您明確啟動的應用程式執行器服務的部署。
- 自定義域-您與應用程序運行器服務關聯的域。您的 Web 應用程序的用戶可以使用此域訪問您的 Web 服務，而不是默認的應用程序運行器子域。如需詳細資訊，請參閱 the section called “自訂網域名稱”。

> **Note**
>
> 為了增強應用程式執行器應用程式的安全性，*.awsapprunner.com 網域註冊在公用尾碼清單 (PSL) 中。為了進一步的安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

- 維護 — App Runner 偶爾在執行 App Runner 服務的基礎結構上執行的活動。維護進行中時，服務狀態會暫時變更為OPERATION_IN_PROGRESS（主控台中的作業進行中）幾分鐘。在此期間，會封鎖對您服務的動作 (例如部署、組態更新、暫停/繼續或刪除)。當服務狀態回到時，請在幾分鐘後再次嘗試此動作RUNNING。

> **Note**
>
> 如果您的操作失敗，並不意味著您的應用程序運行器服務已關閉。您的應用程序處於活動狀態並繼續處理請求 您的服務不太可能遇到任何停機時間。

特別是，如果 App Runner 檢測到託管服務的基礎硬件中的問題，則會遷移您的服務。為了防止任何服務停機時間，App Runner 會將您的服務部署到一組新的執行個體，並將流量轉移給它們（藍綠色部署）。您可能偶爾會看到費用略有暫時增加。

應用程序運行器支持

設定 App Runner 服務時，您可以指定要配置給服務的虛擬 CPU 和記憶體組態。您可以根據選取的運算組態付費。如需定價的詳細資訊，請參閱 AWS Resource Groups 定價。

下表提供 App Runner 支援的 vCPU 和記憶體組態的相關資訊：

CPU	記憶體
0.25 vCPU	0.5 GB
0.25 vCPU	1 GB
0.5 vCPU	1 GB
1 vCPU	2 GB
1 vCPU	3 GB
1 vCPU	4 GB
2 vCPU	4 GB
2 vCPU	6 GB
4 vCPU	8 GB
4 vCPU	10 GB

CPU	記憶體
4 vCPU	十二 GB

應用程序運行器

當您使用 App Runner 時，您可以在 AWS 帳戶. 這些資源可用來存取您的程式碼和管理您的服務。

下表提供這些資源的概觀：

資源名稱	Description
Service	代表應用程式的執行中版本。本指南的其餘部分內容描述了服務類型、管理、組態和監控。 ARN：arn:aws:apprunner: *region*:*account-id* :service/ *service-name* [/*service-id*]
Connection	為您的 App Runner 服務提供訪問存儲在第三方提供商的私有存儲庫。作為單獨的資源存在，用於跨多個服務共享。如需連線的相關資訊，請參閱the section called “連線”。 ARN：arn:aws:apprunner: *region*:*account-id* :connection/ *connection-name* [/*connection-id*]
AutoScalingConfiguration	為您的 App Runner 服務提供控制應用程式自動擴展的設定。作為單獨的資源存在，用於跨多個服務共享。如需自動擴展的相關資訊，請參閱the section called “自動擴展”。 ARN：arn:aws:apprunner: *region*:*account-id* :autoscalingconfiguration/ *config-name* [/*config-revision* [/*config-id*]]
ObservabilityConfiguration	為您的應用程式執行器服務設定其他應用程式可觀察性功能。作為單獨的資源存在，用於跨多個服務共享。若要取得有關觀察性組態的更多資訊，請參閱〈the section called “可觀測性配置”

資源名稱	Description
	ARN：arn:aws:apprunner: *region*:*account-id* :observability configuration/ *config-name* [/*config-revision* [/*config-id*]]
VpcConnector	為您的應用程式執行器服務設定 VPC 設定。作為單獨的資源存在，用於跨多個服務共享。如需 VPC 功能的詳細資訊，請參閱the section called “傳出流量”。 ARN：arn:aws:apprunner: *region*:*account-id* :vpcconnector/ *connector-name* [/*connector-revision* [/*connector-id*]]
VpcIngressConnection	它是用來設定傳入流量的 AWS App Runner 資源。它會在 VPC 介面端點和應用程式執行器服務之間建立連線，讓您的應用程式執行器服務只能在 Amazon VPC 內存取。如需 VPC 功能的詳細資訊IngressConnection，請參閱the section called “啟用私有端點”。 ARN：arn:aws:apprunner: *region*:*account-id* :vpcingressconnection/ *vpc-ingress-connection-name* [/*connector-id*]]

應用運行器資源配額

AWS 對您的帳號強加一些配額 (也稱為限制)，以便在每個 AWS 區域帳號中使用 AWS 資源。下表列出與應用程式執行器資源相關的配額。配額也會列在中的AWS App Runner 端點和配額中AWS 一般參考。

資源配額	Description	預設值	可調節？
Services	您可以在帳戶中為每個服務創建的最大數量 AWS 區域。	30	✓ 是
Connections	您可以在帳戶中為每個連接創建的最大連接數 AWS 區域。您可以在多個服務中使用單一連線。	10	✓ 是

資源配額		Description	預設值	可調節？
Auto scaling configurations	名稱	您在帳戶中針對每個 auto 擴展設定建立的唯一名稱的最大數量 AWS 區域。您可以在多個服務中使用單一 auto 擴展設定。	10	✓ 是
	每個名稱的修訂	您可以在帳戶中為每個 AWS 區域 唯一名稱建立的 auto 擴展組態修訂數目上限。您可以在多個服務中使用單一 auto 擴展組態修訂版本。	5	X 否
Observability configurations	名稱	您在每 AWS 區域個帳戶中建立的可觀察性設定中可以擁有的唯一名稱的最大數目。您可以在多個服務中使用單一可觀察性組態。	10	✓ 是
	每個名稱的修訂	您可以在帳戶中為每 AWS 區域 個唯一名稱建立的可觀察性組態修訂數目上限。您可以在多個服務中使用單一可觀察性組態修訂版本。	10	X 否
VPC connectors		您可以在帳戶中為每 AWS 區域個連接器建立的最大 VPC 連接器數目。您可以在多個服務中使用單一 VPC 連接器。	10	✓ 是
VPC Ingress Connection		您可以在帳戶中為每個連線建立的最大 VPC 輸入連線數目。 AWS 區域您可以使用單一 VPC 入口連線來存取多個應用程式執行器服務。	1	X 否

大多數配額都是可調整的，您可以要求提高配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求提高配額。

基於源圖像的應用程序運行器服務

您可以使 AWS App Runner 用根據兩種根本不同類型的服務來源來建立和管理服務：原始程式碼和來源映像檔。無論源類型如何，App Runner 都會負責啟動，運行，擴展和負載平衡您的服務。您可以使用應用程式執行器的 CI/CD 功能來追蹤來源映像檔或程式碼的變更。當 App Runner 發現更改時，它會自動構建（用於源代碼）並將新版本部署到您的應用程序運行器服務。

本章討論以來源映像檔為基礎的服務。如需有關以原始碼為基礎之服務的資訊，請參閱基於代碼的服務。

來源映像檔是儲存在映像儲存庫中的公用或私人容器映像檔。您將 App Runner 指向映像，並啟動基於此映像執行容器的服務。沒有構建階段是必要的。相反，您提供了一個 ready-to-deploy 圖像。

影像儲存庫提供

App Runner 支援下列映像儲存庫提供者：

- Amazon Elastic Container Registry (Amazon ECR) — 存儲. AWS 帳戶
- Amazon 彈性容器註冊表公共（Amazon ECR 公共）— 存儲可公開讀取的映像。

提供者使用案例

- 在您的帳戶中使用存儲在 Amazon ECR 中的 AWS 圖像
- 在不同帳戶中使用存放在 Amazon ECR 中的 AWS 映像檔
- 使用存儲在 Amazon ECR 公共圖像

在您的帳戶中使用存儲在 Amazon ECR 中的 AWS 圖像

Amazon ECR 將映像存儲在存儲庫中。有私人和公共存儲庫。若要從私有儲存庫將映像部署到應用程式執行器服務，應用程式執行器需要從 Amazon ECR 讀取映像的權限。要將該權限授予應用程序運行器，您需要向應用程序運行器提供訪問角色。這是具有必要 Amazon ECR 動作許可權的 AWS Identity and Access Management (IAM) 角色。當您使用 App Runner 主控台建立服務時，您可以選擇帳戶中的現有角色。或者，您可以使用 IAM 主控台建立新的自訂角色。或者，您可以選擇 App Runner 主控台，根據受管政策為您建立角色。

當您使用應用程式執行器 API 或時 AWS CLI，您會完成兩個步驟的程序。首先，您可以使用 IAM 主控台建立存取角色。您可以使用 App Runner 提供的受管理原則，或輸入您自己的自訂權限。然後，您可以使用 CreateServiceAPI 動作在服務建立期間提供存取角色。

如需「應用程式執行器」服務建立的相關the section called “建立”資訊，

在不同帳戶中使用存放在 Amazon ECR 中的 AWS 映像檔

建立應用程式執行器服務時，您可以使用存放在 Amazon ECR 儲存庫中的映像，該影像屬於您的服務所在 AWS 帳戶以外的帳戶。使用跨帳戶映像檔時，除了上一節中列出的有關同一帳戶映像檔的考量之外，還有一些其他考量。

- 跨帳戶儲存庫應該有附加原則。存放庫原則為您的存取角色提供讀取存放庫中影像的權限。為此目的使用下列原則。取代*access-role-arn*為您存取角色的 Amazon 資源名稱 (ARN)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "access-role-arn"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}
```

  如需將儲存庫政策附加至 Amazon ECR 儲存庫的相關資訊，請參閱 Amazon 彈性容器登錄使用者指南中的設定儲存庫政策聲明。
- 應用程式執行器不支援自動部署 Amazon ECR 映像檔在與您的服務所在帳戶不同的帳戶中。

使用存儲在 Amazon ECR 公共圖像

Amazon ECR 公共存儲可公開讀取的圖像。這些是 Amazon ECR 和 Amazon ECR 公共之間的主要區別，您應該知道在應用程序運行器服務的情況下：

- Amazon ECR 公開映像檔可公開讀取。當您根據 Amazon ECR 公開映像建立服務時，不需要提供存取角色。儲存庫不需要附加任何原則。
- 應用程式執行器不支援 Amazon ECR 公開映像的自動 (連續) 部署。

直接從 Amazon ECR 公共啟動服務

您可以直接啟動在 Amazon ECR 公用畫廊上託管的相容 Web 應用程式的容器映像，做為在應用程式執行器上執行的網路服務。瀏覽圖庫時，請在圖庫頁面上尋找使用應用程式執行器啟動的影像。具有此選項的圖像與應用程序運行器兼容。如需有關圖庫的詳細資訊，請參閱 Amazon ECR 公開使用者指南中的使用 Amazon ECR 公用圖庫。

啟動圖庫影像作為應用程式執行器服務

1. 在影像的圖庫頁面上，選擇「使用應用程式執行器啟動」。

   結果：App Runner 主控台會在新的瀏覽器索引標籤中開啟。主控台會顯示 [建立服務] 精靈，並預先填入大部分必要的新服務詳細資料。

2. 如果您要在主控台顯示的 AWS 區域以外的區域中建立服務，請選擇主控台標頭上顯示的 [區域]。然後，選取另一個「區域」。
3. 在連接埠中，輸入影像應用程式監聽的連接埠號碼。您通常可以在圖像的圖庫頁面上找到它。
4. 選擇性地變更任何其他組態詳細資訊。
5. 選擇 [下一步]，檢閱設定，然後選擇 [建立和部署]。

圖像示例

應用程式執行器團隊會在 Amazon ECR 公共圖庫中維護hello-app-runner範例影像。您可以使用此範例開始建立以影像為基礎的 App Runner 服務。如需詳細資訊，請參閱hello-app-runner。

基於源代碼的應用程序運行服務

您可以使 AWS App Runner 用根據兩種根本不同類型的服務來源來建立和管理服務：原始程式碼和來源映像檔。無論源類型如何，App Runner 都會負責啟動，運行，擴展和負載平衡您的服務。您可以使用應用程式執行器的 CI/CD 功能來追蹤來源映像檔或程式碼的變更。當 App Runner 發現更改時，它會自動構建（用於源代碼）並將新版本部署到您的應用程序運行器服務。

本章討論以原始碼為基礎的服務。如需有關以來源影像為基礎之服務的資訊，請參閱影像式服務。

源代碼是應用程序 Runner 為您構建和部署的應用程序代碼。您可以將 App Runner 指向程式碼儲存庫中的原始碼目錄，然後選擇對應於程式設計平台版本的合適執行階段。App Runner 會根據執行階段的基本影像和您的應用程式程式碼來建置映像檔。然後，它會啟動根據此映像執行容器的服務。

App Runner 提供方便的特定於平台的託管運行時。這些執行階段中的每一個都會從您的原始程式碼建立容器映像檔，並將語言執行階段相依性新增至映像中。您不需要提供容器配置和構建指令，例如 Dockerfile。

本章的副主題討論 App Runner 支援的各種平台，這些平台可為不同的程式設計環境和版本提供受管理的執行階段。

主題

- 原始碼儲存庫提供者
- 來源目錄
- 應用程式執行器管理
- 託管運行時版本和應用程序運行器構建
- 使用 Python 平台
- 使用 Node.js 平台
- 使用 Java 平台
- 使用 .NET 平台
- 使用 PHP 平台
- 使用 Ruby 平台
- 使用 Go 平台

原始碼儲存庫提供者

應用程序 Runner 通過從源代碼存儲庫中讀取源代碼來部署源代碼。應用程序運行器支持兩個源代碼存儲庫提供商：GitHub和 Bitbucket。

從原始程式碼儲存庫提供者部署

要從源代碼存儲庫將源代碼部署到應用程序運行器服務，應用程序運行器建立到它的連接。當您使用 App Runner 主控台建立服務時，您會提供連線詳細資料和來源目錄供 App Runner 部署您的原始程式碼。

連線

您可以提供連線詳細資訊，做為服務建立程序的一部分。當您使用應用程式執行器 API 或時 AWS CLI，連線是個別的資源。首先，您可以使用 CreateConnectionAPI 動作建立連線。然後，您可以在建立服務期間使用 CreateServiceAPI 動作提供連線的 ARN。

來源目錄

當您建立服務時，您也會提供來源目錄。默認情況下，App Runner 使用存儲庫的根目錄作為源目錄。來源目錄是儲存應用程式原始程式碼和組態檔案的原始程式碼儲存庫中的位置。建置和啟動指令也會從來源目錄執行。當您使用應用程式執行器 API 或建立或更新服務時，您可 AWS CLI 以在CreateService和 UpdateServiceAPI 動作中提供來源目錄。如需詳細資訊，請參閱下面的來源目錄章節。

如需建立應用程式執行器服務的詳細資訊，請參閱the section called “建立”。如需應用程式執行器連線的詳細資訊，請參閱the section called “連線”。

來源目錄

當您建立 App Runner 服務時，您可以提供來源目錄，以及儲存庫和分支。將「來源目錄」欄位的值設定為儲存應用程式原始程式碼和組態檔的儲存庫目錄路徑。App Runner 會從您提供的來源目錄路徑執行建置和啟動命令。

從根存放庫目錄輸入來源目錄路徑的絕對值。如果您未指定值，它會預設為儲存庫頂層目錄，也稱為儲存庫根目錄。

除了頂層存放庫目錄之外，您還可以選擇提供不同的來源目錄路徑。這支持 monorepo 存儲庫架構，這意味著多個應用程序的源代碼存儲在一個存儲庫中。要從單個 monorepo 創建和支持多個應用程序運行器服務，請在創建每個服務時指定不同的源目錄。

> **Note**
>
> 如果您為多個 App Runner 服務指定相同的來源目錄，這兩個服務都會個別部署和操作。

如果您選擇使用組`apprunner.yaml`態檔來定義服務參數，請將其放置在存放庫的來源目錄資料夾中。

如果部署觸發程序選項設定為 [自動]，則您在來源目錄中確認的變更將觸發自動部署。 只有來源目錄路徑中的變更才會觸發自動部署。請務必瞭解來源目錄的位置如何影響自動部署的範圍。如需詳細資訊，請參閱中的自動部署部署方法。

> **Note**
>
> 如果您的 App Runner 服務使用 PHP 託管運行時，並且您想要指定默認根存儲庫以外的源目錄，則使用正確的 PHP 運行時版本非常重要。如需詳細資訊，請參閱 使用 PHP 平台。

應用程式執行器管理

應用程序運行器託管平台為各種編程環境提供託管運行時。每個託管運行時都可以根據編程語言或運行時環境的版本輕鬆構建和運行容器。當您使用受管理的執行階段時，App Runner 會從受管理的執行階段映像開始。此映像檔是以 Amazon Linux Docker 映像為基礎，並包含語言執行階段套件，以及一些工具和常用的相依性套件。App Runner 使用此託管運行時映像作為基本映像，並添加應用程序代碼以構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

託管運行時版本和應用程序運行器構建

應用程式執行器現在為您的應用程式提供更新的建置程序 它目前會呼叫在管理執行階段上執行之服務的新組建 (最後一次於 20 23 年 12 月 29 日發行) Python 3.11 和 Node.js 18 上執行。這個經過修訂的建置程序更快速且更有效率。它還可以創建一個佔用空間較小的最終映像，該映像僅包含運行應用程序所需的源代碼，構建成品和運行時。

我們將較新的建置程序稱為經過修訂的 App Runner 組建，而原始建置程序則稱為原始 App Runner 組建。為了避免對早期版本的運行時平台進行更改，App Runner 僅將修訂後的構建應用於特定的運行時版本，通常是新發布的主要版本。

我們已經在`apprunner.yaml`組態檔案中引入了一個新的元件，使修訂的組建向後相容於非常特定的使用案例，並提供更大的彈性來設定應用程式的組建。這是可選`pre-run`參數。在接下來的章節中，我們會說明何時使用此參數，以及其他有關組建的實用資訊。

下表說明哪個版本的 App Runner 版本適用於特定的託管運行時版本。我們將繼續更新此文件，讓您瞭解我們目前的執行階段。

平台	原始構建	修訂版組建
Python – 版本資訊	• Python 3.8 • Python 3.7	• Python (!)
Node.js – 版本資訊	• Node.js 16 • Node.js 14 • Node.js 12	• Node.js 18
Corretto — 版本資訊	• Corretto 11 • Corretto 8	
.NET – 版本資訊	• .NET 6	
PHP – 版本資訊	• PHP 8.1	
Ruby – 版本資訊	• 紅寶石	
Go – 版本資訊	• Go 1	

> **Important**
>
> Python 3.11 — 對於使用 Python 3.11 託管運行時的服務的構建配置，我們有具體的建議。如需詳細資訊，請參閱 Python 平台主題特定執行階段版本的編號說明中的。

更多有關應用程序運行器構建和遷移

當您將應用程式移轉至使用修訂版本的較新執行階段時，可能需要稍微修改組建組態。

若要提供移轉考量的內容，我們會先說明原始 App Runner 組建和修訂版組建的高階程序。接下來我們將介紹您服務的特定屬性，這些屬性可能需要進行一些配置更新。

原始應用程序運行器構建

原始的應用程序運行應用程序構建過程利用了該 AWS CodeBuild 服務。初始步驟以 CodeBuild 服務策劃的影像為基礎。Docker 構建過程遵循使用適用的應用程序運行時管理運行時映像作為基本映像。

一般步驟如下：

1. 在 CodeBuild策劃的圖像中運行`pre-build`命令。

   這些`pre-build`命令是可選的。它們只能在`apprunner.yaml`組態檔案中指定。
2. 使用 CodeBuild 先前步驟中的相同影像執行`build`指令。

   這些`build`命令是必需的。它們可以在應用程序運行器控制台，應用程序運行器 API 或`apprunner.yaml`配置文件中指定。
3. 運行 Docker 構建以根據特定平台和運行時版本的 App Runner 託管運行時映像生成圖像。
4. 從我們在步驟 2 中生成的圖像複製`/app`目錄。目的地是基於我們在步驟 3 中生成的 App Runner 管理運行時映像的圖像。
5. 在生成的 App Runner 託管運行時映像上再次運行`build`命令。我們再次執行建置命令，從我們在步驟 4 中複製到它的`/app`目錄中的原始程式碼產生建置成品。此映像稍後將由應用程序運行器部署在容器中運行您的 Web 服務。

   這些`build`命令是必需的。它們可以在應用程序運行器控制台，應用程序運行器 API 或`apprunner.yaml`配置文件中指定。
6. 從步驟 2 執行 CodeBuild 影像中的`post-build`指令。

這些post-build命令是可選的。它們只能在apprunner.yaml組態檔案中指定。

建置完成之後，App Runner 會部署步驟 5 產生的 App Runner 管理執行階段映像，以便在容器中執行 Web 服務。

修訂後的應用程式執行器

修訂後的建置程序比上一節所述的原始建置程序更快速且更有效率。它消除了在先前版本構建中發生的構建命令的重複。它還可以創建一個佔用空間較小的最終映像，該映像僅包含運行應用程序所需的源代碼，構建成品和運行時。

此構建過程使用 Docker 多階段構建。一般處理步驟如下：

1. 構建階段-啟動 docker 構建過程，該過程在 App Runner 構建映像上執行pre-build和build命令。
   a. 將應用程式原始程式碼複製到/app目錄中。

      > **Note**
      >
      > 此/app目錄被指定為 Docker 構建的每個階段的工作目錄。

   b. 執行 pre-build 命令。

      這些pre-build命令是可選的。它們只能在apprunner.yaml組態檔案中指定。

   c. 運行build命令。

      這些build命令是必需的。它們可以在應用程序運行器控制台，應用程序運行器 API 或apprunner.yaml配置文件中指定。
2. 封裝階段 — 產生最終的客戶容器映像，這也是以 App Runner 執行影像為基礎。
   a. 將/app目錄從先前的「構建」階段複製到新的「運行」映像。這包括您的應用程式原始程式碼和先前階段的建置成品。
   b. 運行pre-run命令。如果您需要使用build指令來修改/app目錄外的執行階段影像，請將相同或必要的指令新增至apprunner.yaml組態檔案的這個區段。

      這是為了支援修訂後的 App Runner 組建而引入的新參數。

      這些pre-run命令是可選的。它們只能在apprunner.yaml組態檔案中指定。

> **備註**
>
> - 只有修訂版本才支援這些pre-run指令。如果您的服務使用使用原始組建的執行階段版本，請勿將它們加入組態檔案。
> - 如果您不需要使用build命令修改目/app錄外的任何內容，則不需要指定pre-run命令。

3. 建置後階段 — 此階段會從「建置」階段繼續執行，並執行post-build命令。
   a. 運行目/app錄內的post-build命令。

      這些post-build命令是可選的。它們只能在apprunner.yaml組態檔案中指定。

構建完成後，應用程序 Runner 然後部署運行映像以在容器中運行 Web 服務。

> **Note**
>
> 配置構建過程apprunner.yaml時，請不要誤導到的「運行」部分中的env條目。即使在步驟 2 (b) 中參考的pre-run命令參數位於 [執行] 區段中，但請勿使用 [執行] 區段中的env參數來設定您的組建。這些命pre-run令僅引用配置文件的「構建」部分中定義的env變量。如需詳細資訊，請參閱「應用程式執行器」設定檔一章執行區段中的。

移轉考量的服務需求

如果您的應用程式環境具有這兩項需求中的任何一項，則您需要透過新增pre-run命令來修改您的組建設定。

- 如果您需要使用build命令修改/app目錄之外的任何內容。
- 如果您需要運行兩次build命令來創建所需的環境。這是一個非常不尋常的要求。絕大多數構建不會這樣做。

/app目錄外的修改

- 修訂後的 App Runner 組建會假設您的應用程式在/app目錄之外沒有相依性。
- 您隨apprunner.yaml檔案、App Runner API 或應用程式執行器主控台提供的命令必須在/app目錄中產生組建成品。

- 您可以修改`pre-build`、和`post-build`指令`build`，以確保所有組建加工品都位於目`/app`錄中。
- 如果您的應用程式需要組建進一步修改服務所產生的映像，則在`/app`目錄之外，您可以使用中的新`pre-run`命令`apprunner.yaml`。如需詳細資訊，請參閱 使用配置文件設置應用程序運行器服務選項。

運行兩次**`build`**命令

- 原始的應用程序運行器構建運行`build`命令兩次，首先在步驟 2 中，然後在步驟 5 中再次運行命令。修訂後的 App Runner 構建可以補救這種冗餘，並且只運行一次`build`命令。如果您的應用程式對於執行兩次`build`命令的不尋常需求，則修訂後的 App Runner 組建會提供使用`pre-run`參數再次指定和執行相同命令的選項。這樣做會保留相同的雙重建置行為。

使用 Python 平台

AWS App Runner Python 平台提供受管理的執行階段。每個執行階段都可以使用以 Python 版本為基礎的 Web 應用程式輕鬆建置和執行容器。當您使用 Python 運行時，應用程序運行器以託管 Python 運行時映像開始。此映像檔是以 Amazon Linux Docker 映像檔為基礎，其中包含 Python 版本的執行階段套件，以及一些工具以及常用的相依性套件。App Runner 使用此託管運行時映像作為基本映像，並添加應用程序代碼以構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 Python 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

Python 執行階段的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`3.8.5`

下列範例會示範版本鎖定：

- `3.8`-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。

- `3.8.5`— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

主題

- Python 運行時配置
- 特定執行階段版本的編號說明
- Python 運行時示例
- Python 運行時發布信息

Python 運行時配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得設定。

特定執行階段版本的編號說明

> **Note**
>
> 應用程式執行程式現在會針對以下執行階段版本的應用程式執行更新的建置程序：Python 3.11 和 Node.js 18。如果您的應用程式在其中一個執行階段版本上執行，請參閱以託管運行時版本和應用程序運行器構建取得有關修訂建置程序的詳細資訊。使用所有其他執行階段版本的應用程式不會受到影響，而且會繼續使用原始建置程序。

Python 3.11（修訂後的應用程序運行器構建）

在受管理的 Python 3.11 執行階段的應用程式中使用下列設定。

- 將 Top 部分中的`runtime`密鑰設置為 `python311`

  Example

  ```
  runtime: python311
  ```

- 使用`pip3`而不是安裝相依性。`pip`
- 使用`python3`解釋器代替`python`。
- 以`pre-run`指令執行`pip3`安裝程式。Python 的`/app`目錄之外安裝依賴關係。由於 App Runner 會針對 Python 3.11 執行修訂的應用程式執行程式組建，因此透過`apprunner.yaml`檔案的 [建置] 區段中的命令安裝在`/app`目錄外的任何項目都將遺失。如需詳細資訊，請參閱 修訂後的應用程式執行器。

  Example

  ```
  run:
    runtime-version: 3.11
    pre-run:
      - pip3 install pipenv
      - pipenv install
      - python3 copy-global-files.py
    command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  ```

如需詳細資訊，請參閱本主題稍後的 Python 3.11 擴充設定檔範例。

Python 運行時示例

下面的例子顯示了用於構建和運行 Python 服務的應用程序運行配置文件。最後一個範例是您可以部署到 Python 執行階段服務的完整 Python 應用程式的原始程式碼。

> **Note**
>
> 這些範例中使用的執行階段版本為 *3.7.7* 和 *3.* 11。您可以將其替換為您要使用的版本。如需最新支援的 Python 執行階段版本，請參閱the section called “版本資訊”。

最小的 Python 配置文件

此範例顯示可與 Python 管理執行階段搭配使用的最小組態檔案。如需 App Runner 使用最小組態檔案所做的假設，請參閱the section called “組態檔案範例”。

Python 3.11 使用`pip3`和命`python3`令。如需詳細資訊，請參閱本主題稍後的 Python 3.11 擴充設定檔範例。

Example 阿普魯人. 羊

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

擴展 Python 配置文件

這個例子顯示了使用 Python 託管運行時的所有配置鍵。

> **Note**
>
> 這些範例中使用的執行階段版本為 *`3.7.7`*。您可以將其替換為您要使用的版本。如需最新支援的 Python 執行階段版本，請參閱the section called “版本資訊”。
> Python 3.11 使用`pip3`和命`python3`令。如需詳細資訊，請參閱本主題稍後的 Python 3.11 擴充設定檔範例。

Example 阿普魯人. 羊

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/test-lib.tar.gz -O - | tar
 -xz
```

```
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

擴展 Python 配置文件-Python 3.11（使用修訂後的構建）

此範例顯示在中使用 Python 3.11 受管理執行階段的所有組態索引鍵。`apprunner.yaml`這個例子包括一個`pre-run`部分，因為這個版本的 Python 使用修訂的應用程序運行器構建。

該`pre-run`參數僅由修訂後的應用程序運行器構建支持。如果您的應用程式使用原始 App Runner 組建支援的執行階段版本，請勿在設定檔中插入此參數。如需詳細資訊，請參閱 [託管運行時版本和應用程序運行器構建](#)。

> **Note**
>
> 這些範例中使用的執行階段版本為 *3.11*。您可以將其替換為您要使用的版本。如需最新支援的 Python 執行階段版本，請參閱[the section called “版本資訊”](#)。

Example 阿普魯人. 羊

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

完整的 Python 應用程序源

此範例顯示可部署至 Python 執行階段服務的完整 Python 應用程式的原始程式碼。

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
    server.serve_forever()
```

Example 阿普魯人. 羊

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
run:
  command: python server.py
```

Python 運行時發布信息

本主題列出了應用程序運行器支持的 Python 運行時版本的完整詳細信息。

支持的運行時版本-修訂後的應用程序

執行時間名稱	次要版本	包含的套件
Python (蟒蛇)	3.11.9	方形精簡版
	3.11.8	方形精簡版
	3.11.7	方形精簡版

備註

- Python 3.11 — 對於使用 Python 3.11 託管運行時的服務的構建配置，我們有具體的建議。如需詳細資訊，請參閱 Python 平台主題特定執行階段版本的編號說明中的。
- App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
Python 3（蟒蛇 3）	3.8.16	方形精簡版
	3.7.16	方形精簡版
	3.8.15	方形精簡版
	3.8.5	方形精簡版
	3.7.15	方形精簡版
	3.7.10	方形精簡版

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 Node.js 平台

AWS App Runner Node.js 平台提供受管理的執行階段。每個執行階段都可讓您輕鬆建置和執行容器，並以 Node.js 版本為基礎的 Web 應用程式。當您使用 Node.js 執行階段時，應用程式執行程式會以受管理的 Node.js 執行階段映像開始。此映像檔是以 Amazon Linux 泊塢視窗映像為基礎，其中包含 Node.js 版本和某些工具的執行階段套件。App Runner 使用此託管運行時映像作為基本映像，並添加您的應用程序代碼來構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 Node.js 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

Node.js 執行階段的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`12.21.0`

下列範例會示範版本鎖定：

- `12.21`-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。
- `12.21.0`— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

主題

- Node.js 執行階段設定

- 特定執行階段版本的編號說明
- Node.js 執行階段範例
- Node.js 執行階段版本資訊

Node.js 執行階段設定

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得您的組態設定。

使用 Node.js 執行階段時，您也可以使用來源存放庫根目錄`package.json`中名為的 JSON 檔案來設定組建和執行階段。使用此檔案，您可以設定 Node.js 引擎版本、相依性套件和各種命令 (命令列應用程式)。諸如 npm 或 yarn 之類的 Package 件管理員會將此檔案解譯為其命令的輸入。

例如：

- npm install安裝由中的`dependencies`和`devDependencies`節點定義的套件`package.json`。
- npm start或npm run start執行中的`scripts/start`節點定義的指令`package.json`。

以下是範例 `package.json` 檔案。

包裝

```
{
  "name": "node-js-getting-started",
  "version": "0.3.0",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "12.21.0"
  },
```

```
  "scripts": {
    "start": "node index.js",
    "test": "node test.js"
  },
  "dependencies": {
    "cool-ascii-faces": "^1.3.4",
    "ejs": "^2.5.6",
    "express": "^4.15.2"
  },
  "devDependencies": {
    "got": "^11.3.0",
    "tape": "^4.7.0"
  }
}
```

如需詳細資訊package.json，請參閱在 npm 文件網站上建立封裝 .json 檔案。

提示

- 如果您的package.json檔案定義了start命令，您可以在 App Runner 設定檔中將其當做run命令使用，如下列範例所示。

  Example

  包裝

  ```
  {
    "scripts": {
      "start": "node index.js"
    }
  }
  ```

  阿普魯人. 羊

  ```
  run:
    command: npm start
  ```

- 當您npm install在開發環境中執行時，npm 會建立檔案package-lock.json。此文件包含剛剛安裝的軟件包版本 npm 的快照。此後，當 npm 安裝依賴關係時，它使用這些確切的版本。如果你安裝 yarn 它創建一個yarn.lock文件。將這些檔案提交至您的原始程式碼儲存庫，以確保您的應用程式已安裝您所開發並測試的相依性版本。

- 您還可以使用應用程序運行器配置文件來配置 Node.js 版本和啟動命令。當您執行此操作時，這些定義會覆寫中的定義`package.json`。中的`node`版本`package.json`與應用程式執行器設定檔中的`runtime-version`值之間的衝突會導致應用程式執行器建置階段失敗。

特定執行階段版本的編號說明

Node.js 18 (已修訂的應用程式執行程式組建

應用程式執行程式現在會針對以下執行階段版本的應用程式執行更新的建置程序：Python 3.11 和 Node.js 18。如果您的應用程式在其中一個執行階段版本上執行，請參閱以託管運行時版本和應用程序運行器構建取得有關修訂建置程序的詳細資訊。使用所有其他執行階段版本的應用程式不會受到影響，而且會繼續使用原始建置程序。

Node.js 執行階段範例

下列範例顯示用於建置和執行 Node.js 服務的應用程式執行器組態檔案。

> **Note**
>
> *`############### 12.21.0 # 18.19.0#`*您可以將其替換為您要使用的版本。如需最新支援的 Node.js 執行階段版本，請參閱the section called “版本資訊”。

最小的 Node.js 配置文件

此範例顯示可與 Node.js 受管理執行階段搭配使用的最小組態檔案。如需 App Runner 使用最小組態檔案所做的假設，請參閱the section called “組態檔案範例”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: nodejs12
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

擴展的 Node.js 配置文件

此範例顯示所有組態索引鍵與 Node.js 受管理執行階段的使用。

> **Note**
>
> 這些範例中使用的執行階段版本為 *12.2* 1.0。您可以將其替換為您要使用的版本。如需最新支援的 Node.js 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: nodejs12
build:
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 12.21.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

擴展 Node.js 配置文件-Node.js 18（使用修訂版本）

此範例顯示在中使用 Node.js 受管理執行階段的所有組態索引鍵 `apprunner.yaml` 此範例包含一個`pre-run`區段，因為此版本的 Node.js 使用修訂的應用程式執行程式組建。

該pre-run參數僅由修訂後的應用程序運行器構建支持。如果您的應用程式使用原始 App Runner 組建支援的執行階段版本，請勿在設定檔中插入此參數。如需詳細資訊，請參閱 託管運行時版本和應用程序運行器構建。

> **Note**
>
> 這些範例中使用的執行階段版本為 *18.19.* 0。您可以將其替換為您要使用的版本。如需最新支援的 Node.js 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: nodejs18
build:
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 18.19.0
  pre-run:
    - node copy-global-files.js
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Node.js 應用程序與咕嚕

這個例子演示了如何配置與咕嚕開發的 Node.js 應用程序。咕嚕是一個命令行 JavaScript 任務亞軍。它運行重複性任務並管理過程自動化以減少人為錯誤。咕嚕和咕嚕插件安裝和使用 NPM 管理。您可以通過包括在源存儲庫的根`Gruntfile.js`文件配置咕嚕。

Example 包裝

```
{
  "scripts": {
    "build": "grunt uglify",
    "start": "node app.js"
  },
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  },
  "dependencies": {
    "express": "^4.15.2"
  },
}
```

Example Gruntfile.js

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Load the plugin that provides the "uglify" task.
```

```
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).
  grunt.registerTask('default', ['uglify']);

};
```

Example 阿普魯人. 羊

> **Note**
>
> 這些範例中使用的執行階段版本為 *12.2* 1.0。您可以將其替換為您要使用的版本。如需最新支援的 Node.js 執行階段版本，請參閱the section called “版本資訊”。

```
version: 1.0
runtime: nodejs12
build:
  commands:
    pre-build:
      - npm install grunt grunt-cli
      - npm install --only=dev
      - npm run build
    build:
      - npm install --production
run:
  runtime-version: 12.21.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Node.js 執行階段版本資訊

本主題列出應用程式執行程式支援的 Node.js 執行階段版本的完整詳細資料。

支持的運行時版本-修訂後的應用程序

執行時間名稱	次要版本	包含的套件
Node.js 18 (上一篇文章)	18.20.3	七分之七、紗

執行時間名稱	次要版本	包含的套件
	18.20.2	故宮 10，紗線 *
	18.19.1	故宮 10，紗線 *
	18.19.0	故宮 10，紗線 *

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
Node.js 16 (上一篇文章 16)	16.20.2	故宮八.
	16.20.1	七八．四、紗線
	16.20.0	七八．四、紗線
	16.19.1	七八．四、紗線
	16.19.0	七八．四、紗線
	16.18.1	七八．四、紗線
	16.17.1	七八．四、紗線
	16.17.0	七八．四、紗線
Node.js 14 (十一月十四日)	14.21.3	故宮六十八，紗
	14.21.2	七分之六．十八、紗線
	14.21.1	七分之六．十八、紗線

執行時間名稱	次要版本	包含的套件
	14.20.1	七分之六．十八、紗線
	14.19.0	七分之六．十八、紗線
Node.js 12 (十一月十二日)	12.22.12	故宮 6.14.16 紗
	12.21.0	七分之六．十六、紗線

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 Java 平台

AWS App Runner Java 平台提供受管理的執行階段。每個執行階段都可以使用基於 Java 版本的 Web 應用程式輕鬆建置和執行容器。當您使用 Java 運行時，應用程序運行器以託管的 Java 運行時映像開始。此映像檔是以 Amazon Linux 泊塢視窗映像為基礎，並包含適用於某個版本的 Java 和某些工具的執行階段套件。App Runner 使用此託管運行時映像作為基本映像，並添加您的應用程序代碼來構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

目前，所有支持的 Java 運行時都基於 Amazon Corretto。如需有效的 Java 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

Amazon 運行時的版本語法：

執行時間	語法	範例
corretto11	`11.0[.openjdk-update [.openjdk-build [.corretto-specific-revision ]]]`	`11.0.13.08.1`
corretto8	`8[.openjdk-update [.openjdk-build [.corretto-specific-revision ]]]`	`8.312.07.1`

下列範例會示範版本鎖定：

- `11.0.13`-鎖定打開 JDK 更新版本。應用程序運行器更新僅打開 JDK 和 Amazon Corretto 低級別構建。
- `11.0.13.08.1`— 鎖定到特定版本。應用程序運行器不會更新您的運行時版本。

主題

- Java 運行時配置
- Java 執行階段範例
- Java 執行階段版本資訊

Java 運行時配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得您的組態設定。

Java 執行階段範例

下面的例子顯示了用於構建和運行 Java 服務的應用程序運行配置文件。最後一個範例是您可以部署至 Corretto 11 執行階段服務的完整 Java 應用程式的原始程式碼。

> **Note**
>
> 這些範例中使用的執行階段版本為 *11.0.* 13.08.1。您可以將其替換為您要使用的版本。如需最新支援的 Java 執行階段版本，請參閱the section called “版本資訊”。

最小框 11 配置文件

此範例顯示可與 Corretto 11 受管理執行階段搭配使用的最小組態檔案。如需 App Runner 使用最小組態檔案所做的假設，請參閱。

Example 阿普魯人. 羊

```
version: 1.0
runtime: corretto11
build:
  commands:
    build:
      - mvn clean package
run:
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
```

擴展框 11 配置文件

此範例顯示如何將所有組態金鑰與 Corretto 11 託管執行階段搭配使用。

> **Note**
>
> 這些範例中使用的執行階段版本為 *11.0.* 13.08.1。您可以將其替換為您要使用的版本。如需最新支援的 Java 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: corretto11
build:
  commands:
    pre-build:
      - yum install some-package
      - scripts/prebuild.sh
    build:
      - mvn clean package
    post-build:
      - mvn clean test
  env:
    - name: M2
      value: "/usr/local/apache-maven/bin"
    - name: M2_HOME
      value: "/usr/local/apache-maven/bin"
run:
  runtime-version: 11.0.13.08.1
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

完整的 Corretto 11 應用程序源

此範例顯示完整 Java 應用程式的原始程式碼，您可以將其部署至 Corretto 11 執行階段服務。

Example src /主/爪/// HelloWorld HelloWorld

```
package com.HelloWorld;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorld {

    @RequestMapping("/")
```

```
    public String index(){
        String s = "Hello World";
        return s;
    }
}
```

Example src/主/爪// HelloWorld /Main.java

```
package com.HelloWorld;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Main {

    public static void main(String[] args) {

        SpringApplication.run(Main.class, args);
    }
}
```

Example 阿普魯人. 羊

```
version: 1.0
runtime: corretto11
build:
  commands:
    build:
      - mvn clean package
run:
  command: java -Xms256m -jar target/HelloWorldJavaApp-1.0-SNAPSHOT.jar .
  network:
    port: 8080
```

Example pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
```

```
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
    <relativePath/>
  </parent>
  <groupId>com.HelloWorld</groupId>
  <artifactId>HelloWorldJavaApp</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
          <groupId>org.junit.vintage</groupId>
          <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
```

```
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Java 執行階段版本資訊

本主題列出應用程式執行程式支援之 Java 執行階段版本的完整詳細資料。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
Corretto 11 (克雷特托 11)	11.0.23.9.1	釋界 3.9.8，搖籃
	11.0.22.7.1	釋界 3.9.6，搖籃
	11.0.21.9.1	釋界 3.9.6，搖籃
	11.0.21.9.1	釋界
	11.0.20.8.1	釋界 3.9.3，搖籃
	11.0.19.7.1	釋界 3.9.3，搖籃
	11.0.18.10.1	釋界 3.9.1，搖籃
	11.0.17.8.1	釋界 3.8.6，搖籃
	11.0.16.9.1	釋界 3.8.6，搖籃
	11.0.13.08.1	釋界 3.6.3，搖籃 6.5
Corretto 8 (克雷托 8)	8.412.08.1	釋界 3.9.8，搖籃
	8.402.08.1	釋界 3.9.6，搖籃
	8.392.08.1	釋界 3.9.6，搖籃
	8.382.05.1	釋界 3.9.4，搖籃

執行時間名稱	次要版本	包含的套件
	8.372.07.1	釋界 3.9.3，搖籃
	8.362.08.1	釋界 3.9.1，搖籃
	8.352.08.1	釋界 3.8.6，搖籃
	8.342.07.4	釋界 3.8.6，搖籃
	8.312.07.1	釋界 3.6.3，搖籃 6.5

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 .NET 平台

AWS App Runner .NET 平台提供受管理的執行階段。每個執行階段都可以使用以 .NET 版本為基礎的 Web 應用程式輕鬆建置和執行容器。當您使用 .NET 執行階段時，應用程式執行階段會以託管 .NET 執行階段映像開始。此映像檔是以 Amazon Linux Docker 映像檔為基礎，其中包含一個 .NET 版本的執行階段套件，以及一些工具和常用的相依性套件。App Runner 使用此託管運行時映像作為基本映像，並添加您的應用程序代碼來構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 .NET 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

.NET 執行階段的版本語法：*major*[.*minor*[.*patch*]]

例如：6.0.9

下列範例會示範版本鎖定：

- 6.0-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。
- 6.0.9— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

主題

- .NET 運行時配置
- .NET 運行時示例
- .NET 運行時版本信息

.NET 運行時配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的BuildCommand和StartCommand成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得您的組態設定。

.NET 運行時示例

下列範例顯示用於建置和執行 .NET 服務的應用程式執行器組態檔案。最後一個範例是您可以部署至 .NET 執行階段服務的完整 .NET 應用程式的原始程式碼。

> **Note**
>
> 這些範例中使用的執行階段版本為 *6.0.9*。您可以將其替換為您要使用的版本。如需支援的最新 .NET 執行階段版本，請參閱 the section called “版本資訊”

最小的 .NET 配置文件

此範例顯示可與 .NET 管理執行階段搭配使用的最小組態檔案。如需 App Runner 使用最小組態檔案所做的假設，請參閱the section called “組態檔案範例”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
```

擴展的 .NET 配置文件

此範例顯示所有組態索引鍵與 .NET 託管執行階段的使用方式。

> **Note**
>
> 這些範例中使用的執行階段版本為 *6.0.9*。您可以將其替換為您要使用的版本。如需支援的最新 .NET 執行階段版本，請參閱 the section called “版本資訊”

Example 阿普魯人. 羊

```
version: 1.0
runtime: dotnet6
build:
  commands:
    pre-build:
      - scripts/prebuild.sh
    build:
```

```
      - dotnet publish -c Release -o out
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 6.0.9
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

完整的 .NET 應用程式

此範例顯示可部署至 .NET 執行階段服務之完整 .NET 應用程式的原始程式碼。

> **Note**
>
> - 運行以下命令來創建一個簡單的 .NET 6 Web 應用程序： `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - 添加`apprunner.yaml`到創建的 .NET 6 網絡應用程序。

Example HelloWorldDotNetApp

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
```

```
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

.NET 運行時版本信息

本主題列出 App Runner 支援的 .NET 執行階段版本的完整詳細資料。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
. NET 6 (網站 6)	6.0.31	.NET 開發套件 6.0.423
	6.0.30	.NET 開發套件 6.0.422
	6.0.29	.NET 開發套件 6.0.421
	6.0.28	.NET 軟體開發套件
	6.0.26	.NET 軟體開發套件 6.0.418
	6.0.25	.NET 軟體開發套件 6.0.417
	6.0.24	.NET 開發套件 6.0.416
	6.0.22	.NET 開發套件 6.0.414
	6.0.21	.NET 開發套件 6.0.413
	6.0.20	.NET 軟體開發套件 6.0.412
	6.0.19	.NET 開發套件 6.0.411
	6.0.16	.NET 開發套件
	6.0.15	.NET 軟體開發套件
	6.0.14	.NET 開發套件 6.0.406
	6.0.13	.NET 軟體開發套件
	6.0.12	.NET 開發套件

執行時間名稱	次要版本	包含的套件
	6.0.11	.NET 開發套件
	6.0.10	.NET 軟件開發套件
	6.0.9	.NET 開發套件

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 PHP 平台

AWS App Runner PHP 平台提供受管理的執行階段。您可以使用每個執行階段，以 PHP 版本為基礎，透過 Web 應用程式建置和執行容器。當您使用 PHP 運行時，應用程序運行器以託管 PHP 運行時映像開始。此映像檔是以 Amazon Linux 碼頭視窗映像為基礎，並包含 PHP 版本和某些工具的執行階段套件。App Runner 使用此託管運行時映像作為基本映像，並添加應用程序代碼以構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 PHP 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

PHP 運行時的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`8.1.10`

以下是版本鎖定的範例：

- `8.1`-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。
- `8.1.10`— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

> **Important**
>
> 如果您想在默認存儲庫根目錄以外的位置指定 App Runner 服務的代碼存儲庫源目錄，則您的 PHP 託管運行時版本必須是 PHP `8.1.22` 或更高版本。之前的 PHP 執行階段版本只`8.1.22`能使用預設的根來源目錄。

主題

- PHP 運行時配置
- 相容性
- PHP 執行階段範例
- PHP 執行階段發行資訊

PHP 運行時配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得設定。

相容性

您可以使用以下 Web 服務器之一在 PHP 平台上運行應用程序運行服務：

- Apache HTTP Server
- NGINX

Apache HTTP Server並NGINX與 PHP-FPM 兼容。您可以使用下列其中一項NGINX來啟動Apache HTTP Server和：

- 監督-如需有關執行 a 的詳細資訊 supervisord，請參閱執行監督器。
- 啟動腳本

有關如何使用 Apache HTTP 服務器或 NGINX 使用 PHP 平台配置應用程序運行器服務的示例，請參閱。the section called “完整的 PHP 應用程序源”

檔案結構

`index.php`必須安裝在 Web 伺服器`root`目錄下的`public`資料夾中。

> **Note**
>
> 建議將`startup.sh`或`supervisord.conf`檔案儲存在 Web 伺服器的根目錄中。請確定指`start`令指向`startup.sh`或`supervisord.conf`檔案的儲存位置。

以下是檔案結構的範例 (如果您使用的話) supervisord。

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

如果您使用啟動指令碼，以下是檔案結構的範例。

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

我們建議您將這些檔案結構儲存在指定給 App Runner 服務的程式碼儲存庫來源目錄中。

```
/<sourceDirectory>/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

> **Important**
>
> 如果您想在默認存儲庫根目錄以外的位置指定 App Runner 服務的代碼存儲庫源目錄，則您的 PHP 託管運行時版本必須是 PHP 8.1.22 或更高版本。之前的 PHP 執行階段版本只8.1.22能使用預設的根來源目錄。
> App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。除非您使用 App Runner 配置文件中的runtime-version關鍵字指定了版本鎖定，否則默認情況下，您的服務將使用最新的運行時間。

PHP 執行階段範例

以下是用於構建和運行 PHP 服務的應用程序運行器配置文件的示例。

最小的 PHP 配置文件

下列範例是您可以搭配 PHP 管理執行階段使用的最小組態檔案。如需最小組態檔案的詳細資訊，請參閱the section called “組態檔案範例”。

Example 阿普魯人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
```

擴展的 PHP 配置文件

下面的例子使用了 PHP 託管運行時的所有配置鍵。

> **Note**
>
> 這些範例中使用的執行階段版本為 *8.1.* 10。您可以將其替換為您要使用的版本。如需最新支援的 PHP 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普鲁人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - echo example build command for PHP
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 8.1.10
  command: ./startup.sh
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

完整的 PHP 應用程序源

下列範例是 PHP 應用程式原始程式碼，您可以使用Apache HTTP Server或來部署至 PHP 執行階段服務NGINX。這些範例假設您使用預設的檔案結構。

使用運行 PHP 平Apache HTTP Server台 supervisord

Example 檔案結構

> **Note**
>
> - 該supervisord.conf文件可以存儲在存儲庫中的任何位置。確保start命令指向supervisord.conf文件的存儲位置。
> - index.php必須安裝在root目錄下的public資料夾中。

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example 主管. 連續

```
[supervisord]
nodaemon=true

[program:httpd]
command=httpd -DFOREGROUND
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example 阿普魯人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

使用運行 PHP 平Apache HTTP Server台 startup script

Example 檔案結構

> **Note**
>
> - 該startup.sh文件可以存儲在存儲庫中的任何位置。確保start命令指向startup.sh文件的存儲位置。
> - index.php必須安裝在root目錄下的public資料夾中。

```
/
## public/
```

```
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start apache
httpd -DFOREGROUND &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - 在將`startup.sh`檔案提交至 Git 儲存庫之前，請務必將檔案儲存為可執行檔。用`chmod +x startup.sh`於設定`startup.sh`檔案的執行權限。
> - 如果您不將`startup.sh`檔案儲存為可執行檔，請在`apprunner.yaml`檔案中輸入`chmod +x startup.sh`為`build`指令。

Example 阿普魯人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
  network:
    port: 8080
```

```
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

使用運行 PHP 平NGINX台 supervisord

Example 檔案結構

> **Note**
>
> - 該supervisord.conf文件可以存儲在存儲庫中的任何位置。確保start命令指向supervisord.conf文件的存儲位置。
> - index.php必須安裝在root目錄下的public資料夾中。

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example 主管. 連續

```
[supervisord]
nodaemon=true

[program:nginx]
command=nginx -g "daemon off;"
autostart=true
```

```
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example 阿普魯人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

使用運行 PHP 平NGINX台 startup script

Example 檔案結構

> **Note**
>
> - 該startup.sh文件可以存儲在存儲庫中的任何位置。確保start命令指向startup.sh文件的存儲位置。
> - index.php必須安裝在root目錄下的public資料夾中。

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start nginx
nginx -g 'daemon off;' &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - 在將startup.sh檔案提交至 Git 儲存庫之前，請務必將檔案儲存為可執行檔。用chmod +x startup.sh於設定startup.sh檔案的執行權限。

- 如果您不將startup.sh檔案儲存為可執行檔，請在apprunner.yaml檔案中輸入chmod +x startup.sh為build指令。

Example 阿普鲁人. 亚姆尔

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

PHP 執行階段發行資訊

本主題列出了應用程序運行器支持的 PHP 運行時版本的完整詳細信息。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
PHP 8.1	8.1.29	
	8.1.28	

執行時間名稱	次要版本	包含的套件
	8.1.27	
	8.1.26	
	8.1.24	
	8.1.22	
	8.1.21	
	8.1.20	
	8.1.19	
	8.1.17	
	8.1.16	
	8.1.14	
	8.1.13	
	8.1.12	
	8.1.10	

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 Ruby 平台

AWS App Runner Ruby 平台提供受管理的執行階段。每個執行階段都可以使用 Ruby 版本為基礎的 Web 應用程式輕鬆建置和執行容器。當您使用 Ruby 運行時，應用程序運行器以託管的 Ruby 運行時映像開始。此映像檔是以 Amazon Linux 泊塢視窗映像為基礎，其中包含 Ruby 版本和一些工具的執行

階段套件。App Runner 使用此託管運行時映像作為基本映像，並添加應用程序代碼以構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 Ruby 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

Ruby 執行階段的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`3.1.2`

下列範例會示範版本鎖定：

- `3.1`-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。
- `3.1.2`— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

主題

- 紅寶石運行配置
- 紅寶石運行時例
- Ruby 執行階段版本資訊

紅寶石運行配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。

- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得您的組態設定。

紅寶石運行時例

下面的例子顯示了用於構建和運行 Ruby 服務的應用程序運行配置文件。

最小的 Ruby 配置文件

這個例子顯示了一個最小的配置文件，你可以與 Ruby 託管的運行時使用。如需 App Runner 使用最小組態檔案所做的假設，請參閱the section called “組態檔案範例”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 8080
```

