開發人員指南

AWS App Runner

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

AWS App Runner: 開發人員指南

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

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

Table of Contents

..... x
什麼是 AWS App Runner？ 1
誰是 App Runner？ 1
存取 App Runner 1
App Runner 定價 2
下一步 2
可用性變更 3
遷移概觀 3
先決條件 3
開始之前 4
遷移演練 4
步驟 1：檢閱現有的 App Runner 組態 5
步驟 2：建立 ECS Express Mode 服務 5
步驟 3：設定 ECS Express 模式的自訂網域 6
步驟 4：使用 Route 53 加權路由轉移流量 7
步驟 5：完成遷移 8
遷移以來源為基礎的部署 9
容器化您的應用程式 10
設定自動部署的 GitHub 動作 10
其他資源 13
設定 14
註冊 AWS 帳戶 14
下一步 14
開始使用 15
先決條件 15
步驟 1：建立 App Runner 服務 16
步驟 2：變更您的服務代碼 26
步驟 3：進行組態變更 27
步驟 4：檢視您服務的日誌 29
步驟 5：清除 31
下一步 31
架構和概念 33
App Runner 概念 33
App Runner 支援的組態 35

App Runner 資源 ... 36
App Runner 資源配額 ... 37
影像型服務 ... 39
映像儲存庫供應商 ... 39
使用您 AWS 帳戶中存放在 Amazon ECR 中的映像 ... 39
在不同 AWS 帳戶中使用存放在 Amazon ECR 中的映像 ... 40
使用存放在 Amazon ECR Public 中的映像 ... 41
影像範例 ... 42
程式碼型服務 ... 43
原始程式碼儲存庫提供者 ... 44
從原始碼儲存庫供應商部署 ... 44
來源目錄 ... 44
App Runner 受管平台 ... 45
受管執行時間版本的終止支援 ... 46
受管執行期版本和 App Runner 組建 ... 47
App Runner 建置和遷移的詳細資訊 ... 49
Python 平台 ... 52
Python 執行期組態 ... 53
特定執行時間版本的呼叫 ... 53
Python 執行時間範例 ... 54
版本資訊 ... 59
Node.js 平台 ... 61
Node.js 執行期組態 ... 62
特定執行時間版本的呼叫 ... 64
Node.js 執行時間範例 ... 64
版本資訊 ... 68
Java 平台 ... 71
Java 執行期組態 ... 72
Java 執行時間範例 ... 73
版本資訊 ... 77
.NET 平台 ... 79
.NET 執行時間組態 ... 80
.NET 執行時間範例 ... 80
版本資訊 ... 82
PHP 平台 ... 84
PHP 執行期組態 ... 85

相容性 ... 86
PHP 執行時間範例 ... 87
版本資訊 ... 95
Ruby 平台 ... 97
Ruby 執行時間組態 ... 98
Ruby 執行時間範例 ... 98
版本資訊 ... 101
Go 平台 ... 102
Go 執行時間組態 ... 103
Go 執行時間範例 ... 103
版本資訊 ... 105
開發 App Runner ... 107
執行時間資訊 ... 107
程式碼開發指導方針 ... 108
App Runner 主控台 ... 110
整體主控台配置 ... 110
服務頁面 ... 110
服務儀表板頁面 ... 111
已連線帳戶頁面 ... 112
Auto Scaling 組態頁面 ... 112
管理您的服務 ... 114
建立 ... 114
先決條件 ... 114
建立服務 ... 115
重建失敗的服務 ... 129
使用 App Runner 主控台重建失敗的 App Runner 服務 ... 130
使用 App Runner API 或 重建失敗的 App Runner 服務 AWS CLI ... 131
部署 ... 132
部署方法 ... 132
手動部署 ... 134
Configuration ... 135
使用 App Runner API 或 設定您的服務 AWS CLI ... 136
使用 App Runner 主控台設定您的服務 ... 136
使用 App Runner 組態檔案設定您的服務 ... 138
可觀測性組態 ... 138
組態資源 ... 139

運作狀態檢查組態 141
連線 142
管理連線 143
自動擴展 144
管理服務的自動擴展 146
管理自動擴展組態資源 147
自訂網域名稱 154
將自訂網域與您的服務建立關聯（連結） 155
取消關聯（取消連結）自訂網域 157
管理自訂網域 157
設定 Amazon Route 53 別名記錄 165
暫停/繼續 167
暫停和刪除比較 167
當您的服務暫停時 168
暫停並繼續您的服務 168
刪除 170
暫停和刪除比較 170
App Runner 會刪除哪些項目？ 170
刪除您的服務 171
參考環境變數 173
參考敏感資料做為環境變數 173
考量事項 174
許可 175
管理環境變數 176
App Runner 主控台 176
App Runner API 或 AWS CLI 178
聯網 184
術語 184
一般條款 184
設定傳出流量的特定術語 185
設定傳入流量的特定詞彙 185
傳入流量 185
標頭 186
啟用私有端點 186
為 App Runner 的端點啟用 IPv6 197
傳出流量 200

VPC 連接器 ... 201
子網路 ... 201
安全群組 ... 202
管理 VPC 存取 ... 203
可觀測性 ... 208
活動 ... 208
追蹤 App Runner 服務活動 ... 208
日誌 (CloudWatch Logs) ... 209
App Runner 日誌群組和串流 ... 210
在主控台中檢視 App Runner 日誌 ... 211
指標 (CloudWatch) ... 213
App Runner 指標 ... 214
在主控台中檢視 App Runner 指標 ... 215
事件處理 (EventBridge) ... 217
建立 EventBridge 規則以對 App Runner 事件採取行動 ... 218
App Runner 事件範例 ... 218
App Runner 事件模式範例 ... 219
App Runner 事件參考 ... 220
API 動作 (CloudTrail) ... 222
CloudTrail 中的 App Runner 資訊 ... 222
了解 App Runner 日誌檔案項目 ... 223
追蹤 (X-Ray) ... 226
檢測您的應用程式以進行追蹤 ... 227
將 X-Ray 許可新增至您的 App Runner 服務執行個體角色 ... 230
為您的 App Runner 服務啟用 X-Ray 追蹤 ... 230
檢視 App Runner 服務的 X-Ray 追蹤資料 ... 230
AWS WAF Web ACL ... 232
傳入 Web 請求流程 ... 232
將 WAF Web ACLs 與您的 App Runner 服務建立關聯 ... 233
考量事項 ... 233
許可 ... 234
管理 Web ACLs ... 235
App Runner 主控台 ... 236
AWS CLI ... 239
測試和記錄 AWS WAF Web ACLs ... 243
App Runner 組態檔案 ... 245

範例 245
組態檔案範例 246
參考資料 248
結構概觀 249
頂端區段 249
組建區段 250
執行區段 252
App Runner API 256
使用 AWS CLI 搭配 App Runner 256
使用 AWS CloudShell 256
取得 的 IAM 許可 AWS CloudShell 257
使用 與 App Runner 互動 AWS CloudShell 257
使用 驗證您的 App Runner 服務 AWS CloudShell 260
疑難排解 262
無法建立服務 262
自訂網域名稱 263
取得自訂網域的建立失敗錯誤 263
取得自訂網域的 DNS 憑證驗證擱置錯誤 264
基本疑難排解命令 264
自訂網域憑證續約 265
請求路由錯誤 266
404 將 HTTP/HTTPS 流量傳送至 App Runner 服務端點時發生錯誤 266
連線至 Amazon RDS 或下游服務失敗 267
當沒有足夠的 IP 地址來啟動或擴展時 269
如何更新您的服務以擁有更多可用的 IPs 270
計算服務所需的 IPs 270
建立新的子網路 (s) 270
將次要 CIDR 區塊連接至 VPC 271
驗證 272
常見缺陷 272
其他資源 273
詞彙表 273
安全 274
資料保護 274
資料加密 275
網際網路隱私權 276

身分與存取管理 276
目標對象 277
使用身分驗證 277
使用政策管理存取權 278
App Runner 和 IAM 280
身分型政策範例 285
使用服務連結角色 290
AWS 受管政策 295
疑難排解 296
日誌記錄和監控 298
法規遵循驗證 298
恢復能力 299
基礎設施安全性 299
VPC 端點 299
設定 App Runner 的 VPC 端點 300
VPC 網路隱私權考量事項 300
使用端點政策搭配 VPC 端點來控制存取 301
與介面端點整合 301
共同責任模型 301
修補程式容器映像 301
安全最佳實務 301
預防性安全最佳實務 302
偵測性安全最佳實務 302
AWS 詞彙表 303

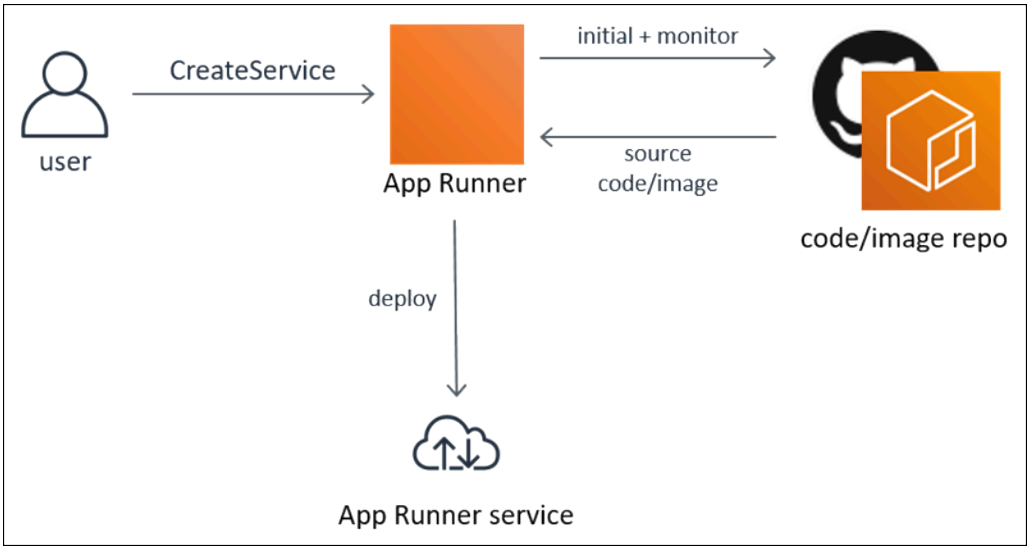
AWS App Runner 不再開放給新客戶。現有客戶可以繼續正常使用該服務。如需詳細資訊，請參閱AWS App Runner 可用性變更。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

什麼是 AWS App Runner？

AWS App Runner 是一種 AWS 服務，可提供快速、簡單且符合成本效益的方式，從原始碼或容器映像直接部署到 AWS 雲端中可擴展且安全的 Web 應用程式。您不需要學習新技術、決定要使用的運算服務，或知道如何佈建和設定 AWS 資源。

App Runner 會直接連線至您的程式碼或映像儲存庫。它提供具有全受管操作、高效能、可擴展性和安全性的自動整合和交付管道。



誰是 App Runner？

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

對於操作團隊，App Runner 會在每次將遞交推送至程式碼儲存庫或將新的容器映像版本推送至映像儲存庫時啟用自動部署。

存取 App Runner

您可以使用下列任一界面來定義和設定 App Runner 服務部署：

- App Runner 主控台 – 提供用於管理 App Runner 服務的 Web 界面。
- App Runner API – 提供執行 App Runner 動作的 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) – 為廣泛的 AWS 服務提供命令，包括 Amazon VPC，並支援 Windows、macOS 和 Linux。如需詳細資訊，請參閱AWS Command Line Interface。

- AWS SDKs – 提供語言特定的 APIs，並負責許多連線詳細資訊，例如計算簽章、處理請求重試和錯誤處理。如需詳細資訊，請參閱 AWS 開發套件。

App Runner 定價

App Runner 提供經濟實惠的方式來執行您的應用程式。您只需為 App Runner 服務使用的資源付費。當請求流量較低時，您的服務會縮減至較少的運算執行個體。您可以控制可擴展性設定：佈建的執行個體數量最低和最高，以及執行個體處理的最高負載。

如需 App Runner 自動擴展的詳細資訊，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

了解如何在下列主題中開始使用 App Runner：

- 設定 – 完成使用 App Runner 的先決條件步驟。
- 開始使用 – 將您的第一個應用程式部署到 App Runner。

AWS App Runner 可用性變更

在仔細考慮之後，我們決定 AWS App Runner 接近新客戶。現有 AWS App Runner 客戶可以繼續正常使用服務，包括建立新的資源和服務。 AWS 會繼續投資 的安全性和可用性 AWS App Runner，但我們不打算推出新功能。

我們建議客戶在遷移時探索 Amazon Elastic Container Service (Amazon ECS) Express 模式 AWS App Runner。Amazon ECS Express Mode 會保留 App Runner 的操作簡單性，同時提供對更廣泛的 Amazon ECS 功能集的存取。透過單一 API 呼叫，您可以提供容器映像和兩個 IAM 角色，Amazon ECS 會在您的帳戶中佈建完整的應用程式堆疊 AWS ，包括 Fargate 上的 ECS 服務、Application Load Balancer、自動擴展和聯網。使用 Amazon ECS Express 模式無需額外費用。您只需為為了執行應用程式而建立的基礎 AWS 資源付費。

本指南說明如何將現有的 App Runner 服務遷移至 ECS Express 模式，並使用 DNS 路由逐步轉移流量。

遷移概觀

本指南使用藍/綠部署方法搭配 DNS 加權路由，將流量從 App Runner 遷移至 ECS Express 模式。這兩個服務會在遷移期間同時執行。您可以使用 Amazon Route 53 （或您的 DNS 供應商） 逐步將流量從 App Runner 服務轉移到 ECS Express Mode 服務，從小百分比開始，並隨著時間增加。此方法可將停機時間降至最低，並可讓您在發生問題時調整 DNS 權重來復原。

典型的遷移包括以下步驟：

1. 檢閱現有 App Runner 服務的組態
2. 使用相同的容器映像建立 ECS Express Mode 服務
3. 如果您使用自訂網域，請為 ECS Express Mode 服務設定相同的自訂網域
4. 使用 DNS 路由將流量從 App Runner 轉移到 ECS Express 模式
5. 完成遷移，並在不再需要 App Runner 服務時將其刪除

先決條件

開始之前，請確定您有下列項目：

- 具有適當 AWS Identity and Access Management 許可來建立和管理 Amazon ECS AWS App Runner、Amazon Route 53 和 Application Load Balancer 資源 AWS 的帳戶

- AWS CLI 使用您 AWS 帳戶的登入資料安裝和設定
- 存放在 Amazon Elastic Container Registry （或其他容器登錄檔） 中的容器映像，以部署至 ECS Express 模式
- ECS Express Mode 所需的 IAM 角色：`ecsTaskExecutionRole`適用於 Amazon ECS 任務執行和 `ecsInfrastructureRoleForExpressServices` ECS Express Mode 基礎設施佈建

如果您想要在遷移期間保留現有的自訂網域，您也需要：

- 您使用 Amazon Route 53 或第三方網域註冊商控制的註冊網域名稱`app.example.com`，例如
- AWS Certificate Manager (ACM) 中符合您自訂網域的 SSL/TLS 憑證。在您部署資源的相同 AWS 區域 中請求公有 ACM 憑證。App Runner 和 Amazon ECS Express 模式都需要 ACM 憑證，才能使用自訂網域來啟用 HTTPS 存取。

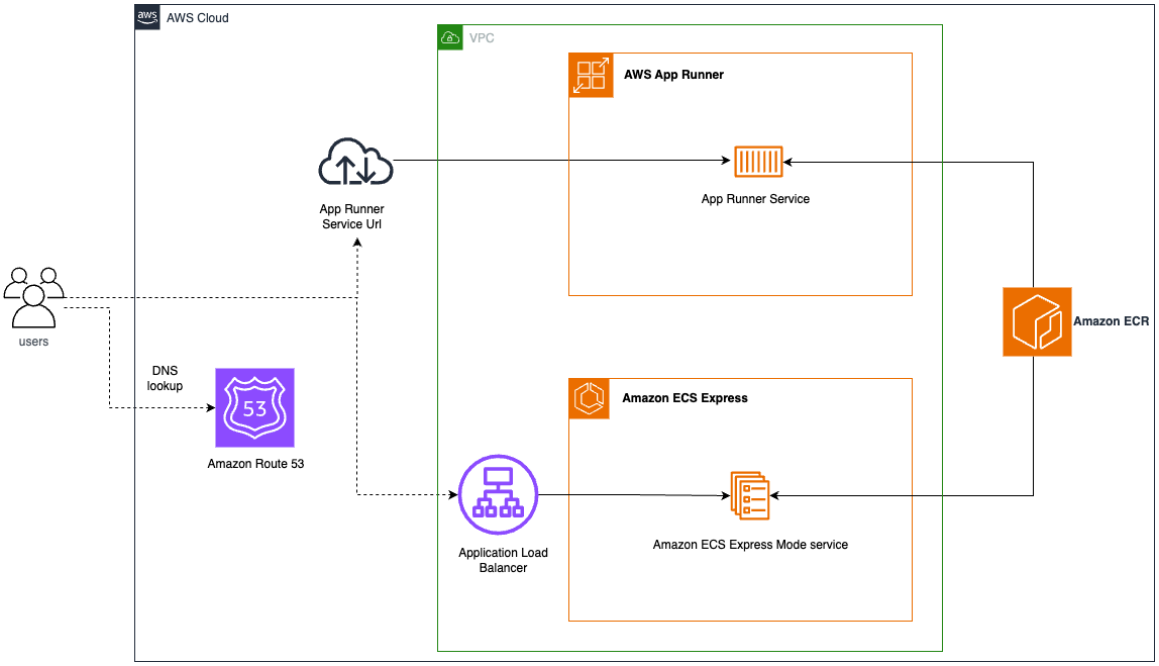
開始之前

- 容器映像需求 — ECS Express Mode 部署容器映像。如果您的 App Runner 服務是從原始程式碼部署，請先新增建立容器映像的建置步驟，並將其推送至登錄檔，例如 Amazon Elastic Container Registry。然後將該映像部署到 ECS Express 模式。如需遷移來源型部署的詳細資訊遷移以來源為基礎的部署，請參閱 。
- 網域行為 — 如果您的 App Runner 服務已使用自訂網域，例如 `app.example.com`，您可以在遷移期間重複使用相同的主機名稱，並透過更新 DNS 在 App Runner 和 ECS Express 模式之間逐步轉移流量。

  如果您的 App Runner 服務僅使用預設的 App Runner 服務 URL，則 ECS Express Mode 服務會有不同的端點。在這種情況下，沒有可用於逐步流量轉移的共用主機名稱。您應該建立並驗證 ECS Express Mode 服務，然後更新用戶端或 DNS 以使用新的端點。

遷移演練

下圖顯示遷移如何使用 Route 53 在 App Runner 服務和 ECS Express Mode 服務之間轉移 DNS 記錄。



步驟 1：檢閱現有的 App Runner 組態

在 App Runner 主控台中，檢閱您現有的服務，並記下您要轉送的值。請至少注意下列事項：

- 容器映像
- 應用程式連接埠
- 環境變數
- 自訂網域名稱，如果已設定
- 與自訂網域相關聯的 ACM 憑證，如果已設定

您也可以檢閱要轉送至新服務的任何其他執行時間設定。

如需自訂網域詳細資訊，請參閱 the section called “自訂網域名稱”。

步驟 2：建立 ECS Express Mode 服務

使用 App Runner 服務所使用的相同容器映像建立 ECS Express Mode 服務。您可以使用 AWS 管理主控台或 建立服務AWS CLI。

CLI 命令範例：

```
aws ecs create-express-gateway-service \
    --execution-role-arn arn:aws:iam::123456789012:role/ecsTaskExecutionRole \
```

```
    --infrastructure-role-arn arn:aws:iam::123456789012:role/
ecsInfrastructureRoleForExpressServices \
    --primary-container '{
        "image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/my-app:latest",
        "containerPort": 8080,
        "environment": [{
            "name": "ENV_VAR_NAME",
            "value": "value"
        }]
    }' \
    --service-name "my-application" \
    --health-check-path "/" \
    --scaling-target '{"minTaskCount":1,"maxTaskCount":4}' \
    --monitor-resources
```

將映像、連接埠、環境變數和擴展值取代為 App Runner 服務中的值。

此命令會在您的帳戶中佈建完整的應用程式堆疊 AWS ，包括 Fargate 上的 ECS 服務、具有目標群組和運作狀態檢查的 Application Load Balancer、自動擴展政策、安全群組和聯網組態，以及預設 URL。

佈建通常需要 3-5 分鐘。您可以在 資源索引標籤下的 Amazon ECS 主控台中追蹤進度。

完成後，請使用主控台中顯示的預設 URL 測試 ECS Express Mode 服務。在繼續流量轉移之前，請確認您的應用程式正常運作。

步驟 3：設定 ECS Express 模式的自訂網域

如果您的 App Runner 服務使用自訂網域，請在轉移流量之前為 ECS Express Mode 服務設定相同的自訂網域。此步驟會設定為 ECS Express Mode 服務建立的 Application Load Balancer，以便接受您網域的流量，並使用 HTTPS 的 ACM 憑證。

- 在 Application Load Balancer 接聽程式規則中，將自訂網域新增為主機標頭條件。使用與 App Runner 服務相關聯的相同網域名稱（例如 `app.example.com`)。這會通知 Application Load Balancer 將流量從您的網域路由到 ECS Express Mode 目標群組。
- 將 SSL 憑證新增至 Application Load Balancer HTTPS 接聽程式。將步驟 1 中記下的 ACM 憑證新增至 HTTPS 接聽程式。

如需詳細說明，請參閱《Amazon ECS 開發人員指南》中的將自訂網域新增至您的服務。

下圖顯示 Application Load Balancer 接聽程式規則中設定主機標頭條件的範例。

步驟 4：使用 Route 53 加權路由轉移流量

如果您的 App Runner 服務已使用自訂網域，您可以使用 Route 53 加權路由逐步將流量轉移到 ECS Express Mode 服務。加權路由可讓您將相同主機名稱的流量路由到多個端點。每個端點都定義為具有自己的權重的個別 DNS 記錄，Route 53 會根據這些權重分配請求。

> **Note**
>
> 本指南使用 Route 53 做為範例。如果您使用其他 DNS 提供者，請使用提供者的流量管理功能進行同等 DNS 變更。

將現有的 App Runner 記錄轉換為加權記錄：

1. 開啟 Route 53 主控台。
2. 選擇託管區域，然後選取網域的託管區域。
3. 找出目前指向 App Runner 的主機名稱現有記錄（例如 `app.example.com`)。
4. 編輯記錄，並將其路由政策變更為加權。
5. 將權重設定為 `100`（這會將所有初始流量導向 App Runner)。
6. 在記錄 ID 下，輸入描述性識別符，例如 `app-runner-service`。
7. 選擇儲存變更。

建立 ECS Express 模式的加權記錄：

1. 在相同的託管區域中建立新的記錄。

2. 使用相同的記錄名稱（例如 `app.example.com`)。
3. 使用相同的記錄類型。
4. 將路由政策設定為加權。
5. 在將流量路由到 下，選擇 Application 和 Classic Load Balancer 的別名。
6. 從下拉式清單中選擇您的 ECS Express Mode Application Load Balancer。
7. 將權重設定為 `0`（在您明確增加權重之前，不會有流量流向 ECS Express 模式）。
8. 在記錄 ID 下，輸入描述性識別符，例如 `ecs-express-service`。
9. 選擇建立記錄。

逐漸轉移流量：

設定 DNS 記錄後，透過增加 ECS Express 模式權重，同時按比例減少 App Runner 權重，開始轉移流量。建議的方法：

- 將 ECS Express 模式設定為 10 / App Runner 設定為 90
- 成功監控和驗證服務處理請求
- 增加至 25 / 75
- 增加至 50/50
- 增加至 75 / 25
- 在 100 / 0 完成

在每個步驟中，先測試應用程式，再轉移其他流量。如果在任何時候發生問題，請透過將權重調整回其先前的值來復原。

> **Important**
>
> 讓 App Runner 服務持續執行驗證期間（例如 24-48 小時），以確認 DNS 變更已全域傳播，並視需要提供轉返選項。如果遇到問題，您可以將 Route 53 權重快速還原回 App Runner。

步驟 5：完成遷移

驗證 ECS Express Mode 服務是否正確處理生產流量且驗證期間已過之後，請完成遷移：

- 在 Route 53 中，移除指向 App Runner 的加權記錄（或將其權重設定為 0)。

- 從 App Runner 服務中移除自訂網域關聯。

刪除 App Runner 服務：

```
aws apprunner delete-service --service-arn your-app-runner-service-arn
```

另請考慮移除不再需要的任何資源：

- App Runner 的 Route 53 加權路由記錄
- 來自 Amazon Elastic Container Registry 的未使用容器映像
- 如果不再需要，則專門為 App Runner 建立的 IAM 角色

> **Note**
> 如果服務是在生產環境中執行，請勿刪除 ECS Express Mode 服務、其 Application Load Balancer 或相關聯的資源。

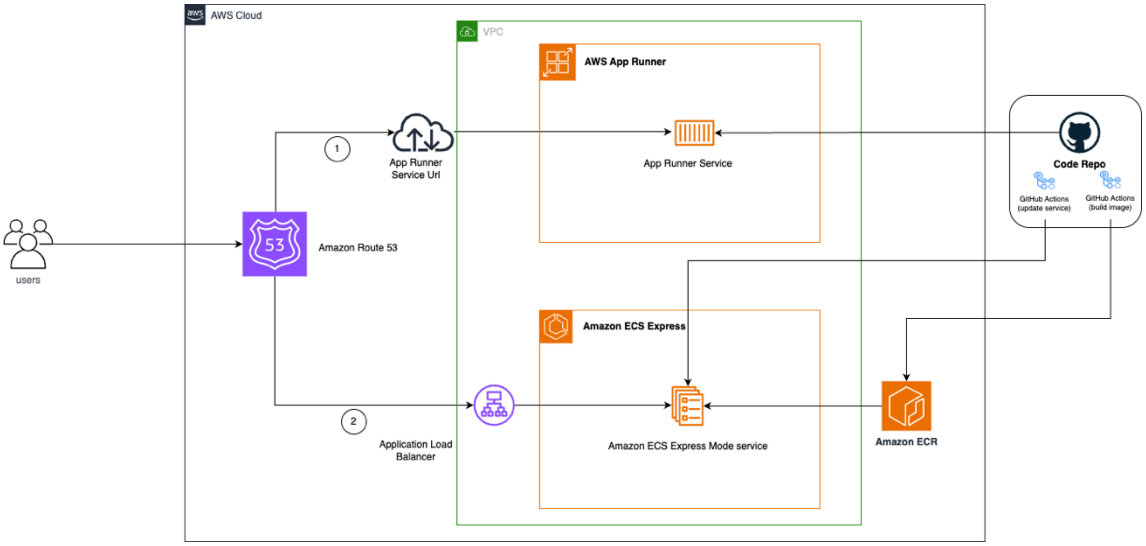
遷移以來源為基礎的部署

如果您現有的 App Runner 服務是從原始程式碼而非容器映像部署，您需要先新增容器化步驟，才能部署至 ECS Express 模式。與 App Runner 不同，ECS Express 模式需要容器映像。不過，您可以使用 CI/CD 工具複寫 App Runner 的自動化部署體驗，例如搭配 Amazon ECS Deploy Express Service GitHub Action 的 GitHub Actions。 GitHub

遷移工作流程有三個階段：

1. 使用 Dockerfile 建置容器映像
2. 將映像推送至容器登錄檔，例如 Amazon Elastic Container Registry
3. 將映像部署到 ECS Express 模式

下圖顯示此工作流程如何使用 GitHub 動作運作：



容器化您的應用程式

如果您的應用程式還沒有 Dockerfile，請在儲存庫根中建立一個。Dockerfile 做為建置原始碼並將原始碼封裝至容器映像的藍圖。

您的儲存庫結構應該包含：

```
your-app/
### src/                    # Application source code
### Dockerfile              # Container build instructions
### package.json            # Dependencies and scripts
### .github/                # GitHub configuration
    ### workflows/          # GitHub Actions workflows
        ### deploy.yml      # ECS Express Mode deployment workflow
```

設定自動部署的 GitHub 動作

若要在程式碼推送時複寫 App Runner 的自動部署，請使用下列項目設定 GitHub 動作：

- 建立 OpenID Connect (OIDC) 提供者，以允許 GitHub 動作擔任 IAM 角色
- 使用 ECS Express 模式和 Amazon Elastic Container Registry 許可建立 IAM 角色 https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_ECS.html
- 建立 ECS Express Mode 所需的兩個 IAM 角色
- 為您的 ECS 資源建立 GitHub 環境變數：`ECS_SERVICE`、`ECS_CLUSTER`、`AWS_REGION`、`AWS_ACCOUNT_ID`和 `ECR_REPOSITORY`

GitHub 動作工作流程範例

在 建立工作流程檔案`.github/workflows/deploy.yml`：

```
name: Build and Deploy to ECS

on:
  push:
    branches: [ main ]

env:
  AWS_REGION: ${{ vars.AWS_REGION }}
  AWS_ACCOUNT_ID: ${{ vars.AWS_ACCOUNT_ID }}
  ECR_REPOSITORY: ${{ vars.ECR_REPOSITORY }}
  ECS_SERVICE: ${{ vars.ECS_SERVICE }}
  ECS_CLUSTER: ${{ vars.ECS_CLUSTER }}

jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    environment: production
    permissions:
      id-token: write
      contents: read

    steps:
    - name: Checkout
      uses: actions/checkout@v6

    - name: Configure AWS credentials
      uses: aws-actions/configure-aws-credentials@v5
      with:
        aws-region: ${{ env.AWS_REGION }}
        role-to-assume: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/github-actions-ecs-
role
        role-session-name: GitHubActionsECSDeployment

    - name: Login to Amazon ECR
      id: login-ecr
      uses: aws-actions/amazon-ecr-login@v2

    - name: Get short commit hash
      run: echo "IMAGE_TAG=${GITHUB_SHA:0:7}" >> $GITHUB_ENV
```

```
    - name: Build, tag, and push image to Amazon ECR
      id: build-image
      env:
        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
      uses: docker/build-push-action@v6
      with:
        context: .
        push: true
        tags: ${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:latest,
${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}

    - name: Deploy to ECS Express Mode
      uses: aws-actions/amazon-ecs-deploy-express-service@v1
      env:
        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
      with:
        service-name: ${{ env.ECS_SERVICE }}
        image: ${{ env.ECR_REGISTRY }}/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}
        execution-role-arn: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/
ecsTaskExecutionRole
        infrastructure-role-arn: arn:aws:iam::${{ env.AWS_ACCOUNT_ID }}:role/
ecsInfrastructureRoleForExpressServices
        cluster: ${{ env.ECS_CLUSTER }}
        container-port: 8080
        environment-variables: |
          [
            {"name": "ENV", "value": "Prod"}
          ]
        cpu: '1024'
        memory: '2048'
        health-check-path: /health
        min-task-count: 1
        max-task-count: 4
        auto-scaling-metric: AVERAGE_CPU
        auto-scaling-target-value: 70
```

當您將程式碼變更推送至主分支時，GitHub Actions 會自動建立新的容器映像，將其推送至 Amazon Elastic Container Registry，並將其部署至 ECS Express Mode 服務。這會複寫您使用 App Runner 的自動化部署體驗。

一旦 ECS Express Mode 服務執行，請依照遷移演練中的步驟 3–5 來設定自訂網域、使用 DNS 路由轉移流量，並完成遷移。

其他資源

- 使用 建立您的第一個快速模式服務 AWS CLI
- 在主控台中建立您的第一個 Amazon ECS Express Mode 服務
- 在快速模式之外更新資源
- the section called “自訂網域名稱”
- Amazon Route 53 加權路由

設定 App Runner

如果您是新 AWS 客戶，請先完成此頁面列出的設定先決條件，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 AWS 帳戶

若要開始使用 AWS，您需要 AWS 帳戶。如需建立 的相關資訊 AWS 帳戶，請參閱《 AWS 帳戶管理參考指南》中的 入門 AWS 帳戶。

下一步

您已完成先決條件步驟。若要將第一個應用程式部署到 App Runner，請參閱 開始使用。

App Runner 入門

AWS App Runner 是一種 AWS 服務，提供快速、簡單且經濟實惠的方式，將現有的容器映像或原始程式碼直接轉換為 中執行的 Web 服務 AWS 雲端。

本教學課程說明如何使用 AWS App Runner 將應用程式部署至 App Runner 服務。它會逐步解說如何設定原始程式碼和部署、服務建置和服務執行時間。它也會說明如何部署程式碼版本、進行組態變更，以及檢視日誌。最後，本教學課程示範如何清除您在遵循教學課程程序時建立的資源。

主題

- 先決條件
- 步驟 1：建立 App Runner 服務
- 步驟 2：變更您的服務代碼
- 步驟 3：進行組態變更
- 步驟 4：檢視您服務的日誌
- 步驟 5：清除
- 下一步

先決條件

開始教學課程之前，請務必採取下列動作：

1. 完成 中的設定步驟設定。
2. 決定是否要使用 GitHub 儲存庫或 Bitbucket 儲存庫。
   - 若要使用 Bitbucket，如果您還沒有 Bitbucket 帳戶，請先建立 Bitbucket 帳戶。如果您是初次使用 Bitbucket，請參閱 Bitbucket Cloud 文件中的 Bitbucket 入門。
   - 若要使用 GitHub，如果您還沒有 GitHub 帳戶，請建立 GitHub 帳戶。如果您是初次使用 GitHub，請參閱 GitHub 文件中的 GitHub 入門。 GitHub

> **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，如果您想要逐步解說從 GitHub 和 Bitbucket 儲存庫進行部署，您可以重複此程序。下次透過建立新的 App Runner 服務，並為其他儲存庫提供者建立新的帳戶連線。

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`。在儲存庫的根目錄中建立檔案，其中包含下列範例中指定的名稱和內容。