擴展紅寶石配置文件

這個例子顯示了使用所有的配置鍵與 Ruby 託管運行時。

> **Note**
>
> 這些範例中使用的執行階段版本為 *3.1.2*。您可以將其替換為您要使用的版本。如需最新支援的 Ruby 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普魯人. 羊

```
version: 1.0
```

```
runtime: ruby31
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - bundle install
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.1.2
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

完整的 Ruby 應用程序源

這些範例顯示您可以部署至 Ruby 執行階段服務的完整 Ruby 應用程式的原始程式碼。

Example 伺服器 .rb

```
# server.rb
require 'sinatra'

get '/' do
  'Hello World!'
end
```

Example config.ru

```
# config.ru

require './server'

run Sinatra::Application
```

Example Gemfile

```
# Gemfile
source 'https://rubygems.org (https://rubygems.org/)'

gem 'sinatra'
gem 'puma'
```

Example 阿普魯人. 羊

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
```

Ruby 執行階段版本資訊

本主題列出 App Runner 支援的 Ruby 執行階段版本的完整詳細資料。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
紅寶石 3.1 (紅寶石 31)	3.1.6	方形精簡版
	3.1.4	方形精簡版
	3.1.3	方形精簡版
	3.1.2	方形精簡版

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

使用 Go 平台

AWS App Runner Go 平台提供受管理的執行階段。每個執行階段都可以使用以 Go 版本為基礎的 Web 應用程式輕鬆建置和執行容器。當您使用 Go 執行階段時，應用程式執行階段會以受管理的 Go 執行階段映像開始。此映像檔是以 Amazon Linux Docker 映像檔為基礎，並包含適用於 Go 版本和某些工具的執行階段套件。App Runner 使用此託管運行時映像作為基本映像，並添加應用程序代碼以構建 Docker 映像。然後，它將部署此映像以在容器中運行 Web 服務。

當您使用應用程式執行器主控台或 CreateServiceAPI 作業建立服務時，您可以指定應用程式執行器服務的執行階段。您也可以將執行階段指定為原始程式碼的一部分。在您包含在程式碼儲存庫中的 App Runner 設定檔中使用`runtime`關鍵字。託管運行時的命名約定是。 <language-name><major-version>

如需有效的 Go 執行階段名稱和版本，請參閱the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行階段更新為最新版本。如果您的應用程式需要特定版本的受管理執行階段，您可以使用 App Runner 設定檔中的`runtime-version`關鍵字來指定它。您可以鎖定到任何級別的版本，包括主要或次要版本。應用程序運行器僅對服務的運行時進行較低級別的更新。

Go 執行階段的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`1.18.7`

下列範例會示範版本鎖定：

- `1.18`-鎖定主要和次要版本。應用程序運行器僅更新補丁版本。
- `1.18.7`— 鎖定到特定的修補程式版本。應用程序運行器不會更新您的運行時版本。

主題

- 去運行時配置
- Go 執行階段範例

- Go 執行階段版本資訊

去運行時配置

當您選擇受管理的執行階段時，您也必須設定最低限度的建置和執行命令。您可以在創建或更新應用程序運行器服務時進行配置。您可以使用下列其中一種方法來執行此操作：

- 使用 App Runner 主控台 — 在建立程序或設定索引標籤的 [設定組建] 區段中指定命令。
- 使用應用程式執行程式 API — 呼叫CreateService或 UpdateServiceAPI 作業。使用CodeConfigurationValues資料類型的`BuildCommand`和`StartCommand`成員指定命令。
- 使用組態檔案 — 在最多三個建置階段中指定一或多個建置命令，以及用來啟動應用程式的單一執行命令。還有其他可選配置設置。

提供組態檔案是選擇性的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定應用程式執行器是在建立時直接取得您的組態設定，還是從組態檔案取得您的組態設定。

Go 執行階段範例

下列範例顯示用於建置和執行 Go 服務的應用程式執行器組態檔案。

最小 Go 配置文件

此範例顯示可與 Go Managed 執行階段搭配使用的最小組態檔案。如需 App Runner 使用最小組態檔案所做的假設，請參閱the section called “組態檔案範例”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
```

擴展 Go 配置文件

這個例子顯示了使用 Go 託管運行時的所有配置鍵。

> **Note**
>
> 這些範例中使用的執行階段版本為 *1.18.7*。您可以將其替換為您要使用的版本。如需最新支援的 Go 執行階段版本，請參閱the section called “版本資訊”。

Example 阿普魯人. 羊

```
version: 1.0
runtime: go1
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - go build main.go
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 1.18.7
  command: ./main
  network:
    port: 3000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

完整 Go 應用程式來源

這些範例顯示您可以部署至 Go 執行階段服務的完整 Go 應用程式的原始程式碼。

Example main.go

```
package main
import (
    "fmt"
    "net/http"
)
```

```
func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprint(w, "<h1>Welcome to App Runner</h1>")
    })
    fmt.Println("Starting the server on :3000...")
    http.ListenAndServe(":3000", nil)
}
```

Example 阿普魯人. 羊

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
  network:
    port: 3000
    env: APP_PORT
```

Go 執行階段版本資訊

本主題列出 App Runner 支援的 Go 執行階段版本的完整詳細資料。

支持的運行時版本-原始應用程序運行

執行時間名稱	次要版本	包含的套件
去 1 (去 1)	1.18.10	
	1.18.9	
	1.18.8	
	1.18.7	

> **Note**
>
> App Runner 為最近發布的特定主要運行時提供了修訂的構建過程。因此，您將在本文檔的某些部分中看到對修訂後的 App Runner 構建和原始 App Runner 構建的引用。如需詳細資訊，請參閱託管運行時版本和應用程序運行器構建。

開發應用程式執行程式碼

本章討論在開發或移轉用於部署的應用程式程式碼時，應考慮的執行階段資訊和開發準則 AWS App Runner。

運行時信息

無論您是提供容器映像還是 App Runner 為您構建一個，App Runner 都會在容器實例中運行您的應用程序代碼。以下是容器執行個體執行階段環境的幾個關鍵層面。

- 框架支持-應用程序運行器支持實現 Web 應用程序的任何圖像。它與您選擇的編程語言以及您使用的 Web 應用程序服務器或框架（如果您使用任何）無關。為了您的方便，我們為各種編程平台提供特定於平台的託管運行時間，以簡化應用程序構建過程和抽象圖像創建。
- 網絡請求 — 應用程序運行器為容器實例提供 HTTP 1.0 和 HTTP 1.1 的支持。如需有關設定服務的詳細資訊，請參閱the section called “組態”。您不需要實現 HTTPS 安全流量的處理。應用程序運行器將所有傳入的 HTTP 請求重定向到相應的 HTTPS 端 您不需要配置任何設置即可重定向 HTTP Web 請求。應用程序運行器將請求傳遞給您的應用程序容器實例之前終止 TLS。

  > **Note**
  >
  > - HTTP 要求總共有 120 秒的要求逾時限制。120 秒包括應用程式讀取要求 (包括主體) 所花費的時間，以及完成寫入 HTTP 回應。
  > - 請求讀取和響應超時限制取決於您使用的應用程序。這些應用程序可能有自己的內部超時，例如 HTTP 服務器的 Python，Gunicorn，有一個 30 秒的默認超時限制。在這種情況下，應用程式的逾時限制會覆寫應用程式執行器 120 秒逾時限制。
  > - 您不需要將 TLS 加密套件或任何其他參數配置為應用程序運行器是完全託管的服務，可以為您管理 TLS 終止。

- 無狀態應用程式 — 目前 App Runner 不支援可設定狀態的應用程式。因此，App Runner 不保證超過處理單個傳入 Web 請求的持續時間的狀態持久性。
- 存儲-應用程序運行器會根據傳入的流量自動擴展或縮小應用程序應用程序的實例。您可以為應用程式執行器應用程式設定自動縮放選項。由於處理 Web 要求的目前作用中執行個體數量是以傳入流量為基礎，因此 App Runner 無法保證檔案在處理單一要求之外仍可持續存在。因此，App Runner 在容器實例中實現了文件系統作為臨時存儲，這需要文件是暫時的。例如，當您暫停和繼續 App Runner 服務時，檔案不會持續存在。

App Runner 為您提供 3 GB 的臨時存儲空間，並將 3 GB 臨時存儲的一部分用於其提取，壓縮和實例上未壓縮的容器映像。剩餘的臨時存儲可以由您的應用程序運行器服務使用。但是，由於其無狀態性質，這不是永久存儲。

> **Note**
>
> 在某些情況下，存儲文件確實持續存在於請求之間。例如，如果下一個請求落在同一個實例上，則存儲文件將保留。在某些情況下，存儲文件在請求之間的持久性可能很有用。例如，在處理要求時，如果 future 的要求可能需要，您可以快取應用程式下載的檔案。這可能會加快 future 的請求處理速度，但無法保證速度的提高。您的程式碼不應假設先前要求中已下載的檔案仍然存在。
>
> 若要使用高輸送量、低延遲的記憶體內資料存放區保證快取，請使用 Amazon ElastiCache 之類的服務。

- 環境變數 — 預設情況下，App Runner 會讓PORT環境變數在容器執行個體中可用。您可以使用連接埠資訊設定變數值，並新增自訂環境變數和值。您也可以參考儲存在AWS Secrets Manager或AWS Systems Manager 參數存放區中的機密資料做為環境變數。如需建立環境變數的詳細資訊，請參閱〈〉參考環境變數。
- 執行個體角色 — 如果您的應用程式程式碼使用服務 API 或其中一個 AWS SDK 呼叫任何 AWS 服務，請使用 AWS Identity and Access Management (IAM) 建立執行個體角色。然後，在創建它時將其附加到您的應用程序運行器服務。在執行個體角色中包含程式碼所需的所有 AWS 服務動作權限。如需詳細資訊，請參閱 the section called “實例角色”。

程式碼開發指南

開發應用程式執行器 Web 應用程式的程式碼時，請考慮下列準則

- 設計無狀態代碼 — 將部署到 App Runner 服務的 Web 應用程序設計為無狀態。您的代碼應該假設在處理單個傳入 Web 請求的持續時間之外，沒有狀態仍然存在。
- 刪除暫存檔案 — 建立檔案時，檔案會儲存在檔案系統中，並佔用服務儲存空間配置的一部分。為了避免 out-of-storage 錯誤，請不要長時間保留臨時文件。在做出檔案快取決策時，平衡儲存體大小與要求處理速度。
- 執行個體啟動 — App Runner 提供五分鐘的執行個體啟動時間。您的執行個體必須在其設定的監聽連接埠上接聽要求，並在啟動後的五分鐘內維持正常狀態。在啟動期間，應用程式執行器執行個體會根據您的 vCPU 組態來配置虛擬 CPU (vCPU)。如需有關可用 vCPU 組態的詳細資訊，請參閱the section called “應用程序運行器支持”。

執行個體成功啟動後，會進入閒置狀態並等待要求。您可以根據執行個體啟動持續時間付費，每次執行個體啟動的最低費用為一分鐘。如需定價的詳細資訊，請參閱 AWS App Runner 定價。

使用應用程式執行器主控

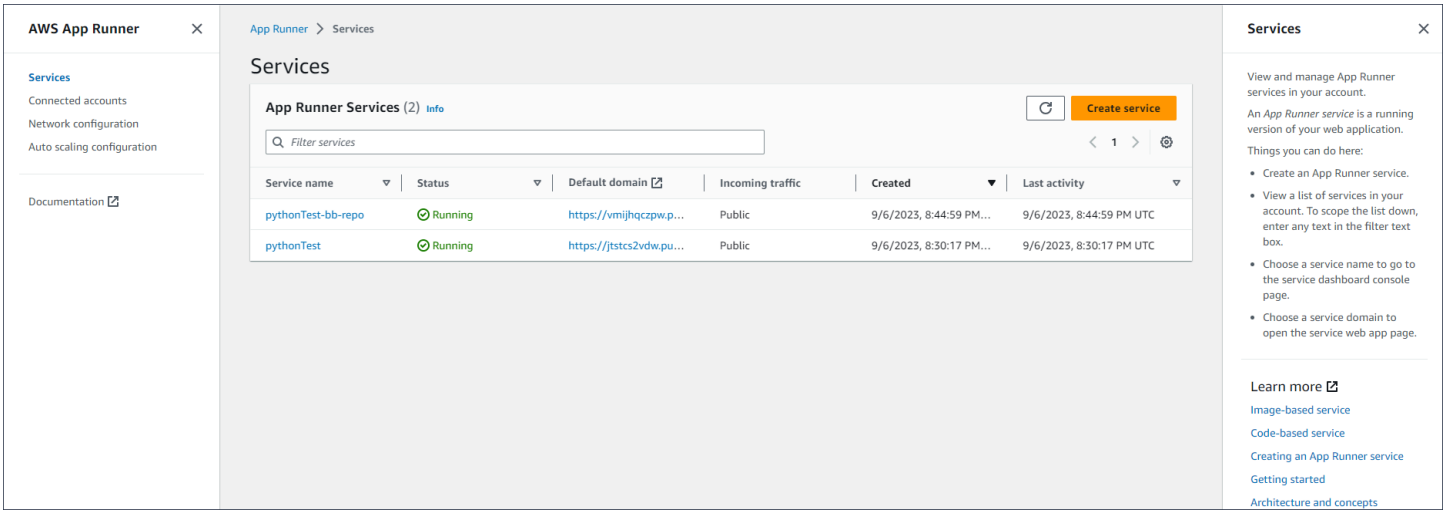
使用主 AWS App Runner 控台建立、管理和監視 App Runner 服務及相關資源，例如連線的帳戶。您可以檢視現有服務、建立新服務及設定服務。您可以檢視 App Runner 服務的狀態，以及檢視記錄、監視活動和追蹤指標。您還可以導航到服務的網站或源存儲庫。

以下各節說明主控台的配置和功能，並將您指向相關資訊。

整體主控台配置

應用程序運行器控制台有三個區域。由左至右：

- 導覽窗格 — 可摺疊或展開的側窗格。使用它來選擇您要使用的頂層控制台頁面。
- 內容窗格 — 主控台頁面的主要部分。使用它來檢視資訊並執行您的工作。
- 說明窗格 — 側窗格以取得更多資訊。展開它以獲取有關您所在頁面的幫助。或者選擇控制台頁面上的任何信息鏈接以獲取上下文幫助。



「服務」頁面

服務頁面會列出您帳戶中的應用程式執行器服務。您可以使用篩選文字方塊來縮小清單的範圍。

前往「服務」頁面

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在導覽窗格中，選擇服務。

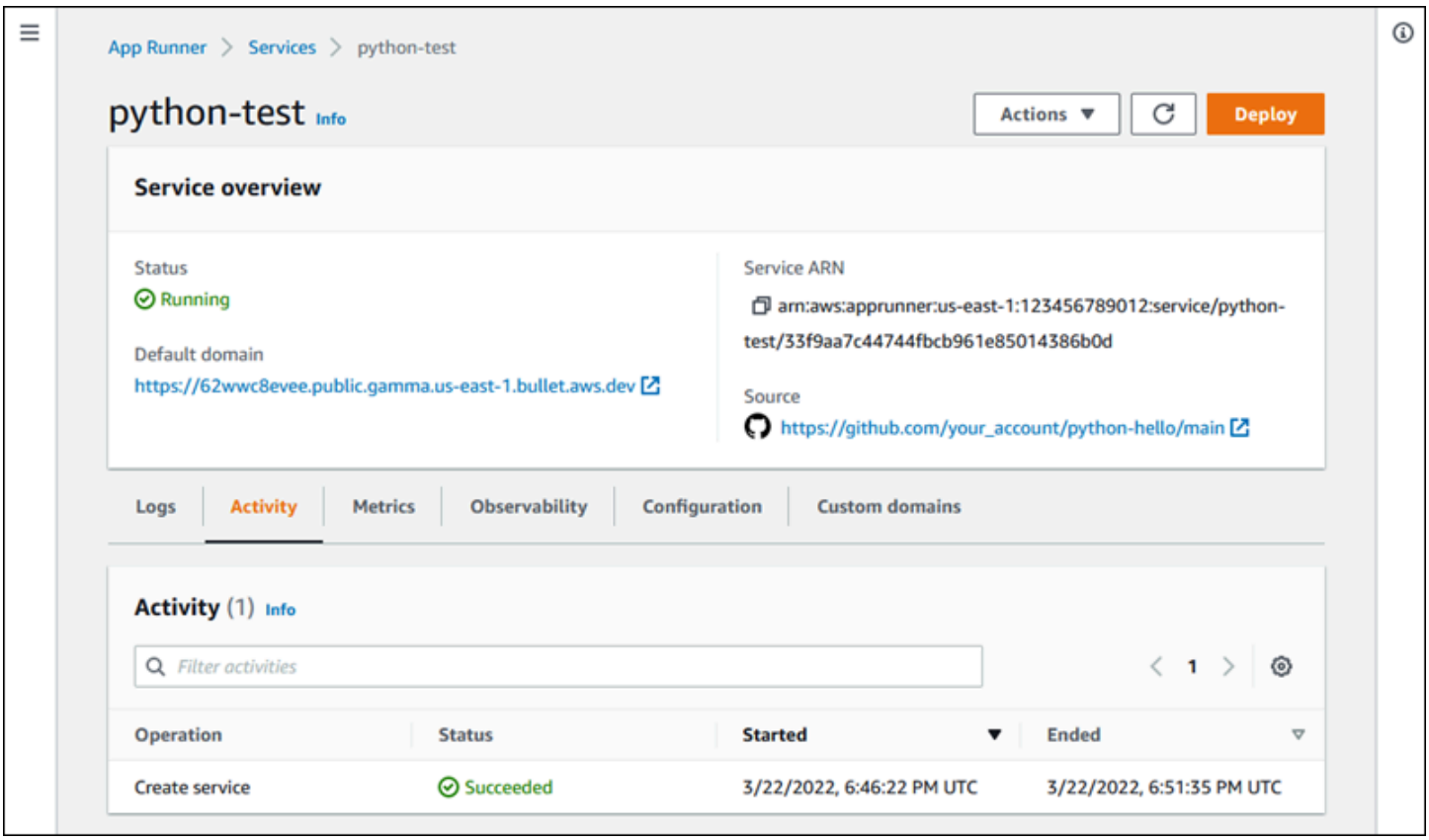
你可以在這裡做的事情：

- 建立應用程式執行器服務。如需詳細資訊，請參閱 the section called “建立”。
- 選擇要移至服務儀表板主控台頁面的服務名稱。
- 選擇服務網域以開啟服務 Web 應用程式頁面。

服務儀表板頁面

您可以查看有關應用程序運行器服務的信息，並從服務儀表板頁面對其進行管理。在頁面頂部，您可以看到服務名稱。

若要前往服務儀表板，請瀏覽至 [服務] 頁面 (請參閱上一節)，然後選擇您的 App Runner 服務。



服務概述部分提供有關 App Runner 服務和您的應用程序的基本詳細信息。你可以在這裡做的事情：

- 檢視服務詳細資料，例如狀態、健全狀況和 ARN。
- 導航到默認域-應用程序 Runner 為服務中運行的 Web 應用程序提供的域。這是應用程序跑步者擁有的`awsapprunner.com`域中的子域。
- 導覽至部署至服務的來源儲存區域。

- 啟動服務的來源儲存庫部署。
- 暫停、繼續及刪除您的服務。

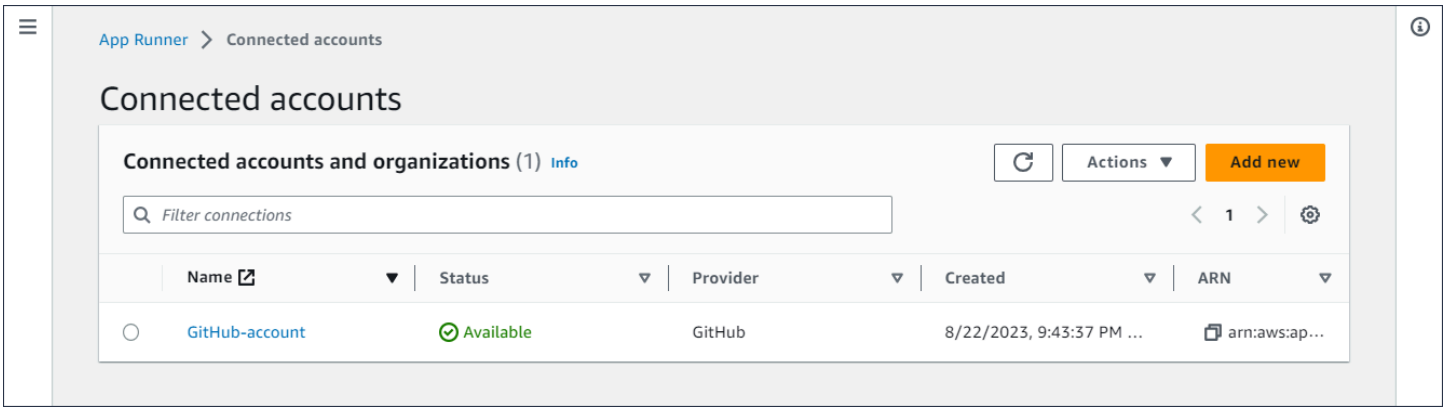
服務概述下方的標籤用於服務管理和觀察性。

[連線的帳戶] 頁面

[連線的帳戶] 頁面會列出與帳戶中原始程式碼儲存庫提供者的 App Runner 連線。您可以使用篩選文字方塊來縮小清單的範圍。如需連線帳戶的詳細資訊，請參閱the section called “連線”。

前往 [已連線的帳戶] 頁面

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [連線的帳戶]。



你可以在這裡做的事情：

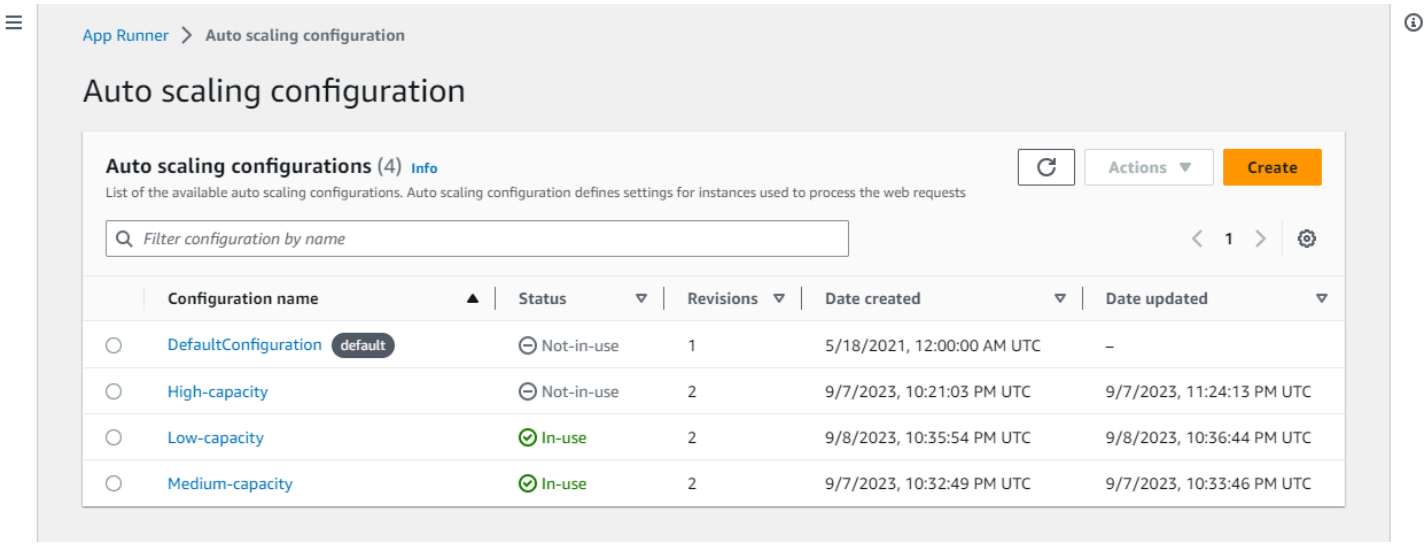
- 檢視您帳戶中的儲存庫提供者連線清單。若要縮小清單的範圍，請在篩選文字方塊中輸入任何文字。
- 選擇連線名稱，以移至相關的提供者帳號或組織。
- 選取連線以完成您剛建立之連線的交握 (作為建立服務的一部分)，或刪除連線。

自動調整規模組態頁面

auto 調整配置頁面列出了您在帳戶中設置的自動擴展配置。您可以設定幾個參數來調整 auto 縮放行為，並將其儲存在不同的設定中，以便稍後指派給一或多個 App Runner 服務。您可以使用篩選文字方塊來縮小清單的範圍。如需 auto 調整比例設定的詳細資訊，請參閱管理服務的 auto 調整。

移至自動縮放設定頁面

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在導覽窗格中，選擇 [自動縮放設定]。



你可以在這裡做的事情：

- 查看您帳戶中現有的 auto 擴展配置列表。
- 為現有的 auto 調整規模組態或修訂版本建立。
- 將 auto 擴展配置設定為您建立的新服務的預設值。
- 刪除模型組態。
- 選取組態的名稱以導覽至「自動縮放修訂」面板以管理修訂。

管理應用程式執行器服務

本章說明如何管理您的 AWS App Runner 服務。在本章中，您將學習如何管理服務的生命週期：建立、設定和刪除服務、將新的應用程式版本部署至服務，以及透過暫停和繼續服務來控制 Web 服務的可用性。您還將學習如何管理服務的其他方面，例如連接和 auto 擴展。

主題

- 創建應用程序運行器服務
- 重建失敗的應用程式執行器服務
- 部署新的應用程式版本到應用程式執行
- 設定應用程式執行器服務
- 管理應用程式執行器
- 管理應用程式執行器自動
- 管理應用程式執行器服務的自訂網域名稱
- 暫停和恢復應用程序運行器服務
- 刪除應用程式執行器服務

創建應用程序運行器服務

AWS App Runner 自動從容器映像檔或原始程式碼儲存庫轉換為可自動調整規模的執行中 Web 服務。您可以將 App Runner 指向來源影像或程式碼，只指定少量必要的設定。如果需要，App Runner 會建置您的應用程式、佈建運算資源，並部署應用程式以在其上執行。

當您建立服務時，應用程式執行器會建立服務資源。在某些情況下，您可能需要提供連線資源。如果您使用 App Runner 主控台，則主控台會以隱含方式建立連線資源。如需應用程式執行器資源類型的詳細資訊，請參閱the section called “應用程序運行器”。這些資源類型的配額都與您的帳號相關聯 AWS 區域。如需詳細資訊，請參閱 the section called “應用運行器資源配額”。

根據來源類型和提供者的不同，建立服務的程序會有細微差異。本主題涵蓋建立這些來源類型的不同程序，以便您可以遵循適合您情況的任何項目。若要使用程式碼範例啟動基本程序，請參閱開始使用。

必要條件

在您建立 App Runner 服務之前，請務必完成下列動作：

- 完成中的設定步驟設定。
- 確保您的應用程序源已準備就緒。您可以使用中的代碼存儲庫 GitHub，Bitbucket 或 Amazon E lastic Container Registry (Amazon ECR) 中的容器映像來創建應用程序運行器服務。

建立服務

本節將逐步介紹兩種 App Runner 服務類型的建立程序：根據原始程式碼，並以容器映像為基礎。

> **Note**
>
> 如果您為服務建立輸出流量 VPC 連接器，接下來的服務啟動程序將會遇到一次性延遲。您可以在建立新服務時或之後使用服務更新來設定此組態。如需詳細資訊，請參閱本指南的 < 使用應用程式執行器聯網 > 一章一次性延遲中的。

從程式碼儲存庫建立服務

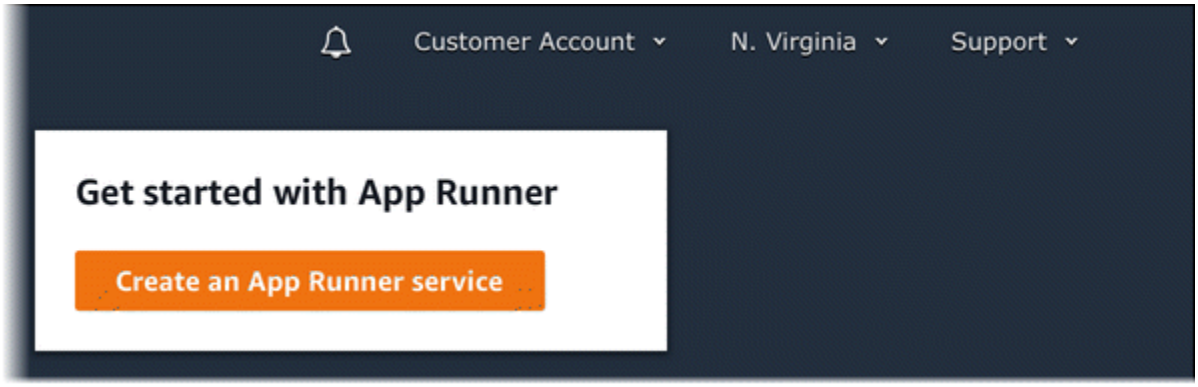
以下各節說明當您的原始碼是GitHub或 Bitbucket 中的程式碼儲存庫時，如何建立應用程式執行器服務。當您使用程式碼儲存庫時，App Runner 必須連線至提供者組織或帳戶。因此，您需要幫助建立此連接。如需應用程式執行器連線的詳細資訊，請參閱the section called “連線”。

當您建立服務時，App Runner 會建置包含應用程式程式碼和相依性的 Docker 映像檔。然後，它會啟動執行此映像檔之容器執行個體的服務。

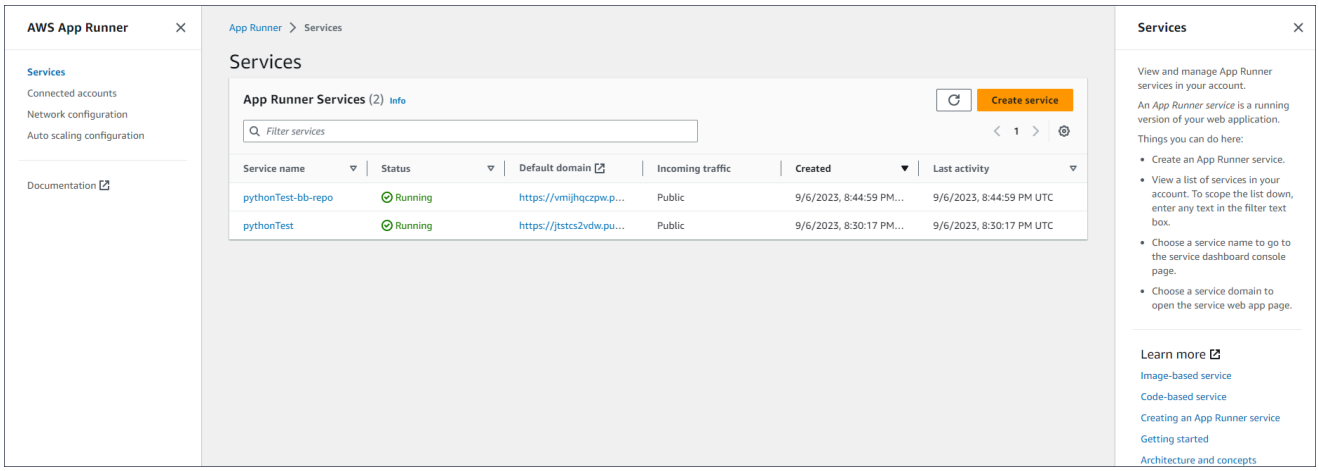
使用應用程式執行器主控台從程式碼建立服務

使用主控台建立應用程式執行器服務

1. 設定您的原始程式碼。
   a. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
   b. 如果還 AWS 帳戶 沒有任何應用程序運行器服務，則顯示控制台主頁。選擇創建應用程序運行器服務。



如果 AWS 帳戶 已有服務，則會顯示含有您服務清單的「服務」頁面。選擇 Create service (建立服務)。



c. 在「來源與部署」頁面的「來源」段落中，選擇「來源程式碼儲存區域」做為「儲存區域」類型。

d. 選取提供者類型。選擇任一GitHub或比特桶。

e. 接下來，為您之前使用過的提供者選取帳戶或組織，或選擇 [新增]。然後，完成提供代碼存儲庫憑據並選擇要連接的帳戶或組織的過程。

f. 在「存放庫」中，選取包含應用程式程式碼的儲存庫。

g. 針對「分支」，選取您要部署的分支。

h. 在來源目錄中，輸入儲存應用程式程式碼和組態檔的來源儲存庫中的目錄。

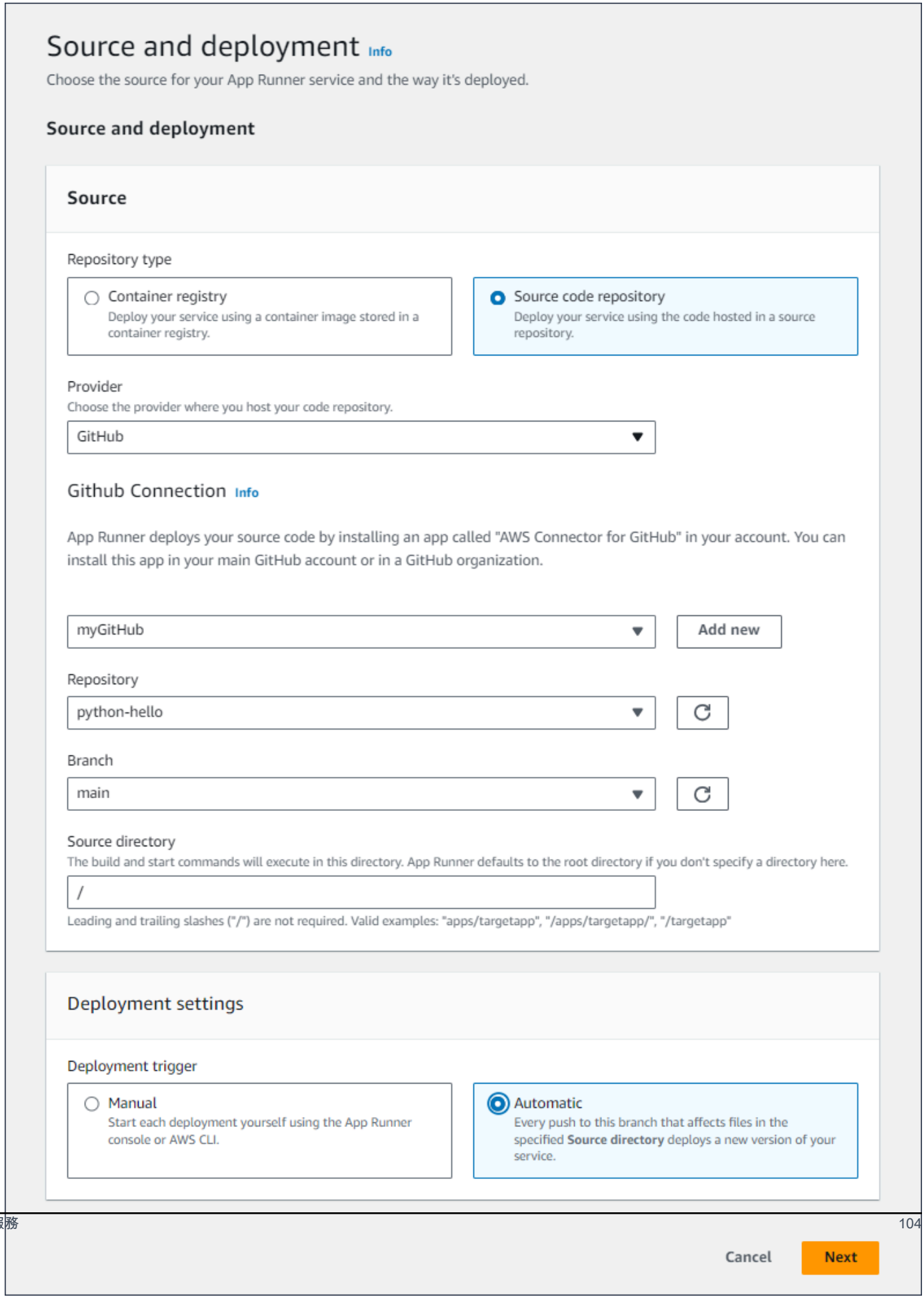
> **Note**
>
> 建置和 start 指令會從您指定的來源目錄執行。應用程序運行器從根處理絕對路徑。如果您未在此處指定值，則目錄預設為儲存庫根目錄。

2. 設定您的部署。

a. 在「部署設定」區段中，選擇「手動」或「自動」。

   如需部署方法的更多資訊，請參閱the section called “部署方法”。

b. 選擇下一步。


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source and deployment
Source
Repository type
Container registry
Deploy your service using a container image stored in a container registry.
Source code repository
Deploy your service using the code hosted in a source repository.
Provider
Choose the provider where you host your code repository.
GitHub
Github Connection Info
App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.
myGitHub
Add new
Repository
python-hello
Branch
main
Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.
/
Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
Every push to this branch that affects files in the specified Source directory deploys a new version of your service.
Cancel
Next

3. 設定應用程式組建。

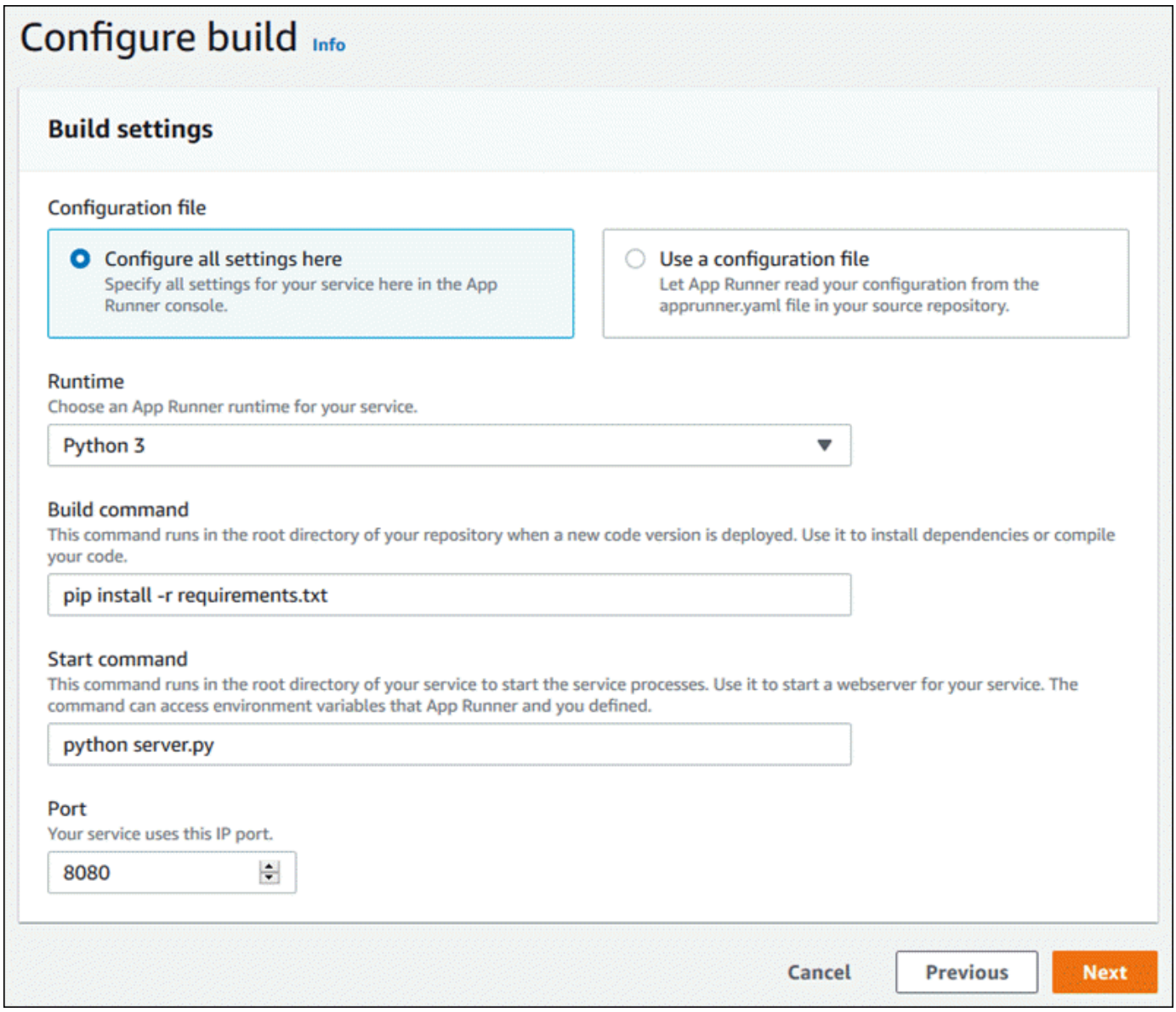
   a. 在 [設定組建] 頁面上，對於 [組態檔案]，選擇 [設定所有設定] (如果您的存放庫不包含 App Runner 組態檔案)，或選擇 [使用組態檔案 (如果有)]。

      > **Note**
      >
      > App Runner 配置文件是將構建配置作為應用程序源的一部分進行維護的一種方法。當您提供一個時，App Runner 會從文件中讀取一些值，並且不允許您在控制台中設置它們。

   b. 提供下列組建設定：

      - 執行階段 — 選擇應用程式的特定受管理執行階段。
      - 建置命令 — 輸入從原始程式碼建置應用程式的命令。這可能是語言特定的工具或程式碼隨附的指令碼。
      - 啟動命令 — 輸入啟動 Web 服務的指令。
      - 連接埠 — 輸入 Web 服務接聽的 IP 連接埠。

   c. 選擇下一步。



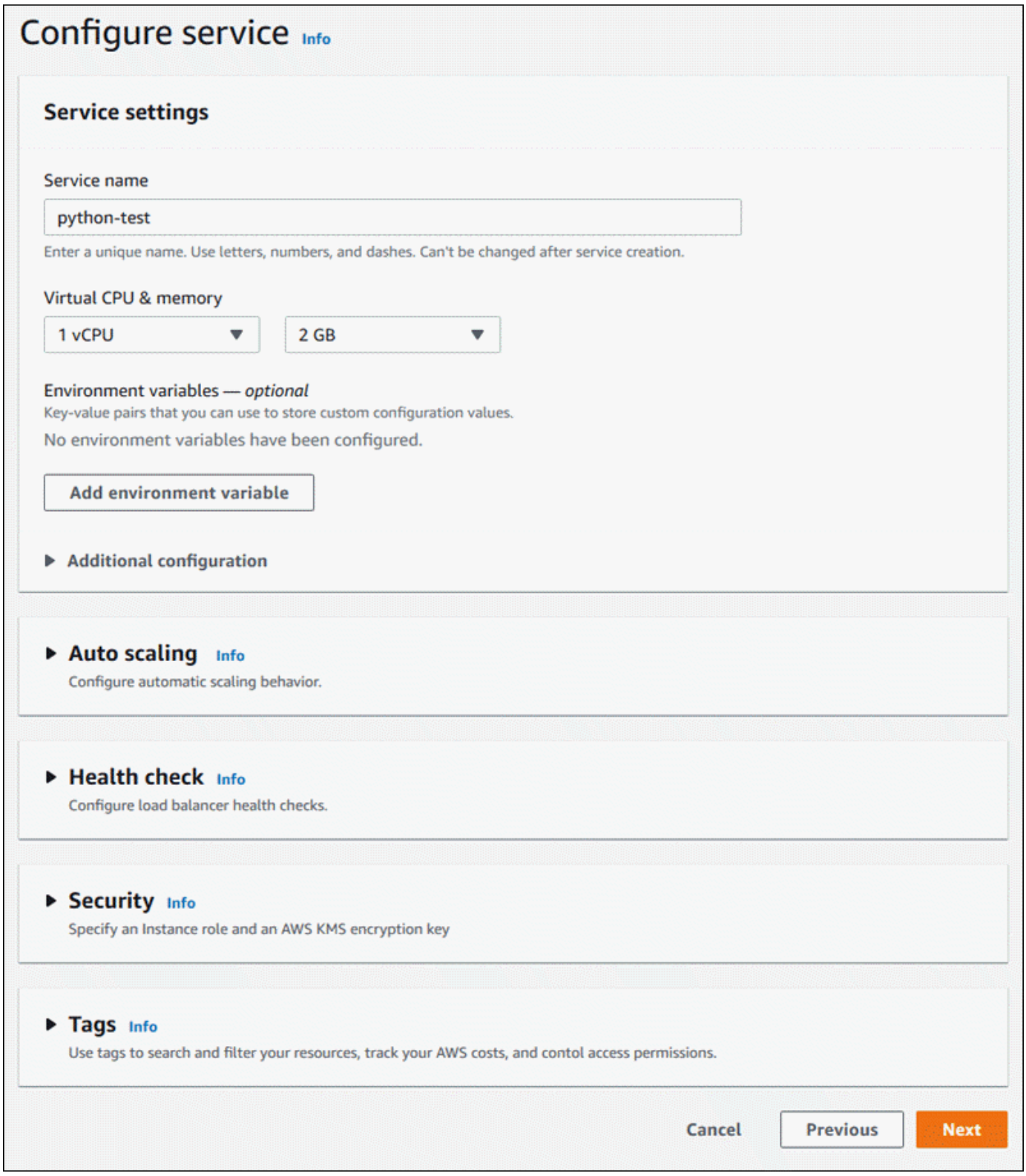
4. 設定您的服務。

   a. 在 [設定服務] 頁面的 [服務設定] 區段中，輸入服務名稱。

      > **Note**
      >
      > 所有其他服務設定都是選擇性的，或是具有主控台提供的預設值。

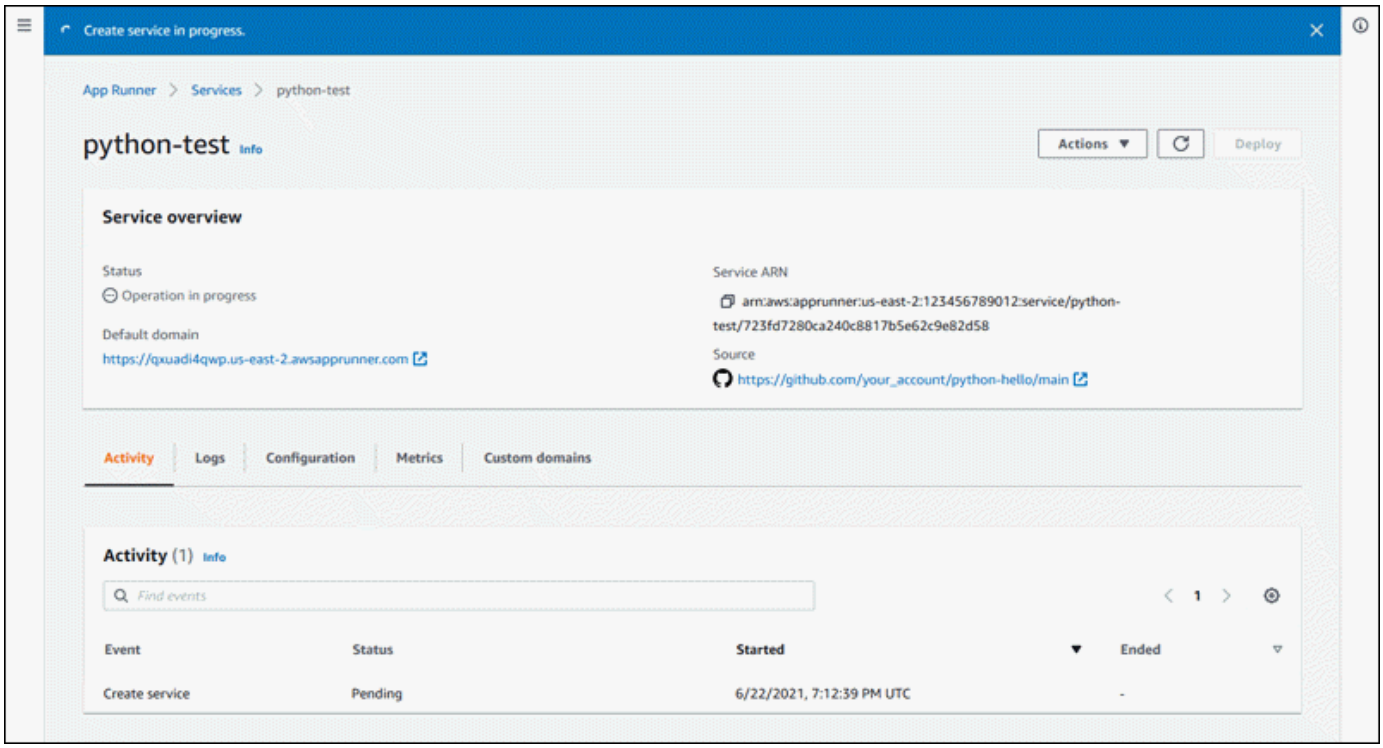
   b. 選擇性地變更或新增其他設定以符合您的應用程式需求。

   c. 選擇下一步。



5. 在 [檢閱並建立] 頁面上，確認您輸入的所有詳細資料，然後選擇 [建立並部署]。

結果：如果成功建立服務，主控台會顯示服務儀表板，其中包含新服務的「服務」概觀。



6. 確認您的服務正在執行。

   a. 在服務儀表板頁面上，等待服務狀態為「執行中」。

   b. 選擇預設網域值。這是您服務網站的 URL。

   c. 使用您的網站並驗證網站是否正常運行。

使用應用程式執行器 API 或從程式碼建立服務 AWS CLI

若要使用應用程式執行器 API 建立服務 AWS CLI，或呼叫 `CreateService` API 動作。如需詳細資訊和範例，請參閱CreateService。如果這是您第一次使用原始程式碼儲存庫 (或 Bitbucket) 的特定組織 GitHub 或帳戶建立服務，請先呼叫CreateConnection。這會在 App Runner 和儲存庫提供者的組織或帳戶之間建立連線。如需應用程式執行器連線的詳細資訊，請參閱the section called “連線”。

如果呼叫傳回顯示 Service 物件的成功回應`"Status": "CREATING"`，您的服務就會開始建立。

有關調用示例，請參閱 AWS App Runner API 參考中的創建源代碼存儲庫服務

從 Amazon ECR 映像創建服務

以下各節說明當您的來源是儲存在 Amazon ECR 中的容器映像時，如何建立應用程式執行器服務。Amazon ECR 是一個 AWS 服務. 因此，若要根據 Amazon ECR 映像建立服務，您需要為應用程式執行者提供包含必要 Amazon ECR 動作許可的存取角色。

> **Note**
>
> 存放在 Amazon ECR 公共圖像可公開使用。因此，如果您的映像檔儲存在 Amazon ECR 公用中，則不需要存取角色。

建立您的服務時，App Runner 會啟動執行您提供之映像檔的容器執行個體的服務。在這種情況下沒有構建階段。

如需詳細資訊，請參閱 影像式服務。

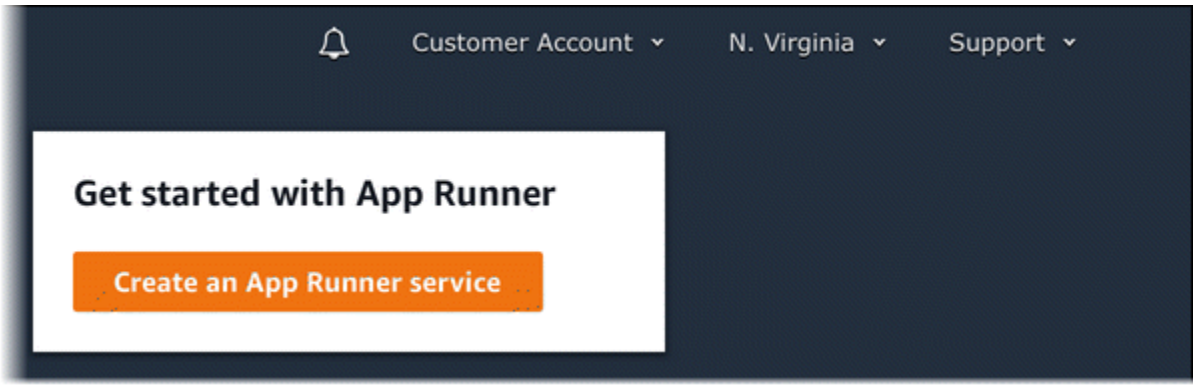
使用應用程式執行器主控台從映像建立服務

使用主控台建立應用程式執行器服務

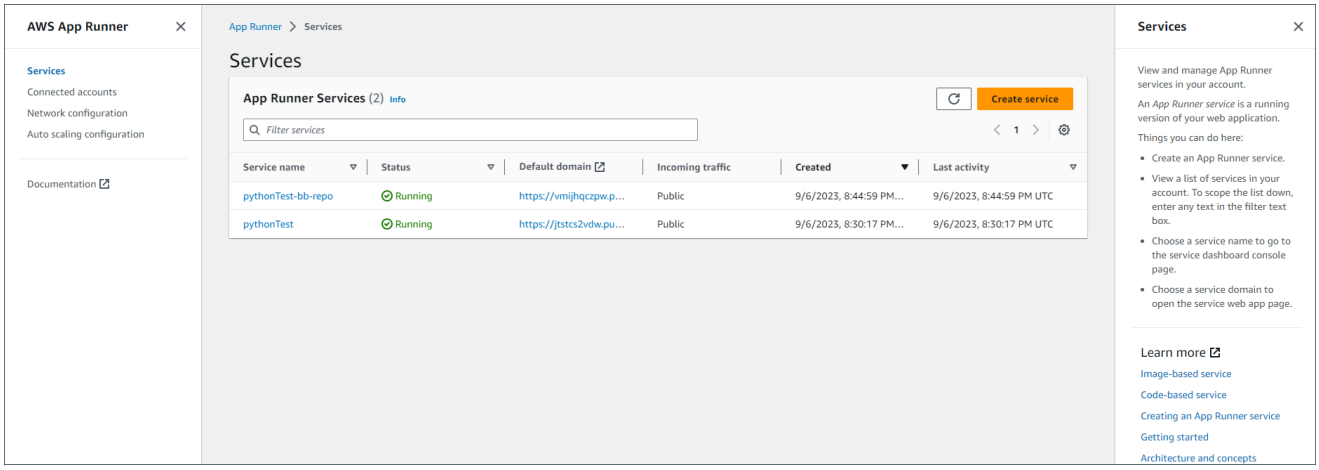
1. 設定您的原始程式碼。

   a. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.

   b. 如果還 AWS 帳戶 沒有任何應用程序運行器服務，則顯示控制台主頁。選擇創建應用程序運行器服務。



   如果 AWS 帳戶 已有服務，則會顯示含有您服務清單的「服務」頁面。選擇 Create service (建立服務)。



c. 在 [來源與部署] 頁面的 [來源] 區段中，選擇 [容器登錄] 做為 [存放庫類型]。

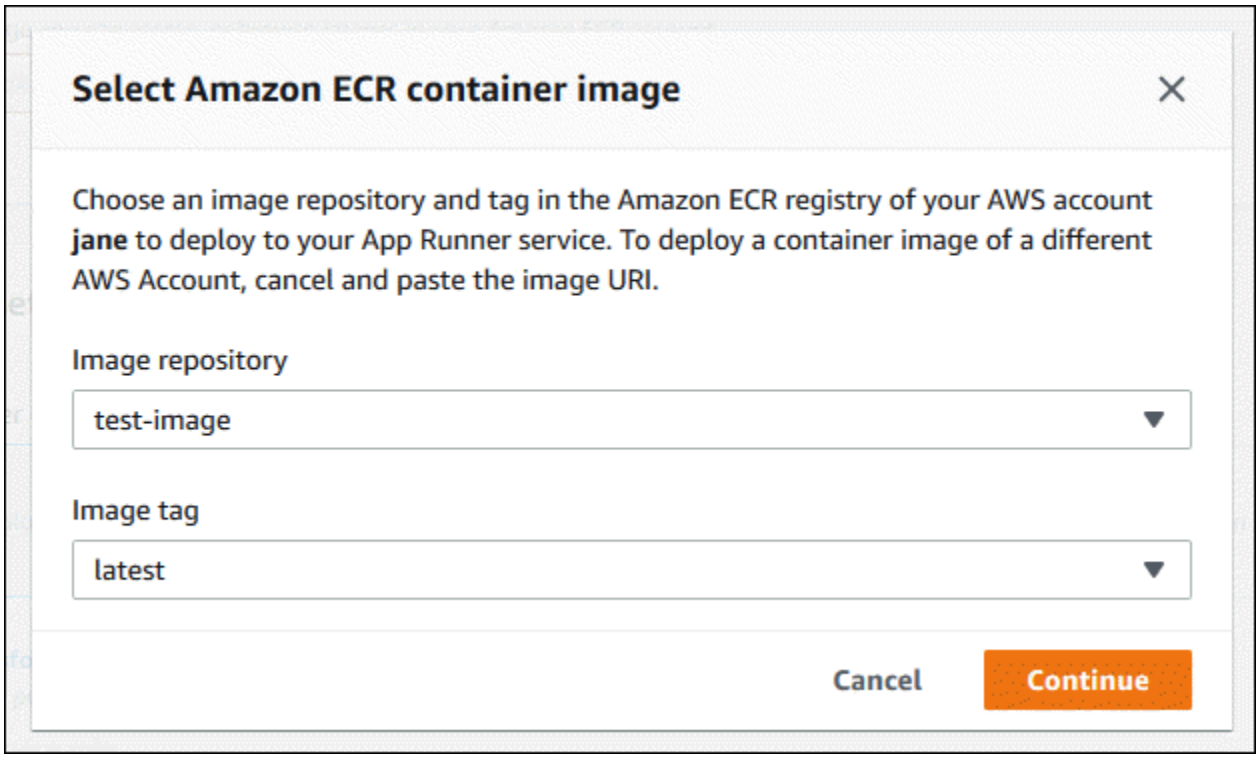
d. 針對「提供者」，請選擇儲存影像的提供者：

- Amazon ECR — 存儲在 Amazon ECR 中的私有映像。
- Amazon ECR 公共圖像 — 存儲在 Amazon ECR 公共區域中的可公開讀取圖像。

e. 針對「容器映像 URI」，選擇「瀏覽」。

f. 在 [選取 Amazon ECR 容器映像] 對話方塊中，對於映像儲存庫，選取包含映像的存放庫。

g. 針對映像標籤，選取您要部署的特定映像標記 (例如最新)，然後選擇 [繼續]。



2. 設定您的部署。

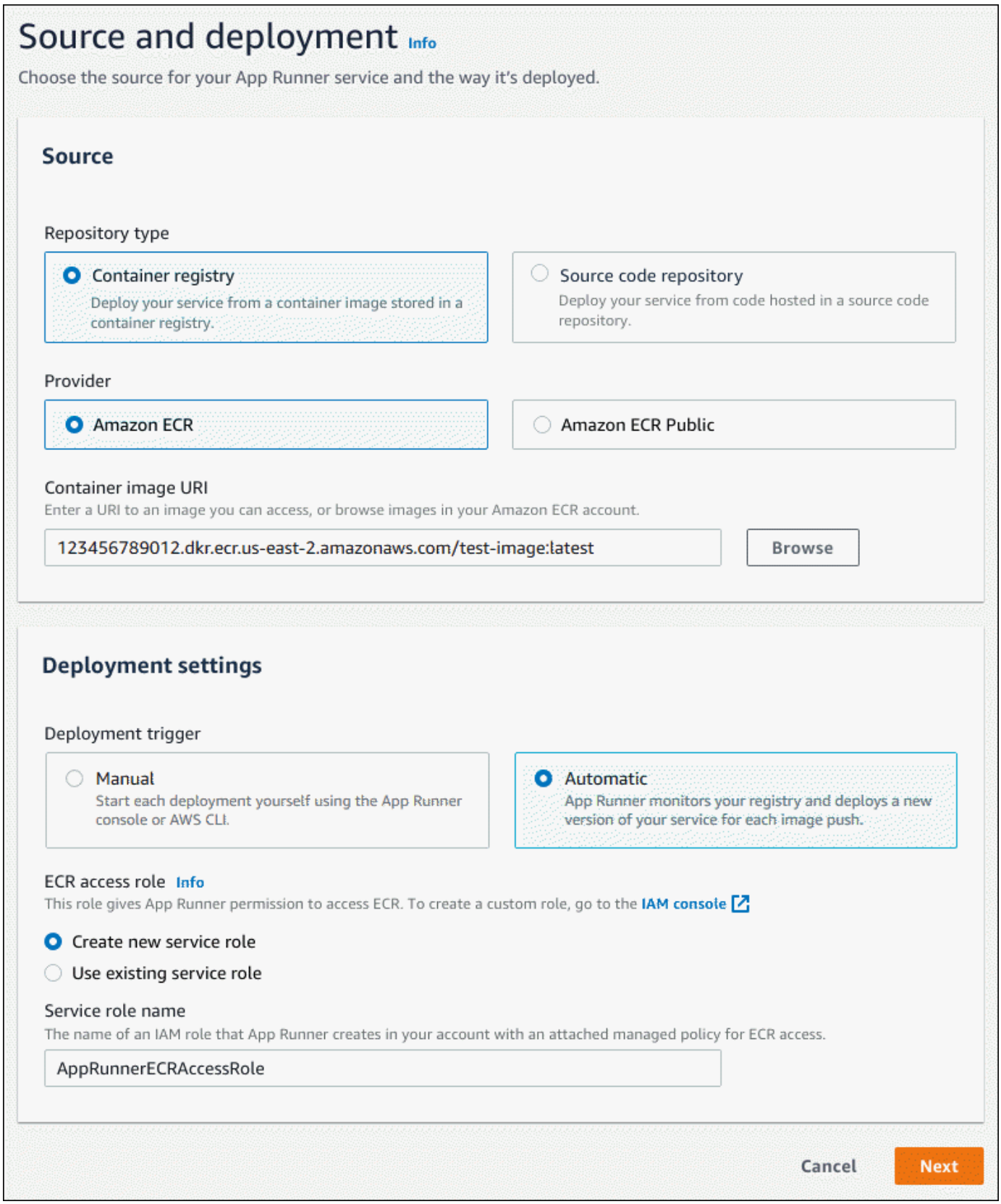
   a. 在「部署設定」區段中，選擇「手動」或「自動」。

   > **Note**
   >
   > 應用程式執行器不支援 Amazon ECR 公開映像檔的自動部署，以及 Amazon ECR 儲存庫中屬於與您服務所在 AWS 帳戶不同的帳戶的映像。

   如需部署方法的更多資訊，請參閱the section called “部署方法”。

   b. [Amazon ECR 提供者] 對於 ECR 存取角色，請選擇帳戶中的現有服務角色，或選擇建立新角色。如果您使用手動部署，也可以選擇在部署時使用 IAM 使用者角色。

   c. 選擇下一步。


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source
Repository type
Container registry
Deploy your service from a container image stored in a container registry.
Source code repository
Deploy your service from code hosted in a source code repository.
Provider
Amazon ECR
Amazon ECR Public
Container image URI
Enter a URI to an image you can access, or browse images in your Amazon ECR account.
123456789012.dkr.ecr.us-east-2.amazonaws.com/test-image:latest
Browse
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
App Runner monitors your registry and deploys a new version of your service for each image push.
ECR access role Info
This role gives App Runner permission to access ECR. To create a custom role, go to the IAM console
Create new service role
Use existing service role
Service role name
The name of an IAM role that App Runner creates in your account with an attached managed policy for ECR access.
AppRunnerECRAccessRole
Cancel
Next

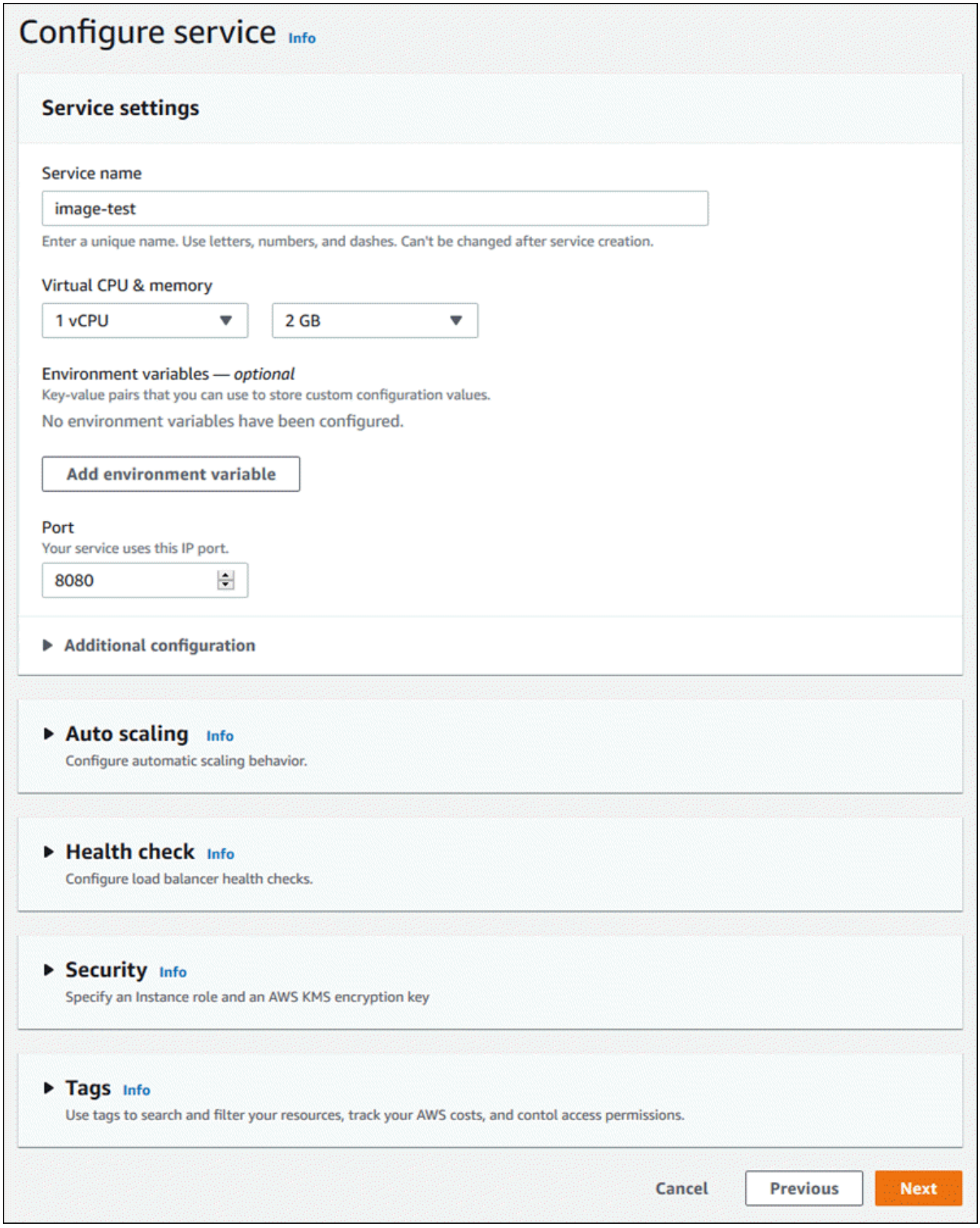
3. 設定您的服務。

   a. 在 [設定服務] 頁面的 [服務設定] 區段中，輸入服務名稱和服務網站接聽的 IP 連接埠。

   > **Note**
   >
   > 所有其他服務設定都是選擇性的，或是具有主控台提供的預設值。

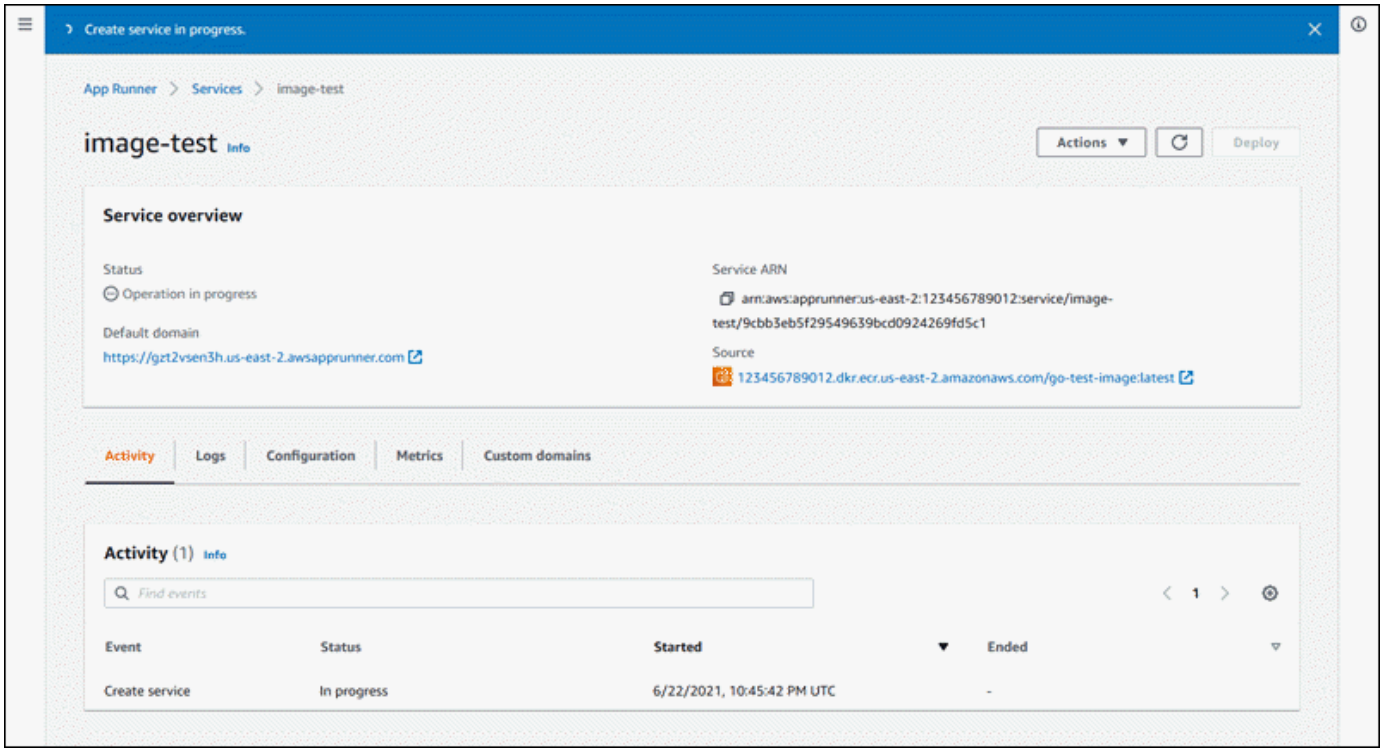
   b. (選擇性) 變更或新增其他設定以符合應用程式的需求。

   c. 選擇下一步。


Configure service Info
Service settings
Service name
image-test
Enter a unique name. Use letters, numbers, and dashes. Can't be changed after service creation.
Virtual CPU & memory
1 vCPU
2 GB
Environment variables — optional
Key-value pairs that you can use to store custom configuration values.
No environment variables have been configured.
Add environment variable
Port
Your service uses this IP port.
8080
Additional configuration
Auto scaling Info
Configure automatic scaling behavior.
Health check Info
Configure load balancer health checks.
Security Info
Specify an Instance role and an AWS KMS encryption key
Tags Info
Use tags to search and filter your resources, track your AWS costs, and contol access permissions.
Cancel
Previous
Next

4. 在 [檢閱並建立] 頁面上，確認您輸入的所有詳細資料，然後選擇 [建立並部署]。

   結果：如果成功建立服務，則主控台會顯示服務儀表板，以及新服務的「服務」概觀。



5. 確認您的服務正在執行。

   a. 在服務儀表板頁面上，等待服務狀態為「執行中」。
   b. 選擇預設網域值。這是您服務網站的 URL。
   c. 使用您的網站並驗證網站是否正常運行。

使用應用程序運行器 API 或從圖像創建服務 AWS CLI

若要使用應用程式執行器 API 建立服務 AWS CLI，或呼叫 CreateServiceAPI 動作。

如果呼叫傳回成功回應且顯示 Service 物件，您的服務建立就會開始`"Status": "CREATING"`。

有關調用示例，請參閱 AWS App Runner API 參考中的創建源映像存儲庫服務

重建失敗的應用程式執行器服務

如果您在建立 App Runner 服務時收到建立失敗的錯誤訊息，您可以執行下列其中一項動作。

- 請按照中the section called “無法建立服務”的步驟識別錯誤的原因。

- 如果在來源或組態中發現錯誤，請進行必要的變更，然後重建服務。
- 如果 App Runner 出現暫時性問題導致服務失敗，請重建失敗的服務，而不會對來源或設定進行任何變更。

您可以通過應用程序運行器控制台或應用程序運行器 API 或 AWS CLI.

使用應用程式執行程式主控台重建失敗的應用程式執行器

Rebuild with updates

建立服務可能會因為各種原因而失敗。發生這種情況時，在重建服務之前，確定並糾正問題的根本原因很重要。如需詳細資訊，請參閱 the section called “無法建立服務”。

使用更新重建失敗的服務

1. 前往服務頁面上的「組態」索引標籤，然後選擇「編輯」。

   頁面會開啟摘要面板，其中顯示所有更新的清單。
2. 進行必要的變更，並在摘要面板中檢閱這些變更。
3. 選擇 [儲存並重建]。

   您可以在服務頁面的 [記錄] 索引標籤上監控進度。

Rebuild without updates

如果暫時性問題導致服務建立失敗，您可以在不修改其來源或組態設定的情況下重建服務。

若要重建失敗的服務而不更新

- 選擇服務頁面右上角的「重建」。

  您可以在服務頁面的 [記錄] 索引標籤上監控進度。
- 如果您的服務無法再次建立，請依照中的疑難排解指示進行the section called “無法建立服務”。進行必要的變更，然後重建您的服務。

使用應用程序運行器 API 或重建失敗的應用程序運行器服務 AWS CLI

Rebuild with updates

若要重建失敗的服務：

1. 按照中的說the section called “無法建立服務”明找出錯誤的原因。
2. 對來源儲存庫的分支或映像檔或造成錯誤的組態進行必要的變更。
3. 透過使用新的原始程式碼儲存庫或來源映像檔儲存庫參數呼叫 UpdateServiceAPI 動作來重建。應用程序運行器從源代碼存儲庫中檢索最新的提交。

Example 使用更新進行重建

在下列範例中，映像式服務的來源組態正在更新。的值會變更Port為80。

更新基於圖像的應用程序運行器服務的input.json文件

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageConfiguration": {
        "Port": "80"
      }
    }
  }
}
```

呼叫 UpdateService API 動作。

```
aws apprunner update-service
--cli-input-json file://input.json
```

Rebuild without updates

若要使用 App Runner API 重建失敗的服務 AWS CLI，或是呼叫 UpdateServiceAPI 動作而不對服務的來源或設定進行任何變更。僅當您的服務創建由於應用程序 Runner 的臨時問題而失敗時，才選擇重建而不進行更新。

部署新的應用程式版本到應用程式執行

當您在中建立服務時 AWS App Runner，可以設定應用程式來源 — 容器映像或來源存放庫。應用程序運行佈建資源來運行您的服務，並將您的應用程序部署到他們。

本主題說明當新版本可用時，將應用程式來源重新部署至 App Runner 服務的方法。這可以是映像存儲庫中的新映像版本，也可以是代碼存儲庫中的新提交。應用程序 Runner 提供了兩種方法來部署到服務：自動和手動。

部署方法

App Runner 提供下列方法，讓您控制應用程式部署的起始方式。

自動部署

當您想要服務的持續整合與部署 (CI/CD) 行為時，請使用自動部署。應用程序運行器監視您的圖像或代碼存儲庫的更改。

圖像存儲庫 — 每當您將新的映像版本推送到映像存儲庫，或將新的提交推送到代碼存儲庫時，App Runner 都會自動將其部署到您的服務，而無需採取進一步的操作。

代碼存儲庫-每當您將新的提交推送到代碼存儲庫中進行更改的源目錄時，App Runner 都會部署您的整個存儲庫。因為只有來源目錄中的變更才會觸發自動部署，因此請務必瞭解來源目錄位置如何影響自動化部署的範圍。

- 頂層目錄 (存放庫根目錄) — 這是您建立服務時為來源目錄設定的預設值。如果您的源目錄設置為此值，這意味著整個存儲庫位於源目錄中。因此，在這種情況下，您推送到源存儲庫的所有提交都將觸發部署。
- 任何不是儲存庫根目錄的目錄路徑 (非預設值) — 因為只有在來源目錄內推送的變更才會觸發自動部署，所以推送至儲存庫不在來源目錄中的任何變更都不會觸發自動部署。因此，您必須使用手動部署來部署您在來源目錄之外推送的變更。

> **Note**
>
> 應用程式執行器不支援 Amazon ECR 公開映像檔的自動部署，以及 Amazon ECR 儲存庫中屬於與您服務所在 AWS 帳戶不同的帳戶的映像。

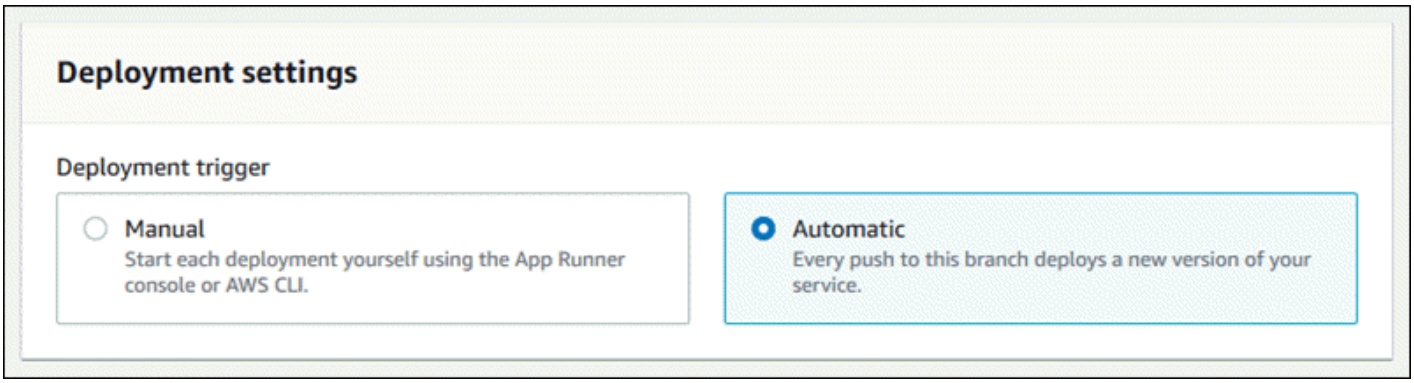
手動部署

當您想要明確地初始化服務的每個部署時，請使用手動部署。如果您為服務設定的存放庫具有您要部署的新版本，則可以啟動部署。如需詳細資訊，請參閱 the section called “手動部署”。

> **Note**
>
> 當您執行手動部署時，App Runner 會從完整存放庫部署來源。

您可以使用下列方式設定服務的部署方法：

- 主控台 — 對於您正在建立的新服務或現有服務，請在 [來源和部署設定] 頁面的 [部署設定] 區段中，選擇 [手動] 或 [自動]。



- API 或 AWS CLI — 在呼叫或動UpdateService作時，將SourceConfiguration參數`AutoDeploymentsEnabled`成員設定`False`為以進行手動部署或`True`用於自動部署。CreateService

> **比較自動與手動部署**
>
> 自動和手動部署都會產生相同的結果：這兩種方法都會部署完整存放庫。
> 這兩種方法之間的區別在於觸發機制：
>
> - 手動部署是透過主控台的部署、呼叫或呼叫 App Runner API 來觸發。 AWS CLI接下來的手動部署章節提供了這些程序的程序。
> - 自動部署是由來源目錄內容中的變更觸發。

手動部署

透過手動部署，您需要明確地啟動服務的每個部署。當您準備好要部署的新版應用程式映像檔或程式碼時，您可以參考以下各節，瞭解如何使用主控台和 API 執行部署。

> **Note**
>
> 當您執行手動部署時，App Runner 會從完整存放庫部署來源。

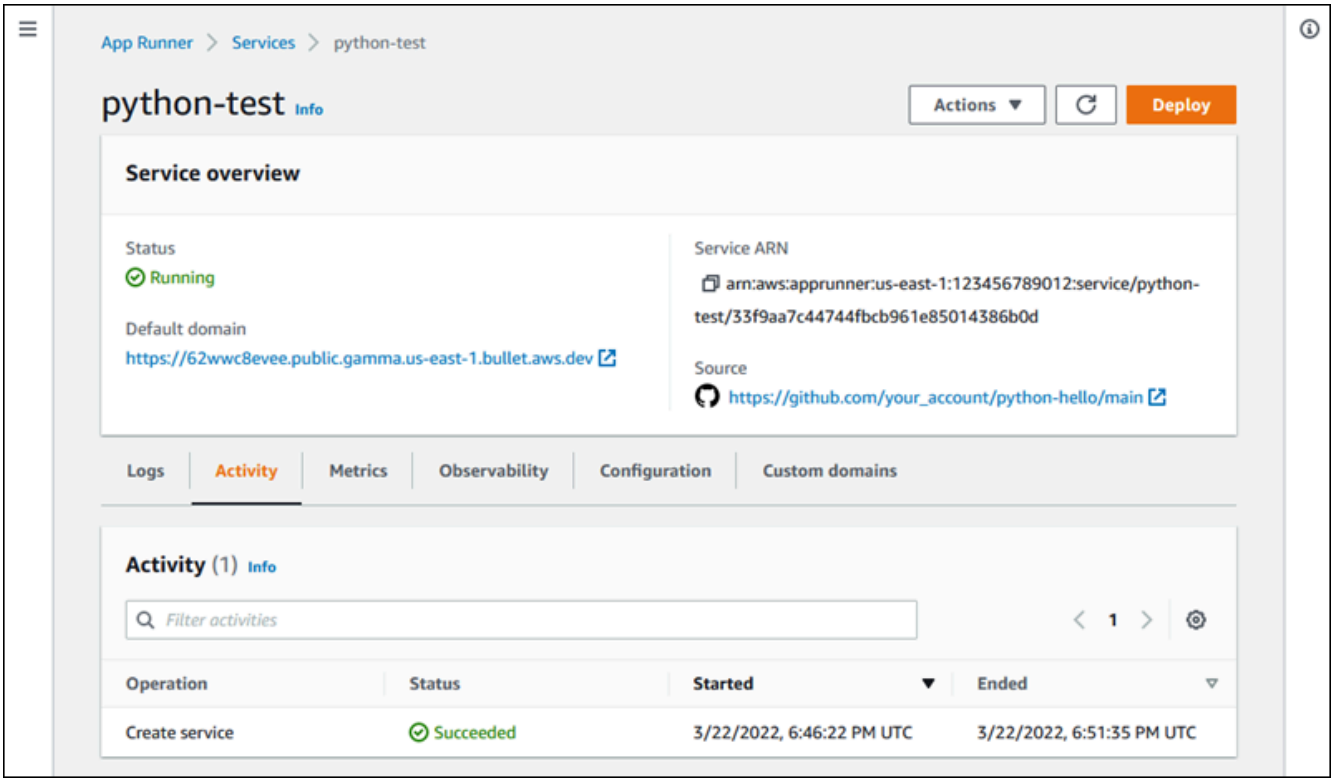
使用下列其中一種方法部署應用程式版本：

App Runner console

若要使用應用程式執行程式主控台

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板，其中包含服務概觀。



3. 選擇部署。

結果：開始部署新版本。在服務儀表板頁面上，服務狀態會變更為 [作業進行中]。

4. 等待部署結束。在服務儀表板頁面上，服務狀態應變回執行中。
5. 若要驗證部署是否成功，請在服務儀表板頁面上，選擇預設網域值 — 這是您服務網站的 URL。檢查您的 Web 應用程式或與其互動，並驗證您的版本變更。

> **Note**
>
> 為了增強應用程式執行器應用程式的安全性，*.awsapprunner.com 網域註冊在公用尾碼清單 (PSL) 中。為了進一步的安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

App Runner API or AWS CLI

若要使用應用程式執行器 API 進行部署 AWS CLI，或呼叫 StartDeploymentAPI 動作。唯一要傳遞的參數是您的服務 ARN。您在建立服務時已設定應用程式來源位置，而 App Runner 可以尋找新版本。如果呼叫傳回成功的回應，您的部署就會開始。

設定應用程式執行器服務

當您建立 AWS App Runner 服務時，您可以設定各種組態值。您可以在建立服務之後變更其中一些組態設定。其他設定只能在建立服務時套用，之後無法變更。本主題討論使用 App Runner API、應用程式執行器主控台和應用程式執行器設定檔的服務設定。

主題

- 使用應用程式執行器 API 設定您的服務，或 AWS CLI
- 使用應用程式執行器主控台設定服務
- 使用應用程序運行器配置文件配置服務
- 設定服務的可觀察性
- 使用可共用資源設定服務設定
- 設定服務的健康狀態檢查

使用應用程式執行器 API 設定您的服務，或 AWS CLI

該 API 定義了服務創建後可以更改哪些設置。下列清單討論相關動作、類型和限制。

- UpdateServiceaction — 可以在創建後調用以更新某些配置設置。
  - 可以更新 — 您可以更新SourceConfigurationInstanceConfiguration、和HealthCheckConfiguration參數中的設定。但是，在中SourceConfiguration，您無法將源類型從代碼切換到圖像或相反。您必須提供與建立服務時所提供的相同儲存庫參數。它是CodeRepository或ImageRepository。

    您也可以更新下列與服務相關聯之個別組態資源的 ARN：

    - AutoScalingConfigurationArn
    - VpcConnectorArn
  - 無法更新 — 您無法變更CreateService動作中可用的ServiceName和EncryptionConfiguration參數。建立之後，就無法變更它們。動UpdateService作不包含這些參數。
  - API 與檔案 — 您可以將CodeConfiguration類型的ConfigurationSource參數 (用於原始程式碼儲存庫的一部分SourceConfiguration) 設定為Repository。在這種情況下，App Runner 會忽略中的配置設置CodeConfigurationValues，並從存儲庫中的配置文件中讀取這些設置。如果設定ConfigurationSource為API，App Runner 會從 API 呼叫取得所有組態設定，並忽略組態檔案，即使有設定檔案也一樣。
- TagResourceaction — 可在建立服務後呼叫，以將標籤新增至服務或更新現有標籤的值。
- UntagResourceaction — 可在建立服務後呼叫，以移除服務中的標籤。

> **Note**
>
> 如果您為服務建立輸出流量 VPC 連接器，接下來的服務啟動程序將會遇到一次性延遲。您可以在建立新服務時或之後使用服務更新來設定此組態。如需詳細資訊，請參閱本指南的 < 使用應用程式執行器聯網 > 一章一次性延遲中的。

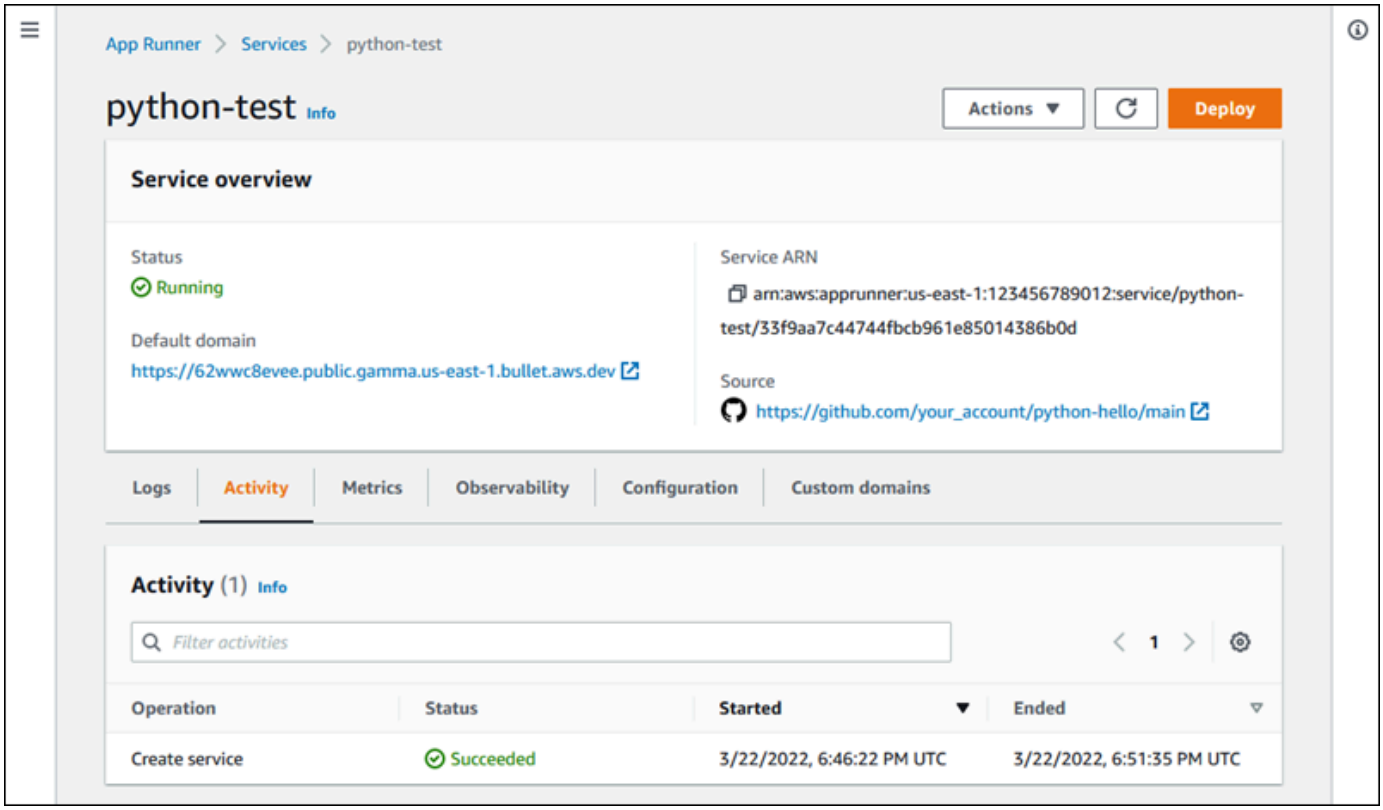
使用應用程式執行器主控台設定服務

主控台使用應用程式執行器 API 來套用設定更新。API 強制執行的更新規則 (如上一節所定義) 會決定您可以使用主控台進行設定的項目。在服務建立期間可用的某些設定，稍後無法修改。此外，如果您決定使用配置文件，則控制台中將隱藏其他設置，並且 App Runner 從文件中讀取它們。

若要設定您的服務

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇 [組態] 索引標籤。

   結果：主控台會在數個區段中顯示服務的目前組態設定：來源和部署、設定組建和設定服務。
4. 若要更新任何類別中的設定，請選擇 「編輯」。
5. 在 [組態編輯] 頁面上，進行所需的變更，然後選擇 [儲存變更]。

> **Note**
>
> 如果您為服務建立輸出流量 VPC 連接器，接下來的服務啟動程序將會遇到一次性延遲。您可以在建立新服務時或之後使用服務更新來設定此組態。如需詳細資訊，請參閱本指南的 < 使用應用程式執行器聯網 > 一章一次性延遲中的。

使用應用程序運行器配置文件配置服務

當您建立或更新 App Runner 服務時，您可以指示 App Runner 從您提供做為來源儲存庫一部分的設定檔讀取某些組態設定。通過這樣做，您可以在源代碼控制下管理與源代碼相關的設置以及代碼本身。配置文件還提供了某些您無法使用控制台或 API 設置的高級設置。如需詳細資訊，請參閱 應用程序運行器配置。

> **Note**
>
> 如果您為服務建立輸出流量 VPC 連接器，接下來的服務啟動程序將會遇到一次性延遲。您可以在建立新服務時或之後使用服務更新來設定此組態。如需詳細資訊，請參閱本指南的 < 使用應用程式執行器聯網 > 一章一次性延遲中的。

設定服務的可觀察性

AWS App Runner 與多種 AWS 服務集成，為您的 App Runner 服務提供廣泛的可觀察性工具套件。如需詳細資訊，請參閱 可觀測性。

App Runner 支持啟用一些可觀察性功能，並通過使用稱為. ObservabilityConfiguration 您可以在建立或更新服務時提供可觀察性組態資源。當您創建新的應用程序運行器服務時，應用程序運行器控制台為您創建一個。提供可觀察性配置是可選的。如果您不提供，App Runner 會提供預設的可觀測性設定。

您可以在多個 App Runner 服務之間共用單一可觀察性設定，以確保它們具有相同的可觀察性行為。如需詳細資訊，請參閱 the section called “配置資源”。

您可以使用可觀測性組態設定下列可觀察性功能：

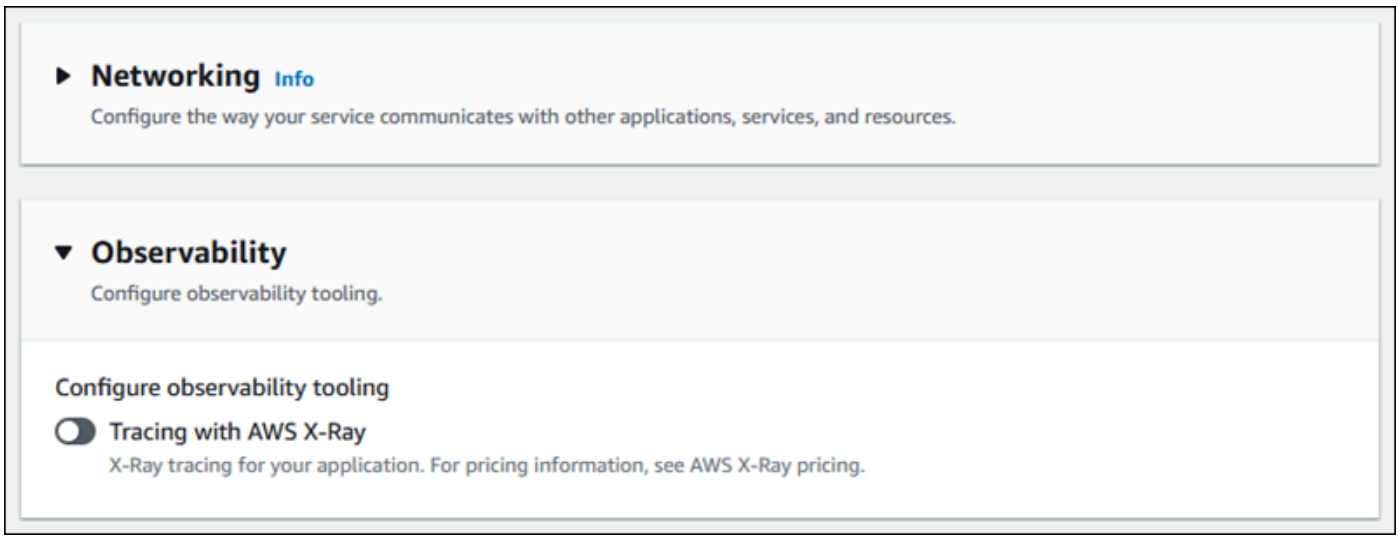
- 追蹤組態 — 追蹤應用程式所提供之要求及其發出之下游呼叫的設定。如需追蹤的詳細資訊，請參閱：the section called “X-Ray 追蹤”。

管理可觀測性

使用下列其中一種方法管理 App Runner 服務的可觀察性：

App Runner console

當您使用 App Runner 主控台建立服務時，或稍後更新其組態時，您可以為服務設定可觀察性功能。查找控制台頁面上的觀察性配置部分。



App Runner API or AWS CLI

當您呼叫CreateService或 UpdateServiceApp Runner API 動作時，您可以使用`ObservabilityConfiguration`參數物件來啟用可觀察性功能，並為您的服務指定可觀察性設定資源。

使用下列應用程式執行器 API 動作來管理您的可觀察性設定資源。

- CreateObservabilityConfiguration— 建立新的可觀測性規劃或對現有組態的修訂。
- ListObservabilityConfigurations— 傳回與您相關聯的觀察性組態清單 AWS 帳戶，以及摘要資訊。
- DescribeObservabilityConfiguration— 傳回可觀測性組態的完整描述。
- DeleteObservabilityConfiguration— 刪除可觀測性組態。您可以刪除特定修訂版本或最新的使用中版本修訂。如果您達到. AWS 帳戶

使用可共用資源設定服務設定

對於某些功能，跨 AWS App Runner 服務共享配置是有意義的。例如，您可能希望一組服務具有相同的 auto 擴展行為。或者，您可能需要為所有服務提供相同的可觀察性設置。應用程序運行器允許您使用單獨的可共享資源共享設置。您建立定義功能組態設定的資源，然後將此組態資源的 Amazon 資源名稱 (ARN) 提供給一或多個 App Runner 服務。

應用程序 Runner 為以下功能實現了可共享的配置資源：

- 自動擴展

- 可觀測性
- VPC 存取

這些功能的文件頁面提供有關可用設定和管理程序的資訊。

使用不同組態資源的功能會共用一些設計特徵和考量。

- 修訂版 — 某些組態資源可以有修訂版本。自動擴展和可觀察性是使用修訂版的兩個配置資源的示例。在這些情況下，每個模型組態都有一個名稱和一個數字版本修訂。模型組態的多個修訂版本具有相同的名稱和不同的修訂版號碼。您可以針對不同的案例使用不同的組態名稱。對於每個名稱，您可以新增多個修訂，以微調特定案例的設定。

  您以名稱建立的第一個模型組態會取得修訂版號碼 1。具有相同名稱的後續模型組態會獲得連續的修訂版號碼（從 2 開始）。您可以將 App Runner 服務與特定的配置修訂版本或配置的最新版本相關聯。
- 共用 — 您可以在多個應用程式執行器服務之間共用單一設定資源。如果您想要在這些服務之間維護相同的組態，這會很有用。特別是，如果您的資源支援修訂版本，您可以設定多個服務以使用組態的最新修訂版本。您可以透過僅指定模型組態名稱，而不是修訂版來執行此操作。您以此方式設定的任何服務都會在您更新服務時接收組態更新。如需有關組態變更的詳細資訊，請參閱the section called “組態”。
- 資源管理 — 您可以使用應用程式執行器來建立和刪除設定。您無法直接更新設定。相反地，對於支援修訂版本的資源，您可以建立現有組態名稱的新修訂版本，以有效地更新組態。

  > **Note**
  >
  > 對於 auto 擴展，您可以使用應用程序運行器控制台和應用程序運行器 API 創建配置和多個修訂。應用程序運行器控制台和應用程序運行器 API 也可以刪除配置和修訂。如需詳細資訊，請參閱管理應用程式執行器自動。
  > 對於其他配置類型（例如可觀測性配置），您只能使用 App Runner 控制台創建具有單個修訂版的配置。若要建立更多修訂，並刪除設定，您必須使用應用程式執行器 API。

- 資源配額 — 您可以為每個配置資源設定的唯一組態名稱和修訂版本數量設定配額 AWS 區域。如果達到這些配額，您必須先刪除組態名稱或至少部分修訂，才能建立更多配額。對於 auto 擴展配置修訂，您可以使用應用程序運行器控制台或應用程序運行器 API 將其刪除。如需詳細資訊，請參閱管理應用程式執行器自動。您必須使用應用程式執行器 API 來刪除其他資源。如需配額的詳細資訊，請參閱 the section called “應用運行器資源配額”。

- 無資源成本 — 您不會產生建立組態資源的額外費用。您可能會產生功能本身的費用（例如，當您開啟 X-Ray 追蹤時，您需要支付正常 AWS X-Ray 費用），但是對於為 App Runner 服務設定功能的 App Runner 組態資源則不會產生費用。

設定服務的健康狀態檢查

AWS App Runner 透過執行健康狀態檢查來監控服務的健全狀況。預設的健康狀態檢查通訊協定為 TCP。應用程序運行器 ping 分配給您的服務的域。您也可以將健全狀況檢查通訊協定設定為 HTTP。應用程序運行器發送健康檢查 HTTP 請求到您的 Web 應用程序。

您可以設定一些與健全狀況檢查相關的設定。下表說明健全狀況檢查設定及其預設值。

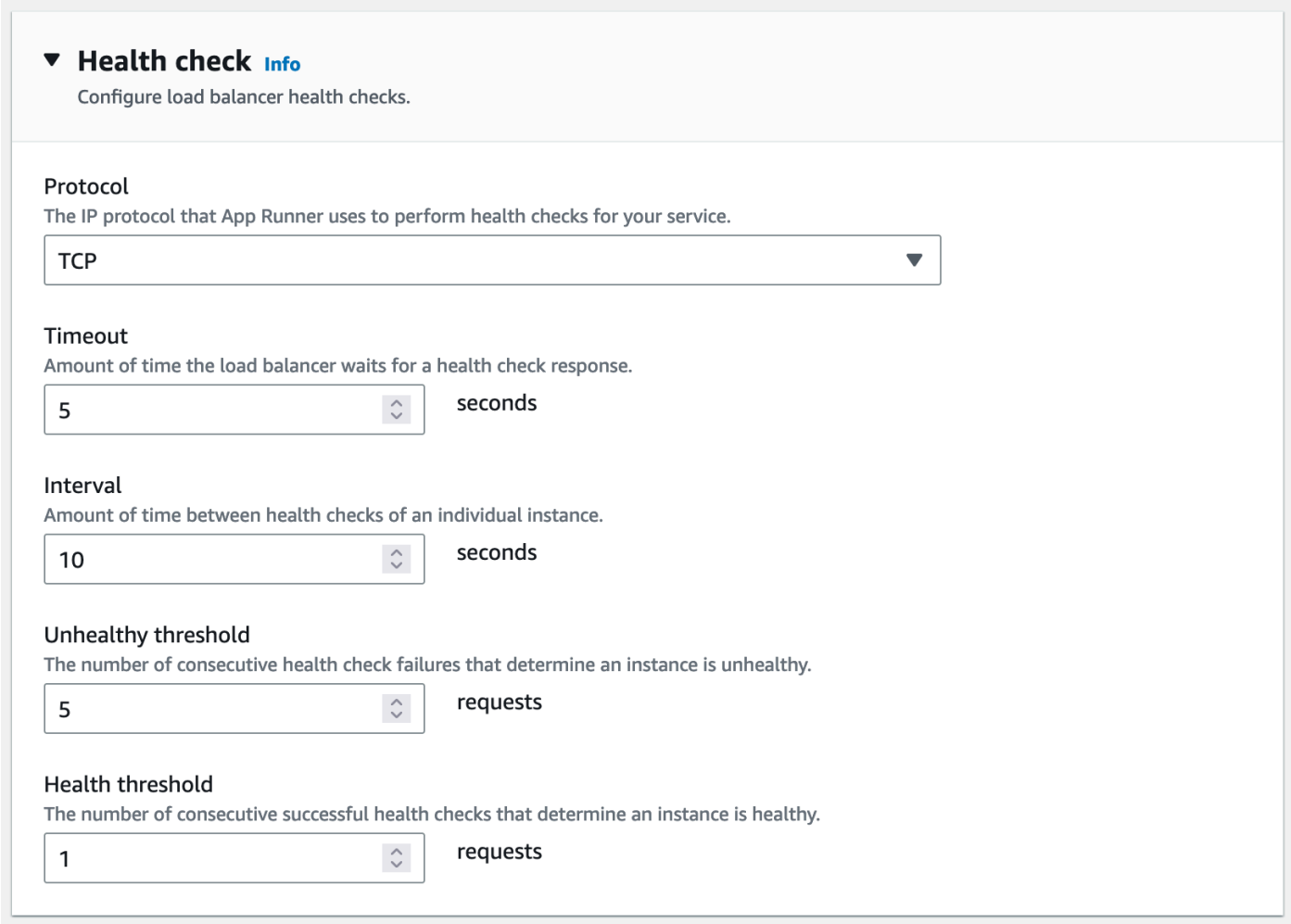
設定	描述	預設
通訊協定	App Runner 用來為您的服務執行運作狀態檢查的 IP 通訊協定。 如果您將通訊協定設定為TCP，App Runner 會在應用程式偵聽的連接埠上偵聽指派給服務的預設網域。 如果您將通訊協定設定為HTTP，App Runner 會將健康狀態檢查要求傳送至設定的路徑。	TCP
路徑	應用程序運行器向其發送 HTTP 健康檢查請求的 URL。僅適用於 HTTP 檢查。	/
Interval (間隔)	運作狀態檢查之間的時間間隔 (以秒為單位)。	5
逾時	在判斷運作狀態檢查回應失敗之前等待運作狀態檢查回應的時間 (以秒為單位)。	2
健康門檻	在 App Runner 判斷服務運作狀態良好之前，必須成功的連續檢查次數。	1
不健康的閾值	在 App Runner 判斷服務運作狀態不良之前，必須失敗的連續檢查次數。	5

設定運作狀態檢查

使用下列其中一種方法，為您的 App Runner 服務設定健康狀態檢查：

App Runner console

當您使用 App Runner 主控台建立 App Runner 服務時，或稍後更新其組態時，您可以設定健康狀態檢查設定。如需完整的主控台程序，請參閱the section called “建立”和the section called “組態”。在這兩種情況下，請尋找主控台頁面上的 [Health 全狀況檢查設定] 區段。



App Runner API or AWS CLI

呼叫CreateService或 UpdateServiceAPI 動作時，可以使用`HealthCheckConfiguration`參數來指定健康狀態檢查設定。

如需有關參數結構的資訊，請參閱 AWS App Runner API 參考資料HealthCheckConfiguration中的。

管理應用程式執行器

當您在中建立服務時 AWS App Runner，可以設定應用程式來源 — 容器映像檔或與提供者一起儲存的來源存放庫。應用程序 Runner 必須與提供商建立經過身份驗證和授權的連接。然後，App Runner 可以讀取您的存儲庫並將其部署到您的服務。當您創建訪問存儲在 AWS 帳戶.