`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()
```

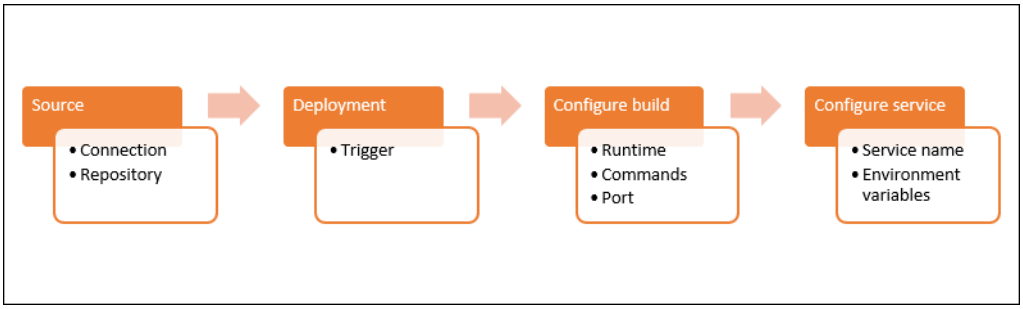
步驟 1：建立 App Runner 服務

在此步驟中，您會根據在 GitHub 或 Bitbucket 上建立的範例原始程式碼儲存庫建立 App Runner 服務，做為 的一部分the section called “先決條件”。此範例包含簡單的 Python 網站。以下是您建立服務的主要步驟：

1. 設定您的原始程式碼。
2. 設定來源部署。

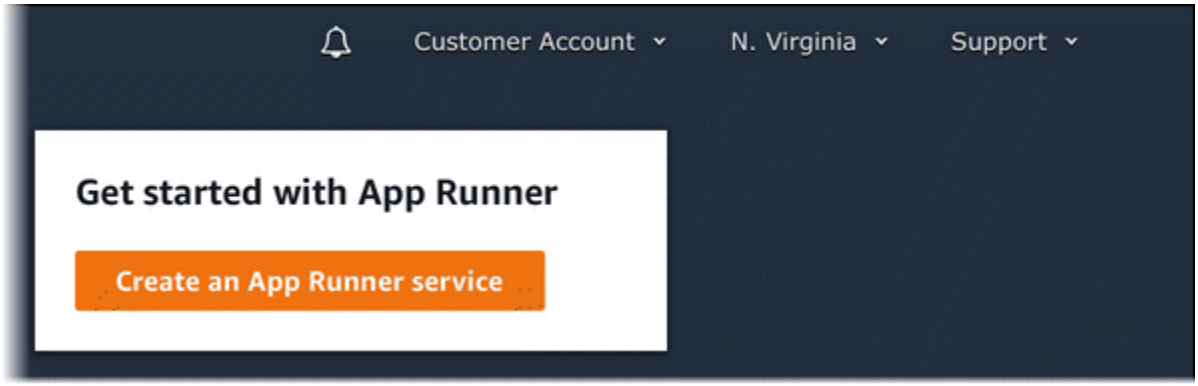
3. 設定應用程式建置。
4. 設定您的服務。
5. 檢閱並確認。

下圖概述建立 App Runner 服務的步驟：

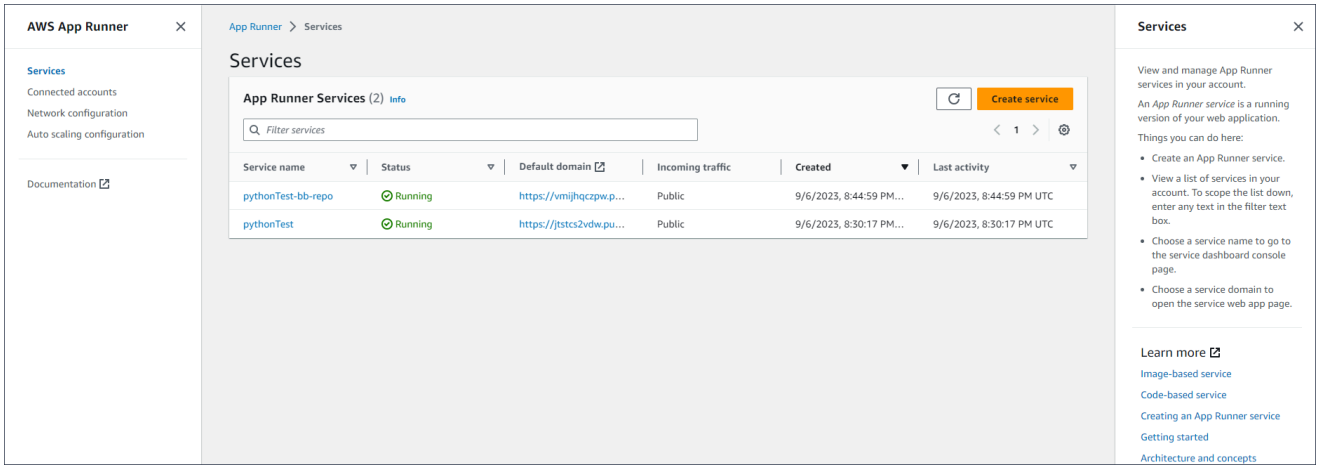


根據原始碼儲存庫建立 App Runner 服務

1. 設定您的原始程式碼。
   a. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
   b. 如果 尚 AWS 帳戶 無任何 App Runner 服務，則會顯示主控台首頁。選擇建立 App Runner 服務。



   如果 AWS 帳戶 有現有的服務，則會顯示服務頁面，其中包含您的服務清單。選擇 Create service (建立服務)。



c. 在來源和部署頁面上的來源區段中，針對儲存庫類型，選擇來源碼儲存庫。

d. 選取提供者類型。選擇 GitHub 或 Bitbucket。

e. 接下來，選擇新增。如果出現提示，請提供您的 GitHub 或 Bitbucket 登入資料。

f. 根據先前選取的提供者類型，選擇下一組步驟。

> **Note**
>
> 將 GitHub 的 AWS 連接器安裝到您的 GitHub 帳戶的下列步驟為一次性步驟。您可以重複使用連線，根據此帳戶中的儲存庫建立多個 App Runner 服務。當您有現有的連線時，請選擇它並跳至儲存庫選擇。
>
> 同樣適用於 Bitbucket 帳戶的 AWS 連接器。如果您同時使用 GitHub 和 Bitbucket 做為 App Runner 服務的原始程式碼儲存庫，則需要為每個提供者安裝一個 AWS Connector。然後，您可以重複使用每個連接器來建立更多 App Runner 服務。

- 對於 GitHub，請遵循下列步驟。

  i. 在下一個畫面上，輸入連線名稱。

  ii. 如果您第一次使用 GitHub 搭配 App Runner，請選取安裝另一個。

  iii. 在 AWS Connector for GitHub 對話方塊中，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 AWS Connector for GitHub，請選擇授權 AWS 連線。

  v. 在 Install AWS Connector for 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。您現在可以在帳戶中選擇儲存庫。

vi. 針對儲存庫，選擇您建立的範例儲存庫 `python-hello`。針對分支，選擇儲存庫的預設分支名稱（例如，主要)。

vii. 將來源目錄保留為預設值。目錄預設為儲存庫根目錄。您在先前的先決條件步驟中，將原始碼存放在儲存庫根目錄中。

- 對於 Bitbucket，請遵循下列步驟。

  i. 在下一個畫面上，輸入連線名稱。

  ii. 如果您第一次使用 Bitbucket 搭配 App Runner，請選取安裝另一個。

  iii. 在AWS CodeStar 請求存取對話方塊中，您可以選取工作區，並授予對 AWS CodeStar 的存取權以進行 Bitbucket 整合。選取您的工作區，然後選取授予存取權。

  iv. 接下來，系統會將您重新導向至 AWS 主控台。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. 針對儲存庫，選擇您建立的範例儲存庫 `python-hello`。針對分支，選擇儲存庫的預設分支名稱（例如，主要)。

  vi. 將來源目錄保留為預設值。目錄預設為儲存庫根目錄。您在先前的先決條件步驟中，將原始碼存放在儲存庫根目錄中。

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，每個新的遞交到您的儲存庫來源目錄都會自動部署您服務的新版本。

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. 設定應用程式建置。

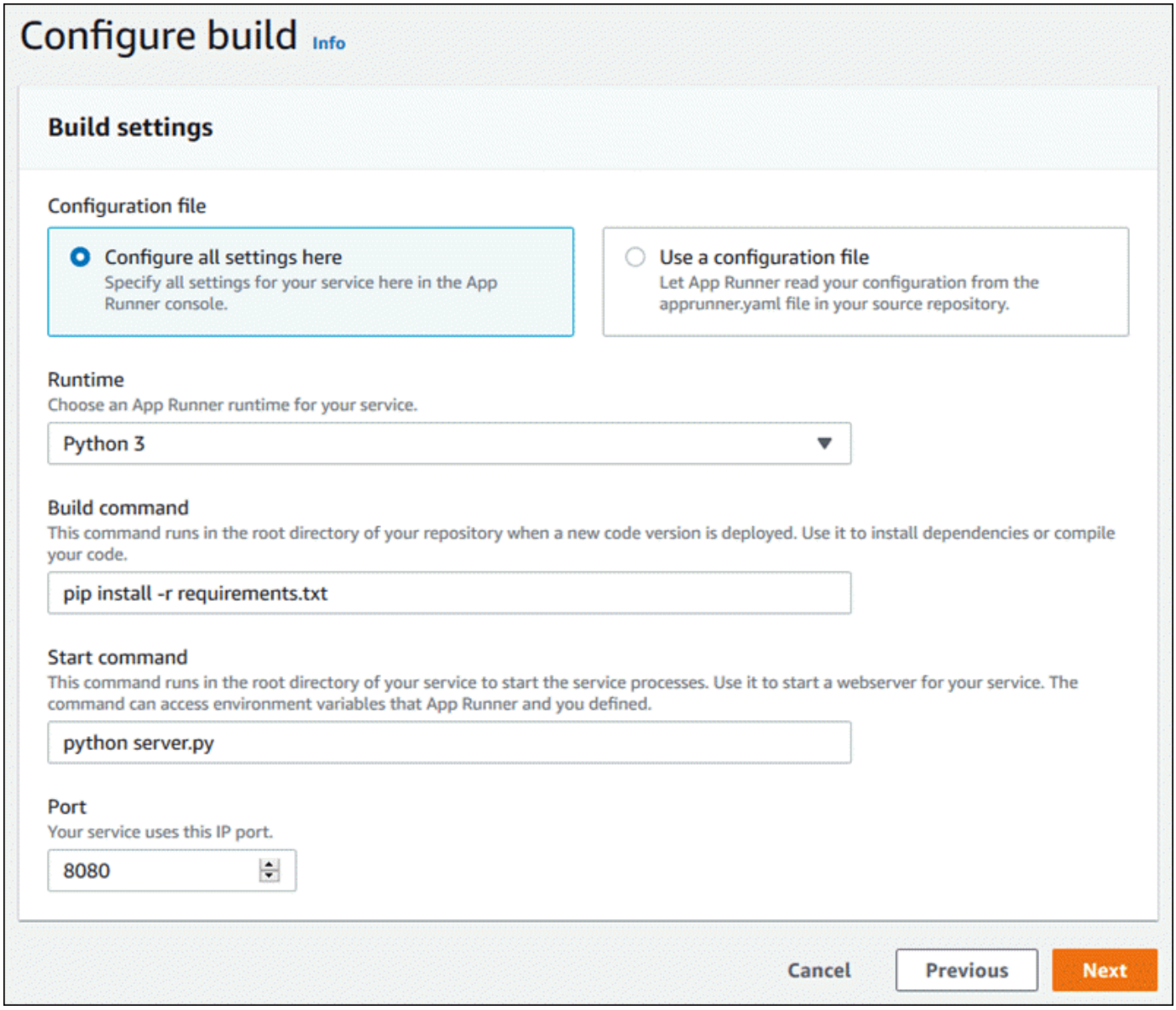
   a. 在設定建置頁面上，針對組態檔案，選擇此處設定所有設定。

   b. 提供下列建置設定：

      - 執行時間 – 選擇 Python 3。
      - 組建命令 – 輸入 **`pip install -r requirements.txt`**。
      - Start 命令 – 輸入 **`python server.py`**。
      - 連接埠 – 輸入 **`8080`**。

   c. 選擇下一步。

> **Note**
>
> Python 3 執行時間會使用基本 Python 3 映像和您的範例 Python 程式碼來建置 Docker 映像。然後，它會啟動執行此映像之容器執行個體的服務。



4. 設定您的服務。

   a. 在設定服務頁面上的服務設定區段中，輸入服務名稱。

   b. 在環境變數下，選取新增環境變數。提供環境變數的下列值。

      - 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*

A tag is a key-value pair that you assign to an AWS resource.

No tags associated with the resource.

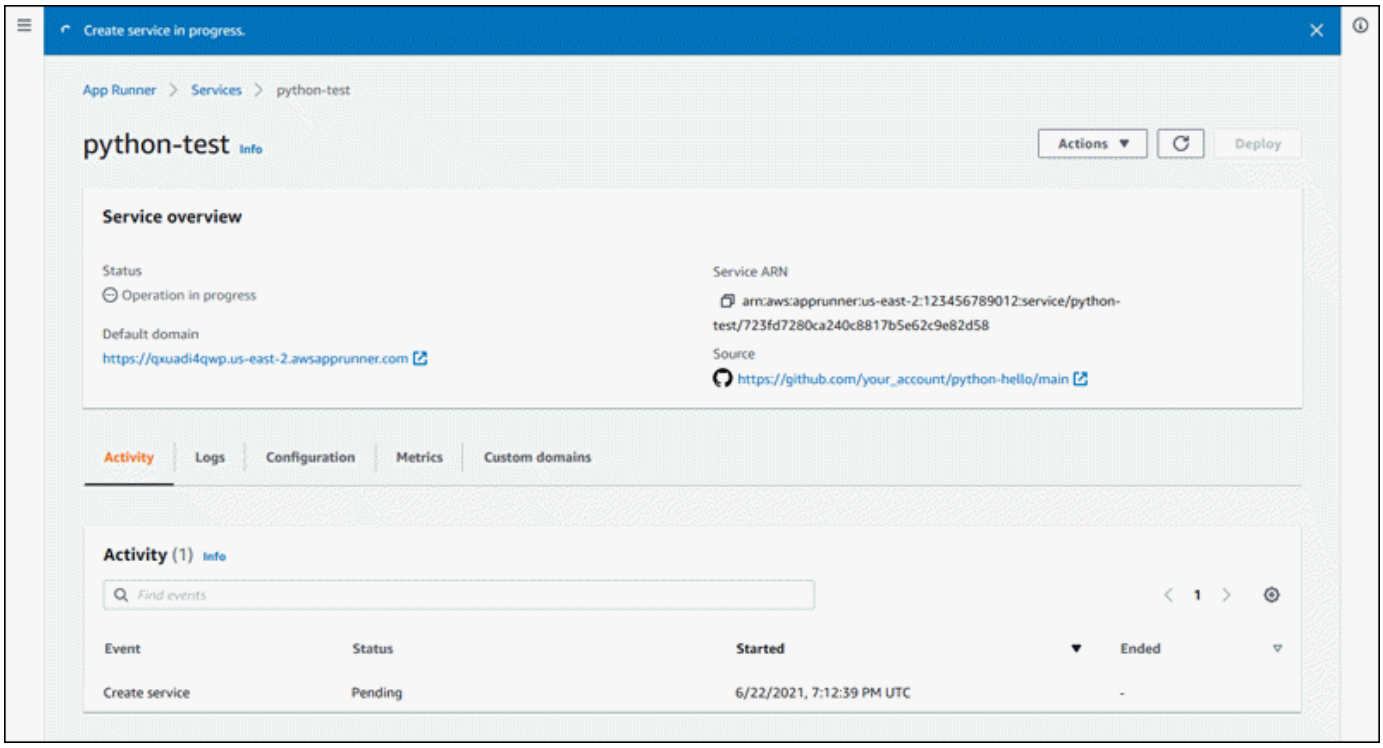
Add new tag

You can add 50 more tags.

Cancel | Previous | Next

5. 在檢閱和建立頁面上，驗證您輸入的所有詳細資訊，然後選擇建立和部署。

   如果服務已成功建立，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 App Runner 應用程式的安全性，在公有字尾清單 (PSL) 中註冊 *.awsapprunner.com 網域。為了進一步提高安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用具有__Host-字首的 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。

   網頁隨即顯示：您好，**####**！

步驟 2：變更您的服務代碼

在此步驟中，您會在儲存庫來源目錄中變更程式碼。App Runner CI/CD 功能會自動建置變更並將其部署到您的服務。

變更您的服務程式碼

1. 導覽至您的範例儲存庫。
2. 編輯名為 的檔案`server.py`。
3. 在指派給變數 的表達式中`message`，將文字變更為 `Hello Good morning`。
4. 儲存變更並將其遞交至儲存庫。
5. 下列步驟說明變更 GitHub 儲存庫中的服務程式碼。
   a. 導覽至您的範例 GitHub 儲存庫。
   b. 選擇檔案名稱`server.py`以導覽至該檔案。
   c. 選擇編輯此檔案（鉛筆圖示）。
   d. 在指派給變數 的表達式中`message`，將文字變更為 `Hello Good morning`。

e. 選擇 Commit changes (遞交變更)。

6. 新的遞交會開始為您的 App Runner 服務部署。在服務儀表板頁面上，服務狀態會變更為進行中的操作。

   等待部署結束。在服務儀表板頁面上，服務狀態應變更回執行中。

7. 確認部署成功：重新整理顯示服務網頁的瀏覽器索引標籤。

   頁面現在會顯示修改後的訊息：早安，**####**！

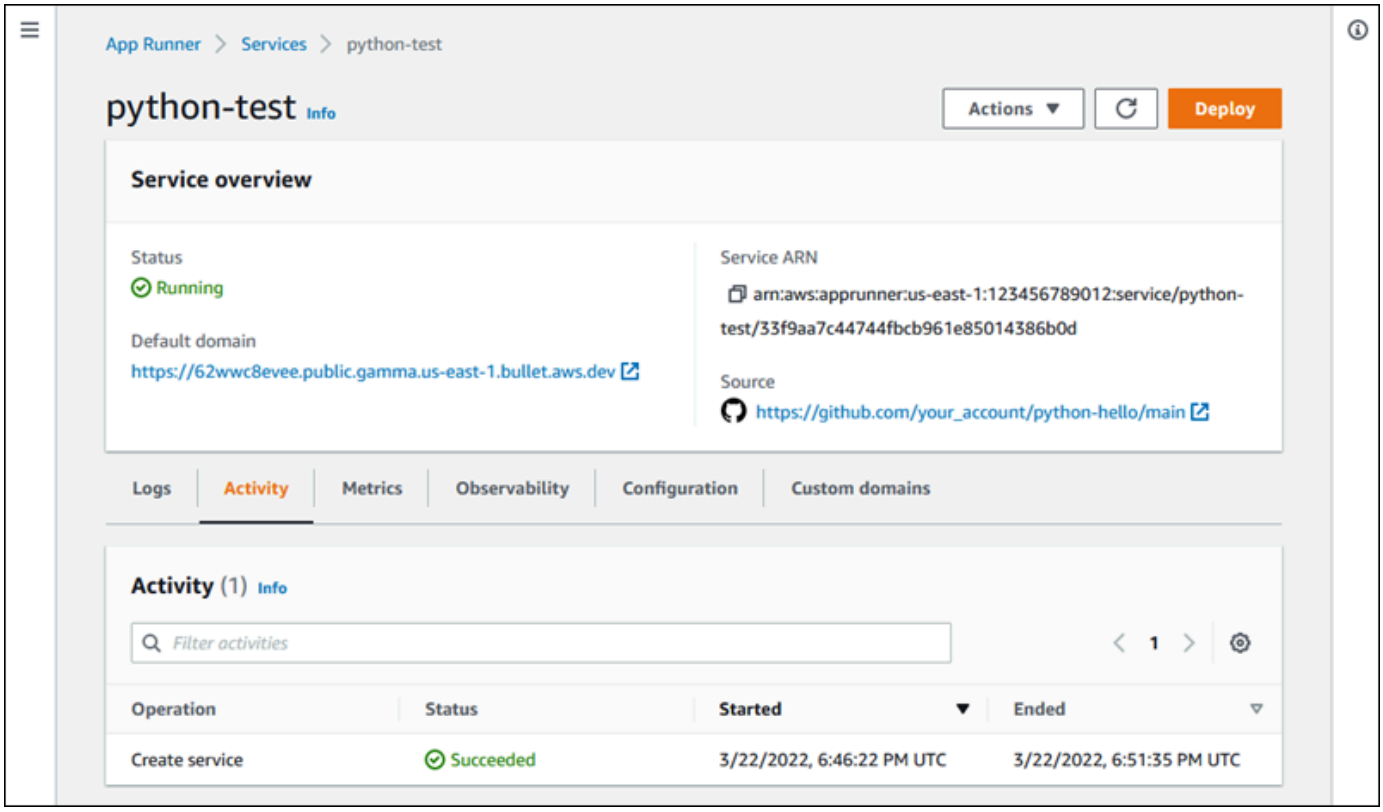
步驟 3：進行組態變更

在此步驟中，您會變更**NAME**環境變數值，以示範服務組態變更。

變更環境變數值

1. 開啟 App Runner 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，然後選擇您的 App Runner 服務。

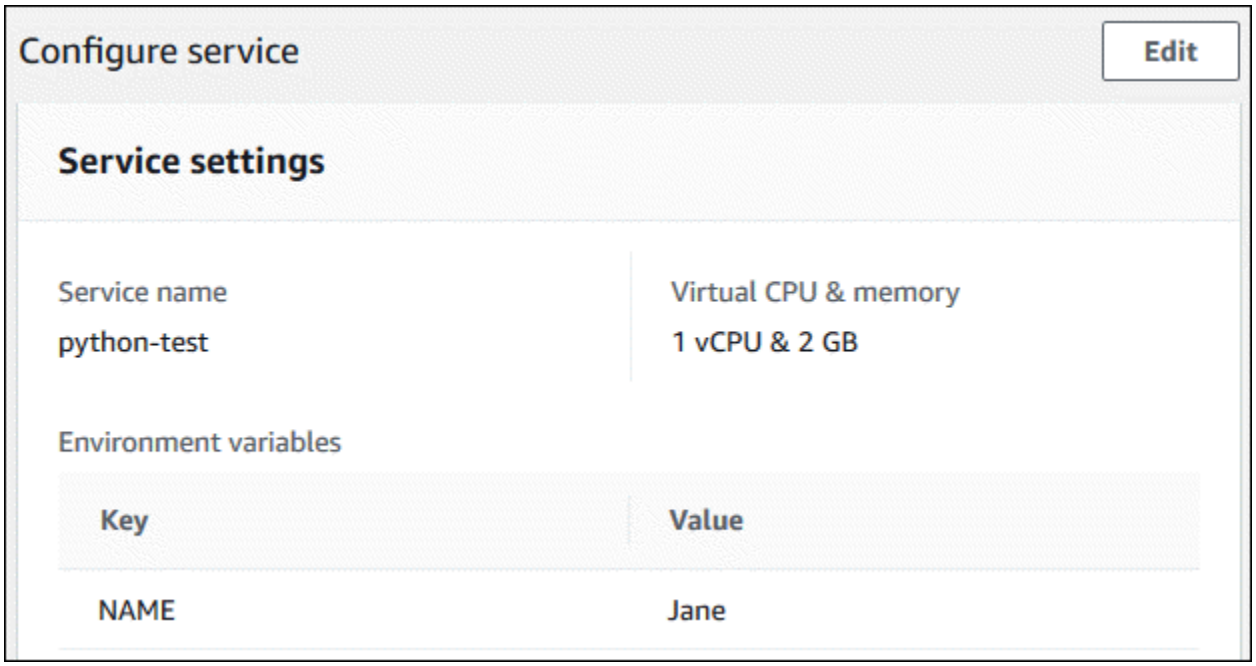
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇組態索引標籤。

   主控台會以數個區段顯示您的服務組態設定。

4. 在設定服務區段中，選擇編輯。



5. 對於具有金鑰 的環境變數**NAME**，請將值變更為不同的名稱。

6. 選擇 Apply changes (套用變更)。

   App Runner 會啟動更新程序。在服務儀表板頁面上，服務狀態會變更為進行中的操作。

7. 等待更新結束。在服務儀表板頁面上，服務狀態應變更回執行中。

8. 確認更新成功：重新整理顯示服務網頁的瀏覽器索引標籤。

   頁面現在會顯示修改後的名稱：早安，**###**！

步驟 4：檢視您服務的日誌

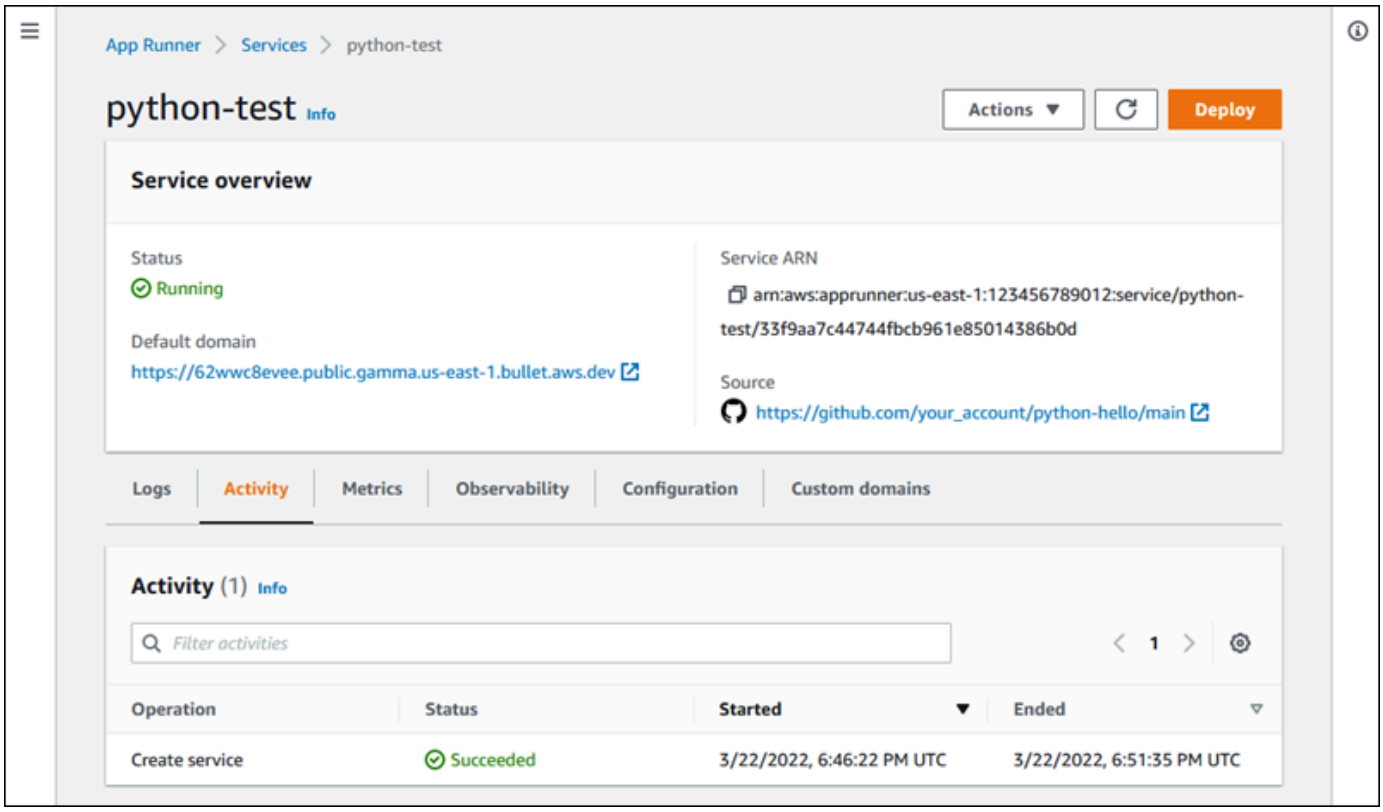
在此步驟中，您可以使用 App Runner 主控台來檢視 App Runner 服務的日誌。App Runner 會將日誌串流至 Amazon CloudWatch Logs (CloudWatch Logs)，並在您服務的儀表板上顯示日誌。如需 App Runner 日誌的資訊，請參閱 the section called “日誌 (CloudWatch Logs)”。

檢視 服務的日誌

1. 開啟 App Runner 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，然後選擇您的 App Runner 服務。

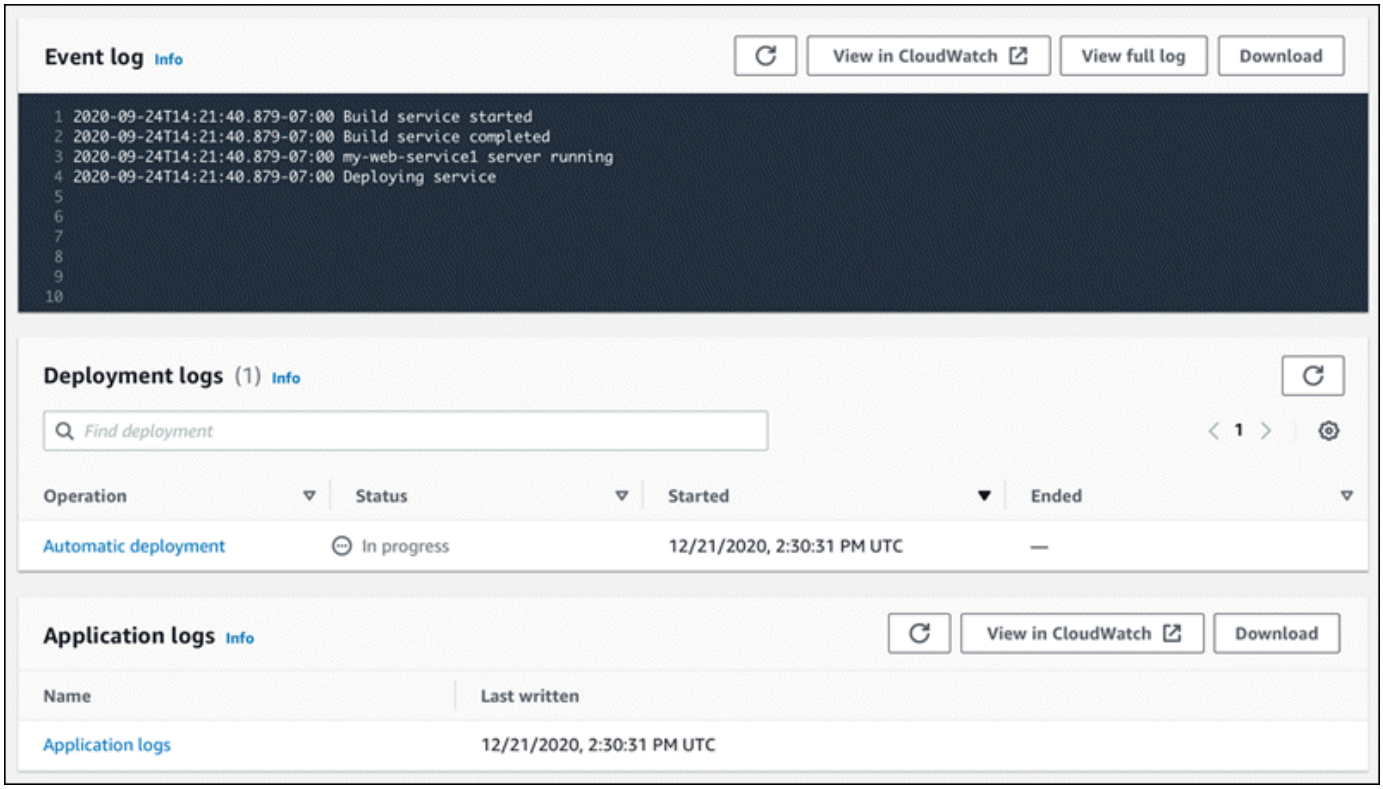
   主控台會顯示服務儀表板，其中包含服務概觀。



3. 在服務儀表板頁面上，選擇日誌索引標籤。

   主控台會在數個區段中顯示幾種類型的日誌：

   - 事件日誌 – App Runner 服務生命週期中的活動。主控台會顯示最新的事件。
   - 部署日誌 – 將來源儲存庫部署到您的 App Runner 服務。主控台會顯示每個部署的個別日誌串流。
   - 應用程式日誌 – 部署至 App Runner 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合併為單一日誌串流。



4. 若要尋找特定部署，請輸入搜尋詞彙，縮小部署日誌清單的範圍。您可以搜尋出現在資料表中的任何值。
5. 若要檢視日誌的內容，請選擇檢視完整日誌（事件日誌）或日誌串流名稱（部署和應用程式日誌）。
6. 選擇下載以下載日誌。對於部署日誌串流，請先選取日誌串流。
7. 選擇在 CloudWatch 中檢視以開啟 CloudWatch 主控台，並使用其完整功能來探索您的 App Runner 服務日誌。對於部署日誌串流，請先選取日誌串流。

> **Note**
> 如果您想要檢視特定執行個體的應用程式日誌，而非合併的應用程式日誌，CloudWatch 主控台特別有用。

步驟 5：清除

您現在已了解如何建立 App Runner 服務、檢視日誌，以及進行一些變更。在此步驟中，您刪除服務以移除不再需要的資源。

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，輸入請求的文字，然後選擇刪除。

   結果：主控台導覽至服務頁面。您剛刪除的服務會顯示刪除狀態。不久後，它會從清單中消失。

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

現在您已部署第一個 App Runner 服務，請進一步了解下列主題：

- 架構和概念 – 與 App Runner 相關的架構、主要概念 AWS 和資源。
- 影像型服務 和 程式碼型服務 – App Runner 可以部署的兩種應用程式來源類型。
- 開發 App Runner – 開發或遷移應用程式程式碼以部署至 App Runner 時應該知道的事項。
- App Runner 主控台 – 使用 App Runner 主控台管理和監控您的服務。
- 管理您的服務 – 管理 App Runner 服務的生命週期。
- 可觀測性 – 透過監控指標、讀取日誌、處理事件、追蹤服務動作呼叫，以及追蹤 HTTP 呼叫等應用程式事件，來取得 App Runner 服務操作的可見性。
- App Runner 組態檔案 – 以組態為基礎的方法，可指定 App Runner 服務的建置和執行時間行為選項。

- App Runner API – 使用 App Runner 應用程式程式設計界面 (API) 來建立、讀取、更新和刪除 App Runner 資源。
- 安全 – AWS 和您在使用 App Runner 和其他 服務時確保雲端安全性的不同方式。

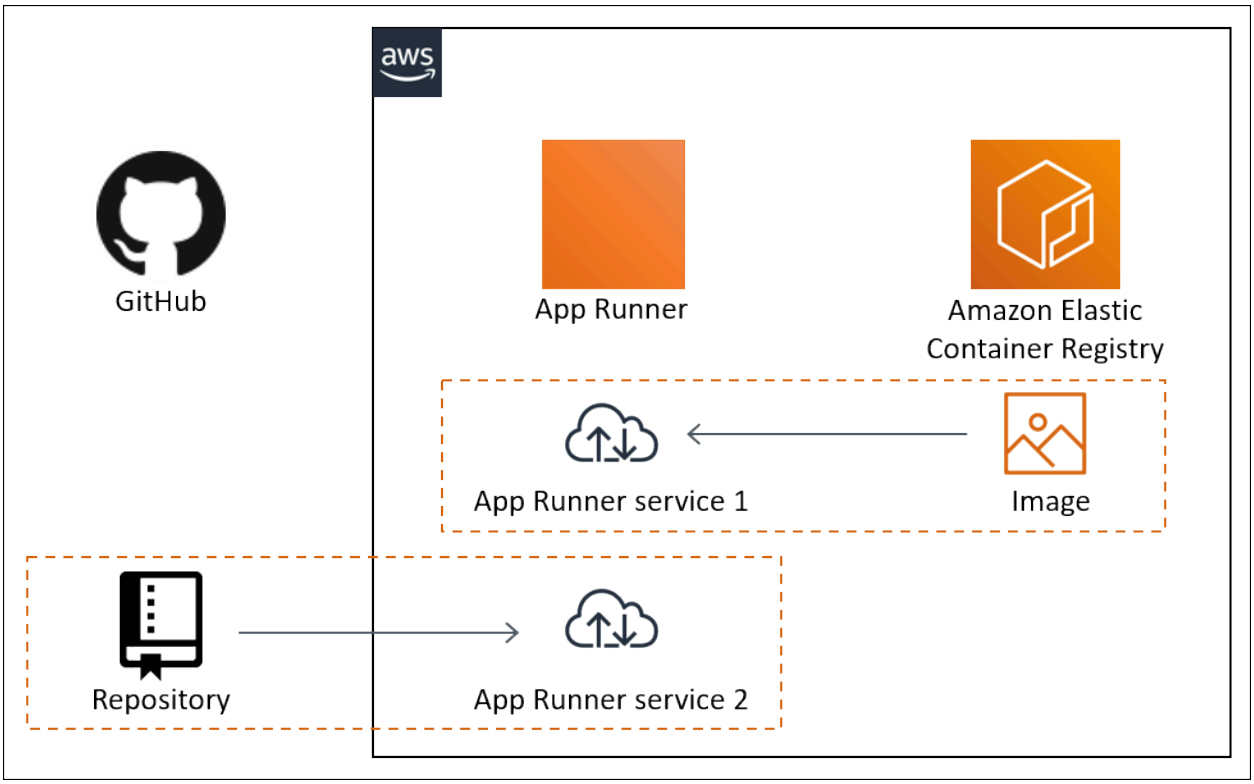
App Runner 架構和概念

AWS App Runner 從儲存庫取得您的原始碼或原始映像，並在 中為您建立和維護執行中的 Web 服務 AWS 雲端。一般而言，您只需要呼叫一個 App Runner 動作 CreateService 即可建立您的服務。

透過來源映像儲存庫，您可以提供 App Runner 可以部署以執行 Web 服務的ready-to-use容器映像。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，並以特定執行時間環境為目標。App Runner 支援多個程式設計平台，每個平台都有一或多個平台主要版本的受管執行時間。

此時，App Runner 可以從 Bitbucket 或 GitHub 儲存庫擷取來源碼，也可以從 中的 Amazon Elastic Container Registry (Amazon ECR) 擷取來源映像 AWS 帳戶。

下圖顯示 App Runner 服務架構的概觀。在圖表中，有兩個範例服務：一個從 GitHub 部署來源碼，另一個從 Amazon ECR 部署來源映像。相同的流程適用於 Bitbucket 儲存庫。



App Runner 概念

以下是與在 App Runner 中執行之 Web 服務相關的重要概念：

- App Runner 服務 – App Runner 根據其原始碼儲存庫或容器映像用來部署和管理應用程式 AWS 的資源。App Runner 服務是您應用程式的執行版本。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 服務提供的來源儲存庫類型：來源碼或來源映像。
- 儲存庫提供者 – 包含應用程式來源的儲存庫服務（例如 GitHub、Bitbucket 或 Amazon ECR)。
- App Runner 連線 – 一種 AWS 資源，可讓 App Runner 存取儲存庫提供者帳戶（例如 GitHub 帳戶或組織）。如需連線的相關資訊，請參閱the section called “連線”。
- 執行時間 – 部署原始程式碼儲存庫的基礎映像。App Runner 為不同的程式設計平台和版本提供各種受管執行時間。如需詳細資訊，請參閱程式碼型服務。
- 部署 – 將來源儲存庫版本（程式碼或映像）套用至 App Runner 服務的動作。服務的第一個部署會做為服務建立的一部分進行。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。您可以設定 App Runner 服務，以自動建置（適用於原始程式碼）並部署應用程式的每個版本，如其在儲存庫中所示。這可以是來源碼儲存庫中的新遞交，或來源映像儲存庫中的新映像版本。
  - 手動部署 – 您明確啟動之 App Runner 服務的部署。
- 自訂網域 – 與 App Runner 服務建立關聯的網域。Web 應用程式的使用者可以使用此網域來存取 Web 服務，而不是預設的 App Runner 子網域。如需詳細資訊，請參閱the section called “自訂網域名稱”。

> **Note**
>
> 為了增強 App Runner 應用程式的安全性，在公有尾碼清單 (PSL) 中註冊 *.awsapprunner.com 網域。為了進一步提高安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用具有`__Host-`字首的 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 服務的基礎設施上執行的活動。進行維護時，服務狀態會暫時變更為 `OPERATION_IN_PROGRESS`（主控台中的操作進行中) 幾分鐘。在此期間，服務上的動作（例如，部署、組態更新、暫停/繼續或刪除）會遭到封鎖。服務狀態回到 時，請在幾分鐘後再次嘗試動作`RUNNING`。

> **Note**
>
> 如果您的動作失敗，並不表示 App Runner 服務已關閉。您的應用程式處於作用中狀態，並持續處理請求。您的服務不太可能遇到任何停機時間。

特別是，如果 App Runner 在託管服務的基礎硬體中偵測到問題，則會遷移您的服務。為了防止任何服務停機，App Runner 會將您的服務部署到一組新的執行個體，並將流量轉移到這些執行個體（藍綠部署）。您偶爾可能會看到費用稍微暫時增加。

App Runner 支援的組態

當您設定 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。您需根據您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	12 GB

App Runner 資源

當您使用 App Runner 時，您可以在 中建立和管理幾種類型的資源 AWS 帳戶。這些資源用於存取您的程式碼和管理 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版本。本指南其餘大部分都說明服務類型、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 服務提供由第三方供應商存放之私有儲存庫的存取權。作為跨多個服務共用的個別資源存在。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 服務提供可控制應用程式自動擴展的設定。作為跨多個服務共用的個別資源存在。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 App Runner 服務設定其他應用程式可觀測性功能。作為跨多個服務共用的個別資源存在。如需可觀測性組態的詳細資訊，請參閱 the section called “可觀測性組態”。

資源名稱	Description
	ARN：`arn:aws:apprunner:` *region*`:`*account-id* `:observability configuration/` *config-name* `[/`*config-revision* `[/`*config-id* `]]`
VpcConnector	設定 App Runner 服務的 VPC 設定。作為跨多個服務共用的個別資源存在。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	這是用來設定傳入流量 AWS App Runner 的資源。它會在 VPC 介面端點與 App Runner 服務之間建立連線，讓您的 App Runner 服務只能從 Amazon VPC 內存取。如需 VPCIngressConnection 功能的詳細資訊，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* `]]`

App Runner 資源配額

AWS 對您的帳戶施加一些配額（也稱為限制），以用於每個 AWS 帳戶的資源用量 AWS 區域。下表列出與 App Runner 資源相關的配額。配額也會列在 的AWS App Runner 端點和配額中AWS 一般參考。

資源配額	Description	預設值	可調整？
Services	您可以在帳戶中為每個 建立的 服務數量上限 AWS 區域。	30	✓ 是
Connections	您可以在帳戶中為每個連線建立的連線數目上限 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	可調整？
Auto scaling configurations	names	您可以在帳戶中為每個 建立的自動擴展組態中擁有的唯一名稱數目上限 AWS 區域。您可以在多個 服務中使用單一自動擴展組態。	10	✓ 是
	每個名稱的修訂	您可以為每個唯一名稱在帳戶中為每個 建立 AWS 區域 的自動擴展組態修訂數目上限。您可以在多個服務中使用單一自動擴展組態修訂。	5	× 否
Observability configurations	names	您可以在帳戶中為每個 建立的可觀測性組態中擁有的唯一名稱數目上限 AWS 區域。您可以在多個服務中使用單一可觀測性組態。	10	✓ 是
	每個名稱的修訂	您可以為每個 AWS 區域 唯一名稱在帳戶中為每個 建立的可觀測性組態修訂數目上限。您可以在多個服務中使用單一可觀測性組態修訂。	10	× 否
VPC connectors		您可以在帳戶中為每個連接器建立的 VPC 連接器數量上限 AWS 區域。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 VPC 輸入連線建立的最大數量 AWS 區域。您可以使用單一 VPC 輸入連線來存取多個 App Runner 服務。	1	× 否

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

以來源映像為基礎的 App Runner 服務

您可以使用 根據兩種基本不同的服務來源類型 AWS App Runner 來建立和管理服務：來源碼和來源映像。無論來源類型為何，App Runner 都會負責啟動、執行、擴展和負載平衡您的服務。您可以使用 App Runner 的 CI/CD 功能來追蹤來源映像或程式碼的變更。當 App Runner 發現變更時，它會自動建置（適用於原始程式碼）並將新版本部署到您的 App Runner 服務。

本章討論以來源映像為基礎的 服務。如需以來源碼為基礎的 服務資訊，請參閱 程式碼型服務。

來源映像是存放在映像儲存庫中的公有或私有容器映像。您可以將 App Runner 指向映像，並啟動根據此映像執行容器的服務。不需要建置階段。反之，您會提供ready-to-deploy的映像。

> **Note**
>
> 提供容器映像時，您必須負責定期更新和修補這些映像。App Runner 管理基礎設施時，您應該確保所提供容器映像的安全性和up-to-date狀態。如需詳細資訊，請參閱 AWS App Runner 文件

映像儲存庫供應商

App Runner 支援下列映像儲存庫提供者：

- Amazon Elastic Container Registry (Amazon ECR) – 儲存私有至 的映像 AWS 帳戶。
- Amazon Elastic Container Registry Public (Amazon ECR Public) – 存放可公開讀取的映像。

供應商使用案例

- 使用您 AWS 帳戶中存放在 Amazon ECR 中的映像
- 在不同 AWS 帳戶中使用存放在 Amazon ECR 中的映像
- 使用存放在 Amazon ECR Public 中的映像

使用您 AWS 帳戶中存放在 Amazon ECR 中的映像

Amazon ECR 會將映像存放在儲存庫中。有私有和公有儲存庫。若要從私有儲存庫將映像部署至 App Runner 服務，App Runner 需要從 Amazon ECR 讀取映像的許可。若要提供該許可給 App Runner，

您需要為 App Runner 提供 存取角色。這是具有必要 Amazon ECR 動作許可的 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，您可以選擇讓 App Runner 主控台根據受管政策為您建立角色。

當您使用 App Runner API 或 時 AWS CLI，您會完成兩個步驟的程序。首先，您可以使用 IAM 主控台來建立存取角色。您可以使用 App Runner 提供的受管政策，或輸入您自己的自訂許可。然後，您可以使用 CreateService API 動作在建立服務期間提供存取角色。

如需建立 App Runner 服務的相關資訊，請參閱 the section called “建立”。

在不同 AWS 帳戶中使用存放在 Amazon ECR 中的映像

建立 App Runner 服務時，您可以使用存放在 Amazon ECR 儲存庫中的映像，該儲存庫屬於您服務所在帳戶以外的 AWS 帳戶。使用跨帳戶映像時，除了上一節關於相同帳戶映像中列出的映像之外，還有幾個額外的考量。

- 跨帳戶儲存庫應附加政策。儲存庫政策為您的存取角色提供讀取儲存庫中映像的許可。為此使用下列政策。使用存取角色的 Amazon Resource Name (ARN) 取代 `Principal`元素中的角色。

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::123456789012:role/MyApplicationRole"},
        "Action": [
          "ecr:BatchGetImage",
          "ecr:DescribeImages",
          "ecr:GetDownloadUrlForLayer"
          ],
          "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/*"
      }
    ]
  }
  ```

  如需有關將儲存庫政策連接至 Amazon ECR 儲存庫的資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的設定儲存庫政策陳述式。

- App Runner 不支援在您服務所在的不同帳戶中自動部署 Amazon ECR 映像。

使用存放在 Amazon ECR Public 中的映像

Amazon ECR Public 存放可公開讀取的映像。以下是 Amazon ECR 和 Amazon ECR Public 之間的主要差異，您應在 App Runner 服務內容中了解這些差異：

- Amazon ECR Public 映像可公開讀取。當您根據 Amazon ECR Public 映像建立服務時，不需要提供存取角色。儲存庫不需要連接任何政策。
- App Runner 不支援 Amazon ECR Public 映像的自動（連續）部署。

直接從 Amazon ECR Public 啟動服務

您可以直接啟動託管在 Amazon ECR Public Gallery 上之相容 Web 應用程式的容器映像，做為在 App Runner 上執行的 Web 服務。瀏覽圖庫時，請在圖庫頁面上尋找使用 App Runner 啟動影像。具有此選項的映像與 App Runner 相容。如需圖庫的詳細資訊，請參閱《Amazon ECR Public 使用者指南》中的使用 Amazon ECR Public Gallery。

將圖庫映像啟動為 App Runner 服務

1. 在映像的圖庫頁面上，選擇使用 App Runner 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，其中大部分所需的新服務詳細資訊已預先填入。
2. 如果您想要在主控台顯示的 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. 針對連接埠，輸入映像應用程式接聽的連接埠號碼。您通常可以在映像的圖庫頁面上找到它。
4. 或者，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。

影像範例

App Runner 團隊會在 Amazon ECR Public Gallery 中維護 hello-app-runner 範例映像。您可以使用此範例來開始建立以映像為基礎的 App Runner 服務。如需詳細資訊，請參閱 hello-app-runner。

以原始程式碼為基礎的 App Runner 服務

您可以使用 根據兩種基本不同的服務來源類型 AWS App Runner 來建立和管理服務：來源碼和來源映像。無論來源類型為何，App Runner 都會負責啟動、執行、擴展和負載平衡您的服務。您可以使用 App Runner 的 CI/CD 功能來追蹤來源映像或程式碼的變更。當 App Runner 發現變更時，它會自動建置（適用於原始程式碼）並將新版本部署到您的 App Runner 服務。

本章討論以來源碼為基礎的 服務。如需以來源映像為基礎的服務資訊，請參閱 影像型服務。

原始程式碼是 App Runner 為您建置和部署的應用程式程式碼。您可以將 App Runner 指向程式碼儲存庫中的來源目錄，然後選擇對應於程式設計平台版本的合適執行時間。App Runner 會根據執行時間的基礎映像和您的應用程式程式碼來建置映像。然後，它會啟動根據此映像執行容器的服務。

App Runner 提供便利的平台特定受管執行時間。這些執行時間中的每一個都會從您的原始程式碼建置容器映像，並將語言執行時間相依性新增至映像。您不需要提供容器組態和建置指示，例如 Dockerfile。

本章的子主題討論 App Runner 支援的各種平台：為不同的程式設計環境和版本提供受管執行期的受管平台。

主題

- 原始程式碼儲存庫提供者
- 來源目錄
- App Runner 受管平台
- 受管執行時間版本的終止支援
- 受管執行期版本和 App Runner 組建
- 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

原始程式碼儲存庫提供者

App Runner 透過從原始程式碼儲存庫讀取原始程式碼來部署原始程式碼。App Runner 支援兩個原始程式碼儲存庫提供者：GitHub 和 Bitbucket。

從原始碼儲存庫供應商部署

若要從原始碼儲存庫將原始碼部署至 App Runner 服務，App Runner 會建立與其的連線。當您使用 App Runner 主控台建立服務時，您會提供連線詳細資訊和來源目錄，讓 App Runner 部署您的原始程式碼。

連線

您可以在服務建立程序中提供連線詳細資訊。當您使用 App Runner API 或 時 AWS CLI，連線是個別的資源。首先，您可以使用 CreateConnection API 動作建立連線。然後，您可以使用 CreateService API 動作在建立服務期間提供連線的 ARN。

來源目錄

當您建立服務時，您也會提供來源目錄。根據預設，App Runner 會使用儲存庫的根目錄做為來源目錄。來源目錄是原始碼儲存庫中存放應用程式原始碼和組態檔案的位置。建置和啟動命令也會從來源目錄執行。當您使用 App Runner API 或 AWS CLI 來建立或更新服務時，您會在 CreateService 和 UpdateService API 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 App Runner 服務的詳細資訊，請參閱 the section called “建立”。如需 App Runner 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的原始碼和組態檔案的儲存庫目錄路徑。App Runner 會從您提供的來源目錄路徑執行建置和啟動命令。

輸入來源目錄路徑的值做為根儲存庫目錄的絕對值。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。

除了頂層儲存庫目錄之外，您還可以選擇提供不同的來源目錄路徑。這支援單儲存庫架構，這表示多個應用程式的原始碼存放在一個儲存庫中。若要從單一單一儲存庫建立和支援多個 App Runner 服務，請在建立每個服務時指定不同的來源目錄。

> **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`apprunner.yaml`組態檔案來定義您的服務參數，請將它放在儲存庫的來源目錄資料夾中。

如果部署觸發選項設定為自動，則您在來源目錄中遞交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。請務必了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，請參閱中的自動化部署部署方法。

> **Note**
>
> 如果您的 App Runner 服務使用 PHP 受管執行期，而且您想要指定預設根儲存庫以外的來源目錄，請務必使用正確的 PHP 執行期版本。如需詳細資訊，請參閱使用 PHP 平台。

App Runner 受管平台

App Runner 受管平台為各種程式設計環境提供受管執行期。每個受管執行期都可讓您根據程式設計語言或執行期環境的版本輕鬆建置和執行容器。當您使用受管執行期時，App Runner 會從受管執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，並包含語言執行期套件，以及一些工具和熱門的相依性套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

受管執行時間版本的終止支援

當受管語言執行時間的官方提供者或社群正式宣告版本為生命週期結束 (EOL) 時，App Runner 會接著宣告版本狀態為終止支援。如果您的服務在已達到終止支援的受管語言執行時間版本上執行，則適用下列政策和建議。

語言執行時間版本的終止支援：

- 現有的服務將繼續執行並為流量提供服務，即使他們使用的執行時間已達到終止支援。不過，它們將在不再接收更新、安全修補程式或技術支援的不支援執行時間上執行。
- 仍然允許更新使用終止支援執行時間的現有服務，但不建議繼續為服務使用終止支援執行時間。
- 無法使用已達終止支援日期的執行時間建立新服務。

具有終止支援狀態的語言版本所需的動作：

- 如果您的服務是以來源映像為基礎，則不需要對該服務採取進一步的動作。
- 如果您的服務是以原始程式碼為基礎，請更新您的服務組態以使用支援的執行時間版本。若要這樣做，請在 App Runner 主控台中選取支援的執行時間版本、更新 apprunner.yaml 組態檔案中`runtime`的欄位，或使用 CreateService/UpdateService API 操作或 IaC 工具來設定 `runtime` 參數。如需支援的執行時間清單，請參閱本章中任何特定執行時間的版本資訊頁面。
- 或者，您可以切換到 App Runner 的容器映像來源選項。如需詳細資訊，請參閱影像型服務。

> **Note**
>
> 如果您要從 Node.js 12、14 或 16 移至 Node.js 22，或從 Python 3.7 或 3.8 移至 Python 3.11，請注意 Node.js 22 和 Python 3.11 會使用修訂過的 App Runner 建置程序，以提供更快且更有效率的建置。若要在升級之前確保相容性，建議您檢閱下一節中的建置程序指引。

下表列出具有指定終止支援日期的 App Runner 受管執行期版本。

執行時間版本	App Runner 終止支援日期
Python 3.8 支援的執行時間	2025 年 12 月 1 日
Python 3.7 支援的執行時間	2025 年 12 月 1 日

執行時間版本	App Runner 終止支援日期
Node.js 18 支援的執行時間	2025 年 12 月 1 日
Node.js 16 支援的執行時間	2025 年 12 月 1 日
Node.js 14 支援的執行時間	2025 年 12 月 1 日
Node.js 12 支援的執行時間	2025 年 12 月 1 日
.NET 6 *	2025 年 12 月 1 日
PHP 8.1 *	2025 年 12 月 31 日
Ruby 3.1 *	2025 年 12 月 1 日
Go 1 *	2025 年 12 月 1 日

* App Runner 不會為標示星號 (*) 的執行時間發行任何新語言版本。這些執行時間如下：.NET、PHP、Ruby 和 Go。如果您為這些執行時間設定了程式碼型服務，我們建議您執行下列其中一個動作：

- 如果適用，請將您的服務組態切換到不同的受支援受管執行時間。
- 或者，使用您偏好的執行時間版本建置自訂容器映像，並使用 App Runner 影像型服務的選項進行部署。您可以在 Amazon ECR 中託管映像。

受管執行期版本和 App Runner 組建

App Runner 為在較近期主要版本執行時間上執行的應用程式提供更新的建置程序。此修訂後的建置程序更快速且更有效率。它也會建立佔用空間較小的最終映像，只包含執行應用程式所需的原始程式碼、建置成品和執行時間。

我們將較新的建置程序稱為修訂後的 App Runner 組建，並將原始建置程序稱為原始 App Runner 組建。為了避免破壞舊版執行時間平台的變更，App Runner 只會將修訂後的組建套用至特定執行時間版本，通常是新發行的主要版本。

我們已將新元件引入 `apprunner.yaml`組態檔案，讓修訂後的組建與非常特定的使用案例回溯相容，並提供更多彈性來設定應用程式的組建。這是選用`pre-run`參數。我們會說明何時使用此參數，以及以下各節中有關組建的其他實用資訊。

下表說明哪些版本的 App Runner 組建適用於特定的受管執行時間版本。我們會繼續更新本文件，讓您隨時了解我們目前的執行時間。

平台	執行時間版本	建置程序	App Runner 終止支援日期
Python – 版本資訊	Python 3.11（！）	已修訂	
	Python 3.8	原始的	2025 年 12 月 1 日
	Python 3.7	原始的	2025 年 12 月 1 日
Node.js – 版本資訊	Node.js 22	已修訂	
	Node.js 18	已修訂	2025 年 12 月 1 日
	Node.js 16	原始的	2025 年 12 月 1 日
	Node.js 14	原始的	2025 年 12 月 1 日
	Node.js 12	原始的	2025 年 12 月 1 日
Corretto – 版本資訊	Corretto 11	原始的	
	Corretto 8	原始的	
.NET – 版本資訊	.NET 6 *	原始的	2025 年 12 月 1 日
PHP – 版本資訊	PHP 8.1 *	原始的	2025 年 12 月 31 日
Ruby – 版本資訊	Ruby 3.1 *	原始的	2025 年 12 月 1 日
Go – 版本資訊	Go 1 *	原始的	2025 年 12 月 1 日

> **Note**
>
> 某些列出的執行時間包括終止支援日期。如需詳細資訊，請參閱the section called “受管執行時間版本的終止支援”。

> **Important**
>
> Python 3.11 – 對於使用 Python 3.11 受管執行時間的服務建置組態，我們提供特定建議。如需詳細資訊，請參閱 Python 平台主題特定執行時間版本的呼叫中的 。

App Runner 建置和遷移的詳細資訊

當您將應用程式遷移至使用修訂組建的較新執行期時，您可能需要稍微修改組建組態。

為了提供遷移考量的內容，我們會先說明原始 App Runner 組建和修訂組建的高階程序。接下來我們將有一個章節，說明可能需要一些組態更新之服務的特定屬性。

原始 App Runner 組建

原始 App Runner 應用程式建置程序會利用 AWS CodeBuild 服務。初始步驟是根據 CodeBuild 服務所策劃的影像。以下 Docker 建置程序會使用適用的 App Runner 受管執行期映像做為基礎映像。

一般步驟如下：

1. 在 CodeBuild 策劃的映像中執行`pre-build`命令。

   這些`pre-build`命令是選用的。它們只能在`apprunner.yaml`組態檔案中指定。
2. 在上一個步驟的相同映像上使用 CodeBuild 執行`build`命令。

   `build` 命令是必要的。它們可以在 App Runner 主控台、App Runner API 或`apprunner.yaml`組態檔案中指定。
3. 執行 Docker 組建，根據特定平台和執行期版本的 App Runner 受管執行期映像產生映像。
4. 從我們在步驟 2 中產生的映像複製`/app`目錄。目的地是根據我們在步驟 3 中產生的 App Runner 受管執行期映像的映像。
5. 在產生的 App Runner 受管執行期映像上再次執行`build`命令。我們會再次執行組建命令，從我們在步驟 4 中複製的`/app`目錄中的原始程式碼產生組建成品。App Runner 稍後會部署此映像，以在容器中執行您的 Web 服務。

   `build` 命令是必要的。它們可以在 App Runner 主控台、App Runner API 或`apprunner.yaml`組態檔案中指定。
6. 在步驟 2 的 CodeBuild 映像中執行`post-build`命令。

這些post-build命令是選用的。它們只能在apprunner.yaml組態檔案中指定。

建置完成後，App Runner 會從步驟 5 部署產生的 App Runner 受管執行期映像，以在容器中執行您的 Web 服務。

修訂後的 App Runner 組建

修訂後的建置程序比上一節所述的原始建置程序更快且更有效率。它消除了先前版本組建中發生的組建命令的重複。它也會建立佔用空間較小的最終映像，只包含執行應用程式所需的原始程式碼、建置成品和執行時間。

此建置程序使用 Docker 多階段建置。一般程序步驟如下：

1. 建置階段 — 在 App Runner 建置映像之上啟動執行 pre-build和 build命令的 Docker 建置程序。
   a. 將應用程式原始碼複製到 /app目錄。

      > **Note**
      > 此/app目錄會在 Docker 組建的每個階段中指定為工作目錄。

   b. 執行 pre-build 命令。

      這些pre-build命令是選用的。它們只能在apprunner.yaml組態檔案中指定。

   c. 執行build命令。

      build 命令是必要的。它們可以在 App Runner 主控台、App Runner API 或apprunner.yaml組態檔案中指定。
2. 封裝階段 — 產生最終客戶容器映像，這也以 App Runner 執行映像為基礎。
   a. 將/app目錄從先前的建置階段複製到新的執行映像。這包括您的應用程式原始程式碼和上一個階段的建置成品。
   b. 執行pre-run命令。如果您需要使用 build命令修改/app目錄外部的執行期映像，請將相同或必要的命令新增至apprunner.yaml組態檔案的此區段。

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

      這些pre-run命令是選用的。它們只能在apprunner.yaml組態檔案中指定。

> **備註**
>
> - pre-run 命令僅受修訂的組建支援。如果您的服務使用使用原始建置的執行期版本，請勿將其新增至組態檔案。
> - 如果您不需要使用 build命令修改/app目錄外的任何內容，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並執行post-build命令。
   a. 在 /app目錄中執行post-build命令。

      這些post-build命令是選用的。它們只能在apprunner.yaml組態檔案中指定。

建置完成後，App Runner 接著會部署執行映像，以在容器中執行您的 Web 服務。

> **Note**
>
> 設定建置程序apprunner.yaml時，請勿誤導至 執行區段中的env項目。即使步驟 2(b) 中參考的pre-run命令參數位於執行區段中，也請勿使用執行區段中的 env 參數來設定建置。這些pre-run命令只會參考組態檔案建置區段中定義的env變數。如需詳細資訊，請參閱 App Runner 組態檔案章節執行區段中的 。