應用程序運行器在稱為連接的資源中維護連接信息。App Runner 控制台和本指南也將連接稱為已連接的帳戶。當您建立需要第三方連線資訊的服務時，App Runner 需要連線資源。以下是關於連線的一些重要資訊：

- 提供商 — 應用程序運行器目前需要與GitHub或 Bitbucket 的連接資源。
- 共用 — 您可以使用連線資源來建立使用相同儲存庫提供者帳戶的多個 App Runner 服務。
- 資源管理 — 在 App Runner 中，您可以創建和刪除連接。不過，您無法修改現有的連線。
- 資源配額 — 連線資源具有與每個資源 AWS 帳戶 中的配額相關聯的設定配額 AWS 區域。如果達到此配額，您可能需要先刪除連線，才能連線到新的提供者帳戶。您可以使用 App Runner 主控台或 API 刪除連線，如下節所述the section called “管理連線”。如需詳細資訊，請參閱 the section called “應用運行器資源配額”。

管理連線

使用下列其中一種方法管理您的應用程式執行器連線：

App Runner console

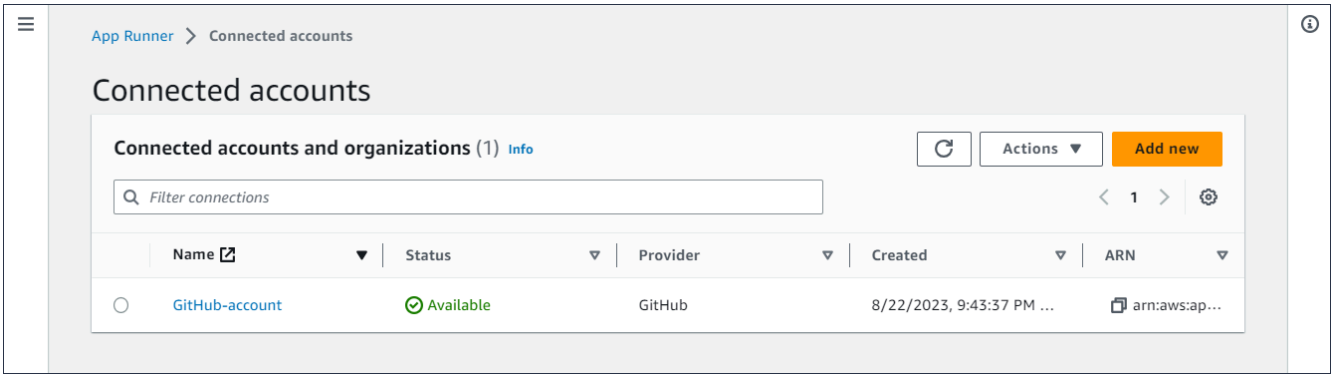
當您使用 App Runner 主控台建立服務時，您會提供連線詳細資料。您不必明確地建立連線資源。在主控台中，您可以選擇連線到之前連線過的 GitHub 或 Bitbucket 帳戶，或是連線到新帳戶。必要時，應用程序運行器為您創建一個連接資源。對於新的連線，某些提供者會要求您完成驗證交換，然後才能使用連線。主控台會引導您完成此程序。

控制台還具有用於管理現有連接的頁面。如果您在建立服務時未執行連線，則可以完成連線的驗證握手。您也可以刪除不再使用的連線。下列程序顯示如何管理儲存區域提供者連線。

管理您帳戶中的連線

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [連線的帳戶]。

   主控台接著會顯示您帳戶中的儲存庫提供者連線清單。



3. 您現在可以對清單上的任何連線執行下列其中一個動作：

   - 開啟 GitHub /Bitbucket 帳戶或組織 — 選擇連線名稱。
   - 完成認證交握 — 選取連線，然後從 [動作] 功能表中選擇 [完成交握]。主控台會引導您完成驗證交換程序。
   - [刪除連線] — 選取連線，然後從 [動作] 功能表中選擇 [刪除]。按照刪除提示上的說明進行操作。

App Runner API or AWS CLI

您可以使用下列應用程式執行器 API 動作來管理您的連線。

- CreateConnection— 建立存放庫提供者帳戶的連線。建立連線之後，您必須使用 App Runner 主控台手動完成驗證交握。此過程將在上一節中進行說明。
- ListConnections— 傳回與您關聯的應用程式執行器連線清單 AWS 帳戶。
- DeleteConnection— 刪除連線。如果您達到的連線配額，您可能需要刪除不必要的連線 AWS 帳戶。

管理應用程式執行器自動

AWS App Runner 自動為您的 App Runner 應用程式擴展或縮減運算資源，特別是執行個體。自動調整可在流量繁重時提供適當的要求處理，並在流量變慢時降低成本。

自動縮放配置

您可以設定幾個參數來調整服務的 auto 擴展行為。應用程序 Runner 在稱為AutoScalingConfiguration可共享資源中維護 auto 縮放設置。您可以先建立並維護獨立的 auto 資源調整設定，然後再將其指派給服務。將它們與服務相關聯之後，您可以繼續維護組態。您也可以在建立

新服務或設定現有服務的過程中，選擇建立新的自動擴展設定。建立新的 auto 資源調整配置後，您可以將其與服務建立關聯，並繼續建立或設定服務的程序。

命名和修訂

auto 調整比例配置具有名稱和數字版本修訂。模型組態的多個修訂版本具有相同的名稱和不同的修訂版號碼。您可以針對不同的 auto 擴展案例使用不同的組態名稱，例如高可用性或低成本。對於每個名稱，您可以新增多個修訂，以微調特定案例的設定。每個模型組態最多可以有 10 個唯一的 auto 調整資源配置名稱和最多 5 個修訂版。如果您達到上限且需要建立更多，則可以刪除一個，然後再建立另一個上限。App Runner 將不允許您刪除設置為默認設置或正在由活動服務使用的配置。如需配額的詳細資訊，請參閱 the section called “應用運行器資源配額”。

設定預設組態

當您建立或更新 App Runner 服務時，您可以提供 auto 調整規模設定資源。提供 auto 擴展配置是可選的。如果您未提供，App Runner 會提供預設的 auto 縮放設定，其中包含建議值。auto 縮放配置功能為您提供設置自己的默認 auto 縮放配置的選項，而不是使用 App Runner 提供的默認設置。一旦您指定另一個 auto 動擴展組態作為預設值，該組態就會自動指派為預設值給您 future 建立的新服務。新的預設指定不會影響先前為現有服務設定的關聯。

使用 auto 擴展來設定服務

您可以在多個 App Runner 服務之間共用單一 auto 擴展設定，以確保服務具有相同的 auto 擴展行為。如需使用 App Runner 主控台或 App Runner API 設定 auto 調整規模設定的詳細資訊，請參閱本主題後續章節。如需可共用資源的更多一般資訊，請參閱the section called “配置資源”。

可配置設置

您可以設定下列 auto 縮放設定：

- 最大並行處理 — 執行處理所處理的並行要求數目上限。當並行要求數目超過此配額時，應用程式執行器會擴充服務。
- 大小上限 — 您的服務可擴充至的執行個體數目上限。這是可同時處理服務流量的最高執行個體數目。
- 最小大小 — App Runner 可為您的服務佈建的執行個體數目下限。服務一律至少有這個數目的佈建執行個體。其中一些執行個體會主動處理流量。其餘部分都是符合成本效益的運算容量儲備的一部分，可以快速啟動。您需要支付所有佈建執行個體的記憶體使用量費用。您只需支付使用中子集的 CPU 使用率。

> **Note**
>
> vCPU 資源計數會決定應用程式執行器可提供給服務的執行個體數目。這是常駐在服務中的 Fargate 隨選 vCPU 資源計數的可調整配額值。 AWS Fargate (Fargate) 若要檢視您帳戶的 vCPU 配額設定或要求提高配額，請使用中的 Service Quotas 主控台。 AWS Management Console如需詳細資訊，請參閱 Amazon 彈性容器AWS Fargate 服務開發人員指南中的服務配額。

管理服務的 auto 調整

使用下列其中一種方法管理 App Runner 服務的 auto 調整規模：

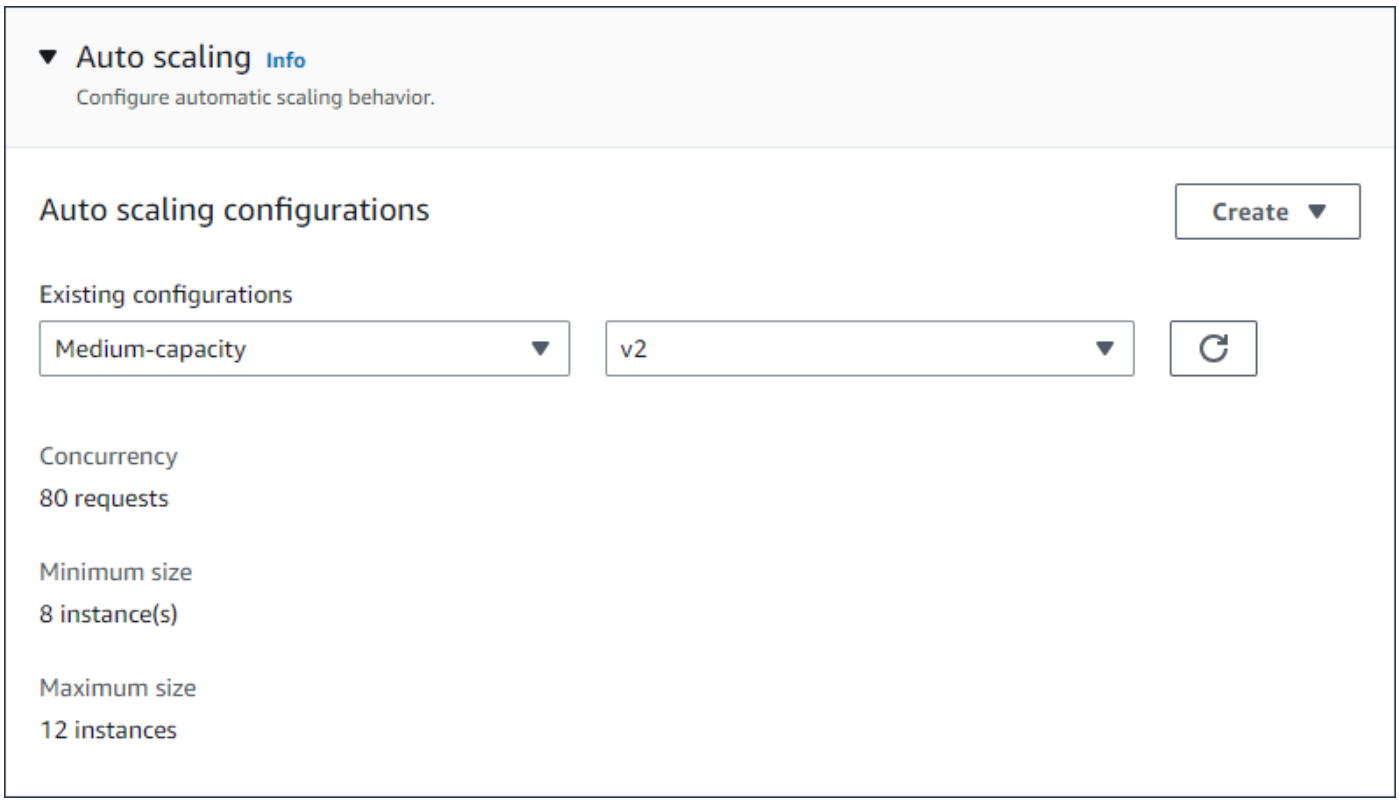
App Runner console

當您使用 App Runner 主控台建立服務或更新服務組態時，您可以指定 auto 擴展設定。

> **Note**
>
> 當您變更與服務相關聯的 auto 擴展設定或修訂版時，您的服務會重新部署。

auto Scaling 設定頁面提供數個選項，可為您的服務設定自動調整規模。

- 欲指派現有組態與版本修訂-從「現有組態」下拉式清單中選擇一個值。最新的修訂版本將預設在相鄰的下拉式清單中。如果存在您希望選取的其他版序，請從「版序」下拉式清單中執行此操作。修訂版本的組態值隨即顯示。
- 若要建立並指派新的 auto 調整配置，請從「建立」功能表中選取「建立新 ASC」。這將啟動「新增自訂自 auto 調整比例」設定頁面。輸入 auto 調整比例參數的模型組態名稱和值。然後選取 [新增]。App Runner 為您創建新的 auto 縮放配置資源，並返回到「自動縮放」部分，並選擇並顯示新配置。
- 欲建立並指派新版本修訂-首先從「現有組態」下拉式清單中選取組態名稱。然後從「建立」功能表選取「建立 ASC 版次」。這將啟動「新增自訂自 auto 調整比例」設定頁面。輸入 auto 調整比例參數的值。然後選取 [新增]。App Runner 為您創建一個新的 auto 縮放配置修訂版，並將您返回到自動縮放部分，並選擇並顯示新版本。



App Runner API or AWS CLI

當您呼叫CreateService或 UpdateServiceApp Runner API 動作時，您可以使用`AutoScalingConfigurationArn`參數為您的服務指定 auto 動調整規模設定資源。

下一節提供管理 auto 擴展設定資源的指引。

管理 auto 調整設定資源

使用下列其中一種方法管理您帳戶的 App Runner auto 擴展配置和修訂：

App Runner console

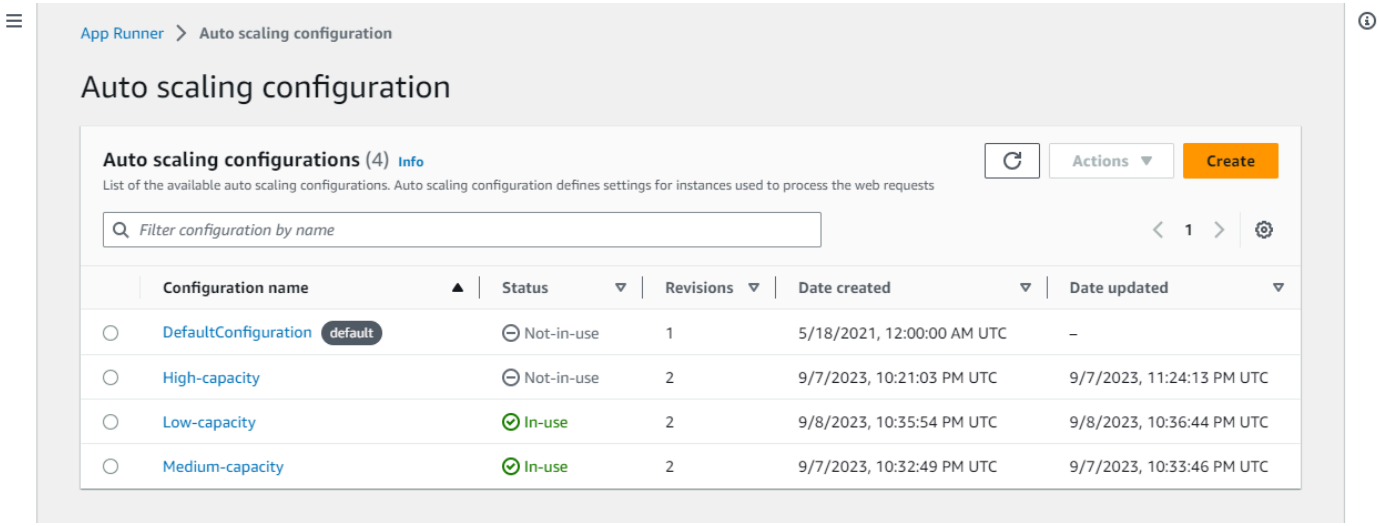
管理 auto 調整設定

auto 調整配置頁面列出了您在帳戶中設置的自動擴展配置。您可以在此頁面上創建和管理 auto 擴展配置，然後將它們分配給一個或多個 App Runner 服務。

您可以從此頁面執行下列任一項作業：

- 建立新的 auto 縮放設定。
- 為現有的 auto 調整規模組態建立新的修訂版本。

- 刪除 auto 調整比例組態。
- 將 auto 縮放組態設定為預設值。



管理帳戶中的 auto 調整設定

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在導覽窗格中，選擇 [自動縮放設定]。主控台會顯示您帳戶中的 auto 調整設定清單。

   您現在可以執行下列任一項作業。

   - 若要建立新的 auto 縮放設定，請依照下列步驟執行。

     a. 在自動縮放組態頁面上，選取建立。

        建立 auto 調整規模組態頁面隨即顯示。

     b. 輸入模型組態名稱、並行、最小大小及最大大小的值。

     c. (選擇性) 如果您要新增標籤，請選取 [自動新標籤]。然後在出現的字段上輸入「名稱」和「值」（可選）。

     d. 選取建立。

   - 若要為現有的 auto 調整規模組態建立新的修訂版本，請遵循下列步驟。

     a. 在自動調整規模組態頁面上，選取需要新修訂版本之組態旁邊的選項按鈕。然後從「動作」功能表選取「建立修訂」。

        [建立修訂] 頁面隨即顯示。

     b. 開啟，輸入「並行」、「最小大小」和「最大大小」的值。

c. (選擇性) 如果您要新增標籤，請選取 [自動新標籤]。然後在出現的字段上輸入「名稱」和「值」（可選）。

d. 選取建立。

- 若要刪除 auto 縮放組態，請遵循下列步驟。

a. 在 [自動調整比例配置] 頁面上，選取您需要刪除之組態旁邊的圓鈕。

b. 從「動作」功能表選取「刪除」。

c. 若要繼續刪除，請在確認對話方塊中選取 [刪除]。否則，請選取「取消」。

> **Note**
>
> App Runner 會驗證您的刪除選項未設定為預設值，或任何使用中的服務目前正在使用中。

- 若要將 auto 縮放組態設定為預設值，請依照下列步驟執行。

a. 在 [自動調整比例配置] 頁面上，選取您需要設定為預設值的組態旁邊的選項按鈕。

b. 從「動作」功能表中選取「設為預設值」。

c. 會顯示一個對話方塊，通知您 App Runner 將使用最新修訂版作為您建立的所有新服務的預設設定。選取 [確認] 以繼續。否則請選取「取消」

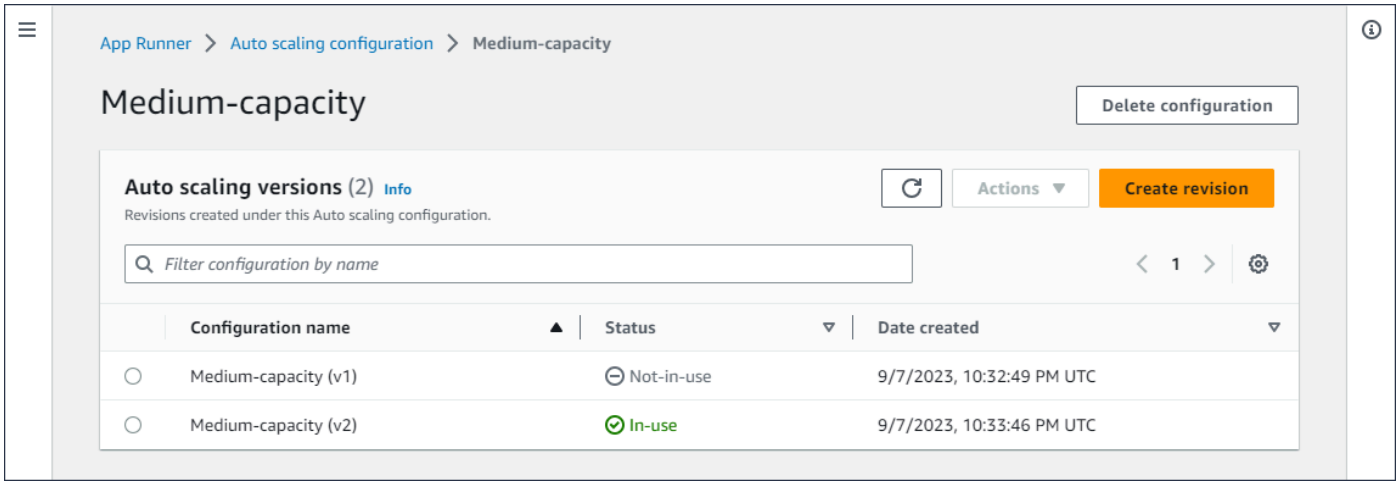
> **Note**
>
> - 當您將 auto 動擴展配置設定設定設定為預設值時，系統會自動將其指派為您 future 建立的新服務的預設組態。
> - 新的預設指定不會影響先前為現有服務設定的關聯。
> - 如果指定的默認 auto 縮放配置具有修訂版，則 App Runner 將其最新版本指定為默認版本。

管理修訂

控制台還有一個頁面，用於創建和管理稱為 auto Scaling 修訂版本的現有自動擴展修訂。在「自動調整資源調整規模組態」頁面上選取組態名稱，以存取此頁面。

您可以從「自動縮放修訂」頁面執行下列任一項作業：

- 建立新的 auto 縮放修訂版本。
- 將 auto 縮放組態修訂版本設定為預設值。
- 刪除修訂。
- 刪除整個 auto 縮放配置，包括所有關聯的修訂。
- 檢視修訂版本的組態詳細資訊。
- 檢視與修訂相關聯的服務清單。
- 變更所列服務的修訂版本。



管理帳戶中的 auto 縮放修訂

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在導覽窗格中，選擇 [自動縮放設定]。主控台會顯示您帳戶中的 auto 調整設定清單。本管理 auto 調整設定節中先前的一組程序包括此頁面的螢幕影像。
3. 現在，您可以深入研究特定的 auto 擴展配置，以查看和管理其所有修訂版本。在 auto 縮放模型組態窗格的模型組態名稱欄下，選擇一個自動調整比例模型組態名稱。選取實際名稱，而非選項按鈕。這會在「自動縮放修訂」頁面上導覽至該組態的所有修訂版本清單。
4. 您現在可以執行下列任一項作業。
   - 若要為現有的 auto 調整規模組態建立新的修訂版本，請遵循下列步驟。

     a. 在「自動縮放修訂」頁面上，選取「建立修訂版」。

        [建立修訂] 頁面隨即顯示。

     b. 輸入「並行」、「最小大小」和「最大大小」的值。

c. (選擇性) 如果您要新增標籤，請選取 [自動新標籤]。然後在出現的字段上輸入「名稱」和「值」（可選）。

d. 選取建立。

- 若要刪除整個 auto 縮放設定 (包括所有關聯的修訂)，請依照下列步驟執行。

a. 選取頁面右上角的 [刪除組態]。

b. 若要繼續刪除，請在確認對話方塊中選取 [刪除]。否則，請選取「取消」。

> **Note**
>
> App Runner 會驗證您的刪除選項未設定為預設值，或任何使用中的服務目前正在使用中。

- 若要將 auto 縮放修訂版本設定為預設值，請遵循下列步驟。

a. 選取您需要設定為預設修訂的修訂旁邊的選項按鈕。

b. 從「動作」功能表中選取「設為預設值」。

> **Note**
>
> - 當您將 auto 動擴展配置設定設定設定為預設值時，系統會自動將其指派為您 future 建立的新服務的預設組態。
> - 新的預設指定不會影響先前為現有服務設定的關聯。

- 若要檢視修訂版本的組態詳細資訊，請依照下列步驟執行。

  - 選取修訂旁邊的選項按鈕。

    修訂版的組態詳細資料 (包括 ARN) 會顯示在下方的分割面板中。請參閱此程序結束時的螢幕影像。

- 若要檢視與修訂相關聯的服務清單，請依照下列步驟執行。

  - 選取修訂旁邊的選項按鈕。

    「服務」面板會顯示在下方的分割面板中，位於修訂版組態詳細資料下方。面板會列出所有使用此 auto 調整規模組態修訂版的服務。請參閱此程序結束時的螢幕影像。

- 若要變更所列服務的修訂版本，請依照下列步驟執行。

a. 選取修訂旁邊的選項按鈕 (如果您尚未這麼做)。

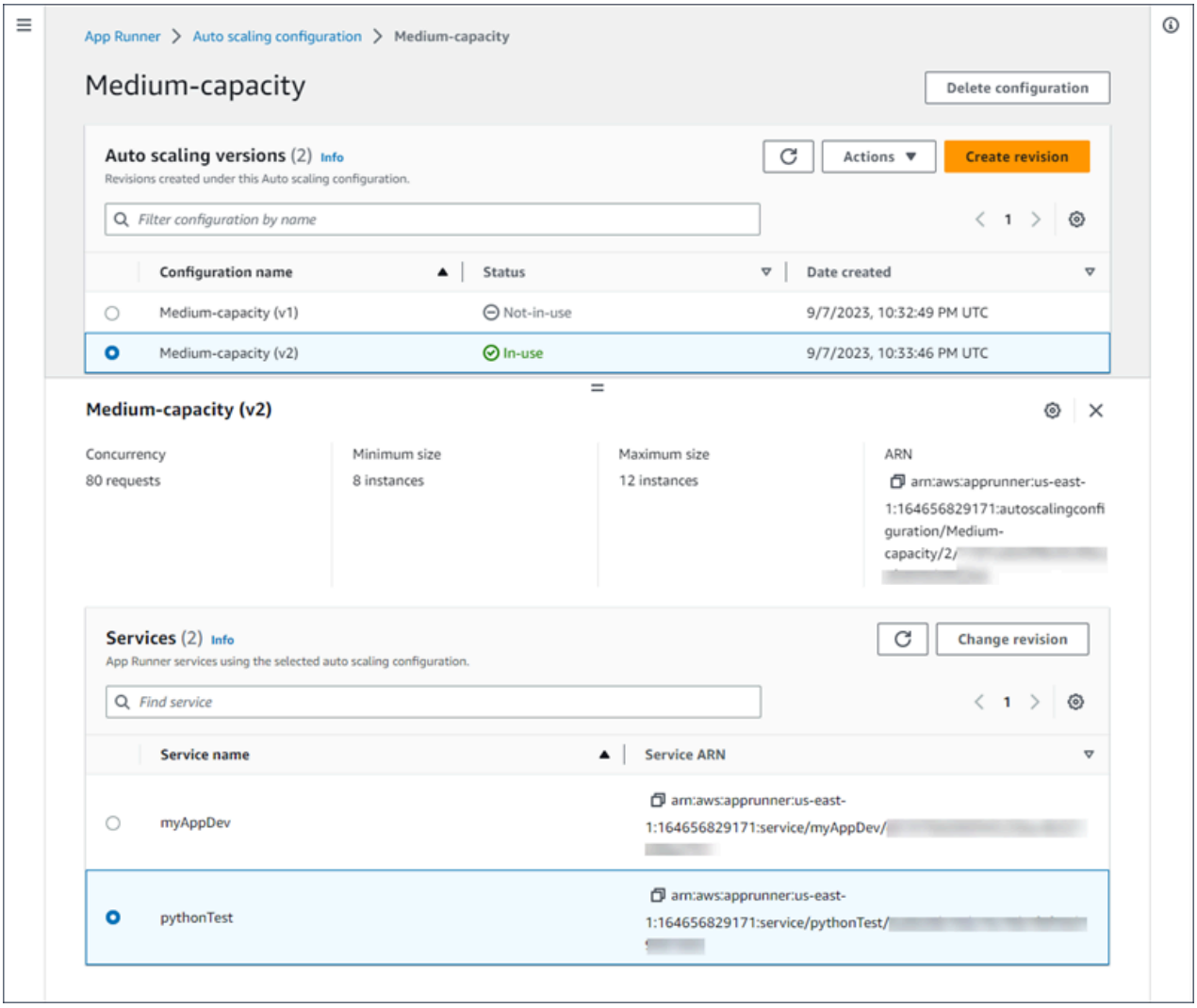
「服務」面板會顯示在下方的分割面板中，位於修訂版組態詳細資料下方。面板會列出所有使用此 auto 調整規模組態修訂版的服務。請參閱此程序結束時的螢幕影像。

b. 在「服務」面板上，選取您要修改之服務旁邊的圓鈕。然後選取 [變更修訂]。

c. 「變更 ASC 修訂」面板隨即顯示。從下拉式清單中的可用版本中選擇。只有您先前選擇的 auto 調整比例組態的修訂版本可用。如果您需要變更為不同的 auto 調整比例組態，請遵循前一節中的程序the section called “管理服務的 auto 調整”。

選取「更新」以繼續進行變更。否則請選取「取消」

> **Note**
>
> 當您變更與服務相關聯的修訂版本時，您的服務會重新部署。
> 您必須在此面板上選取「重新整理」，才能看到更新的關聯。
> 若要查看進行中的活動和服務重新部署的狀態，請使用面板導覽至「App Runner > 服務」，選取服務，然後從「服務概觀」面板檢視「記錄檔」索引標籤。



App Runner API or AWS CLI

使用下列應用程式執行器 API 動作來管理您的 auto 動擴展配置資源。

- CreateAutoScalingConfiguration-建立新的 auto 縮放組態或現有組態的版本修訂。
- UpdateDefaultAutoScalingConfiguration將 auto 縮放組態設定為預設值。現有的默認 auto 動縮放配置將自動設置為非默認值。
- ListAutoScalingConfigurations— 傳回與您相關聯的 auto 調整設定清單 AWS 帳戶，以及摘要資訊。
- ListServicesForAutoScalingConfiguration— 使用 auto 擴展配置返回關聯的應用程序運行器服務的列表。

- DescribeAutoScalingConfiguration— 傳回自 auto 調整配置的完整描述。
- DeleteAutoScalingConfiguration— 刪除 auto 調整比例組態。您可以刪除頂層 auto 縮放組態、某個組態的特定修訂版本，或與頂層組態相關聯的所有修訂版本。使用可選`DeleteAllRevisions`參數刪除所有修訂。如果您達到的 auto 擴展配置資源配額 AWS 帳戶，則可能需要刪除不必要的 auto 擴展配置。

管理應用程式執行器服務的自訂網域名稱

當您建立 AWS App Runner 服務時，應用程式執行器會為其分配網域名稱。這是網域中由應用程式執行者擁有的子`awsapprunner.com`網域。您可以使用網域名稱來存取在服務中執行的 Web 應用程式。

> **Note**
>
> 為了增強應用程式執行器應用程式的安全性，*.awsapprunner.com 網域註冊在公用尾碼清單 (PSL) 中。為了進一步的安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用`__Host-`前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

如果您擁有域名，則可以將其關聯到您的 App Runner 服務。App Runner 驗證您的新網域之後，除了 App Runner 網域之外，您還可以使用網域存取應用程式。您最多可以關聯五個自訂網域。

> **Note**
>
> 您可以選擇性地包含`www`網域的子網域。不過，目前只有 API 才支援此功能。應用程序運行器控制台不支持包括`www`域的子域。

> **Note**
>
> AWS App Runner 不支持使用 Route 53 私有託管區域。私有託管區域可自訂 Amazon VPC 流量的網域名稱解析。如需私有託管區域的詳細資訊，請參閱 Route 53 說明文件中的使用私有託管區域。

將自訂網域關聯 (連結) 至您的服務

將自訂網域與服務相關聯時，必須將 CNAME 記錄和 DNS 目標記錄新增至 DNS 伺服器。以下各節提供 CNAME 記錄和 DNS 目標記錄以及如何使用它們的相關資訊。

> **Note**
>
> 如果您使用 Amazon Route 53 做為 DNS 供應商，App Runner 會自動使用必要的憑證驗證和 DNS 記錄來設定您的自訂網域，以連結至應用程式執行器 Web 應用程式。當您使用 App Runner 主控台將自訂網域連結至服務時，就會發生這種情況。以下管理自訂網域主題提供了更多資訊。

CNAME 記錄

當您將自訂網域與服務建立關聯時，App Runner 會為您提供一組憑證驗證記錄以進行憑證驗證。您必須將這些憑證驗證記錄新增至您的網域名稱系統 (DNS) 伺服器。將 App Runner 提供的憑證驗證記錄新增至您的 DNS 伺服器。這樣，應用程序運行器可以驗證您擁有或控制域。

> **Note**
>
> 若要自動續約您的自訂網域憑證，請確定不要從 DNS 伺服器刪除憑證驗證記錄。如需如何解決與憑證續訂相關問題的資訊，請參閱the section called “自訂網域憑證續約”。

應用程序運行器使用 ACM 來驗證域。如果您在 DNS 記錄中使用 CAA 記錄，請確定至少有一個 CAA 記錄參照`amazon.com`。否則，ACM 無法驗證網域並成功建立您的網域。

如果您收到與 CAA 相關的錯誤，請參閱下列連結以瞭解如何解決這些錯誤：

- 憑證授權單位授權 (CAA) 問題
- 如何解決發行或更新 ACM 憑證時的 CAA 錯誤？
- 自訂網域名稱

> **Note**
>
> 如果您使用 Amazon Route 53 做為 DNS 供應商，App Runner 會自動使用必要的憑證驗證和 DNS 記錄來設定您的自訂網域，以連結至應用程式執行器 Web 應用程式。當您使用 App

Runner 主控台將自訂網域連結至服務時，就會發生這種情況。以下管理自訂網域主題提供了更多資訊。

DNS 目標記錄

將 DNS 目標記錄添加到 DNS 服務器以定位應用程序運行器域。如果您選擇此選項，請為自訂網域新增一筆記錄，為www子網域新增另一筆記錄。然後，在 App Runner 控制台中等待自定義域狀態變為活動狀態。這通常需要幾分鐘的時間，但最多可能需要 24—48 小時 (1—2 天)。驗證您的自訂網域後，App Runner 會開始將流量從此網域路由到您的 Web 應用程式。

> **Note**
>
> 為了更好地與應用程序運行器服務的兼容性，我們建議您使用 Amazon 路線 53 作為您的 DNS 提供商。如果您不使用 Amazon Route 53 來管理公用 DNS 記錄，請聯絡您的 DNS 供應商，以瞭解如何新增記錄。
>
> 如果您使用 Amazon Route 53 做為 DNS 提供者，您可以為子網域新增 CNAME 或別名記錄。針對根網域，請確定您使用別名記錄。

您可以從 Amazon 路線 53 或其他供應商購買網域名稱。若要透過 Amazon Route 53 購買網域名稱，請參閱 Amazon Route 53 開發人員指南中的註冊新網域。

如需有關如何在 Route 53 中設定 DNS 目標的指示，請參閱 Amazon Route 53 開發人員指南中的將流量路由到您的資源。

如需有關如何在其他註冊商 (例如 GoDaddy Shopify、Hover 等) 上設定 DNS 目標的指示，請參閱他們有關新增 DNS Target 記錄的特定文件。

指定要與您的應用程式執行器服務相關聯的網域

您可以透過下列方式指定要與 App Runner 服務產生關聯的網域：

- 根網域 — DNS 有一些固有的限制，可能會阻止您為根網域名稱建立 CNAME 記錄。例如，如果您的網域名稱是`example.com`，您可以建立 CNAME 記錄，將流量路由`acme.example.com`到您的 App Runner 服務。不過，您無法建立 CNAME 記錄，將流量路由`example.com`至您的應用程式執行器服務。若要建立根網域，請確定您已新增別名記錄。

  別名記錄是 Route 53 特有的，並且與 CNAME 記錄相比具有以下優點：

  - Route 53 可為根網域或子網域建立別名記錄，提供您更大的彈性。例如，如果您的網域名稱是example.com，您可以建立將要求路由傳送example.comacme.example.com至 App Runner 服務的記錄。
  - 這是更具成本效益。這是因為 Route 53 不會針對使用別名記錄路由傳送流量的要求收取費用。
- 子網域 — 例如，login.example.com或admin.login.example.com。您也可以選擇將www子網域關聯為相同作業的一部分。您可以為子網域新增 CNAME 或別名記錄。
- 萬用字元 — 例如，*.example.com. 在這種情況下，您無法使用該www選項。您只能將萬用字元指定為根網域的直接子網域，而且只能單獨指定萬用字元。這些不是有效的規格：login*.example.com，*.login.example.com。此萬用字元規格會關聯所有直接子網域，且不會建立根網域本身的關聯性。根網域必須在個別的作業中關聯。

更具體的網域關聯會覆寫較不特定的網域關聯。例如，login.example.com覆寫*.example.com。會使用更具體關聯的憑證和 CNAME。

下列範例顯示如何使用多個自訂網域關聯：

1. example.com與服務的首頁相關聯。啟用www要關聯的www.example.com。
2. login.example.com與服務的登錄頁面相關聯。
3. *.example.com與自訂「找不到」頁面相關聯。

取消關聯 (取消連結) 自訂網域

您可以取消關聯（取消鏈接）自定義域與您的應用程序運行器服務。當您取消連結網域時，App Runner 會停止將流量從此網域路由傳送至您的 Web 應用程式。

> **Note**
>
> 您必須刪除與 DNS 伺服器取消關聯的網域記錄。

應用程序運行器內部創建跟踪域有效性的證書。這些憑證會儲存在 AWS Certificate Manager (ACM) 中。App Runner 在域與您的服務斷開關聯後或刪除服務後的 7 天內不會刪除這些證書。

管理自訂網域

使用下列其中一種方法管理 App Runner 服務的自訂網域：

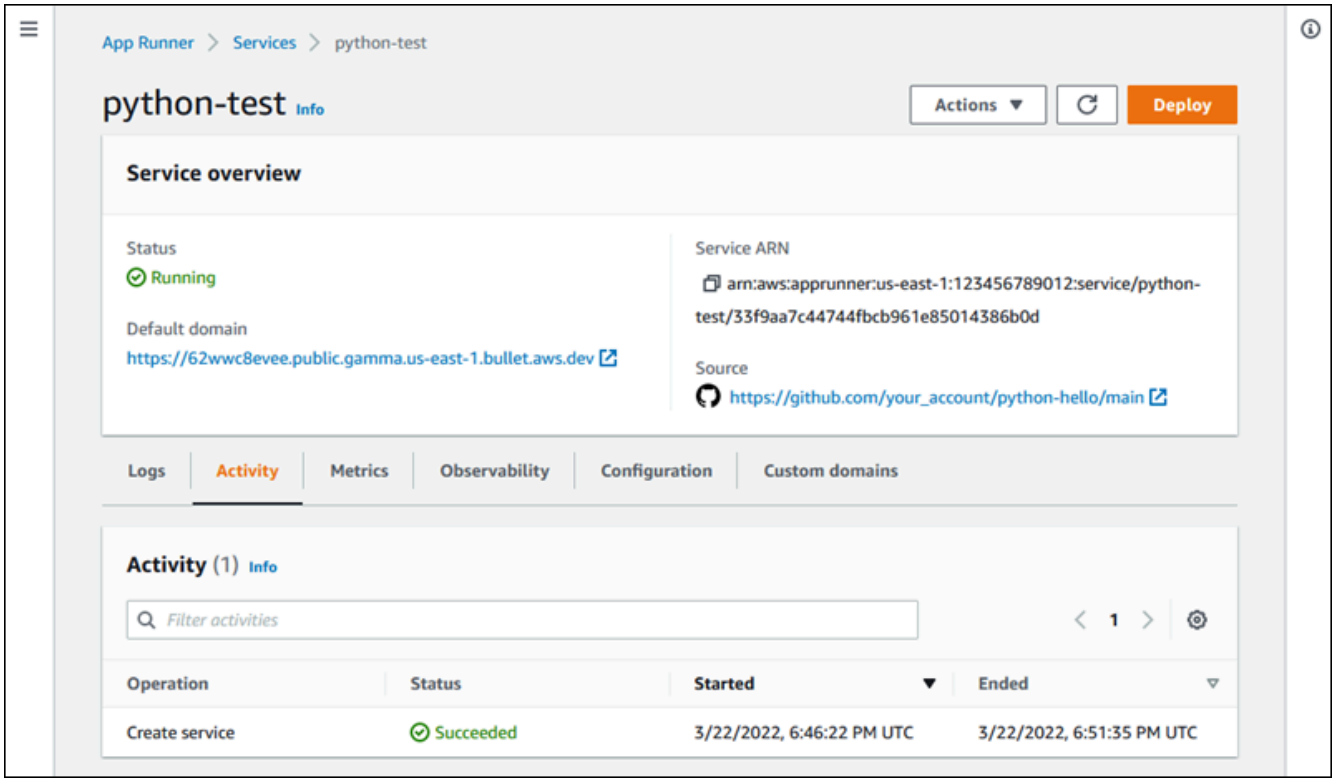
> **Note**
>
> 為了更好地與應用程序運行器服務的兼容性，我們建議您使用 Amazon 路線 53 作為您的 DNS 提供商。如果您不使用 Amazon Route 53 來管理公用 DNS 記錄，請聯絡您的 DNS 供應商，以瞭解如何新增記錄。
> 如果您使用 Amazon Route 53 做為 DNS 提供者，您可以為子網域新增 CNAME 或別名記錄。針對根網域，請確定您使用別名記錄。

App Runner console

使用應用程式執行器主控台建立關聯 (連結) 自訂網域

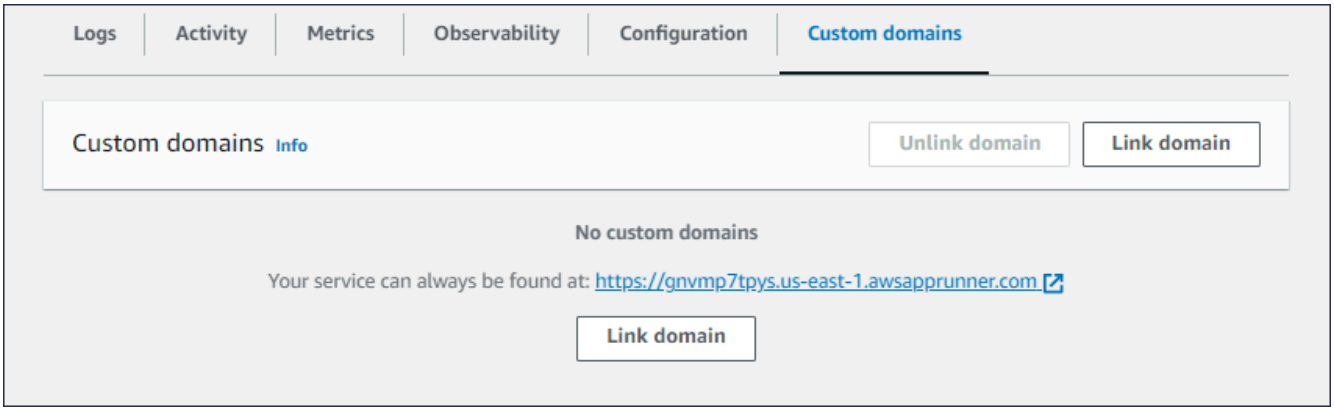
1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

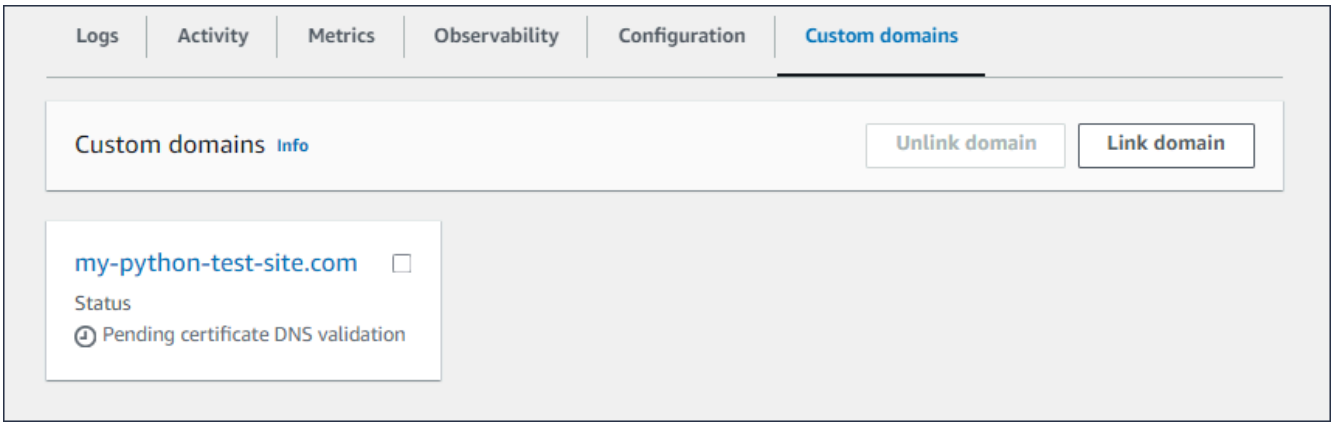
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇 [自訂網域] 索引標籤。

   主控台會顯示與您的服務相關聯的自訂網域，或 [無自訂網域]。

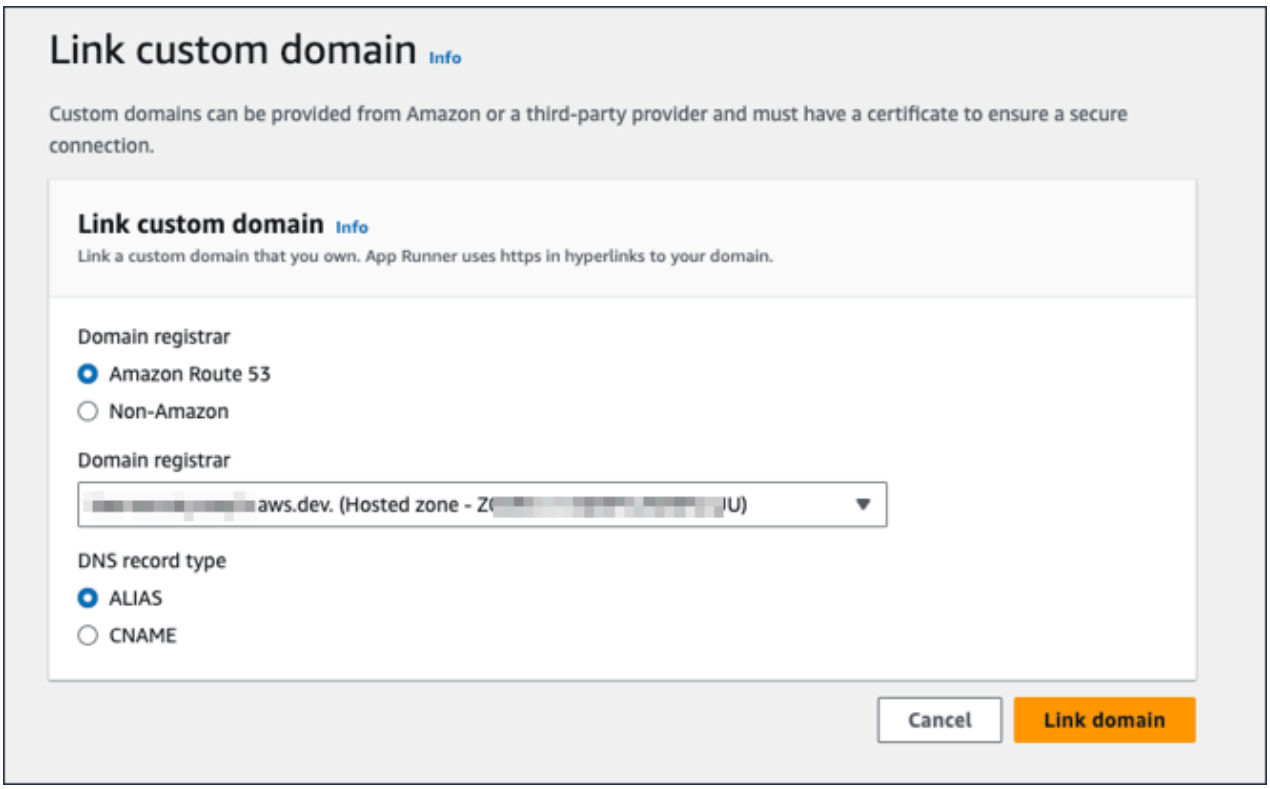




4. 在「自訂網域」標籤上，選擇「連結網域」。
5. 連結自訂網域頁面隨即顯示。
   - 如果您的自訂網域是透過 Amazon Route 53 註冊的，請為網域註冊商選取 Amazon Route 53。
     a. 從下拉式清單中選取「網域名稱」。此清單會顯示您的 Route 53 網域名稱和託管區域 ID 的名稱。

     > **Note**
     > 您必須先從管理其他應用程式執行程式資源的相同 AWS 帳戶，使用 Amazon Route 53 服務建立 Route 53 網域。

     b. 選取 DNS 記錄類型。
     c. 選擇「連結網域」。

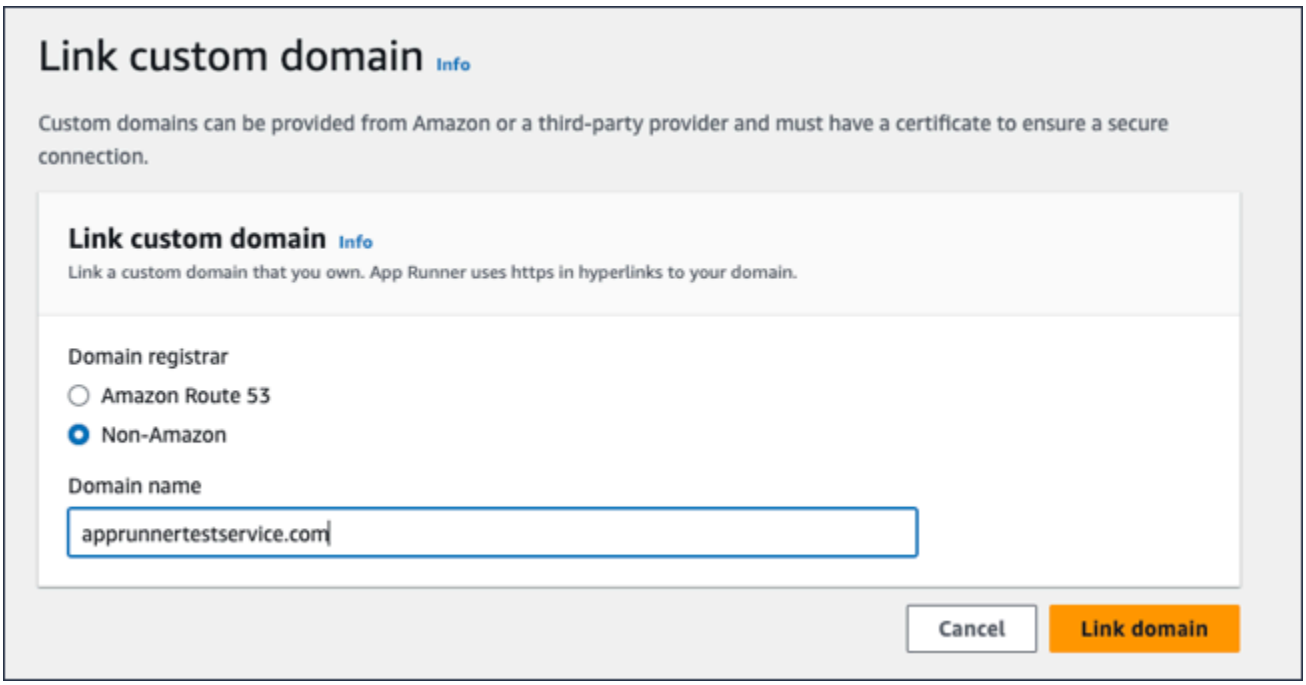


> **Note**
>
> 如果 App Runner 顯示錯誤訊息，指出自動設定嘗試失敗，您可以手動設定 DNS 記錄來繼續。如果先前已取消相同網域名稱與服務的連結，而沒有指向之後刪除服務的 DNS 提供者記錄，就會發生此問題。在這種情況下，應用程序運行器被阻止自動覆蓋這些記錄。若要完成 DNS 組態，請略過此程序中的其餘步驟，然後遵循中的指示設定 Amazon 路線 53 別名記錄。

- 如果您的自訂網域是向其他網域註冊商註冊，請為網域註冊商選取非 Amazon。

  a. 輸入網域名稱。

  b. 選擇「連結網域」。



6. [設定 DNS] 頁面隨即顯示。

   - 如果 Amazon Route 53 是您的 DNS 提供商，則此步驟是可選的。

     此時應用程式執行程式已自動設定您的 Route 53 網域，並使用必要的憑證驗證和 DNS 記錄。

     > **Note**
     >
     > 如果先前已取消與服務的連結相同的網域名稱，但沒有指向之後刪除服務的 DNS 提供者記錄，則 App Runner 嘗試的自動設定可能會失敗。若要解決此問題並完成 DNS 關聯，請繼續執行 [設定 DNS] 頁面上的步驟 (1) 和 (2)，將目前的目標和憑證記錄複製到 DNS 提供者。