遷移考量的服務需求

如果您的應用程式環境有這兩個需求之一，則您需要透過新增pre-run命令來修改建置組態。

- 如果您需要使用 build命令修改/app目錄外的任何內容。
- 如果您需要執行build命令兩次，以建立所需的環境。這是非常不尋常的要求。絕大多數的組建都不會這樣做。

/app目錄外的修改

- 修訂後的 App Runner 組建假設您的應用程式在 /app 目錄外沒有相依性。
- 您隨apprunner.yaml檔案、App Runner API 或 App Runner 主控台提供的命令必須在 /app目錄中產生建置成品。
- 您可以修改 pre-build、 build和 post-build命令，以確保所有建置成品都位於 /app目錄中。

- 如果您的應用程式需要 建置進一步修改服務產生的映像，您可以在 `/app` 目錄中使用新`pre-run`命令`apprunner.yaml`。如需詳細資訊，請參閱使用組態檔案設定 App Runner 服務選項。

執行**`build`**命令兩次

- 原始 App Runner 組建會執行`build`命令兩次，首先在步驟 2 中執行，然後再次在步驟 5 中執行。修訂後的 App Runner 組建可修復此備援，而且只會執行`build`命令一次。如果您的應用程式應該對`build`命令執行兩次有不尋常的需求，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組建行為。

使用 Python 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Python 3.7 和 Python 3.8 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner Python 平台提供受管執行期。每個執行時間都可讓您根據 Python 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 Python 執行期時，App Runner 會從受管 Python 執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，包含 Python 版本的執行時間套件，以及一些工具和熱門的相依性套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

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

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

Python 執行時間的版本語法： *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`3.8.5`

下列範例示範版本鎖定：

- `3.8` – 鎖定主要和次要版本。App Runner 只會更新修補程式版本。
- `3.8.5` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

主題

- Python 執行期組態
- 特定執行時間版本的呼叫
- Python 執行時間範例
- Python 執行期發行資訊

Python 執行期組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

特定執行時間版本的呼叫

> **Note**
>
> App Runner 現在會根據下列執行時間版本執行應用程式的更新建置程序：Python 3.11、Node.js 22 和 Node.js 18。如果您的應用程式在其中一個執行時間版本上執行，請參閱受管執行期版本和 App Runner 組建 以取得修訂建置程序的詳細資訊。使用所有其他執行時間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 （已修訂 App Runner 組建）

針對受管 Python 3.11 執行時間，在 apprunner.yaml 中使用下列設定。

- 將頂端區段中的`runtime`金鑰設定為 `python311`

  Example

  ```
  runtime: python311
  ```

- 使用 `pip3`而非 `pip` 安裝相依性。
- 使用`python3`解譯器而非 `python`。
- 以`pre-run`命令執行`pip3`安裝程式。Python 會在 `/app`目錄之外安裝相依性。由於 App Runner 會針對 Python 3.11 執行修訂後的 App Runner 組建，因此透過 `apprunner.yaml` 檔案組建區段中的命令在`/app`目錄外安裝的任何內容都會遺失。如需詳細資訊，請參閱修訂後的 App Runner 組建。

  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 擴充組態檔案範例。

Python 執行時間範例

下列範例顯示用於建置和執行 Python 服務的 App Runner 組態檔案。最後一個範例是完整 Python 應用程式的原始程式碼，您可以部署到 Python 執行時間服務。

> **Note**
>
> 這些範例中使用的執行時間版本為 *`3.7.7`* 和 *`3.11#`*您可以將其取代為您想要使用的版本。如需最新支援的 Python 執行期版本，請參閱 the section called “版本資訊”。

最小 Python 組態檔案

此範例顯示您可以搭配 Python 受管執行時間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 `python3`命令。如需詳細資訊，請參閱本主題稍後的 Python 3.11 擴充組態檔案範例。

Example apprunner.yaml

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

擴充 Python 組態檔案

此範例顯示使用具有 Python 受管執行時間的所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本是 *3.7.7*。您可以將其取代為您想要使用的版本。如需最新支援的 Python 執行期版本，請參閱 the section called “版本資訊”。
> Python 3.11 使用 `pip3`和 `python3`命令。如需詳細資訊，請參閱本主題稍後的 Python 3.11 擴充組態檔案範例。

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-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"
```

擴充 Python 組態檔案 — Python 3.11 （使用修訂的組建）

此範例顯示在 中使用所有組態金鑰搭配 Python 3.11 受管執行時間。 `apprunner.yaml`此範例包含 `pre-run`區段，因為此版本的 Python 使用修訂過的 App Runner 建置。

只有修訂後的 App Runner 組建才支援 `pre-run` 參數。如果您的應用程式使用原始 App Runner 組建支援的執行時間版本，請勿將此參數插入您的組態檔案中。如需詳細資訊，請參閱受管執行期版本和 App Runner 組建。

> **Note**
>
> 這些範例中使用的執行時間版本為 *`3.11#`*您可以將其取代為您想要使用的版本。如需最新支援的 Python 執行期版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-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"
```

完成 Python 應用程式來源

此範例顯示您可以部署到 Python 執行期服務的完整 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 apprunner.yaml

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

Python 執行期發行資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Python 3.7 和 Python 3.8 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

本主題列出 App Runner 支援的 Python 執行期版本的完整詳細資訊。

支援的執行時間版本 — 修訂後的 App Runner 組建

執行時間名稱	次要版本	包含的套件
Python 3.11 (python311)	3.11.14	SQLite 3.50.2
	3.11.13	SQLite 3.50.2
	3.11.13	SQLite 3.50.1
	3.11.12	SQLite 3.50.0
	3.11.11	SQLite 3.49.1
	3.11.10	SQLite 3.46.1
	3.11.9	SQLite 3.46.1
	3.11.8	SQLite 3.45.2
	3.11.7	SQLite 3.44.2

> **備註**
>
> - Python 3.11 – 對於使用 Python 3.11 受管執行時間的服務建置組態，我們提供特定建議。如需詳細資訊，請參閱 Python 平台主題特定執行時間版本的呼叫中的 。

- App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
Python 3 (python3)	3.8.20	SQLite 3.50.2
	3.8.20	SQLite 3.50.1
	3.8.20	SQLite 3.50.0
	3.8.16	SQLite 3.46.1
	3.8.15	SQLite 3.40.0
	3.7.16	SQLite 3.50.2
	3.7.16	SQLite 3.50.0
	3.7.15	SQLite 3.40.0
	3.7.10	SQLite 3.40.0

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

使用 Node.js 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Node.js 12、Node.js 14、Node.js 16 和 Node.js 18 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner Node.js 平台提供受管執行期。每個執行時間都可讓您根據 Node.js 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 Node.js 執行期時，App Runner 會從受管 Node.js 執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，並包含 Node.js 版本和一些工具的執行時間套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

如需有效的 Node.js 執行時間名稱和版本，請參閱 the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

Node.js 執行時間的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`22.14.0`

下列範例示範版本鎖定：

- `22.14` – 鎖定主要和次要版本。App Runner 只會更新修補程式版本。
- `22.14.0` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

主題

- Node.js 執行期組態
- 特定執行時間版本的呼叫

- Node.js 執行時間範例
- Node.js 執行時間版本資訊

Node.js 執行期組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

特別是使用 Node.js 執行時間，您也可以使用來源儲存庫根`package.json`目錄中名為 的 JSON 檔案來設定建置和執行時間。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）。npm 或 yarn 等套件管理員會將此檔案解譯為其命令的輸入。

例如：

- npm install 在 中安裝由 `dependencies`和 `devDependencies`節點定義的套件`package.json`。
- npm start 或 會npm run start執行 中`scripts/start`節點定義的命令`package.json`。

以下是範例 `package.json` 檔案。

package.json

```
{
  "name": "node-js-getting-started",
  "version": "0.3.0",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "22.14.0"
  },
```

```
  "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 的詳細資訊`package.json`，請參閱 npm Docs 網站上的建立 package.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 組態檔案中將其用作run命令，如下列範例所示。

  Example

  package.json

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

  apprunner.yaml

  ```
  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`。此檔案包含剛安裝的套件版本 npm 的快照。之後，當 npm 安裝相依性時，它會使用這些確切版本。如果您安裝 yarn，它會建立 `yarn.lock` 檔案。將這些檔案遞交至原始碼儲存庫，以確保您的應用程式已安裝您開發和測試的相依性版本。

- 您也可以使用 App Runner 組態檔案來設定 Node.js 版本和啟動命令。當您這樣做時，這些定義會覆寫 中的定義`package.json`。中的`node`版本`package.json`與 App Runner 組態檔案中`runtime-version`的值衝突會導致 App Runner 建置階段失敗。

特定執行時間版本的呼叫

Node.js 22 和 Node.js 18（已修訂 App Runner 組建）

App Runner 現在會根據下列執行時間版本執行應用程式的更新建置程序：Python 3.11、Node.js 22 和 Node.js 18。如果您的應用程式在其中一個執行時間版本上執行，請參閱 受管執行期版本和 App Runner 組建 以取得修訂建置程序的詳細資訊。使用所有其他執行時間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 執行時間範例

下列範例顯示用於建置和執行 Node.js 服務的 App Runner 組態檔案。

> **Note**
>
> 這些範例中使用的執行時間版本為 *`22.14.0`*。您可以將其取代為您想要使用的版本。如需最新支援的 Node.js 執行時間版本，請參閱 the section called “版本資訊”。

最小 Node.js 組態檔案

此範例顯示您可以搭配 Node.js 受管執行時間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 apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

擴充 Node.js 組態檔案

此範例顯示搭配 Node.js 受管執行時間使用所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本為 *22.14.0*。您可以將其取代為您想要使用的版本。如需最新支援的 Node.js 執行時間版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
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: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

擴充 Node.js 組態檔案 – Node.js 22 （使用修訂的組建）

此範例顯示使用 中具有 Node.js 受管執行時間的所有組態金鑰。 `apprunner.yaml`此範例包含 `pre-run`區段，因為此版本的 Node.js 使用修訂過的 App Runner 組建。

只有修訂後的 App Runner 組建才支援 `pre-run` 參數。如果您的應用程式使用原始 App Runner 組建支援的執行時間版本，請勿將此參數插入您的組態檔案中。如需詳細資訊，請參閱受管執行期版本和 App Runner 組建。

> **Note**
>
> 這些範例中使用的執行時間版本為 *22.14.0#*您可以將其取代為您想要使用的版本。如需最新支援的 Node.js 執行時間版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
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: 22.14.0
  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 應用程式搭配 Grunt

此範例說明如何設定使用 Grunt 開發的 Node.js 應用程式。Grunt 是命令列 JavaScript 任務執行器。它執行重複性任務並管理程序自動化，以減少人為錯誤。使用 npm 安裝和管理 Grunt 和 Grunt 外掛程式。您可以透過在來源儲存庫的根目錄中包含 `Gruntfile.js` 檔案來設定 Grunt。

Example package.json

```
{
  "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 apprunner.yaml

> **Note**
>
> 這些範例中使用的執行時間版本為 *22.14.0*。您可以將其取代為您想要使用的版本。如需最新支援的 Node.js 執行時間版本，請參閱 the section called “版本資訊”。

```
version: 1.0
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install grunt grunt-cli
      - npm install --only=dev
      - npm run build
    build:
      - npm install --production
run:
  runtime-version: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Node.js 執行時間版本資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Node.js 12、Node.js 14、Node.js 16 和 Node.js 18 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

> **Note**
>
> App Runner 的標準棄用政策是在執行時間的任何主要元件達到社群長期支援 (LTS) 結束且不再提供安全性更新時棄用執行時間。在某些情況下，App Runner 可能會在執行時間支援的語言版本end-of-support日期之後，將執行時間的棄用延遲一段有限期間。這類案例的範例可能是擴展對執行時間的支援，以允許客戶有時間進行遷移。

本主題列出 App Runner 支援的 Node.js 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 修訂後的 App Runner 組建

執行時間名稱	次要版本	包含的套件
Node.js 22 (nodejs22)	22.21.1	npm 10.9.4， yarn 1.22.22
	22.20.0	npm 10.9.3， yarn 1.22.22
	22.17.0	npm 10.9.2， yarn 1.22.22
	22.16.0	npm 10.9.2， yarn 1.22.22
	22.14.0	npm 10.9.2， yarn 1.22.22

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到修訂後的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

支援的執行時間版本 — 修訂後的 App Runner 組建

執行時間名稱	次要版本	包含的套件
Node.js 18 (nodejs18)	18.20.8	npm 10.8.2， yarn 1.22.22
	18.20.7	npm 10.8.2， yarn 1.22.22

執行時間名稱	次要版本	包含的套件
	18.20.6	npm 10.8.2， yarn 1.22.22
	18.20.5	npm 10.8.2， yarn 1.22.22
	18.20.4	npm 10.7.0， yarn 1.22.22
	18.20.3	npm 10.7.0， yarn 1.22.22
	18.20.2	npm 10，毛線 *
	18.19.1	npm 10，毛線 *
	18.19.0	npm 10，毛線 *

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
Node.js 16 (nodejs16)	16.20.2	npm 8.19.4， yarn 1.22.22
	16.20.1	npm 8.19.4，毛線 *
	16.20.0	npm 8.19.4，毛線 *
	16.19.1	npm 8.19.4，毛線 *
	16.19.0	npm 8.19.4，毛線 *
	16.18.1	npm 8.19.4，毛線 *
	16.17.1	npm 8.19.4，毛線 *
	16.17.0	npm 8.19.4，毛線 *
Node.js 14 (nodejs14)	14.21.3	npm 6.14.18， yarn 1.22.22
	14.21.2	npm 6.14.18，毛線 *

執行時間名稱	次要版本	包含的套件
	14.21.1	npm 6.14.18，毛線 *
	14.20.1	npm 6.14.18，毛線 *
	14.19.0	npm 6.14.18，毛線 *
Node.js 12 (nodejs12)	12.22.12	npm 6.14.16， yarn 1.22.22
	12.21.0	npm 6.14.16，毛線 *

使用 Java 平台

AWS App Runner Java 平台提供受管執行期。每個執行時間都可讓您根據 Java 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 Java 執行期時，App Runner 會從受管 Java 執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，並包含 Java 版本的執行時間套件和一些工具。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

目前，所有支援的 Java 執行時間都以 Amazon Corretto 為基礎。如需有效的 Java 執行期名稱和版本，請參閱 the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

Amazon Corretto 執行時間的版本語法：

執行時間	語法	範例
corretto11	`11.0[.`*`openjdk-update`*` [.`*`openjdk-b`*	`11.0.13.08.1`

執行時間	語法	範例
	uild [.*corretto-specific-revision*]]]	
corretto8	8[.*openjdk-update* [.*openjdk-build* [.*corretto-specific-revision*]]]	8.312.07.1

下列範例示範版本鎖定：

- 11.0.13 – 鎖定 Open JDK 更新版本。App Runner 只會更新 Open JDK 和 Amazon Corretto 低階組建。
- 11.0.13.08.1 – 鎖定至特定版本。App Runner 不會更新您的執行時間版本。

主題

- Java 執行期組態
- Java 執行時間範例
- Java 執行期版本資訊

Java 執行期組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

Java 執行時間範例

下列範例顯示用於建置和執行 Java 服務的 App Runner 組態檔案。最後一個範例是完整 Java 應用程式的原始碼，您可以部署到 Corretto 11 執行時間服務。

> **Note**
>
> 這些範例中使用的執行時間版本為 *11.0.13.08.1*。您可以將其取代為您想要使用的版本。如需最新支援的 Java 執行期版本，請參閱 the section called “版本資訊”。

最小 Corretto 11 組態檔案

此範例顯示您可以搭配 Corretto 11 受管執行時間使用的最小組態檔案。如需 App Runner 使用最少組態檔案所做的假設，請參閱 。

Example apprunner.yaml

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

擴充 Corretto 11 組態檔案

此範例示範如何使用所有組態金鑰搭配 Corretto 11 受管執行時間。

> **Note**
>
> 這些範例中使用的執行時間版本為 *11.0.13.08.1*。您可以將其取代為您想要使用的版本。如需最新支援的 Java 執行期版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
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"
```

完成 Corretto 11 應用程式來源

此範例顯示完整 Java 應用程式的原始程式碼，您可以部署到 Corretto 11 執行時間服務。

Example src/main/java/com/HelloWorld/HelloWorld.java

```
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/main/java/com/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 apprunner.yaml

```
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 執行期版本資訊

本主題列出 App Runner 支援的 Java 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
Corretto 11 (corretto11)	11.0.28.6.1	Maven 3.9.11、Gradle 6.9.4
	11.0.27.6.1	Maven 3.9.10、Gradle 6.9.4
	11.0.27.6.1	Maven 3.9.9、Gradle 6.9.4
	11.0.26.4.1	Maven 3.9.9、Gradle 6.9.4
	11.0.25.9.1	Maven 3.9.9、Gradle 6.9.4
	11.0.24.8.1	Maven 3.9.9、Gradle 6.9.4
	11.0.23.9.1	Maven 3.9.8、Gradle 6.9.4
	11.0.22.7.1	Maven 3.9.6、Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.6、Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.5、Gradle 6.9.4
	11.0.20.8.1	Maven 3.9.3、Gradle 6.9.4
	11.0.19.7.1	Maven 3.9.3、Gradle 6.9.4
	11.0.18.10.1	Maven 3.9.1、Gradle 6.9.4
	11.0.17.8.1	Maven 3.8.6、Gradle 6.9.3
	11.0.16.9.1	Maven 3.8.6、Gradle 6.9.2
	11.0.13.08.1	Maven 3.6.3、Gradle 6.5

執行時間名稱	次要版本	包含的套件
Corretto 8 (corretto8)	8.472.08.1	Maven 3.9.11、Gradle 6.9.4
	8.462.08.1	Maven 3.9.11、Gradle 6.9.4
	8.452.09.2	Maven 3.9.10、Gradle 6.9.4
	8.452.09.2	Maven 3.9.9、Gradle 6.9.4
	8.452.09.1	Maven 3.9.9、Gradle 6.9.4
	8.442.06.1	Maven 3.9.9、Gradle 6.9.4
	8.432.06.1	Maven 3.9.9、Gradle 6.9.4
	8.422.05.1	Maven 3.9.9、Gradle 6.9.4
	8.412.08.1	Maven 3.9.8、Gradle 6.9.4
	8.402.08.1	Maven 3.9.6、Gradle 6.9.4
	8.392.08.1	Maven 3.9.6、Gradle 6.9.4
	8.382.05.1	Maven 3.9.4、Gradle 6.9.4
	8.372.07.1	Maven 3.9.3、Gradle 6.9.4
	8.362.08.1	Maven 3.9.1、Gradle 6.9.4
	8.352.08.1	Maven 3.8.6、Gradle 6.9.3
	8.342.07.4	Maven 3.8.6、Gradle 6.9.2
	8.312.07.1	Maven 3.6.3、Gradle 6.5

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到修訂後的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

使用 .NET 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 .NET 6 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner .NET 平台提供受管執行時間。每個執行時間都可讓您根據 .NET 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 .NET 執行期時，App Runner 會從受管 .NET 執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，包含 .NET 版本的執行時間套件，以及一些工具和熱門相依性套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

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

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

.NET 執行時間的版本語法： *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`6.0.9`

下列範例示範版本鎖定：

- `6.0` – 鎖定主要和次要版本。App Runner 只會更新修補程式版本。

- `6.0.9` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

主題

- .NET 執行時間組態
- .NET 執行時間範例
- .NET 執行時間版本資訊

.NET 執行時間組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

.NET 執行時間範例

下列範例顯示用於建置和執行 .NET 服務的 App Runner 組態檔案。最後一個範例是完整 .NET 應用程式的原始程式碼，您可以部署到 .NET 執行時間服務。

> **Note**
>
> 這些範例中使用的執行時間版本為 *`6.0.9`*。您可以將其取代為您想要使用的版本。如需最新支援的 .NET 執行時間版本，請參閱 the section called “版本資訊”。

最小 .NET 組態檔案

此範例顯示您可以搭配 .NET 受管執行時間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 apprunner.yaml

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

擴充的 .NET 組態檔案

此範例顯示搭配 .NET 受管執行時間使用所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本為 *6.0.9*。您可以將其取代為您想要使用的版本。如需最新支援的 .NET 執行時間版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
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"
```

完成 .NET 應用程式來源

此範例顯示您可以部署到 .NET 執行時間服務的完整 .NET 應用程式的原始碼。

> **Note**
>
> - 執行下列命令來建立簡單的 .NET 6 Web 應用程式： `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - 將 `apprunner.yaml`新增至建立的 .NET 6 Web 應用程式。

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 執行時間版本資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 .NET 6 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

本主題列出 App Runner 支援的 .NET 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
.NET 6 (dotnet6)	6.0.36	.NET SDK 6.0.428
	6.0.33	.NET SDK 6.0.425
	6.0.32	.NET SDK 6.0.424
	6.0.31	.NET SDK 6.0.423
	6.0.30	.NET SDK 6.0.422
	6.0.29	.NET SDK 6.0.421
	6.0.28	.NET SDK 6.0.420
	6.0.26	.NET SDK 6.0.418
	6.0.25	.NET SDK 6.0.417
	6.0.24	.NET SDK 6.0.416
	6.0.22	.NET SDK 6.0.414
	6.0.21	.NET SDK 6.0.413
	6.0.20	.NET SDK 6.0.412
	6.0.19	.NET SDK 6.0.411
	6.0.16	.NET SDK 6.0.408
	6.0.15	.NET SDK 6.0.407
	6.0.14	.NET SDK 6.0.406
	6.0.13	.NET SDK 6.0.405
	6.0.12	.NET SDK 6.0.404

執行時間名稱	次要版本	包含的套件
	6.0.11	.NET SDK 6.0.403
	6.0.10	.NET SDK 6.0.402
	6.0.9	.NET SDK 6.0.401

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

使用 PHP 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 31 日結束對 PHP 8.1 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner PHP 平台提供受管執行時間。您可以使用每個執行時間，根據 PHP 版本使用 Web 應用程式建置和執行容器。當您使用 PHP 執行期時，App Runner 會從受管 PHP 執行期映像開始。此映像是以 Amazon Linux Docker 映像為基礎，並包含 PHP 版本和一些工具的執行時間套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

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

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定

它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

PHP 執行時間的版本語法：`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。App Runner 只會更新修補程式版本。
- `8.1.10` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

> **Important**
>
> 如果您想要在預設儲存庫根目錄以外的位置指定 App Runner 服務的程式碼儲存庫來源目錄，您的 PHP 受管執行時間版本必須是 PHP `8.1.22`或更新版本。之前的 PHP 執行時間版本`8.1.22`只能使用預設的根來源目錄。

主題

- PHP 執行期組態
- 相容性
- PHP 執行時間範例
- PHP 執行期版本資訊

PHP 執行期組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

相容性

您可以使用下列其中一個 Web 伺服器在 PHP 平台上執行 App Runner 服務：

- Apache HTTP Server
- NGINX

Apache HTTP Server 和 NGINX 與 PHP-FPM 相容。您可以使用下列NGINX其中一項啟動 Apache HTTP Server和 ：

- 監督 - 如需執行 的詳細資訊supervisord，請參閱執行監督。
- 啟動指令碼

如需如何使用 Apache HTTP Server 或 NGINX 透過 PHP 平台設定 App Runner 服務的範例，請參閱 the section called “完成 PHP 應用程式來源”。

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**
>
> 建議您將 `startup.sh`或 `supervisord.conf` 檔案存放在 Web 伺服器的根目錄中。請確定`start`命令指向存放 `startup.sh`或 `supervisord.conf` 檔案的位置。

如果您使用的是 ，以下是 檔案結構的範例supervisord。

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

如果您使用的是啟動指令碼，下列是檔案結構的範例。

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

我們建議您將這些檔案結構存放在為 App Runner 服務指定的程式碼儲存庫來源目錄中。

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

> **Important**
>
> 如果您想要在預設儲存庫根目錄以外的位置指定 App Runner 服務的程式碼儲存庫來源目錄，您的 PHP 受管執行時間版本必須是 PHP `8.1.22`或更新版本。之前的 PHP 執行時間版本`8.1.22`只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，將服務的執行時間更新為最新版本。您的服務預設會使用最新的執行時間，除非您使用 App Runner 組態檔案中的 `runtime-version`關鍵字指定版本鎖定。

PHP 執行時間範例

以下是用於建置和執行 PHP 服務的 App Runner 組態檔案範例。

最小 PHP 組態檔案

下列範例是您可以搭配 PHP 受管執行時間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 apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
```

```
run:
  command: ./startup.sh
```

擴充 PHP 組態檔案

下列範例使用具有 PHP 受管執行時間的所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本為 *8.1.10*。您可以將其取代為您想要使用的版本。如需最新支援的 PHP 執行期版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
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"
```

完成 PHP 應用程式來源

下列範例是 PHP 應用程式原始碼，您可以使用 Apache HTTP Server或 來部署至 PHP 執行時間服務NGINX。這些範例假設您使用預設檔案結構。

Apache HTTP Server 使用 搭配 執行 PHP 平台 supervisord

Example檔案結構

> **Note**
>
> - `supervisord.conf` 檔案可以存放在儲存庫中的任何位置。請確定`start`命令指向`supervisord.conf`檔案的存放位置。
> - `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 supervisord.conf

```
[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 apprunner.yaml

```
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>
```

Apache HTTP Server 使用 搭配 執行 PHP 平台 startup script

Example檔案結構

> **Note**
>
> - `startup.sh` 檔案可以存放在儲存庫的任何位置。請確定`start`命令指向`startup.sh`檔案的存放位置。
> - `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**
>
> - 請務必先將`startup.sh`檔案儲存為可執行檔，再將其遞交至 Git 儲存庫。使用 `chmod +x startup.sh` 設定 `startup.sh` 檔案的執行許可。
> - 如果您未將`startup.sh`檔案儲存為可執行檔，請在您的`apprunner.yaml`檔案中輸入`chmod +x startup.sh`作為`build`命令。

Example apprunner.yaml

```
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>
```

NGINX 使用 搭配 執行 PHP 平台 supervisord

Example檔案結構

> **Note**
>
> - `supervisord.conf` 檔案可以存放在儲存庫中的任何位置。請確定`start`命令指向`supervisord.conf`檔案的存放位置。
> - `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 supervisord.conf

```
[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 apprunner.yaml

```
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>
```

NGINX 使用 搭配 執行 PHP 平台 startup script

Example檔案結構

> **Note**
>
> - `startup.sh` 檔案可以存放在儲存庫的任何位置。請確定`start`命令指向`startup.sh`檔案的存放位置。
> - `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**
>
> - 請務必先將`startup.sh`檔案儲存為可執行檔，再將其遞交至 Git 儲存庫。使用 `chmod +x startup.sh` 設定 `startup.sh` 檔案的執行許可。

- 如果您未將`startup.sh`檔案儲存為可執行檔，請在您的`apprunner.yaml`檔案中輸入`chmod +x startup.sh`作為`build`命令。

Example apprunner.yaml

```
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 執行期版本資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 31 日終止對 PHP 8.1 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

本主題列出 App Runner 支援的 PHP 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
PHP 8.1 (php81)	8.1.33	
	8.1.32	
	8.1.31	
	8.1.29	
	8.1.28	
	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 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

使用 Ruby 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Ruby 3.1 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner Ruby 平台提供受管執行期。每個執行時間都可讓您根據 Ruby 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 Ruby 執行期時，App Runner 會從受管 Ruby 執行期映像開始。此映像以 Amazon Linux Docker 映像為基礎，並包含 Ruby 版本的執行時間套件和一些工具。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

如需有效的 Ruby 執行時間名稱和版本，請參閱 the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

Ruby 執行時間的版本語法： *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`3.1.2`

下列範例示範版本鎖定：

- `3.1` – 鎖定主要和次要版本。App Runner 只會更新修補程式版本。

- `3.1.2` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

主題

- Ruby 執行時間組態
- Ruby 執行時間範例
- Ruby 執行時間版本資訊

Ruby 執行時間組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

Ruby 執行時間範例

下列範例顯示用於建置和執行 Ruby 服務的 App Runner 組態檔案。

最小 Ruby 組態檔案

此範例顯示您可以搭配 Ruby 受管執行時間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 apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
```

```
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 8080
```

擴充 Ruby 組態檔案

此範例顯示搭配 Ruby 受管執行時間使用所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本是 *3.1.2#*您可以將其取代為您想要使用的版本。如需最新支援的 Ruby 執行時間版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
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 應用程式來源

這些範例顯示完整 Ruby 應用程式的原始程式碼，您可以部署到 Ruby 執行時間服務。

Example server.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 apprunner.yaml

```
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 執行時間版本資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Ruby 3.1 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

本主題列出 App Runner 支援的 Ruby 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
Ruby 3.1 (ruby31)	3.1.7	SQLite 3.50.2
	3.1.7	SQLite 3.50.1
	3.1.7	SQLite 3.50.0
	3.1.6	SQLite 3.49.1
	3.1.4	SQLite 3.46.0
	3.1.3	SQLite 3.41.0
	3.1.2	SQLite 3.39.4

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

使用 Go 平台

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Go 1.18 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

AWS App Runner Go 平台提供受管執行時間。每個執行時間都可讓您根據 Go 版本使用 Web 應用程式輕鬆建置和執行容器。當您使用 Go 執行時間時，App Runner 會從受管 Go 執行時間映像開始。此映像以 Amazon Linux Docker 映像為基礎，並包含 Go 版本和一些工具的執行時間套件。App Runner 使用此受管執行期映像做為基礎映像，並新增您的應用程式程式碼來建置 Docker 映像。然後，它會部署此映像，以在容器中執行您的 Web 服務。

當您使用 App Runner 主控台或 CreateService API 操作建立服務時，您可以指定 App Runner 服務的執行時間。您也可以指定執行時間做為原始程式碼的一部分。在程式碼儲存庫中包含的 App Runner 組態檔案中使用 `runtime`關鍵字。受管執行時間的命名慣例為 *`<language-name><major-version>`*。

如需有效的 Go 執行時間名稱和版本，請參閱 the section called “版本資訊”。

App Runner 會在每次部署或服務更新時，將服務的執行時間更新為最新版本。如果您的應用程式需要特定版本的受管執行時間，您可以使用 App Runner 組態檔案中的 `runtime-version`關鍵字來指定它。您可以鎖定任何層級的版本，包括主要或次要版本。App Runner 只會對服務的執行時間進行較低層級的更新。

Go 執行時間的版本語法：*`major`*`[.`*`minor`*`[.`*`patch`*`]]`

例如：`1.18.7`

下列範例示範版本鎖定：

- `1.18` – 鎖定主要和次要版本。App Runner 只會更新修補程式版本。
- `1.18.7` – 鎖定至特定修補程式版本。App Runner 不會更新您的執行時間版本。

主題

- Go 執行時間組態
- Go 執行時間範例

- Go 執行時間版本資訊

Go 執行時間組態

當您選擇受管執行時間時，您也必須至少設定建置和執行命令。您可以在建立或更新 App Runner 服務時設定它們。您可以使用下列其中一種方法執行此操作：

- 使用 App Runner 主控台 – 在建立程序或組態索引標籤的設定建置區段中指定命令。
- 使用 App Runner API – 呼叫 CreateService 或 UpdateService API 操作。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指定一或多個建置命令，以及用於啟動應用程式的單一執行命令。還有其他選用的組態設定。

提供組態檔案是選用的。當您使用主控台或 API 建立 App Runner 服務時，您可以指定 App Runner 在建立時直接從組態檔案取得您的組態設定。

Go 執行時間範例

下列範例顯示用於建置和執行 Go 服務的 App Runner 組態檔案。

Minimal Go 組態檔案

此範例顯示您可以搭配 Go 受管執行時間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 apprunner.yaml

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

Extended Go 組態檔案

此範例顯示使用 Go 受管執行時間的所有組態金鑰。

> **Note**
>
> 這些範例中使用的執行時間版本為 *1.18.7#*您可以將其取代為您想要使用的版本。如需最新支援的 Go 執行時間版本，請參閱 the section called “版本資訊”。

Example apprunner.yaml

```
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"
```

完成 Go 應用程式來源

這些範例顯示您可以部署到 Go 執行時間服務的完整 Go 應用程式的原始碼。

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 apprunner.yaml

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

Go 執行時間版本資訊

> **Important**
>
> App Runner 將於 2025 年 12 月 1 日結束對 Go 1.18 的支援。如需建議和詳細資訊，請參閱 the section called “受管執行時間版本的終止支援”。

本主題列出 App Runner 支援的 Go 執行時間版本的完整詳細資訊。

支援的執行時間版本 — 原始 App Runner 組建

執行時間名稱	次要版本	包含的套件
Go 1 (go1)	1.18.10	
	1.18.9	

執行時間名稱	次要版本	包含的套件
	1.18.8	
	1.18.7	

> **Note**
>
> App Runner 為最近發佈的特定主要執行時間提供修訂後的建置程序。因此，您會在本文件的特定章節中看到已修訂的 App Runner 組建和原始 App Runner 組建的參考。如需詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

開發 App Runner 的應用程式程式碼

本章討論開發或遷移應用程式程式碼以進行部署時應考慮的執行期資訊和開發指導方針 AWS App Runner。

執行時間資訊

無論您是提供容器映像，還是 App Runner 為您建置映像，App Runner 都會在容器執行個體中執行您的應用程式碼。以下是容器執行個體執行期環境的一些關鍵層面。

- 架構支援 – App Runner 支援實作 Web 應用程式的任何映像。如果您使用任何程式設計語言，這與您選擇的程式設計語言和您使用的 Web 應用程式伺服器或架構無關。為了您的方便，我們為各種程式設計平台提供平台特定的受管執行期，以簡化應用程式建置程序和抽象映像建立。
- Web 請求 – App Runner 為容器執行個體提供 HTTP 1.0 和 HTTP 1.1 的支援。如需設定服務的詳細資訊，請參閱 the section called “Configuration”。您不需要實作 HTTPS 安全流量的處理。App Runner 會將所有傳入的 HTTP 請求重新導向至對應的 HTTPS 端點。您不需要設定任何設定，即可啟用重新導向 HTTP Web 請求。App Runner 會在將請求傳遞至應用程式容器執行個體之前終止 TLS。

  > **Note**
  > - HTTP 請求的請求逾時限制總計為 120 秒。120 秒包含應用程式讀取請求所需的時間，包括內文，並完成寫入 HTTP 回應。
  > - 請求讀取和回應逾時限制取決於您使用的應用程式。這些應用程式可能有自己的內部逾時，例如適用於 Python 的 HTTP 伺服器 Gunicorn，具有 30 秒的預設逾時限制。在這種情況下，應用程式的逾時限制會覆寫 App Runner 120 秒的逾時限制。
  > - 您不需要設定 TLS 密碼套件或任何其他參數，因為 App Runner 是全受管服務， 會為您管理 TLS 終止。

- 無狀態應用程式 – App Runner 目前不支援有狀態的應用程式。因此，App Runner 不保證狀態持久性超過處理單一傳入 Web 請求的持續時間。
- 儲存 – App Runner 會根據傳入流量，為您的 App Runner 應用程式自動擴展或縮減執行個體。您可以為 App Runner 應用程式設定 Auto Scaling 選項。由於處理 Web 請求的目前作用中執行個體數量是以傳入流量為基礎，App Runner 無法保證檔案可以持續超過單一請求的處理。因此，App Runner 會將容器執行個體中的檔案系統實作為暫時性儲存，這表示檔案是暫時性的。例如，當您暫停和繼續 App Runner 服務時，檔案不會保留。

App Runner 為您提供 3 GB 的暫時性儲存，並使用 3 GB 的暫時性儲存的一部分，用於執行個體上的已提取、壓縮和未壓縮的容器映像。App Runner 服務可以使用剩餘的暫時性儲存。不過，由於其無狀態性質，這不是永久的儲存體。

> **Note**
>
> 在某些情況下，儲存檔案會在請求之間保留。例如，如果下一個請求落在同一個執行個體上，儲存檔案將會保留。在某些情況下，跨請求儲存檔案的持久性可能很有用。例如，在處理請求時，如果未來請求可能需要，您可以快取應用程式下載的檔案。這可能會加速未來的請求處理，但無法保證速度增加。您的程式碼不應假設先前請求中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進行保證快取，請使用 Amazon ElastiCache 等服務。

- 環境變數 – 根據預設，App Runner 會在您的容器執行個體中提供`PORT`環境變數。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。您也可以參考存放在 AWS Secrets Manager或 AWS Systems Manager 參數存放區中的敏感資料做為環境變數。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 AWS 服務 APIs 或其中一個 AWS SDKs 呼叫任何 服務，請使用 AWS Identity and Access Management (IAM) 建立執行個體角色。然後，在建立 App Runner 服務時將其連接到該服務。在執行個體角色中包含程式碼所需的所有 AWS 服務動作許可。如需詳細資訊，請參閱the section called “執行個體角色”。

程式碼開發指導方針

為 App Runner Web 應用程式開發程式碼時，請考慮這些準則。

- 修補容器映像 – 提供容器映像時，您必須負責定期更新和修補這些映像。App Runner 管理基礎設施時，您應該確保所提供容器映像的安全性和up-to-date狀態。如需詳細資訊，請參閱 AWS App Runner 文件
- 設計無狀態程式碼 – 將您部署至 App Runner 服務的 Web 應用程式設計為無狀態。您的程式碼應該假設沒有任何狀態持續超過處理單一傳入 Web 請求的持續時間。
- 刪除暫存檔案 – 當您建立檔案時，這些檔案會存放在檔案系統上，並佔用您服務儲存配置的一部分。為了避免out-of-storage錯誤，請勿長時間保留暫存檔案。在進行檔案快取決策時，平衡儲存體大小與請求處理速度。

- 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接聽連接埠上接聽請求，並在啟動後五分鐘內正常運作。在啟動期間，App Runner 執行個體會根據您的 vCPU 組態配置虛擬 CPU (vCPU)。如需可用 vCPU 組態的詳細資訊，請參閱 the section called “App Runner 支援的組態”。

  執行個體成功啟動後，會進入閒置狀態並等待請求。您根據執行個體啟動持續時間付費，每個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 App Runner 主控台

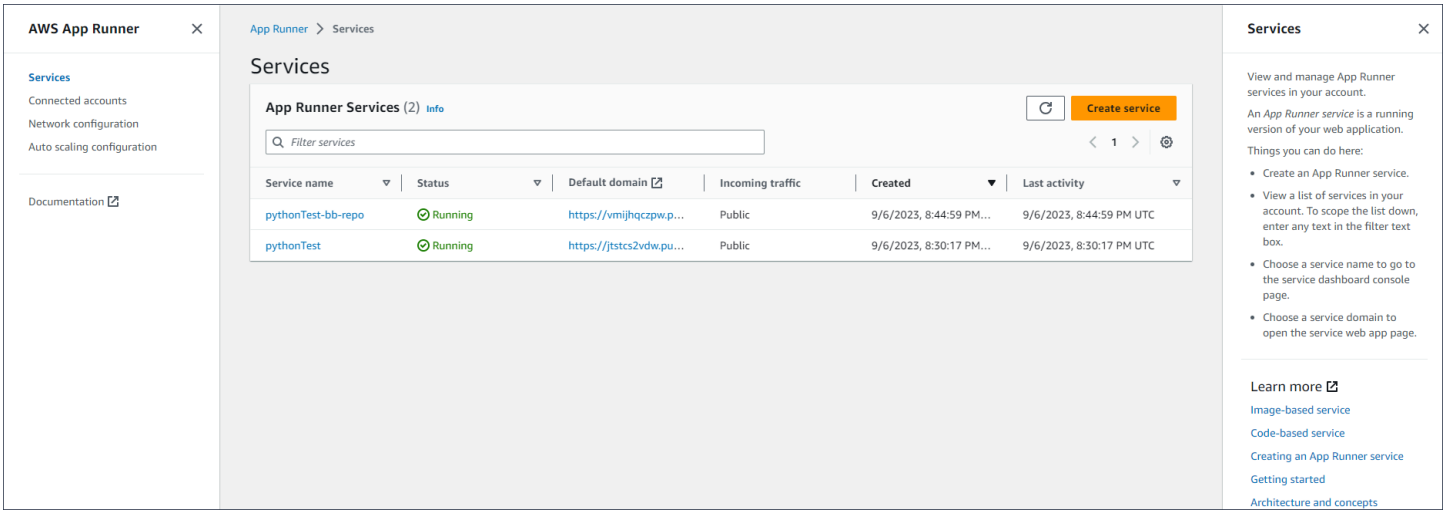
使用 AWS App Runner 主控台來建立、管理和監控您的 App Runner 服務和相關資源，例如連線帳戶。您可以檢視現有的服務、建立新的服務，以及設定服務。您可以檢視 App Runner 服務的狀態，以及檢視日誌、監控活動和追蹤指標。您也可以導覽至 服務的網站或來源儲存庫。

下列各節說明 主控台的配置和功能，並指出相關資訊。

整體主控台配置

App Runner 主控台有三個區域。從左到右：

- 導覽窗格 – 可摺疊或展開的側邊窗格。使用它來選擇您要使用的頂層主控台頁面。
- 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任務。
- 說明窗格 – 提供詳細資訊的側邊窗格。展開它以取得有關您所處頁面的說明。或選擇主控台頁面上的任何資訊連結，以取得內容說明。



服務頁面

服務頁面列出您帳戶中的 App Runner 服務。您可以使用篩選條件文字方塊縮小清單的範圍。

前往服務頁面

1. 開啟 App Runner 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。

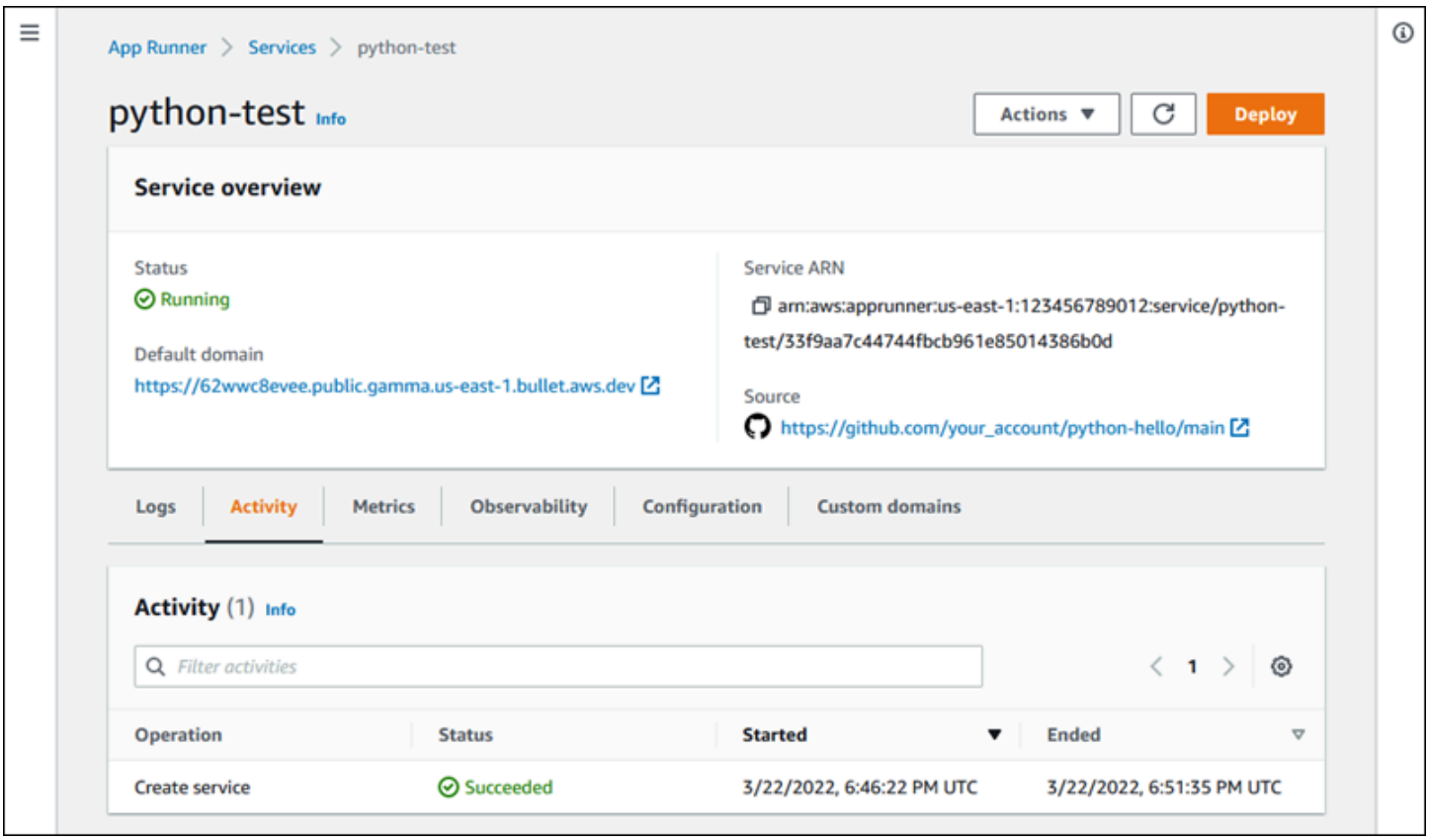
您可以在這裡執行的動作：

- 建立 App Runner 服務。如需詳細資訊，請參閱the section called “建立”。
- 選擇服務名稱以前往服務儀表板主控台頁面。
- 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

您可以檢視 App Runner 服務的相關資訊，並從服務破折號頁面進行管理。在頁面頂端，您可以看到服務名稱。

若要前往服務儀表板，請導覽至服務頁面（請參閱上一節），然後選擇您的 App Runner 服務。



服務概觀區段提供 App Runner 服務和應用程式的基本詳細資訊。您可以在這裡執行的動作：

- 檢視狀態、運作狀態和 ARN 等服務詳細資訊。
- 導覽至預設網域：App Runner 為服務中執行的 Web 應用程式提供的網域。這是 App Runner 所擁有`awsapprunner.com`網域中的子網域。
- 導覽至部署至服務的來源儲存庫。

- 開始將來源儲存庫部署到您的 服務。
- 暫停、繼續和刪除您的服務。

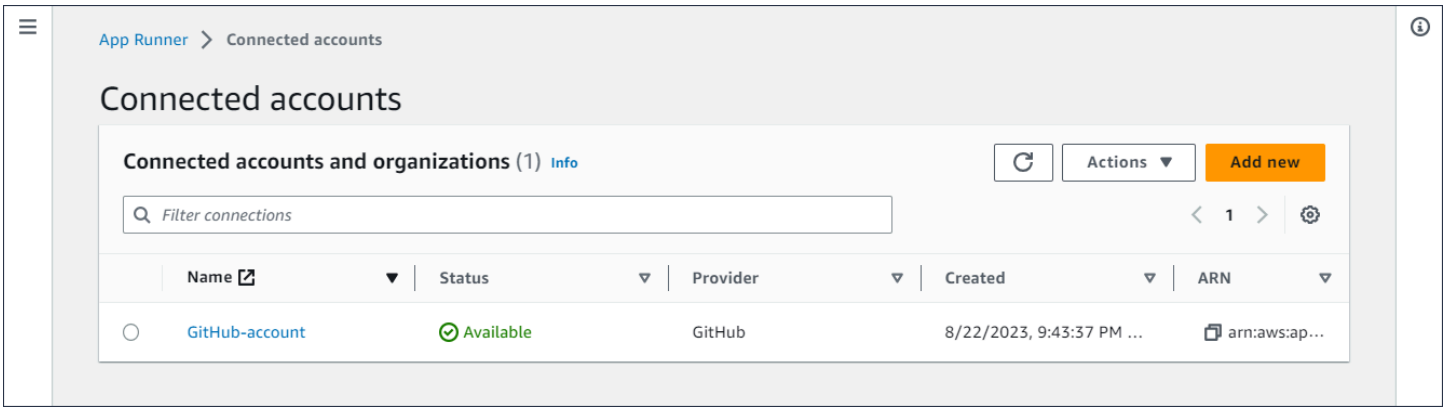
服務概觀下方的索引標籤適用於服務管理和可觀測性。

已連線帳戶頁面

已連線帳戶頁面會列出您帳戶中來源碼儲存庫供應商的 App Runner 連線。您可以使用篩選條件文字方塊縮小清單的範圍。如需連線帳戶的詳細資訊，請參閱 the section called “連線”。

若要前往已連線帳戶頁面

1. 開啟 App Runner 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，選擇已連線帳戶。



您可以在這裡執行的動作：

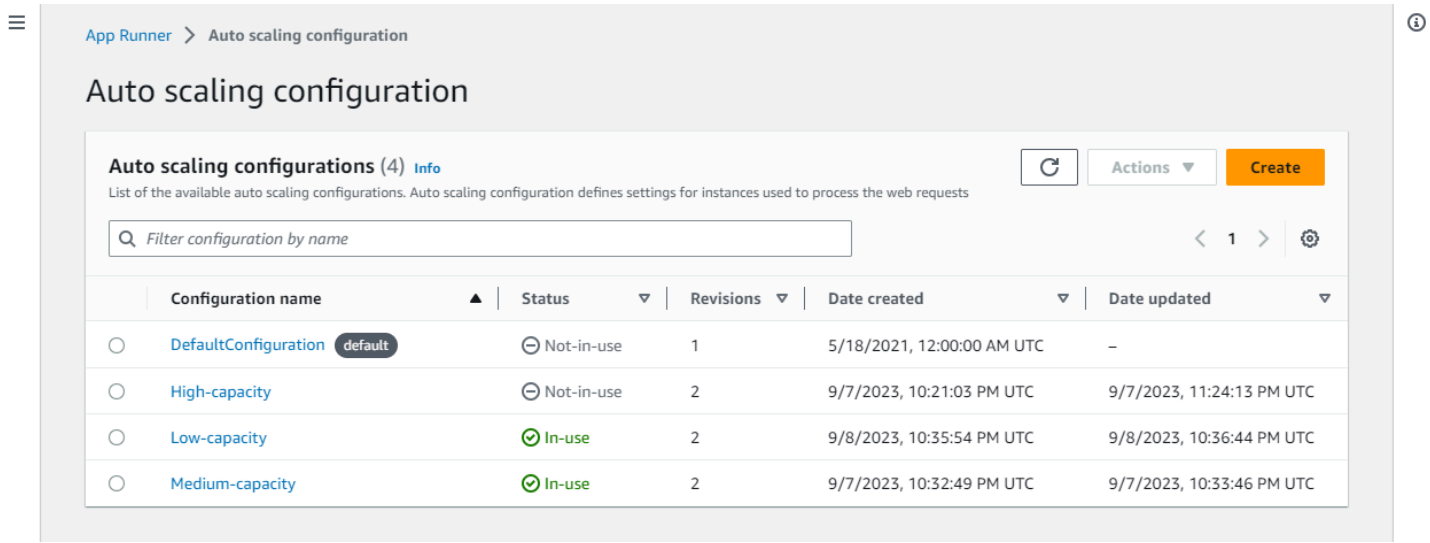
- 檢視您帳戶中的儲存庫提供者連線清單。若要縮小清單範圍，請在篩選文字方塊中輸入任何文字。
- 選擇連線名稱以前往相關的提供者帳戶或組織。
- 選取連線以完成您剛建立之連線的交握（做為建立服務的一部分），或刪除連線。

Auto Scaling 組態頁面

自動調整規模組態頁面列出您在帳戶中設定的自動調整規模組態。您可以設定幾個參數來調整自動擴展行為，並將其儲存在不同的組態中，以供稍後指派給一或多個 App Runner 服務。您可以使用篩選條件文字方塊縮小清單的範圍。如需自動擴展組態的詳細資訊，請參閱 管理服務的自動擴展。

若要前往 Auto Scaling 組態頁面

1. 開啟 App Runner 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，選擇自動擴展組態。



您可以在這裡執行的動作：

- 檢視您帳戶中現有的自動擴展組態清單。
- 建立新的自動擴展組態或現有組態的修訂。
- 將自動擴展組態設定為您建立之新服務的預設值。
- 刪除組態。
- 選取要導覽至 Auto Scaling 修訂面板以管理修訂的組態名稱。

管理您的 App Runner 服務

本章說明如何管理您的 AWS App Runner 服務。在本章中，您將了解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到您的服務，以及暫停和繼續您的服務來控制 Web 服務的可用性。您也會了解如何管理服務的其他層面，例如連線和自動擴展。

主題

- 建立 App Runner 服務
- 重建失敗的 App Runner 服務
- 將新的應用程式版本部署至 App Runner
- 設定 App Runner 服務
- 管理 App Runner 連線
- 管理 App Runner 自動擴展
- 管理 App Runner 服務的自訂網域名稱
- 暫停和繼續 App Runner 服務
- 刪除 App Runner 服務

建立 App Runner 服務

AWS App Runner 會自動從容器映像或原始程式碼儲存庫轉換到自動擴展的執行中 Web 服務。您可以將 App Runner 指向來源映像或程式碼，只指定少量的必要設定。App Runner 會視需要建置您的應用程式、佈建運算資源，並部署您的應用程式以在其上執行。

當您建立服務時，App Runner 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，主控台會隱含地建立連線資源。如需 App Runner 資源類型的詳細資訊，請參閱 the section called “App Runner 資源”。這些資源類型具有與每個 中您的帳戶相關聯的配額 AWS 區域。如需詳細資訊，請參閱the section called “App Runner 資源配額”。

根據來源類型和提供者，建立服務的程序存在細微差異。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程序。如需使用程式碼範例啟動基本程序，請參閱 開始使用。

先決條件

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。
- 請確定您的應用程式來源已就緒。您可以使用 GitHub 中的程式碼儲存庫、Bitbucket 或 Amazon Elastic Container Registry (Amazon ECR) 中的容器映像來建立 App Runner 服務。

建立服務

本節會逐步解說兩種 App Runner 服務類型的建立程序：以原始碼為基礎，以及以容器映像為基礎。

> **Note**
>
> 如果您為服務建立傳出流量 VPC 連接器，接下來的服務啟動程序將經歷一次性延遲。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 Networking with App Runner 章節一次性延遲中的 。

從程式碼儲存庫建立服務

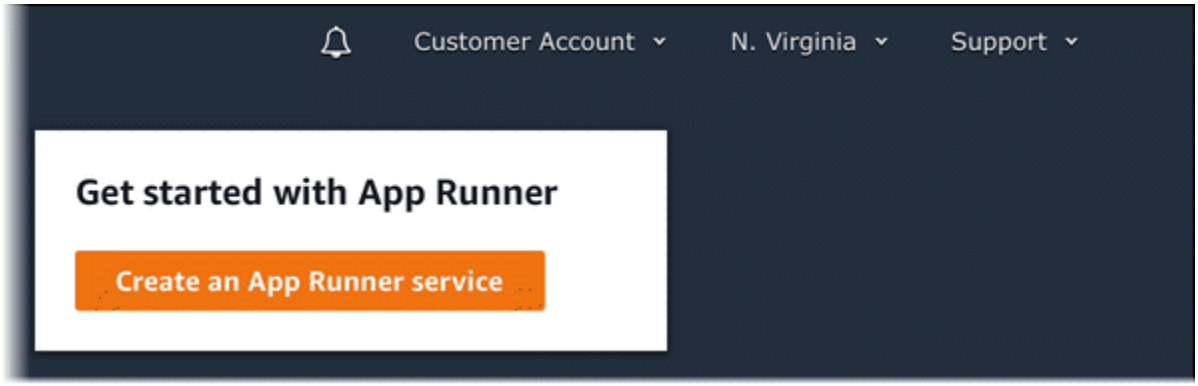
下列各節說明如何在來源為 GitHub 或 Bitbucket 中的程式碼儲存庫時建立 App Runner 服務。當您使用程式碼儲存庫時，App Runner 必須連線到提供者組織或帳戶。因此，您需要協助建立此連線。如需 App Runner 連線的詳細資訊，請參閱 the section called “連線”。

當您建立服務時，App Runner 會建置 Docker 映像，其中包含您的應用程式程式碼和相依性。然後，它會啟動執行此映像之容器執行個體的服務。

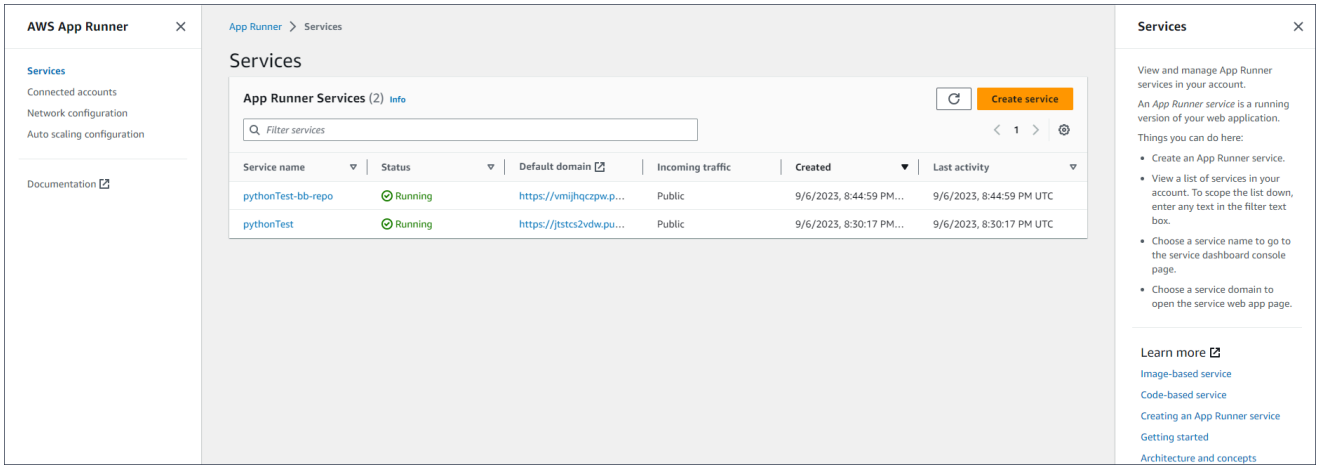
使用 App Runner 主控台從程式碼建立服務

使用主控台建立 App Runner 服務

1. 設定您的原始程式碼。
   a. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
   b. 如果 尚 AWS 帳戶 無任何 App Runner 服務，則會顯示主控台首頁。選擇建立 App Runner 服務。



如果 AWS 帳戶 有現有的服務，則會顯示包含服務清單的服務頁面。選擇 Create service (建立服務)。



c. 在來源和部署頁面的來源區段中，針對儲存庫類型，選擇來源碼儲存庫。

d. 選取提供者類型。選擇 GitHub 或 Bitbucket。

e. 接著，為您之前使用的提供者選取帳戶或組織，或選擇新增。然後，完成提供程式碼儲存庫登入資料，以及選擇要連線的帳戶或組織的程序。

f. 針對儲存庫，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. 對於來源目錄，在存放應用程式程式碼和組態檔案的來源儲存庫中輸入目錄。

> **Note**
>
> 組建和啟動命令會從您指定的來源目錄執行。App Runner 會將路徑視為根的絕對值。如果您不在此處指定值，目錄會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。

   如需部署方法的詳細資訊，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. 設定應用程式建置。

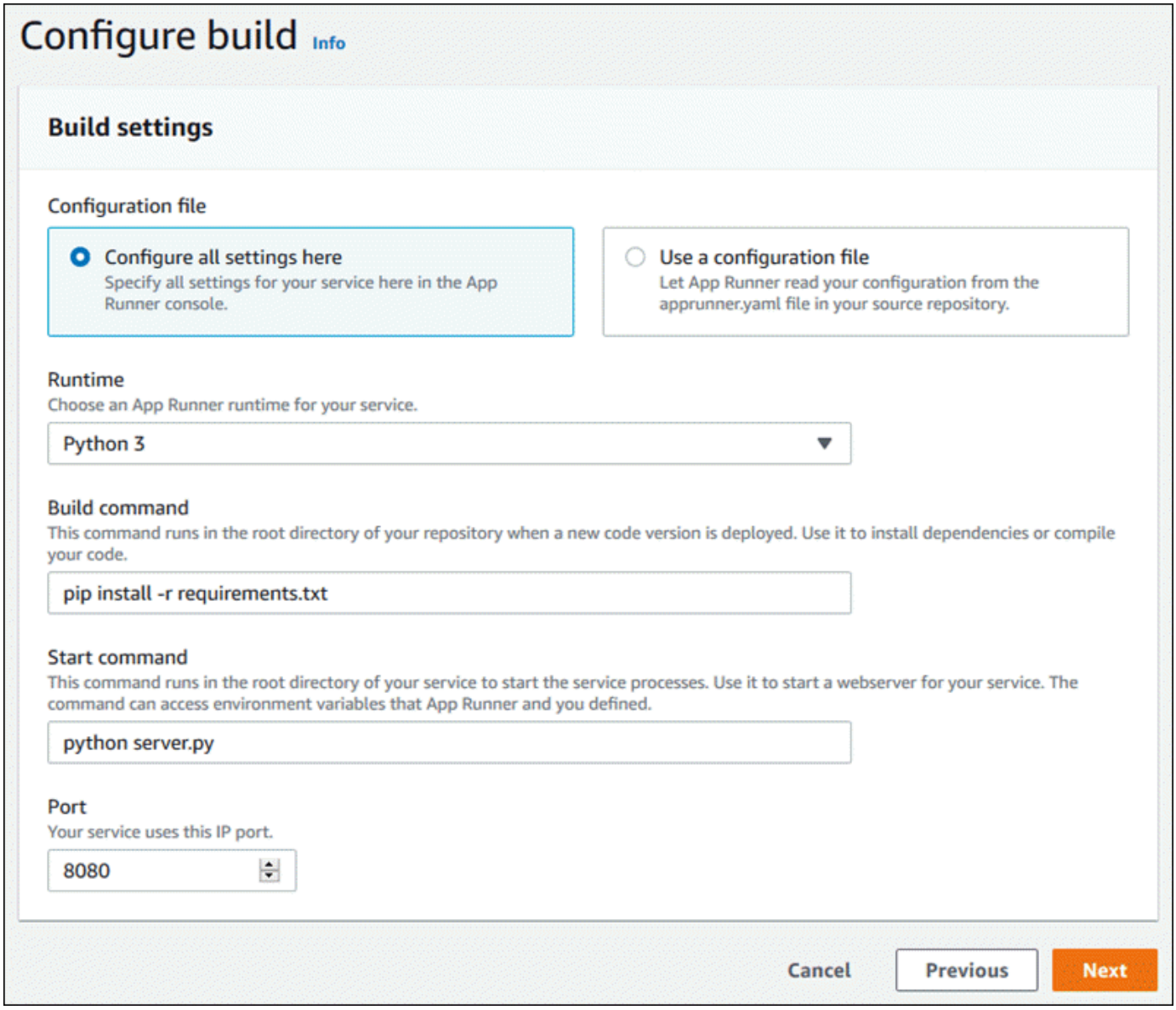
   a. 在設定建置頁面上，針對組態檔案，如果您的儲存庫不包含 App Runner 組態檔案，請選擇此處設定所有設定，或者如果包含，則選擇使用組態檔案。

   > **Note**
   >
   > App Runner 組態檔案是將建置組態維護為應用程式來源一部分的一種方式。當您提供時，App Runner 會從 檔案讀取一些值，而且不會讓您在 主控台中設定這些值。

   b. 提供下列建置設定：

      - 執行時間：為您的應用程式選擇特定的受管執行時間。
      - Build 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。
      - Start 命令 – 輸入啟動 Web 服務的命令。
      - 連接埠：輸入 Web 服務接聽的 IP 連接埠。

   c. 選擇下一步。



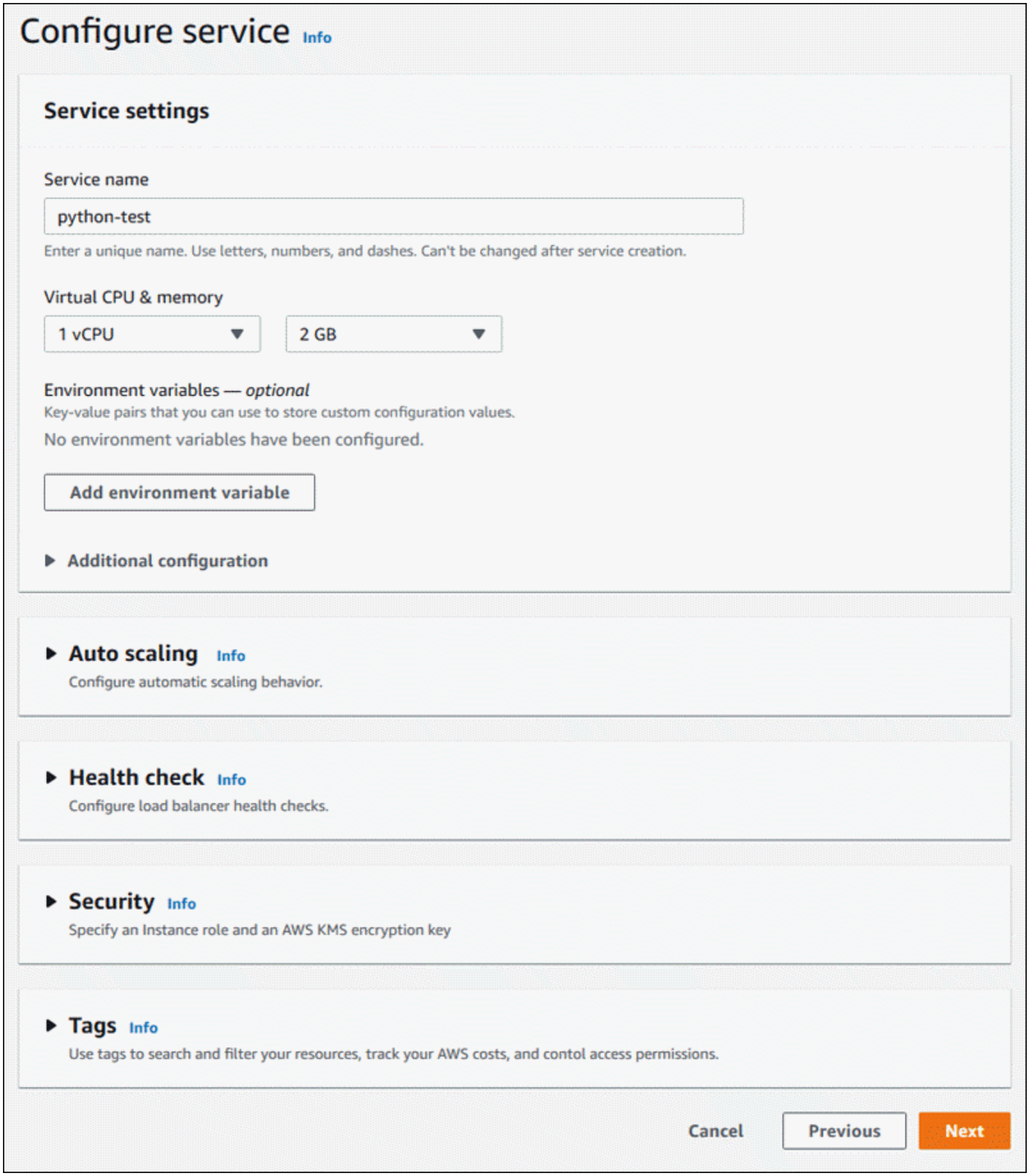
4. 設定您的服務。

   a. 在設定服務頁面上的服務設定區段中，輸入服務名稱。

   > **Note**
   >
   > 所有其他服務設定皆為選用，或具有主控台提供的預設值。

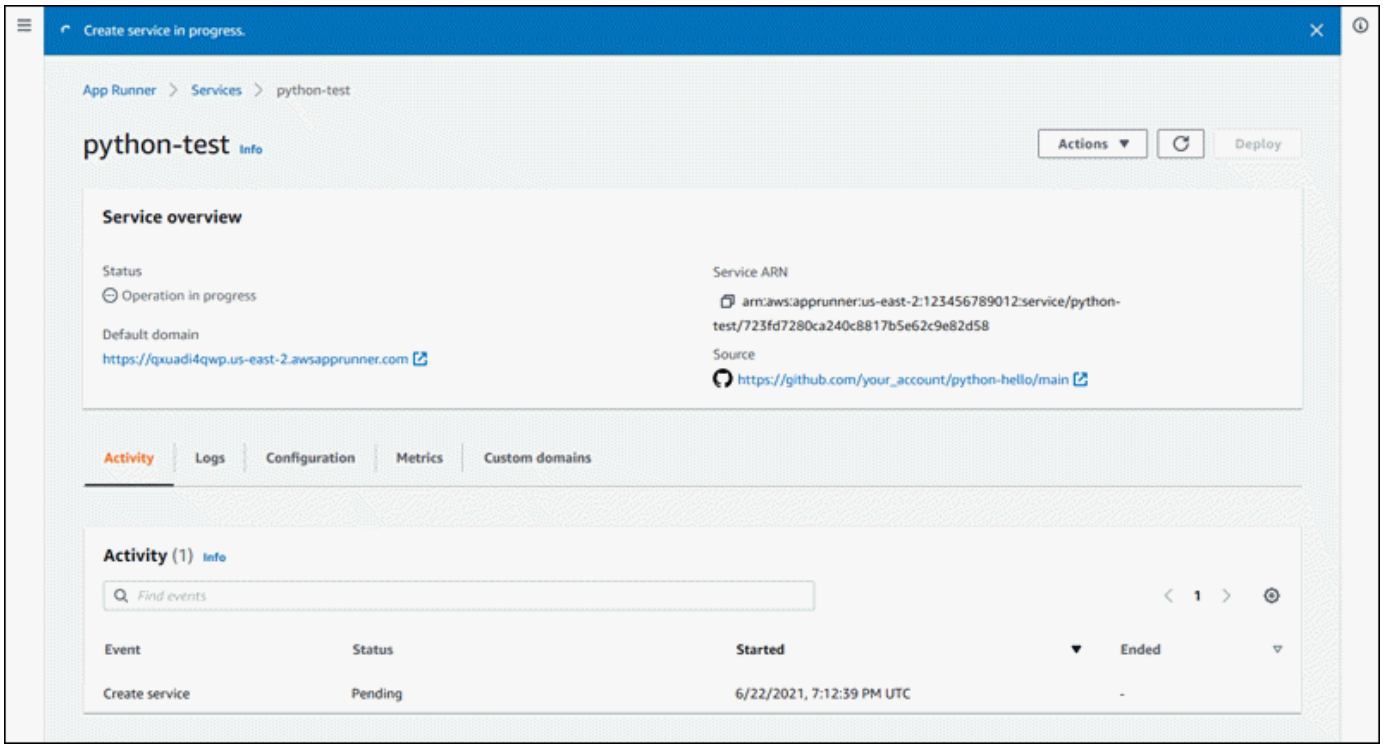
   b. 選擇性地變更或新增其他設定，以符合您的應用程式需求。

   c. 選擇下一步。



5. 在檢閱和建立頁面上，驗證您輸入的所有詳細資訊，然後選擇建立和部署。

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其是否正常運作。

使用 App Runner API 或 從程式碼建立服務 AWS CLI

若要使用 App Runner API 或 建立服務 AWS CLI，請呼叫 `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來源碼儲存庫 (GitHub 或 Bitbucket) 的特定組織或帳戶建立服務，請先呼叫 CreateConnection。這會在 App Runner 與儲存庫提供者的組織或帳戶之間建立連線。如需 App Runner 連線的詳細資訊，請參閱 the section called “連線”。

如果呼叫傳回成功回應，且服務物件顯示 `"Status": "CREATING"`，則您的服務會開始建立。

如需範例呼叫，請參閱 AWS App Runner API 參考中的建立原始碼儲存庫服務

從 Amazon ECR 映像建立服務

下列各節顯示當您的來源是存放在 Amazon ECR 中的容器映像時，如何建立 App Runner 服務。Amazon ECR 是 AWS 服務。因此，若要根據 Amazon ECR 映像建立服務，您需要為 App Runner 提供包含必要 Amazon ECR 動作許可的存取角色。

> **Note**
>
> 存放在 Amazon ECR Public 中的映像可公開取得。因此，如果您的映像存放在 Amazon ECR Public 中，則不需要存取角色。

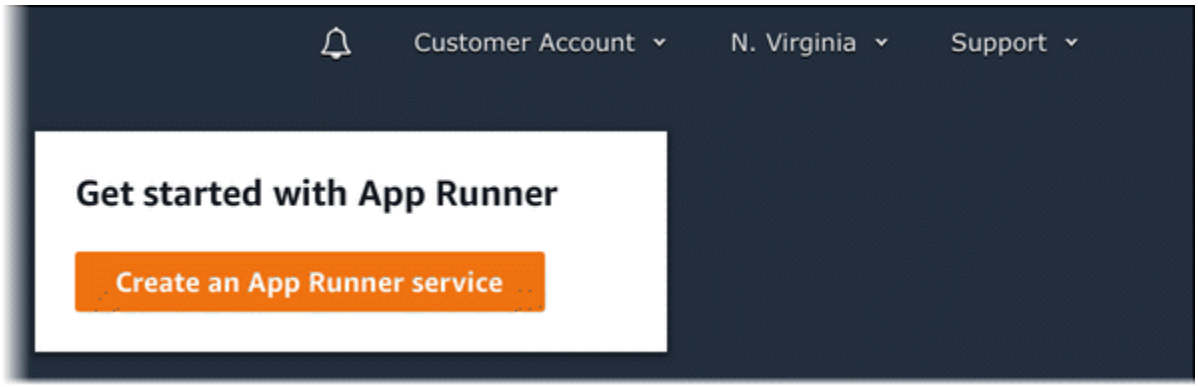
建立服務時，App Runner 會啟動執行您所提供映像之容器執行個體的服務。在這種情況下，沒有建置階段。

如需詳細資訊，請參閱影像型服務。

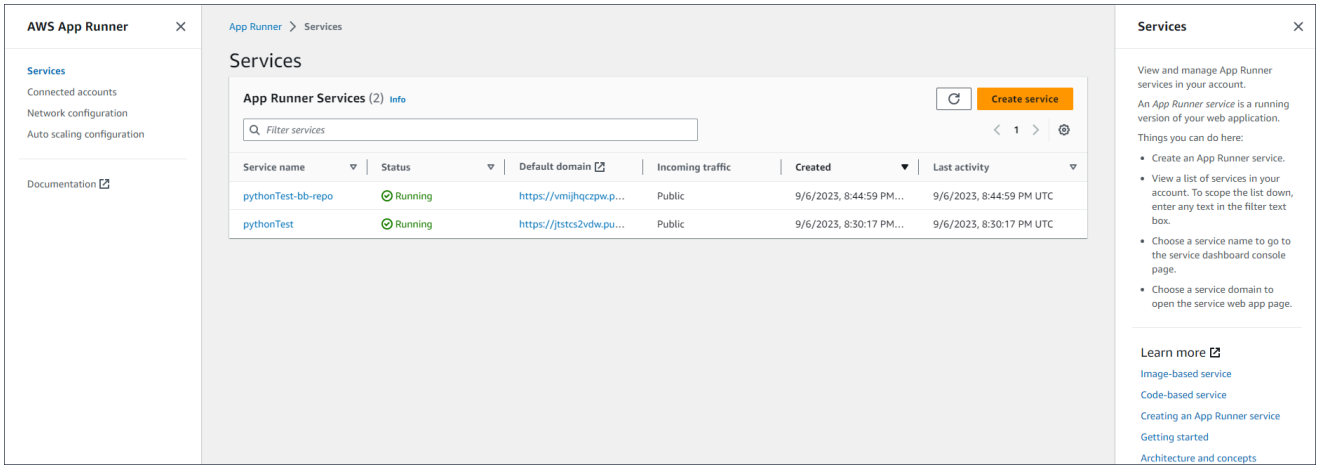
使用 App Runner 主控台從映像建立服務

使用主控台建立 App Runner 服務

1. 設定您的原始程式碼。
   a. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
   b. 如果 尚 AWS 帳戶 無任何 App Runner 服務，則會顯示主控台首頁。選擇建立 App Runner 服務。



   如果 AWS 帳戶 有現有的服務，則會顯示服務頁面，其中包含您的服務清單。選擇 Create service (建立服務)。



c. 在來源和部署頁面的來源區段中，針對儲存庫類型，選擇容器登錄檔。

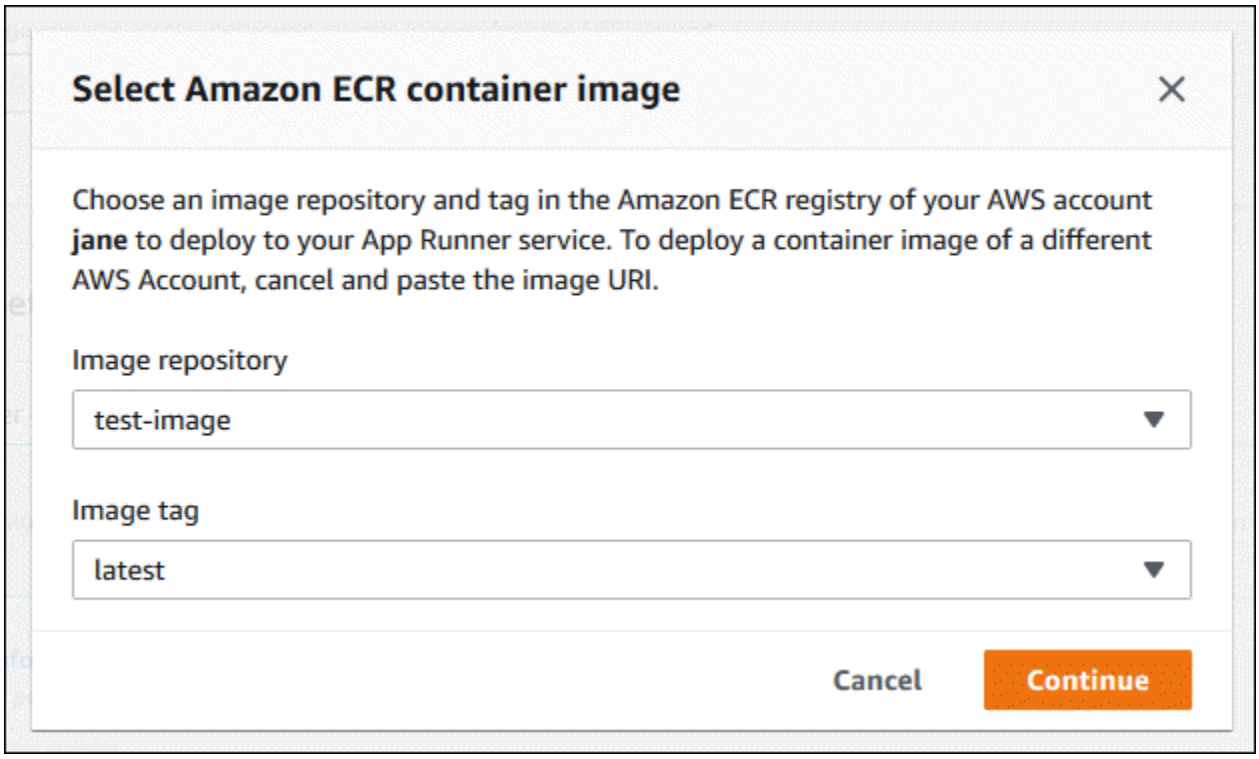
d. 針對提供者，選擇存放映像的提供者：

- Amazon ECR – 存放在 Amazon ECR 中的私有映像。
- Amazon ECR Public – 存放在 Amazon ECR Public 中的可公開讀取映像。

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，針對映像儲存庫，選取包含映像的儲存庫。

g. 針對映像標籤，選取您要部署的特定映像標籤（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**
   >
   > App Runner 不支援 Amazon ECR 公有映像的自動部署，以及 Amazon ECR 儲存庫中屬於您服務所在帳戶以外之 AWS 帳戶的映像的自動部署。

   如需部署方法的詳細資訊，請參閱 the section called “部署方法”。

   b. 【Amazon ECR 提供者】 對於 ECR 存取角色，選擇帳戶中現有的服務角色，或選擇建立新的角色。如果您使用的是手動部署，您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，輸入服務名稱和您的服務網站接聽的 IP 連接埠。

   > **Note**
   >
   > 所有其他服務設定皆為選用，或具有主控台提供的預設值。

   b. （選用）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

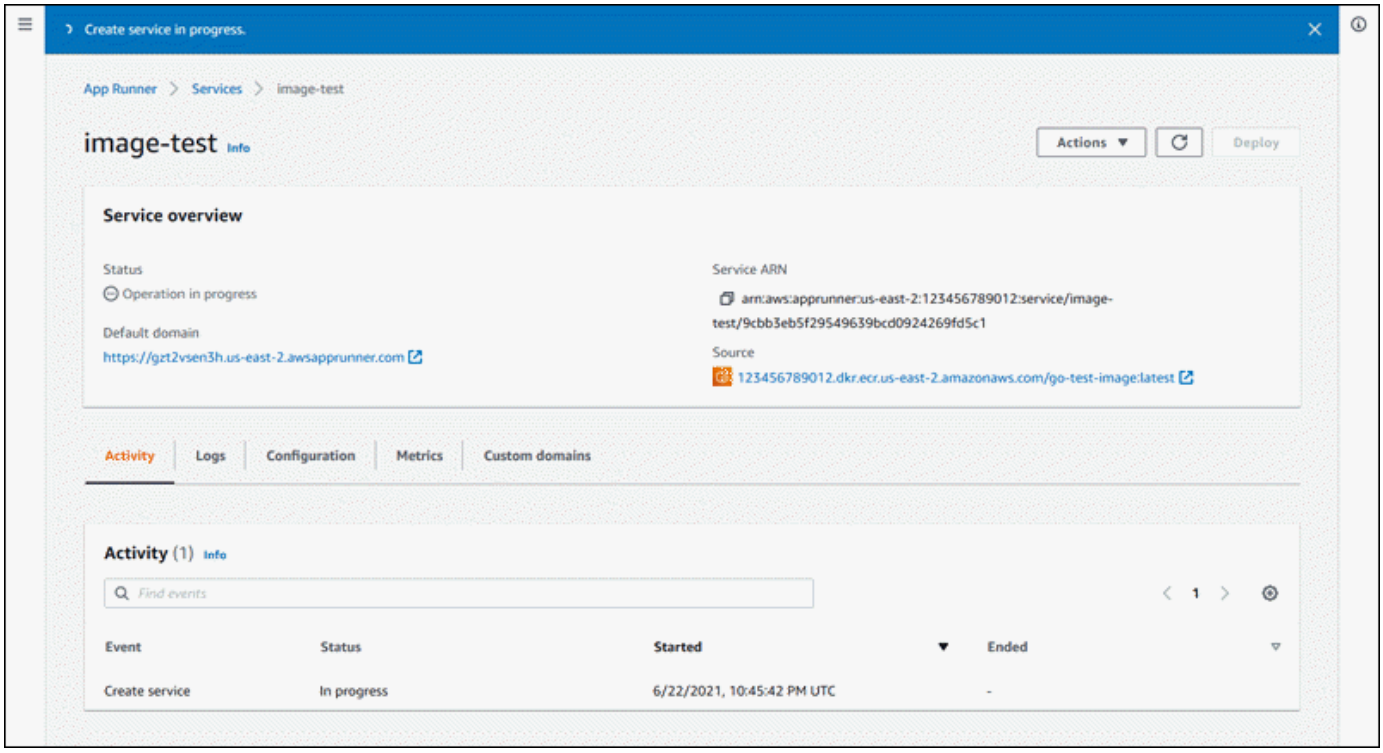
▶ **Tags** Info

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

Cancel Previous Next

4. 在檢閱和建立頁面上，驗證您輸入的所有詳細資訊，然後選擇建立和部署。

   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。



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其是否正常運作。

使用 App Runner API 或 從映像建立服務 AWS CLI

若要使用 App Runner API 或 建立服務 AWS CLI，請呼叫 CreateService API 動作。

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

如需範例呼叫，請參閱 AWS App Runner API 參考中的建立來源映像儲存庫服務

重建失敗的 App Runner 服務

如果您在建立 App Runner 服務時收到無法建立的錯誤，您可以執行下列其中一項操作。

- 請依照 中的步驟the section called “無法建立服務”來識別錯誤的原因。

- 如果您在來源或組態中發現錯誤，請進行必要的變更，然後重建您的服務。
- 如果 App Runner 的暫時問題導致您的服務失敗，請重建失敗的服務，而不對來源或組態進行任何變更。

您可以透過 App Runner 主控台或 App Runner API 或 AWS CLI 重建失敗的服務。

使用 App Runner 主控台重建失敗的 App Runner 服務

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，請務必在重建服務之前識別並修正問題的根本原因。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，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. 選擇儲存並重新建置。

   您可以在服務頁面的日誌索引標籤上監控進度。

Rebuild without updates

如果暫時問題導致您的服務建立失敗，您可以重建服務，而無需修改其來源或組態設定。

在沒有更新的情況下重建失敗的服務

- 選擇服務頁面右上角的重建。

  您可以在服務頁面的日誌索引標籤上監控進度。
- 如果您的服務無法再次建立，請遵循中的疑難排解指示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 App Runner API 或 重建失敗的 App Runner 服務 AWS CLI

Rebuild with updates

若要重建失敗的服務：

1. 依照 中的指示the section called “無法建立服務”尋找錯誤的原因。
2. 對分支或來源儲存庫的映像或造成錯誤的組態進行必要的變更。
3. 使用新的原始碼儲存庫或來源映像儲存庫參數呼叫 UpdateService API 動作來重建。App Runner 會從原始程式碼儲存庫擷取最新的遞交。

Example使用更新重建

在下列範例中，正在更新以映像為基礎的服務的來源組態。的值`Port`已變更為 `80`。

更新以映像為基礎的 App Runner 服務`input.json`檔案

```
{
  "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 或 重建失敗的服務 AWS CLI，請呼叫 UpdateService API 動作，而不對服務的來源或組態進行任何變更。只有在您的服務建立因 App Runner 暫時問題而失敗時，才選擇重建而不進行更新。

將新的應用程式版本部署至 App Runner

當您在 中建立服務時 AWS App Runner，您可以設定應用程式來源 - 容器映像或來源儲存庫。App Runner 會佈建資源來執行您的服務，並將您的應用程式部署到這些服務。

本主題說明如何在有新版本可用時，將應用程式來源重新部署至 App Runner 服務。這可以是映像儲存庫中的新映像版本，或程式碼儲存庫中的新遞交。App Runner 提供兩種方法來部署至服務：自動和手動。

部署方法

App Runner 提供下列方法來讓您控制啟動應用程式部署的方式。

自動部署

當您需要服務的持續整合和部署 (CI/CD) 行為時，請使用自動部署。App Runner 會監控您的映像或程式碼儲存庫是否有變更。

映像儲存庫 – 每當您將新的映像版本推送至映像儲存庫，或將新的遞交推送至程式碼儲存庫時，App Runner 會自動將其部署至您的服務，而無需對您採取進一步動作。

程式碼儲存庫 – 每當您將新的遞交推送到在來源目錄中進行變更的程式碼儲存庫時，App Runner 都會部署整個儲存庫。由於只有來源目錄中的變更會觸發自動部署，因此請務必了解來源目錄位置如何影響自動部署的範圍。

- 頂層目錄（儲存庫根目錄）– 這是您建立服務時為來源目錄設定的預設值。如果您的來源目錄設定為此值，這表示整個儲存庫位於來源目錄中。因此，您推送到來源儲存庫的所有遞交都會在此情況下觸發部署。
- 任何不是儲存庫根目錄的目錄路徑（非預設）– 因為只有在來源目錄中推送的變更才會觸發自動部署，因此任何推送到不在來源目錄中的儲存庫的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外推送的變更。

> **Note**
>
> App Runner 不支援 Amazon ECR 公有映像的自動部署，以及 Amazon ECR 儲存庫中屬於您服務所在帳戶以外之 AWS 帳戶的映像的自動部署。

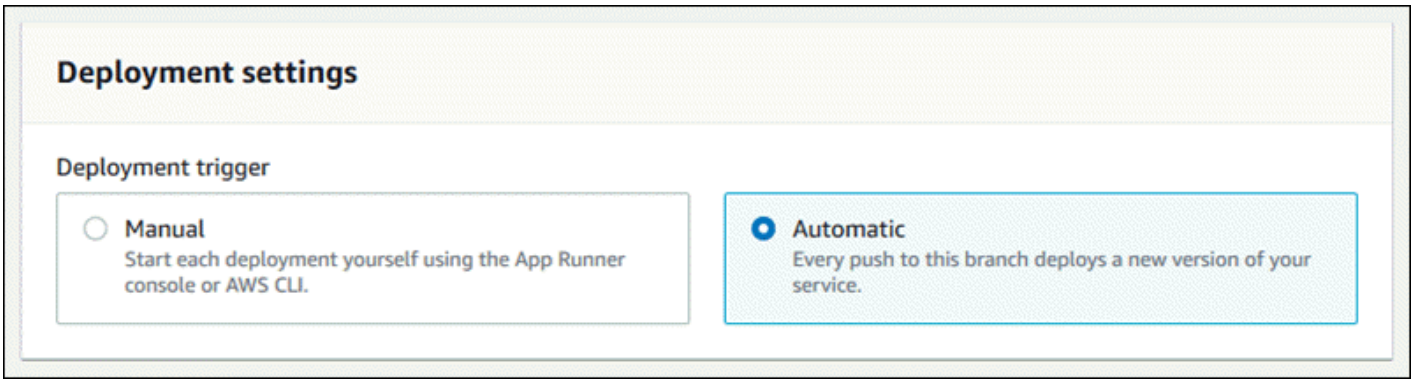
手動部署

當您想要明確啟動服務的每個部署時，請使用手動部署。如果您為服務設定的儲存庫有您要部署的新版本，您可以啟動部署。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 會從完整的儲存庫部署來源。

您可以透過下列方式設定 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，請在來源和部署組態頁面的部署設定區段中，選擇手動或自動。



- API 或 AWS CLI – 在呼叫 CreateService 或 UpdateService 動作時，將 SourceConfiguration 參數`AutoDeploymentsEnabled`的成員設定為 `False` 以進行手動部署或`True`自動部署。

> **比較自動和手動部署**
>
> 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的儲存庫。
> 這兩種方法之間的差異是觸發機制：
>
> - 從主控台部署、呼叫 或呼叫 App Runner API AWS CLI會觸發手動部署。以下手動部署章節提供這些程序。
> - 自動部署是由來源目錄內容內的變更所觸發。

手動部署

使用手動部署時，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本應用程式映像或程式碼時，您可以參考以下章節，了解如何使用 主控台和 API 執行部署。

> **Note**
>
> 當您執行手動部署時，App Runner 會從完整的儲存庫部署來源。

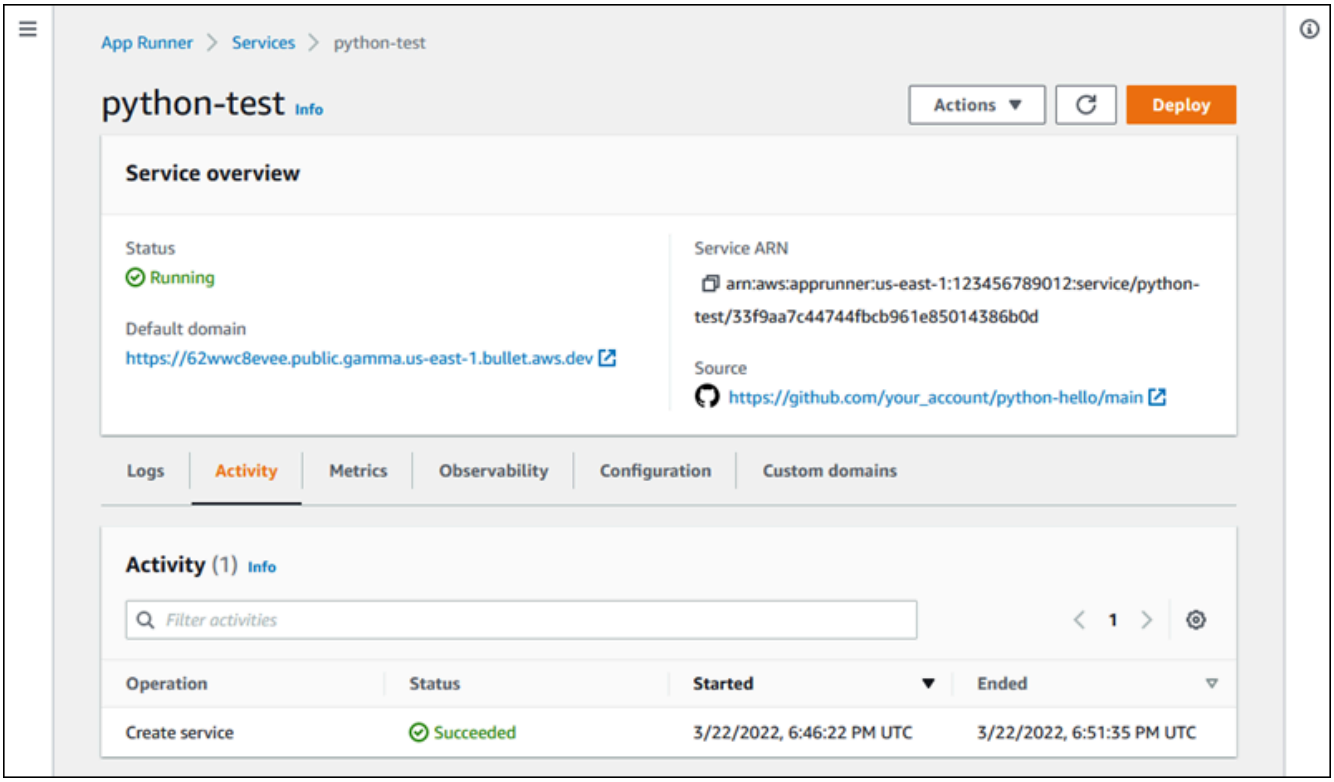
使用下列其中一種方法部署應用程式的版本：

App Runner console

使用 App Runner 主控台部署

1. 開啟 App Runner 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，然後選擇您的 App Runner 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：新版本的部署開始。在服務儀表板頁面上，服務狀態會變更為進行中的操作。

4. 等待部署結束。在服務儀表板頁面上，服務狀態應變更回執行中。
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，這是服務網站的 URL。檢查您的 Web 應用程式或與之互動，並驗證您的版本變更。

> **Note**
>
> 為了增強 App Runner 應用程式的安全性，在公有尾碼清單 (PSL) 中註冊 *.awsapprunner.com 網域。為了進一步提高安全性，如果您需要為 App Runner 應用程式設定預設網域名稱中的敏感 Cookie，建議您使用字__Host-首為 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 App Runner API 或 部署 AWS CLI，請呼叫 StartDeployment API 動作。要傳遞的唯一參數是您的服務 ARN。您已在建立服務時設定應用程式來源位置，而 App Runner 可以找到新版本。如果呼叫傳回成功回應，您的部署就會開始。

設定 App Runner 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、App Runner 主控台和 App Runner 組態檔案的服務組態。

主題

- 使用 App Runner API 或 設定您的服務 AWS CLI
- 使用 App Runner 主控台設定您的服務
- 使用 App Runner 組態檔案設定您的服務
- 為您的服務設定可觀測性
- 使用可分割資源設定服務設定
- 為您的服務設定運作狀態檢查

使用 App Runner API 或 設定您的服務 AWS CLI

API 定義可在服務建立後變更的設定。下列清單討論相關動作、類型和限制。

- UpdateService 動作 – 可在建立後呼叫 ，以更新某些組態設定。
  - 可以更新 – 您可以更新 `SourceConfiguration`、 `InstanceConfiguration`和 `HealthCheckConfiguration` 參數中的設定。不過，在 中`SourceConfiguration`，您無法將來源類型從程式碼切換到映像或反之亦然。您必須提供與建立服務時所提供的相同儲存庫參數。這是 `CodeRepository`或 `ImageRepository`。