   - 複製憑證驗證記錄和 DNS 目標記錄，並將其新增至您的 DNS 伺服器。然後，應用程序運行器可以驗證您擁有或控制該域。

     > **Note**
     >
     > 若要自動續約您的自訂網域憑證，請確定不要從 DNS 伺服器刪除憑證驗證記錄。

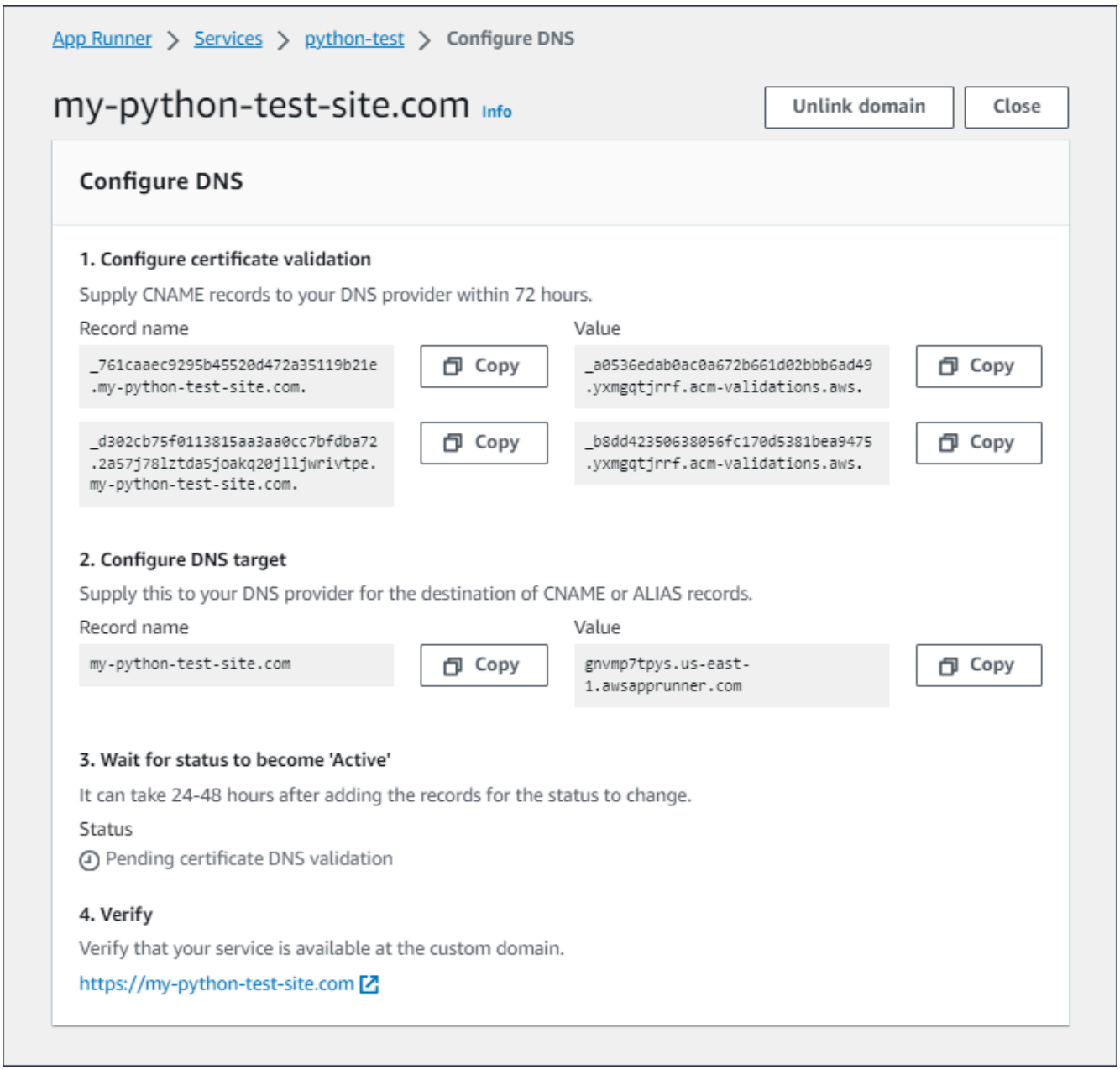
- 如需有關設定憑證驗證的詳細資訊，請參閱AWS Certificate Manager 使用指南中的 DNS 驗證。
- 如需如何使用 Amazon Route 53 別名記錄設定 DNS 目標的相關資訊，請參閱the section called “設定 Amazon 路線 53 別名記錄”。

- 如果您使用的是 Amazon 路線 53 以外的 DNS 提供商，請按照以下步驟操作。

  - 複製憑證驗證記錄和 DNS 目標記錄，並將其新增至您的 DNS 伺服器。然後，應用程序運行器可以驗證您擁有或控制該域。

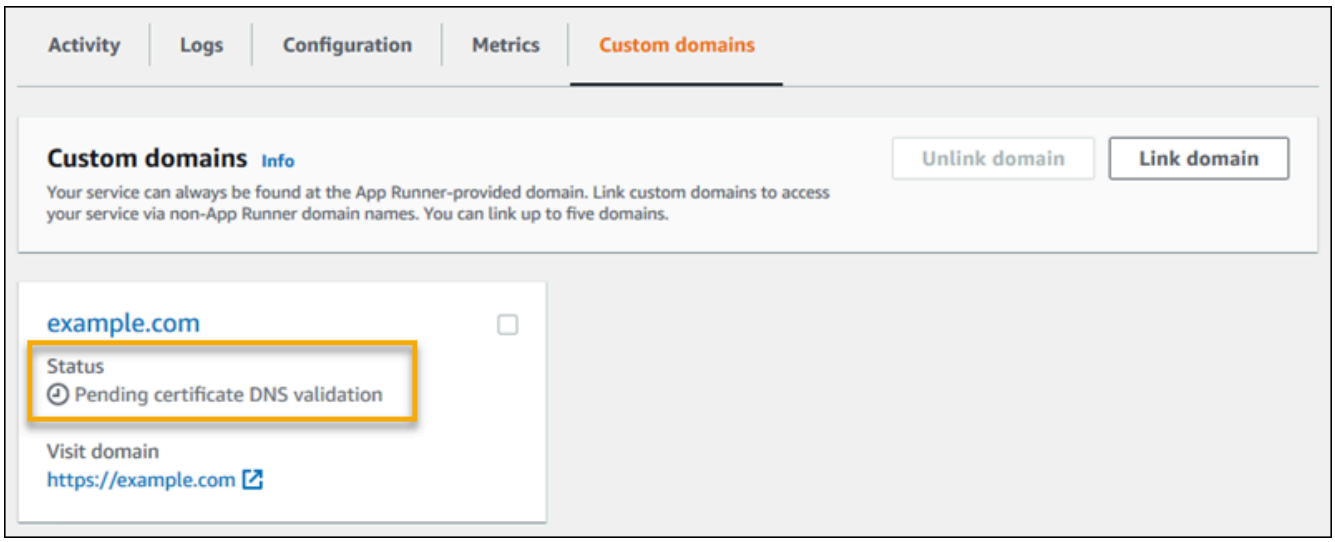
    > **Note**
    >
    > 若要自動續約您的自訂網域憑證，請確定不要從 DNS 伺服器刪除憑證驗證記錄。

    - 如需有關設定憑證驗證的詳細資訊，請參閱AWS Certificate Manager 使用指南中的 DNS 驗證。
    - 如需有關如何在其他註冊商 (例如 GoDaddy Shopify、Hover 等) 上設定 DNS 目標的指示，請參閱他們有關新增 DNS 目標的特定文件。

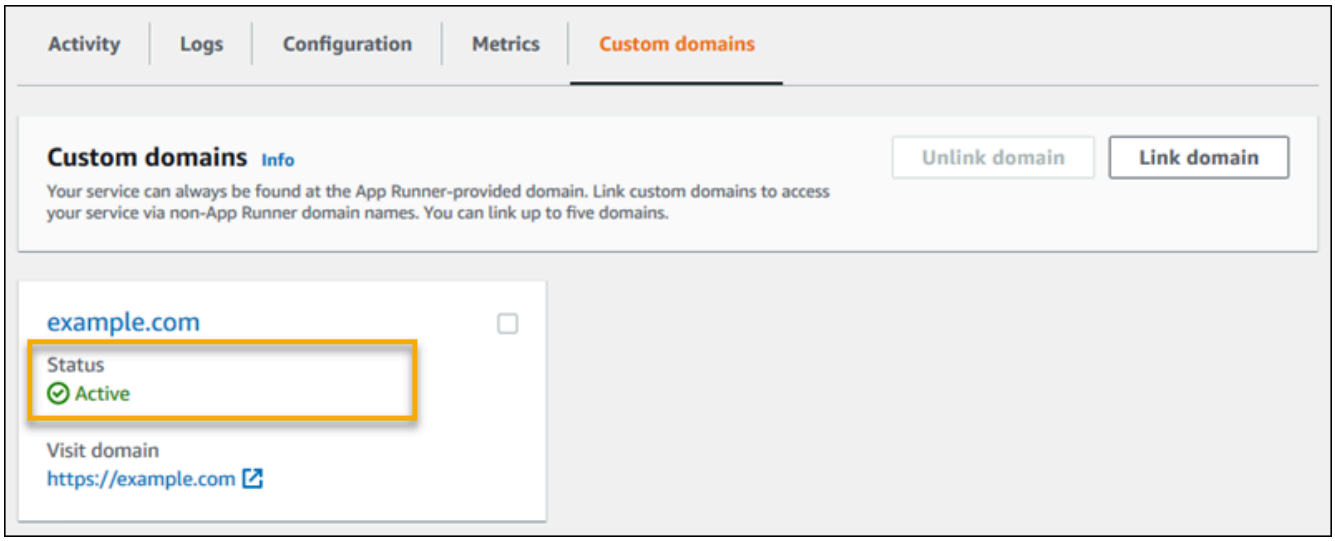


7. 選擇關閉

控制台再次顯示儀表板。[自訂網域] 索引標籤有新的方塊，顯示您剛才在擱置憑證 DNS 驗證狀態中連結的網域。



8. 當網域狀態變更為「作用中」時，請瀏覽至網域來驗證網域是否適用於路由流量。



> **Note**
>
> 如需有關如何疑難排解自訂網域相關錯誤的指示，請參閱the section called “自訂網域名稱”。

使用 App Runner 主控台取消關聯 (取消連結) 自訂網域

1. 在 [自訂網域] 索引標籤上，選取您要取消關聯的網域磚，然後選擇 [取消連結網域]。
2. 在「取消連結網域」對話方塊中，選擇「取消連結網域」來確認動作。

> **Note**
>
> 您必須刪除與 DNS 伺服器取消關聯的網域記錄。

App Runner API or AWS CLI

若要使用應用程式執行器 API 將自訂網域與您的服務建立關聯 AWS CLI，或呼叫 AssociateCustomDomainAPI 動作。呼叫成功時，會傳回描述與您的服務相關聯之自訂網域的CustomDomain物件。該對象顯示某個CREATING狀態，并包含對CertificateValidationRecord象列表。呼叫也會傳回可用來設定 DNS 目標的目標別名。這些是您可以新增至 DNS 的記錄。

若要使用應用程式執行器 API 取消自訂網域與服務的關聯 AWS CLI，或呼叫 DisassociateCustomDomainAPI 動作。當呼叫成功時，會傳回描述與服務中斷關聯的自訂網域的CustomDomain物件。該對象顯示某個DELETING狀態。

主題

- 為您的目標 DNS 設定 Amazon 路線 53 別名記錄

為您的目標 DNS 設定 Amazon 路線 53 別名記錄

> **Note**
>
> 如果 Amazon 路線 53 是您的 DNS 提供商，則不需要遵循此過程。在這種情況下，應用程序運行器使用必要的證書驗證和 DNS 記錄自動配置您的 Route 53 域，以鏈接到應用程序運行器 Web 應用程序。
> 如果 App Runner 的自動配置嘗試失敗，請按照此過程完成 DNS 配置。如果先前已取消相同網域名稱與服務的連結，但沒有指向之後刪除服務的 DNS 提供者記錄，則會封鎖 App Runner 無法自動覆寫這些記錄。此程序說明如何將它們手動複製到路由 53 DNS。

您可以使用 Amazon Route 53 作為 DNS 提供商，將流量路由到您的應用程序運行器服務。這是一個高可用性和可擴展的域名系統（DNS）Web 服務。Amazon Route 53 記錄包含控制流量路由到應用程式執行器服務的方式的設定。您可以建立 CNAME 記錄或別名記錄。如需 CNAME 和別名記錄的比較，請參閱 Amazon Route 53 開發人員指南中的別名和非別名記錄之間的選擇。

> **Note**
>
> Amazon 路線 53 目前支援在 2022 年 8 月 1 日之後建立的服務的別名記錄。

Amazon Route 53 console

設定 Amazon 路線 53 別名記錄

1. 登入 AWS Management Console 並開啟 Route 53 主控台。
2. 在導覽窗格中，選擇 Hosted zones (託管區域)。
3. 選擇您要用來將流量路由到 App Runner 服務的託管區域的名稱。
4. 選擇建立記錄。
5. 指定下列值：
   - 路由策略：選擇適用的路由策略。如需詳細資訊，請參閱選擇路由原則。
   - 記錄名稱：輸入您要用來將流量路由到 App Runner 服務的網域名稱。預設值為託管區域名稱。例如，如果託管區域的名稱是，`example.com`而您想要用`acme.example.com`來將流量路由到您的環境，請輸入`acme`。
   - 流量值/路由傳送至：選擇 App Runner 應用程式的別名，然後選擇端點所在的區域。選擇您要將流量路由到的應用程式的網域名稱。
   - 記錄類型：接受預設值，A — IPv4 位址。
   - 評估目標健全狀況：接受預設值 「是」。
6. 選擇建立記錄。

您建立的 Route 53 別名記錄會在 60 秒內傳播到所有 Route 53 伺服器上。當 Route 53 伺服器與您的別名記錄一起傳播時，您可以使用您建立的別名記錄名稱，將流量路由傳送至 App Runner 服務。

如需如何疑難排解 DNS 變更花費太長時間才能傳播的詳細資訊，請參閱為什麼 DNS 變更在 Route 53 和公用解析器中傳播需要這麼長時間？ 。

Amazon Route 53 API or AWS CLI

若要使用 Amazon 路線 53 API 設定 Amazon 路線 53 別名記錄，或 AWS CLI 呼叫 ChangeResourceRecordSetsAPI 動作。若要瞭解 Route 53 的目標託管區域 ID，請參閱服務端點。

暫停和恢復應用程序運行器服務

如果您需要暫時停用 Web 應用程式並停止執行程式碼，您可以暫停 AWS App Runner 服務。App Runner 會將服務的運算容量縮減為零。

當您準備好再次執行應用程式時，您可以繼續執行 App Runner 服務。App Runner 會佈建新的運算容量、將您的應用程式部署到該容量，並執行應用程式。您的應用程式來源不會重新部署，也不需要建置。相反，應用程序運行器恢復您當前部署的版本。您的應用程序保留其應用程序運行器域

Important

- 當您暫停服務時，您的應用程式會遺失其狀態。例如，您的代碼使用的任何臨時存儲都將丟失。對於程式碼而言，暫停和繼續服務等同於部署至新服務。
- 如果您因程式碼中的瑕疵而暫停服務 (例如發現的錯誤或安全性問題)，則無法在繼續服務之前部署新版本。

  因此，我們建議您繼續執行服務，並回復到上一個穩定的應用程式版本。
- 當您繼續服務時，App Runner 會部署您暫停服務之前使用的最後一個應用程式版本。如果您在暫停服務後新增了任何新的來源版本，即使選取了自動部署，App Runner 也不會自動部署它們。例如，假設您在映像存儲庫中有新的映像版本，或者在代碼存儲庫中有新的提交。這些版本不會自動部署。

  若要部署較新的版本，請在恢復 App Runner 服務之後，執行手動部署或將其他版本新增至來源儲存庫。

暫停和刪除比較

暫停您的應用程序運行器服務以暫時禁用它。只會終止運算資源，而且您儲存的資料 (例如，包含應用程式版本的容器映像檔) 會保持不變。恢復服務的速度非常快，您的應用程式已準備好部署到新的計算資源。您的應用程序運行器域保持不變。

刪除您的應用程序運行器服務以將其永久刪除。您儲存的資料將被刪除。如果您需要重新創建服務，則 App Runner 需要再次獲取源代碼，並在代碼存儲庫時構建它。您的 Web 應用程式 取得一個新的 App Runner 網域。

當您的服務暫停時

當您暫停服務且處於「已暫停」狀態時，服務會以不同方式回應動作要求，包括 API 呼叫或主控台作業。暫停服務時，您仍然可以執行 App Runner 動作，這些動作不會以影響其執行階段的方式修改服務的定義或設定。換句話說，如果動作變更執行中服務的行為、縮放比例或其他特性，您就無法在暫停的服務上執行該動作。

下列清單提供您可以和無法在暫停服務上執行的 API 動作的相關資訊。同樣允許或拒絕等效的主控台作業。

您可以在暫停服務上執行的動作

- *List**和*Describe**動作 — 只讀取資訊的動作。
- *DeleteService*— 您可以隨時刪除服務。
- *TagResource*、*UntagResource* — 標籤與服務相關聯，但不屬於其定義的一部分，也不會影響其執行階段行為。

您無法在暫停服務上執行的動作

- *StartDeployment*動作 (或使用主控台的手動部署)
- *UpdateService* (或使用控制台進行配置更改，標記更改除外)
- *CreateCustomDomainAssociations*, *DeleteCustomDomainAssociations*
- *CreateConnection*, *DeleteConnection*

暫停和恢復您的服務

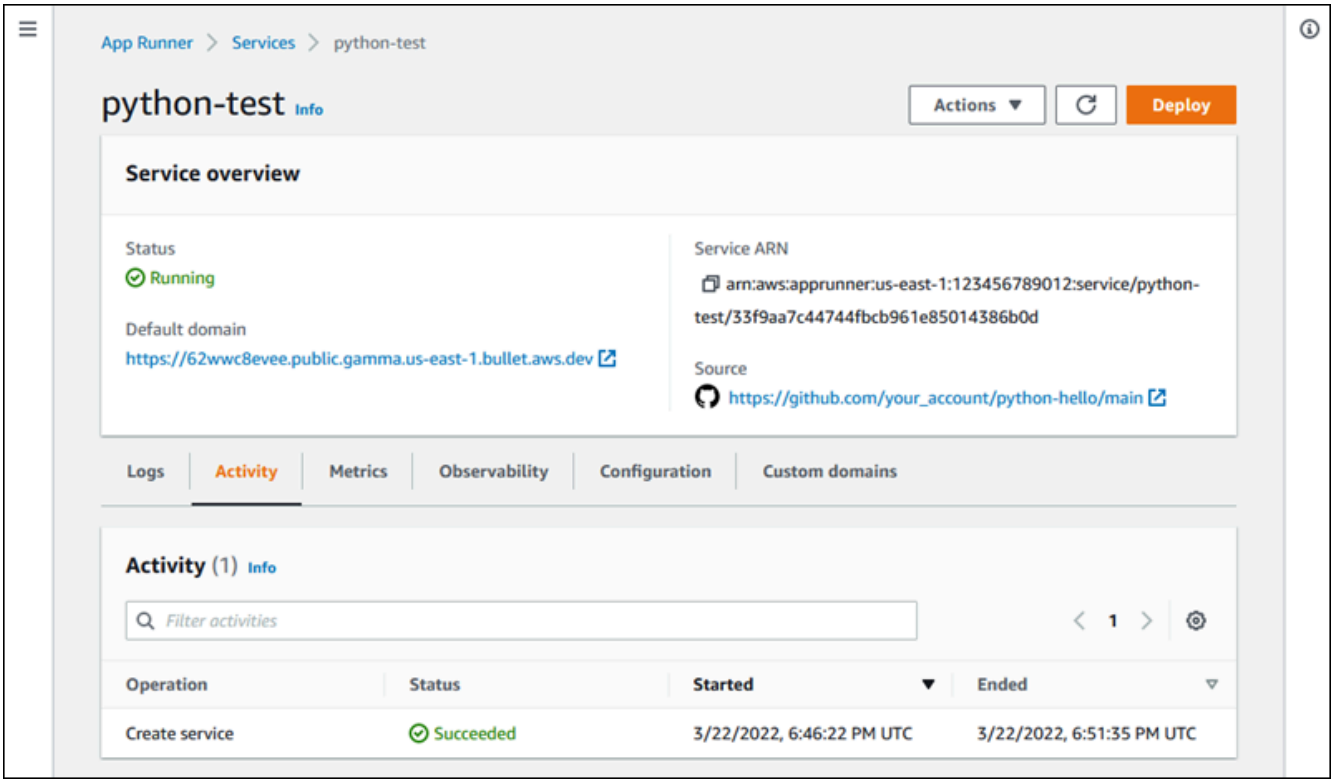
使用下列其中一種方法暫停並繼續您的應用程式執行器服務：

App Runner console

使用應用程式執行器主控台暫停服務

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板，其中包含服務概觀。



3. 選擇 [動作]，然後選擇 [暫停]。

   在服務儀表板頁面上，服務狀態會變更為 [作業進行中]，然後變更為 [暫停]。您的服務現已暫停。

使用應用程式執行器主控台恢復服務

1. 選擇 [動作]，然後選擇 [繼續]。

   在服務儀表板頁面上，服務狀態會變更為 [作業進行中]。

2. 等待服務恢復。在服務儀表板頁面上，服務狀態會變回執行中。
3. 若要確認繼續服務是否成功，請在服務儀表板頁面上選擇 App Runner 網域值。這是您服務網站的 URL。確認您的 Web 應用程式是否正常執行。

App Runner API or AWS CLI

若要使用應用程式執行器 API 暫停服務 AWS CLI，或呼叫 PauseServiceAPI 動作。如果調用返回顯示服務對象的成功響應`"Status": "OPERATION_IN_PROGRESS"`，則應用程序運行器開始暫停您的服務。

若要使用應用程式執行器 API 繼續服務 AWS CLI，或呼叫 ResumeServiceAPI 動作。如果調用返回顯示服務對象的成功響應`"Status": "OPERATION_IN_PROGRESS"`，則應用程序運行器開始恢復您的服務。

刪除應用程式執行器服務

當您想要終止在 AWS App Runner 服務中執行的 Web 應用程式時，您可以刪除該服務。刪除服務會停止執行中的 Web 服務、移除基礎資源，並刪除關聯的資料。

您可能會因為下列一或多個原因而想要刪除應用程式執行器服務：

- 您不再需要 Web 應用程序-例如，它已淘汰，或者它是您已完成使用的開發版本。
- 您已達到 App Runner 服務配額 — 您想要在其中建立新服務， AWS 區域 而且您已達到與您帳戶相關聯的配額。如需詳細資訊，請參閱 the section called “應用運行器資源配額”。
- 安全性或隱私權考量 — 您希望 App Runner 刪除其為您的服務儲存的資料。

暫停和刪除比較

暫停您的應用程序運行器服務以暫時禁用它。只會終止運算資源，而且您儲存的資料 (例如，包含應用程式版本的容器映像檔) 會保持不變。恢復服務的速度非常快，您的應用程式已準備好部署到新的計算資源。您的應用程序運行器域保持不變。

刪除您的應用程序運行器服務以將其永久刪除。您儲存的資料將被刪除。如果您需要重新創建服務，則 App Runner 需要再次獲取源代碼，並在代碼存儲庫時構建它。您的 Web 應用程式 取得一個新的 App Runner 網域。

應用程序亞軍刪除什麼？

當您刪除服務時，App Runner 會刪除一些關聯的項目，而不會刪除其他項目。下列清單提供詳細資訊。

應用程序運行器刪除的項目：

- 容器映像檔 — 您部署的映像檔副本，或是應用程式執行器從原始程式碼建置的映像檔。它存儲在 Amazon Elastic Container Registry (Amazon ECR) 使用由應用程序運行器擁有的內部 AWS 帳戶 。
- 服務配置 — 與您的應用程序運行器服務相關聯的配置設置。它們會使用應用程式執行器擁有的內部 AWS 帳戶 儲存在 Amazon DynamoDB 中。

應用程序運行器不刪除的項目：

- 連線 — 您可能有與您的服務相關聯的連線。應用程式執行器連線是個別的資源，可能會在多個應用程式執行器服務之間共用。如果您不再需要連接，則可以明確刪除它。如需詳細資訊，請參閱 the section called “連線”。
- 自訂網域憑證 — 如果您將自訂網域連結至 App Runner 服務，App Runner 會在內部建立追蹤網域有效性的憑證。它們會儲存在 AWS Certificate Manager (ACM) 中。App Runner 在域與您的服務中斷鏈接後或服務被刪除後的七天內不會刪除證書。如需詳細資訊，請參閱 the section called “自訂網域名稱”。

刪除您的服務

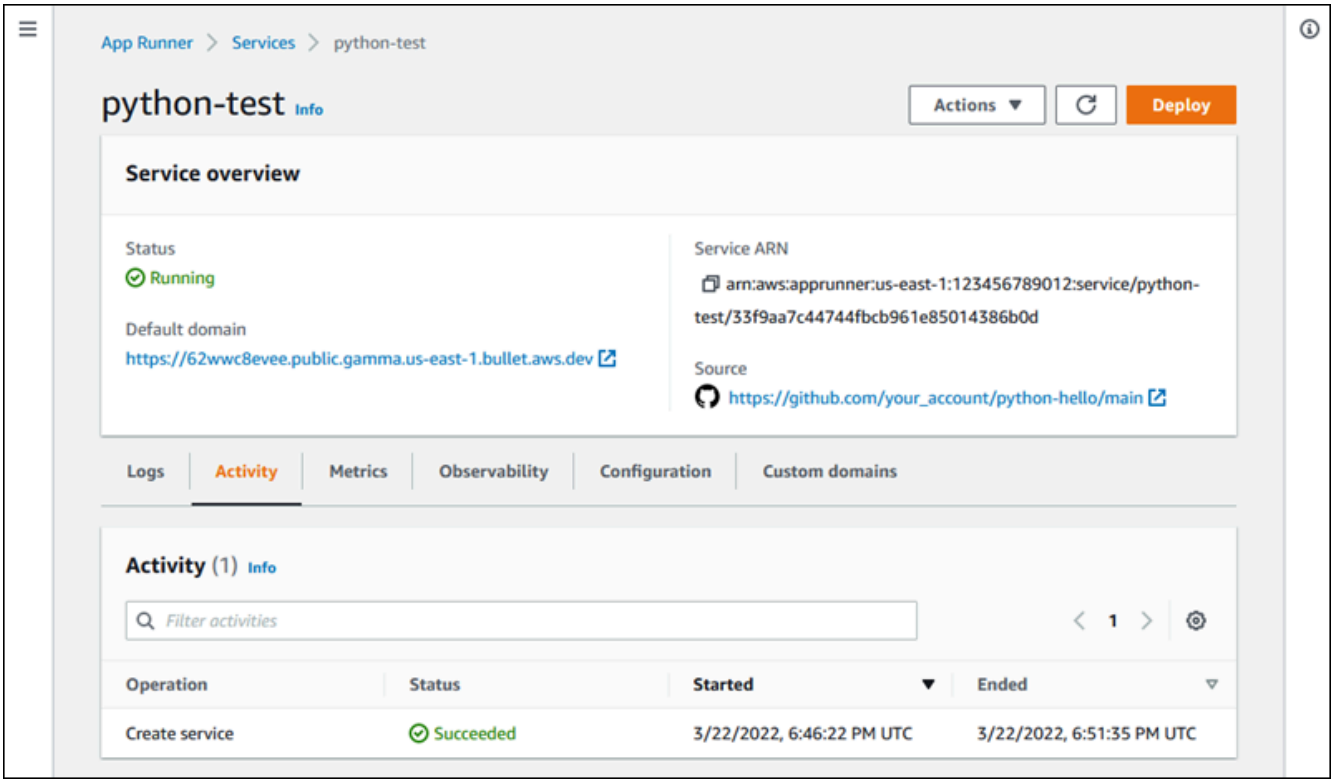
使用下列其中一種方法刪除您的應用程式執行器服務：

App Runner console

使用應用程式執行器主控台刪除您的服務

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板，其中包含服務概觀。



3. 選擇動作，然後選擇刪除。

   主控台會帶您前往 [服務] 頁面。刪除的服務會顯示作業中狀態，然後服務就會從清單中消失。您的服務現已刪除。

App Runner API or AWS CLI

若要使用應用程式執行器 API 刪除您的服務 AWS CLI，或呼叫 DeleteServiceAPI 動作。如果調用返回顯示服務對象的成功響應`"Status": "OPERATION_IN_PROGRESS"`，則應用程序運行器開始刪除您的服務。

參考環境變數

使用 App Runner，您可以在建立服務或更新服務時，在服務中將機密和設定參考為環境變數。

您可以將純文字中的逾時和重試計數等非機密組態資料引用為索引鍵值配對。您在純文本中引用的配置數據不會加密，其他人可以在 App Runner 服務配置和應用程序日誌中看到。

> **Note**
>
> 出於安全原因，請勿在 App Runner 服務中引用純文本中的任何敏感數據。

將敏感資料參考為環境變數

App Runner 支援在服務中安全地將敏感資料引用為環境變數。請考慮儲存您要參考的敏感資料AWS Secrets Manager或AWS Systems Manager 參數存放區。然後，您可以從 App Runner 主控台或呼叫 API，在服務中安全地將它們引用為環境變數。這有效地將秘密和參數管理與應用程式程式碼和服務組態分開，從而改善在 App Runner 上執行之應用程式的整體安全性。

> **Note**
>
> 應用程序運行器不會向您收取引用 Secrets Manager 和 SSM 參數存儲作為環境變量的費用。不過，您需要為使用 Secrets Manager 和 SSM 參數存放區支付標準定價。
> 如需關於定價的詳細資訊，請參閱下列資訊：
>
> - AWS Secrets Manager 定價
> - AWS SSM 參數存放區定價

以下是將敏感資料參考為環境變數的程序：

1. 將機密資料 (例如 API 金鑰、資料庫認證、資料庫連線參數或應用程式版本) 儲存為秘密或參數到 AWS Secrets Manager 或 AWS Systems Manager 參數存放區中。
2. 更新執行個體角色的 IAM 政策，以便應用程式執行器可以存取秘密管理員和 SSM 參數存放區中儲存的密碼和參數。如需詳細資訊，請參閱 許可。

3. 透過指派名稱並提供其 Amazon 資源名稱 (ARN)，安全地將機密和參數引用為環境變數。您可以在建立服務或更新服務的組態時新增環境變數。您可以使用下列其中一個選項來新增環境變數：
   - 應用程式執行器
   - 應用程序亞軍 API
   - `apprunner.yaml` 組態檔案

   > **Note**
   >
   > 建立或更新 App Runner 服務時，您無法指派`PORT`為環境變數的名稱。它是應用程式執行器服務的保留環境變數。

   如需如何參考密碼和參數的詳細資訊，請參閱管理環境變數。

> **Note**
>
> 由於 App Runner 僅儲存秘密和參數 ARN 的參考，因此在 App Runner 服務設定和應用程式記錄檔中的其他人看不到敏感資料。

考量事項

- 請務必使用適當的權限更新執行個體角色，以存取參數存放區中 AWS Secrets Manager 或中的機密和參 AWS Systems Manager 數。如需詳細資訊，請參閱 許可。
- 請確定 AWS Systems Manager 參數存放區與您要啟動或更新的服務位於相同 AWS 帳戶 的服務中。目前，您無法跨帳戶參考 SSM 參數存放區參數。
- 當密碼和參數值旋轉或變更時，它們不會在 App Runner 服務中自動更新。重新部署您的應用程序運行器服務，因為應用程序運行器僅在部署期間提取秘密
- 您還可以選擇通過應用程序運行器服務中的 SDK 直接調用 AWS Secrets Manager 和 AWS Systems Manager 參數存儲。
- 為了避免錯誤，請在將它們引用為環境變量時確保以下內容：
  - 您可以指定密碼的右 ARN。
  - 您可以指定參數的正確名稱或 ARN。

許可

若要啟用存放在 AWS Secrets Manager 或 SSM 參數存放區中的參考密碼和參數，請將適當的許可新增至執行個體角色的 IAM 政策，以存取 Secrets Manager 和 SSM 參數存放區。

> **Note**
>
> 未經您的許可，應用程序運行器無法訪問您帳戶中的資源。您可以透過更新 IAM 政策來提供許可。

您可以使用下列政策範本在 IAM 主控台中更新執行個體角色。您可以修改這些原則範本以符合您的特定需求。如需更新執行個體角色的詳細資訊，請參閱《IAM 使用者指南》中的修改角色。

> **Note**
>
> 您也可以在建立環境變數時，從 App Runner 主控台複製下列範本。

將下列範本複製到您的執行個體角色，以新增參考密碼的權限AWS Secrets Manager。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt*"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
        "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
      ]
    }
  ]
}
```

將下列範本複製到您的執行個體角色，以便從AWS Systems Manager參數存放區新增參考參數的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter_name>"
      ]
    }
  ]
}
```

管理您的環境變數

使用下列其中一種方法來管理 App Runner 服務的環境變數：

- the section called “應用程式執行器”
- the section called “應用程序運行器 API 或 AWS CLI”

應用程式執行器

在 App Runner 主控台上建立服務或更新服務時，您可以新增環境變數。

新增環境變數

若要新增環境變數

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 根據您要建立或更新服務，執行下列其中一個步驟：
   - 如果您要建立新服務，請選擇 [建立應用程式執行器服務]，然後移至 [設定服務]。
   - 如果您要更新現有的服務，請選取您要更新的服務，然後前往服務的 [組態] 索引標籤。
3. 轉到環境變量-服務設置下的可選。
4. 根據您的需求選擇以下任一選項：
   - 從環境變數來源中選擇純文字，然後在環境變數名稱和環境變數值下分別輸入其鍵值對。

> **Note**
>
> 如果您要參照非敏感資料，請選擇「純文字」。此資料未加密，而且其他人可以在 App Runner 服務設定和應用程式記錄檔中看到。

- 從環境變數來源中選擇 Secret Manager，以參照儲存在服務中 AWS Secrets Manager 做為環境變數的密碼。分別在環境變數名稱和環境變數值下提供您要參考的機密的環境變數名稱和 Amazon 資源名稱 (ARN)。
- 從環境變數來源中選擇「SSM 參數存放區」，將儲存在 SSM 參數存放區中的參數作為服務中的環境變數參照。在環境變數名稱和環境變數值下分別提供您要參考之參數的環境變數名稱和 ARN。

> **Note**
>
> - 建立或更新 App Runner 服務時，您無法指派`PORT`為環境變數的名稱。它是應用程式執行器服務的保留環境變數。
> - 如果 SSM 參數存放區參數與您要啟動的服務相同 AWS 區域 ，您可以指定完整的 Amazon 資源名稱 (ARN) 或參數名稱。如果參數位於不同的「區域」中，則需要指定完整的 ARN。
> - 請確定您所參考的參數與您正在啟動或更新的服務位於相同的帳戶中。目前，您無法跨帳戶參考 SSM 參數存放區參數。

5. 選擇新增環境變數以參照另一個環境變數。
6. 展開 IAM 政策範本，以檢視和複製為和 SSM 參數存放區提供 AWS Secrets Manager 的 IAM 政策範本。只有在尚未使用所需許可更新執行個體角色的 IAM 政策時，才需要執行此操作。如需詳細資訊，請參閱 許可。

移除環境變數

刪除環境變數之前，請確定您的應用程式程式碼已更新以反映相同的程式碼。如果應用程式程式碼未更新，您的應用程式執行器服務可能會失敗。

若要移除環境變數

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 移至您要更新之服務的「組態」索引標籤。

3. 轉到環境變量-服務設置下的可選。
4. 在您要移除的環境變數旁邊選擇「移除」。您會收到確認刪除的訊息。
5. 選擇刪除。

應用程序運行器 API 或 AWS CLI

您可以參考儲存在 Secrets Manager 和 SSM 參數存放區中的機密資料，方法是將它們新增為服務中的環境變數。

> **Note**
>
> 更新執行個體角色的 IAM 政策，以便應用程式執行器可以存取秘密管理員和 SSM 參數存放區中儲存的機密和參數。如需詳細資訊，請參閱 許可。

若要將密碼和組態參照為環境變數

1. 在密碼管理員或 SSM 參數存放區中建立密碼或組態。

   下列範例顯示如何使用 SSM 參數存放區建立密碼和參數。

   Example 創建一個秘密-請求

   下列範例顯示如何建立代表資料庫認證的密碼。

   ```
   aws secretsmanager create-secret \
   –name DevRdsCredentials \
   –description "Rds credentials for development account." \
   –secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
   ```

   Example 創建一個秘密-響應

   ```
   arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
   ```

   Example 建立組態-請求

   下列範例顯示如何建立代表 RDS 連接字串的參數。

   ```
   aws systemsmanager put-parameter \
   ```

```
–name DevRdsConnectionString \
–value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
–type "String" \
–description "Rds connection string for development account."
```

Example 建立組態-回應

```
arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
```

2. 將密碼和組態新增為環境變數，以參照儲存在 Secrets Manager 和 SSM 參數存放區中的密碼和組態。您可以在建立或更新應用程式執行器服務時新增環境變數。

   下列範例顯示如何在程式碼型和以映像為基礎的 App Runner 服務上，將機密和組態參考為環境變數。

   Example 基於圖像的應用程序運行器服務的輸入 .json 文件

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "<image-identifier>",
      "ImageConfiguration": {
        "Port": "<port>",
        "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
          "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
        }
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example 映像式應用程式執行器服務 — 請求

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example 映像式應用程式執行器服務 — 回應

```
{
...
      "ImageRepository": {
         "ImageIdentifier":"<image-identifier>",
         "ImageConfiguration":{
            "Port": "<port>",
            "RuntimeEnvironmentSecrets":{
               "Credential1":
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
               "Credential2": "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
            },
            "ImageRepositoryType":"ECR"
      }
   },
   "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

Example 基於代碼的應用程序運行器服務的輸入 .json 文件

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-
github-connection/XXXXXXXXXX"
    },
    "AutoDeploymentsEnabled": false,
    "CodeRepository": {
```

```
      "RepositoryUrl": "<repository-url>",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      },
      "CodeConfiguration": {
        "ConfigurationSource": "API",
        "CodeConfigurationValues": {
          "Runtime": "<runtime>",
          "BuildCommand": "<build-command>",
          "StartCommand": "<start-command>",
          "Port": "<port>",
          "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
            "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
          }
        }
      }
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example 基於代碼的應用運行器服務-請求

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example 基於代碼的應用運行器服務-響應

```
{
...
   "SourceConfiguration":{
      "CodeRepository":{
         "RepositoryUrl":"<repository-url>",
         "SourceCodeVersion":{
```

```
            "Type":"Branch",
            "Value":"main"
         },
         "CodeConfiguration":{
            "ConfigurationSource":"API",
            "CodeConfigurationValues":{
               "Runtime":"<runtime>",
               "BuildCommand":"<build-command>",
               "StartCommand":"<start-command>",
               "Port":"<port>",
               "RuntimeEnvironmentSecrets":{
                  "Credential1" :
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXX",
                  "Credential2" : "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
               }
            }
         }
      },
      "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

3. apprunner.yaml模型會更新以反映新增的密碼。

   以下是更新apprunner.yaml模型的範例。

   Example **apprunner.yaml**

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - python -m pip install flask
run:
  command: python app.py
  network:
    port: 8080
  env:
```

```
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from:
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX"
    - name: my-parameter
      value-from: "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter-
name>"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

與應用程序亞軍聯網

本章說明 AWS App Runner 服務的網路組態。

從本章中，您將學到以下內容：

- 如何為私人和公有端點設定傳入流量。如需詳細資訊，請參閱為傳入流量設定網路組態。
- 如何設定傳出流量以存取 Amazon VPC 中執行的其他應用程式。如需詳細資訊，請參閱針對傳出流量啟用 VPC 存取。

> **Note**
>
> 應用程式執行器目前僅支援公用傳入流量的雙堆疊 (IPv4 和 IPv6) 位址類型。對於傳出流量和私人傳入流量，僅支援 IPv4。

主題

- 術語
- 設定連入流量的網路組態
- 為傳出流量啟用 VPC 存取

術語

為了知道如何自定義您的網絡流量以滿足您的需求，讓我們了解本章中使用的以下術語。

一般條款

若要瞭解與 Amazon 虛擬私有雲 (VPC) 建立關聯需要什麼，讓我們了解以下術語：

- VPC：Amazon VPC 是邏輯隔離的虛擬網路，可讓您完全控制虛擬網路環境，包括資源配置、連線和安全性。這是一個虛擬網路，與您在自己的資料中心中操作的傳統網路非常類似。
- VPC 介面端點：VPC 介面端點 (AWS PrivateLink 資源) 將 VPC 連接到端點服務。建立 VPC 介面端點，將流量傳送至使用 Network Load Balancer 分配流量的端點服務。使用 DNS 來解析目的地為端點服務的流量。
- 地區：每個區域是一個單獨的地理區域，您可以在其中託管應用程序運行器服務。

- 可用區域：可用區域是區 AWS 域內的隔離位置。它是一或多個具備備援電源、網路和連線能力的獨立資料中心。可用區域會協助您使生產應用程式具備高可用性、容錯能力和可擴展性。
- 子網路：子網路是 VPC 中的一系列 IP 位址。子網必須位於單一可用區域。您可以將 AWS 資源啟動到指定的子網路中。針對必須連線至網際網路的資源使用公有子網，並針對不會連線至網際網路的資源使用私有子網。
- 安全性群組：安全性群組控制允許存取和離開與其相關聯之資源的流量。安全群組提供額外的安全層，以保護每個子網路中的 AWS 資源，讓您更好地控制網路流量。當您建立 VPC 時，其具有一個預設的安全群組。您可以為每個 VPC 建立額外的安全群組。您只能將安全群組與建立該群組的 VPC 內的資源相關聯。
- 雙堆疊：雙堆疊是一種支援來自 IPv4 和 IPv6 端點之網路流量的位址類型。

設定傳出流量的特定術語

VPC 連接器

VPC 連接器是一種應用程式執行器資源，可讓應用程式執行器服務存取在私有 Amazon VPC 中執行的應用程式。

設定傳入流量的特定詞彙

若要瞭解如何讓您的服務只能在 Amazon VPC 內私有存取，讓我們了解下列術語：

- VPC 入口連線：VPC 入口連線是應用程式執行器資源，可為傳入流量提供應用程式執行器端點。當您在 App Runner 主控台上為傳入流量選擇私人端點時，應用程式執行器會在幕後指派 VPC 入口連線資源。VPC 入口連線資源會將您的應用程式執行器服務連線到 Amazon VPC 的 VPC 介面端點。

  > **Note**
  >
  > 如果您使用的是應用程式執行器 API，則不會自動建立 VPC 入口連線資源。

- 私有端點：私有端點是應用程式執行器主控台選項，您可以選擇將傳入網路流量設定為僅可從 Amazon VPC 內存取。

設定連入流量的網路組態

您可以將服務設定為接收來自私有或公用端點的傳入流量。

公用端點是預設組態。它將您的服務打開到來自公共互聯網的任何傳入流量。此外，您還可以靈活地選擇服務的網際網路通訊協定第 4 版 (IPv4) 或雙堆疊 (IPv4 和 IPv6) 位址類型。

私有端點僅允許來自 Amazon VPC 的流量存取您的應用程式執行器服務。這是通過為您的應用程序運行器服務設置 VPC 接口端點（一種 AWS PrivateLink 資源）來實現的。因此，在 Amazon VPC 和您的應用程式執行器服務之間建立私有連線。

> **Note**
>
> 應用程式執行程式目前僅支援公用端點的雙堆疊 (IPv4 和 IPv6) 位址類型。對於私有端點，僅支援 IPv4。

以下是針對內送流量設定網路組態時所涵蓋的主題：

- 如何設定傳入流量，使您的服務只能在 Amazon VPC 內私有使用。如需詳細資訊，請參閱針對傳入流量啟用私人端點。
- 如何設定您的服務以接收來自雙堆疊位址類型的網際網路流量。如需詳細資訊，請參閱針對公用傳入流量啟用雙堆疊。

標頭

透過應用程式執行器，您可以存取進入應用程式之流量的原始來源 IPv4 和 IPv6 位址。原始來源 IP 位址會透過將`X-Forwarded-For`要求標頭指派給它們來保留。這可讓您的應用程式在需要時擷取原始來源 IP 位址。

> **Note**
>
> 如果您的服務設定為使用私有端點，則無法使用`X-Forwarded-For`要求標頭來存取原始來源 IP 位址。如果使用，它檢索假值。

為傳入流量啟用私有端點

根據預設，當您建立 AWS App Runner 服務時，可透過網際網路存取該服務。但是，您也可以將應用程序運行器服務設為私有，並且只能從 Amazon Virtual Private Cloud（Amazon VPC）中訪問。

使用您的 App Runner 服務私有，您可以完全控制傳入流量，從而增加了一層安全性。這在各種使用案例中很有幫助，包括執行內部 API、企業 Web 應用程式或仍在開發中的應用程式，這些應用程式需要更高層級的隱私權和安全性，或者需要符合特定合規性需求。

> **Note**
>
> 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
> 若要進一步了解基礎設施安全和安全群組 (包括最佳實務)，請參閱 Amazon VPC 使用者指南中的以下主題：使用安全群組控制網路流量和控制 AWS 資源的流量。

當您的應用程式執行器服務是私有的時候，您可以從 Amazon VPC 中存取您的服務。不需要網際網路閘道、NAT 裝置或 VPN 連線。

> **Note**
>
> 應用程式執行器目前僅支援公用傳入流量的雙堆疊 (IPv4 和 IPv6) 位址類型。對於傳出流量和私人傳入流量，僅支援 IPv4。

考量事項

- 在為 App Runner 設定 VPC 介面端點之前，請先檢閱AWS PrivateLink 指南中的考量事項。
- 應用程式執行器不支援 VPC 端點原則。默認情況下，允許通過 VPC 界面端點對應用程序運行器的完全訪問。或者，您可以將安全群組與端點網路介面相關聯，以控制透過 VPC 介面端點傳送至 App Runner 的流量。
- 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
- 啟用私有端點後，您的服務只能從 VPC 存取，而且無法從網際網路存取。
- 為了獲得更高的可用性，建議您為 VPC 介面端點在可用區域中選取至少兩個不同的子網路。我們不建議只使用一個子網路。
- 您可以使用相同的 VPC 介面端點存取 VPC 中的多個應用程式執行器服務。

如需本節所使用術語的詳細資訊，請參閱術語。

許可

以下是啟用私有端點所需的權限清單：

- ec2：CreateTags
- ec2：CreateVpcEndpoint
- ec2：ModifyVpcEndpoint
- ec2：DeleteVpcEndpoints
- ec2：DescribeSubnets
- ec2：DescribeVpcEndpoints
- ec2：DescribeVpcs

VPC 介面端點

虛擬私人雲端介面端點是將 Amazon VPC 連接到端點服務的AWS PrivateLink資源。您可以透過傳遞 VPC 介面端點來指定要存取應用程式執行器服務的 Amazon VPC。若要建立 VPC 介面端點，請指定下列項目：

- Amazon VPC 以啟用連接。
- 新增安全性群組。依預設，會將安全群組指派給 VPC 介面端點。您可以選擇關聯自訂安全性群組，以進一步控制內送網路流量。
- 新增子網路。為了確保更高的可用性，建議您從中存取 App Runner 服務的每個可用區域選取至少兩個子網路。網路介面端點會在您為 VPC 介面端點啟用的每個子網路中建立。這些是由請求者管理的網路介面，可做為流量進入 App Runner 的進入點。申請者受管的網路介面是 AWS 服務代表您在您的 VPC 中建立的網路介面。
- 如果您使用的是 API，請新增應用程式執行器 VPC 介面端點`Servicename`。例如

```
com.amazonaws.region.apprunner.requests
```

您可以使用下列其中一項 AWS 服務建立 VPC 介面端點：

- 應用程序運行控制台 如需詳細資訊，請參閱管理私有端點。
- Amazon VPC 控制台或 API，以及 AWS Command Line Interface （ AWS CLI ）。如需詳細資訊，請參閱AWS PrivateLink 指南 AWS PrivateLink中的 AWS 服務 透過存取。

> **Note**
>
> 根據AWS PrivateLink 定價為您使用的每個 VPC 介面端點收費。因此，為了提高成本效益，您可以使用相同的 VPC 介面端點來存取 VPC 中的多個 App Runner 服務。但是，為了更好地隔離，請考慮為每個 App Runner 服務關聯不同的 VPC 界面端點。

VPC Ingress Connection

VPC 入口連線是一種應用程式執行器資源，可為傳入流量指定應用程式執行器端點。當您在 App Runner 主控台上為傳入流量選擇私人端點時，應用程式執行器會在幕後指派 VPC 入口連線資源。選擇此選項可僅允許來自 Amazon VPC 的流量存取您的應用程式執行器服務。VPC 入口連線資源會將您的應用程式執行器服務連線到 Amazon VPC 的 VPC 介面端點。只有在使用 API 作業設定傳入流量的網路設定時，才能建立 VPC 輸入連線資源。如需如何建立 VPC 輸入連線資源的詳細資訊，請參閱 AWS App Runner API 參考資料CreateVpcIngressConnection中的。

> **Note**
>
> 應用程式執行器的一個 VPC 入口連線資源可以連接到 Amazon VPC 的一個 VPC 介面端點。此外，您只能為每個應用程式執行器服務建立一個 VPC 入口連線資源。

私有端點

私有端點是應用程式執行器主控台選項，您可以選擇是否只想從 Amazon VPC 接收傳入流量。在 App Runner 主控台上選擇私有端點選項，可讓您選擇透過設定其 VPC 介面端點，將服務連線到 VPC。在幕後，應用程式執行器會將 VPC 輸入連線資源指派給您設定的 VPC 介面端點。

> **Note**
>
> 私人端點僅支援 IPv4 網路流量。

Summary

只允許來自 Amazon VPC 的流量存取您的應用程式執行器服務，將您的服務設為私有。為了實現這一目標，您可以使用應用程式執行器或 Amazon VPC 為所選 Amazon VPC 建立 VPC 界面端點。在 App Runner 主控台上，當您為內送流量啟用私人端點時，可以建立 VPC 介面端點。然後，應用程式執行

器會自動建立 VPC 入口連線資源，並連線至 VPC 介面端點和您的應用程式執行器服務。這會建立私人服務連線，以確保只有來自所選 VPC 的流量才能存取您的 App Runner 服務。

管理私有端點

使用下列其中一種方法管理內送流量的私人端點：

- the section called “應用程式執行器”
- the section called “應用程序運行器 API 或 AWS CLI”

> **Note**
>
> 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
> 若要進一步了解基礎設施安全和安全群組 (包括最佳實務)，請參閱 Amazon VPC 使用者指南中的以下主題：使用安全群組控制網路流量和控制 AWS 資源的流量。

應用程式執行器

當您使用 App Runner 主控台建立服務時，或稍後更新其組態時，您可以選擇設定傳入流量。

若要設定傳入流量，請選擇下列其中一項。

- 公共端點：使您的服務可以通過互聯網訪問所有服務。依預設，會選取公用端點。
- 私有端點：讓您的應用程式執行器服務只能在 Amazon VPC 內存取。

> **Note**
>
> 目前，應用程式執行器僅針對公用端點支援 IPv6。在 Amazon 虛擬私有雲端 (Amazon VPC) 中託管的應用程式執行器服務不支援 IPv6 端點。如果您將使用雙堆疊公用端點的服務更新到私有端點，則 App Runner 服務將預設為僅支援來自 IPv4 端點的流量，且無法從 IPv6 端點接收流量。

啟用私有端點

將私有端點與您要存取之 Amazon VPC 的虛擬私人雲端界面端點相關聯，以啟用私有端點。您可以建立新的 VPC 介面端點，也可以選擇現有的端點。

若要建立 VPC 介面端點

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到配置服務下的網絡部分。
3. 針對 [內送網路流量] 選擇 [私人端點]。開啟使用 VPC 介面端點連線至 VCP 的選項。
4. 選擇 [建立新端點]。建立新的 VPC 介面端點對話方塊隨即開啟。
5. 輸入 VPC 介面端點的名稱。
6. 從可用的下拉式清單中選擇所需的 VPC 介面端點。
7. 從下拉式清單中選擇安全性群組。新增安全群組可為 VPC 介面端點提供額外的安全層。建議您選擇兩個以上的安全性群組。如果您未選擇安全性群組，App Runner 會將預設安全群組指派給 VPC 介面端點。確定安全性群組規則不會封鎖想要與 App Runner 服務通訊的資源。安全性群組規則必須允許與您的應用程式執行器服務互動的資源。

   > **Note**
   >
   > 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
   >
   > 若要進一步了解基礎設施安全和安全群組 (包括最佳實務)，請參閱 Amazon VPC 使用者指南中的以下主題：使用安全群組控制網路流量和控制 AWS 資源的流量。

8. 從下拉式清單中選擇所需的子網路。建議您針對從中存取 App Runner 服務的每個可用區域選取至少兩個子網路。
9. (選擇性) 選擇「新增標籤」，然後輸入標籤關鍵字和標籤值。
10. 選擇建立。[設定服務] 頁面隨即開啟，並在頂端列顯示成功建立 VPC 介面端點的訊息。

選擇現有的 VPC 介面端點

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到配置服務下的網絡部分。

3. 針對 [內送網路流量] 選擇 [私人端點]。開啟使用 VPC 介面端點連線至 VPC 的選項。顯示可用的 VPC 介面端點清單。
4. 選擇 VPC 介面端點下列出的必要 VPC 介面端點。
5. 選擇 [下一步] 建立您的服務。應用程序運行器啟用私有端點。

> **Note**
>
> 建立服務之後，您可以視需要選擇編輯與 VPC 介面端點關聯的安全性群組和子網路。

若要檢查私人端點的詳細資料，請移至您的服務，然後展開組態索引標籤下的 [網路] 區段。它會顯示與私有端點相關聯的 VPC 和 VPC 介面端點的詳細資料。

更新 VPC 介面端點

建立 App Runner 服務之後，您可以編輯與私有端點相關聯的 VPC 介面端點。

> **Note**
>
> 您無法更新端點名稱和 VPC 欄位。

若要更新 VPC 介面端點

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到您的服務，然後在左側面板上選擇網絡配置。
3. 選擇 [內送流量] 以檢視與個別服務相關聯的 VPC 介面端點。
4. 選擇您要編輯的 VPC 介面端點。
5. 選擇編輯。開啟用於編輯 VPC 介面端點的對話方塊。
6. 選擇所需的安全性群組和子網路，然後按一下更新。顯示 VPC 介面端點詳細資料的頁面隨即開啟，頂端列上顯示成功更新 VPC 介面端點的訊息。

> **Note**
>
> 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP

資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
若要進一步了解基礎設施安全和安全群組 (包括最佳實務)，請參閱 Amazon VPC 使用者指南中的以下主題：使用安全群組控制網路流量和控制 AWS 資源的流量。