    您也可以更新與服務相關聯之個別組態資源的下列 ARNs：

    - `AutoScalingConfigurationArn`
    - `VpcConnectorArn`
  - 無法更新 – 您無法變更 CreateService 動作中可用的 `ServiceName`和 `EncryptionConfiguration` 參數。它們建立後就無法變更。UpdateService 動作不包含這些參數。
  - API 與 檔案 – 您可以將 CodeConfiguration 類型的 `ConfigurationSource` 參數 （用於做為的一部分的原始碼儲存庫`SourceConfiguration`) 設定為 `Repository`。在此情況下，App Runner 會忽略 中的組態設定`CodeConfigurationValues`，並從儲存庫中的組態檔案讀取這些設定。如果您將 `ConfigurationSource` 設定為 `API`，App Runner 會從 API 呼叫取得所有組態設定，並忽略組態檔案，即使組態檔案存在。
- TagResource 動作 – 可在建立服務之後呼叫，以將標籤新增至服務或更新現有標籤的值。
- UntagResource 動作 – 可在建立服務之後呼叫，以從服務中移除標籤。

> **Note**
>
> 如果您為服務建立傳出流量 VPC 連接器，接下來的服務啟動程序將經歷一次性延遲。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 Networking with App Runner 章節一次性延遲中的 。

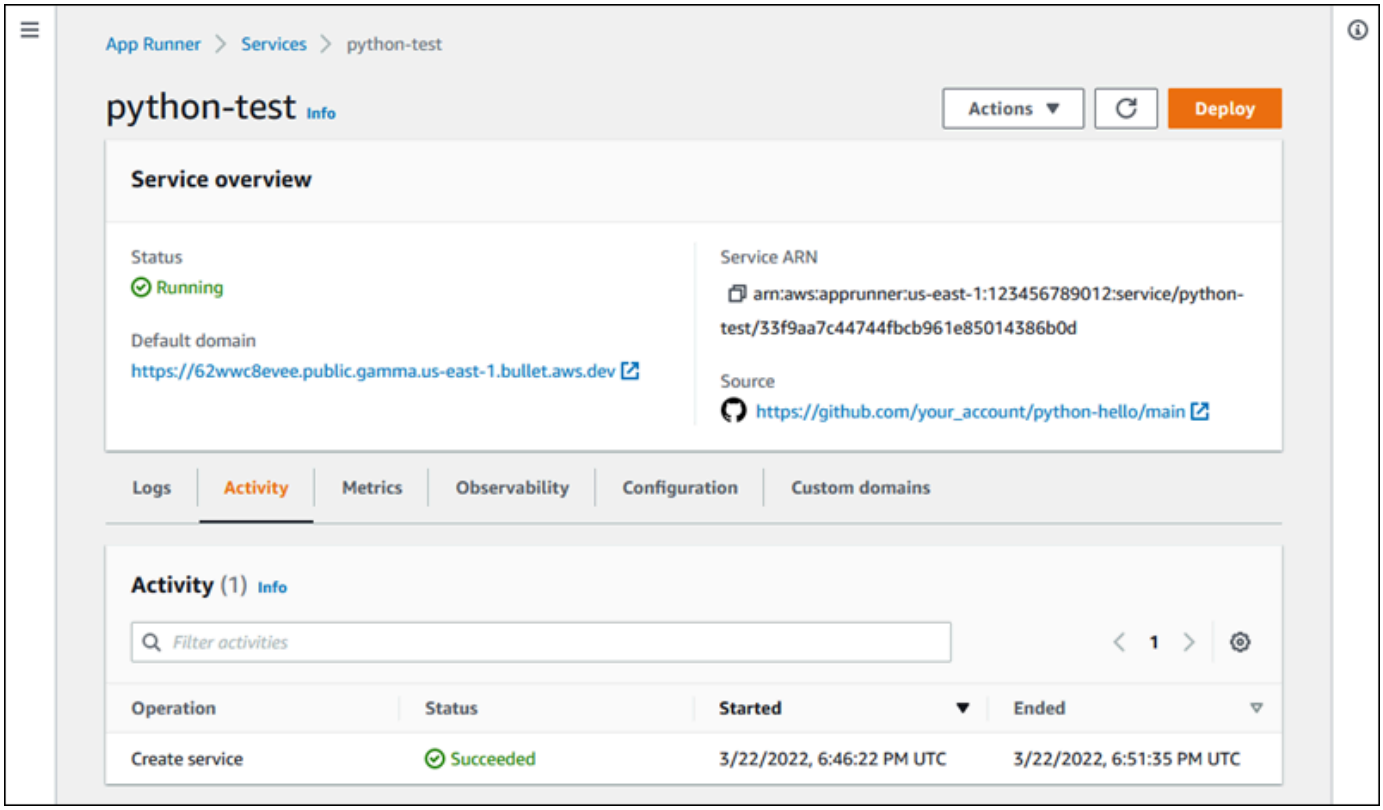
使用 App Runner 主控台設定您的服務

主控台使用 App Runner API 來套用組態更新。API 強制執行的更新規則，如上一節所定義，決定您可以使用 主控台設定的內容。某些在服務建立期間可用的設定稍後無法進行修改。此外，如果您決定使用組態檔案，則主控台中會隱藏其他設定，而 App Runner 會從檔案讀取這些設定。

設定您的服務

1. 開啟 App Runner 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，然後選擇您的 App Runner 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：主控台會以數個區段顯示服務的目前組態設定：來源和部署、設定建置和設定服務。
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，進行任何所需的變更，然後選擇儲存變更。

> **Note**
>
> 如果您為服務建立傳出流量 VPC 連接器，接下來的服務啟動程序將經歷一次性延遲。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 Networking with App Runner 章節一次性延遲中的 。

使用 App Runner 組態檔案設定您的服務

當您建立或更新 App Runner 服務時，您可以指示 App Runner 從您作為來源儲存庫的一部分提供的組態檔案中讀取一些組態設定。透過這樣做，您可以在來源控制下管理與原始程式碼相關的設定，以及程式碼本身。組態檔案也提供您無法使用主控台或 API 設定的特定進階設定。如需詳細資訊，請參閱App Runner 組態檔案。

> **Note**
>
> 如果您為服務建立傳出流量 VPC 連接器，接下來的服務啟動程序將經歷一次性延遲。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 Networking with App Runner 章節一次性延遲中的 。

為您的服務設定可觀測性

AWS App Runner 與數個 AWS 服務整合，為您的 App Runner 服務提供廣泛的可觀測性工具套件。如需詳細資訊，請參閱可觀測性。

App Runner 支援啟用一些可觀測性功能，並使用稱為 ObservabilityConfiguration 的可分割資源來設定其行為。您可以在建立或更新服務時提供可觀測性組態資源。當您建立新的 App Runner 服務時，App Runner 主控台會為您建立一個。提供可觀測性組態是選用的。如果您沒有提供，App Runner 會提供預設可觀測性組態。

您可以跨多個 App Runner 服務共用單一可觀測性組態，以確保它們具有相同的可觀測性行為。如需詳細資訊，請參閱the section called “組態資源”。

您可以使用可觀測性組態來設定下列可觀測性功能：

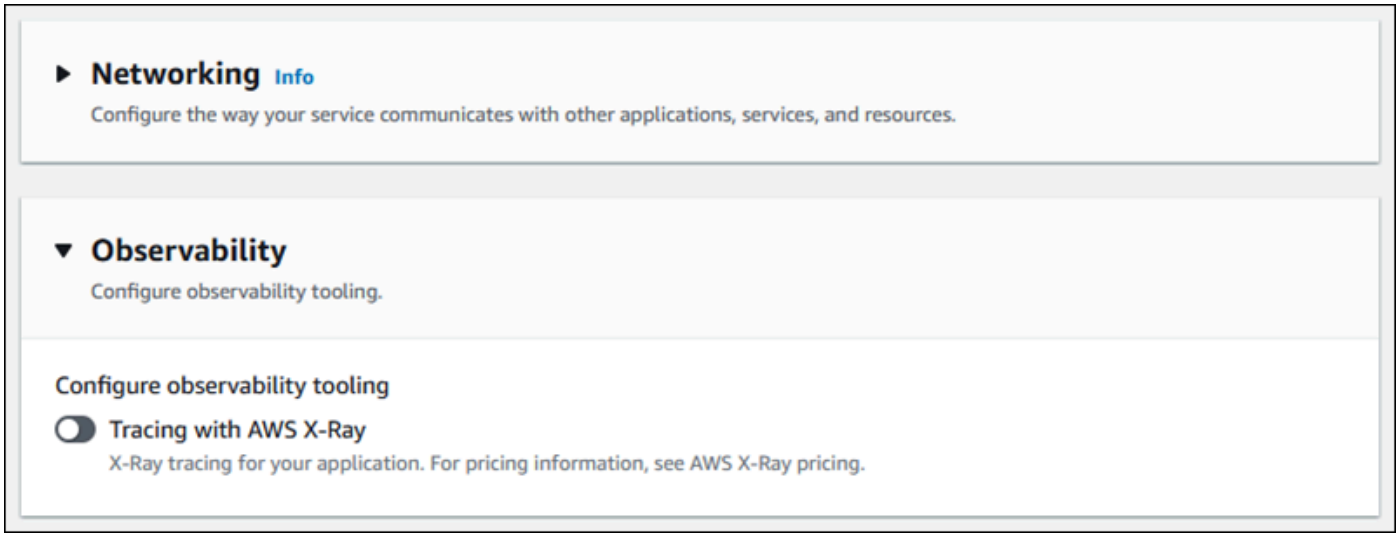
- 追蹤組態 – 追蹤應用程式所提供服務之請求及其發出之下游呼叫的設定。如需追蹤的詳細資訊，請參閱：the section called “追蹤 (X-Ray)”。

管理可觀測性

使用下列其中一種方法來管理 App Runner 服務的可觀測性：

App Runner console

當您使用 App Runner 主控台建立服務，或稍後更新其組態時，您可以為服務設定可觀測性功能。尋找主控台頁面上的可觀測性組態區段。



App Runner API or AWS CLI

當您呼叫 CreateService 或 UpdateService App Runner API 動作時，您可以使用 `ObservabilityConfiguration` 參數物件來啟用可觀測性功能，並為您的服務指定可觀測性組態資源。

使用下列 App Runner API 動作來管理您的可觀測性組態資源。

- CreateObservabilityConfiguration – 建立新的可觀測性組態或現有組態的修訂。
- ListObservabilityConfigurations – 傳回與您的 相關聯的可觀測性組態清單 AWS 帳戶，其中包含摘要資訊。
- 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或最新的作用中修訂。如果您達到 的可觀測性組態配額，您可能需要刪除不必要的可觀測性組態 AWS 帳戶。

使用可分割資源設定服務設定

對於某些功能，跨 AWS App Runner 服務共用組態是合理的。例如，您可能希望一組服務具有相同的自動擴展行為。或者，您可能想要對所有 服務進行相同的可觀測性設定。App Runner 可讓您使用個別的可分割資源來共用設定。您可以建立資源來定義功能的一組組態設定，然後將此組態資源的 Amazon Resource Name (ARN) 提供給一或多個 App Runner 服務。

App Runner 會實作下列功能的可共用組態資源：

- 自動擴展

- 可觀測性
- VPC 存取

每個功能的文件頁面提供可用設定和管理程序的相關資訊。

使用個別組態資源的功能會共用一些設計特性和考量事項。

- 修訂 – 有些組態資源可以有修訂。自動擴展和可觀測性是使用修訂的兩個組態資源的範例。在這些情況下，每個組態都有一個名稱和一個數值修訂。組態的多個修訂具有相同的名稱和不同的修訂編號。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。

  您使用名稱建立的第一個組態會取得修訂編號 1。名稱相同的後續組態會取得連續的修訂編號（以 2 開頭）。您可以將 App Runner 服務與特定組態修訂或組態的最新修訂建立關聯。
- 共用 – 您可以在多個 App Runner 服務之間共用單一組態資源。如果您想要在這些服務中維護相同的組態，這會很有用。特別是，如果您的資源支援修訂，您可以將多個 服務設定為使用組態的最新修訂。您可以只指定組態名稱，但不能指定修訂。當您更新服務時，您以這種方式設定的任何服務都會收到組態更新。如需組態變更的詳細資訊，請參閱 the section called “Configuration”。
- 資源管理 – 您可以使用 App Runner 來建立和刪除組態。您無法直接更新組態。相反地，對於支援修訂的資源，您可以建立新的現有組態名稱修訂，以有效地更新組態。

> **Note**
>
> 對於自動擴展，您可以使用 App Runner 主控台和 App Runner API 來建立組態和多個修訂。App Runner 主控台和 App Runner API 也可以刪除組態和修訂。如需詳細資訊，請參閱管理 App Runner 自動擴展。
>
> 對於其他組態類型，例如可觀測性組態，您只能使用 App Runner 主控台建立具有單一修訂的組態。若要建立更多修訂和刪除組態，您必須使用 App Runner API。

- 資源配額 – 您可以為每個組態資源擁有的唯一組態名稱和修訂數目設定配額 AWS 區域。如果您達到這些配額，您必須先刪除組態名稱或至少部分修訂，才能建立更多。對於自動擴展組態修訂，您可以使用 App Runner 主控台或 App Runner API 來刪除它們。如需詳細資訊，請參閱管理 App Runner 自動擴展。您必須使用 App Runner API 來刪除其他資源。如需配額的詳細資訊，請參閱 the section called “App Runner 資源配額”。
- 無資源成本 – 建立組態資源不會產生額外費用。您可能會為功能本身產生成本（例如，開啟 X-Ray 追蹤時需支付正常 AWS X-Ray 成本），但不會針對為 App Runner 服務設定功能的應用程式執行器組態資源收取費用。

為您的服務設定運作狀態檢查

AWS App Runner 透過執行運作狀態檢查來監控服務的運作狀態。預設的運作狀態檢查通訊協定為 TCP。App Runner 會 Ping 指派給您服務的網域。或者，您可以將運作狀態檢查通訊協定設定為 HTTP。App Runner 會將運作狀態檢查 HTTP 請求傳送至您的 Web 應用程式。

您可以設定與運作狀態檢查相關的一些設定。下表說明運作狀態檢查設定及其預設值。

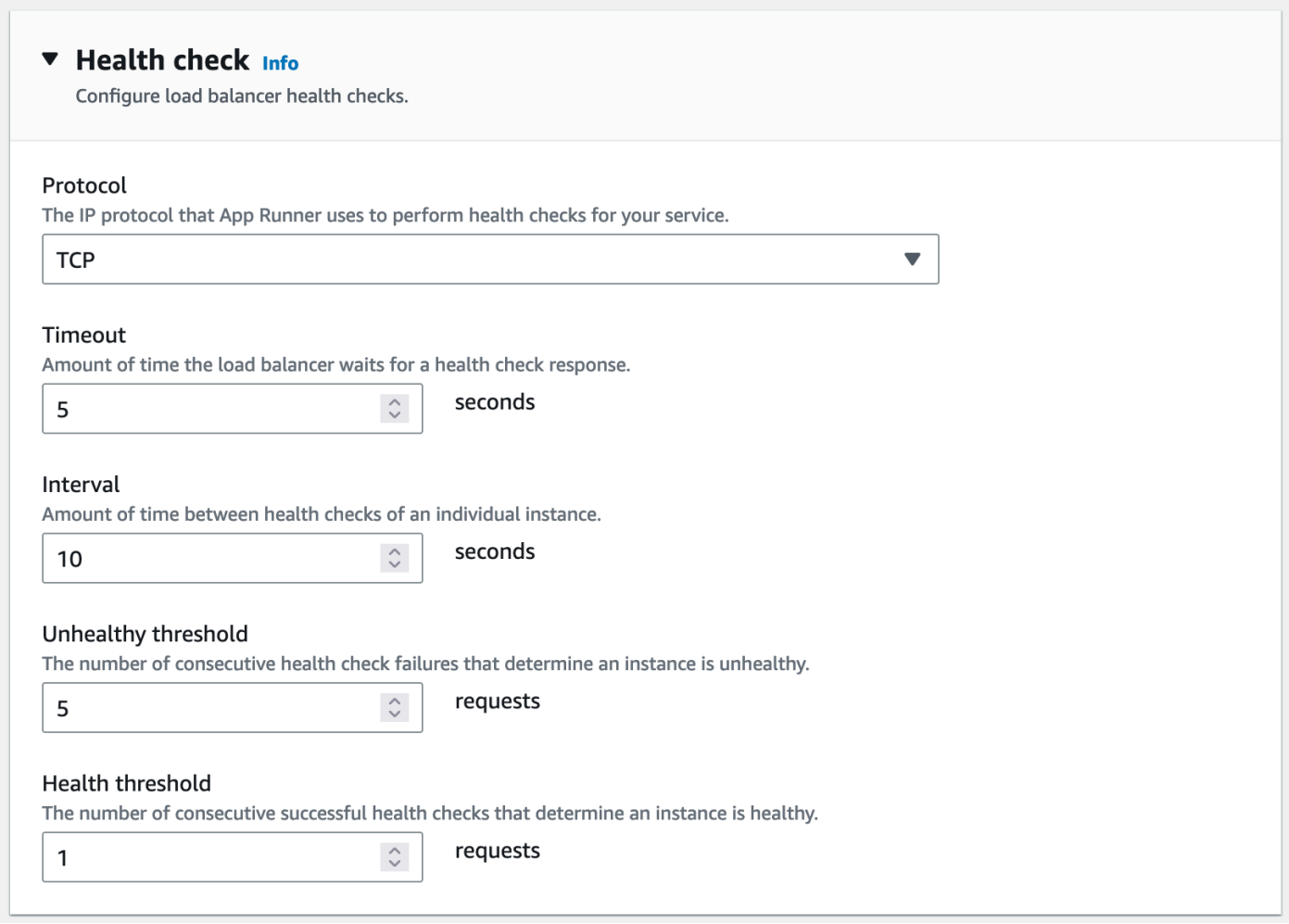
設定	描述	預設
通訊協定	App Runner 用來為您的服務執行運作狀態檢查的 IP 通訊協定。 如果您將通訊協定設定為 TCP，App Runner 會在應用程式接聽的連接埠 ping 指派給服務的預設網域。 如果您將通訊協定設定為 HTTP，App Runner 會將運作狀態檢查請求傳送至設定的路徑。	TCP
路徑	App Runner 傳送 HTTP 運作狀態檢查請求的 URL。僅適用於 HTTP 檢查。	/
Interval	運作狀態檢查之間的時間間隔 (以秒為單位)。	5
Timeout (逾時)	在判斷運作狀態檢查回應失敗之前等待運作狀態檢查回應的時間 (以秒為單位)。	2
運作狀態良好的閾值	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
運作狀態不佳閾值	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來設定 App Runner 服務的運作狀態檢查：

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，您可以設定運作狀態檢查設定。如需完整的主控台程序，請參閱 the section called “建立”和 the section called “Configuration”。在這兩種情況下，請尋找主控台頁面上的運作狀態檢查組態區段。



App Runner API or AWS CLI

當您呼叫 CreateService 或 UpdateService API 動作時，您可以使用 `HealthCheckConfiguration` 參數來指定運作狀態檢查設定。

如需 參數結構的相關資訊，請參閱 AWS App Runner API 參考中的 HealthCheckConfiguration。

管理 App Runner 連線

當您在 中建立服務時 AWS App Runner，您可以設定應用程式來源 - 容器映像或存放於供應商的來源儲存庫。App Runner 必須與提供者建立已驗證和授權的連線。然後，App Runner 可以讀取您的儲存

庫，並將其部署到您的服務。當您建立存取存放在 中的程式碼的服務時，App Runner 不需要建立連線 AWS 帳戶。

App Runner 會在稱為連線的 資源中維護連線資訊。App Runner 主控台和本指南也將連線稱為連線帳戶。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。以下是有關連線的一些重要資訊：

- 供應商 – App Runner 目前需要使用 GitHub 或 Bitbucket 的連線資源。
- 共用 – 您可以使用連線資源來建立多個使用相同儲存庫提供者帳戶的 App Runner 服務。
- 資源管理 – 在 App Runner 中，您可以建立和刪除連線。不過，您無法修改現有的連線。
- 資源配額 – 連線資源在每個 AWS 帳戶 中具有與 相關聯的設定配額 AWS 區域。如果您達到此配額，您可能需要先刪除連線，才能連線到新的供應商帳戶。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 “App Runner 資源配額”。

管理連線

使用下列其中一種方法管理您的 App Runner 連線：

App Runner console

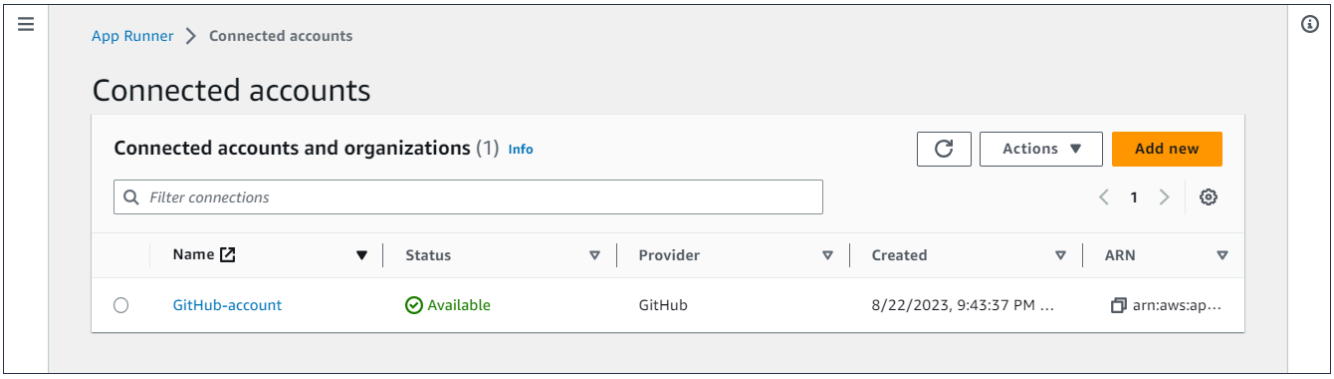
當您使用 App Runner 主控台建立服務時，請提供連線詳細資訊。您不需要明確建立連線資源。在 主控台中，您可以選擇連線至您之前連線的 GitHub 或 Bitbucket 帳戶，或連線至新帳戶。必要時，App Runner 會為您建立連線資源。對於新的連線，某些供應商會要求您先完成身分驗證交握，才能使用連線。主控台會引導您完成此程序。

主控台也有管理現有連線的頁面。如果您在建立服務時未這麼做，則可以完成連線的身分驗證交握。您也可以刪除不再使用的連線。下列程序說明如何管理儲存庫提供者連線。

管理帳戶中的連線

1. 開啟 App Runner 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，選擇已連線帳戶。

   然後，主控台會顯示您帳戶中儲存庫提供者連線的清單。



3. 您現在可以使用清單中的任何連線執行下列其中一個動作：

   - 開啟 GitHub/Bitbucket 帳戶或組織 – 選擇連線的名稱。
   - 完成身分驗證交握 – 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身分驗證交握程序。
   - 刪除連線 – 選取連線，然後從動作功能表中選擇刪除。遵循刪除提示的指示。

App Runner API or AWS CLI

您可以使用下列 App Runner API 動作來管理您的連線。

- CreateConnection – 建立與儲存庫提供者帳戶的連線。建立連線後，您必須使用 App Runner 主控台手動完成身分驗證交握。此程序會在上一節中說明。
- ListConnections – 傳回與 相關聯的 App Runner 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 App Runner 自動擴展

AWS App Runner 會自動為您的 App Runner 應用程式擴展或縮減運算資源，特別是執行個體。當流量繁重時，自動擴展可提供足夠的請求處理，並在流量變慢時降低成本。

自動擴展組態

您可以設定幾個參數來調整服務的自動擴展行為。App Runner 會在稱為 AutoScalingConfiguration 的可分割資源中維護自動擴展設定。您可以建立和維護獨立的自動擴展組態，然後再將其指派給 服務。它們與服務建立關聯後，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組態。建立新的自動擴展組態後，您可以將其關聯至 服務，並繼續建立或設定服務的程序。

命名和修訂

自動擴展組態具有名稱和數值修訂。組態的多個修訂具有相同的名稱和不同的修訂編號。您可以針對不同的自動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。每個組態最多可以有 10 個唯一的自動擴展組態名稱和最多 5 個修訂。如果您達到限制且需要建立更多，您可以刪除一個，然後建立另一個。App Runner 不允許您刪除設定為預設或作用中服務正在使用的組態。如需配額的詳細資訊，請參閱 the section called “App Runner 資源配額”。

設定預設組態

當您建立或更新 App Runner 服務時，您可以提供自動擴展組態資源。提供自動擴展組態是選用的。如果您未提供，App Runner 會提供具有建議值的預設自動擴展組態。自動擴展組態功能可讓您選擇設定自己的預設自動擴展組態，而不是使用 App Runner 提供的預設組態。將另一個自動擴展組態指定為預設值後，該組態會自動指派為您未來建立的新服務的預設值。新的預設指定不會影響先前為現有服務設定的關聯。

設定具有自動擴展的服務

您可以跨多個 App Runner 服務共用單一自動擴展組態，以確保服務具有相同的自動擴展行為。如需使用 App Runner 主控台或 App Runner API 設定自動擴展組態的詳細資訊，請參閱本主題接下來的章節。如需可共用資源的一般資訊，請參閱 the section called “組態資源”。

可設定的設定

您可以設定下列自動擴展設定：

- 並行上限 – 執行個體處理的並行請求數目上限。當並行請求數量超過此配額時，App Runner 會擴展服務。
- 大小上限 – 您的服務可擴展的執行個體數量上限。這是可同時處理服務流量的執行個體數量上限。
- 大小下限 – App Runner 可為您的服務佈建的執行個體數目下限。服務一律至少具有此數量的佈建執行個體。其中一些執行個體會主動處理流量。其餘部分是經濟實惠的運算容量預留的一部分，已準備好快速啟動。您支付所有佈建執行個體的記憶體用量。您只需支付作用中子集的 CPU 用量。

> **Note**
>
> vCPU 資源計數會決定 App Runner 可提供給服務的執行個體數量。這是服務中 Fargate 隨需 vCPU 資源計數的可調整配額值 AWS Fargate 。若要檢視您帳戶的 vCPU 配額設定或請

求提高配額，請使用 中的 Service Quotas 主控台 AWS 管理主控台。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的AWS Fargate 服務配額。

管理服務的自動擴展

使用下列其中一種方法來管理 App Runner 服務的自動擴展：

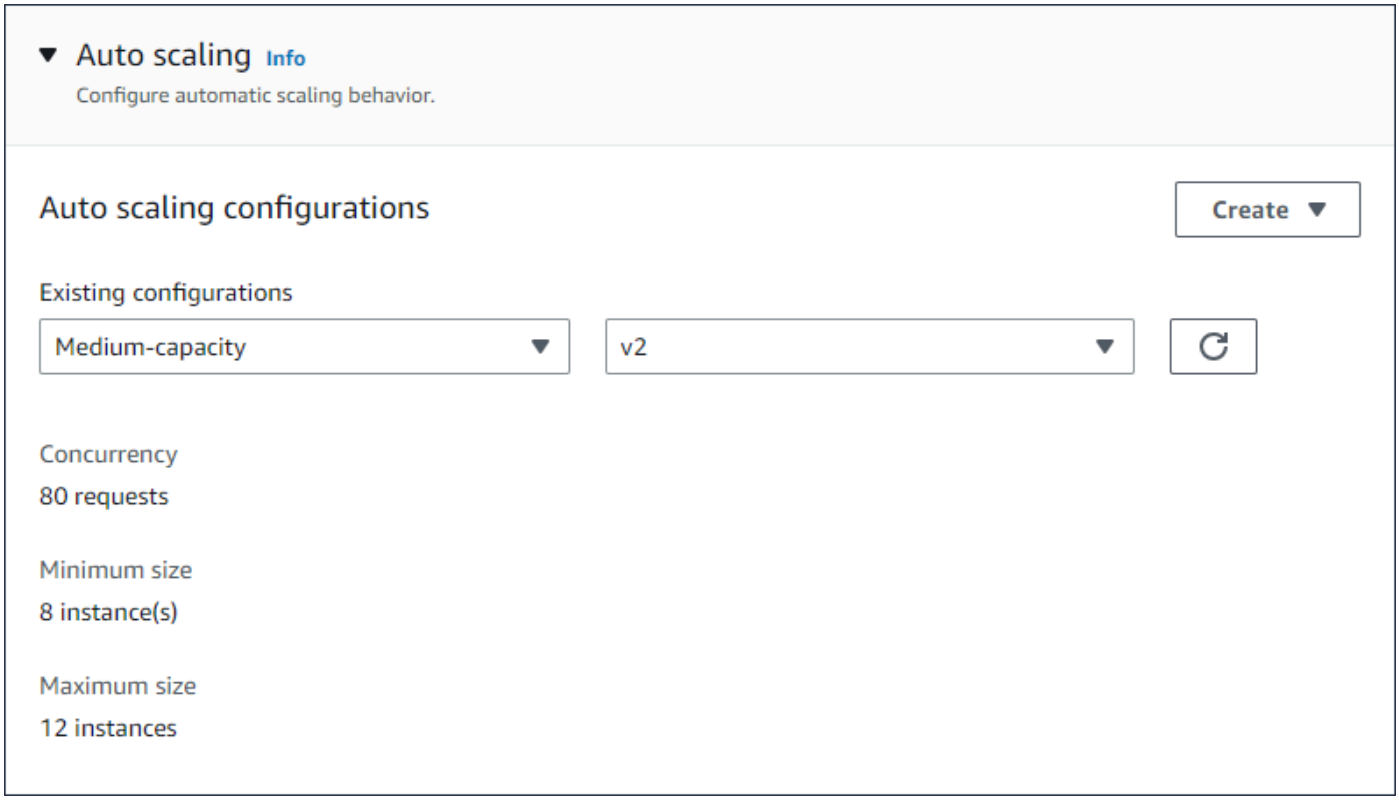
App Runner console

當您使用 App Runner 主控台建立服務或更新服務組態時，您可以指定自動擴展組態。

> **Note**
>
> 當您變更與服務相關聯的自動擴展組態或修訂時，您的服務會重新部署。

Auto Scaling 組態頁面提供多種選項，可為您的服務設定自動擴展。

- 若要指派現有組態和修訂 – 從現有組態下拉式清單中選擇值。在相鄰下拉式清單中，將預設使用最新的修訂版本。如果存在您希望選取的不同修訂，請從修訂下拉式清單中執行此操作。修訂版本顯示的組態值。
- 若要建立和指派新的自動擴展組態 – 從建立功能表中選取建立新 ASC。 這會啟動新增自訂自動擴展組態頁面。輸入自動擴展參數的組態名稱和值。然後選取新增。App Runner 會為您建立新的自動擴展組態資源，並在選取並顯示新組態的情況下，將您返回自動擴展區段。
- 若要建立和指派新的修訂 – 首先從現有組態下拉式清單中選取組態名稱。然後從建立功能表中選取建立 ASC 修訂。 這會啟動新增自訂自動擴展組態頁面。輸入自動擴展參數的值。然後選取新增。App Runner 會為您建立新的自動擴展組態修訂，並在選取並顯示新修訂的情況下，將您返回自動擴展區段。



App Runner API or AWS CLI

當您呼叫 CreateService 或 UpdateService App Runner API 動作時，您可以使用 `AutoScalingConfigurationArn` 參數來指定服務的自動擴展組態資源。

下一節提供管理自動擴展組態資源的指引。

管理自動擴展組態資源

使用下列其中一種方法來管理帳戶的 App Runner 自動擴展組態和修訂：

App Runner console

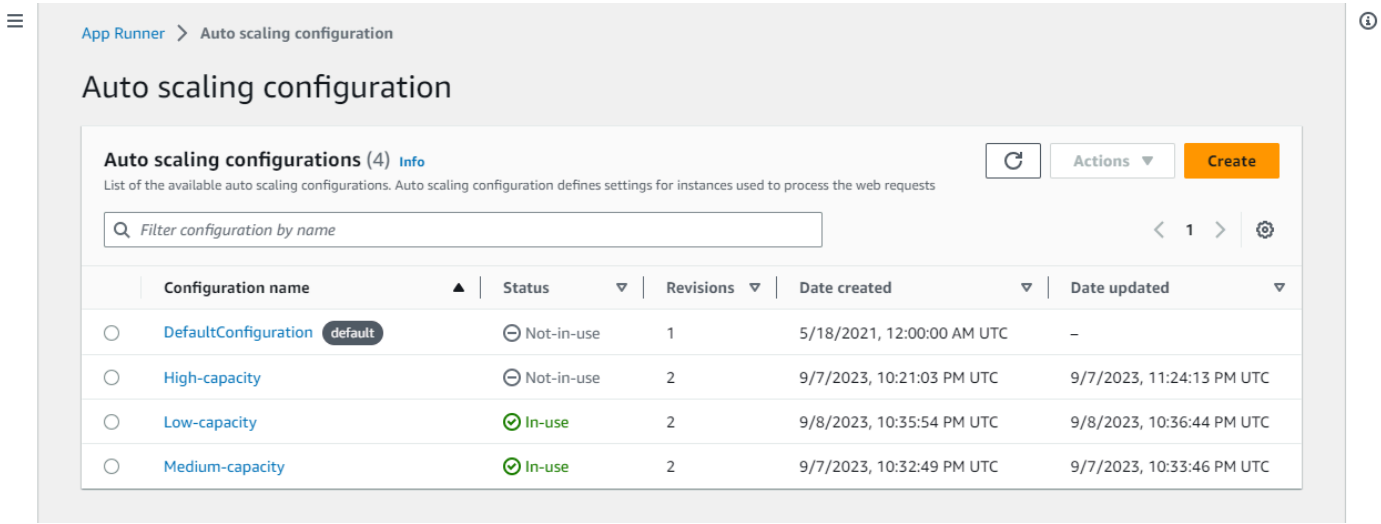
管理自動擴展組態

自動調整規模組態頁面列出您在帳戶中設定的自動調整規模組態。您可以在此頁面建立和管理自動擴展組態，稍後再將其指派給一或多個 App Runner 服務。

您可以從此頁面執行下列任何操作：

- 建立新的自動擴展組態。
- 為現有的自動擴展組態建立新的修訂。

- 刪除自動擴展組態。
- 將自動擴展組態設定為預設值。



管理您帳戶中的自動擴展組態

1. 開啟 App Runner 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，選擇自動調整規模組態。主控台會顯示您帳戶中的自動擴展組態清單。

   您現在可以執行下列任何動作。

   - 若要建立新的自動擴展組態，請遵循下列步驟。

     a. 在自動擴展組態頁面上，選取建立。

     隨即顯示建立自動擴展組態頁面。

     b. 輸入組態名稱、並行、大小下限和大小上限的值。

     c. （選用）如果您想要新增標籤，請選取自動新標籤。然後在出現的欄位上輸入名稱和值（選用）。

     d. 選取建立。

   - 若要為現有的自動擴展組態建立新的修訂，請遵循下列步驟。

     a. 在自動調整規模組態頁面上，選取需要新修訂之組態旁的選項按鈕。然後從動作功能表中選取建立修訂。

     隨即顯示建立修訂頁面。

     b. 在 上，輸入並行、大小下限和大小上限的值。

c. （選用）如果您想要新增標籤，請選取自動新標籤。然後在出現的欄位上輸入名稱和值（選用）。

d. 選取建立。

- 若要刪除自動擴展組態，請遵循下列步驟。

  a. 在自動擴展組態頁面上，選取您要刪除之組態旁的選項按鈕。

  b. 從動作功能表中選取刪除。

  c. 若要繼續刪除，請在確認對話上選取刪除。否則，請選取取消。

  > **Note**
  >
  > App Runner 會驗證您的刪除選擇未設定為預設，或目前正由任何作用中的服務使用中。

- 若要將自動擴展組態設定為預設值，請依照下列步驟進行。

  a. 在自動調整規模組態頁面上，選取您需要設定為預設值之組態旁的選項按鈕。

  b. 從動作功能表中選取設為預設。

  c. 此時會顯示一個對話方塊，通知您 App Runner 將使用最新的修訂版做為您建立的所有新服務的預設組態。選取確認以繼續。否則，請選取取消。

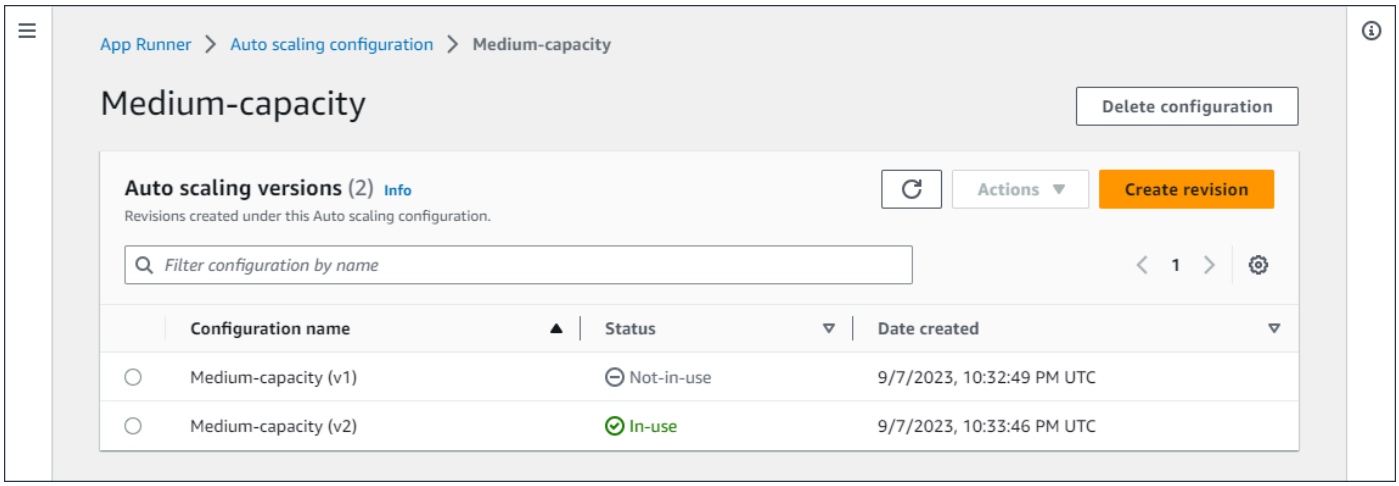
  > **Note**
  >
  > - 當您將自動擴展組態設定為預設時，它會自動指派為您未來建立的新服務的預設組態。
  > - 新的預設指定不會影響先前為現有服務設定的關聯。
  > - 如果指定的預設自動擴展組態有修訂版，App Runner 會將其最新修訂版指派為預設值。

管理修訂

主控台也有一個頁面，用於建立和管理您現有的自動擴展修訂，稱為自動擴展修訂。在自動擴展組態頁面上選取組態的名稱來存取此頁面。

您可以從 Auto Scaling 修訂頁面執行下列任何操作：

- 建立新的自動擴展修訂。
- 將自動擴展組態修訂設定為預設值。
- 刪除修訂。
- 刪除整個自動擴展組態，包括所有相關聯的修訂。
- 檢視修訂的組態詳細資訊。
- 檢視與修訂相關聯的服務清單。
- 變更所列服務的修訂。



管理您帳戶中的自動擴展修訂

1. 開啟 App Runner 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，選擇自動調整規模組態。主控台會顯示您帳戶中的自動擴展組態清單。??? 區段中的先前一組程序包含此頁面的螢幕影像。
3. 現在，您可以深入了解特定的自動擴展組態，以檢視和管理其所有修訂。在自動調整規模組態窗格的組態名稱欄下，選擇自動調整規模組態名稱。選取實際名稱，而非選項按鈕。這會導覽至 Auto Scaling 修訂頁面上該組態的所有修訂清單。
4. 您現在可以執行下列任何動作。
   - 若要為現有的自動擴展組態建立新的修訂，請遵循下列步驟。
     a. 在自動擴展修訂頁面上，選取建立修訂。

        隨即顯示建立修訂頁面。

     b. 輸入並行、大小下限和大小上限的值。

c. （選用）如果您想要新增標籤，請選取自動新標籤。然後在出現的欄位上輸入名稱和值（選用）。

d. 選取建立。

- 若要刪除整個自動擴展組態，包括所有相關聯的修訂，請遵循下列步驟。

  a. 選取頁面右上角的刪除組態。

  b. 若要繼續刪除，請在確認對話上選取刪除。否則，請選取取消。

  > **Note**
  >
  > App Runner 會驗證您的刪除選擇未設定為預設，或目前正由任何作用中的服務使用中。

- 若要將自動擴展修訂設定為預設，請遵循下列步驟。

  a. 選取您需要設定為預設值的修訂旁的選項按鈕。

  b. 從動作功能表中選取設為預設。

  > **Note**
  >
  > - 當您將自動擴展組態設定為預設時，它會自動指派為您未來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。

- 若要檢視修訂的組態詳細資訊，請遵循下列步驟。

  - 選取修訂版旁的選項按鈕。

    修訂版的組態詳細資訊，包括 ARN，會顯示在下方分割面板中。請參閱此程序結尾的螢幕影像。

- 若要檢視與修訂相關聯的服務清單，請遵循下列步驟。

  - 選取修訂版旁的選項按鈕。

    服務面板會顯示在下方分割面板的修訂組態詳細資訊下方。面板會列出使用此自動擴展組態修訂的所有 服務。請參閱此程序結尾的螢幕影像。

- 若要變更所列服務的修訂，請遵循下列步驟。

a. 如果您尚未這麼做，請選取修訂旁的選項按鈕。

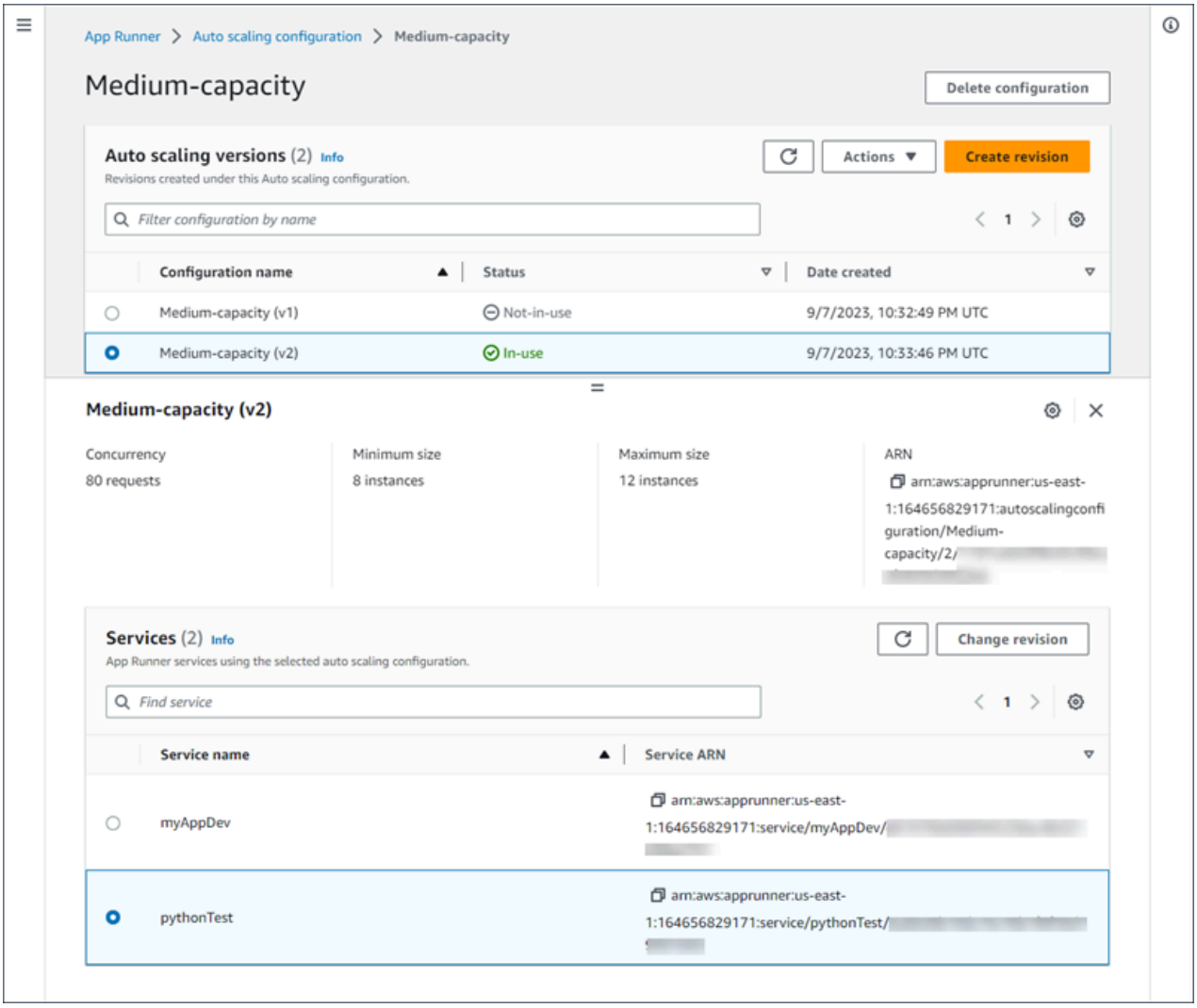
   服務面板會顯示在下方分割面板的修訂組態詳細資訊下方。面板會列出使用此自動擴展組態修訂的所有 服務。請參閱此程序結尾的螢幕影像。

b. 在服務面板上，選取您要修改之服務旁的選項按鈕。然後選取變更修訂。

c. 隨即顯示變更 ASC 修訂面板。從下拉式清單中的可用修訂中進行選擇。只有您先前選擇的自動擴展組態修訂可用。如果您需要變更為不同的自動擴展組態，請遵循上一節中的程序the section called “管理服務的自動擴展”。

   選取更新以繼續變更。否則，請選取取消。

> **Note**
>
> 當您變更與服務相關聯的修訂版時，系統會重新部署您的服務。
> 您必須在此面板上選取重新整理，以查看更新的關聯。
> 若要查看進行中的活動和服務重新部署的狀態，請使用面板導覽列導覽至 App Runner > Services，選取服務，然後從服務概觀面板檢視日誌索引標籤。



App Runner API or AWS CLI

使用下列 App Runner API 動作來管理您的自動擴展組態資源。

- CreateAutoScalingConfiguration – 建立新的自動擴展組態或現有組態的修訂。
- UpdateDefaultAutoScalingConfiguration – 將自動擴展組態設定為預設值。現有的預設自動擴展組態將自動設定為非預設。
- ListAutoScalingConfigurations – 傳回與 相關聯的自動擴展組態清單 AWS 帳戶，其中包含摘要資訊。
- ListServicesForAutoScalingConfiguration – 使用自動擴展組態傳回相關聯 App Runner 服務的清單。

- DescribeAutoScalingConfiguration – 傳回自動擴展組態的完整描述。
- DeleteAutoScalingConfiguration – 刪除自動擴展組態。您可以刪除頂層自動擴展組態、特定修訂為 1，或與頂層組態相關聯的所有修訂。使用選用`DeleteAllRevisions`參數來刪除所有修訂。如果您達到 的自動擴展組態資源配額 AWS 帳戶，您可能需要刪除不必要的自動擴展組態。

管理 App Runner 服務的自訂網域名稱

當您建立 AWS App Runner 服務時，App Runner 會為其配置網域名稱。這是 App Runner 所擁有`awsapprunner.com`網域中的子網域。您可以使用網域名稱來存取服務中執行的 Web 應用程式。

> **Note**
>
> 為了增強 App Runner 應用程式的安全性，在公有尾碼清單 (PSL) 中註冊 *.awsapprunner.com 網域。為了進一步提高安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用字`__Host-`首為 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。

如果您擁有網域名稱，您可以將其與 App Runner 服務建立關聯。在 App Runner 驗證您的新網域之後，除了 App Runner 網域之外，您還可以使用您的網域來存取您的應用程式。您最多可以關聯五個自訂網域。

> **Note**
>
> 您可以選擇性地包含網域的`www`子網域。不過，這目前僅由 API 支援。App Runner 主控台不支援包含網域的`www`子網域。

> **Note**
>
> AWS App Runner 不支援使用 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 文件中的使用私有託管區域。

將自訂網域與您的服務建立關聯（連結）

當您將自訂網域與服務建立關聯時，您必須將 CNAME 記錄和 DNS 目標記錄新增至您的 DNS 伺服器。下列各節提供 CNAME 記錄和 DNS 目標記錄的資訊，以及如何使用這些記錄。

> **Note**
>
> 如果您使用 Amazon Route 53 做為 DNS 供應商，App Runner 會使用必要的憑證驗證和 DNS 記錄自動設定您的自訂網域，以連結至您的 App Runner 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。如此一來，App Runner 就可以驗證您擁有或控制網域。

> **Note**
>
> 若要自動續約自訂網域憑證，請確定您不會從 DNS 伺服器刪除憑證驗證記錄。如需如何解決憑證續約相關問題的詳細資訊，請參閱 the section called “自訂網域憑證續約”。

App Runner 使用 ACM 來驗證網域。如果您在 DNS 記錄中使用 CAA 記錄，請確定至少有一個 CAA 記錄參考 `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，請參閱以下連結以了解如何解決這些問題：

- 憑證授權機構授權 (CAA) 問題
- 如何解決發行或續約 ACM 憑證的 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 會使用必要的憑證驗證和 DNS 記錄自動設定您的自訂網域，以連結至您的 App Runner 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更多資訊。

DNS 目標記錄

將 DNS 目標記錄新增至您的 DNS 伺服器，以將 App Runner 網域設為目標。如果您選擇此選項，請為自訂網域新增一筆記錄，並為www子網域新增一筆記錄。然後，在 App Runner 主控台中等待自訂網域狀態變為作用中。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）。驗證您的自訂網域時，App Runner 會開始將此網域的流量路由到您的 Web 應用程式。

> **Note**
>
> 為了提高與 App Runner 服務的相容性，我們建議您使用 Amazon Route 53 做為 DNS 供應商。如果您未使用 Amazon Route 53 來管理公有 DNS 記錄，請聯絡您的 DNS 供應商以了解如何新增記錄。
>
> 如果您使用 Amazon Route 53 做為 DNS 供應商，則可以為子網域新增 CNAME 或別名記錄。對於根網域，請確定您使用別名記錄。

您可以從 Amazon Route 53 或其他供應商購買網域名稱。若要使用 Amazon Route 53 購買網域名稱，請參閱《Amazon Route 53 開發人員指南》中的註冊新的網域。

如需如何在 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。

如需如何在其他註冊商上設定 DNS 目標的指示，例如 GoDaddy、Shopify、Hover 等，請參閱其新增 DNS 目標記錄的特定文件。

指定要與您的 App Runner 服務建立關聯的網域

您可以透過下列方式指定要與 App Runner 服務建立關聯的網域：

- 根網域 – DNS 有一些固有限制，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 的流量路由`acme.example.com`到您的 App Runner 服務。不過，您無法建立將 流量路由`example.com`至 App Runner 服務的 CNAME 記錄。若要建立根網域，請確定您新增別名記錄。

  別名記錄是 Route 53 特有的，相較於 CNAME 記錄，具有下列優點：

  - Route 53 為您提供更多彈性，因為可以為根網域或子網域建立別名記錄。例如，如果您的網域名稱是 `example.com`，您可以建立記錄，將 `example.com`或 的請求路由`acme.example.com`到您的 App Runner 服務。
  - 它更具成本效益。這是因為 Route 53 不會針對使用別名記錄來路由流量的請求收費。
- 子網域 – 例如 `login.example.com`或 `admin.login.example.com`。您也可以選擇性地將`www`子網域關聯為相同操作的一部分。您可以為子網域新增 CNAME 或別名記錄。
- 萬用字元 – 例如 `*.example.com`。在此情況下，您無法使用 `www`選項。您只能將萬用字元指定為根網域的直接子網域，而且只能自行指定。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立即的子網域，而不會關聯根網域本身。根網域必須在個別操作中建立關聯。

更具體的網域關聯會覆寫較不具體的網域關聯。例如， 會`login.example.com`覆寫 `*.example.com`。會使用更特定關聯的憑證和 CNAME。

下列範例示範如何使用多個自訂網域關聯：

1. `example.com` 與您服務的首頁建立關聯。啟用 `www`以關聯 `www.example.com`。
2. `login.example.com` 與服務的登入頁面建立關聯。
3. `*.example.com` 與自訂「找不到」頁面建立關聯。

取消關聯（取消連結）自訂網域

您可以將自訂網域與 App Runner 服務取消關聯（取消連結）。當您取消連結網域時，App Runner 會停止將此網域的流量路由到您的 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。

App Runner 會在內部建立追蹤網域有效性的憑證。這些憑證存放在 AWS Certificate Manager (ACM) 中。App Runner 會在網域與您的服務取消關聯後或刪除服務後 7 天內刪除這些憑證。

管理自訂網域

使用下列其中一種方法來管理 App Runner 服務的自訂網域：

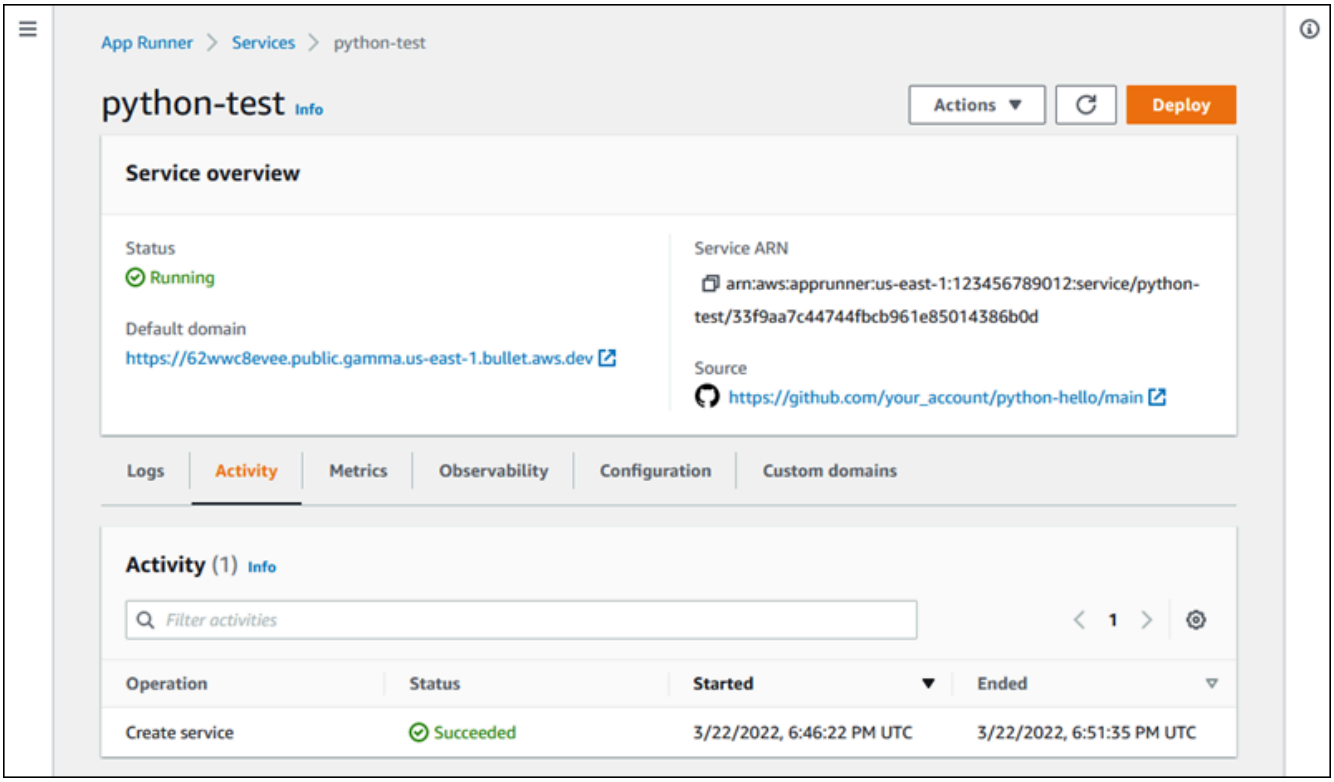
> **Note**
>
> 為了提高與 App Runner 服務的相容性，我們建議您使用 Amazon Route 53 做為 DNS 供應商。如果您未使用 Amazon Route 53 來管理公有 DNS 記錄，請聯絡您的 DNS 供應商以了解如何新增記錄。
>
> 如果您使用 Amazon Route 53 做為 DNS 供應商，您可以新增子網域的 CNAME 或別名記錄。對於根網域，請確定您使用別名記錄。

App Runner console

使用 App Runner 主控台關聯（連結）自訂網域

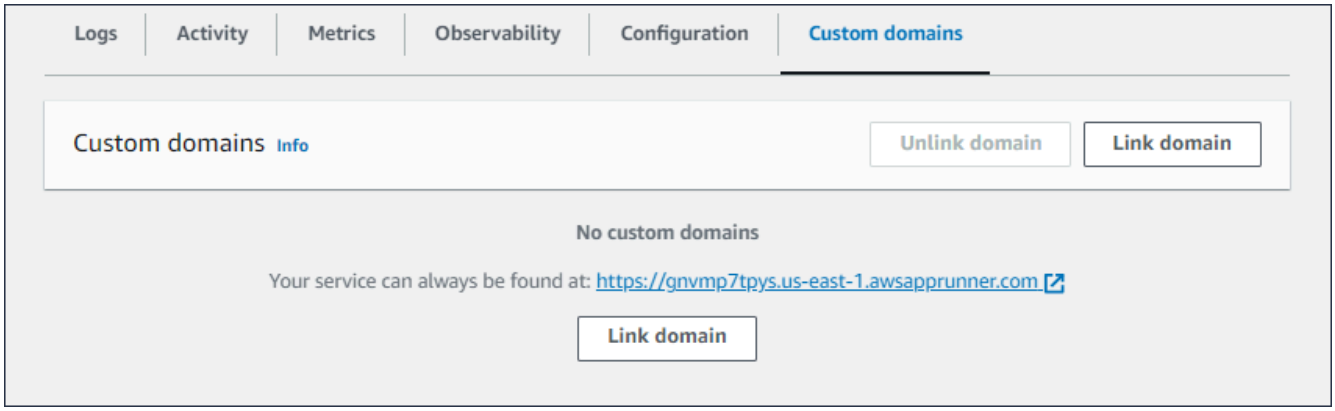
1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇服務，然後選擇您的 App Runner 服務。

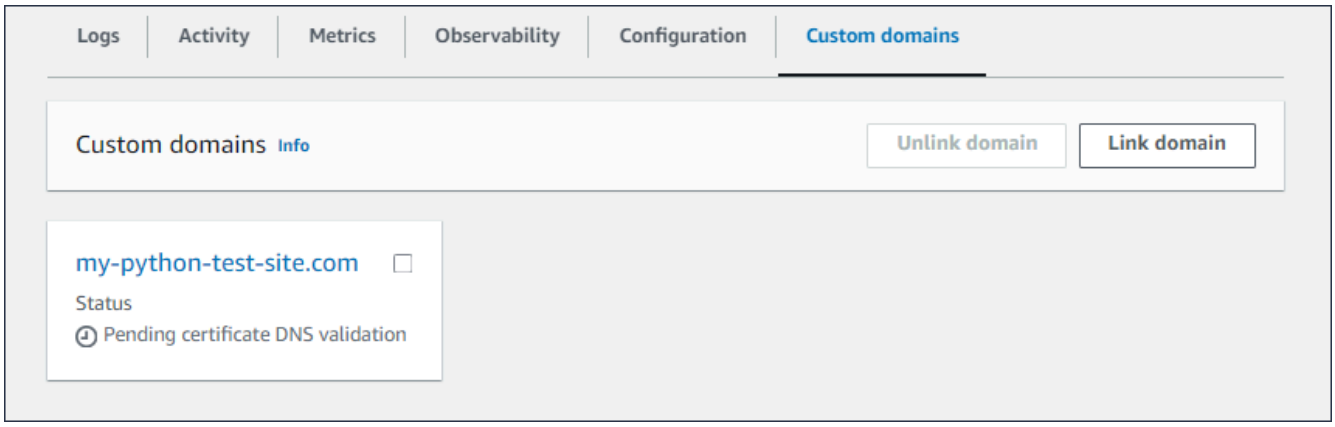
   主控台會顯示服務儀表板，其中包含服務概觀。



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，或沒有自訂網域。

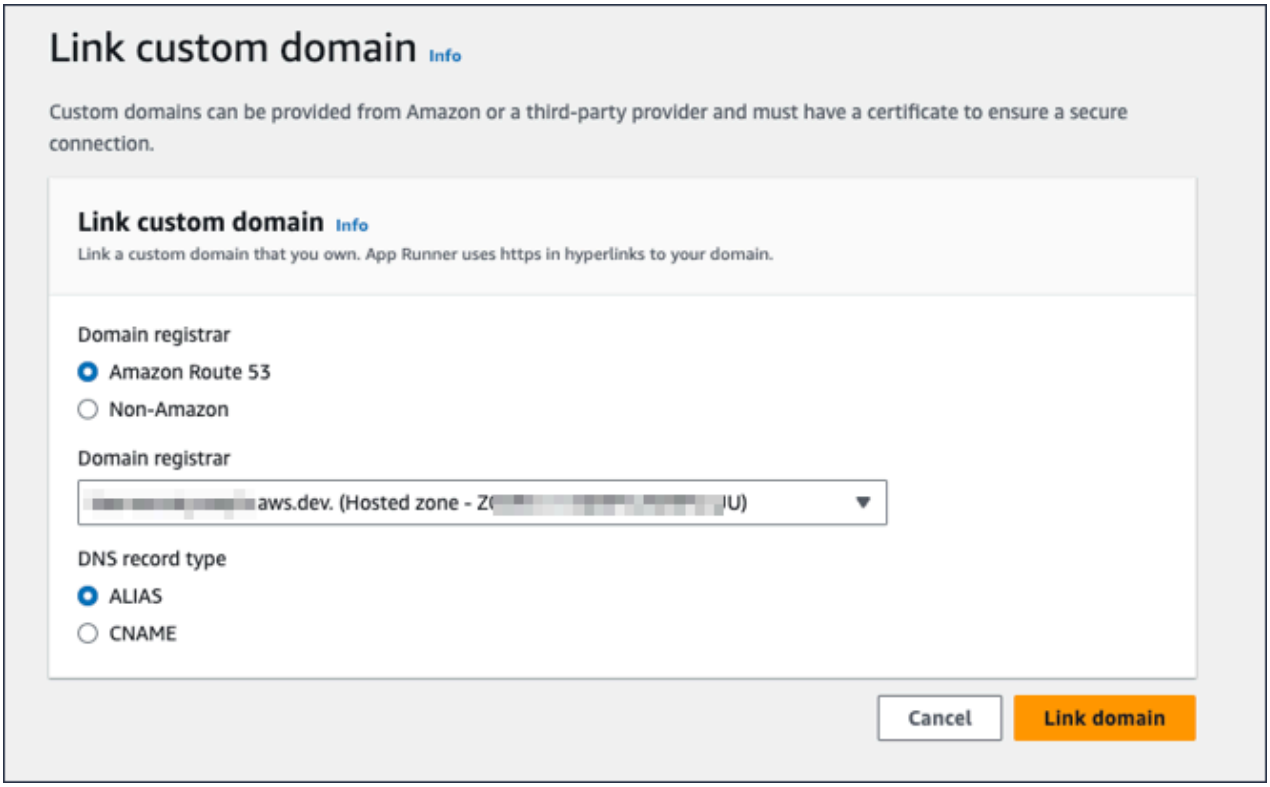




4. 在自訂網域索引標籤上，選擇連結網域。
5. 隨即顯示連結自訂網域頁面。
   - 如果您的自訂網域已向 Amazon Route 53 註冊，請選取網域註冊商的 Amazon Route 53。
     a. 從下拉式清單中選取網域名稱。此清單會顯示 Route 53 網域名稱的名稱和託管區域 ID。

     > **Note**
     > 您必須先從用來管理其他 App Runner 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。

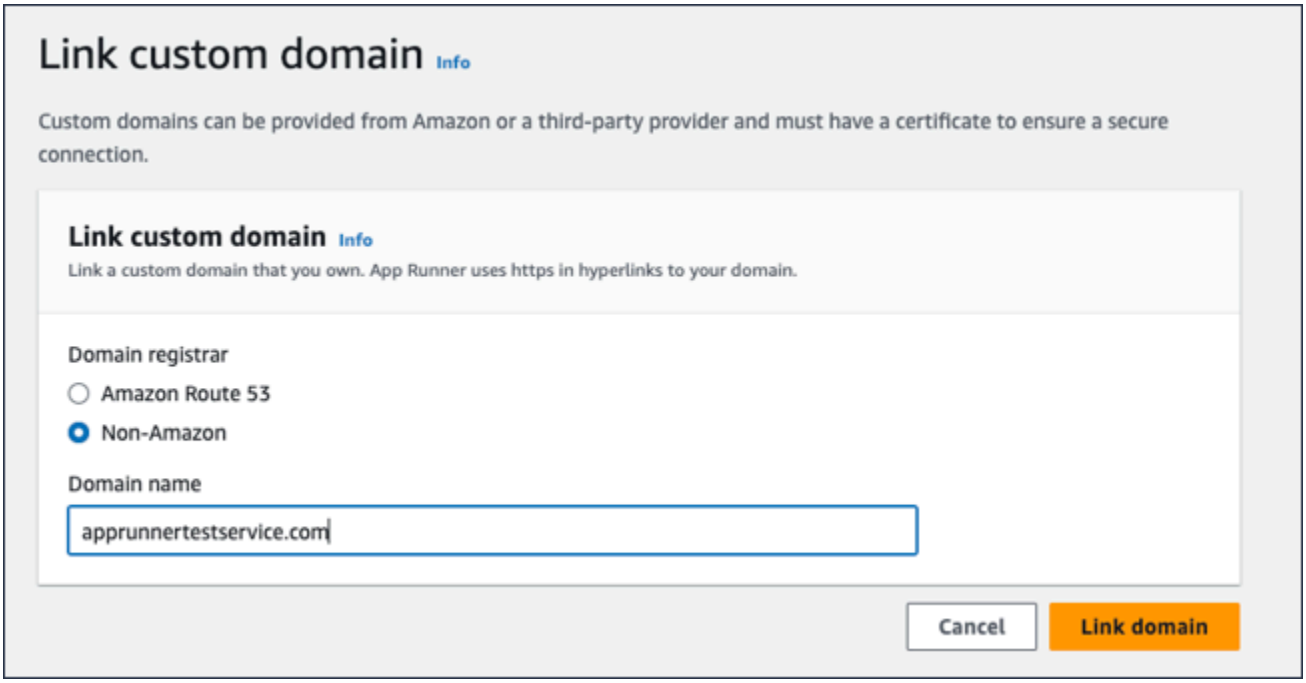


> **Note**
>
> 如果 App Runner 顯示錯誤訊息，指出自動組態嘗試失敗，您可以手動設定 DNS 記錄以繼續。如果先前從服務取消連結相同的網域名稱，而 DNS 提供者記錄指向之後要刪除的服務，則可能會發生此問題。在此情況下，App Runner 會遭到封鎖，無法自動覆寫這些記錄。若要完成 DNS 組態，請略過此程序中的其餘步驟，然後遵循 中的指示設定 Amazon Route 53 別名記錄。

- 如果您的自訂網域已向另一個網域註冊商註冊，請選取非 Amazon for Domain 註冊商。

  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 供應商，則此步驟為選用。

     此時，App Runner 已使用所需的憑證驗證和 DNS 記錄自動設定 Route 53 網域。

     > **Note**
     >
     > 如果此相同的網域名稱先前已從服務取消連結，而沒有 DNS 提供者記錄指向之後要刪除的服務，App Runner 嘗試的自動組態可能會失敗。若要解決此問題並完成 DNS 關聯，請繼續設定 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 供應商。

   - 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，App Runner 可以驗證您擁有或控制網域。

     > **Note**
     >
     > 若要自動續約自訂網域憑證，請務必不要從 DNS 伺服器刪除憑證驗證記錄。

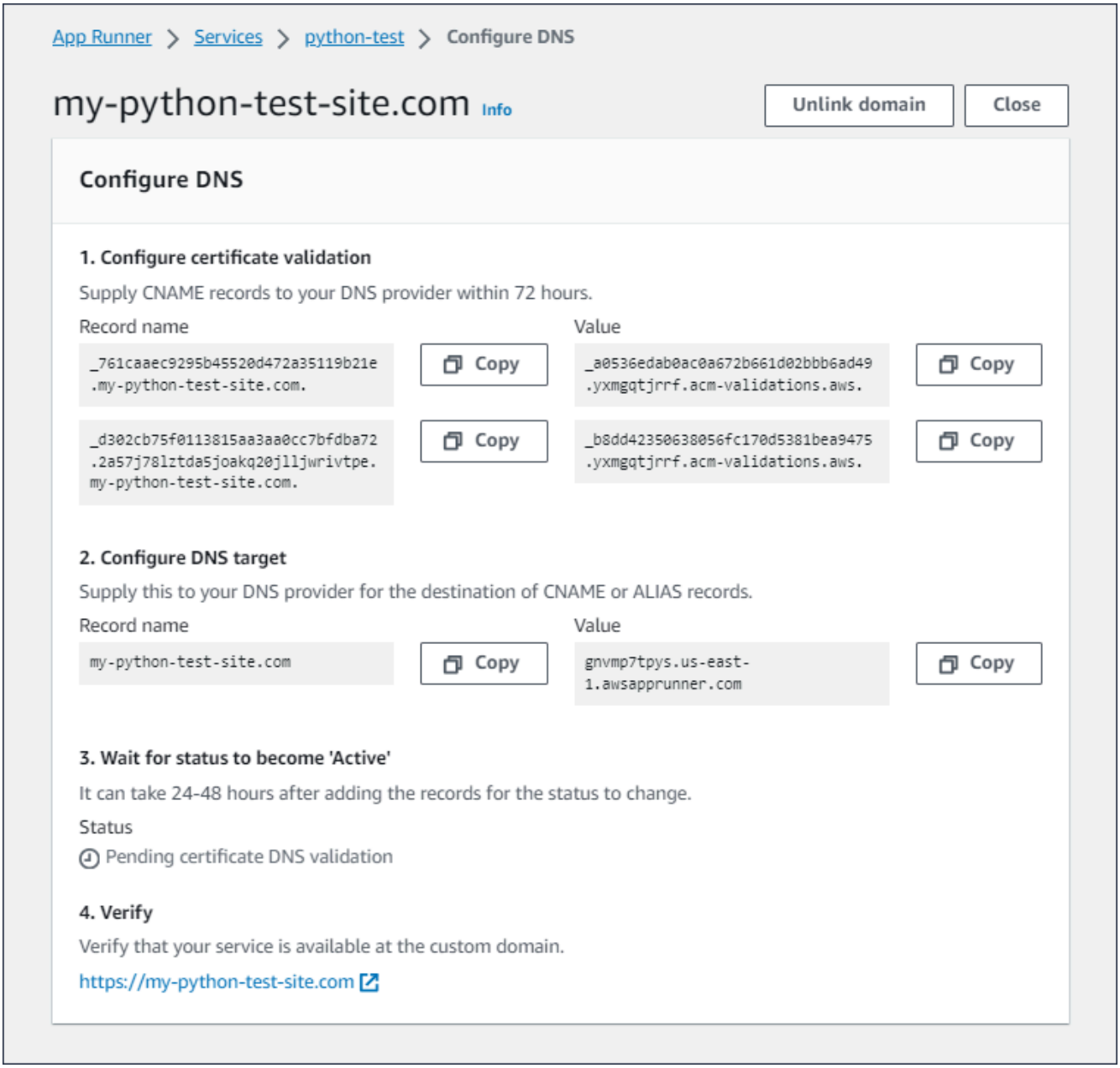
     - 如需設定憑證驗證的詳細資訊，請參閱AWS Certificate Manager 《使用者指南》中的 DNS 驗證。

- 如需如何使用 Amazon Route 53 別名記錄設定 DNS 目標的詳細資訊，請參閱 the section called “設定 Amazon Route 53 別名記錄”。

- 如果您使用的是 Amazon Route 53 以外的 DNS 供應商，請遵循下列步驟。
  - 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，App Runner 可以驗證您擁有或控制網域。

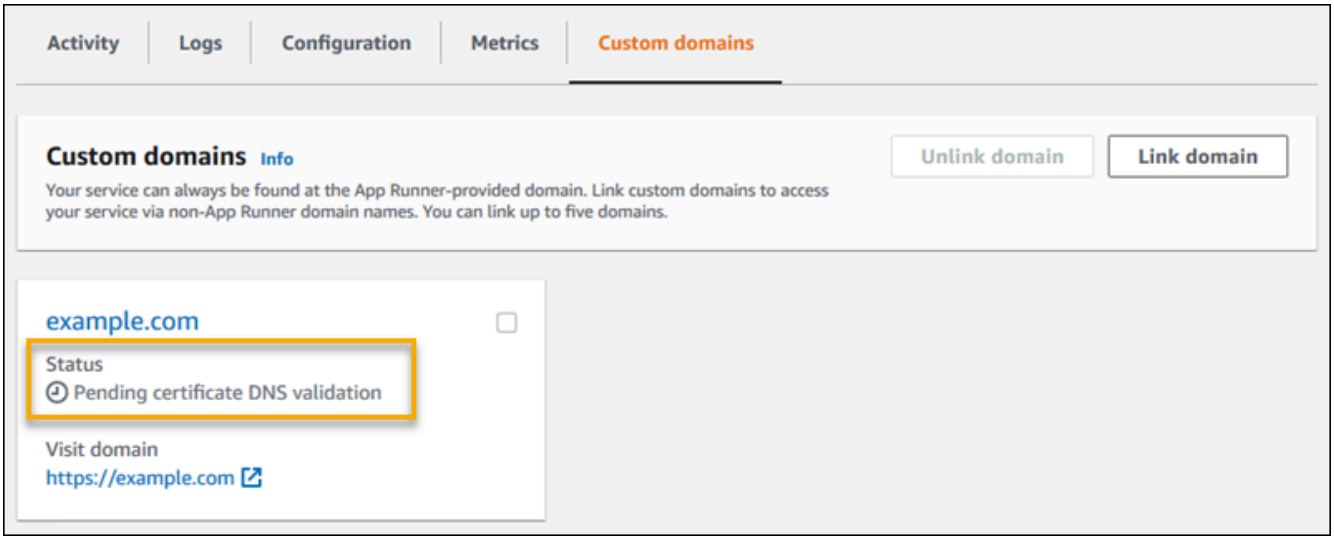
    > **Note**
    >
    > 若要自動續約自訂網域憑證，請務必不要從 DNS 伺服器刪除憑證驗證記錄。

    - 如需設定憑證驗證的詳細資訊，請參閱AWS Certificate Manager 《 使用者指南》中的 DNS 驗證。
    - 如需如何在其他登錄器上設定 DNS 目標的指示，例如 GoDaddy、Shopify、Hover 等，請參閱其新增 DNS 目標的特定文件。

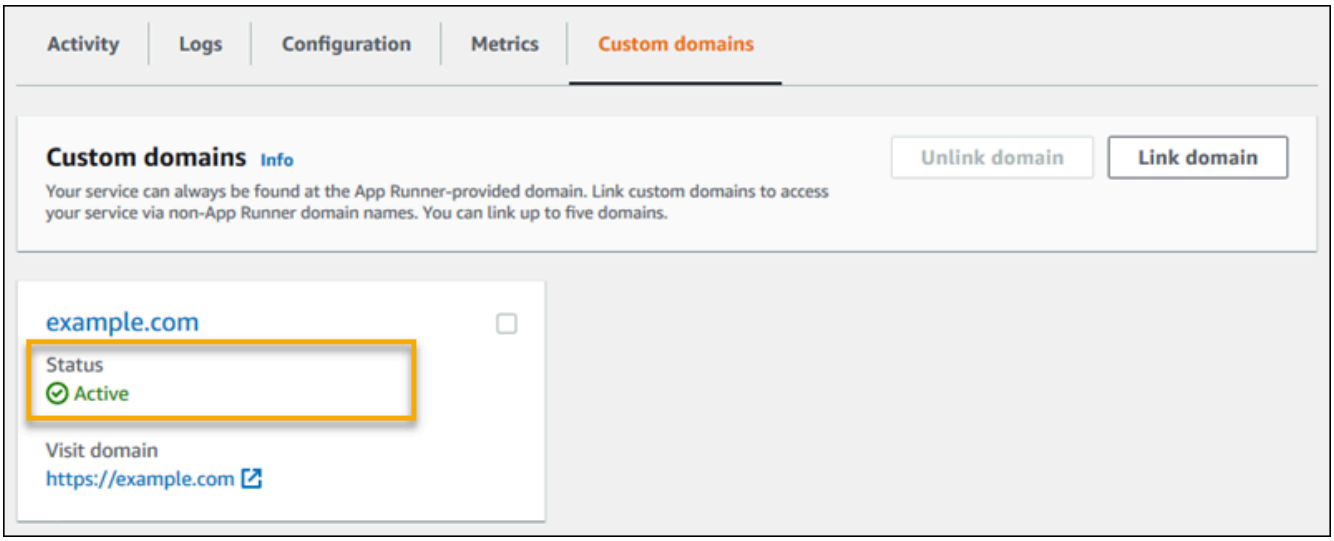


7. 選擇關閉

主控台會再次顯示儀表板。自訂網域索引標籤有新的圖磚，顯示您剛在待定憑證 DNS 驗證狀態中連結的網域。



8. 當網域狀態變更為作用中時，請瀏覽網域以驗證該網域是否適用於路由流量。



> **Note**
>
> 如需如何對與自訂網域相關的錯誤進行故障診斷的說明，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圖磚，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驗證動作。

> **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 App Runner API 或 將自訂網域與服務建立關聯 AWS CLI，請呼叫 AssociateCustomDomain API 動作。當呼叫成功時，會傳回描述與服務相關聯之自訂網域的 CustomDomain 物件。物件會顯示`CREATING`狀態，並包含 CertificateValidationRecord 物件的清單。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 App Runner API 或 取消自訂網域與服務的關聯 AWS CLI，請呼叫 DisassociateCustomDomain API 動作。當呼叫成功時，會傳回 CustomDomain 物件，描述要與您的服務取消關聯的自訂網域。物件會顯示`DELETING`狀態。

主題

- 設定目標 DNS 的 Amazon Route 53 別名記錄

設定目標 DNS 的 Amazon Route 53 別名記錄

> **Note**
>
> 如果 Amazon Route 53 是您的 DNS 供應商，則不需要遵循此程序。在此情況下，App Runner 會使用必要的憑證驗證和 DNS 記錄自動設定 Route 53 網域，以連結至您的 App Runner Web 應用程式。
>
> 如果 App Runner 的自動組態嘗試失敗，請依照此程序完成 DNS 組態。如果先前已從服務取消連結相同的網域名稱，而沒有指向服務之後刪除的 DNS 提供者記錄，App Runner 會遭到封鎖，無法自動覆寫這些記錄。此程序說明如何手動將其複製到 Route 53 DNS。

您可以使用 Amazon Route 53 做為 DNS 供應商，將流量路由到您的 App Runner 服務。它是一種高可用性和可擴展的網域名稱系統 (DNS) Web 服務。Amazon Route 53 記錄包含控制流量路由到 App Runner 服務的方式的設定。您可以建立 CNAME 記錄或 ALIAS 記錄。如需 CNAME 與別名記錄的比較，請參閱《Amazon Route 53 開發人員指南》中的在別名與非別名記錄之間進行選擇。

> **Note**
>
> Amazon Route 53 目前支援 2022 年 8 月 1 日之後建立之服務的別名記錄。

Amazon Route 53 console

設定 Amazon Route 53 別名記錄

1. 登入 AWS 管理主控台 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至 App Runner 服務的託管區域名稱。
4. 選擇建立記錄。
5. 指定下列值：
   - 路由政策：選擇適用的路由政策。如需詳細資訊，請參閱選擇路由政策。
   - 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`而您想要使用 `acme.example.com` 將流量路由到您的環境，請輸入 `acme`。
   - 值/路由流量至：選擇 App Runner 應用程式的別名，然後選擇端點的來源區域。選擇您要路由流量之應用程式的網域名稱。
   - 記錄類型：接受預設值 A – IPv4 地址。
   - 評估目標運作狀態：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 秒內在所有 Route 53 伺服器上傳播。當您的別名記錄傳播 Route 53 伺服器時，您可以使用您建立的別名記錄名稱，將流量路由到您的 App Runner 服務。

如需有關如何疑難排解 DNS 變更是否太長而無法傳播的資訊，請參閱為什麼我的 DNS 變更在 Route 53 和公有解析程式中傳播需要這麼長的時間？。

Amazon Route 53 API or AWS CLI

使用 Amazon Route 53 API 設定 Amazon Route 53 別名記錄，或 AWS CLI 呼叫 ChangeResourceRecordSets API 動作。若要了解 Route 53 的目標託管區域 ID，請參閱服務端點。

暫停和繼續 App Runner 服務

如果您需要暫時停用 Web 應用程式並停止程式碼執行，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 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。相反地，App Runner 會繼續使用您目前部署的版本。您的應用程式會保留其 App Runner 網域。

> **Important**
>
> - 當您暫停服務時，您的應用程式會失去其狀態。例如，您程式碼使用的任何暫時性儲存體都會遺失。針對您的程式碼，暫停和繼續您的服務相當於部署到新服務。
> - 如果您因為程式碼中的瑕疵（例如，發現的錯誤或安全問題）而暫停服務，則無法在繼續服務之前部署新版本。
>
> 因此，我們建議您讓服務保持執行，並改為回復到上次穩定的應用程式版本。
> - 當您繼續服務時，App Runner 會部署在暫停服務之前所使用的最後一個應用程式版本。如果您在暫停服務之後新增任何新的來源版本，即使選取自動部署，App Runner 也不會自動部署這些版本。例如，假設您在映像儲存庫中有新的映像版本，或在程式碼儲存庫中有新的遞交。這些版本不會自動部署 。
>
> 若要部署較新版本，請在繼續 App Runner 服務之後，執行手動部署或將另一個版本新增至來源儲存庫。

暫停和刪除比較

暫停您的 App Runner 服務以暫時停用它。只有運算資源會終止，而您的儲存資料（例如，具有應用程式版本的容器映像）會保持不變。繼續您的服務非常快速，您的應用程式已準備好部署到新的運算資源。您的 App Runner 網域保持不變。

刪除您的 App Runner 服務以永久移除它。您的已儲存資料會遭到刪除。如果您需要重新建立服務，App Runner 需要再次擷取您的來源，如果它是程式碼儲存庫，也需要建置它。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

當您暫停服務且其處於暫停狀態時，它會對動作請求做出不同的回應，包括 API 呼叫或主控台操作。當服務暫停時，您仍然可以執行 App Runner 動作，這些動作不會以影響服務執行時間的方式修改服務的定義或組態。換句話說，如果動作變更執行中服務的行為、擴展或其他特性，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。類似的允許或拒絕對等主控台操作。

您可以在暫停的服務上執行的動作

- *List** 和 *Describe**動作 – 僅讀取資訊的動作。
- *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，不會影響其執行時間行為。

您無法在暫停的服務上執行的動作

- *StartDeployment* 動作 （或使用主控台手動部署)
- *UpdateService* （或使用 主控台的組態變更，標記變更除外）
- *CreateCustomDomainAssociations*, *DeleteCustomDomainAssociations*
- *CreateConnection*, *DeleteConnection*

暫停並繼續您的服務

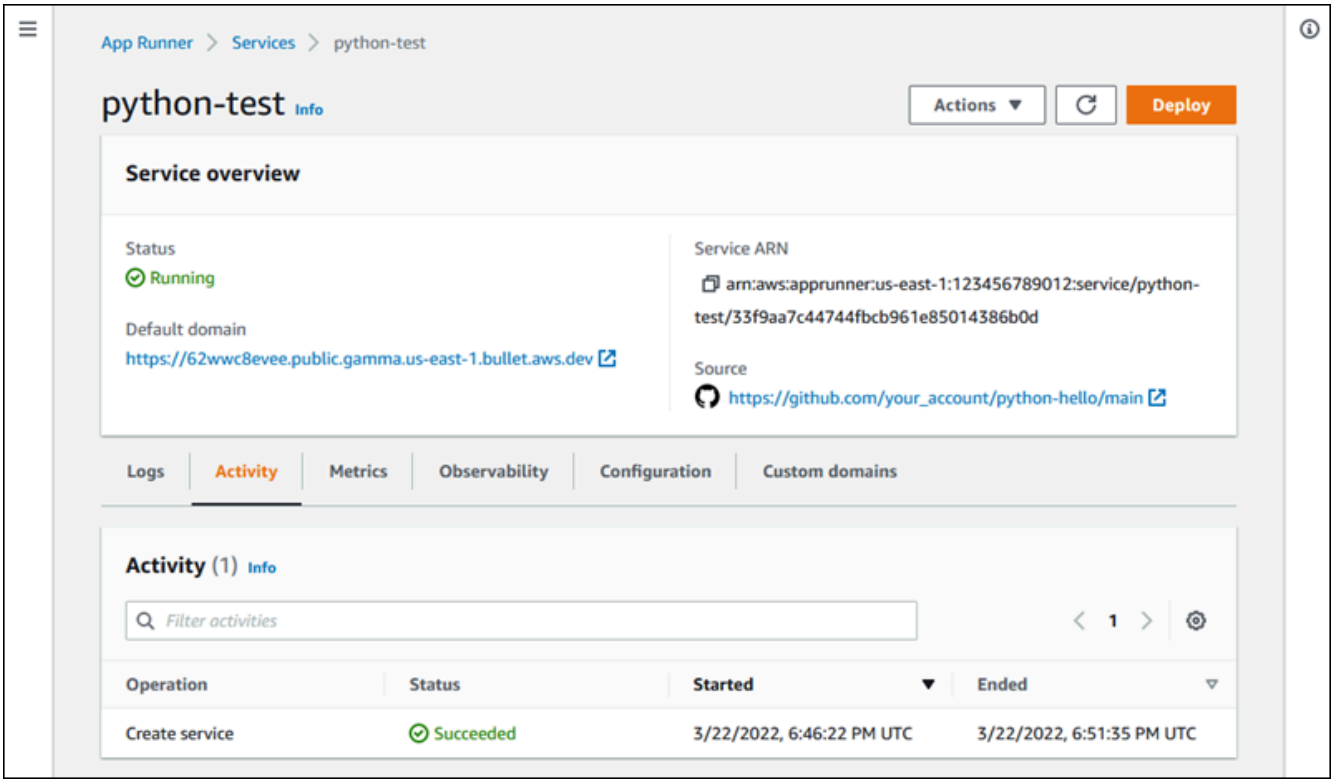
使用下列其中一種方法暫停和繼續您的 App Runner 服務：

App Runner console

使用 App Runner 主控台暫停您的服務

1. 開啟 App Runner 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，然後選擇您的 App Runner 服務。

   主控台會顯示具有服務概觀的服務儀表板。



3. 選擇動作，然後選擇暫停。

   在服務儀表板頁面上，服務狀態會變更為進行中的操作，然後變更為已暫停。您的服務現在已暫停。

使用 App Runner 主控台繼續您的服務

1. 選擇動作，然後選擇繼續。

   在服務儀表板頁面上，服務狀態會變更為進行中的操作。

2. 等待服務繼續。在服務儀表板頁面上，服務狀態會變更回執行中。
3. 若要驗證繼續服務是否成功，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 應用程式是否正確執行。

App Runner API or AWS CLI

若要使用 App Runner API 或 暫停服務 AWS CLI，請呼叫 PauseService API 動作。如果呼叫傳回成功回應，且服務物件顯示 `"Status": "OPERATION_IN_PROGRESS"`，App Runner 會開始暫停您的服務。

若要使用 App Runner API 或 繼續您的服務 AWS CLI，請呼叫 ResumeService API 動作。如果呼叫傳回成功回應，且服務物件顯示 `"Status": "OPERATION_IN_PROGRESS"`，App Runner 會開始恢復您的服務。

刪除 App Runner 服務

當您想要終止服務中執行的 Web 應用程式時 AWS App Runner ，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，以及刪除您的相關資料。

由於以下一個或多個原因，您可能想要刪除 App Runner 服務：

- 您不再需要 Web 應用程式 – 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 – 您想要在相同的 中建立新的服務， AWS 區域 並且已達到與 帳戶相關聯的配額。如需詳細資訊，請參閱the section called “App Runner 資源配額”。
- 安全性或隱私權考量 – 您希望 App Runner 刪除其為您的服務存放的資料。

暫停和刪除比較

暫停您的 App Runner 服務以暫時停用它。只有運算資源會終止，而您的儲存資料（例如應用程式版本的容器映像）會保持不變。繼續您的服務非常快速，您的應用程式已準備好部署到新的運算資源。您的 App Runner 網域保持不變。

刪除您的 App Runner 服務以永久移除它。您的已儲存資料會遭到刪除。如果您需要重新建立服務，App Runner 需要再次擷取您的來源，如果它是程式碼儲存庫，也需要建置它。您的 Web 應用程式 取得一個新的 App Runner 網域。

App Runner 會刪除哪些項目？

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

App Runner 刪除的項目：

- 容器映像 – 您部署的映像複本，或 App Runner 從原始程式碼建置的映像。它使用 App Runner 擁有的內部 存放在 Amazon Elastic Container Registry AWS 帳戶 (Amazon ECR) 中。
- 服務組態 – 與您的 App Runner 服務相關聯的組態設定。它們會使用 App Runner 擁有的內部 存放在 AWS 帳戶 Amazon DynamoDB 中。

App Runner 未刪除的項目：

- 連線 – 您可能有與服務相關聯的連線。App Runner 連線是在數個 App Runner 服務之間共用的個別資源。如果您不再需要連線，您可以明確刪除它。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。它們存放在 AWS Certificate Manager (ACM) 中。App Runner 不會在網域從服務取消連結後的七天內刪除憑證，也不會在刪除服務之後刪除憑證。如需詳細資訊，請參閱the section called “自訂網域名稱”。

刪除您的服務

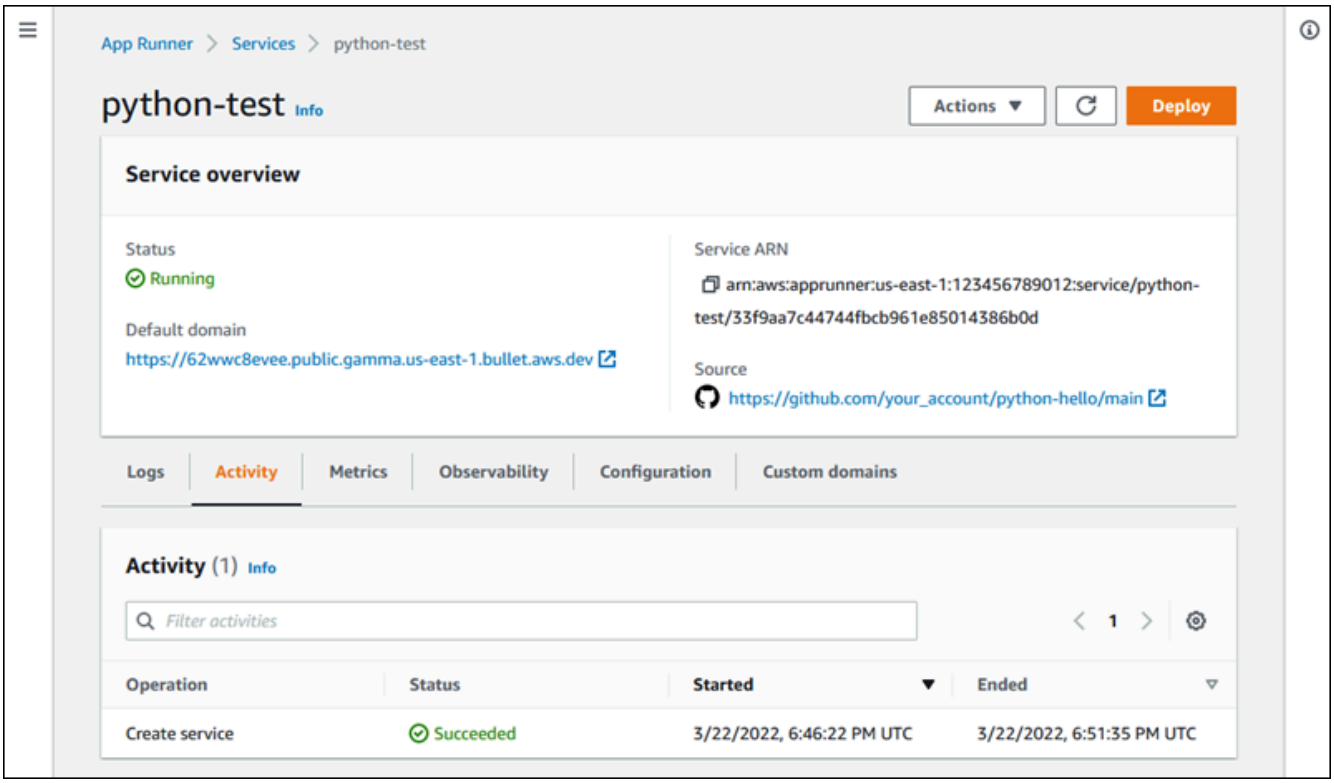
使用下列其中一種方法刪除您的 App Runner 服務：

App Runner console

使用 App Runner 主控台刪除您的服務

1. 開啟 App Runner 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，然後選擇您的 App Runner 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。您的服務現在已刪除。

App Runner API or AWS CLI

若要使用 App Runner API 或 刪除您的服務 AWS CLI，請呼叫 DeleteService API 動作。如果呼叫傳回成功回應，且服務物件顯示 `"Status": "OPERATION_IN_PROGRESS"`，則 App Runner 會開始刪除您的服務。

參考環境變數

透過 App Runner，您可以在建立服務或更新服務時，將秘密和組態參考為服務中的環境變數。

您可以參考非敏感組態資料，例如純文字中的逾時和重試計數作為鍵/值對。您在純文字中參考的組態資料不會加密，其他人可以在 App Runner 服務組態和應用程式日誌中看到。

> **Note**
>
> 基於安全考量，請勿在 App Runner 服務中參考純文字中的任何敏感資料。

參考敏感資料做為環境變數

App Runner 支援在服務中安全地參考敏感資料做為環境變數。考慮在 AWS Secrets Manager或 AWS Systems Manager 參數存放區中存放您要參考的敏感資料。然後，您可以從 App Runner 主控台或呼叫 API，安全地在服務中將其參考為環境變數。這可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**
>
> App Runner 不會因為參考 Secrets Manager 和 SSM 參數存放區做為環境變數而向您收費。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

以下是參考敏感資料做為環境變數的程序：

1. 在 或 AWS Secrets Manager AWS Systems Manager 參數存放區中，將 API 金鑰、資料庫登入資料、資料庫連線參數或應用程式版本等敏感資料儲存為秘密或參數。
2. 更新執行個體角色的 IAM 政策，讓 App Runner 可以存取存放在 Secrets Manager 和 SSM 參數存放區中的秘密和參數。如需詳細資訊，請參閱 許可。

3. 透過指派名稱並提供其 Amazon Resource Name (ARN)，將秘密和參數安全地參考為環境變數。您可以在建立服務或更新服務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：
   - App Runner 主控台
   - App Runner API
   - `apprunner.yaml` 組態檔案

> **Note**
> 您無法在建立或更新 App Runner 服務時，將 指派`PORT`為環境變數的名稱。這是 App Runner 服務的預留環境變數。

如需如何參考秘密和參數的詳細資訊，請參閱管理環境變數。

> **Note**
> 由於 App Runner 只會儲存對秘密和參數 ARNs參考，因此其他人在 App Runner 服務組態和應用程式日誌中看不到敏感資料。