刪除 VPC 介面端點

如果您不希望您的 App Runner 服務可以私人訪問，則可以將傳入流量設置為「公共」。變更為公用會移除私有端點，但不會刪除 VPC 介面端點

若要刪除 VPC 介面端點

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到您的服務，然後在左側面板上選擇網絡配置。
3. 選擇 [內送流量] 以檢視與個別服務相關聯的 VPC 介面端點。

> **Note**
> 刪除 VPC 介面端點之前，請透過更新服務將其從其連線的所有服務中移除。

4. 選擇刪除。

   如果有服務連線至 VPC 介面端點，則您會收到無法刪除 VPC 介面端點訊息。如果沒有服務連線至 VPC 介面端點，您會收到確認刪除的訊息。
5. 選擇刪除。[網路組態] 頁面會針對 [內送] 流量開啟，頂端列上顯示成功刪除 VPC 介面端點的訊息。

應用程序運行器 API 或 AWS CLI

您可以在只能從 Amazon VPC 內存取的應用程式執行器上部署應用程式。

如需將服務設為私人所需權限的資訊，請參閱the section called “許可”。

> **Note**
> 目前，應用程式執行器僅針對公用端點支援 IPv6。在 Amazon 虛擬私有雲端 (Amazon VPC) 中託管的應用程式執行器服務不支援 IPv6 端點。如果您將使用雙堆疊公用端點的服務更新到

私有端點，則 App Runner 服務將預設為僅支援來自 IPv4 端點的流量，且無法從 IPv6 端點接收流量。

建立到 Amazon VPC 的私有服務連接

1. 建立 VPC 介面端點 (AWS PrivateLink 資源) 以連線至應用程式執行器。若要這麼做，請指定要與應用程式產生關聯的子網路和安全群組。以下是建立 VPC 介面端點的範例。

   > **Note**
   >
   > 如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。因此，與 WAF Web ACL 相關聯的應用程式執行器私有服務的來源 IP 規則不會遵守 IP 型規則。
   > 若要進一步了解基礎設施安全和安全群組 (包括最佳實務)，請參閱 Amazon VPC 使用者指南中的以下主題：使用安全群組控制網路流量和控制 AWS 資源的流量。

   Example

   ```
   aws ec2 create-vpc-endpoint
    --vpc-endpoint-type: Interface
    --service-name: com.amazonaws.us-east-1.apprunner.requests
    --subnets: subnet1, subnet2
    --security-groups: sg1
   ```

2. 透過 CLI 使用CreateService或UpdateService應用程式執行器 API 動作來參考 VPC 介面端點。將您的服務設定為不可公開存取。`False`在`IsPubliclyAccessible`參`NetworkConfiguration`數成`IngressConfiguration`員中設定為。以下是參考 VPC 介面端點的範例。

   Example

   ```
   aws apprunner create-service
    --network-configuration: ingress-configuration=<ingress_configuration>
    --service-name: com.amazonaws.us-east-1.apprunner.requests
    --source-configuration: <source_configuration>
    # Ingress Configuration
    {
   ```

```
 "IsPubliclyAccessible": False
 }
```

3. 呼叫 `create-vpc-ingress-connection` API 動作，為應用程式執行器建立 VPC 入口連線資源，並將其與您在上一步中建立的 VPC 介面端點建立關聯。它會傳回用來存取指定 VPC 中服務的網域名稱。以下是建立 VPC 輸入連線資源的範例。

   Example 請求

```
aws apprunner create-vpc-ingress-connection
 --service-arn: <apprunner_service_arn>
 --ingress-vpc-configuration: {"VpcId":<vpc_id>, "VpceId": <vpce_id>}
 --vpc-ingress-connection-name: <vic_connection_name>
```

   Example 回應

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
   "Status": "PENDING_CREATION",
   "AccountId": <connection_owner_id>,
   "DomainName": <domain_name_associated_with_vpce>,
   "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
   "CreatedAt": <date_created>
}
```

更新 VPC 輸入連線

您可以更新 VPC 輸入連線資源。VPC 輸入連線必須處於下列其中一種狀態，才能更新：

- AVAILABLE
- 失敗 (_C) 建立
- 失敗 (_U) 更新

以下是更新 VPC 輸入連線資源的範例。

Example 請求

```
aws apprunner update-vpc-ingress-connection
```

```
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example 回應

```
{
    "VpcIngressConnectionArn":  <vpc_ingress_connection_arn>,
    "VpcIngressConnectionName":  <vic_connection_name>,
    "ServiceArn":  <apprunner_service_arn>,
    "Status": "FAILED_UPDATE",
    "AccountId":  <connection_owner_id>,
    "DomainName":  <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt":  <date_created>
}
```

刪除 VPC 輸入連線

如果您不再需要 Amazon VPC 的私有連線，可以刪除 VPC 入口連線資源。

VPC 輸入連線必須處於下列其中一種狀態才能刪除：

- AVAILABLE
- 建立失敗
- 更新失敗
- 刪除失敗

以下是刪除 VPC 輸入連線的範例

Example 請求

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example 回應

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
```

```
    "Status": "PENDING_DELETION",
    "AccountId": <connection_owner_id>,
    "DomainName": <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt": <date_created>,
    "DeletedAt": <date_deleted>
}
```

使用下列應用程式執行器 API 動作來管理服務的私人入埠流量。

- CreateVpcIngressConnection— 建立新的 VPC 入口連線資源。當您要將 App Runner 服務與 Amazon VPC 端點相關聯時，App Runner 會需要此資源。
- ListVpcIngressConnections— 傳回與您 AWS 帳戶相關聯的 AWS App Runner VPC 入口連線端點清單。
- DescribeVpcIngressConnection— 傳回 AWS App Runner VPC 入口連線資源的完整說明。
- UpdateVpcIngressConnection— 更新 AWS App Runner VPC 入口連線資源。
- DeleteVpcIngressConnection— 刪除與應用程式執行器服務相關聯的應用程式執行程式 VPC 入口連線資源。

有關使用應用程序運行器 API 的更多信息，請參閱應用運行器 API 參考指南。

為公用傳入流量啟用 IPv6

如果您希望服務接收來自 IPv6 位址或 IPv4 和 IPv6 位址的內送網路流量，請為公用端點選擇雙堆疊位址類型。當您建立新的應用程式時，您可以在 [設定服務] > [網路] 區段下找到此設定。如需如何使用應用程式執行器主控台或應用程式執行器 API 啟用 IPv6 的詳細資訊，請參閱the section called “管理公用端點的雙堆疊”。

如需有關在上採用 IPv6 的詳細資訊 AWS，請參閱上的 IPv6 AWS。

應用程序運行器僅支持公共應用運行器服務端點的雙堆棧。對於所有應用程序運行器私有服務，僅支持 IPv4。

> **Note**
>
> 當您將 IP 地址類型設置為雙堆棧，並將網絡配置從公共端點更改為私有端點時，App Runner 將自動將您的地址類型更改為 IPv4。這是因為應用程式執行器僅針對公用端點支援 IPv6。

瞭解有關 IPv4 與 IPv6 的背景資訊

IPv4 網路層 (通常用於在網際網路上路由傳送網路流量) 使用 32 位元位址配置。這個位址空間有限，而且可能會用盡大量的網路裝置。因此，網路位址轉譯 (NAT) 通常用於透過單一公用網路位址路由多個 IPv4 位址。

IPv6 是網際網路通訊協定的較新版本，以 IPv4 為基礎，並使用 128 位元定址配置擴充位址空間。使用 IPv6，您可以使用幾乎無限數量的連接設備來構建網絡。由於網路位址數量龐大，IPv6 不需要 NAT。

IPv4 和 IPv6 端點彼此不相容，因為 IPv4 端點無法接收內送 IPv6 流量，反之亦然。雙堆疊提供方便的解決方案，可同時支援 IPv4 和 IPv6 網路流量。

管理公共傳入流量的雙堆疊

使用下列其中一種方法管理公用傳入流量的雙堆疊位址類型：

- the section called “應用程式執行器”
- the section called “應用程序運行器 API 或 AWS CLI”

應用程式執行器

當您使用 App Runner 主控台建立服務時，或稍後更新其設定時，您可以為內送網際網路流量選擇雙堆疊位址類型。

啟用雙堆疊位址類型

1. 建立或更新服務時，請展開 [設定服務] 下的 [網路] 區段。
2. 針對 [內送網路流量] 選擇 [公用端點]。公用端點 IP 位址類型選項隨即開啟。
3. 展開「公用端點 IP 位址類型」以檢視下列 IP 位址類型。
   - IPv4
   - 雙堆疊 (IPv4 和 IPv6)

> **Note**
>
> 如果您未展開公用端點 IP 位址類型來進行選取，則應用程式執行器會將 IPv4 指派為預設設定。

4. 選擇「雙堆疊」(IPv4 和 IPv6)。
5. 如果您要建立服務，請選擇 [下一步]，然後選擇 [建立並部署] 否則，如果您要更新服務，請選擇 [儲存變更]。

   部署服務後，您的應用程式會開始接收來自 IPv4 和 IPv6 端點的網路流量。

> **Note**
>
> 目前，應用程式執行器僅針對公用端點支援 IPv6。在 Amazon 虛擬私有雲端 (Amazon VPC) 中託管的應用程式執行器服務不支援 IPv6 端點。如果您將使用雙堆疊公用端點的服務更新到私有端點，則 App Runner 服務將預設為僅支援來自 IPv4 端點的流量，且無法從 IPv6 端點接收流量。

若要變更地址類型

1. 按照以下步驟更新服務並導航到網絡。
2. 瀏覽至 [內送網路流量] 下的 [公用端點 IP 位址類型]，然後選取所需的位址類型。
3. 選擇儲存變更。您的服務會根據您的選擇進行更新。

應用程序運行器 API 或 AWS CLI

當您呼叫CreateService或UpdateService應用程式執行器 API 動作時，請使用`NetworkConfiguration`參數的`IpAddressType`成員來指定位址類型。您可以指定的支援值為`IPV4`和`DUAL_STACK`。指定是`DUAL_STACK`否要讓服務從 IPv4 和 IPv6 端點接收網際網路流量。如果未為指定任何值`IpAddressType`，則依預設會套用 IPv4。

以下是使用雙堆疊作為 IP 位址建立服務的範例。此範例會呼叫`input.json`檔案。

Example 要求建立具有雙堆疊支援的服務

```
aws apprunner create-service \
 --cli-input-json file://input.json
```

Example **`input.json`** 的內容

```
{
  "ServiceName": "example-service",
```

```
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
      "ImageConfiguration": {
        "Port": "8000"
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK"
    }
  }
}
```

Example 回應

```
{
  "Service": {
    "ServiceName": "example-service",
    "ServiceId": "<service-id>",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/example-
service/<service-id>",
    "ServiceUrl": "1234567890.us-east-2.awsapprunner.com",
    "CreatedAt": "2023-10-16T12:30:51.724000-04:00",
    "UpdatedAt": "2023-10-16T12:30:51.724000-04:00",
    "Status": "OPERATION_IN_PROGRESS",
    "SourceConfiguration": {
      "ImageRepository": {
        "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
        "ImageConfiguration": {
          "Port": "8000"
        },
        "ImageRepositoryType": "ECR_PUBLIC"
      },
      "AutoDeploymentsEnabled": false
    },
    "InstanceConfiguration": {
      "Cpu": "1024",
      "Memory": "2048"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
```

```
      "Interval": 5,
      "Timeout": 2,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK",
      "EgressConfiguration": {
        "EgressType": "DEFAULT"
      },
      "IngressConfiguration": {
        "IsPubliclyAccessible": true
      }
    }
  },
  "OperationId": "24bd100b1e111ae1a1f0e1115c4f11de"
}
```

> **Note**
>
> 目前，應用程式執行器僅針對公用端點支援 IPv6。在 Amazon 虛擬私有雲端 (Amazon VPC) 中託管的應用程式執行器服務不支援 IPv6 端點。如果您將使用雙堆疊公用端點的服務更新到私有端點，則 App Runner 服務將預設為僅支援來自 IPv4 端點的流量，且無法從 IPv6 端點接收流量。

如需 API 參數的詳細資訊，請參閱NetworkConfiguration。

為傳出流量啟用 VPC 存取

根據預設，您的 AWS App Runner 應用程式可以將訊息傳送到公用端點。這包括您自己的解決方案 AWS 服務，以及任何其他公共網站或 Web 服務。您的應用程式甚至可以從 Amazon 虛擬私有雲端 (Amazon VPC) 將訊息傳送到在 VPC 中執行的應用程式的公有端點。如果您在啟動環境時未設定 VPC，應用程式執行器會使用預設的 VPC，這是公用的。

您可以選擇在自訂 VPC 中啟動環境，以自訂傳出流量的網路和安全性設定。您可以讓 AWS App Runner 服務從 Amazon 虛擬私有雲端 (Amazon VPC) 存取在私有 VPC 中執行的應用程式。執行此操作之後，您的應用程式可以與 Amazon 虛擬私人雲端 (Amazon VPC) 中託管的其他應用程式連接並將訊息傳送到其他應用程式。範例包括 Amazon RDS 資料庫 ElastiCache、Amazon 以及在私有 VPC 中託管的其他私有服務。

VPC 連接器

您可以透過從應用程式執行器主控台 (稱為 VPC 連接器) 建立 VPC 端點，將您的服務與 VPC 建立關聯。若要建立 VPC 連接器，請指定 VPC、一或多個子網路，以及選擇性地指定一或多個安全群組。設定 VPC 連接器之後，您可以將其與一或多個應用程式執行器服務搭配使用。

一次性延遲

如果您將 App Runner 服務設定為輸出流量的自訂 VPC 連接器，則可能會遇到 2 到 5 分鐘的一次性啟動延遲。啟動程序會等到 VPC 連接器已準備好連線至其他資源，然後才會將服務狀態設定為執行中。您可以在第一次建立自訂 VPC 連接器時使用自訂 VPC 連接器來設定服務，也可以稍後執行服務更新來設定服務。

請注意，如果您對其他服務重複使用相同的 VPC 連接器組態，則不會有任何延遲。VPC 連接器組態以安全群組和子網路組合為基礎。對於指定的 VPC 連接器組態，延遲只會在初始建立 VPC 連接器超平面 ENI (彈性網路介面) 期間發生一次。

更多關於自訂 VPC 連接器和 AWS 超平面

應用程式執行器中的 VPC 連接器是以超平面 (AWS Hyperplane) 為基礎，這是位於多個 AWS 資源 (例如 Net work Load Balancer、NAT 閘道和 AWS) 背後的內部 Amazon 網路系統。 PrivateLink AWS Hyperplane 技術提供高輸送量和低延遲功能，以及更高的共用程度。當您建立 VPC 連接器並將其與服務產生關聯時，會在子網路中建立超平面 ENI。VPC 連接器組態以安全群組和子網路組合為基礎，您可以跨多個 App Runner 服務參考相同的 VPC 連接器。因此，基礎超平面 ENI 會在您的應用程式執行器服務之間共用。即使您擴大處理請求負載所需的任務數量，並使 VPC 中的 IP 空間更有效利用，這種共用也是可行的。如需詳細資訊，請參閱 AWS 容器部落格中的深入探討 AWS 應用程式執行器 VPC 聯網。

子網路

每個子網路都位於特定的可用區域中。為了獲得高可用性，建議您至少選取三個可用區域的子網路。如果該區域的可用區域少於三個，建議您在所有支援的可用區域中選取子網路。

為 VPC 選取子網路時，請確定您選擇的是私有子網路，而非公有子網路。這是因為，當您建立 VPC 連接器時，應用程式執行器服務會在每個子網路中建立超平面 ENI。每個超平面 ENI 只會分配一個私有 IP 地址，並標記為AWSAppRunnerManaged密鑰的標籤。如果您選擇公有子網路，執行 App Runner 服務時會發生錯誤。不過，如果您的服務需要存取網際網路或其他公眾上的某些服務 AWS 服務，請參閱the section called “選取子網路時的考量 ”。

選取子網路時的考量

- 當您將服務連線到 VPC 時，輸出流量無法存取公用網際網路。來自應用程式的所有輸出流量都會透過服務所連線的 VPC 導向。VPC 的所有網路規則都適用於應用程式的輸出流量。這表示您的服務無法存取公用網際網路和 AWS API。若要取得存取權，請執行下列其中一個動作：
  - 透過 NAT 閘道 Connect 子網路連線到網際網路。
  - 為您要存取的設 AWS 服務 定 VPC 端點。透過使 AWS PrivateLink用，您的服務會保留在 Amazon VPC 內。
- 某些可用區域中的某些可用區域 AWS 區域 不支援可與 App Runner 服務搭配使用的子網路。如果您在這些可用區域中選擇子網路，則無法建立或更新服務。在這些情況下，App Runner 會提供詳細的錯誤訊息，指向不支援的子網路和可用區域。發生這種情況時，請從要求中移除不支援的子網路來進行疑難排解，然後再試一次。

安全群組

您可以選擇性地在指定的子網路 AWS 下指定 App Runner 用來存取的安全性群組。如果您未指定安全性群組，應用程式執行器會使用 VPC 的預設安全性群組。預設的安全群組會允許所有傳出流量。

新增安全性群組可為 VCP 連接器提供額外的安全性，讓您可以更好地控制網路流量。VPC 連接器僅用於應用程式的輸出通訊。您可以使用輸出規則來允許與所需目的地端點進行通訊。您還必須確保與目標資源相關聯的任何安全群組都具有適當的輸入規則。否則，這些資源無法接受來自 VPC 連接器安全性群組的流量。

> **Note**
>
> 將服務與 VPC 建立關聯時，下列流量不會受到影響：
>
> - 入站流量 — 應用程式接收的內送訊息不會受到關聯 VPC 的影響。郵件會透過與您的服務相關聯的公用網域名稱路由傳送，且不會與 VPC 互動。
> - 應用程序運行器流量 — App Runner 代表您管理多個操作，例如提取源代碼和圖像，推送日誌和檢索秘密。這些動作產生的流量不會透過 VPC 路由傳送。

若要深入瞭解如何與 Amazon VPC AWS App Runner 整合，請參閱AWS 應用程式執行器虛擬私人雲端網路。

> **Note**
>
> 對於傳出流量應用程序運行器目前僅支持 IPv4。

管理 VPC 存取

> **Note**
>
> 如果您為服務建立輸出流量 VPC 連接器，接下來的服務啟動程序將會遇到一次性延遲。您可以在建立新服務時或之後使用服務更新設定此組態。如需詳細資訊，請參閱本指南的 < 使用應用程式執行器聯網 > 一章一次性延遲中的。

使用下列其中一種方法管理應用程式執行器服務的 VPC 存取權：

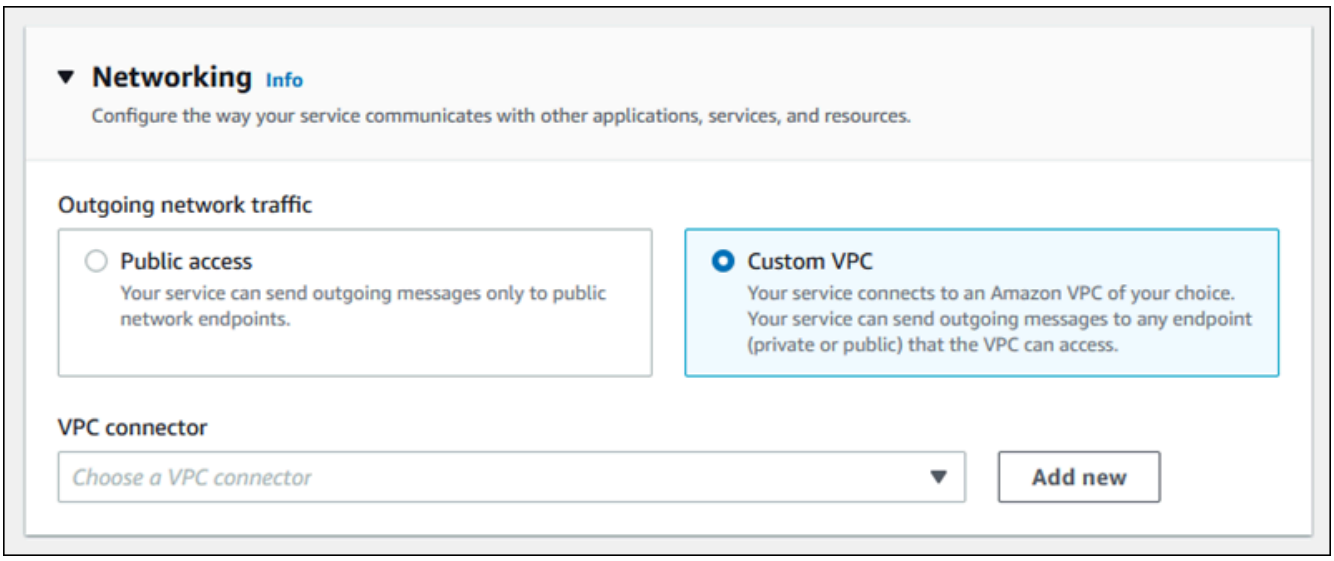
App Runner console

當您使用 App Runner 主控台建立服務時，或稍後更新其組態時，您可以選擇設定傳出流量。尋找主控台頁面上的 [網路設定] 區段。對於外寄網路流量，請選擇下列項目：

- 公共訪問：將您的服務與其他人的公共端點相關聯 AWS 服務。
- 自訂虛擬私人雲端：將您的服務與來自 Amazon VPC 的虛擬私人雲端產生關聯。您的應用程式可以與 Amazon VPC 中託管的其他應用程式連接並傳送訊息。

若要啟用自訂 VPC

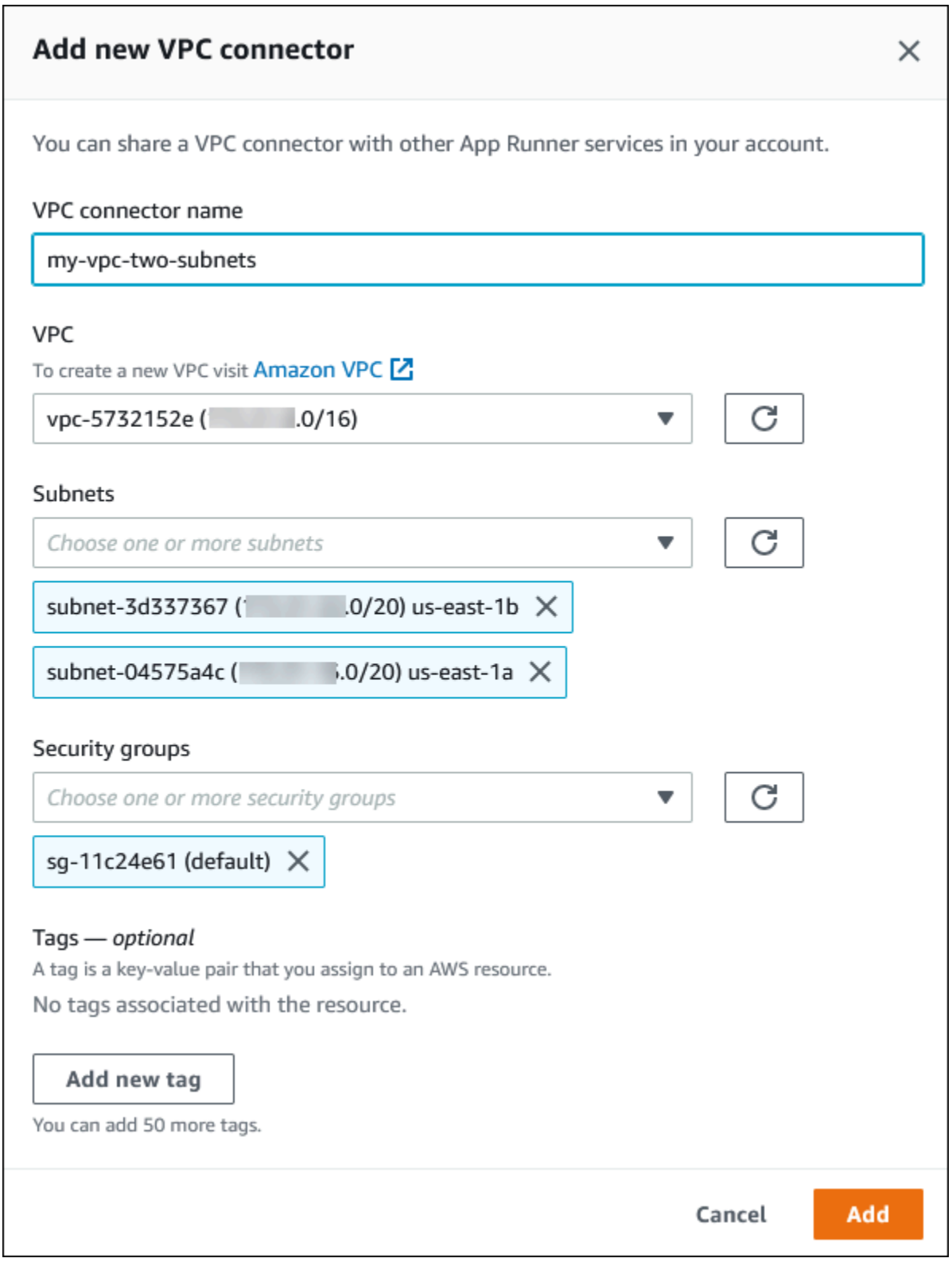
1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到配置服務下的網絡部分。



3. 選擇 [自訂 VPC] 做為 [外寄網路流量]。
4. 在功能窗格中，選擇 VPC 連接器。

   如果您已建立 VPC 連接器，則主控台會在您的帳戶中顯示 VPC 連接器清單。您可以選擇現有的 VPC 連接器，然後選擇 [下一步] 以檢閱組態。然後，轉到最後一步。或者，您可以使用下列步驟新增 VPC 連接器。
5. 選擇 [新增]，為您的服務建立新的 VPC 連接器。

   然後，將開啟 [新增 VPC 連接器] 對話方塊。



6. 輸入 VPC 連接器的名稱，然後從可用清單中選取所需的 VPC。
7. 若為子網路，請為您打算從中存取 App Runner 服務的每個可用區域選取一個子網路。為獲得更好的可用性，請選擇三個子網路。或者，如果子網路少於三個，請選擇所有可用的子網路。

> **Note**
>
> 請務必將私人子網路指派給 VPC 連接器。如果您將公用子網路指派給 VPC 連接器，則您的服務無法在更新期間自動建立或復原。

8. (選擇性) 在「安全性」群組中，選取要與端點網路介面關聯的安全群組。
9. (選用) 若要新增標籤，請選擇 Add new tag (新增標籤)，然後輸入標籤的鍵和值。
10. 選擇新增。

    您建立的 VPC 連接器的詳細資料會顯示在 VPC 連接器下。
11. 選擇 [下一步] 檢閱您的組態，然後選擇 [建立並部署]。

    App Runner 會為您建立 VPC 連接器資源，然後將其與您的服務產生關聯。如果成功建立服務，則主控台會顯示服務儀表板，以及新服務的服務概觀。

App Runner API or AWS CLI

當您呼叫CreateService或 UpdateServiceApp Runner API 動作時，請使用`NetworkConfiguration`參數的`EgressConfiguration`成員為您的服務指定 VPC 連接器資源。

使用下列應用程式執行器 API 動作來管理您的 VPC 連接器資源。

- CreateVpcConnector— 建立新的 VPC 連接器。
- ListVpcConnectors— 傳回與您 AWS 帳戶相關聯的 VPC 連接器清單。此清單包含完整描述。
- DescribeVpcConnector— 傳回 VPC 連接器的完整描述。
- DeleteVpcConnector— 刪除 VPC 連接器。如果您達到的 VPC 連接器配額 AWS 帳戶，您可能需要刪除不必要的 VPC 連接器。

若要在具有 VPC 輸出存取權的應用程式執行器上部署應用程式，您必須先建立 VPC 連接器。您可以指定一或多個要與應用程式關聯的子網路和安全性群組來執行此操作。然後，您可以在「建立」或UpdateService透過 CLI 參照 VPC 連接器，如下列範例所示：

```
            cat > vpc-connector.json <<EOF
{
"VpcConnectorName": "my-vpc-connector",
```

```
"Subnets": [
"subnet-a",
"subnet-b",
"subnet-c"
],
"SecurityGroups": [
"sg-1",
"sg-2"
]
}
EOF

aws apprunner create-vpc-connector \
--cli-input-json file:///vpc-connector.json

cat > service.json <<EOF

{
"ServiceName": "my-vpc-connected-service",
"SourceConfiguration": {
"ImageRepository": {
"ImageIdentifier": "<ecr-image-identifier> ",
"ImageConfiguration": {
"Port": "8000"
},
"ImageRepositoryType": "ECR"
}
},
"NetworkConfiguration": {
"EgressConfiguration": {
"EgressType": "VPC",
"VpcConnectorArn": "arn:aws:apprunner:..../my-vpc-connector"
}
}
}
EOF

aws apprunner create-service \
--cli-input-json file:///service.js
```

您的應用程序運行器服務的觀察性

AWS App Runner 與多種 AWS 服務集成，為您的 App Runner 服務提供廣泛的可觀察性工具套件。本章中的主題描述了這些功能。

主題

- 跟踪應用運行器服務活動
- 檢視串流至 CloudWatch 記錄的應用程式執行器記錄
- 檢視報告給的應用程式執行器服務 CloudWatch
- 處理應用程序運行器事件 EventBridge
- 記錄應用程序運行器 API 調用 AWS CloudTrail
- 使用 X-Ray 追蹤您的應用程式執行程式

跟踪應用運行器服務活動

AWS App Runner 使用操作列表來跟踪應用程序運行器服務中的活動。作業代表 API 動作的非同步呼叫，例如建立服務、更新組態和部署服務。以下各節說明如何在 App Runner 主控台中追蹤活動，以及如何使用 API。

追蹤應用程式執行器服務

使用下列其中一種方法追蹤您的應用程式執行器服務活動：

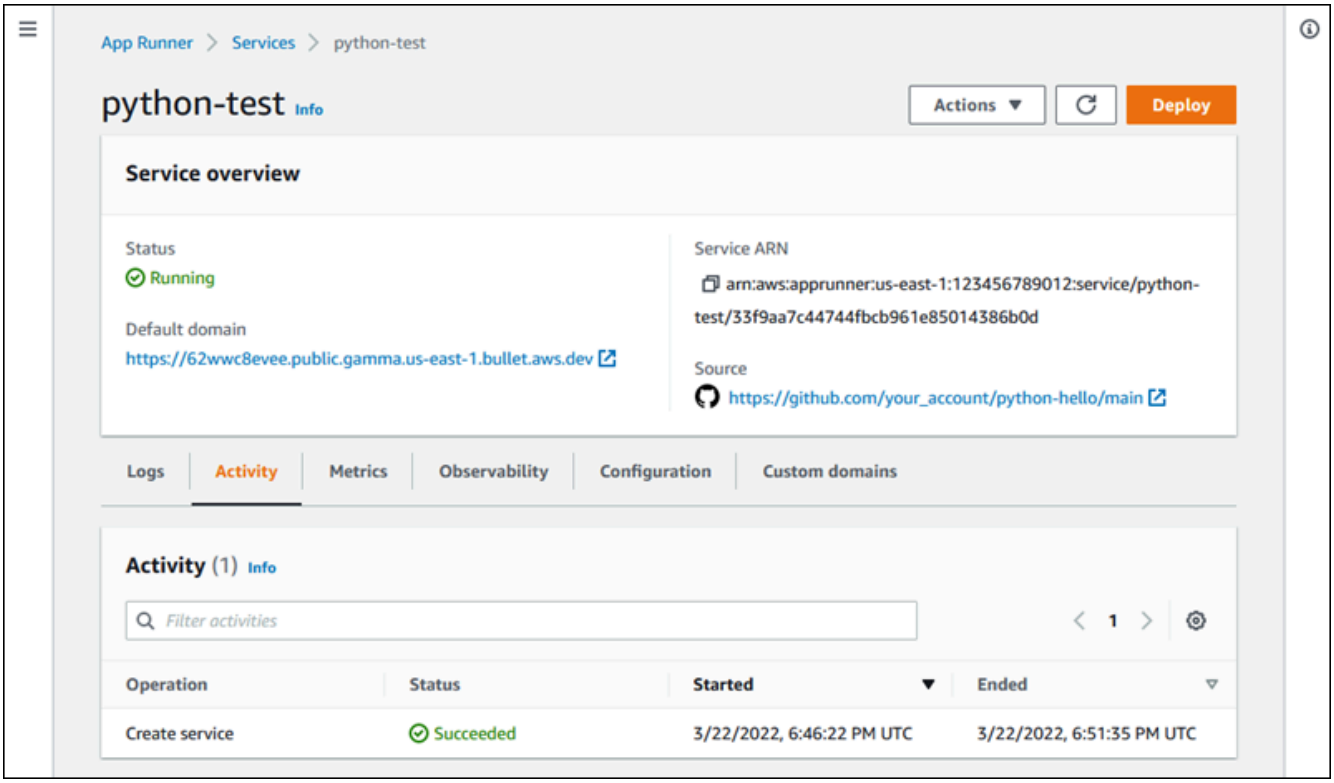
App Runner console

App Runner 主控台會顯示您的應用程式執行器服務活動，並提供更多探索作業的方式。

若要檢視服務的活動

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板及服務概觀。



3. 在服務儀表板頁面上，選擇 [活動] 索引標籤 (如果尚未選取)。

   控制台顯示操作列表。

4. 若要尋找特定作業，請輸入搜尋字詞，以縮小清單的範圍。您可以搜尋表格中顯示的任何值。
5. 選擇任何列出的操作以查看或下載相關日誌。

App Runner API or AWS CLI

指定應用程式執ListOperations行器服務的 Amazon 資源名稱 (ARN) 動作會傳回此服務上發生的作業清單。每個清單項目都包含一個作業 ID 和一些追蹤詳細資料。

檢視串流至 CloudWatch 記錄的應用程式執行器記錄

您可以使用 Amazon CloudWatch Logs 監控、存放和存取資源在各種 AWS 服務中產生的日誌檔。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。

AWS App Runner 收集應用程式部署和作用中服務的輸出，並將其串流至 CloudWatch 記錄檔。以下各節列出了應用程序運行器日誌流，並向您展示如何在應用程序運行器控制台中查看它們。

應用程式執行器記錄群組和串

CloudWatch 記錄會將記錄資料保存在記錄資料流中，以進一步組織在記錄群組中。記錄資料流是來自特定來源的一系列記錄事件。日誌群組是共用相同保留、監控和存取控制設定的日誌串流群組。

應用程序運行器定義了兩個 CloudWatch 日誌日誌組，每個日誌都有多個日誌流，每個日誌流，每個日誌組 AWS 帳戶。

服務記錄

服務日誌組包含由 App Runner 生成的日誌輸出，因為它管理您的應用程序運行器服務並對其採取行動。

記錄群組名稱	範例
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/service`

在服務記錄檔群組中，App Runner 會建立事件記錄資料流，以擷取 App Runner 服務生命週期中的活動。例如，這可能是啟動您的應用程序或暫停它。

此外，App Runner 會為與您的服務相關的每個長時間執行的非同步作業建立記錄資料流。記錄資料流名稱會反映作業類型和特定作業 ID。

部署是一種作業類型。部署記錄包含 App Runner 建立服務或部署新版應用程式時所執行之建置和部署步驟的記錄輸出。部署記錄資料流名稱開頭為`deployment/`，並以執行部署作業的 ID 結尾。此作業可能是CreateService呼叫初始應用程式部署，或StartDeployment呼叫每個進一步的部署。

在部署記錄中，每個記錄訊息都以前置詞開頭：

- `[AppRunner]`— 應用程式執行器在部署期間產生的輸出。
- `[Build]`-輸出您自己的構建腳本。

記錄串流名稱	範例
`events`	N/A (固定名稱)

記錄串流名稱	範例
operation-type /*operation-id*	deployment/c2c8eeedea164f45 9cf78f12a8953390

應用程式記錄

應用程式記錄群組包含執行中應用程式程式碼的輸出。

記錄群組名稱	範例
/aws/apprunner/ *service-n ame* /*service-id* /application	/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application

在應用程式記錄群組中，App Runner 會為執行應用程式的每個執行個體 (縮放單位) 建立記錄資料流。

記錄串流名稱	範例
instance/ *instance-id*	instance/1a80bc9134a84699b7 b3432ebeebb591

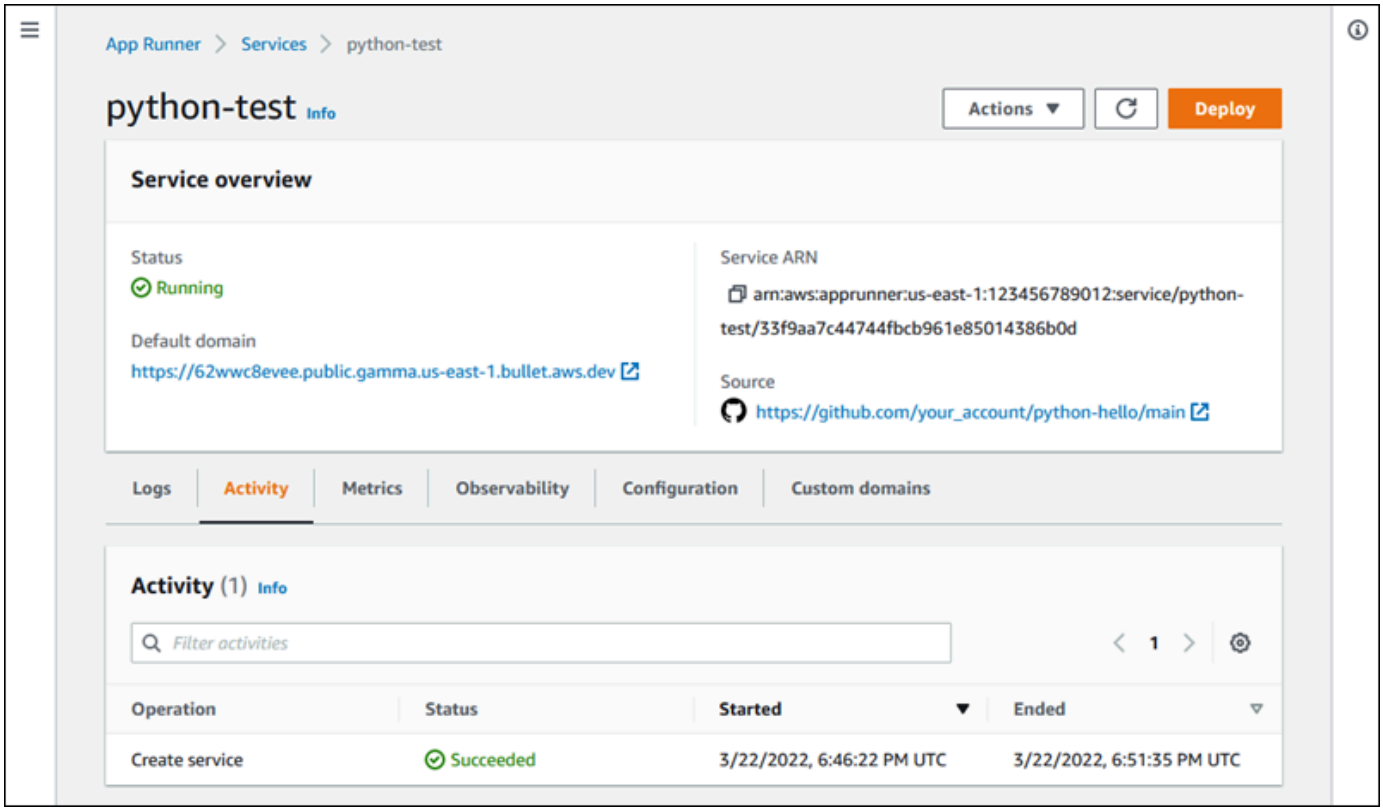
在主控台中檢視應用程式執行程式

App Runner 主控台會顯示服務的所有記錄檔摘要，並可讓您檢視、探索和下載這些記錄檔。

若要檢視服務的記錄

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

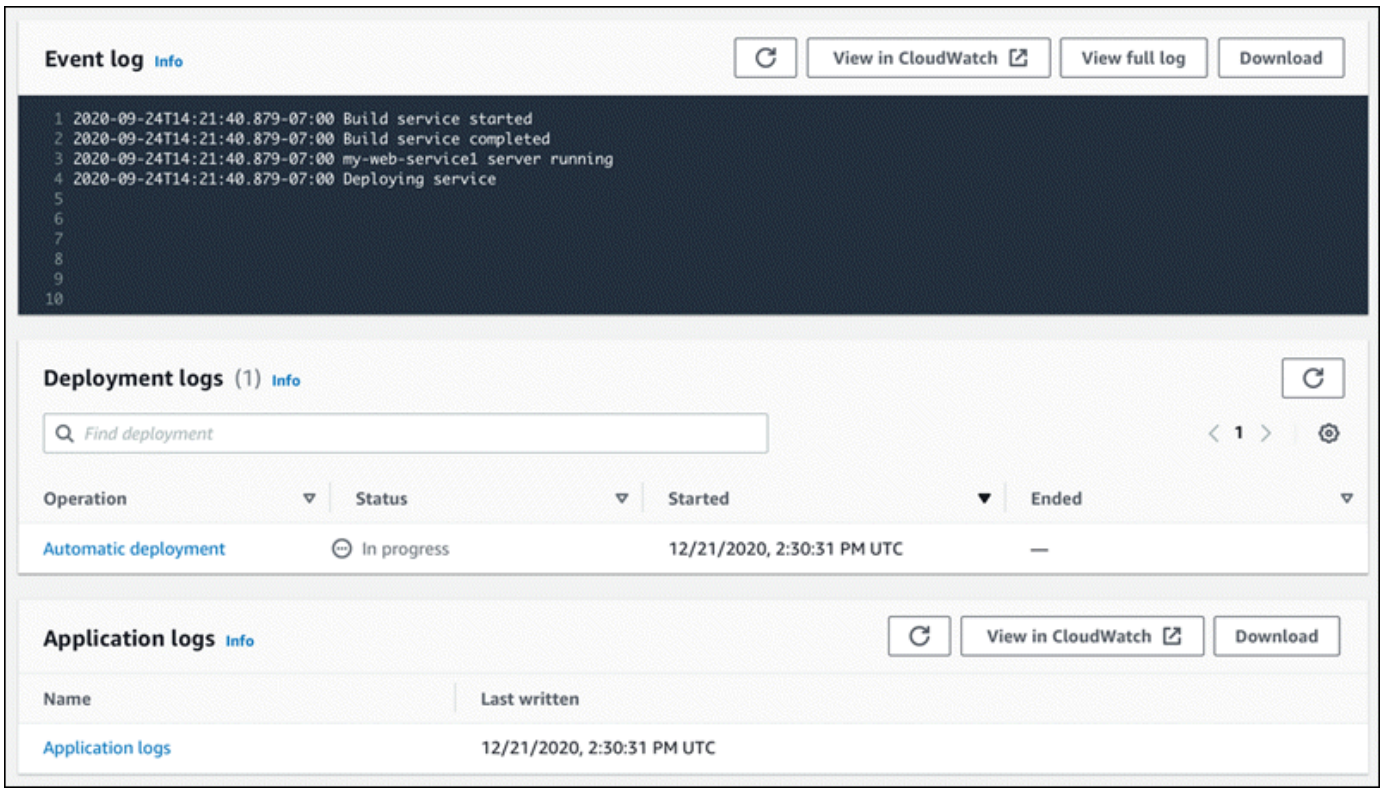
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇記錄檔索引標籤。

   主控台會在數個區段中顯示幾種類型的記錄檔：

   - 事件日誌 — 應用程序運行器服務生命週期中的活動。主控台會顯示最新的事件。
   - 部署記錄 — 來源儲存庫部署到您的應用程式執行器服務。主控台會針對每個部署顯示個別的記錄資料流。
   - 應用程式記錄檔 — 部署到應用程式執行器服務的 Web 應用程式輸出。主控台會將所有執行中執行個體的輸出結合到單一記錄串流中。



4. 若要尋找特定部署，請輸入搜尋詞彙，以縮小部署記錄清單的範圍。您可以搜尋表格中顯示的任何值。
5. 若要檢視記錄檔的內容，請選擇 [檢視完整記錄檔 (事件記錄檔)] 或記錄資料流名稱 (部署和應用程式記錄檔)。
6. 選擇 [下載] 以下載記錄檔。對於部署記錄串流，請先選取記錄串流。
7. 選擇 [檢視 CloudWatch於] 開啟 CloudWatch 主控台，並使用其完整功能瀏覽您的 App Runner 服務記錄檔。對於部署記錄串流，請先選取記錄串流。

> **Note**
>
> 如果您想要檢視特定執行個體的應用程式記錄檔，而不是合併的應用程式記錄檔，CloudWatch 主控台特別有用。

檢視報告給的應用程式執行器服務 CloudWatch

Amazon CloudWatch 監控您的 Amazon Web Services（AWS）資源和您實時運行 AWS 的應用程序。您可以用 CloudWatch 來收集和追蹤指標，這些指標是您可以針對資源和應用程式測量的變數。您

也可以使用它來建立監視指標的警示。達到特定臨界值時，會 CloudWatch 傳送通知或自動變更受監控的資源。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

AWS App Runner 收集各種指標，使您可以更好地了解 App Runner 服務的使用情況，性能和可用性。有些指標會追蹤執行 Web 服務的個別執行個體，而其他指標則處於整體服務層級。以下各節列出了應用程序運行器指標，並向您展示如何在應用程序運行器控制台中查看它們。

應用程式執行器

App Runner 會收集與您的服務相關的以下指標，並將其發佈到`AWS/AppRunner`命名空間 CloudWatch 中。

> **Note**
>
> 在 2023 年 8 月 23 日之前，CPU 使用率和記憶體使用率指標是根據 vCPU 單位和已使用的記憶體 MB (而非今天計算的百分比使用率) 為基礎。如果您的應用程式在此日期之前在 App Runner 上執行，而且您選擇返回在 App Runner 或 CloudWatch 主控台上檢視此日期的指標，您會看到這兩個單位的度量顯示，並且也會看到一些不規則的情況。

> **Important**
>
> 您必須在 2023 年 8 月 23 日之前，根據 CPU 使用率和記憶體使用率指標值來更新任何 CloudWatch 警示。根據使用率百分比 (而非 vCPU 或 MB)，將警示更新為觸發。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

系統會針對每個執行個體 (擴展單位) 個別收集執行個體層級指標

什麼是測量？	指標	Description
CPU utilization	`CPUUtilization`	一分鐘期間內的平均 CPU 使用率百分比，超出服務組態保留的 CPU 使用率總計。
Memory utilization	`MemoryUtilization`	在服務組態保留的記憶體總計中，一分鐘期間的平均記憶體使用量百分比。

系統會針對整個服務收集服務層次測量結果。

什麼是測量？	指標	Description
CPU utilization	`CPUUtilization`	在一分鐘期間內，所有執行處理的聚總 CPU 使用率百分比超出服務組態保留的 CPU 使用率總計。
Memory utilization	`MemoryUtilization`	在服務組態保留的記憶體總計中，一分鐘期間內，所有執行處理的彙總記憶體使用量百分比。
Concurrency	`Concurrency`	服務正在處理的並行請求的大約數目。
HTTP request count	`Requests`	服務接收到的 HTTP 要求數目。

HTTP status counts	`2xxStatusResponses`
	
	`4xxStatusResponses`
	
	`5xxStatusResponses`
		傳回每個回應狀態的 HTTP 要求數目，依類別 (2XX、4XX、5XX) 分組。

HTTP request latency	`RequestLatency`	Web 服務處理 HTTP 要求所花費的時間 (以毫秒為單位)。
Instance counts	`ActiveInstances`	為您的服務處理 HTTP 要求的執行個體數目。 **Note** 如果`ActiveInstances` 測量結果顯示為零，表示沒有服務要求。它並不表示服務的執行個體數目為零。

在主控台中檢視應用程式執行程式

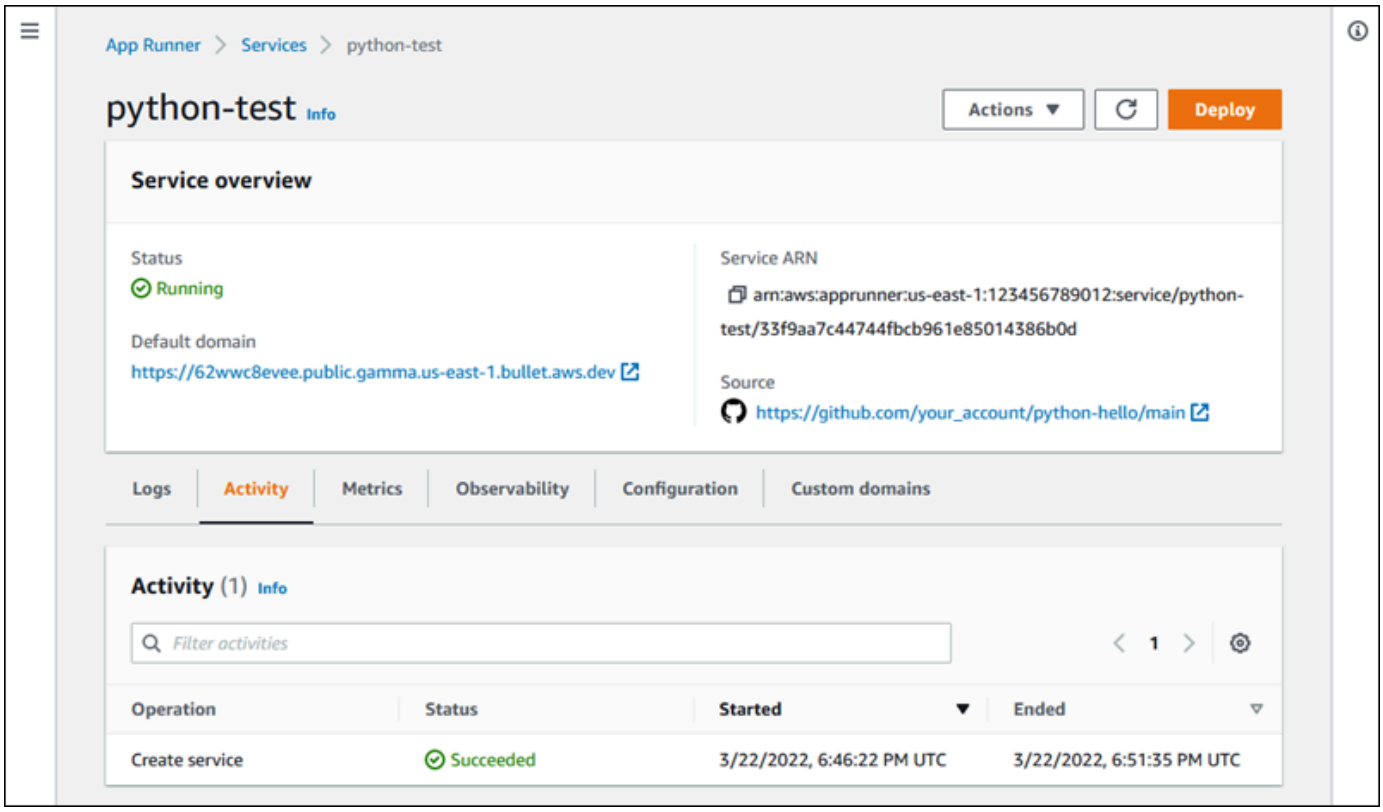
App Runner 控制台以圖形方式顯示 App Runner 為您的服務收集的指標，並提供更多探索它們的方法。

> **Note**
>
> 此時，主控台只會顯示服務測量結果。若要檢視執行個體指標，請使用 CloudWatch 主控台。

若要檢視服務的記錄

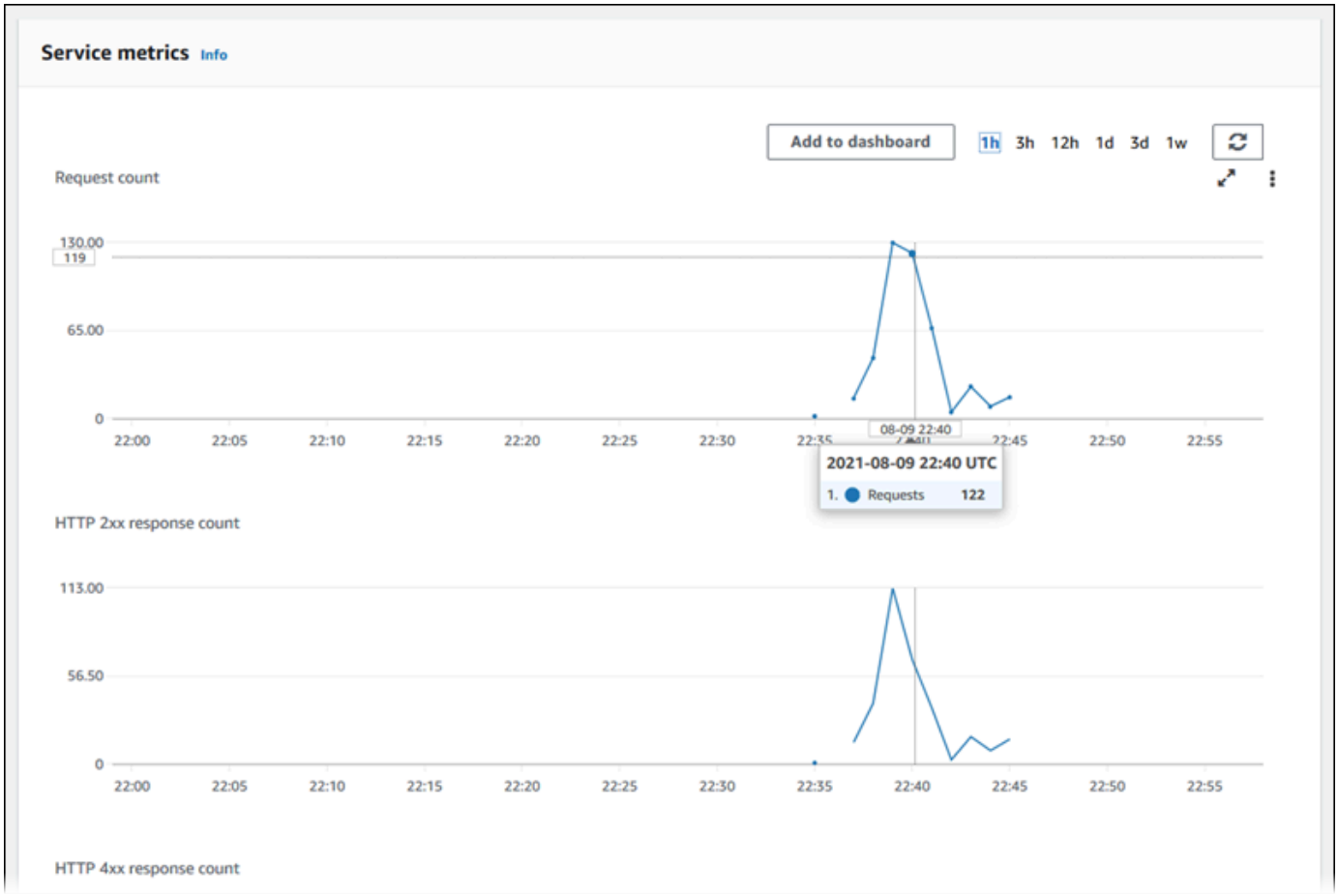
1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 在功能窗格中，選擇 [服務]，然後選擇您的應用程式執行器服務。

   主控台會顯示服務儀表板，其中包含「服務概觀」。



3. 在服務儀表板頁面上，選擇「指標」索引標籤。

   主控台會顯示一組量度圖表。



4. 選擇持續時間 (例如，12 小時)，將量度圖表範圍設定為該持續時間的最近期間。
5. 選擇其中一個圖形區段頂端的「新增至儀表板」，或使用任何圖形上的功能表，將相關度量新增至CloudWatch主控台中的儀表板，以供進一步調查。

處理應用程序運行器事件 EventBridge

使用 Amazon EventBridge，您可以設定事件驅動的規則，以監控來自 AWS App Runner 服務的即時資料串流是否有特定模式。符合規則的模式時， EventBridge 會在目標 (例如 AWS Lambda Amazon ECS 和 Amazon SNS) 中啟動動作。 AWS Batch例如，您可以設定傳送電子郵件通知的規則，方法是在對服務的部署失敗時傳送 Amazon SNS 主題。或者，您可以設定 Lambda 函數，在服務更新失敗時通知 Slack 通道。如需詳細資訊 EventBridge，請參閱 Amazon EventBridge 使用者指南。

應用程序運行器將以下事件類型發送到 EventBridge

- 服務狀態變更 — 應用程式執行器服務狀態的變更。例如，服務狀態變更為DELETE_FAILED。
- 服務作業狀態變更 — App Runner 服務上長時間非同步作業狀態的變更。例如，服務已開始建立、服務更新已順利完成，或服務部署完成但發生錯誤。

建立 EventBridge 規則以對應用程式執行器事件採取行動

EventBridge 事件是定義某些標準 EventBridge 欄位的物件，例如來源 AWS 服務和詳細資料 (事件) 類型，以及包含事件詳細資訊的事件特定欄位集。要創建 EventBridge 規則，您可以使用 EventBridge 控制台定義事件模式（應該跟踪哪些事件）並指定目標操作（應該在比賽中進行什麼操作）。事件模式類似於它匹配的事件。您可以指定要匹配的字段的子集，並為每個字段指定可能值的列表。本主題提供應用程式執行器事件和事件模式的範例。

如需有關建立 EventBridge 規則的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的為 AWS 服務建立規則。

> **Note**
>
> 某些服務支援中的預先定義模式 EventBridge。這簡化了事件模式的建立方式。您可以在表單上選取欄位值，然後為您 EventBridge 產生模式。目前，應用程式執行器不支援預先定義的模式。您必須輸入該模式作為 JSON 對象。您可以使用本主題中的範例作為起點。

應用程序運行器事件

這些是應用程序運行器發送到事件的一些示例 EventBridge。

- 服務狀態變更事件。具體而言，從變更為RUNNING狀態OPERATION_IN_PROGRESS的服務。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T11:54:23Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "previousServiceStatus": "OPERATION_IN_PROGRESS",
    "currentServiceStatus": "RUNNING",
    "serviceName": "my-app",
```

```
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service status is set to RUNNING.",
    "severity": "INFO"
  }
}
```

- 作業狀態變更事件。特別是，成功完成的UpdateService作業。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Operation Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T18:43:48Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "operationStatus": "UpdateServiceCompletedSuccessfully",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service update completed successfully. New application and
 configuration is deployed.",
    "severity": "INFO"
  }
}
```

應用程序運行器事件模式

下列範例會示範您可以在 EventBridge 規則中使用的事件模式，以比對一或多個 App Runner 事件。事件模式類似於一個事件。僅包含您要比對的欄位，並為每個欄位提供清單而非標量。

- 比對特定帳戶 (服務已不再處於狀態) 的所有服務RUNNING狀態變更事件。

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
```

```
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- 比對特定帳戶 (作業失敗) 之服務的所有作業狀態變更事件。

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```

應用程序運行器事件

服務狀態變更

服務狀態變更事件已detail-type設定為AppRunner Service Status Change。它具有下列詳細資料欄位和值：

```
"serviceId": "your service ID",
"serviceName": "your service name",
"message": "Service status is set to CurrentStatus.",
"previousServiceStatus": "any valid service status",
"currentServiceStatus": "any valid service status",
"severity": "varies"
```

操作狀態變更

作業狀態變更事件已detail-type設定為AppRunner Service Operation Status Change。它具有下列詳細資料欄位和值：

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

下表列出所有可能的狀態碼和相關訊息。

Status	訊息
CreateServiceStarted	服務建立已開始。
CreateServiceCompletedSuccessfully	服務建立成功完成。
CreateServiceFailed	服務建立失敗。如需詳細資訊，請參閱服務記錄。
DeleteServiceStarted	服務刪除已啟動。
DeleteServiceCompletedSuccessfully	服務刪除成功完成。
DeleteServiceFailed	服務刪除失敗。
UpdateServiceStarted	
UpdateServiceCompletedSuccessfully	服務更新已順利完成。部署新的應用程式和組態。
	服務更新已順利完成。已部署新組態。
UpdateServiceFailed	服務更新失敗。如需詳細資訊，請參閱服務記錄。
DeploymentStarted	部署已開始。
DeploymentCompletedSuccessfully	已成功完成部署。
DeploymentFailed	部署失敗。如需詳細資訊，請參閱服務記錄。
PauseServiceStarted	服務暫停已啟動。

Status	訊息
PauseServiceCompletedSuccessfully	服務暫停已順利完成。
PauseServiceFailed	服務暫停失敗。
ResumeServiceStarted	服務恢復已啟動。
ResumeServiceCompletedSuccessfully	服務恢復成功完成。
ResumeServiceFailed	服務恢復失敗。

記錄應用程序運行器 API 調用 AWS CloudTrail

應用程序 Runner 與服務集成 AWS CloudTrail，該服務可提供用戶，角色或應用程序運行器中的 AWS 服務所採取的操作的記錄。 CloudTrail捕獲應用程序運行器的所有 API 調用作為事件。擷取的呼叫包括來自 App Runner 主控台的呼叫，以及對應用程式執行器 API 作業的程式碼呼叫。如果您建立追蹤，您可以啟用持續傳遞 CloudTrail 事件到 Amazon S3 儲存貯體，包括應用程式執行器的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的信息 CloudTrail，您可以確定向 App Runner 發出的請求，提出請求的 IP 地址，提出請求的人員，提出請求的時間以及其他詳細信息。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

應用程序運行器信息 CloudTrail

CloudTrail 在您創建帳戶 AWS 帳戶 時啟用。當活動發生在 App Runner 中時，該活動會與 CloudTrail 事件歷史記錄中的其他 AWS 服務事件一起記錄在事件中。您可以檢視、搜尋和下載您的 AWS 帳戶. 如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

若要持續記錄您 AWS 帳戶的事件 (包括 App Runner 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀

- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

所有應用程序運行器操作都由記錄 CloudTrail 並記錄在 AWS App Runner API 參考中。例如，呼叫`CreateService`、`DeleteConnection`、和`StartDeployment`動作會在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

瞭解應用程式執行器記錄檔項

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單個請求，包括有關請求的操作，操作的日期和時間以及請求參數的信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範`CreateService`動作的 CloudTrail 記錄項目。

> **Note**
>
> 基於安全性考量，某些屬性值會在記錄檔中編輯，並以文字`HIDDEN_DUE_TO_SECURITY_REASONS`取代。這樣可以防止意外暴露秘密信息。但是，您仍然可以看到這些屬性在請求中傳遞或在響應中返回。

`CreateService`應用程式執行器動作的範例 CloudTrail 記錄項目

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
```

```
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/aws-user",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "aws-user"
  },
  "eventTime": "2020-10-02T23:25:33Z",
  "eventSource": "apprunner.amazonaws.com",
  "eventName": "CreateService",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/77.0.3865.75 Safari/537.36",
  "requestParameters": {
    "serviceName": "python-test",
    "sourceConfiguration": {
      "codeRepository": {
        "repositoryUrl": "https://github.com/github-user/python-hello",
        "sourceCodeVersion": {
          "type": "BRANCH",
          "value": "main"
        },
        "codeConfiguration": {
          "configurationSource": "API",
          "codeConfigurationValues": {
            "runtime": "python3",
            "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "port": "8080",
            "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
          }
        }
      },
      "autoDeploymentsEnabled": true,
      "authenticationConfiguration": {
        "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/your-
connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "healthCheckConfiguration": {
      "protocol": "HTTP"
    },
    "instanceConfiguration": {
      "cpu": "256",
```

```
      "memory": "1024"
    }
  },
  "responseElements": {
    "service": {
        "serviceName": "python-test",
        "serviceId": "dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceArn": "arn:aws:apprunner:us-east-2:123456789012:service/python-test/
dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceUrl": "generated domain",
        "createdAt": "2020-10-02T23:25:32.650Z",
        "updatedAt": "2020-10-02T23:25:32.650Z",
        "status": "OPERATION_IN_PROGRESS",
        "sourceConfiguration": {
            "codeRepository": {
                "repositoryUrl": "https://github.com/github-user/python-hello",
                "sourceCodeVersion": {
                    "type": "Branch",
                    "value": "main"
                },
                "sourceDirectory": "/",
                "codeConfiguration": {
                    "codeConfigurationValues": {
                        "configurationSource": "API",
                        "runtime": "python3",
                        "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "port": "8080",
                        "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    }
                }
            },
            "autoDeploymentsEnabled": true,
            "authenticationConfiguration": {
                "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/
your-connection/e7656250f67242d7819feade6800f59e"
            }
        },
        "healthCheckConfiguration": {
            "protocol": "HTTP",
            "path": "/",
            "interval": 5,
            "timeout": 2,
            "healthyThreshold": 3,
```

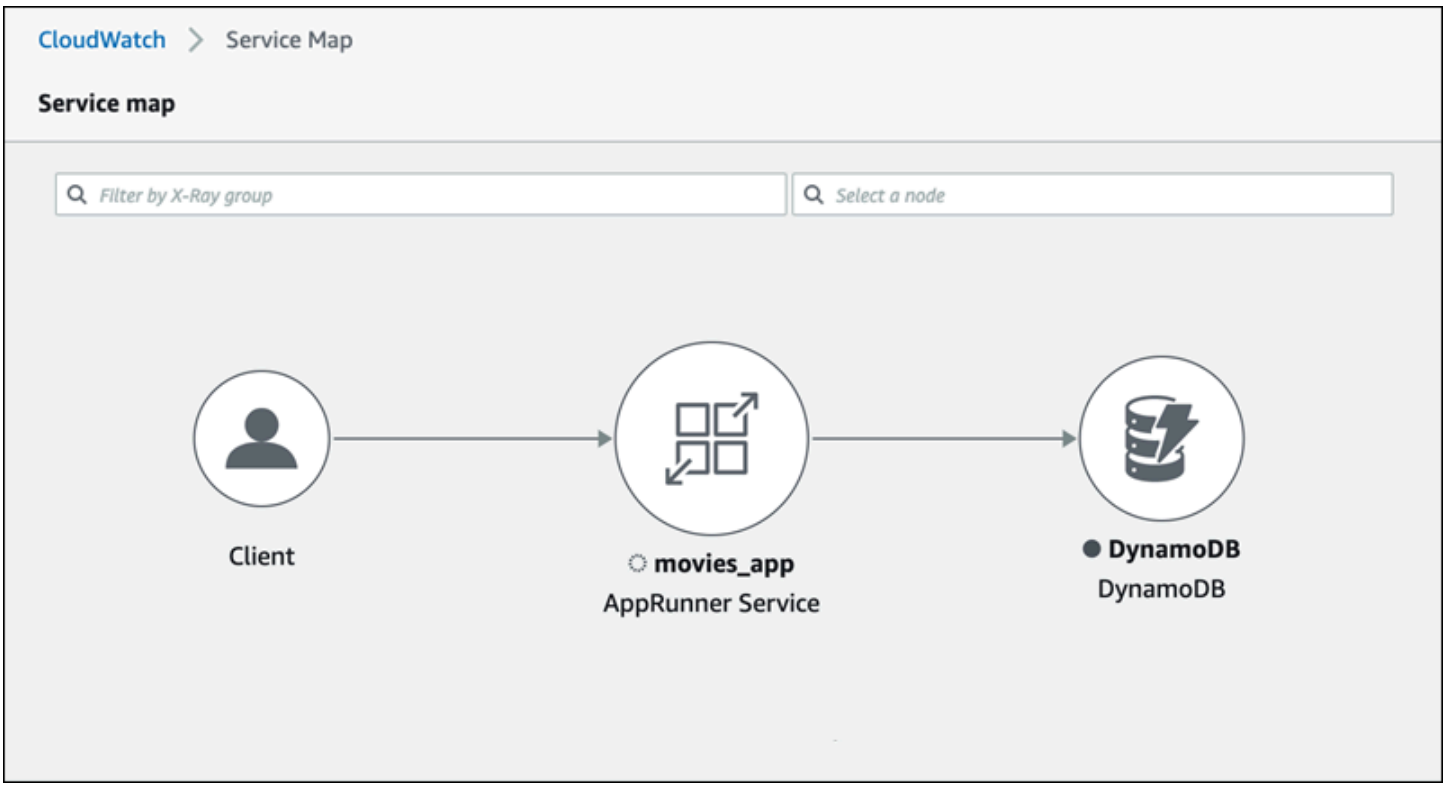
```
            "unhealthyThreshold": 5
        },
        "instanceConfiguration": {
            "cpu": "256",
            "memory": "1024"
        },
        "autoScalingConfigurationSummary": {
            "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
            "autoScalingConfigurationName": "DefaultConfiguration",
            "autoScalingConfigurationRevision": 1
        }
    }
},
  "requestID": "1a60af60-ecf5-4280-aa8f-64538319ba0a",
  "eventID": "e1a3f623-4d24-4390-a70b-bf08a0e24669",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```


使用 X-Ray 追蹤您的應用程式執行程式

AWS X-Ray 是一項服務，可收集應用程式所提供之要求的相關資料，並提供工具供您檢視、篩選和深入瞭解該資料，以識別問題和最佳化機會。對於應用程式的任何追蹤要求，您不僅可以查看要求和回應的詳細資訊，還可以查看應用程式對下游 AWS 資源、微服務、資料庫和 HTTP Web API 進行呼叫的詳細資訊。

X-Ray 會使用來自為雲端應用程式提供動力的 AWS 資源追蹤資料，以產生詳細的服務圖表。此服務圖表顯示用戶端、前端服務和後端服務，而前端服務會呼叫後端服務來處理請求和保留資料。使用服務圖形來識別瓶頸、延遲劇增的狀況和待解決的其他問題，以提升應用程式的效能。

如需 X-Ray 的詳細資訊，請參閱《AWS X-Ray 開發人員指南》。



檢測您的追蹤應用程式

檢測您的 App Runner 服務應用程式 OpenTelemetry，以便使用可攜式遙測規格進行追蹤。目前，App Runner 支援發行AWS 版 OpenTelemetry (ADOT)，這是一種使用 AWS 服務收集和呈現遙測資訊的 OpenTelemetry實作。X-Ray 實現跟踪組件。

根據您在應用程式中使用的特定 ADOT SDK，ADOT 最多支援兩種檢測方法：自動和手動。如需有關 SDK 檢測的詳細資訊，請參閱 ADOT 文件，並在導覽窗格中選擇您的 SDK。

運行時設置

以下是檢測 App Runner 服務應用程式以進行追蹤的一般執行階段設定指示。

若要設定執行階段的追蹤

1. 按照發行AWS 版 OpenTelemetry（ADOT）中為您的運行時提供的說明來檢測您的應用程序。
2. 如果您使用的是源代碼存儲庫，請在`apprunner.yaml`文件的`build`部分中安裝所需的`OTEL`依賴關係，或者如果您使用的是容器映像，則在 Dockerfile 中安裝所需的依賴關係。
3. 如果您使用的是源代碼存儲庫，請在`apprunner.yaml`文件中設置環境變量，或者如果您使用的是容器映像，則在 Dockerfile 中設置環境變量。

Example 環境變數

> **Note**
>
> 下列範例會列出要新增至`apprunner.yaml`檔案的重要環境變數。如果您使用的是容器映像，請將這些環境變量添加到 Dockerfile 中。但是，每個運行時都可以有自己的特質，您可能需要將更多的環境變量添加到以下列表中。有關運行時特定說明和如何為運行時設置應用程序的示例的更多信息，請參閱入門下的AWS Distro OpenTelemetry 並轉到您的運行時。

```
env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_EXPORTER_OTLP_ENDPOINT
      value: http://localhost:4317
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

> **Note**
>
> `OTEL_METRICS_EXPORTER=none`是應用程式執行器的重要環境變數，因為 App Runner Otel 收集器不接受指標記錄。它只接受指標跟踪。

運行時設置示例

下面的例子演示了使用 ADOT Python SDK 自動檢測您的應用程序。SDK 會自動產生包含遙測資料的跨度，描述應用程式中 Python 架構所使用的值，而不需要新增任何一行 Python 程式碼。您只需要在兩個源文件中添加或修改幾行。

首先，加入一些相依性，如下列範例所示。

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
```

```
opentelemetry-propagator-aws-xray~=1.0
```

然後，檢測您的應用程序。執行方式取決於您的服務來源 — 來源映像檔或原始程式碼。

Source image

當您的服務源是映像時，您可以直接檢測 Docker 文件，該文件控制構建容器映像並在映像中運行應用程序。下面的例子顯示了一個 Python 應用程序的檢測碼頭文件。儀器添加以粗體強調。

Example Dockerfile

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install python3.7 -y && curl -O https://bootstrap.pypa.io/get-pip.py &&
 python3 get-pip.py && yum update -y
COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
RUN opentelemetry-bootstrap --action=install
ENV OTEL_PYTHON_DISABLED_INSTRUMENTATIONS=urllib3
ENV OTEL_METRICS_EXPORTER=none
ENV OTEL_RESOURCE_ATTRIBUTES='service.name=example_app'
CMD OTEL_PROPAGATORS=xray OTEL_PYTHON_ID_GENERATOR=xray opentelemetry-instrument
 python3 app.py
EXPOSE 8080
```

Source code repository

當您的服務源是包含應用程序源的存儲庫時，您可以使用 App Runner 配置文件設置間接檢測您的映像。這些設置控制應用程序運行器生成的 Docker 文件，並用於為您的應用程序構建映像。下列範例會顯示 Python 應用程式的已檢測應用程式執行器組態檔案。儀器添加以粗體強調。

Example 阿普魯人. 羊

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
      - opentelemetry-bootstrap --action=install
run:
  command: opentelemetry-instrument python app.py
  network:
```

```
    port: 8080
  env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_PYTHON_ID_GENERATOR
      value: xray
    - name: OTEL_PYTHON_DISABLED_INSTRUMENTATIONS
      value: urllib3
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

將 X-Ray 權限新增至您的應用程式執行個體角色

若要搭配 App Runner 服務使用 X-Ray 追蹤，您必須向服務執行個體提供與 X-Ray 服務互動的權限。您可以將執行個體角色與服務建立關聯，並新增具有 X-Ray 權限的受管政策。如需有關應用程式執行個體角色的詳細資訊，請參閱the section called “實例角色”。將AWSXRayDaemonWriteAccess受管政策新增至執行個體角色，並在建立期間將其指派給您的服務。

為您的應用程式執行器服務啟用 X-Ray

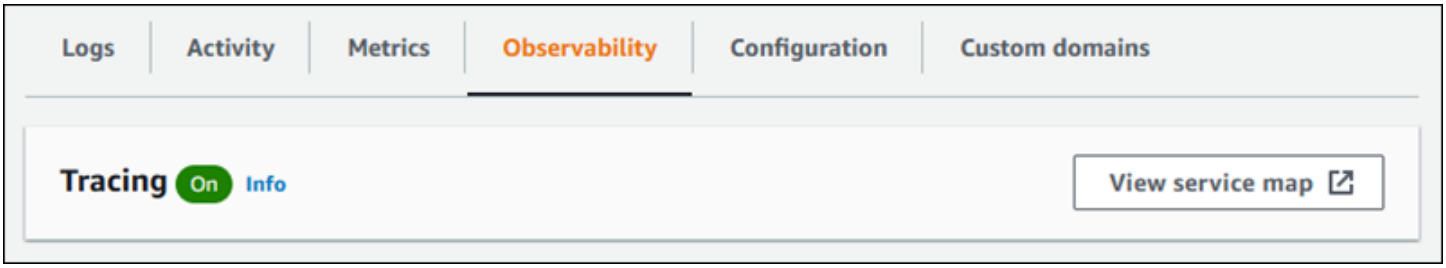
當您建立服務時，應用程式執行器預設會停用追蹤。您可以為您的服務啟用 X-Ray 追蹤，做為規劃可觀測性的一部分。如需詳細資訊，請參閱 the section called “管理可觀測性”。

如果您使用應用程式執行器 API 或 AWS CLI，則ObservabilityConfiguration資源TraceConfiguration物件中的物件會包含追蹤設定。若要停用追蹤，請勿指定TraceConfiguration物件。

在主控台和 API 案例中，請務必將上一節中討論的執行個體角色與 App Runner 服務建立關聯。

查看應用程序運行器服務的 X-Ray 跟踪數據

在 App Runner 主控台中服務儀表板頁面的「可觀察性」索引標籤上，選擇「檢視服務對應」以導覽至 Amazon CloudWatch 主控台。



使用 Amazon CloudWatch 主控台檢視應用程式所提供請求的服務對應和追蹤。服務對應會顯示要求延遲等資訊，以及與其他應用程式和 AWS 服務的互動。您新增至程式碼的自訂註解可讓您輕鬆搜尋追蹤。如需詳細資訊，請參閱 Amazon ServiceLens 使用 CloudWatch 者指南中的使用來監控應用程式的運作狀態。

將 AWS WAF Web ACL 與您的服務產生關聯

AWS WAF 是一個 Web 應用程序防火牆，可用於保護您的應用程序運行器服務。使用 AWS WAF Web 訪問控制列表（Web ACL），您可以保護您的應用程序 Runner 服務端點免受常見的 Web 漏洞攻擊和不需要的漫遊器。

Web ACL 為您提供了對應用程序運行器服務的所有傳入 Web 請求的精細控制。您可以在 Web ACL 中定義規則，以允許、封鎖或監控 Web 流量，以確保只有經過授權和合法的要求才會傳送到您的 Web 應用程式和 API。您可以根據特定的業務和安全性需求自訂 Web ACL 規則。若要進一步了解基礎設施安全性和套用網路 ACL 的最佳實務，請參閱 Amazon VPC 使用者指南中的控制網路流量。

> **Important**
>
> 與 WAF Web ACL 相關聯的應用程式執行器私人服務的來源 IP 規則不遵守 IP 型規則。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。

傳入的 Web 請求流程

當 AWS WAF Web ACL 與應用程式執行器服務相關聯時，傳入的 Web 要求會執行下列程序：

1. 應用程序運行器將原始請求的內容轉發到 AWS WAF。
2. AWS WAF 檢查請求並將其內容與您在 Web ACL 中指定的規則進行比較。
3. 根據其檢查， AWS WAF 返回`allow`或`block`響應應用程序運行器。
   - 如果傳回回`allow`應，App Runner 會將要求轉寄至您的應用程式。
   - 如果傳回回`block`應，應用程式執行器會封鎖要求，避免到達您的 Web 應用程式。它轉發從您的`block`應用程序 AWS WAF 的響應。

> **Note**
>
> 默認情況下，如果沒有響應返回，則應用程序運行器阻止請求 AWS WAF。

如需有關 AWS WAF Web ACL 的詳細資訊，請參閱AWS WAF 開發人員指南中的 Web 存取控制清單 (Web ACL)。

> **Note**
>
> 您需要支付標準 AWS WAF 定價。您不需要為應用程式執行器服務使用 AWS WAF Web ACL 產生任何額外費用。 如需有關定價的詳細資訊，請參閱AWS WAF 定價。

將 WAF 網絡 ACL 關聯到您的應用程序運行器服務

以下是將 AWS WAF Web ACL 與您的應用程式執行器服務相關聯的高階程序：

1. 在 AWS WAF 主控台中建立 Web ACL。如需詳細資訊，請參閱AWS WAF 開發人員指南中的建立 Web ACL。
2. 更新您的 AWS Identity and Access Management (IAM) 許可 AWS WAF。如需詳細資訊，請參閱許可。
3. 使用下列其中一種方法，將 Web ACL 與應用程式執行器服務產生關聯：
   - 應用程式執行器主控台：建立或更新應用程式執行器服務時，使用應用程式執行器主控台關聯現有的 Web ACL。如需指示，請參閱管理 AWS WAF 網路 ACL。
   - AWS WAF 控制台：使用控制 AWS WAF 台為現有的應用程序運行器服務關聯 Web ACL。如需詳細資訊，請參閱開發人員指南中的建立 Web ACL 與 AWS 資源的關聯或取消關聯。AWS WAF
   - AWS CLI：使用公用 API 建立網路 ACL 的 AWS WAF 關聯。如需有關 AWS WAF 公用 API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 AssociateWebACL。

考量事項

- 與 WAF Web ACL 相關聯的應用程式執行器私人服務的來源 IP 規則不遵守 IP 型規則。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。
- 一個應用程式執行器服務只能與一個 Web ACL 產生關聯。但是，您可以將一個 Web ACL 與多個應用程式執行器服務以及多個 AWS 資源相關聯。範例包括 Amazon Cognito 使用者集區和 Application Load Balancer 資源。
- 建立 Web ACL 時，在 Web ACL 完全傳播之前會經過少量時間，並且可供應用程式執行器使用。傳輸時間可以是幾秒鐘到分鐘數。 AWS WAF `WAFUnavailableEntityException`當您嘗試在網頁 ACL 完全傳播之前建立關聯時，會傳回 a。

如果您在 Web ACL 完全傳播之前重新整理瀏覽器或離開 App Runner 主控台，則關聯將無法發生。但是，您可以在應用程序運行器控制台中進行導航。

- AWS WAF 當您呼叫處於無效狀態的 App Runner 服務的下列其中一個 AWS WAF API 時，會傳回`WAFNonexistentItemException`錯誤：
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  您的應用程序運行器服務的無效狀態包括：
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`
  - `OPERATION_IN_PROGRESS`

  > **Note**
  >
  > `OPERATION_IN_PROGRESS`只有當您的應用程序運行器服務被刪除時，狀態才無效。

- 您的請求可能會導致承載大於 AWS WAF 可檢查內容的限制。有關如何 AWS WAF 處理來自 App Runner 的過大請求的詳細信息，請參閱AWS WAF 開發人員指南中的超大請求組件處理，以了解如何 AWS WAF 處理來自 App Runner 的過大請求。
- 如果您未設定適當的規則或流量模式變更，Web ACL 可能無法有效保護您的應用程式。

許可

若要使用中的 Web ACL AWS App Runner，請新增下列的 IAM 許可 AWS WAF：

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

如需 IAM 許可的詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。

以下是更新後的 IAM 政策範例 AWS WAF。此 IAM 政策包含與應用程式執行器服務搭配使用的必要許可。

Example

```
{
  {
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "wafv2:ListResourcesForWebACL",
            "wafv2:GetWebACLForResource",
            "wafv2:AssociateWebACL",
            "wafv2:DisassociateWebACL",
            "apprunner:ListAssociatedServicesForWebAcl",
            "apprunner:DescribeWebAclForService",
            "apprunner:AssociateWebAcl",
            "apprunner:DisassociateWebAcl"
         ],
         "Resource":"*"
      }
   ]
}
```

> **Note**
>
> 雖然您必須授與 IAM 許可，但列出的操作僅限許可，並且不對應於 API 操作。

管理 AWS WAF 網頁 ACL

使用下列其中一種方法管理應用程式執行 AWS WAF 器服務的 Web ACL：

- the section called “應用程式執行器”
- the section called “AWS CLI”

應用程式執行器

在 App Runner 主控台上建立服務或更新現有服務時，您可以關聯或取消關聯 AWS WAF Web ACL。

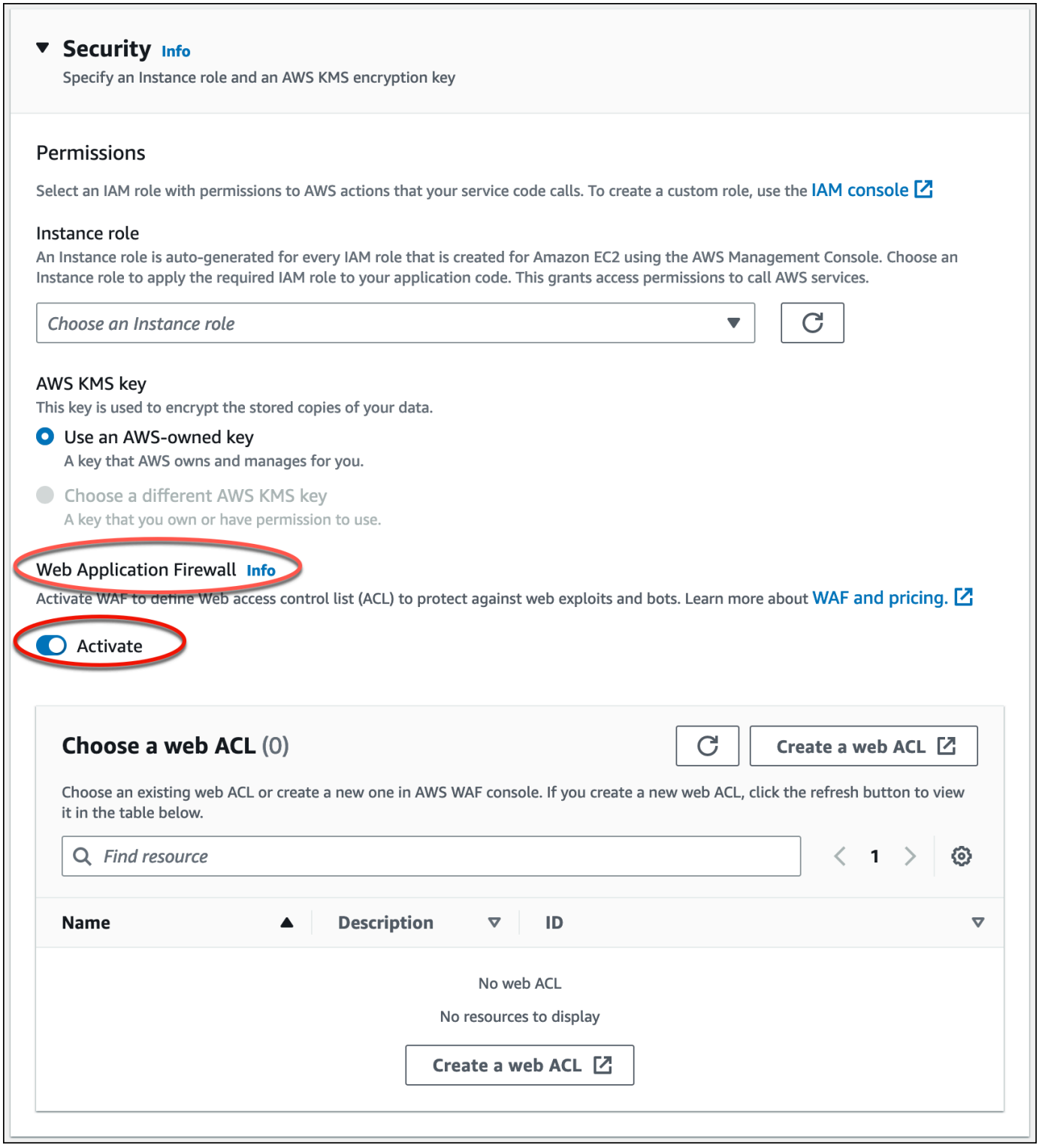
> **Note**
>
> - 應用程式執行器服務只能與一個 Web ACL 產生關聯。但是，除了其他 AWS 資源之外，您還可以將一個 Web ACL 與多個應用程式執行器服務相關聯。
> - 在建立網路 ACL 關聯之前，請務必更新的 IAM 許可 AWS WAF。如需詳細資訊，請參閱 許可。

建立 AWS WAF 網路 ACL 的關聯

> **Important**
>
> 與 WAF Web ACL 相關聯的應用程式執行器私人服務的來源 IP 規則不遵守 IP 型規則。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。

關聯 AWS WAF 網 ACL 的步驟

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 根據您要建立或更新服務，執行下列其中一個步驟：
   - 如果您要建立新服務，請選擇 [建立應用程式執行器服務]，然後移至 [設定服務]。
   - 如果您要更新現有的服務，請選擇 [設定] 索引標籤，然後選擇 [設定服務] 下的 [編輯]。
3. 轉到 Web 應用程序防火牆下的安全。
4. 選擇「啟用」切換按鈕以檢視選項。



5. 執行下列步驟之一：

   - 若要建立現有 Web ACL 的關聯：請從「選擇 Web ACL」表格中選擇所需的 Web ACL，以與您的應用程式執行器服務建立關聯。

- 若要建立新的 Web ACL：選擇「建立 Web ACL」以使用 AWS WAF 主控台建立新的 Web ACL。如需詳細資訊，請參閱AWS WAF 開發人員指南中的建立 Web ACL。
  1. 選擇重新整理按鈕，在「選擇 Web ACL」表格中檢視新建立的 Web ACL。
  2. 選取所需的腹板 ACL。

6. 如果您要建立新服務，請選擇 [下一步]；如果您要更新現有的服務，請選擇 [儲存變更]。選取的 Web ACL 會與您的應用程式執行器服務相關聯。
7. 若要驗證 Web ACL 關聯，請選擇服務的「組態」索引標籤，然後前往「設定服務」。捲動至 [安全性] 下的 [Web 應用程式防火牆]，以檢視與服務相關聯之 Web ACL 的詳細資料。

> **Note**
>
> 建立 Web ACL 時，在 Web ACL 完全傳播之前會經過少量時間，並且可供應用程式執行器使用。傳輸時間可以是幾秒鐘到分鐘數。 AWS WAF `WAFUnavailableEntityException`當您嘗試在網頁 ACL 完全傳播之前建立關聯時，會傳回 a。
> 如果您在 Web ACL 完全傳播之前重新整理瀏覽器或離開 App Runner 主控台，則關聯將無法發生。但是，您可以在應用程序運行器控制台中進行導航。

取消網路 ACL 的關聯 AWS WAF

您可以通過更新應用程序運行器服務來取消不再需要的 AWS WAF Web ACL 的關聯。

取消網頁 ACL 的關聯 AWS WAF 的步驟

1. 開啟應用程式執行器主控台，然後在 [區域] 清單中選取您的 AWS 區域.
2. 轉到您要更新的服務的配置選項卡，然後選擇配置服務下的編輯。
3. 轉到 Web 應用程序防火牆下的安全。
4. 停用「啟動」切換按鈕。您會收到確認刪除的訊息。
5. 選擇確認。Web ACL 與您的應用程式執行器服務取消關聯。

> **Note**
>
> - 如果您要將您的服務與其他 Web ACL 相關聯，請從「選擇 Web ACL」表格中選取 Web ACL。應用程式執行器會取消目前 Web ACL 的關聯，並啟動與選取的 Web ACL 相關聯的程序。

- 如果沒有其他應用程式執行器服務或資源使用取消關聯的 Web ACL，請考慮刪除 Web ACL。否則，您將繼續產生費用。如需定價的詳細資訊，請參閱AWS WAF 定價。有關如何刪除網頁 ACL 的說明，請參閱 AWS WAF API 參考資料中的 DeleteWebACL。
- 您無法刪除與其他作用中應用程式執行器服務或其他資源相關聯的 Web ACL。

AWS CLI

您可以使用 AWS WAF 公用 API 來關聯或取消關聯 AWS WAF Web ACL。您要與 Web ACL 建立關聯或取消關聯的應用程式執行器服務必須處於有效狀態。

AWS WAF 當您呼叫處於無效狀態的 App Runner 服務的下列其中一個 AWS WAF API 時，會傳回`WAFNonexistentItemException`錯誤：

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

您的應用程序運行器服務的無效狀態包括：

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`只有當您的應用程序運行器服務被刪除時，狀態才無效。

如需有關 AWS WAF 公用 API 的詳細資訊，請參閱 AWS WAF API 參考指南。

> **Note**
>
> 更新您的 AWS WAF. 如需詳細資訊，請參閱 許可。

使用以下方式關聯 AWS WAF 網 ACL AWS CLI

> **Important**
>
> 與 WAF Web ACL 相關聯的應用程式執行器私人服務的來源 IP 規則不遵守 IP 型規則。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。

關聯 AWS WAF 網 ACL 的步驟

1. 使用您偏好的一 AWS WAF 組規則動作`Allow`或對服務的 Web 請求，為您`Block`的服務建立 Web ACL。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 CreateWebACL。

   Example 建立網路 ACL-請求

   ```
   aws wafv2
   create-web-acl
   --region <region>
   --name <web-acl-name>
   --scope REGIONAL
   --default-action Allow={}
   --visibility-config <file-name.json>
   # This is the file containing the WAF web ACL rules.
   ```

2. 使用`associate-web-acl` AWS WAF 公用 API 將您建立的 Web ACL 與應用程式執行器服務建立關聯。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 AssociateWebACL。

   > **Note**
   >
   > 建立 Web ACL 時，在 Web ACL 完全傳播之前會經過少量時間，並且可供應用程式執行器使用。傳輸時間可以是幾秒鐘到分鐘數。 AWS WAF `WAFUnavailableEntityException`當您嘗試在網頁 ACL 完全傳播之前建立關聯時，會傳回 a。
   >
   > 如果您在 Web ACL 完全傳播之前重新整理瀏覽器或離開 App Runner 主控台，則關聯將無法發生。但是，您可以在應用程序運行器控制台中進行導航。

Example 建立網路 ACL-請求的關聯

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. 使用`get-web-acl-for-resource` AWS WAF 公共 API 驗證 Web ACL 與您的應用程序運行器服務相關聯。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南ForResource中的 GetWebACL。

   Example 驗證資源的 Web ACL-請求

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   如果沒有與您的服務相關聯的 Web ACL，您會收到空白回應。

使用刪除 AWS WAF 網路 ACL AWS CLI

如果 AWS WAF Web ACL 與應用程式執行器服務相關聯，則無法刪除該 ACL。

刪除 AWS WAF 網路 ACL 的步驟

1. 使用`disassociate-web-acl` AWS WAF 公共 API 取消 Web ACL 與應用程序運行器服務的關聯。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 DisassociateWebACL。

   Example 取消網路 ACL-請求的關聯

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. 使用`get-web-acl-for-resource` AWS WAF 公共 API 確認 Web ACL 與您的應用程式執行器服務斷開關聯。

Example 確認網頁 ACL 已取消關聯-請求

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

解除關聯的 Web ACL 不會列出您的應用程式執行器服務。如果沒有與您的服務相關聯的 Web ACL，您會收到空白回應。

3. 使用`delete-web-acl` AWS WAF 公共 API 刪除取消關聯的 Web ACL。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 DeleteWebACL。

   Example 刪除網路 ACL-請求

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   --id <web_acl_id>
   --lock-token <web_acl_lock_token>
   --region <region>
   ```

4. 確認已使用`list-web-acl` AWS WAF 公用 API 刪除網頁 ACL。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 ListWebACL。

   Example 確認已刪除網頁 ACL-請求

   ```
   aws wafv2 list-web-acls
   --scope REGIONAL
   --region <region>
   ```

   刪除的網 ACL 將不再列示出來。

   > **Note**
   >
   > 如果 Web ACL 與其他作用中的應用程式執行器服務或其他資源 (例如 Amazon Cognito 使用者集區) 相關聯，則無法刪除 Web ACL。

列出與 Web ACL 相關聯的應用程式執行器服務

一個 Web ACL 可以與多個應用程序運行器服務和其他資源相關聯。使用`list-resources-for-web-acl` AWS WAF 公共 API 列出與 Web ACL 關聯的應用程序運行器服務。如需 AWS WAF API 的詳細資訊，請參閱 AWS WAF API 參考指南中的 ListResourcesForWebACL。

Example 列出與 Web ACL 關聯的應用程序運行器服務-請求

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example 列出與 Web ACL 關聯的應用程序運行器服務-響應

下列範例說明沒有與 Web ACL 相關聯的應用程式執行器服務時的回應。

```
{
  "ResourceArns": []
}
```

Example 列出與 Web ACL 關聯的應用程序運行器服務-響應

下列範例說明存在與 Web ACL 相關聯的應用程式執行器服務時的回應。

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

測試和記錄 AWS WAF 網頁 ACL

當您在 Web ACL 中將規則動作設定為「計數」時，會將要求 AWS WAF 新增至符合規則的要求計數。若要使用 App Runner 服務測試 Web ACL，請將規則動作設定為 [計數]，並考量符合每個規則的要求數量。例如，您可以為符合您確定為一般使用者流量的大量請求的`Block`動作設定規則。在這種情況下，您可能需要重新配置規則。如需詳細資訊，請參閱AWS WAF 開發人員指南中的測試和調整您的 AWS WAF 保護。

您也可以設定 AWS WAF 將請求標頭記錄到 Amazon CloudWatch 日誌日誌群組、亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體或 Amazon 資料 Firehose。如需詳細資訊，請參閱《AWS WAF 開發人員指南》中的記錄 Web ACL 流量。

若要存取與應用程式執行器服務相關聯的 Web ACL 相關的記錄檔，請參閱下列記錄欄位：

- `httpSourceName`：包含 `APPRUNNER`
- `httpSourceId`：包含 `customeraccountid-apprunnerserviceid`

如需詳細資訊，請參閱AWS WAF 開發人員指南中的記錄範例。

> **Important**
>
> 與 WAF Web ACL 相關聯的應用程式執行器私人服務的來源 IP 規則不遵守 IP 型規則。這是因為我們目前不支援將要求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私人服務。如果您的應用程式執行器應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須針對私有端點使用安全群組規則，而不是 WAF Web ACL。

使用配置文件設置應用程序運行器服務選項

> **Note**
>
> 組態檔案僅適用於以原始程式碼為基礎的服務。您無法將組態檔與影像式服務搭配使用。

當您使用原始程式碼儲存庫建立 AWS App Runner 服務時， AWS App Runner 需要建置和啟動服務的相關資訊。您可以在每次使用 App Runner 主控台或 API 建立服務時提供此資訊。或者，您也可以使用組態檔來設定服務選項。您在檔案中指定的選項會成為來源儲存庫的一部分，對這些選項所做的任何變更的追蹤方式與追蹤原始程式碼變更的方式類似。您可以使用應用程式執行器設定檔來指定比 API 支援更多的選項。如果您只需要 API 支援的基本選項，則不需要提供設定檔。

應用程序運行器配置文件是在應用程序存儲庫的源目錄`apprunner.yaml`中命名的 YAML 文件。它為您的服務提供構建和運行時選項。此檔案中的值會指示 App Runner 如何建置和啟動服務，並提供執行階段內容，例如網路設定和環境變數。

應用程序運行器配置文件不包括操作設置，例如 CPU 和內存。

如需應用程式執行器組態檔案的範例，請參閱the section called “範例”。如需完整的參考指南，請參閱the section called “參考資料”。

主題

- 應用程序運行器配置文件
- 應用程序運行器配置文件

應用程序運行器配置文件

> **Note**
>
> 組態檔案僅適用於以原始碼為基礎的服務。您無法將組態檔與影像式服務搭配使用。

下面的例子演示了 AWS App Runner 配置文件。有些是最小的，只包含所需的設置。其他則完整，包括所有組態檔案區段。如需應用程式執行器組態檔案的概觀，請參閱〈〉應用程序運行器配置。

組態檔案範例

最小配置文件

使用最小的配置文件，App Runner 進行了以下假設：

- 在建置或執行期間不需要自訂環境變數。
- 使用最新的執行階段版本。
- 使用預設的連接埠號碼和連接埠環境變數。

Example 阿普鲁人. 亚姆尔

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

完整的配置文件

此範例顯示使用`apprunner.yaml`原始格式與受管理執行階段的所有組態金鑰。

Example 阿普鲁人. 亚姆尔

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
```

```
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

完整的配置文件-（使用修訂版本）

此範例顯示使用託管執行階段中的`apprunner.yaml`所有組態金鑰。

該`pre-run`參數僅由修訂後的應用程序運行器構建支持。如果您的應用程式使用原始 App Runner 組建支援的執行階段版本，請勿在設定檔中插入此參數。如需詳細資訊，請參閱 [託管運行時版本和應用程序運行器構建]。

> **Note**
>
> 由於這個例子是針對 Python 3.11 的，所以我們使用`pip3`和`python3`命令。如需詳細資訊，請參閱 Python 平台主題[特定執行階段版本的編號說明]中的。

Example 阿普魯人. 亚姆尔

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
```

```
      - wget -c https://s3.amazonaws.com/DOC-EXAMPLE-BUCKET/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

如需特定受管理執行階段組態檔案的範例，請參閱下基於代碼的服務的特定執行階段子主題。

應用程序運行器配置文件

> **Note**
>
> 組態檔案僅適用於以原始程式碼為基礎的服務。您無法將組態檔與影像式服務搭配使用。

本主題是 AWS App Runner 組態檔之語法與語意的完整參考指南。如需應用程式執行器組態檔案的概觀，請參閱〈〉應用程序運行器配置。

應用程序運行器配置文件是一個 YAML 文件。命名它`apprunner.yaml`，並將其放置在應用程序存儲庫的源目錄中。

結構概述

應用程序運行器配置文件是一個 YAML 文件。命名它`apprunner.yaml`，並將其放置在應用程序存儲庫的源目錄中。

應用程序運行器配置文件包含以下主要部分：

- 頂部-包含頂級密鑰
- 構建部分-配置構建階段
- 執行區段 — 設定執行階段

頂部

檔案頂端的金鑰會提供有關檔案和服務執行階段的一般資訊。以下是可用的按鍵：

- `version`— 必要。應用程序運行器配置文件版本。理想情況下，請使用最新版本。

  語法

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`— 必要。應用程式使用的執行階段名稱。若要瞭解 App Runner 提供的不同程式設計平台的可用執行階段，請參閱基於代碼的服務。

  > **Note**
  >
  > 託管運行時的命名約定是。 <language-name><major-version>

語法

```
runtime: runtime-name
```

Example

```
runtime: python3
```

構建部分

構建部分配置應用程序運行器服務部署的構建階段。您可以指定建置命令和環境變數。構建命令是必需的。

此區段以build:索引鍵開頭，並具有下列子機碼：

- commands— 必要。指定應用程式執行器在各種建置階段執行的命令。包括下列子機碼：
  - pre-build-可選。應用程式執行程式在建置之前執行的命令。例如，安裝npm相依性或測試程式庫。
  - build— 必要。應用程序運行器運行來構建您的應用程序的命令。例如，使用pipenv.
  - post-build-可選。應用程式執行程式在建置之後執行的命令。例如，使用 Maven 將構建成品打包到 JAR 或 WAR 文件中，或運行測試。

語法

```
build:
  commands:
    pre-build:
      - command
      - …
    build:
      - command
      - …
    post-build:
      - command
      - …
```

Example

```
build:
  commands:
    pre-build:
      - yum install openssl
    build:
      - pip install -r requirements.txt
    post-build:
      - python manage.py test
```

- env-可選。指定建置階段的自訂環境變數。定義為名稱-值純量映射。您可以在構建命令中按名稱引用這些變量。

> **Note**
>
> 此組態檔案中的兩個不同位置有兩個不同的env項目。其中一組位於 [建置] 區段中，另一組位於 [執行] 區段中。
>
> - 建env置程序期間，「建置」(Bu ild) 區段中的集合可由pre-buildbuildpost-build、、和pre-run指令參考。
>
> 重要-請注意，這些pre-run命令位於此文件的「運行」部分中，即使它們只能訪問「構建」部分中定義的環境變量。
>
> - 執行階段環境中的run命令可以參考 [執行] 區段中的env集合。

語法

```
build:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
    - …
```

Example

```
build:
```

```
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
```

執行區段

運行部分配置應用程序運行應用程序部署的容器運行階段。您可以指定執行階段版本、執行前指令 (僅限修訂格式)、啟動指令、網路連接埠和環境變數。

此區段以`run:`索引鍵開頭，並具有下列子機碼：

- `runtime-version`-可選。指定您想要鎖定您的應用程式執行階段服務的執行階段版本。

  依預設，只會鎖定主要版本。App Runner 會在每次部署或服務更新時使用執行階段可用的最新次要版本和修補程式版本。如果您指定主要版本和次要版本，則兩者都會被鎖定，而 App Runner 僅更新修補程式版本。如果您指定主要、次要版本和修補程式版本，您的服務會鎖定在特定的執行階段版本上，而 App Runner 絕不會對其進行更新。

  語法

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  >
  > 某些平台的執行階段具有不同的版本元件。如需詳細資訊，請參閱特定的平台

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run`-可選。 僅限已修訂的組建使用。指定將應用程式從組建映像複製到執行映像後，應用程式執行的命令。您可以在此輸入指令來修改目`/app`錄外的執行影像。例如，如果您需要安裝位於`/`

app目錄之外的其他全域相依性，請在此子區段中輸入必要的指令來執行此作業。如需有關應用程式執行器建置程序的詳細資訊，請參閱[託管運行時版本和應用程序運行器構建](#)。

> **Note**
>
> - 重要 — 即使pre-run命令列在 [執行] 區段中，它們也只能參考此組態檔案之 [Build] 區段中定義的環境變數。它們無法參考此「執行」區段中定義的環境變數。
> - 該pre-run參數僅由修訂後的應用程序運行器構建支持。如果您的應用程式使用原始 App Runner 組建支援的執行階段版本，請勿在設定檔中插入此參數。如需詳細資訊，請參閱[託管運行時版本和應用程序運行器構建](#)。

語法

```
run:
  pre-run:
      - command
      - …
```

- command— 必要。App Runner 在完成應用程式建置之後用來執行應用程式的命令。

  語法

  ```
  run:
    command: command
  ```

- network-可選。指定應用程式偵聽的連接埠。其包括以下內容：
  - port-可選。如果有指定，這是您的應用程式偵聽的連接埠號碼。預設值為 8080。
  - env-可選。如果指定，App Runner 會將連接埠號碼傳遞給此環境變數中的容器，以及 (而不是) 在預設環境變數中傳遞相同的連接埠號碼PORT。換句話說，如果您指定env，App Runner 會在兩個環境變數中傳遞連接埠號碼。

  語法

  ```
  run:
    network:
      port: port-number
      env: env-variable-name
  ```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- env-可選。定義執行階段的自訂環境變數。定義為名稱-值純量映射。您可以在執行階段環境中依名稱參考這些變數。

> **Note**
>
> 此組態檔案中的兩個不同位置有兩個不同的env項目。其中一組位於 [建置] 區段中，另一組位於 [執行] 區段中。
>
> - 建env置程序期間，「建置」(Bu ild) 區段中的集合可由pre-buildbuildpost-build、、和pre-run指令參考。
>
> 重要-請注意，這些pre-run命令位於此文件的「運行」部分中，即使它們只能訪問「構建」部分中定義的環境變量。
>
> - 執行階段環境中的run命令可以參考 [執行] 區段中的env集合。

語法

```
run:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
  secrets:
    - name: name1
      value-from: arn:aws:secretsmanager:region:aws_account_id:secret:secret-id
    - name: name2
      value-from: arn:aws:ssm:region:aws_account_id:parameter/parameter-name
    - …
```

Example

```
run:
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-
S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

應用程序運行器 API

AWS App Runner 應用程序編程接口（API）是一個 RESTful API，用於向應用程序運行器服務發出請求。您可以使用 API 來建立、列出、描述、更新和刪除 AWS 帳戶.