考量事項

- 請務必使用適當的許可來更新執行個體角色，以存取 參數存放區中 AWS Secrets Manager 或 中的 AWS Systems Manager 秘密和參數。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 AWS 帳戶 的服務位於相同的 中。目前，您無法跨帳戶參考 SSM 參數存放區參數。
- 當秘密和參數值輪換或變更時，它們不會在您的 App Runner 服務中自動更新。重新部署您的 App Runner 服務，因為 App Runner 只會在部署期間提取秘密和參數。
- 您也可以選擇透過 App Runner 服務中的 SDK 直接呼叫 AWS Secrets Manager 和 AWS Systems Manager 參數存放區。
- 為了避免錯誤，在將它們參考為環境變數時，請確定下列事項：
  - 您可以指定秘密的正確 ARN。
  - 您可以指定 參數的正確名稱或 ARN。

許可

若要啟用參考存放在 AWS Secrets Manager 或 SSM 參數存放區中的秘密和參數，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，App Runner 無法存取您帳戶中的資源。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 主控台中更新您的執行個體角色。您可以修改這些政策範本，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，以新增從 參考秘密的許可AWS Secrets Manager。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue",
            "kms:Decrypt*"
        ],
        "Resource": [
            "arn:aws:secretsmanager:us-east-1:111122223333:secret:my-secret",
            "arn:aws:kms:us-east-1:111122223333:key/my-key"
         ]
     }
   ]
}
```

將下列範本複製到執行個體角色，以新增從AWS Systems Manager參數存放區參考參數的許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
            "arn:aws:ssm:us-east-1:111122223333:parameter/my-parameter"
            ]
        }
    ]
}
```

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 “App Runner 主控台”
- the section called “App Runner API 或 AWS CLI”

App Runner 主控台

當您在 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

新增環境變數

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 根據您是要建立或更新服務，請執行下列其中一個步驟：
    - 如果您要建立新的服務，請選擇建立 App Runner 服務，然後前往設定服務。

- 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. 前往環境變數 - 服務設定下的選用。
4. 根據您的需求選擇下列任一選項：

- 從環境變數來源中選擇純文字，並在環境變數名稱和環境變數值下分別輸入其鍵值對。

> **Note**
>
> 如果您想要參考非敏感資料，請選擇純文字。此資料不會加密，其他人可以在 App Runner 服務組態和應用程式日誌中看到。

- 從環境變數來源中選擇 Secrets Manager，以參考存放在 中的秘密 AWS Secrets Manager 做為服務中的環境變數。分別在環境變數名稱和環境變數值下，提供您參考之秘密的環境變數名稱和 Amazon Resource Name (ARN)。
- 從環境變數來源選擇 SSM 參數存放區，以參考存放在 SSM 參數存放區中的參數做為服務中的環境變數。分別在環境變數名稱和環境變數值下，提供您參考之參數的環境變數名稱和 ARN。

> **Note**
>
> - 您無法在建立或更新 App Runner 服務時，將 指派`PORT`為環境變數的名稱。這是 App Runner 服務的預留環境變數。
> - 如果 SSM 參數存放區參數與您要啟動 AWS 區域 的服務位於相同位置，您可以指定完整 Amazon Resource Name (ARN) 或 參數的名稱。如果 參數位於不同的區域，您需要指定完整的 ARN。
> - 請確定您參考的 參數與您啟動或更新的服務位於相同的 帳戶。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考另一個環境變數。
6. 展開 IAM 政策範本，以檢視和複製為 AWS Secrets Manager 和 SSM 參數存放區提供的 IAM 政策範本。只有在您尚未使用所需的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。如果應用程式碼未更新，您的 App Runner 服務可能會失敗。

移除環境變數

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往您要更新之服務的組態索引標籤。
3. 前往環境變數 - 服務設定下的選用。
4. 選擇您要移除的環境變數旁的移除。您會收到確認刪除的訊息。
5. 選擇 刪除。

App Runner API 或 AWS CLI

您可以參考存放在 Secrets Manager 和 SSM 參數存放區中的敏感資料，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，讓 App Runner 可以存取存放在 Secrets Manager 和 SSM 參數存放區的秘密和參數。如需詳細資訊，請參閱 許可。

參考秘密和組態做為環境變數

1. 在 Secrets Manager 或 SSM 參數存放區中建立秘密或組態。

   下列範例示範如何使用 SSM 參數存放區建立秘密和參數。

   Example建立秘密 - 請求

   下列範例示範如何建立代表資料庫登入資料的秘密。

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

   Example建立秘密 - 回應

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

Example建立組態 - 請求

下列範例示範如何建立代表 RDS 連線字串的參數。

```
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. 將秘密和組態新增為環境變數，以參考儲存在 Secrets Manager 和 SSM 參數存放區中的秘密和組態。您可以在建立或更新 App Runner 服務時新增環境變數。

   下列範例示範如何在程式碼型和映像型 App Runner 服務上將秘密和組態參考為環境變數。

   Example影像型 App Runner 服務的 Input.json 檔案

```
{
  "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以映像為基礎的 App Runner 服務 – 請求

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

Example以映像為基礎的 App Runner 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程式碼型 App Runner 服務的 Input.json 檔案

```
{
  "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程式碼型 App Runner 服務 – 請求

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

Example程式碼型 App Runner 服務 – 回應

```
{
...
   "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秘密。

   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"
```

使用 App Runner 進行聯網

本章說明 AWS App Runner 服務的聯網組態。

從本章您將學到以下內容：

- 如何設定私有和公有端點的傳入流量。如需詳細資訊，請參閱設定傳入流量的網路組態。
- 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，請參閱啟用傳出流量的 VPC 存取。

主題

- 術語
- 設定傳入流量的網路組態
- 啟用傳出流量的 VPC 存取

術語

為了了解如何自訂您的網路流量以符合您的需求，讓我們了解本章中使用的下列術語。

一般條款

若要了解與 Amazon Virtual Private Cloud (VPC) 建立關聯的必要條件，讓我們了解下列術語：

- VPC：Amazon VPC 是一種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。它是一種虛擬網路，與您要在自己的資料中心操作的傳統網路非常相似。
- VPC 介面端點：VPC 介面端點是 AWS PrivateLink 資源，可將 VPC 連線至端點服務。建立 VPC 介面端點，將流量傳送至使用 Network Load Balancer 分佈流量的端點服務。使用 DNS 來解析目的地為端點服務的流量。
- 區域：每個區域都是獨立的地理區域，您可以在其中託管 App Runner 服務。
- 可用區域：可用區域是 AWS 區域內的隔離位置。這是一或多個具有備援電源、聯網和連線能力的離散資料中心。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 IP 地址的範圍。子網必須位於單一可用區域。您可以在指定的子網路中啟動 AWS 資源。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。

- 安全群組：安全群組控制允許存取的流量，並保留與其相關聯的資源。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，讓您更能控制網路流量。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其建立所在 VPC 內的資源建立關聯。
- 雙堆疊：雙堆疊是一種地址類型，可支援來自 IPv4 和 IPv6 端點的網路流量。

設定傳出流量的特定術語

VPC 連接器

VPC Connector 是一種 App Runner 資源，可讓 App Runner 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

若要了解如何讓服務只能從 Amazon VPC 內私下存取，讓我們了解下列術語：

- VPC 傳入連線：VPC 傳入連線是一種 App Runner 資源，可為傳入流量提供 App Runner 端點。當您在 App Runner 主控台上為傳入流量選擇私有端點時，App Runner 會在幕後指派 VPC 傳入連線資源。VPC 傳入連線資源會將您的 App Runner 服務連線至 Amazon VPC 的 VPC 介面端點。

  > **Note**
  > 如果您使用的是 App Runner API，則不會自動建立 VPC 輸入連線資源。

- 私有端點：私有端點是您選取的 App Runner 主控台選項，可將傳入網路流量設定為只能在 Amazon VPC 內存取。

設定傳入流量的網路組態

您可以將服務設定為接收來自私有或公有端點的傳入流量。

公有端點是預設組態。它會將您的服務開啟至任何來自公有網際網路的傳入流量。它還可讓您靈活地為您的服務選擇 IPv4 或雙堆疊 (IPv4 和 IPv6) 地址類型。

私有端點僅允許來自 Amazon VPC 的流量存取您的 App Runner 服務。這是透過為您的 App Runner 服務設定 VPC 介面端點 AWS PrivateLink 、 資源來實現的。因此，在 Amazon VPC 和 App Runner 服務之間建立私有連線。它還可讓您靈活地為您的服務選擇 IPv4 或雙堆疊 (IPv4 和 IPv6) 地址類型。

以下是設定傳入流量的網路組態時所涵蓋的主題：

- 如何設定傳入流量，讓您的服務只能從 Amazon VPC 內私下使用。如需詳細資訊，請參閱啟用傳入流量的私有端點。
- 如何設定您的服務以接收來自雙堆疊地址類型的網際網路流量。如需詳細資訊，請參閱為公有傳入流量啟用雙堆疊。

標頭

透過 App Runner，您可以存取進入應用程式的流量原始來源 IPv4 和 IPv6 地址。將`X-Forwarded-For`請求標頭指派給原始來源 IP 地址，即可保留原始來源 IP 地址。這可讓您的應用程式在需要時擷取原始來源 IP 地址。

> **Note**
>
> 如果您的服務設定為使用私有端點，則`X-Forwarded-For`請求標頭無法用於存取原始來源 IP 地址。如果使用，它會擷取 false 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。不過，您也可以將 App Runner 服務設為私有，並且只能在 Amazon Virtual Private Cloud (Amazon VPC) 內存取。

透過 App Runner 服務私有，您可以完全控制傳入的流量，增加額外的安全層。這在各種使用案例中很有用，包括執行內部 APIs、公司 Web 應用程式或仍在開發中且需要更高層級隱私權和安全性的應用程式，或需要滿足特定合規要求的應用程式。

> **Note**
>
> 如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。
> 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》中的下列主題：使用安全群組控制網路流量和控制 AWS 資源的流量。 https://docs.aws.amazon.com//vpc/latest/userguide/vpc-security-groups.html

當您的 App Runner 服務為私有時，您可以從 Amazon VPC 內存取您的服務。不需要網際網路閘道、NAT 裝置或 VPN 連線。

> **Note**
>
> App Runner 支援IPv4流量和傳出流量的 IPv4 和雙堆疊 (IPv4 和 IPv6)。

考量事項

- 在您設定 App Runner 的 VPC 介面端點之前，請檢閱《 AWS PrivateLink 指南》中的考量事項。
- App Runner 不支援 VPC 端點政策。根據預設，允許透過 VPC 介面端點完整存取 App Runner。或者，您可以將安全群組與端點網路介面建立關聯，以控制透過 VPC 介面端點傳送至 App Runner 的流量。
- 如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。
- 啟用私有端點之後，您的服務只能從 VPC 存取，也無法從網際網路存取。
- 為了提高可用性，建議您為 VPC 介面端點在可用區域之間至少選取兩個不同的子網路。我們不建議僅使用一個子網路。
- 如果您要為 IP 地址類型選擇雙堆疊選項，請確保您的子網路可以支援雙堆疊流量。
- 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。

如需本節所用詞彙的資訊，請參閱術語。

許可

以下是啟用私有端點所需的許可清單：

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

- ec2：DescribeVpcs

VPC 介面端點

VPC 介面端點是將 Amazon VPC 連線至端點服務AWS PrivateLink的資源。您可以透過傳遞 VPC 介面端點，指定要在其中存取 App Runner 服務的 Amazon VPC。若要建立 VPC 介面端點，請指定下列項目：

- 啟用連線的 Amazon VPC。
- 新增安全群組。根據預設，安全群組會指派給 VPC 介面端點。您可以選擇建立自訂安全群組的關聯，以進一步控制傳入的網路流量。
- 新增子網路。為了確保更高的可用性，建議您為存取 App Runner 服務的每個可用區域至少選取兩個子網路。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。這些是請求者管理的網路介面，可做為目的地為 App Runner 之流量的進入點。申請者管理的網路界面是 AWS 服務代表您在 VPC 中建立的網路界面。
- 如果您使用 API，請新增 App Runner VPC 介面端點 `Servicename`。例如

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

您可以使用下列其中一個 AWS 服務建立 VPC 介面端點：

- App Runner 主控台。如需詳細資訊，請參閱管理私有端點。
- Amazon VPC 主控台或 API，以及 AWS Command Line Interface (AWS CLI)。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的「透過 AWS PrivateLink存取 AWS 服務」。

> **Note**
>
> 您需要為根據AWS PrivateLink 定價使用的每個 VPC 介面端點付費。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。不過，為了獲得更好的隔離，請考慮為每個 App Runner 服務關聯不同的 VPC 介面端點。

VPC Ingress Connection

VPC 傳入連線是一種 App Runner 資源，可指定傳入流量的 App Runner 端點。當您在 App Runner 主控台上為傳入流量選擇私有端點時，App Runner 會在幕後指派 VPC 傳入連線資源。選擇此選項，僅

允許來自 Amazon VPC 的流量存取您的 App Runner 服務。VPC 傳入連線資源會將您的 App Runner 服務連線至 Amazon VPC 的 VPC 介面端點。只有在您使用 API 操作來設定傳入流量的網路設定時，才能建立 VPC 傳入連線資源。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 參考中的 CreateVpcIngressConnection。

> **Note**
>
> App Runner 的一個 VPC 傳入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 App Runner 服務建立一個 VPC 輸入連線資源。

私有端點

私有端點是 App Runner 主控台選項，您可以選擇是否只想要接收來自 Amazon VPC 的傳入流量。在 App Runner 主控台上選擇私有端點選項，可讓您透過設定 VPC 介面端點，將服務連線至 VPC。在幕後，App Runner 會將 VPC 輸入連線資源指派給您設定的 VPC 介面端點。

摘要

只允許來自 Amazon VPC 的流量存取您的 App Runner 服務，讓您的服務成為私有。若要達成此目的，您可以使用 App Runner 或 Amazon VPC 為選取的 Amazon VPC 建立 VPC 介面端點。在 App Runner 主控台上，當您為傳入流量啟用私有端點時，您可以建立 VPC 介面端點。然後，App Runner 會自動建立 VPC 輸入連線資源，並連線至 VPC 介面端點和您的 App Runner 服務。這會建立私有服務連線，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來管理傳入流量的私有端點：

- the section called “App Runner 主控台”
- the section called “App Runner API 或 AWS CLI”

> **Note**
>
> 如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。

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》中的下列主題：使用安全群組控制網路流量和控制 AWS 資源的流量。 https://docs.aws.amazon.com//vpc/latest/userguide/vpc-security-groups.html

App Runner 主控台

當您使用 App Runner 主控台建立服務，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。

- 公有端點：讓所有服務都能透過網際網路存取您的服務。預設會選取公有端點。
- 私有端點：讓 App Runner 服務只能從 Amazon VPC 內存取。

啟用私有端點

將私有端點與您要存取的 Amazon VPC 的 VPC 介面端點建立關聯，以啟用私有端點。您可以建立新的 VPC 介面端點，或選擇現有的 VPC 介面端點。

建立 VPC 介面端點

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往設定服務下的聯網區段。
3. 針對傳入網路流量選擇私有端點。使用 VPC 介面端點連線至 VCP 的選項隨即開啟。
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。建議選擇兩個或多個安全群組。如果您未選擇安全群組，App Runner 會將預設安全群組指派給 VPC 介面端點。確保安全群組規則不會封鎖想要與 App Runner 服務通訊的資源。安全群組規則必須允許將與您的 App Runner 服務互動的資源。

> **Note**
>
> 如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。

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》中的下列主題：使用安全群組控制網路流量和控制 AWS 資源的流量。 https://docs.aws.amazon.com//vpc/latest/userguide/vpc-security-groups.html

8. 從下拉式清單中選擇所需的子網路。建議您為存取 App Runner 服務的每個可用區域至少選取兩個子網路。

> **Note**
> 如果您要為雙堆疊設定端點，請確定您的基礎設施和 VPC 端點支援雙堆疊流量。

9. （選用）選擇新增標籤，然後輸入標籤索引鍵和標籤值。
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 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往設定服務下的聯網區段。
3. 針對傳入網路流量選擇私有端點。使用 VPC 介面端點連線至 VPC 的選項隨即開啟。隨即顯示可用的 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. 選擇下一步以建立您的服務。App Runner 會啟用私有端點。

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

若要檢查私有端點的詳細資訊，請前往您的服務，並展開組態索引標籤下的聯網區段。它會顯示與私有端點相關聯的 VPC 和 VPC 介面端點的詳細資訊。

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。

更新 VPC 介面端點

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往您的服務，然後在左側面板選擇聯網組態。
3. 選擇傳入流量，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。編輯 VPC 介面端點的對話方塊隨即開啟。
6. 選擇所需的安全群組和子網路，然後按一下更新。顯示 VPC 介面端點詳細資訊的頁面會開啟，並在頂端列顯示 VPC 介面端點成功更新的訊息。

> **Note**
> 如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。
> 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》中的下列主題：使用安全群組控制網路流量和控制 AWS 資源的流量。 https://docs.aws.amazon.com//vpc/latest/userguide/vpc-security-groups.html

刪除 VPC 介面端點

如果您不希望 App Runner 服務可私下存取，您可以將傳入流量設定為公有。變更為公有會移除私有端點，但不會刪除 VPC 介面端點

刪除 VPC 介面端點

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往您的服務，然後在左側面板選擇聯網組態。
3. 選擇傳入流量，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，請先更新服務，將其從其連線的所有服務中移除。

4. 選擇 刪除。

   如果有服務連接到 VPC 介面端點，您會收到無法刪除 VPC 介面端點訊息。如果沒有服務連接到 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傳入流量開啟，並在頂端列成功刪除 VPC 介面端點的訊息。

App Runner API 或 AWS CLI

您可以在只能從 Amazon VPC 內存取的 App Runner 上部署應用程式。

如需讓服務成為私有所需許可的資訊，請參閱 the section called “許可”。

建立 Amazon VPC 的私有服務連線

1. 建立 VPC 介面端點，即 AWS PrivateLink 資源，以連線至 App Runner。若要這樣做，請指定要與應用程式建立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 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。因此，與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。
   > 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》中的下列主題：使用安全群組控制網路流量和控制 AWS 資源的流量。 https://docs.aws.amazon.com//vpc/latest/userguide/vpc-security-groups.html

   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 App Runner 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。在 `NetworkConfiguration` 參數`IngressConfiguration`的成員`False`中將 `IsPubliclyAccessible`設定為 。或者，您可以將 `IpAddressType` 欄位設定為 `IPV4`或 `DUAL_STACK`。如果未設定，則此值預設為 IPV4。下列範例參考 VPC 介面端點。

   Example

```
aws apprunner create-service
 --network-configuration:
   {
   "IngressConfiguration":
      {
      "IsPubliclyAccessible": False
      },
      "IpAddressType": "IPV4"
   }
 --service-name: com.amazonaws.us-east-1.apprunner.requests
 --source-configuration: <source_configuration>
```

3. 呼叫 `create-vpc-ingress-connection` API 動作來建立 App Runner 的 VPC 輸入連線資源，並將其與您在上一個步驟中建立的 VPC 介面端點建立關聯。它會傳回用於在指定 VPC 中存取服務的網域名稱。以下是建立 VPC 傳入連線資源的範例。

   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 傳入連線資源。VPC 傳入連線必須處於下列其中一種狀態，才能更新：

- AVAILABLE
- FAILED_CREATION
- FAILED_UPDATE

以下是更新 VPC 傳入連線資源的範例。

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

如果您不再需要與 Amazon VPC 的私有連線，您可以刪除 VPC 傳入連線資源。

VPC 傳入連線必須處於下列其中一種狀態，才能刪除：

- AVAILABLE
- 失敗的建立

- 失敗的更新
- 失敗的刪除

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 App Runner API 動作來管理服務的私有傳入流量。

- CreateVpcIngressConnection – 建立新的 VPC 傳入連線資源。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 傳入連線端點清單。
- DescribeVpcIngressConnection – 傳回 AWS App Runner VPC Ingress Connection 資源的完整描述。
- UpdateVpcIngressConnection – 更新 AWS App Runner VPC 傳入連線資源。
- DeleteVpcIngressConnection – 刪除與 App Runner 服務相關聯的 App Runner VPC Ingress Connection 資源。

如需使用 App Runner API 的詳細資訊，請參閱 App Runner API 參考指南。

為傳入流量啟用 IPv6

如果您希望服務接收來自 IPv6 地址或 IPv4 和 IPv6 地址的傳入網路流量，請選擇端點的雙堆疊地址類型。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。下列程序說明如何使用 App Runner 主控台或 App Runner API 啟用 IPv4 或雙堆疊 (IPv6 和 IPv4)。

管理傳入流量的雙堆疊

使用下列其中一種方法來管理傳入流量的雙堆疊地址類型：

- the section called “App Runner 主控台”
- the section called “App Runner API 或 AWS CLI”

> **Note**
>
> 下列程序說明如何管理公有傳入流量的網路地址類型。如需有關管理私有端點的雙堆疊或 IPv4 地址類型的資訊，請參閱 the section called “管理私有端點”。

App Runner 主控台

當您使用 App Runner 主控台建立服務，或稍後更新其組態時，您可以選擇傳入網際網路流量的雙堆疊地址類型。

啟用雙堆疊地址類型

1. 建立或更新服務時，請展開設定服務下的聯網區段。
2. 選擇傳入網路流量的公有端點。如果您選取公有端點，則會開啟端點 IP 地址類型選項。

   the section called “管理私有端點” 如需管理私有端點雙堆疊或 IPv4 地址類型的程序，請參閱 。
3. 展開端點 IP 地址類型以檢視下列 IP 地址類型。

   - IPv4
   - 雙堆疊 (IPv4 和 IPv6)

   > **Note**
   >
   > 如果您未展開端點 IP 地址類型進行選擇，則 App Runner 會將 IPv4 指派為預設組態。

4. 選擇雙堆疊 (IPv4 和 IPv6)。
5. 如果您要建立服務，請選擇下一步，然後選擇建立和部署。否則，如果您要更新服務，請選擇儲存變更。

   部署服務時，您的應用程式會開始從 IPv4 和 IPv6 端點接收網路流量。

變更地址類型

1. 依照步驟更新服務並導覽至聯網。
2. 導覽至傳入網路流量下的端點 IP 地址類型，然後選取所需的地址類型。
3. 選擇儲存變更。您的服務會隨著您的選擇而更新。

App Runner API 或 AWS CLI

當您呼叫 CreateService 或 UpdateService App Runner API 動作時，請使用 `NetworkConfiguration` 參數`IpAddressType`的成員來指定地址類型。您可以指定的支援值為 `IPv4`和 `DUAL_STACK`。指定`DUAL_STACK`您是否希望服務從 IPv4 和 IPv6 端點接收網際網路流量。如果您未指定 的任何值`IpAddressType`，預設會套用 IPv4。

> **Note**
>
> 如需私有端點範例，請參閱 the section called “App Runner API 或 AWS CLI”。

以下是使用雙堆疊做為 IP 地址建立服務的範例。此範例會呼叫 `input.json` 檔案。

Example請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"
}
```

如需 API 參數的詳細資訊，請參閱 NetworkConfiguration。

啟用傳出流量的 VPC 存取

根據預設，您的 AWS App Runner 應用程式可以傳送訊息至公有端點。這包括您自己的解決方案 AWS 服務，以及任何其他公有網站或 Web 服務。您的應用程式甚至可以從 Amazon Virtual Private Cloud (Amazon VPC) 將訊息傳送至 VPC 中執行之應用程式的公有端點。如果您在啟動環境時未設定 VPC，App Runner 會使用預設的 VPC，這是公有的。

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 Virtual Private Cloud (Amazon VPC) 存取在私有 VPC 中執行的應用程式。執行此操作後，您的應用程式可以與 連線，並將訊息傳送到 Amazon Virtual Private Cloud (Amazon VPC) 中託管的其他應用程式。例如，Amazon RDS 資料庫、Amazon ElastiCache，以及託管在私有 VPC 中的其他私有服務。

VPC 連接器

您可以從稱為 VPC Connector 的 App Runner 主控台建立 VPC 端點，將服務與 VPC 建立關聯。若要建立 VPC 連接器，請指定 VPC、一或多個子網路，以及選擇性一或多個安全群組。設定 VPC Connector 之後，您可以將其與一或多個 App Runner 服務搭配使用。

一次性延遲

如果您使用用於傳出流量的自訂 VPC 連接器設定 App Runner 服務，它可能會遇到 2 到 5 分鐘的一次性啟動延遲。啟動程序會等到 VPC Connector 準備好連線到其他資源，再將服務狀態設定為執行中。您可以在第一次建立服務時，使用自訂 VPC 連接器來設定服務，也可以在之後透過執行服務更新來進行設定。

請注意，如果您為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 連接器組態是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 Connector Hyperplane ENIs（彈性網路介面）期間發生一次。

有關自訂 VPC 連接器和 AWS Hyperplane 的詳細資訊

App Runner 中的 VPC 連接器是以 AWS Hyperplane 為基礎，Hyperplane 是位於數種 AWS 資源後方的內部 Amazon 網路系統，例如 Network Load Balancer、NAT Gateway 和 AWS PrivateLink。 AWS Hyperplane 技術提供高輸送量和低延遲功能，以及更高程度的共用。當您建立 VPC 連接器並將其與服務建立關聯時，會在子網路中建立 Hyperplane ENI。VPC 連接器組態是以安全群組和子網路組合為基礎，您可以在多個 App Runner 服務之間參考相同的 VPC 連接器。因此，基礎 Hyperplane ENIs會跨 App Runner 服務共用。此共用是可行的，即使您擴展處理請求負載所需的任務數量，也會導致 VPC 中 IP 空間的更有效率使用率。如需詳細資訊，請參閱 AWS 容器部落格中的 Deep Dive on AWS App Runner VPC Networking。

子網路

每個子網路都位於特定的可用區域。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。

選取 VPC 的子網路時，請確定您選擇的是私有子網路，而不是公有子網路。這是因為當您建立 VPC 連接器時，App Runner 服務會在每個子網路中建立 Hyperplane ENI。每個 Hyperplane ENI 只會指派一個私有 IP 地址，並使用 AWSAppRunnerManaged 金鑰的標籤進行標記。如果您選擇公有子網路，則執行 App Runner 服務時發生錯誤。不過，如果您的服務需要存取網際網路或其他公有服務 AWS 服務，請參閱 the section called “選取子網路時的考量事項”。

選取子網路時的考量事項

- 當您將服務連線至 VPC 時，傳出流量無法存取公有網際網路。來自您應用程式的所有傳出流量都會透過您服務所連線的 VPC 導向。VPC 的所有聯網規則都適用於應用程式的傳出流量。這表示您的服務無法存取公有網際網路和 AWS APIs。若要取得存取權，請執行下列其中一項操作：
  - 透過 NAT Gateway 將子網路連接至網際網路。
  - 為您要存取 AWS 服務 的 設定 VPC 端點。您的服務會使用 保留在 Amazon VPC 中 AWS PrivateLink。
- 有些 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。如果您選擇這些可用區域中的子網路，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，請移除請求中不支援的子網路進行故障診斷，然後再試一次。
- 您選取的子網路必須具有相同的 IP 地址類型，可以是 IPv4 或雙堆疊。

安全群組

您可以選擇性地指定 App Runner 用來在指定子網路 AWS 下存取的安全群組。如果您未指定安全群組，App Runner 會使用 VPC 的預設安全群組。預設的安全群組會允許所有傳出流量。

新增安全群組可為 VPC 連接器提供額外的安全層，讓您更能控制網路流量。VPC Connector 僅用於來自應用程式的傳出通訊。您可以使用傳出規則來允許與所需目的地端點的通訊。您也必須確保與目的地資源相關聯的任何安全群組都有適當的傳入規則。否則，這些資源無法接受來自 VPC Connector 安全群組的流量。

> **Note**
>
> 當您將服務與 VPC 建立關聯時，下列流量不會受到影響：
>
> - 傳入流量 – 應用程式接收的傳入訊息不受關聯的 VPC 影響。訊息會透過與您服務相關聯的公有網域名稱路由，而不會與 VPC 互動。
> - App Runner 流量 – App Runner 會代表您管理數個動作，例如提取原始程式碼和映像、推送日誌，以及擷取秘密。這些動作產生的流量不會透過您的 VPC 路由。

若要進一步了解 如何與 Amazon VPC AWS App Runner 整合，請參閱 AWS App Runner VPC 網路。

管理 VPC 存取

> **Note**
>
> 如果您為服務建立傳出流量 VPC 連接器，接下來的服務啟動程序將經歷一次性延遲。您可以在建立新服務時或之後使用服務更新來設定新服務的此組態。如需詳細資訊，請參閱本指南的 Networking with App Runner 章節一次性延遲中的 。

使用下列其中一種方法來管理 App Runner 服務的 VPC 存取：

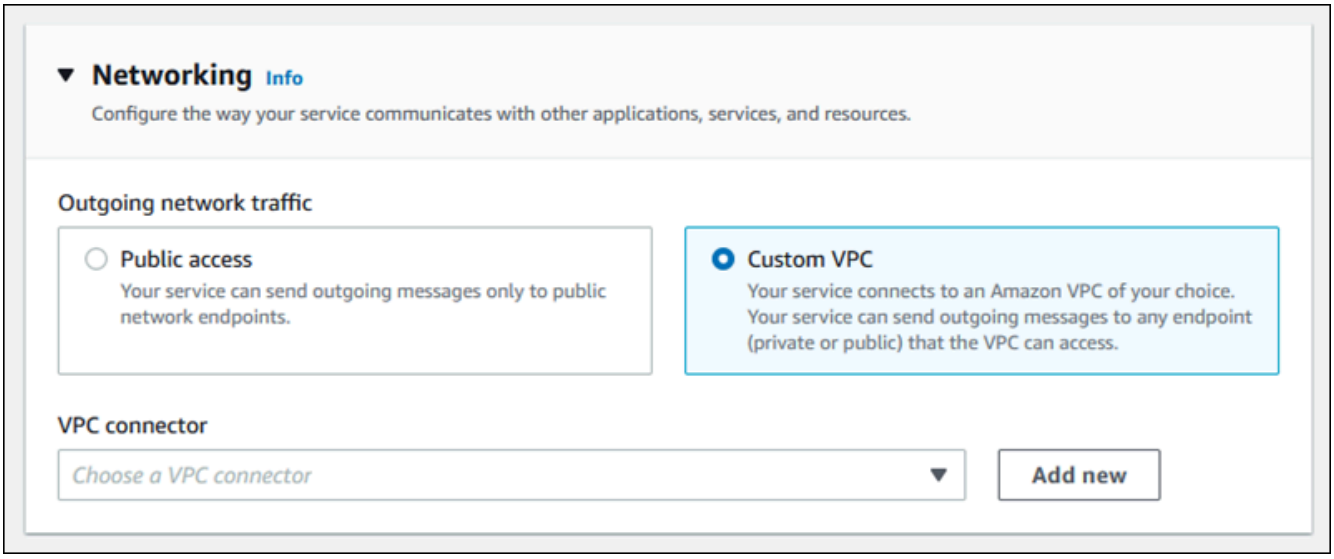
App Runner console

當您使用 App Runner 主控台建立服務，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組態區段。對於傳出網路流量，請在下列中選擇 ：

- 公有存取：將您的服務與其他公有端點建立關聯 AWS 服務。
- 自訂 VPC：將您的服務與來自 Amazon VPC 的 VPC 建立關聯。您的應用程式可與 連線，並將訊息傳送至 Amazon VPC 中託管的其他應用程式。

啟用自訂 VPC

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往設定服務下的聯網區段。



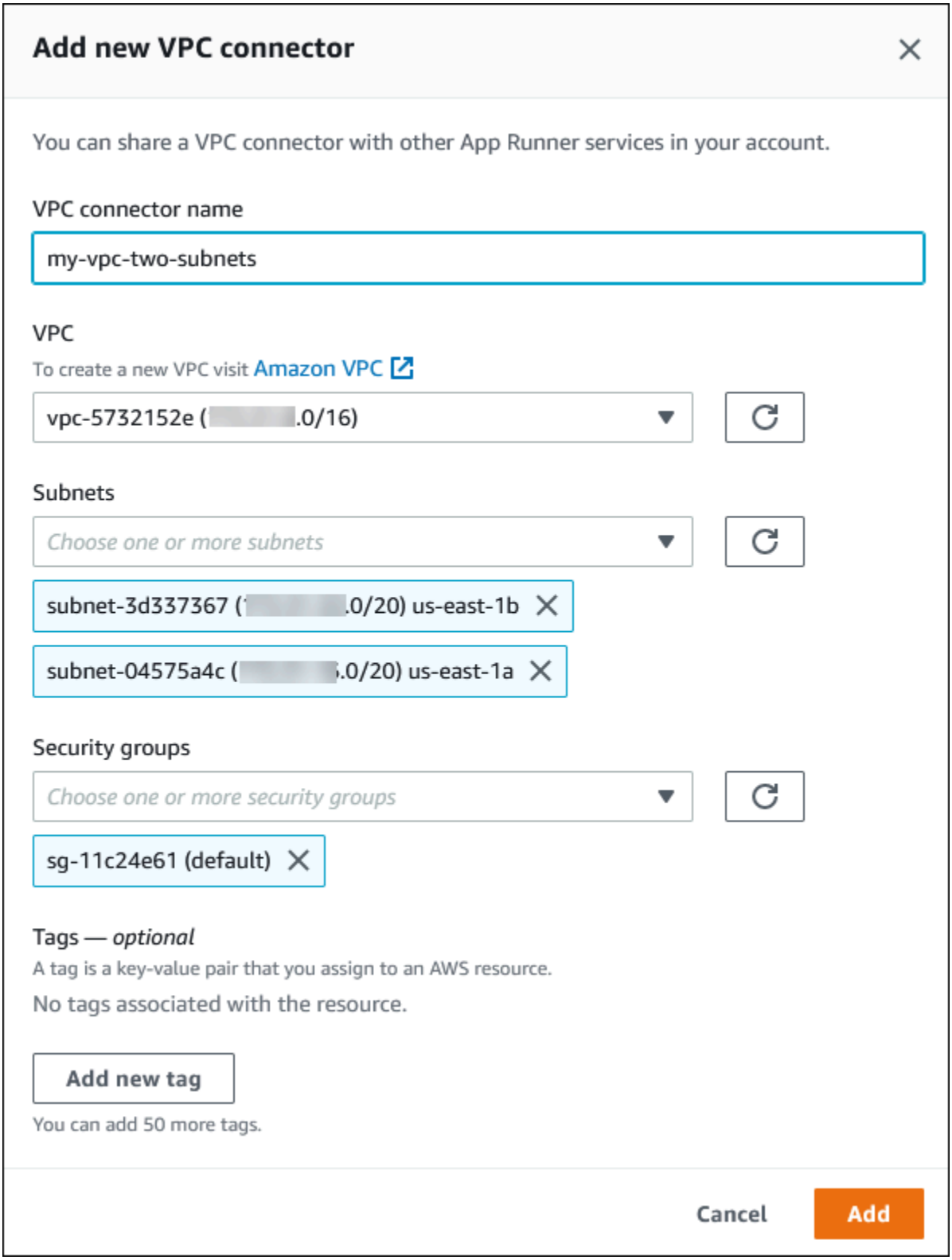
3. 針對傳出網路流量選擇自訂 VPC。

4. 在導覽窗格中，選擇 VPC 連接器。

   如果您建立了 VPC 連接器，主控台會顯示您帳戶中的 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您的組態。然後，移至最後一個步驟。或者，您可以使用下列步驟新增新的 VPC 連接器。

5. 選擇新增，為您的服務建立新的 VPC 連接器。

   然後，新增 VPC 連接器對話方塊隨即開啟。



6. 輸入 VPC 連接器的名稱，然後從可用清單中選擇所需的 VPC。
7. 針對子網路，為您計劃從中存取 App Runner 服務的每個可用區域選擇一個子網路。為了提高可用性，請選擇三個子網路。或者，如果少於三個子網路，請選擇所有可用的子網路。

> **Note**
>
> - 請務必將私有子網路指派給 VPC 連接器。如果您將公有子網路指派給 VPC 連接器，您的服務無法在更新期間自動建立或復原。
> - 如果您的傳出流量是雙堆疊，請確定您在 VPC 主控台中選取的所有子網路都已設定為雙堆疊。

8. （選用）對於安全群組，選取要與端點網路介面建立關聯的安全群組。
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 連接器詳細資訊會顯示在 VPC 連接器下方。
11. 選擇下一步以檢閱您的組態，然後選擇建立和部署。

    App Runner 會為您建立 VPC 連接器資源，然後將其與您的服務建立關聯。如果服務已成功建立，主控台會顯示服務儀表板，其中包含新服務的服務概觀。

App Runner API or AWS CLI

當您呼叫 CreateService 或 UpdateService App Runner API 動作時，請使用 `NetworkConfiguration` 參數`EgressConfiguration`的成員來指定服務的 VPC 連接器資源。