您可以直接在應用程式程式碼中呼叫 API，也可以使用其中一個 AWS SDK。

如需完整的 API 參考資訊，請參閱 AWS App Runner API 參考資料。

如需 AWS 開發人員工具的詳細資訊，請參閱要建置的工具 AWS。

主題

- 使用與應用 AWS CLI 程式執行程式搭配使用
- 使用 AWS CloudShell 來使用 AWS App Runner

使用與應用 AWS CLI 程式執行程式搭配使用

對於命令列指令碼，請使用AWS CLI來呼叫應用程式執行器服務。如需完整的 AWS CLI 參考資訊，請參閱《指令參考》中的AWS CLI apprunner。

AWS CloudShell 可讓您略過 AWS CLI 在開發環境中安裝，並在中使用它。 AWS Management Console 除了避免安裝之外，您也不需要配置憑據，也不需要指定區域。您的 AWS Management Console 工作階段會將此內容提供給 AWS CLI. 如需使用範例的詳 CloudShell細資訊，請參閱〈〉the section called “使用 AWS CloudShell”。

使用 AWS CloudShell 來使用 AWS App Runner

AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console 您可以使用偏好的 shell（Bash PowerShell 或 Z 外殼 AWS App Runner）對 AWS 服務（包括）運行 AWS CLI 命令。另外，您無需下載或安裝命令列工具即可執行此操作。

您可以AWS CloudShell 從啟動 AWS Management Console，並且您用來登入主控台的 AWS 認證會自動在新的 shell 工作階段中使用。這種使用 AWS CloudShell 者的預先驗證可讓您在使用 AWS CLI 版本 2 (預先安裝在殼層的計算環境中) 與 App Runner 等 AWS 服務互動時略過設定認證。

主題

- 取得的 IAM 許可 AWS CloudShell
- 與應用程序運行器交互使 AWS CloudShell

- 使用驗證應用程序運行器服務 AWS CloudShell

取得的 IAM 許可 AWS CloudShell

管理員可以使用提供的存取管理資源將許可授予 IAM 使用者 AWS Identity and Access Management，以便他們可以存取 AWS CloudShell 和使用環境的功能。

系統管理員授與使用者存取權的最快方法是透過 AWS 受管理的原則。AWS 受管政策是由 AWS建立並管理的獨立政策。的下列 AWS 受管政策 CloudShell 可附加至 IAM 身分：

- `AWSCloudShellFullAccess`：授予使用權限 AWS CloudShell 以完全訪問所有功能。

如果您想要限制 IAM 使用者可以執行的動作範圍 AWS CloudShell，您可以建立使用`AWSCloudShellFullAccess`受管政策做為範本的自訂政策。如需有關限制中使用者可使用的動作的詳細資訊 CloudShell，請參閱《使用AWS CloudShell 者指南》中的「使用 IAM 政策管理 AWS CloudShell 存取和使用」。

> **Note**
>
> 您的 IAM 身分還需要一項政策，該政策授予與應用程序 Runner 進行呼叫的權限。如需詳細資訊，請參閱 the section called “應用程式執行器和 IAM”。

與應用程序運行器交互使 AWS CloudShell

AWS CloudShell 從啟動之後 AWS Management Console，您可以立即開始使用指令行介面與 App Runner 互動。

在下列範例中，您會使用 in 擷取其 AWS CLI 中一個 App Runner 服務的相關資訊 CloudShell。

> **Note**
>
> AWS CLI 在中使用時 AWS CloudShell，您無需下載或安裝任何其他資源。此外，因為您已經在 Shell 中驗證身分，因此無需設定憑證即可呼叫。

Example 使用檢索應用運行器服務信息 AWS CloudShell

1. 從中 AWS Management Console，您可以選擇 CloudShell 導覽列上的下列可用選項來啟動：

- 選擇圖 CloudShell 示。
- 開始**cloudshell**在搜尋方塊中輸入內容，然後在搜尋結果中看到該CloudShell選項時選擇該選項。

2. 要在控制台會話的 AWS 區域中列出您 AWS 帳戶中的所有當前應用程序 Runner 服務，請在命令行中輸入以下 CloudShell命令：

```
$ aws apprunner list-services
```

輸出會列出您服務的摘要資訊。

```
{
  "ServiceSummaryList": [
    {
      "ServiceName": "my-app-1",
      "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-20T19:05:25Z",
      "UpdatedAt": "2020-11-23T12:41:37Z",
      "Status": "RUNNING"
    },
    {
      "ServiceName": "my-app-2",
      "ServiceId": "ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-app-2/
ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceUrl": "e2m8rrrx33.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-06T23:15:30Z",
      "UpdatedAt": "2020-11-23T13:21:22Z",
      "Status": "RUNNING"
    }
  ]
}
```

3. 若要取得特定 App Runner 服務的詳細說明，請使用上一個步驟擷取的其中一個 ARN，在 CloudShell 命令列中輸入下列命令：

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

輸出會列出您所指定服務的詳細說明。

```
{
  "Service": {
    "ServiceName": "my-app-1",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-23T12:41:37Z",
    "Status": "RUNNING",
    "SourceConfiguration": {
      "CodeRepository": {
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        },
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "[pip install -r requirements.txt]",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "API"
        }
      },
      "AutoDeploymentsEnabled": true,
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/my-
github-connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
```

```
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 10,
      "Timeout": 5,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    }
  }
}
```

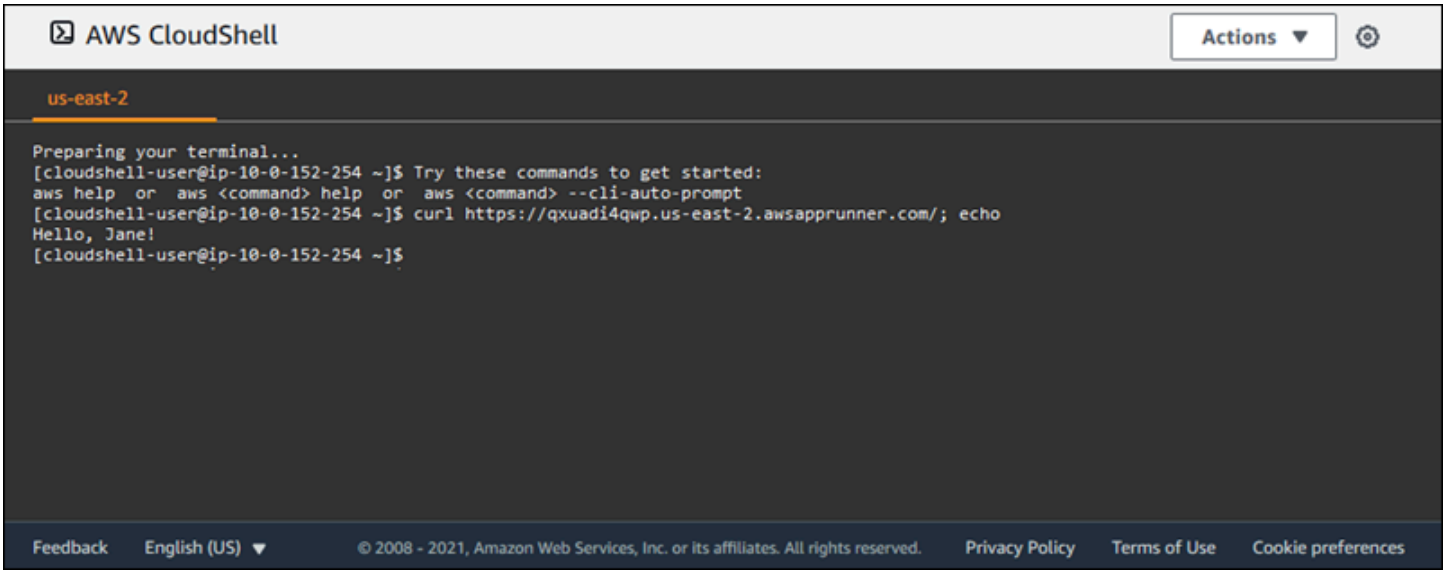
使用驗證應用程序運行器服務 AWS CloudShell

當您創建 App Runner 服務時，App Runner 會為您的服務的網站創建一個默認域，並將其顯示在控制台中（或將其返回在 API 調用結果中）。您可以使用撥 CloudShell 打電話到您的網站，並驗證它是否正常運作。

例如，如中所述建立 App Runner 服務之後開始使用，請在中執行下列命令 CloudShell：

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

輸出應顯示預期的頁面內容。


AWS CloudShell
Actions
us-east-2
Preparing your terminal...
[cloudshell-user@ip-10-0-152-254 ~]$ Try these commands to get started:
aws help or aws <command> help or aws <command> --cli-auto-prompt
[cloudshell-user@ip-10-0-152-254 ~]$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
Hello, Jane!
[cloudshell-user@ip-10-0-152-254 ~]$
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

疑難排解

本章針對您在使用 AWS App Runner 服務時可能遇到的常見錯誤和問題提供疑難排解步驟。錯誤訊息可能會出現在服務頁面的主控台、API 或 [記錄] 索引標籤上。

如需更多故障診段建議和常見支援問題的解答，請瀏覽 知識中心。

主題

- 服務無法建立時
- 自訂網域名稱
- HTTP/HTTPS 請求路由錯誤
- 當服務無法連接到 Amazon RDS 或下游服務

服務無法建立時

如果您嘗試建立 App Runner 服務失敗，服務就會進入CREATE_FAILED狀態。此狀態會在主控台上顯示為「建立失敗」。服務可能因為與下列一或多項相關的問題而無法建立：

- 您的應用程式碼
- 構建過程
- 組態
- 資源配額
- 您的服務使用 AWS 服務 的基礎暫時性問題

若要疑難排解無法建立的服務，建議您執行下列動作。

1. 閱讀服務事件和記錄檔，瞭解造成服務無法建立的原因。
2. 對程式碼或組態進行任何必要的變更。
3. 如果您已達到服務配額，請刪除一或多個服務。
4. 如果您達到其他資源配額，則可以在可調整的情況下增加配額。
5. 完成上述所有步驟後，請再次嘗試重建服務。如需有關如何重建服務的資訊，請參閱the section called “重建失敗的服務”。

> **Note**
>
> 可能導致問題的可調整資源配額之一是 Fargate 隨需 vCPU 資源。
> vCPU 資源計數會決定應用程式執行器可提供給服務的執行個體數目。這是常駐在服務中的 Fargate 隨選 vCPU 資源計數的可調整配額值。 AWS Fargate (Fargate) 若要檢視您帳戶的 vCPU 配額設定或要求提高配額，請使用中的 Service Quotas 主控台。 AWS Management Console如需詳細資訊，請參閱 Amazon 彈性容器AWS Fargate 服務開發人員指南中的服務配額。

> **Important**
>
> 除了針對失敗服務的初始建立嘗試外，您不會產生任何額外費用。即使失敗的服務無法使用，它仍然會計入您的服務配額中。App Runner 不會自動刪除失敗的服務，因此請確保在完成失敗分析後將其刪除。

自訂網域名稱

本節說明如何疑難排解和解決連結至自訂網域時可能會遇到的各種錯誤。

> **Note**
>
> 為了增強應用程式執行器應用程式的安全性，*.awsapprunner.com 網域註冊在公用尾碼清單 (PSL) 中。為了進一步的安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

取得自訂網域的建立失敗錯誤

- 檢查此錯誤是否因為 CAA 記錄有問題。如果 DNS 樹狀結構中的任何位置沒有 CAA 記錄，您會收到訊息`fail open`，並核 AWS Certificate Manager 發憑證以驗證自訂網域。這允許應用程序運行器接受自定義域。如果您在 DNS 記錄中使用 CAA 認證，請確定至少包含一個網域的 CAA 記錄`amazon.com`。否則，ACM 將無法發行憑證。因此，無法建立應用程式執行程式的自訂網域。

下列範例會使用 DNS 查閱工具 DiG 來顯示缺少必要項目的 CAA 記錄。此範例使用`example.com`做為自訂網域。在範例中執行下列命令以檢查 CAA 記錄。

```
...
;; QUESTION SECTION:
;example.com.            IN  CAA

;; ANSWER SECTION:
example.com.        7200    IN  CAA 0 iodef "mailto:hostmaster@example.com"
example.com.        7200    IN  CAA 0 issue "letsencrypt.org"
...note absence of "amazon.com" in any of the above CAA records...
```

- 更正網域記錄，並確定至少包含一個 CAA 記錄`amazon.com`。
- 重試將自訂網域與應用程式執行器連結。

如需如何解決 CAA 錯誤的指示，請參閱下列內容：

- 憑證授權單位授權 (CAA) 問題
- 如何解決發行或更新 ACM 憑證時的 CAA 錯誤？

取得自訂網域的 DNS 憑證驗證擱置錯誤

- 檢查您是否跳過了自定義域設置中的重要步驟。此外，請檢查您是否使用 DNS 查閱工具 (例如 DiG) 錯誤地設定了 DNS 記錄。特別是，檢查以下錯誤：
  - 任何錯過的步驟。
  - 不支援的字元，例如 DNS 記錄中的雙引號。
- 糾正錯誤。
- 重試將自訂網域與應用程式執行器連結。

如需如何解決 CAA 驗證錯誤的指示，請參閱下列內容。

- DNS 驗證
- the section called “自訂網域名稱”

基本的疑難排解

- 確認可以找到服務。

```
aws apprunner list-services
```

- 描述服務並檢查其狀態。

```
aws apprunner describe-service --service-arn
```

- 檢查自訂網域的狀態。

```
aws apprunner describe-custom-domains --service-arn
```

- 列出所有進行中的作業。

```
aws apprunner list-operations --service-arn
```

自訂網域憑證續約

當您將自訂網域新增至服務時，App Runner 會提供您新增至 DNS 伺服器的一組 CNAME 記錄。這些 CNAME 記錄包括憑證記錄。應用程序運行器使用 AWS Certificate Manager （ACM）來驗證域。應用程式執行器會驗證這些 DNS 記錄，以確保此網域的持續擁有權。如果您從 DNS 區域移除 CNAME 記錄，App Runner 將無法再驗證 DNS 記錄，且自訂網域憑證無法自動續約。

本節說明如何解決下列自訂網域憑證續約問題：

- the section called “CNAME 會從 DNS 伺服器中移除”.
- the section called “憑證已過期”.

CNAME 會從 DNS 伺服器中移除

- 使用 DescribeCustomDomainsAPI 或應用程式執行器主控台中的「自訂網域」設定擷取您的 CNAME 記錄。如需有關已儲存 CNAME 的資訊，請參閱CertificateValidationRecords。
- 將憑證驗證 CNAME 記錄新增至您的 DNS 伺服器。然後，應用程序運行器可以驗證您是否擁有該域。新增 CNAME 記錄之後，DNS 記錄最多可能需要 30 分鐘才能傳播。應用程式執行器和 ACM 也可能需要數小時才能重試憑證更新程序。如需如何新增 CNAME 記錄的指示，請參閱the section called “管理自訂網域”。

憑證已過期

- 取消關聯（取消鏈接），然後使用應用程序運行器控制台或 API 關聯（鏈接）應用程序運行器服務的自定義域。應用程式執行器會建立新的憑證驗證 CNAME 記錄。
- 將新的憑證驗證 CNAME 記錄新增至您的 DNS 伺服器。

  如需如何取消關聯 (取消連結) 與關聯 (連結) 自訂網域的指示，請參閱。the section called “管理自訂網域”

如何驗證證書是否已成功更新

您可以檢查憑證記錄的狀態，以確認您的憑證是否已順利續約。您可以使用 curl 等工具來檢查憑證的狀態。

如需有關憑證續約的詳細資訊，請參閱下列連結：

- 為什麼我的 ACM 證書被標記為不符合續訂資格？
- ACM 憑證的受管理續約
- DNS 驗證

HTTP/HTTPS 請求路由錯誤

本節說明如何疑難排解和解決將 HTTP/HTTPS 流量路由到應用程式執行器服務端點時可能遇到的錯誤。

404 發送 HTTP/HTTPS 流量到應用程序運行器服務端點時未找到錯誤

- 確認指向 HTTP 要求中的服務 URL，因為 App Runner 會使用主機標頭資訊來路由要求。`Host Header`大多數客戶端（例如cURL和 Web 瀏覽器）會自動將主機頭指向服務 URL。如果您的用戶端未將服務 URL 設定為`Host Header`，表示您會收到`404 Not Found`錯誤訊息。

  Example 主機標頭不正確

  ```
  $ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
  HTTP/1.1 404 Not Found
  transfer-encoding: chunked
  ```

  Example 正確的主機標頭

  ```
  $ ~ curl -I -H "host: testservice.awsapprunner.com" https://
  testservice.awsapprunner.com/
  HTTP/1.1 200 OK
  content-length: 11772
  content-type: text/html; charset=utf-8
  ```

- 確認您的用戶端是否正確設定要求路由到公用或私人服務的伺服器名稱指示器 (SNI)。對於 TLS 終止和請求路由，應用程序運行器使用 HTTPS 連接中設置的 SNI。

當服務無法連接到 Amazon RDS 或下游服務

如果您的服務無法連線至 Amazon RDS 資料庫或其他下游應用程式或服務，可能會出現網路組態問題。本主題會引導您完成一些步驟，以判斷您的網路組態是否有任何問題，以及更正這些問題的選項。若要深入瞭解 App Runner 的輸出流量設定，請參閱[為傳出流量啟用 VPC 存取](#)。

> **Note**
>
> 若要檢視您的 VPC 連接器設定，請從 App Runner 主控台左側導覽窗格中，選取 [網路設定]。然後選取 [外寄流量] 索引標籤。選取 VPC 連接器。下一頁顯示有關 VPC 連接器的詳細資料。您可以在此頁面檢視並向下展開下列項目：使用 VPC 的子網路、安全性群組和 App Runner 服務。

縮小應用程式無法連線至其他下游服務的原因

1. 確定 VPC 連接器中使用的子網路是私有子網路。如果連接器設定了公用子網路，您的服務將會遇到錯誤，因為每個子網路的基礎超平面 ENI (彈性網路介面) 沒有公用 IP 空間。

   如果您的 VPC 連接器使用公用子網路，您可以使用下列選項來更正此組態：

   a. 建立新的私有子網路，並使用它來取代 VPC 連接器的公有子網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的虛擬私人雲端適用的子網路。
   b. 透過 NAT 閘道路由現有的公用子網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。
2. 確認 VPC 連接器的安全群組輸入和輸出規則正確無誤。在 App Runner 主控台左側導覽窗格中，選取 [網路設定] > [外寄流量]。從清單中選取 VPC 連接器。下一頁會列出您可以選取要檢查的安全性群組。
3. 對於您嘗試連線的 RDS 執行個體或其他下游服務，確認安全群組輸入和輸出規則是否正確。如需詳細資訊，請參閱 App Runner 應用程式嘗試連線之下游服務的服務指南。
4. 要確認應用程序運行器配置之外沒有其他類型的網絡設置問題，請嘗試連接到 RDS 或 App Runner 之外的下游服務：

   a. 從相同虛擬私人雲端中的 Amazon EC2 執行個體，嘗試連接至 RDS 執行個體或服務。
   b. 如果您嘗試連線到服務 VPC 端點，請透過從相同 VPC 中的 EC2 執行個體存取相同的端點來驗證連線。
5. 如果步驟 4 中的任一連接測試失敗，則在您的 AWS帳戶中使用其他資源的 App Runner 配置之外很可能存在問題。請聯絡 Sup AWS port 部門以取得協助，以便進一步隔離並修正其他網路組態的問題。
6. 如果您按照步驟 4 中的指示成功連線至 RDS 執行個體或下游服務，請繼續執行此步驟中的指示。我們將透過啟用和檢查超平面 ENI 流量記錄來檢查流量是否正在進入 ENI。

   > **Note**
   >
   > 若要能夠完成這些步驟並取得必要的 ENI 流程記錄資訊，在 App Runner 服務成功啟動之後，必須進行 RDS 或下游服務的連線嘗試。當應用程式處於「執行中」狀態時，必須執行 RDS 或下游服務的連線作業。否則，ENI 可以作為應用程序執行器的回滾工作流程的一部分進行清理。這種方法可確保 ENI 仍然可用於進一步調查。

   a. 從主 AWS 控台啟動 EC2 主控台。

b. 從左側導覽窗格的「網路與安全性」群組中，選取「網路介面」。

c. 捲動至「介面類型」和「描述」欄，以在與 VPC 連接器相關聯的子網路中找到 ENI。他們將具有以下命名模式。

- 接口類型：遠門
- 描述：開頭為 AWSAppRunner ENI(範例:AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)

d. 使用列開頭的核取方塊來選取套用的 ENI。

e. 從「動作」功能表選取「建立流程記錄」。

f. 在提示中輸入資訊，然後選取頁面底部的「建立流程跳線」。

g. 檢查產生的流程記錄。

- 如果您在測試連線時流量正在進入 ENI，則問題與 ENI 設定無關。除了 App Runner 服務之外，您的 AWS 帳戶中的其他資源可能存在網路設定問題。聯絡 S AWS upport 以取得進一步協助。
- 如果您在測試連線時流量未進入 ENI，我們建議您聯絡 Sup AWS port 部門，以查看 Fargate 服務是否有任何已知問題。

h. 使用網路 Reachability Analyzer 工具。此工具可在虛擬網路路徑中的來源無法連線時識別封鎖元件，以協助判斷網路錯誤設定。如需詳細資訊，請參閱什麼是 Reachability Analyzer？在 Amazon VPC Reachability Analyzer 指南中。

輸入應用程式執行者 ENI 作為來源，並輸入 RDS ENI 作為目的地。

7. 如果您無法進一步縮小問題範圍，或者在完成前述步驟之後仍然無法連線至 RDS 或下游服務，我們建議您聯絡 Sup AWS port 以取得進一步協助。

應用程序運行器中的

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端安全性 — AWS 負責保護中執行 AWS 服務的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。若要深入瞭解適用於的規範遵循計劃 AWS App Runner，請參閱合規計劃的AWS 服務範圍範圍)。
- 雲端安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文檔可幫助您了解如何在使用 App Runner 時應用共同的責任模型。下列主題說明如何設定 App Runner 以符合安全性和合規性目標。您還將學習如何使用其他 AWS 服務來幫助您監視和保護應用程序運行器資源。

主題

- 應用程序運行器中的數據
- 應用程序運行器的身份和訪問管理
- 在應用程序運行器中進行日誌
- 應用程式執行器的合規性
- 應用程序運行程序中
- 基礎結構安全 AWS App Runner
- 搭配 VPC 端點使用應用程式執行器
- 應用程序運行器中的配置和漏洞分
- 應用程式執行器的安全性最佳

應用程序運行器中的數據

AWS 共用責任模型適用於中的資料保護 AWS App Runner。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。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 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用控制台，API 或 AWS SDK 使用應 AWS 服務 用程序運行器或其他應用程序運行器時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需其他「應用程式執行器安全」安全性主題，

主題

- 使用加密保護資料
- 網際網路流量隱私權

使用加密保護資料

AWS App Runner 從您指定的儲存庫讀取應用程式來源 (來源映像檔或原始程式碼)，並儲存它以供部署至您的服務。如需詳細資訊，請參閱 建築與概念。

資料保護是指在傳輸中（往返 App Runner）和靜態（儲存在 AWS 資料中心時）保護資料。

如需有關資料保護的詳細資訊，請參閱the section called “資料保護”。

如需其他「應用程式執行器安全」安全性主題，

傳輸中加密

您可以透過兩種方式實現資料保護：使用傳輸層安全性 (TLS) 加密連線，或使用用戶端加密 (物件在傳送前已加密)。這兩種方法都能保護您的應用程式資料。若要保護連線安全，請在您的應用程式、其開發人員和系統管理員及其使用者傳送或接收任何物件時，使用 TLS 加密連線。應用程式執行器會設定您的應用程式以透過 TLS 接收流量。

用戶端加密不是保護您提供給 App Runner 進行部署的來源映像檔或程式碼的有效方法。應用程序 Runner 需要訪問您的應用程序源，因此無法對其進行加密。因此，請務必確保開發或部署環境與 App Runner 之間的連線安全。

靜態加密和金鑰管理

為了保護應用程序的靜態數據，App Runner 會對應用程序源映像或源包的所有存儲副本進行加密。當您建立應用程式執行器服務時，您可以提供 AWS KMS key. 如果您提供了一個，App Runner 會使用您提供的密鑰來加密您的源。如果您不提供一個，應用程序運行器使用 AWS 受管金鑰 代替。

如需 App Runner 服務建立參數的詳細資訊，請參閱CreateService。如需有關 AWS Key Management Service (AWS KMS) 的資訊，請參閱開AWS Key Management Service 發人員指南。

網際網路流量隱私權

App Runner 使用 Amazon Virtual Private Cloud (Amazon VPC) 在 App Runner 應用程式中的資源之間建立界限，並控制它們、現場部署網路和網際網路之間的流量。如需有關 Amazon VPC 安全性的詳細資訊，請參閱 Amazon VPC 使用者指南中的 Amazon VPC 中的網路間流量隱私權。

如需將應用程式執行器應用程式與自訂 Amazon VPC 產生關聯的資訊，請參閱。the section called “傳出流量”

如需有關使用 VPC 端點保護對應用程式執行器的要求的資訊，請參閱the section called “VPC 端點”。

如需有關資料保護的詳細資訊，請參閱the section called “資料保護”。

如需其他「應用程式執行器安全」安全性主題，

應用程序運行器的身份和訪問管理

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證（登錄）和授權（具有權限）以使用應用程序運行器資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

如需其他「應用程式執行器安全」安全性主題，

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- 應用程式執行程式如何搭配 IAM
- 應用程式執行器身分識別原則範例
- 將服務連結角色用於應用程式執行
- AWS 受管理的政策 AWS App Runner
- 疑難排解應用程式執行器身分

物件

根據您在應用程式執行器中執行的工作，使用方式 AWS Identity and Access Management (IAM) 會有所不同。

服務使用者 — 如果您使用 App Runner 服務來完成工作，則管理員會為您提供所需的認證和權限。當您使用更多 App Runner 功能來完成工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取應用程式執行器中的功能，請參閱疑難排解應用程式執行器身分。

服務管理員 — 如果您負責公司的應用程式執行器資源，您可能擁有完整的應用程式執行器存取權。確定您的服務用戶應訪問哪些 App Runner 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何搭配 App Runner 使用 IAM，請參閱應用程式執行程式如何搭配 IAM。

IAM 管理員 — 如果您是 IAM 管理員，可能需要了解如何撰寫政策以管理 App Runner 存取權的詳細資訊。若要檢視可在 IAM 中使用的應用程式執行器身分型政策範例，請參閱。應用程式執行器身分識別原則範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料

都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI

或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (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。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

應用程式執行程式如何搭配 IAM

在您使用 IAM 管理存取權限之前 AWS App Runner，您應該瞭解哪些 IAM 功能可與 App Runner 搭配使用。若要取得應用程式執行器和其他 AWS 服務如何與 IAM 搭配運作的高階檢視，請參閱 IAM 使用者指南中的與 IAM 搭配使用的AWS 服務。

如需其他「應用程式執行器安全」安全性主題，

主題

- 應用程式執行器身分識別型原
- 應用程式執行資源型政策
- 基於應用運行器標籤的授權
- 應用程式執行者使用
- 應用程式執行者 IAM 角

應用程式執行器身分識別型原

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。應用程序運行器支持特定的操作，資源和條件鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

App Runner 中的策略操作在操作之前使用以下前綴：`apprunner:`。例如，若要授予某人使用 Amazon EC2 `RunInstances` API 作業來執行 Amazon EC2 執行個體的許可，請在其政策中加入 `ec2:RunInstances` 動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。App Runner 會定義它自己的一組動作，用來描述您可以使用此服務執行的工作。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

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 Describe 文字的所有動作，請包含以下動作：

```
"Action": "apprunner:Describe*"
```

要查看應用程序運行器操作的列表，請參閱服務授權參考 AWS App Runner中由定義的操作。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

應用程序運行器資源具有以下 ARN 結構：

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

如需 ARN 格式的詳細資訊，請參閱中的 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。AWS 一般參考

例如，若要在陳述式中指定my-service服務，請使用下列 ARN：

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

若要指定屬於特定帳戶的所有服務，請使用萬用字元 (*)：

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

某些應用程式執行器動作 (例如用於建立資源的動作) 無法在特定資源上執行。在這些情況下，您必須使用萬用字元 (*)。

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

要查看應用程序運行器資源類型及其 ARN 的列表，請參閱服務授權參考 AWS App Runner中由定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS App Runner定義的動作。

條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

應用程序運行器支持使用一些全局條件鍵。若要查看所有 AWS 全域條件金鑰，請參閱 IAM 使用者指南中的AWS 全域條件內容金鑰。

應用程式執行器會定義一組服務特定的條件金鑰。此外，App Runner 支援以標籤為基礎的存取控制，這是使用條件索引鍵來實作的。如需詳細資訊，請參閱 the section called “基於應用運行器標籤的授權”。

若要查看 App Runner 條件金鑰清單，請參閱服務授權參考 AWS App Runner中的條件金鑰。若要瞭解您可以使用條件索引鍵的動作和資源，請參閱定義的動作 AWS App Runner。

範例

若要檢視應用程式執行器身分識別型原則的範例，請參閱。應用程式執行器身分識別原則範例

應用程式執行資源型政策

應用程序運行器不支持基於資源的策略。

基於應用運行器標籤的授權

您可以將標籤附加到應用程序運行器資源，或將請求中的標籤傳遞給應用程序運行器 如需根據標籤控制存取，請使用 `apprunner: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。如需標記應用程式執行器資源的詳細資訊，請參閱the section called “組態”。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤控制對應用程式執行器服務的存取。

應用程式執行者使用

要使用應用程序運行器，IAM 用戶需要應用程序運行器操作的許可。向使用者授與許可的常用方式是將政策附加到 IAM 使用者或群組。如需有關管理使用者許可的詳細資訊，請參閱 IAM 使用者指南中的變更 IAM 使用者的許可。

App Runner 提供兩個受管理的政策，您可以將其附加到使用者。

- `AWSAppRunnerReadOnlyAccess`— 授予列出和查看有關應用程序運行器資源詳細信息的權限。
- `AWSAppRunnerFullAccess`— 授予所有應用程序運行器操作的權限。

若要更精細地控制使用者權限，您可以建立自訂原則並將其附加至您的使用者。如需詳細資訊，請參閱 IAM 使用者指南中的建立 IAM 政策。

如需使用者策略的範例，請參閱the section called “使用者政策”。

AWSAppRunnerReadOnlyAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

AWSAppRunnerFullAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/apprunner.amazonaws.com/
AWSServiceRoleForAppRunner",
        "arn:aws:iam::*:role/aws-service-role/networking.apprunner.amazonaws.com/
AWSServiceRoleForAppRunnerNetworking"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": [
            "apprunner.amazonaws.com",
            "networking.apprunner.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "apprunner.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AppRunnerAdminAccess",
      "Effect": "Allow",
      "Action": "apprunner:*",
      "Resource": "*"
```

```
    }
  ]
}
```

應用程式執行者 IAM 角

IAM 角色是您 AWS 帳戶 內部具有特定許可的實體。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代表您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

應用程式執行器支援服務連結角色 如需建立或管理 App Runner 服務連結角色的相關資訊，請參閱the section called “使用服務連結角色”。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 使用者可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

應用程式執行器支援一些服務角色。

存取角色

存取角色是應用程式執行器用來存取帳戶中 Amazon Elastic Container Registry (Amazon ECR) 中映像的角色。需要在 Amazon ECR 中訪問圖像，並且不需要 Amazon ECR 公共。在 Amazon ECR 中根據映像建立服務之前，請使用 IAM 建立服務角色並在其中使用`AWSAppRunnerServicePolicyForECRAccess`受管政策。然後，您可以在呼叫SourceConfiguration參數AuthenticationConfiguration成員中的 CreateServiceAPI 時，或使用應用程式執行器主控台建立服務時，將此角色傳遞給應用程式執行器。

AWSAppRunnerServicePolicyForECRAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
```

```
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 如果您為存取角色建立自己的自訂原則，請務必`"Resource": "*"`為`ecr:GetAuthorizationToken`動作指定。權杖可用於存取您有權存取的任何 Amazon ECR 登錄。

建立存取角色時，請務必新增將 App Runner 服務主體宣告`build.apprunner.amazonaws.com`為受信任實體的信任原則。

存取角色的信任原則

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "build.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果您使用 App Runner 主控台建立服務，主控台可以自動為您建立存取角色，並為新服務選擇該角色。主控台也會列出您帳戶中的其他角色，您可以視需要選取其他角色。

實例角色

執行個體角色是 App Runner 用來提供服務運算執行個體所需 AWS 服務動作權限的選用角色。如果您的應用程式程式碼呼叫 AWS 動作 (API)，您需要為 App Runner 提供執行個體角色。您可以在執行個

體角色中內嵌必要的權限，或是建立您自己的自訂原則，然後在執行個體角色中使用它。我們無法預測您的代碼使用哪些調用。因此，我們不會針對此目的提供受管理的政策。

在建立 App Runner 服務之前，請使用 IAM 建立具有所需自訂或內嵌政策的服務角色。然後，當您呼叫InstanceConfiguration參數`InstanceRoleArn`成員中的 CreateServiceAPI 時，或當您使用 App Runner 主控台建立服務時，可以將此角色當做執行個體角色傳遞給 App Runner。

建立執行個體角色時，請務必新增將 App Runner 服務主體宣告`tasks.apprunner.amazonaws.com`為受信任實體的信任原則。

執行個體角色的信任原則

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tasks.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果您使用 App Runner 主控台建立服務，主控台會列出您帳戶中的角色，而且您可以選取為此目的建立的角色。

如需有關建立服務的資訊，請參閱the section called “建立”。

應用程式執行器身分識別原則範例

依預設，IAM 使用者和角色沒有建立或修改 AWS App Runner 資源的權限。他們也無法使用 AWS Management Console AWS CLI、或 AWS API 執行工作。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 原則文件建立 IAM 身分型原則，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立原則。

如需其他「應用程式執行器安全」安全性主題，

主題

- 政策最佳實務
- 使用者政策
- 根據標籤控制對應用程式執行器服務的存取

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以在您的帳戶中建立、存取或刪除 App Runner 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 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。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，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，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

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。

使用者政策

若要存取應用程式執行器主控台，IAM 使用者必須擁有最少一組權限。這些權限必須允許您列出和查看有關 AWS 帳戶. 如果您建立的以身分識別為基礎的原則，該原則的限制性高於最低所需權限，則主控台將無法如同具有該原則的使用者使用。

App Runner 提供兩個受管理的政策，您可以將其附加到使用者。

- `AWSAppRunnerReadOnlyAccess`— 授予列出和查看有關應用程序運行器資源詳細信息的權限。
- `AWSAppRunnerFullAccess`— 授予所有應用程序運行器操作的權限。

若要確保使用者可以使用 App Runner 主控台，請至少將`AWSAppRunnerReadOnlyAccess`受管理的原則附加至使用者。您可以改為附加`AWSAppRunnerFullAccess`受管理的策略，或新增特定的其他權限，以允許使用者建立、修改和刪除資源。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。而是僅允許存取與您要允許使用者執行的 API 作業相符的動作。

下列範例示範自訂使用者原則。您可以使用它們作為定義自己的自訂使用者原則的起點。複製範例，和 (或) 移除動作、縮減資源範圍，以及新增條件。

範例：主控台和連線管理使用者原則

此範例原則可啟用主控台存取，並允許建立和管理連線。它不允許應用程序運行器服務的創建和管理。它可以附加到用戶，其角色是管理對源代碼資產的 App Runner 服務訪問。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner:DeleteConnection"
      ],
      "Resource": "*"
    }
  ]
}
```

範例：使用條件索引鍵的使用者政策

本節中的範例會示範依賴某些資源屬性或動作參數的條件式權限。

此示例策略允許創建應用程序運行器服務，但拒絕使用名為prod的連接。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
      "Effect": "Allow",
      "Action": "apprunner:CreateService",
      "Resource": "*",
      "Condition": {
        "ArnNotLike": {
          "apprunner:ConnectionArn": "arn:aws:apprunner:*:*:connection/prod/*"
        }
      }
    }
  ]
}
```

此範例原則可讓您preprod只使用名為的 auto 調整規模設定來更新名為的 App Runner 服務preprod。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
      "Effect": "Allow",
      "Action": "apprunner:UpdateService",
      "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
      "Condition": {
        "ArnLike": {
          "apprunner:AutoScalingConfigurationArn":
 "arn:aws:apprunner:*:*:autoscalingconfiguration/preprod/*"
        }
      }
    }
  ]
}
```

根據標籤控制對應用程式執行器服務的存取

您可以使用以身分識別為基礎的原則中的條件，根據標記控制對 App Runner 資源的存取。此範例顯示如何建立允許刪除應用程式執行器服務的原則。但是，只有在服務標籤 `Owner` 的值是該使用者的使用者名稱時，才會授予該許可。此政策也會授予在主控台完成此動作的必要許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:*:*:service/*",
      "Condition": {
        "StringEquals": {"apprunner:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果名為的使用者`richard-roe`嘗試刪除 App Runner 服務，則必須標記該服務`Owner=richard-roe`或`owner=richard-roe`。否則他便會被拒絕存取。條件標籤鍵 `Owner` 符合 `Owner` 和 `owner`，因為條件索引鍵名稱不區分大小寫。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。

將服務連結角色用於應用程式執行

AWS App Runner 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至應用程式執行器的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

主題

- 使用角色進行管理
- 使用角色進行網路

使用角色進行管理

AWS App Runner 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至應用程式執行器的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

服務連結角色可讓您輕鬆設定 App Runner，因為您不必手動新增必要的權限。App Runner 定義了其服務鏈接角色的權限，除非另有定義，否則只有 App Runner 可以承擔其角色。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。這可以保護您的 App Runner 資源，因為您無法無意中刪除訪問資源的權限。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (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。

應用程式執行器的服務連結角色權

應用程式執行器使用名為AWSServiceRoleForAppRunner的服務連結角色。

該角色允許應用程序運行器執行以下任務：

- 將日誌推送到 Amazon CloudWatch 日誌日誌群組。
- 創建 Amazon CloudWatch 活動規則以訂閱 Amazon Elastic Container Registry (Amazon ECR) 映像推送。
- 將追蹤資訊傳送至 AWS X-Ray。

服 AWSServiceRoleForAppRunner 務連結角色會信任下列服務擔任該角色：

- `apprunner.amazonaws.com`

AWSServiceRoleForAppRunner 服務連結角色的權限原則包含 App Runner 代表您完成動作所需的所有權限。

AppRunnerServiceRolePolicy 受管理政策

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

```
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutRetentionPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/apprunner/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/apprunner/*:log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets",
        "events:DescribeRule",
        "events:EnableRule",
        "events:DisableRule"
      ],
      "Resource": "arn:aws:events:*:*:rule/AWSAppRunnerManagedRule*"
    }
  ]
}
```

X-Ray 追蹤政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

為應用程式執行者建立服務連結角色

您不需要手動建立一個服務連結角色。當您在 AWS Management Console、或 AWS API 中建立應用程式執行器服務時 AWS CLI，應用程式執行器會為您建立服務連結的角色。

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。當您創建應用程序運行器服務時，應用程序運行器再次為您創建服務鏈接的角色。

編輯應用程式執行器的服務連結角色

應用程式執行器不允許您編輯 AWSServiceRoleForAppRunner 服務連結的角色。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。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除應用程式執行器的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，務必清除您的服務連結角色，之後才能以手動方式將其刪除。

清除服務連結角色

在您使用 IAM 刪除服務連結角色之前，您必須先刪除該角色所使用的任何資源。

在 App Runner 中，這意味著刪除您帳戶中的所有應用程序運行器服務。若要瞭解刪除應用程式執行器服務，請參閱the section called “刪除”。

> **Note**
>
> 當您嘗試刪除資源時，如果 App Runner 服務正在使用該角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

手動刪除 服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForAppRunner 服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

應用程式執行者服務連結角色的支援區

App Runner 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 AWS 一般參考中的 AWS App Runner 端點和配額。

使用角色進行網路

AWS App Runner 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至應用程式執行器的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

服務連結角色可讓您輕鬆設定 App Runner，因為您不必手動新增必要的權限。App Runner 定義了其服務鏈接角色的權限，除非另有定義，否則只有 App Runner 可以承擔其角色。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。這可以保護您的 App Runner 資源，因為您無法無意中刪除訪問資源的權限。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (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。

應用程式執行器的服務連結角色權

應用程式執行器使用名為AWSServiceRoleForAppRunnerNetworking的服務連結角色。

該角色允許應用程序運行器執行以下任務：

- 將 VPC 連接到您的應用程序運行器服務並管理網絡接口。

服 AWSServiceRoleForAppRunnerNetworking 務連結角色會信任下列服務擔任該角色：

- networking.apprunner.amazonaws.com

名為的角色權限原則 AppRunnerNetworkingServiceRolePolicy 包含 App Runner 代表您完成動作所需的所有權限。

AppRunnerNetworkingServiceRolePolicy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateNetworkInterface",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "AWSAppRunnerManaged"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        },
        "StringLike": {
          "aws:RequestTag/AWSAppRunnerManaged": "*"
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DeleteNetworkInterface",
      "Resource": "*",
      "Condition": {
        "Null": {
          "ec2:ResourceTag/AWSAppRunnerManaged": "false"
        }
      }
    }
  ]
}
```

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。

為應用程式執行者建立服務連結角色

您不需要手動建立一個服務連結角色。當您在、或 AWS API 中建立 VPC 連接器時 AWS Management Console，應用程式執行器會為您建立服務連結角色。 AWS CLI

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。當您建立 VPC 連接器時，App Runner 會再次為您建立服務連結的角色。

編輯應用程式執行器的服務連結角色

應用程式執行器不允許您編輯 AWSServiceRoleForAppRunnerNetworking 服務連結的角色。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。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除應用程式執行器的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，務必清除您的服務連結角色，之後才能以手動方式將其刪除。

清除服務連結角色

在您使用 IAM 刪除服務連結角色之前，您必須先刪除該角色所使用的任何資源。

在 App Runner 中，這表示將 VPC 連接器與帳戶中所有 App Runner 服務取消關聯，並刪除 VPC 連接器。如需詳細資訊，請參閱 the section called “傳出流量”。

> **Note**
>
> 當您嘗試刪除資源時，如果 App Runner 服務正在使用該角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForAppRunnerNetworking 服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

應用程式執行者服務連結角色的支援區

App Runner 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS App Runner 端點和配額。

AWS 受管理的政策 AWS App Runner

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管理策略的應用程式執行器

檢視有關 App Runner AWS 受管政策更新的詳細資料，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動警示，請訂閱 App Runner 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AWSAppRunnerReadOnlyAccess – 新政策	App Runner 新增了一項新政策，允許使用者列出並檢視有關應用程式執行器資源的詳細資料	2022年2月24日
AWSAppRunnerFullAccess – 更新現有政策	App Runner 更新了`iam:CreateServiceLinkedRole` 動作的資源清單，以允許建立`AWSServiceRoleForAppRunnerNetworking` 服務連結角色。	2022年2月8日
AppRunnerNetworkingServiceRolePolicy – 新政策	App Runner 新增了一項新政策，允許應用程式執行器撥打電話至 Amazon Virtual Private Cloud，將 VPC 連接到您的應用程式執行器服務，並代表 App Runner 服務管理網路界面。該策略用於`AWSServiceRoleForAppRunnerNetworking` 服務連結角色。	2022年2月8日
AWSAppRunnerFullAccess – 新政策	應用程序運行器添加了新的策略，以允許用戶執行所有應用程序運行器操作	2022年1月10日
AppRunnerServiceRolePolicy – 新政策	應用程序運行器添加了一項新政策，允許應用程序運行器代表應用程序運行器服務撥打 Amazon CloudWatch 日誌和 Amazon CloudWatch 事件的調用。該策略用於`AWSServiceRoleForAppRunner` 服務連結角色。	二零二一年三月一日
AWSAppRunnerServicePolicyForECRAccess – 新政策	應用程序運行器添加了一個新的政策，允許應用程序運行器訪問 Amazon Elastic Container Registry (Amazon ECR) 圖像在您的帳戶。	二零二一年三月一日
應用程式執行器開始追蹤	應用程式執行器開始追蹤其 AWS 受管理政策的變更。	二零二一年三月一日

疑難排解應用程式執行器身分

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 AWS App Runner 常見問題。

如需其他「應用程式執行器安全」安全性主題，

主題

- 我沒有授權在應用程序運行器中執行操作
- 我想允許我以外的人訪問我 AWS 帳戶 的應用程序運行器資源

我沒有授權在應用程序運行器中執行操作

如果 AWS Management Console 告訴您您沒有執行動作的授權，請聯絡您的系統管理員以尋求協助。您的管理員是提供您 AWS 登入認證的人員。

如果名為的 IAM 使用者`marymajor`嘗試使用主控台來檢視有關 App Runner 服務的詳細資料，但沒有`apprunner:DescribeService`權限，就會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

在此情況下，Mary 會要求管理員更新策略，以允許她使用`apprunner:DescribeService`動作存取*`my-example-service`*資源。

我想允許我以外的人訪問我 AWS 帳戶 的應用程序運行器資源

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：

- 要了解應用程序運行器是否支持這些功能，請參閱應用程式執行程式如何搭配 IAM。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。

- 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)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

在應用程序運行器中進行日誌

監控是維持服務可靠性、可用性和效能的重要組成部 AWS App Runner 分。從 AWS 解決方案的所有部分收集監視資料，可讓您在發生故障時更輕鬆地除錯。App Runner 與多種 AWS 工具集成，用於監視您的應用程序運行器服務並響應潛在事件。

Amazon CloudWatch 警報

使用 Amazon CloudWatch 警示，您可以觀看指定期間內的服務指標。如果測量結果超過指定期間數目的指定臨界值，您會收到通知。

App Runner 會收集有關整體服務以及執行 Web 服務的執行個體 (縮放單位) 的各種指標。如需詳細資訊，請參閱 度量 (CloudWatch)。

應用程式記錄

應用程式執行器會收集應用程式程式碼的輸出，並將其串流至 Amazon CloudWatch 日誌。什麼是在這個輸出是由你。例如，您可以包含對 Web 服務發出請求的詳細記錄。這些日誌記錄在安全性和訪問審核中可能很有用。如需詳細資訊，請參閱 記錄檔 (CloudWatch 記錄檔)。

AWS CloudTrail 動作記錄檔

應用程序 Runner 與服務集成 AWS CloudTrail，該服務可提供用戶，角色或應用程序運行器中的 AWS 服務所採取的操作的記錄。 CloudTrail 捕獲應用程序運行器的所有 API 調用作為事件。您可以在 CloudTrail 主控台中檢視最近的事件，也可以建立追蹤，以便將 CloudTrail 事件持續傳遞到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如需更多詳細資訊，請參閱 API 動作 (CloudTrail)。

應用程式執行器的合規性

協力廠商稽核人員會評估其安全性與合規性， AWS App Runner 做為多個 AWS 合規計畫的一部分。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於您資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您滿足特定合規性架構所要求的入侵偵測需求，例如 PCI DSS 等各種合規性需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

如需其他「應用程式執行器安全」安全性主題，

應用程序運行程序中

AWS 全球基礎架構是圍繞 AWS 區域 和可用區域建立的。 AWS 區域 提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用

程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱AWS 全域基礎結構。

AWS App Runner 代表您管理和自動化 AWS 全球基礎架構的使用。使用 App Runner 時，您可以從 AWS 提供的可用性和容錯機制中受益。

如需其他「應用程式執行器安全」安全性主題，

基礎結構安全 AWS App Runner

作為受管服務， AWS App Runner 受 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序保護。

您可以使用 AWS 已發佈的 API 呼叫透過網路存取應用程式執行器。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

如需其他「應用程式執行器安全」安全性主題，

搭配 VPC 端點使用應用程式執行器

您的 AWS 應用程式可能會將 AWS App Runner 服務與 AWS 服務 從 Amazon 虛擬私有雲 (Amazon VPC) 在 VPC 中執行的其他服務整合。部分應用程式可能會從 VPC 內向應用程式執行器發出要求。例如，您可 AWS CodePipeline 以使用持續部署到您的應用程式執行器服務。提高應用程序安全性的一種方法是通過 VPC 端點發送這些 App Runner 請求（並將請求發送給其他請求 AWS 服務）。

使用 VPC 端點，您可以將您的 VPC 以私密方式連接到由支援的 VPC 端點服務 AWS 服務 和 VPC 端點服務。 AWS PrivateLink您不需要網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。

VPC 中的資源不會使用公用 IP 位址與應用程式執行器資源進行互動。您的 VPC 和應用程序運行器之間的流量不會離開 Amazon 網絡。如需 VPC 端點的詳細資訊，請參閱指AWS PrivateLink 南中的 VPC 端點。

> **Note**
>
> 默認情況下，應用程序運行器服務中的 Web 應用程序在應用程序運行器提供和配置的 VPC 中運行。這個 VPC 是公開的。這意味著它已連接到互聯網。您可以選擇將應用程式與自訂 VPC 產生關聯。如需詳細資訊，請參閱 the section called “傳出流量”。
> 您可以將服務設定為存取網際網路 (包括 AWS API)，即使您的服務已連線至 VPC 也是如此。如需如何針對 VPC 輸出流量啟用公用網際網路存取的指示，請參閱the section called “選取子網路時的考量 ”。
> 應用程式執行程式不支援為您的應用程式建立 VPC 端點。

為應用程式執行器設定 VPC 端點

若要在 VPC 中為 App Runner 服務建立介面 VPC 端點，請遵循指南中的建立介面端點程序。AWS PrivateLink 在 Service Name (服務名稱) 中，選擇 `com.amazonaws.region.apprunner`。

VPC 網路隱私權考量

> **Important**
>
> 針對 App Runner 使用 VPC 端點並不能確保來自 VPC 的所有流量都不會離開網際網路。VPC 可能是公開的。此外，解決方案的某些部分可能不會使用 VPC 端點進行 AWS API 呼叫。例如， AWS 服務 可能會使用其公有端點呼叫其他服務。如果 VPC 中的解決方案需要流量隱私，請閱讀本節。

若要確保 VPC 中網路流量的隱私權，請考慮下列事項：

- 啟用 DNS 名稱 — 部分應用程式可能仍會使用`apprunner.region.amazonaws.com`公用端點透過網際網路傳送要求給 App Runner。如果您的 VPC 設定了網際網路存取，則這些要求會成功，而不會有任何指示。您可以確保在建立端點時啟用「啟用 DNS 名稱」，以防止這種情況發生。默認情況下，它設置為 true。這會在 VPC 中新增 DNS 項目，此項目會將公有服務端點映射至界面 VPC 端點。
- 為其他服務設定 VPC 端點 — 您的解決方案可能會將請求傳送給其他 AWS 服務服務。例如， AWS CodePipeline 可能會將請求傳送至 AWS CodeBuild。為這些服務設定 VPC 端點，並在這些端點上啟用 DNS 名稱。

- 設定私人虛擬私人雲端 — 如果可能 (如果您的解決方案根本不需要網際網路存取)，請將您的 VPC 設定為私有，這表示它沒有網際網路連線。這可確保遺失的 VPC 端點會導致可見錯誤，以便您可以新增遺失的端點。

使用端點政策搭配 VPC 端點來控制存取

應用程式執行器不支援 VPC 端點原則。默認情況下，允許通過接口端點對應用程序運行器的完全訪問。或者，您可以將安全群組與端點網路介面相關聯，以控制透過介面端點傳送至 App Runner 的流量。

與介面端點整合

應用程序運行器支持 AWS PrivateLink，它為應用程序運行器提供私人連接，並消除了流量暴露在互聯網上。若要讓您的應用程式能夠使用將要求傳送至 App Runner AWS PrivateLink，請設定一種稱為介面端點的 VPC 端點類型。如需詳細資訊，請參閱AWS PrivateLink 指南中的介面 VPC 端點 (AWS PrivateLink)。

應用程序運行器中的配置和漏洞分

AWS 我們的客戶有責任達到高度軟體元件的安全性和合規性。如需詳細資訊，請參閱 AWS 共用的責任模型。

如需其他「應用程式執行器安全」安全性主題，

應用程式執行器的安全性最佳

AWS App Runner 在您開發和實作自己的安全性原則時，提供數項安全性功能供您考量。以下最佳實務為一般準則，並不代表完整的安全解決方案。因為這些最佳實務可能不適合或無法滿足您的環境，所以請將它們視為實用建議，不要當成指示。

如需其他「應用程式執行器安全」安全性主題，

預防性安全最佳實務

預防性安全控制會嘗試在事件發生前防止事件發生。

實作最低權限存取

應用程式執行器為 IAM 使用者和存取角色提供 AWS Identity and Access Management (IAM) 受管政策。這些受管政策會指定正確操作 App Runner 服務所需的所有權限。

您的應用程式可能不需要受管政策中的所有許可。您可以對其進行自定義，並僅授予用戶和 App Runner 服務執行其任務所需的權限。這特別關係到不同使用者角色可能有不同許可需求的使用者政策。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限存取是相當重要的一環。

偵測性安全最佳實務

偵測性安全控制會在安全違規發生後識別出它們。它們可協助您偵測潛在的安全威脅或事件。

實作監控

監控是維護 App Runner 解決方案的可靠性，安全性，可用性和性能的重要組成部分。 AWS 提供數種工具和服務來協助您監控 AWS 服務。

以下是一些要監控的項目範例：

- 適用於應用程式執行器的 Amazon CloudWatch 指標 — 為關鍵應用程式執行器指標和應用程式的自訂指標設定警示。如需詳細資訊，請參閱 度量 (CloudWatch)。
- AWS CloudTrail 項目 — 追蹤可能影響可用性的動作，例如`PauseService`或`DeleteConnection`。如需詳細資訊，請參閱 API 動作 (CloudTrail)。

AWS 詞彙表

有關最新 AWS 術語，請參閱AWS 詞彙表 參考文獻中的AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。