使用下列 App Runner API 動作來管理您的 VPC Connector 資源。

- CreateVpcConnector – 建立新的 VPC 連接器。
- ListVpcConnectors – 傳回與您的 相關聯的 VPC 連接器清單 AWS 帳戶。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 傳出存取權的 App Runner 上部署應用程式，您必須先建立 VPC 連接器。您可以透過指定要與應用程式建立關聯的一或多個子網路和安全群組來執行此操作。然後，您可以透過 CLI 在 Create 或 UpdateService 中參考 VPC Connector，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
```

App Runner 服務的可觀測性

AWS App Runner 與數個 AWS 服務整合，為您的 App Runner 服務提供廣泛的可觀測性工具套件。本章中的主題說明這些功能。

主題

- 追蹤 App Runner 服務活動
- 檢視串流至 CloudWatch Logs 的 App Runner 日誌
- 檢視報告給 CloudWatch 的 App Runner 服務指標
- 在 EventBridge 中處理 App Runner 事件
- 使用 記錄 App Runner API 呼叫 AWS CloudTrail
- 使用 X-Ray 追蹤 App Runner 應用程式

追蹤 App Runner 服務活動

AWS App Runner 會使用操作清單來追蹤 App Runner 服務中的活動。操作代表對 API 動作的非同步呼叫，例如建立服務、更新組態和部署服務。下列各節說明如何在 App Runner 主控台和使用 API 追蹤活動。

追蹤 App Runner 服務活動

使用下列其中一種方法追蹤您的 App Runner 服務活動：

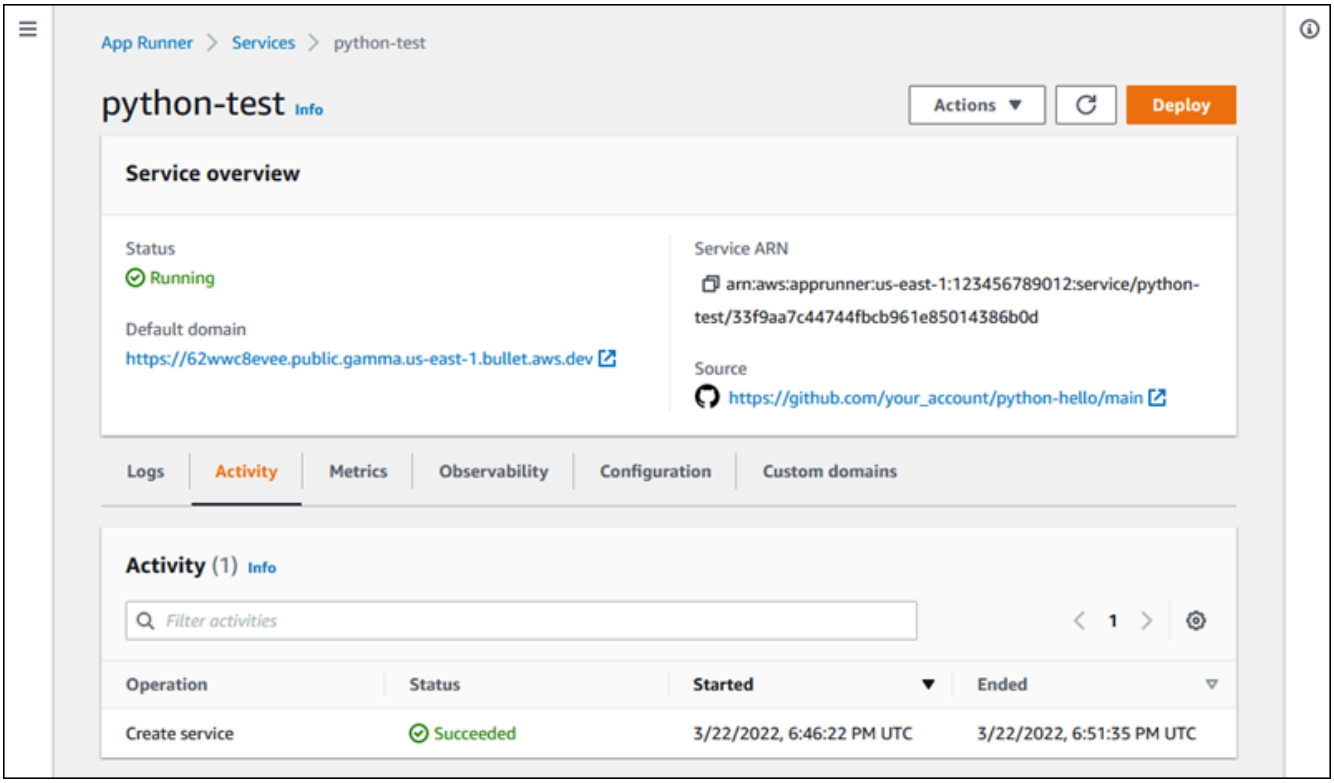
App Runner console

App Runner 主控台會顯示您的 App Runner 服務活動，並提供更多探索操作的方法。

檢視 服務的活動

1. 開啟 App Runner 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，然後選擇您的 App Runner 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，如果尚未選擇活動索引標籤，請選擇該索引標籤。

   主控台會顯示操作清單。

4. 若要尋找特定操作，請輸入搜尋詞彙以縮小清單範圍。您可以搜尋出現在資料表中的任何值。
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

ListOperations 動作，指定 App Runner 服務的 Amazon Resource Name (ARN)，會傳回此服務上發生的操作清單。每個清單項目都包含操作 ID 和一些追蹤詳細資訊。

檢視串流至 CloudWatch Logs 的 App Runner 日誌

您可以使用 Amazon CloudWatch Logs 來監控、存放和存取各種 AWS 服務中資源產生的日誌檔案。如需更多資訊，請參閱 Amazon CloudWatch Logs 使用者指南。

AWS App Runner 會收集應用程式部署和作用中服務的輸出，並將其串流至 CloudWatch Logs。下列各節列出 App Runner 日誌串流，並說明如何在 App Runner 主控台中檢視它們。

App Runner 日誌群組和串流

CloudWatch Logs 會將日誌資料保留在日誌串流中，以便進一步組織在日誌群組中。日誌串流是來自特定來源的一系列日誌事件。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。

App Runner 會為您 中的每個 App Runner 服務定義兩個 CloudWatch Logs 日誌群組，每個群組都有多個日誌串流 AWS 帳戶。

服務日誌

服務日誌群組包含 App Runner 在管理 App Runner 服務並對其採取行動時所產生的記錄輸出。

日誌群組名稱	範例
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/service`

在服務日誌群組中，App Runner 會建立事件日誌串流，以擷取 App Runner 服務生命週期中的活動。例如，這可能會啟動或暫停您的應用程式。

此外，App Runner 會為每個與您的服務相關的長時間執行非同步操作建立日誌串流。日誌串流名稱反映操作類型和特定操作 ID。

部署是一種操作。部署日誌包含 App Runner 在您建立服務或部署應用程式新版本時所執行建置和部署步驟的記錄輸出。部署日誌串流名稱以 開頭`deployment/`，並以執行部署的操作 ID 結尾。此操作是初始應用程式部署的 CreateService 呼叫，或是每個後續部署的 StartDeployment 呼叫。

在部署日誌中，每個日誌訊息都以字首開頭：

- `[AppRunner]` – App Runner 在部署期間產生的輸出。
- `[Build]` – 您自己的建置指令碼輸出。

日誌串流名稱	範例
`events`	不適用 （固定名稱）

日誌串流名稱	範例
operation-type /*operation-id*	deployment/c2c8eeedea164f459cf78f12a8953390

應用程式記錄

應用程式日誌群組包含執行中應用程式程式碼的輸出。

日誌群組名稱	範例
/aws/apprunner/ *service-name* /*service-id* /application	/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/application

在應用程式日誌群組中，App Runner 會為每個執行您應用程式的執行個體（擴展單位）建立日誌串流。

日誌串流名稱	範例
instance/ *instance-id*	instance/1a80bc9134a84699b7b3432ebeebb591

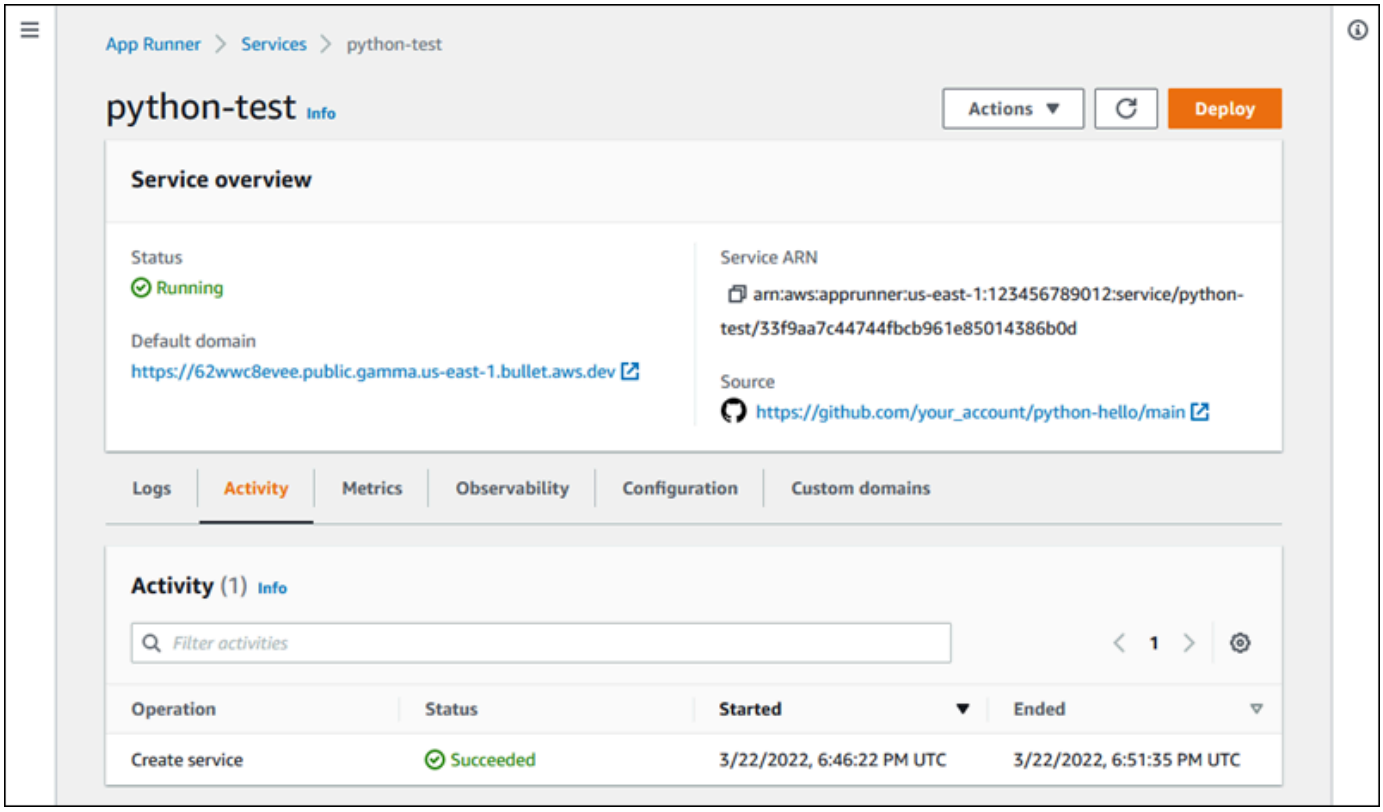
在主控台中檢視 App Runner 日誌

App Runner 主控台會顯示服務的所有日誌摘要，並可讓您檢視、探索和下載日誌。

檢視 服務的日誌

1. 開啟 App Runner 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，然後選擇您的 App Runner 服務。

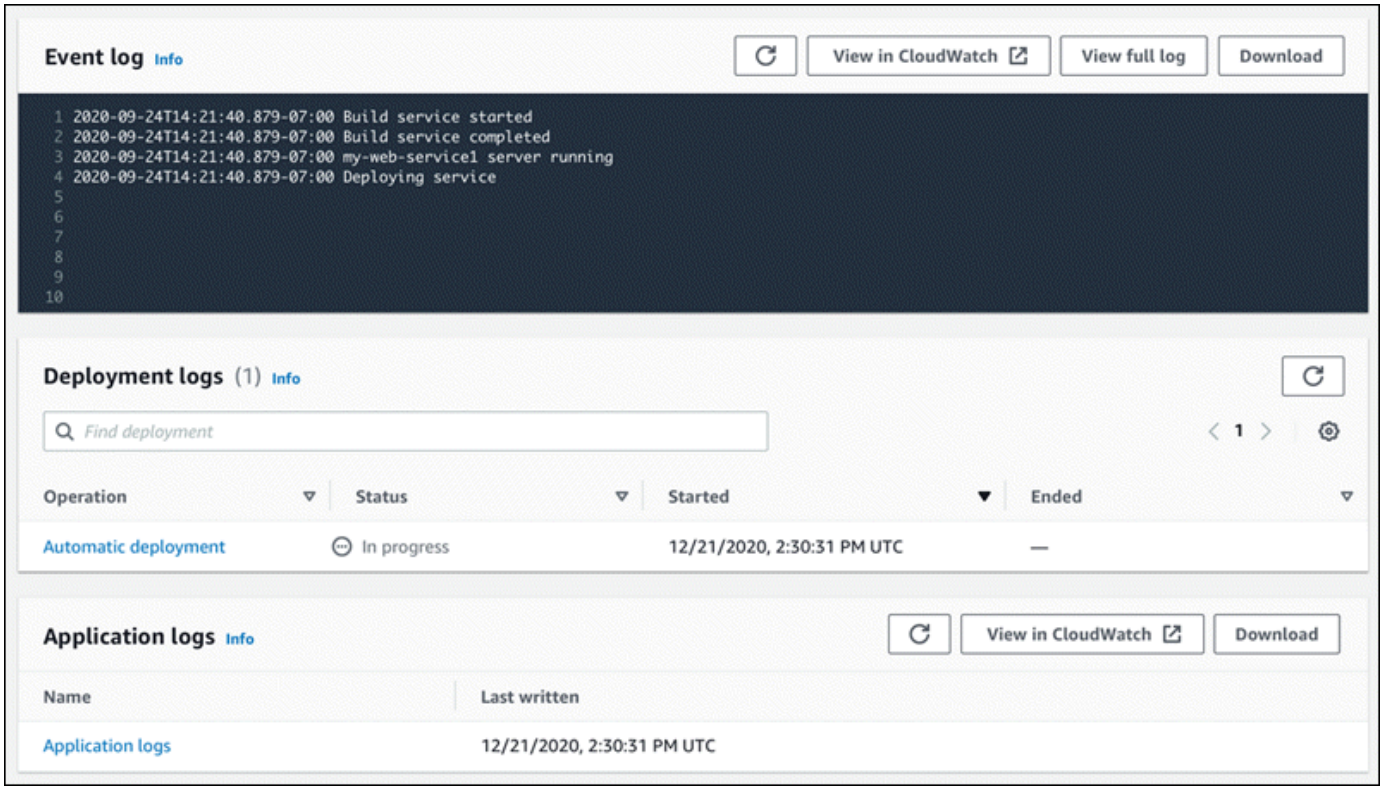
   主控台會顯示具有服務概觀的服務儀表板。



3. 在服務儀表板頁面上，選擇日誌索引標籤。

   主控台會在數個區段中顯示幾種類型的日誌：

   - 事件日誌 – App Runner 服務生命週期中的活動。主控台會顯示最新的事件。
   - 部署日誌 – 將來源儲存庫部署到您的 App Runner 服務。主控台會顯示每個部署的個別日誌串流。
   - 應用程式日誌 – 部署至 App Runner 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合併為單一日誌串流。



4. 若要尋找特定部署，請輸入搜尋詞彙，縮小部署日誌清單的範圍。您可以搜尋出現在資料表中的任何值。
5. 若要檢視日誌的內容，請選擇檢視完整日誌（事件日誌）或日誌串流名稱（部署和應用程式日誌）。
6. 選擇下載以下載日誌。對於部署日誌串流，請先選取日誌串流。
7. 選擇在 CloudWatch 中檢視以開啟 CloudWatch 主控台，並使用其完整功能來探索您的 App Runner 服務日誌。對於部署日誌串流，請先選取日誌串流。

> **Note**
>
> 如果您想要檢視特定執行個體的應用程式日誌，而非合併的應用程式日誌，CloudWatch 主控台特別有用。

檢視報告給 CloudWatch 的 App Runner 服務指標

Amazon CloudWatch AWS 會即時監控您的 Amazon Web Services (AWS) 資源和您在其上執行的應用程式。您可以使用 CloudWatch 收集和追蹤指標，這些是您可以為您的資源和應用程式測量的變數。

您也可以使用它來建立監看指標的警示。達到特定閾值時，CloudWatch 會傳送通知，或自動變更受監控的資源。如需更多資訊，請參閱 Amazon CloudWatch 使用者指南。

AWS App Runner 會收集各種指標，讓您更清楚 App Runner 服務的使用情況、效能和可用性。有些指標會追蹤執行 Web 服務的個別執行個體，有些則位於整體服務層級。下列各節列出 App Runner 指標，並說明如何在 App Runner 主控台中檢視它們。

App Runner 指標

App Runner 會收集與您的服務相關的下列指標，並將其發佈至 `AWS/AppRunner` 命名空間中的 CloudWatch。

> **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 單位和使用的記憶體數，而不是使用率百分比，如今天計算。如果您的應用程式在此日期之前在 App Runner 上執行，而且您選擇在 App Runner 或 CloudWatch 主控台上返回檢視此日期的指標，則您會看到兩個單位的指標顯示，因此也會看到一些異常。

> **Important**
>
> 您需要在 2023 年 8 月 23 日之前更新任何以 CPU 使用率和記憶體使用率指標值為基礎的 CloudWatch 警示。更新警示以根據百分比使用率觸發，而不是 vCPU 或 MB。如需更多資訊，請參閱 Amazon CloudWatch 使用者指南。

每個執行個體（擴展單位）都會個別收集執行個體層級指標。

測量了什麼？	指標	Description
CPU utilization	`CPUUtilization`	服務組態預留的 CPU 總用量中，一分鐘期間內的平均 CPU 用量百分比。
Memory utilization	`MemoryUtilization`	服務組態預留的總記憶體中一分鐘期間的平均記憶體用量百分比。

會針對整個服務收集服務層級指標。

測量了什麼？	指標	Description
CPU utilization	`CPUUtilization`	在一分鐘期間內所有執行個體的彙總 CPU 用量百分比，超出服務組態預留的總 CPU 用量。
Memory utilization	`MemoryUtilization`	服務組態預留的總記憶體中，一分鐘期間內所有執行個體的彙總記憶體用量百分比。
Concurrency	`Concurrency`	服務正在處理的並行請求的大致數量。
HTTP request count	`Requests`	服務收到的 HTTP 請求數量。

HTTP status counts	`2xxStatusResponses`
	`4xxStatusResponses`
	`5xxStatusResponses`
		傳回每個回應狀態的 HTTP 請求數目，依類別分組 (2XX、4XX, 5XX)。

HTTP request latency	`RequestLatency`	您的 Web 服務處理 HTTP 請求所需的時間，以毫秒為單位。
Instance counts	`ActiveInstances`	正在為您的服務處理 HTTP 請求的執行個體數量。 **Note** 如果`ActiveInstances` 指標顯示零，表示沒有服務的請求。它不表示您服務的執行個體數量為零。

在主控台中檢視 App Runner 指標

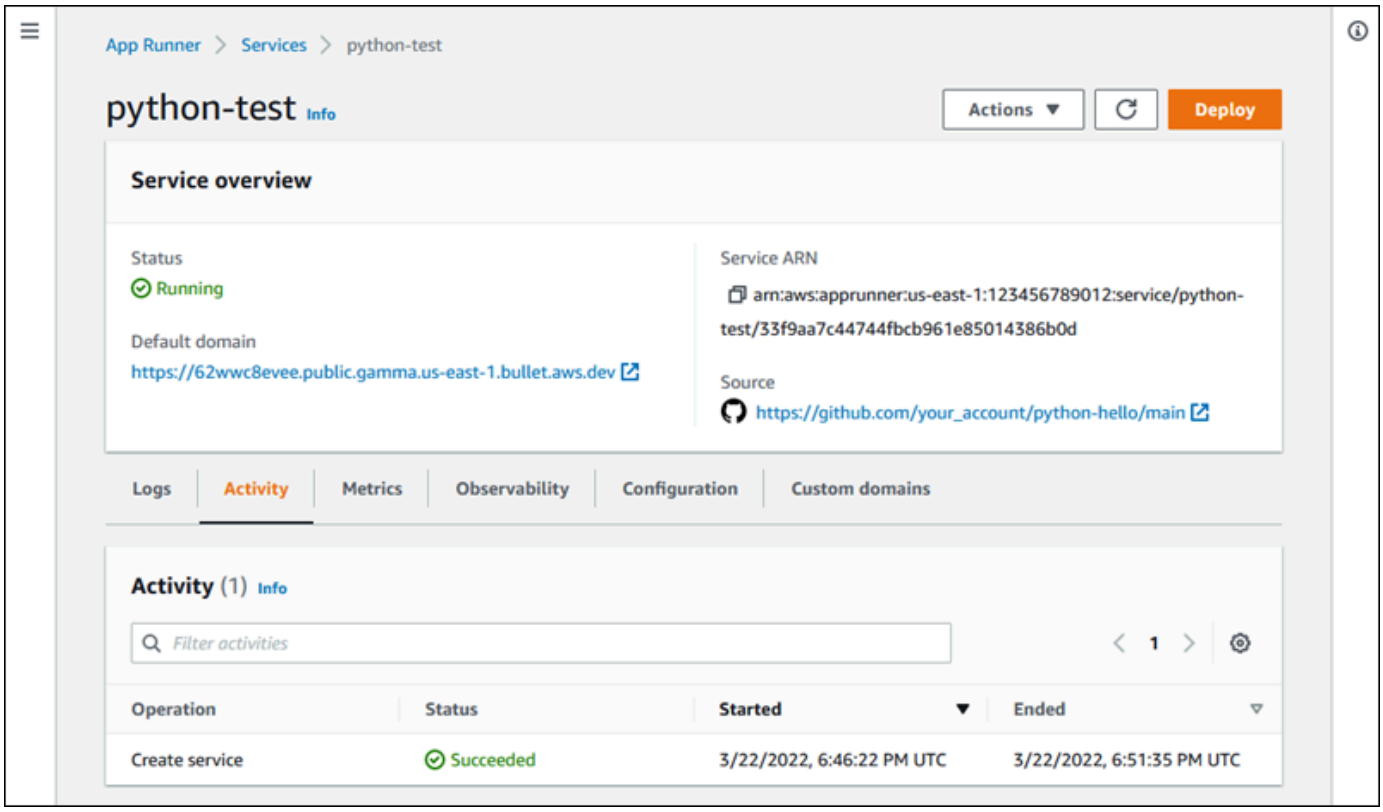
App Runner 主控台會以圖形顯示 App Runner 為您的服務收集的指標，並提供更多探索方法。

> **Note**
>
> 目前，主控台只會顯示服務指標。若要檢視執行個體指標，請使用 CloudWatch 主控台。

檢視 服務的日誌

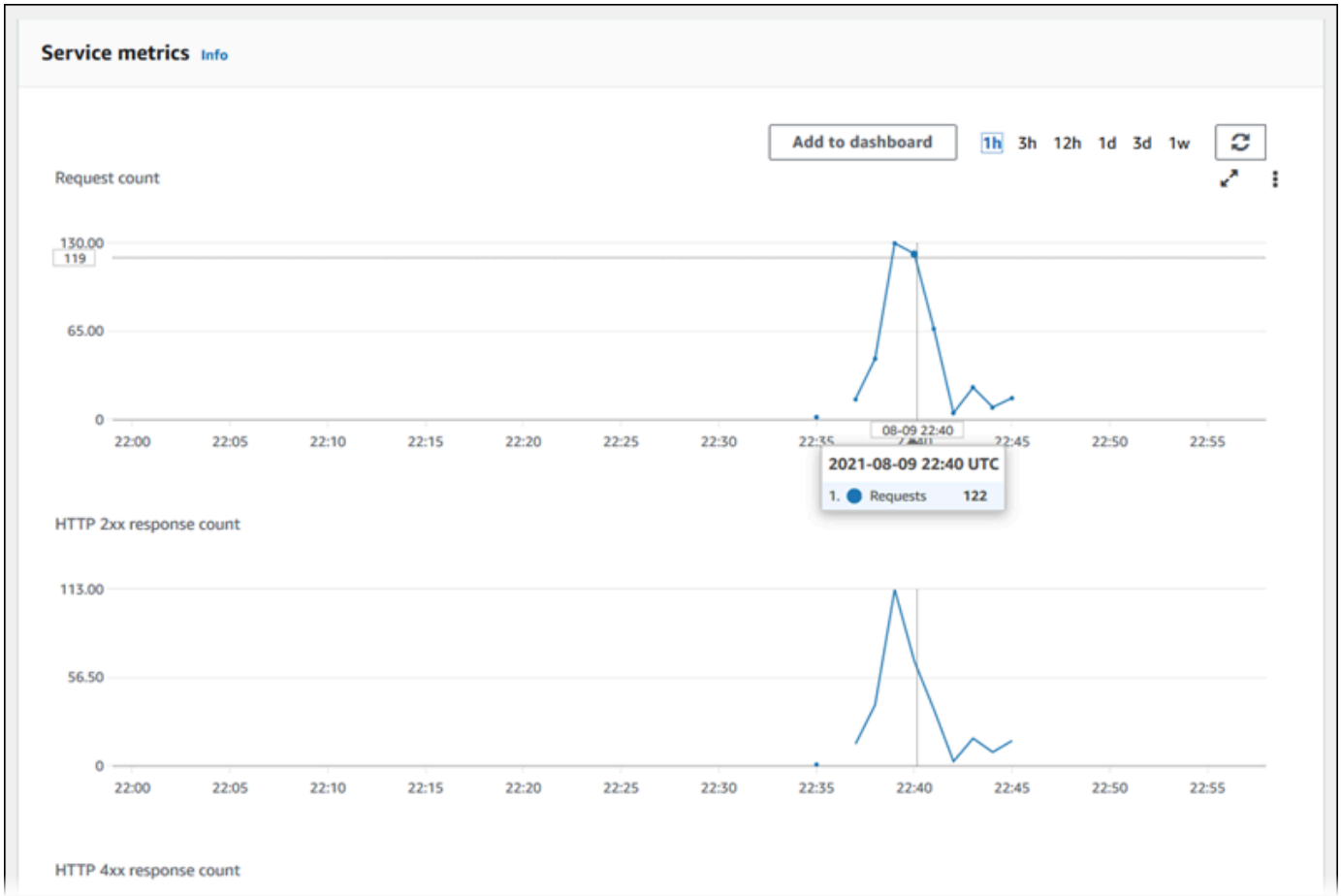
1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 在導覽窗格中，選擇服務，然後選擇您的 App Runner 服務。

   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。

   主控台會顯示一組指標圖表。



4. 選擇持續時間（例如 12 小時），將指標圖形範圍限定在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，將相關指標新增至 CloudWatch 主控台中的儀表板，以進行進一步調查。

在 EventBridge 中處理 App Runner 事件

使用 Amazon EventBridge，您可以設定事件驅動規則，以監控 AWS App Runner 服務中的即時資料串流是否有特定模式。當規則的模式相符時，EventBridge 會在目標中啟動動作 AWS Lambda，例如 AWS Batch Amazon ECS 和 Amazon SNS。例如，您可以在部署至服務失敗時發出 Amazon SNS 主題訊號，以設定傳送電子郵件通知的規則。或者，您可以設定 Lambda 函數，以便在服務更新失敗時通知 Slack 頻道。如需 EventBridge 的詳細資訊，請參閱 Amazon EventBridge 使用者指南。

App Runner 會將下列事件類型傳送至 EventBridge

- 服務狀態變更 – App Runner 服務的狀態變更。例如，服務狀態已變更為 `DELETE_FAILED`。
- 服務操作狀態變更 – App Runner 服務上長時間非同步操作的狀態變更。例如，服務開始建立、服務更新成功完成，或服務部署完成但發生錯誤。

建立 EventBridge 規則以對 App Runner 事件採取行動

EventBridge 事件是定義一些標準 EventBridge 欄位的物件，例如來源 AWS 服務和詳細資訊（事件）類型，以及具有事件詳細資訊的事件特定欄位集。若要建立 EventBridge 規則，您可以使用 EventBridge 主控台來定義事件模式（應該追蹤哪些事件），並指定目標動作（應該對配對執行的動作）。事件模式類似於其相符的事件。您可以指定要比對的欄位子集，並為每個欄位指定可能值的清單。本主題提供 App Runner 事件和事件模式的範例。

如需建立 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**
>
> 有些服務支援 EventBridge 中的預先定義模式。這簡化了事件模式的建立方式。您在表單上選取欄位值，EventBridge 就會為您產生模式。目前，App Runner 不支援預先定義的模式。您必須將模式輸入為 JSON 物件。您可以使用本主題中的範例做為起點。

App Runner 事件範例

這些是 App Runner 傳送至 EventBridge 之事件的一些範例。

- 服務狀態變更事件。具體而言，從 `OPERATION_IN_PROGRESS`變更為 `RUNNING` 狀態的服務。

```
{
  "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"
  }
}
```

- 操作狀態變更事件。具體而言，UpdateService操作已成功完成。

```
{
  "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"
  }
}
```

App Runner 事件模式範例

下列範例示範您可以在 EventBridge 規則中使用的事件模式，以符合一或多個 App Runner 事件。事件模式類似於事件。僅包含您要比對的欄位，並提供清單，而不是每個欄位的純量。

- 比對特定帳戶服務的所有服務狀態變更事件，其中服務不再處於 RUNNING 狀態。

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
```

```
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- 比對特定帳戶服務的所有操作狀態變更事件，其中操作失敗。

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```

App Runner 事件參考

服務狀態變更

服務狀態變更事件已`detail-type`設定為 `AppRunner Service Status Change`。它具有下列詳細資訊欄位和值：

```
"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

操作狀態變更事件已`detail-type`設定為 `AppRunner Service Operation Status Change`。它具有下列詳細資訊欄位和值：

```
"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。

狀態	訊息
CreateServiceStarted	服務建立已開始。
CreateServiceCompletedSuccessfully	服務建立已成功完成。
CreateServiceFailed	服務建立失敗。如需詳細資訊，請參閱 服務日誌。
DeleteServiceStarted	服務刪除已開始。
DeleteServiceCompletedSuccessfully	服務刪除已成功完成。
DeleteServiceFailed	服務刪除失敗。
UpdateServiceStarted	
UpdateServiceCompletedSuccessfully	服務更新已成功完成。已部署新的應用程式和組態。
	服務更新已成功完成。已部署新組態。
UpdateServiceFailed	服務更新失敗。如需詳細資訊，請參閱 服務日誌。
DeploymentStarted	部署已開始。
DeploymentCompletedSuccessfully	部署已成功完成。
DeploymentFailed	部署失敗。如需詳細資訊，請參閱 服務日誌。
PauseServiceStarted	服務暫停已啟動。

狀態	訊息
PauseServiceCompletedSuccessfully	服務暫停已成功完成。
PauseServiceFailed	服務暫停失敗。
ResumeServiceStarted	服務繼續已啟動。
ResumeServiceCompletedSuccessfully	服務繼續成功完成。
ResumeServiceFailed	服務繼續失敗。

使用 記錄 App Runner API 呼叫 AWS CloudTrail

App Runner 已與 整合 AWS CloudTrail，此服務提供由使用者、角色或 App Runner 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 App Runner 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 App Runner 主控台的呼叫，以及對 App Runner API 操作的程式碼呼叫。如果您建立線索，您可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 App Runner 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的事件歷史記錄檢視最新事件。使用 CloudTrail 收集的資訊，您可以判斷向 App Runner 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「AWS CloudTrail 使用者指南」。

CloudTrail 中的 App Runner 資訊

當您建立帳戶 AWS 帳戶 時，您的 上會啟用 CloudTrail。當活動在 App Runner 中發生時，該活動會與事件歷史記錄中的其他服務 AWS 事件一起記錄在 CloudTrail 事件中。您可以在 中檢視、搜尋和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html。

若要持續記錄 中的事件 AWS 帳戶，包括 App Runner 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 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。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有 App Runner 動作，並記錄在 AWS App Runner API 參考中。例如，對 `CreateService`、`DeleteConnection` 以及 `StartDeployment` 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

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 userIdentity 元素。

了解 App Runner 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。事件即為來自任何來源的單一請求，其中包含請求動作、動作日期和時間，以及請求參數的相關資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 `CreateService` 動作的 CloudTrail 日誌項目。

> **Note**
>
> 基於安全考量，某些屬性值會在日誌中修訂，並以文字 取代`HIDDEN_DUE_TO_SECURITY_REASONS`。這可防止意外暴露秘密資訊。不過，您仍然可以看到這些屬性在請求中傳遞或在回應中傳回。

`CreateService` App Runner 動作的範例 CloudTrail 日誌項目

```
{
  "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,
```

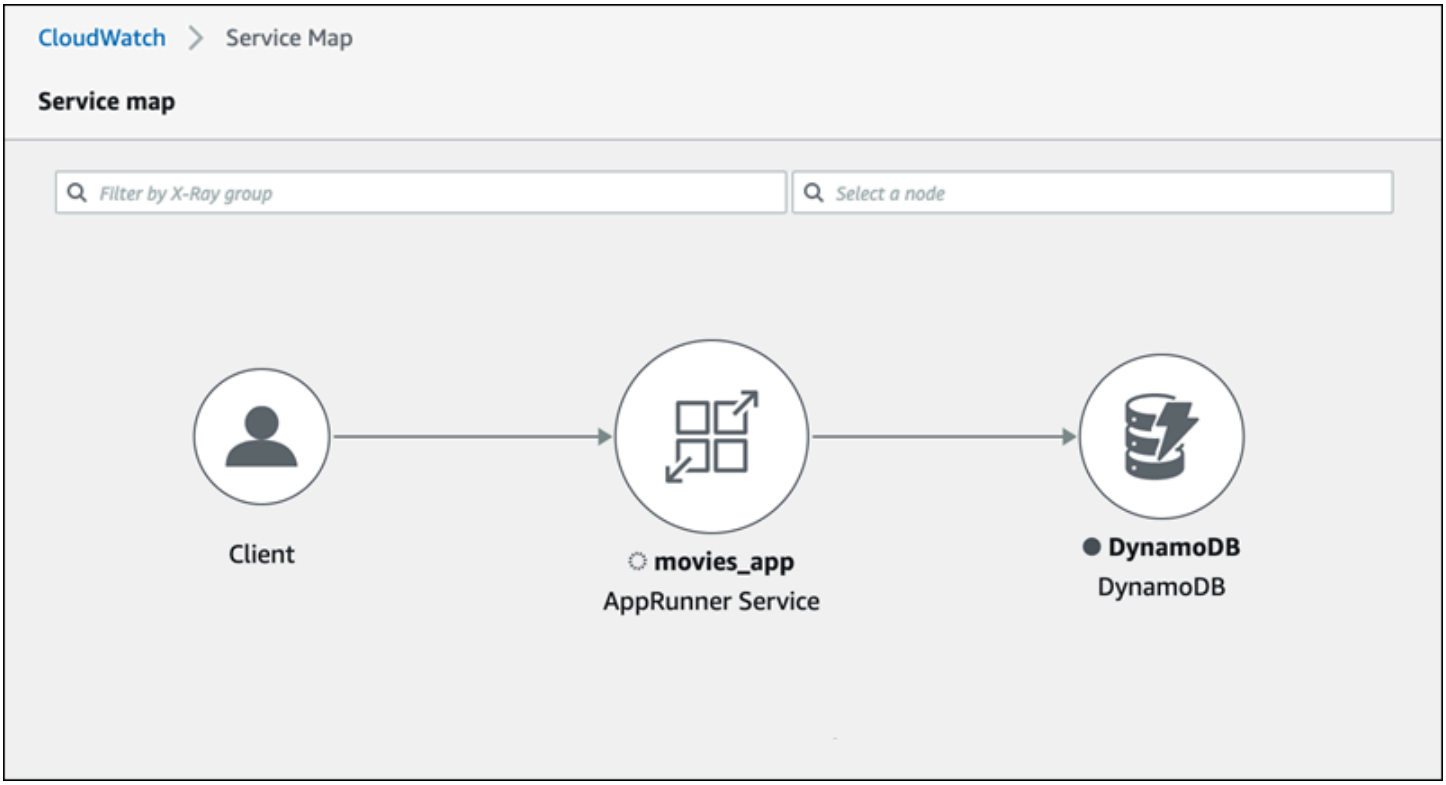
```
            "healthyThreshold": 3,
            "unhealthyThreshold": 5
        },
        "instanceConfiguration": {
            "cpu": "256",
            "memory": "1024"
        },
        "autoScalingConfigurationSummary": {
            "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
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 App Runner 應用程式

AWS X-Ray 是一項 服務，可收集應用程式提供的請求相關資料，並提供可用來檢視、篩選和深入了解該資料的工具，以識別問題和最佳化的機會。對於應用程式的任何追蹤請求，您不僅可以查看請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 APIs發出的呼叫的詳細資訊。

X-Ray 使用來自支援雲端應用程式 AWS 的資源的追蹤資料來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》。



檢測您的應用程式以進行追蹤

使用可攜式遙測規格 OpenTelemetry 檢測 App Runner 服務應用程式以進行追蹤。目前，App Runner 支援 AWS Distro for OpenTelemetry (ADOT)，這是使用 AWS 服務收集和呈現遙測資訊的 OpenTelemetry 實作。X-Ray 實作追蹤元件。

根據您在應用程式中使用的特定 ADOT 開發套件，ADOT 最多支援兩種檢測方法：自動和手動。如需使用 SDK 進行檢測的詳細資訊，請參閱 ADOT 文件，然後在導覽窗格中選擇您的 SDK。

執行期設定

以下是檢測 App Runner 服務應用程式以進行追蹤的一般執行期設定指示。

設定執行時間的追蹤

1. 遵循 AWS Distro for OpenTelemetry (ADOT) 中針對執行時間提供的指示來檢測您的應用程式。
2. 如果您使用的是原始碼儲存庫，請在 `apprunner.yaml` 檔案的 `build`區段中安裝必要的`OTEL`相依性；如果您使用的是容器映像，請在 Dockerfile 中安裝必要的相依性。
3. 如果您使用原始碼儲存庫，請在 `apprunner.yaml` 檔案中設定您的環境變數；如果您使用容器映像，請在 Dockerfile 中設定您的環境變數。

Example環境變數

> **Note**
>
> 下列範例列出要新增至 `apprunner.yaml` 檔案的重要環境變數。如果您使用容器映像，請將這些環境變數新增至 Dockerfile。不過，每個執行時間都可以有自己的特性，您可能需要將更多環境變數新增至下列清單。如需執行時間特定指示的詳細資訊，以及如何為執行時間設定應用程式的範例，請參閱入門下的 AWS Distro for OpenTelemetry 並前往執行時間。

```
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 App Runner 的重要環境變數，因為 App Runner Otel 收集器不接受指標記錄。它只接受指標追蹤。

執行時間設定範例

下列範例示範使用 ADOT Python SDK 自動檢測您的應用程式。軟體開發套件會自動產生遙測資料，說明應用程式中 Python 架構使用的值，而不會新增單行 Python 程式碼。您只需要在兩個來源檔案中新增或修改幾行。

首先，新增一些相依性，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，檢測您的應用程式。執行此作業的方式取決於您的服務來源 - 來源映像或來源碼。

Source image

當您的服務來源是映像時，您可以直接檢測控制建置容器映像並在映像中執行應用程式的 Dockerfile。下列範例顯示 Python 應用程式的檢測 Dockerfile。檢測新增項目以粗體強調。

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

當您的服務來源是包含應用程式來源的儲存庫時，您可以使用 App Runner 組態檔案設定間接檢測映像。這些設定控制 App Runner 產生的 Dockerfile，並用來建置應用程式的映像。下列範例顯示 Python 應用程式的檢測 App Runner 組態檔案。檢測新增項目以粗體強調。

Example apprunner.yaml

```
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 許可新增至您的 App Runner 服務執行個體角色

若要搭配 App Runner 服務使用 X-Ray 追蹤，您必須為服務的執行個體提供與 X-Ray 服務互動的許可。您可以建立執行個體角色與服務的關聯，並新增具有 X-Ray 許可的受管政策來執行此操作。如需 App Runner 執行個體角色的詳細資訊，請參閱 the section called “執行個體角色”。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 App Runner 服務啟用 X-Ray 追蹤

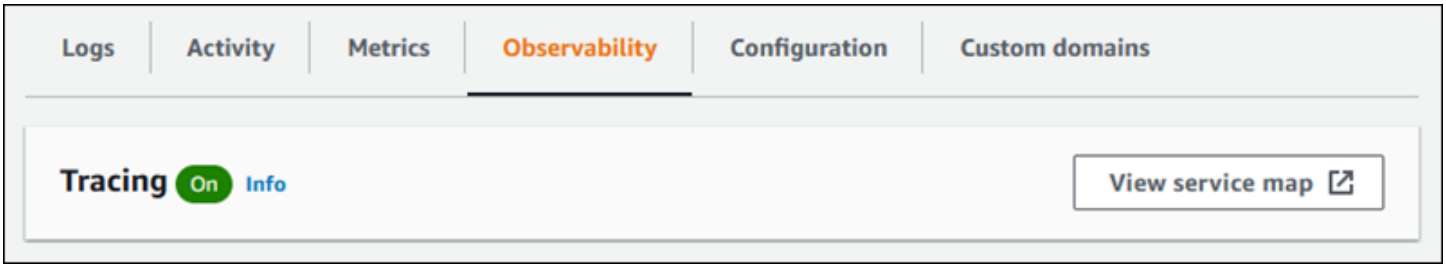
當您建立服務時，App Runner 預設會停用追蹤。您可以在設定可觀測性時為服務啟用 X-Ray 追蹤。如需詳細資訊，請參閱the section called “管理可觀測性”。

如果您使用 App Runner API 或 AWS CLI，則 ObservabilityConfiguration 資源物件中的 TraceConfiguration 物件會包含追蹤設定。 ObservabilityConfiguration 若要停用追蹤，請勿指定`TraceConfiguration`物件。

在主控台和 API 案例中，請務必將上一節討論的執行個體角色與 App Runner 服務建立關聯。

檢視 App Runner 服務的 X-Ray 追蹤資料

在 App Runner 主控台中服務儀表板頁面的可觀測性索引標籤上，選擇檢視服務映射以導覽至 Amazon CloudWatch 主控台。



使用 Amazon CloudWatch 主控台來檢視應用程式提供之請求的服務映射和追蹤。服務映射會顯示請求延遲以及與其他應用程式 AWS 和服務互動等資訊。您新增至程式碼的自訂註釋可讓您輕鬆搜尋追蹤。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 ServiceLens 監控應用程式的運作狀態。

將 AWS WAF Web ACL 與您的服務建立關聯

AWS WAF 是一種 Web 應用程式防火牆，可用來保護 App Runner 服務。透過 AWS WAF Web 存取控制清單 (Web ACLs)，您可以保護 App Runner 服務端點，防範常見的 Web 入侵和不需要的機器人。

Web ACL 可讓您對 App Runner 服務的所有傳入 Web 請求進行精細控制。您可以在 Web ACL 中定義規則，以允許、封鎖或監控 Web 流量，以確保只有授權和合法的請求才能到達您的 Web 應用程式和 APIs。您可以根據您的特定業務和安全性需求自訂 Web ACL 規則。若要進一步了解套用網路 ACLs 的基礎設施安全性和最佳實務，請參閱《Amazon VPC 使用者指南》中的控制網路流量。

> **Important**
>
> 與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。

傳入 Web 請求流程

當 AWS WAF Web ACL 與 App Runner 服務相關聯時，傳入的 Web 請求會經歷下列程序：

1. App Runner 會將原始伺服器請求的內容轉送至 AWS WAF。
2. AWS WAF 會檢查請求，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， 會 AWS WAF 傳回 `allow`或 `block` 回應給 App Runner。
   - 如果傳回`allow`回應，App Runner 會將請求轉送到您的應用程式。
   - 如果傳回`block`回應，App Runner 會封鎖請求到達您的 Web 應用程式。它會將`block`回應從 轉送 AWS WAF 到您的應用程式。

   > **Note**
   >
   > 如果未傳回任何回應，App Runner 預設會封鎖請求 AWS WAF。

如需 AWS WAF Web ACLs，請參閱《 AWS WAF 開發人員指南》中的 Web 存取控制清單 (Web ACLs)。

> **Note**
>
> 您支付標準 AWS WAF 定價。針對 App Runner 服務使用 AWS WAF Web ACLs 不會產生任何額外費用。
> 如需 定價的詳細資訊，請參閱 AWS WAF 定價。

將 WAF Web ACLs 與您的 App Runner 服務建立關聯

以下是將 AWS WAF Web ACL 與您的 App Runner 服務建立關聯的高階程序：

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 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 App Runner 服務建立關聯：
   - App Runner 主控台：建立或更新 App Runner 服務時，使用 App Runner 主控台關聯現有的 Web ACL。如需說明，請參閱管理 AWS WAF Web ACLs。
   - AWS WAF 主控台：使用現有 App Runner 服務的 AWS WAF 主控台來關聯 Web ACL。如需詳細資訊，請參閱《 開發人員指南》中的將 Web ACL 與 AWS 資源建立關聯或取消關聯。 AWS WAF
   - AWS CLI：使用 AWS WAF 公有 APIs 關聯 Web ACL。如需 AWS WAF 公有 APIs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 ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。
- App Runner 服務只能與一個 Web ACL 建立關聯。不過，您可以將一個 Web ACL 與多個 App Runner 服務和多個 AWS 資源建立關聯。範例包括 Amazon Cognito 使用者集區和 Application Load Balancer 資源。

- 當您建立 Web ACL 時，在 Web ACL 完全傳播並可供 App Runner 使用之前，會經過一小段時間。傳播時間可以是幾秒鐘到幾分鐘。當您嘗試在 Web ACL 完全傳播之前建立關聯`WAFUnavailableEntityException`時， 會 AWS WAF 傳回 。

  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，則關聯會失敗。不過，您可以在 App Runner 主控台內導覽。
- AWS WAF 當您為處於無效狀態的 App Runner 服務呼叫下列其中一個 AWS WAF APIs 時， 會傳回`WAFNonexistentItemException`錯誤：
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  App Runner 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 App Runner 服務遭到刪除時，狀態才會無效。
- 您的請求可能會導致承載大於 AWS WAF 可檢查項目的限制。如需有關 如何處理來自 App Runner AWS WAF 的過大請求的詳細資訊，請參閱《 AWS WAF 開發人員指南》中的過大請求元件處理，以了解 如何處理來自 App Runner AWS WAF 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

若要在 中使用 Web ACL AWS App Runner，請新增下列 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 的更新 IAM 政策範例 AWS WAF。此 IAM 政策包含使用 App Runner 服務的必要許可。

Example

JSON

```
{
   "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 Web ACLs

使用下列其中一種方法來管理 App Runner 服務的 AWS WAF Web ACLs：

- the section called “App Runner 主控台”
- the section called “AWS CLI”

App Runner 主控台

當您在 App Runner 主控台上建立服務或更新現有的服務時，您可以關聯或取消與 AWS WAF Web ACL 的關聯。

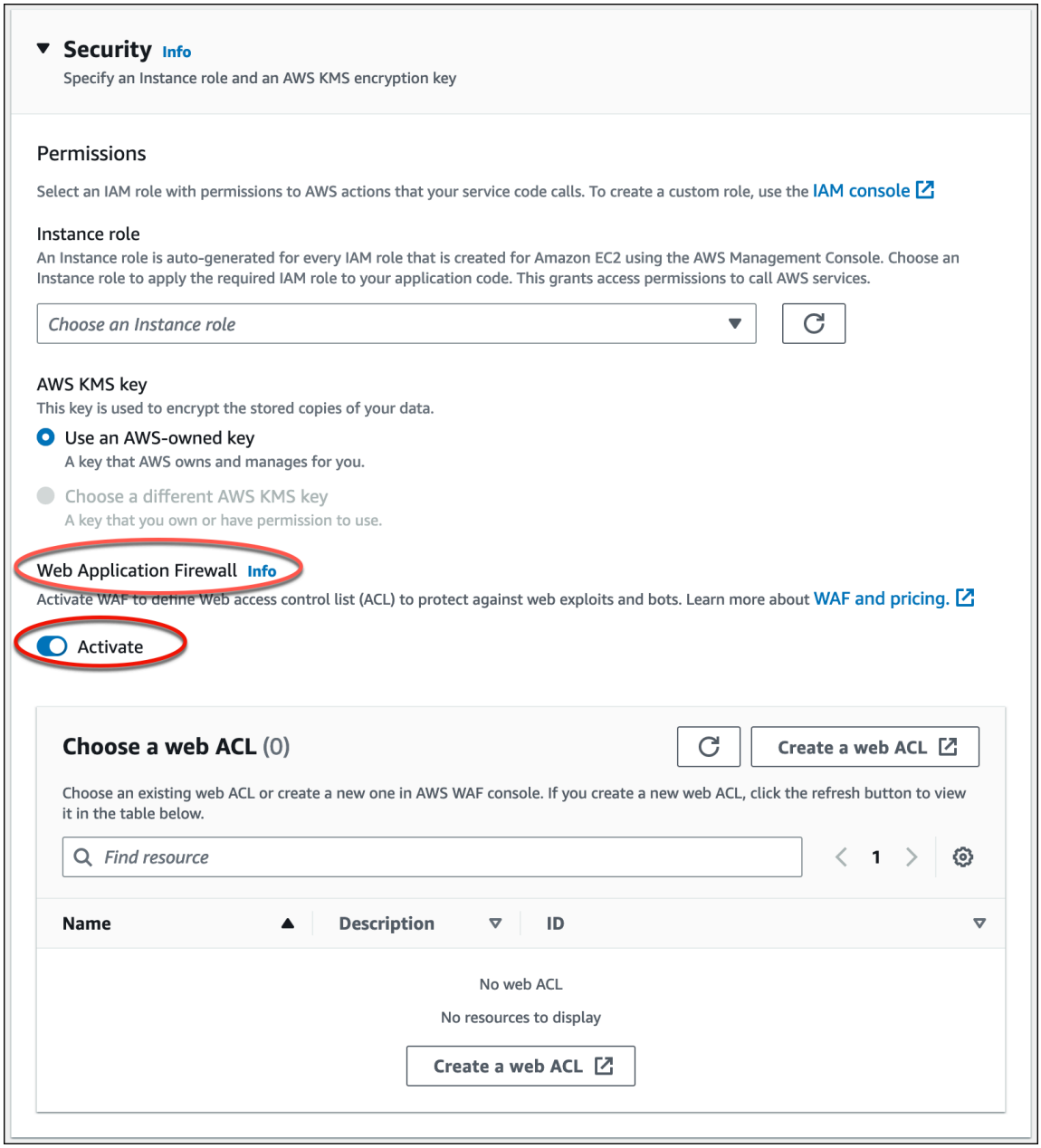
> **Note**
>
> - App Runner 服務只能與一個 Web ACL 建立關聯。不過，除了其他 AWS 資源之外，您還可以將一個 Web ACL 與多個 App Runner 服務建立關聯。
> - 建立 Web ACL 關聯之前，請務必更新 的 IAM 許可 AWS WAF。如需詳細資訊，請參閱 許可。

關聯 AWS WAF Web ACL

> **Important**
>
> 與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。

建立 AWS WAF Web ACL 的關聯

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 根據您是要建立或更新服務，請執行下列其中一個步驟：
   - 如果您要建立新的服務，請選擇建立 App Runner 服務，然後前往設定服務。
   - 如果您要更新現有的服務，請選擇組態索引標籤，然後選擇設定服務下的編輯。
3. 前往安全下的 Web 應用程式防火牆。
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：從選擇要與您的 App Runner 服務建立關聯的 Web ACL 資料表中選擇所需的 Web ACL。
- 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，即可在選擇 Web ACL 資料表中檢視新建立的 Web ACL。
  2. 選取所需的 Web ACL。

6. 如果您要建立新服務，請選擇下一步，如果您要更新現有服務，請選擇儲存變更。選取的 Web ACL 與您的 App Runner 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。捲動至安全下的 Web 應用程式防火牆，以檢視與您的服務相關聯的 Web ACL 詳細資訊。

> **Note**
>
> 當您建立 Web ACL 時，在 Web ACL 完全傳播並可供 App Runner 使用之前，會經過一小段時間。傳播時間可以是幾秒鐘到幾分鐘。當您嘗試在 Web ACL 完全傳播之前建立關聯`WAFUnavailableEntityException`時， 會 AWS WAF 傳回 。
> 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，則關聯會失敗。不過，您可以在 App Runner 主控台內導覽。

取消與 AWS WAF Web ACL 的關聯

您可以更新 App Runner 服務，取消不再需要的 AWS WAF Web ACl 關聯。

取消與 AWS WAF Web ACl 的關聯

1. 開啟 App Runner 主控台，然後在區域清單中選取您的 AWS 區域。
2. 前往您要更新之服務的組態索引標籤，然後在設定服務下選擇編輯。
3. 前往安全下的 Web 應用程式防火牆。
4. 停用啟用切換按鈕。您會收到確認刪除的訊息。
5. 選擇確認。Web ACL 與您的 App Runner 服務取消關聯。

> **Note**
>
> - 如果您想要將服務與其他 Web ACL 建立關聯，請從選擇 Web ACL 資料表中選取 Web ACL。App Runner 會取消目前 Web ACL 的關聯，並啟動與所選 Web ACL 建立關聯的程序。
> - 如果沒有其他 App Runner 服務或資源使用取消關聯的 Web ACL，請考慮刪除 Web ACL。否則，您將繼續產生成本。如需定價的詳細資訊，請參閱AWS WAF 定價。如需有關如何刪除 Web ACL 的說明，請參閱 AWS WAF API 參考中的 DeleteWebACL。
> - 您無法刪除與其他作用中 App Runner 服務或其他資源相關聯的 Web ACL。

AWS CLI

您可以使用 AWS WAF 公有 APIs 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 App Runner 服務必須處於有效狀態。

AWS WAF 當您呼叫下列其中一個 AWS WAF APIs來取得處於無效狀態的 App Runner 服務時， 會傳回`WAFNonexistentItemException`錯誤：

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

App Runner 服務的無效狀態包括：

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS` 只有在刪除您的 App Runner 服務時，狀態才會無效。

如需 AWS WAF 公有 APIs的詳細資訊，請參閱 AWS WAF API 參考指南。

> **Note**
>
> 更新 的 IAM 許可 AWS WAF。如需詳細資訊，請參閱 許可。

使用 建立 AWS WAF Web ACL 的關聯 AWS CLI

> **Important**
>
> 與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服

務。如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。

建立 AWS WAF Web ACL 的關聯

1. 使用您偏好的規則動作集來建立服務的 AWS WAF Web ACL`Block`，`Allow`或為您的服務建立 Web 請求。如需 AWS WAF APIs的詳細資訊，請參閱 AWS WAF API 參考指南中的 CreateWebACL。

   Example建立 Web 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 公有 API，將您建立的 Web ACL 與 App Runner 服務建立關聯。如需 AWS WAF APIs的詳細資訊，請參閱 AWS WAF API 參考指南中的 AssociateWebACL。

   > **Note**
   >
   > 當您建立 Web ACL 時，在 Web ACL 完全傳播並可供 App Runner 使用之前，會經過一小段時間。傳播時間可以是幾秒鐘到幾分鐘。當您嘗試在 Web ACL 完全傳播之前建立關聯`WAFUnavailableEntityException`時， 會 AWS WAF 傳回 。
   > 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，則關聯會失敗。不過，您可以在 App Runner 主控台內導覽。

   Example關聯 Web ACL - 請求

   ```
   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 公有 API 確認 Web ACL 與您的 App Runner 服務相關聯。如需 AWS WAF APIs的詳細資訊，請參閱 API AWS WAF 參考指南中的 GetWebACLForResource。

   Example驗證資源的 Web ACL - 請求

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   如果沒有與您服務相關聯的 Web ACLs，您會收到空白回應。

使用 刪除 AWS WAF Web ACL AWS CLI

如果 AWS WAF Web ACL 與 App Runner 服務相關聯，則無法刪除。

刪除 AWS WAF Web ACL

1. 使用`disassociate-web-acl` AWS WAF 公有 API 取消 Web ACL 與 App Runner 服務的關聯。如需 AWS WAF APIs的詳細資訊，請參閱 API AWS WAF 參考指南中的 DisassociateWebACL。

   Example取消與 Web ACL 的關聯 - 請求

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. 使用`get-web-acl-for-resource` AWS WAF 公有 API 確認 Web ACL 已與您的 App Runner 服務取消關聯。

   Example確認 Web ACL 已解除關聯 - 請求

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

取消關聯的 Web ACL 不會列出您的 App Runner 服務。如果沒有與您服務相關聯的 Web ACLs，您會收到空白回應。

3. 使用`delete-web-acl` AWS WAF 公有 API 刪除取消關聯的 Web ACL。如需 AWS WAF APIs的詳細資訊，請參閱 API AWS WAF 參考指南中的 DeleteWebACL。

   Example刪除 Web 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 Web ACL 已使用`list-web-acl` AWS WAF 公有 API 刪除。如需 AWS WAF APIs的詳細資訊，請參閱 AWS WAF API 參考指南中的 ListWebACLs。

   Example確認 Web ACL 已刪除 - 請求

```
aws wafv2 list-web-acls
--scope REGIONAL
--region <region>
```

   刪除的 Web ACL 不會再列出。

> **Note**
>
> 如果 Web ACL 與其他作用中的 App Runner 服務或其他資源相關聯，例如 Amazon Cognito 使用者集區，則無法刪除 Web ACL。

列出與 Web ACL 相關聯的 App Runner 服務

Web ACL 可以與多個 App Runner 服務和其他資源建立關聯。使用`list-resources-for-web-acl` AWS WAF 公有 API 列出與 Web ACL 相關聯的 App Runner 服務。如需 AWS WAF APIs的詳細資訊，請參閱 AWS WAF API 參考指南中的 ListResourcesForWebACL。

Example列出與 Web ACL 相關聯的 App Runner 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 相關聯的 App Runner 服務 - 回應

下列範例說明當沒有 App Runner 服務與 Web ACL 相關聯時的回應。

```
{
  "ResourceArns": []
}
```

Example列出與 Web ACL 相關聯的 App Runner 服務 - 回應

下列範例說明當有 App Runner 服務與 Web ACL 相關聯時的回應。

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

測試和記錄 AWS WAF Web ACLs

當您將規則動作設定為 Web ACL 中的計數時， 會將請求 AWS WAF 新增至符合規則的請求計數。若要使用 App Runner 服務測試 Web ACL，請將規則動作設定為計數，並考慮符合每個規則的請求量。例如，您可以為符合大量請求`Block`的動作設定規則，而這些請求是您確定為正常使用者流量。在這種情況下，您可能需要重新設定規則。如需詳細資訊，請參閱《 AWS WAF 開發人員指南》中的測試和調校您的 AWS WAF 保護。

您也可以設定 AWS WAF 將請求標頭記錄到 Amazon CloudWatch Logs 日誌群組、Amazon Simple Storage Service (Amazon S3) 儲存貯體或 Amazon Data 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 App Runner 服務相關聯的 Web ACL 相關日誌，請參閱下列日誌欄位：

- `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》中的日誌範例。

> **Important**
>
> 與 WAF Web ACLs 相關聯的 App Runner 私有服務的來源 IP 規則不遵守以 IP 為基礎的規則。這是因為我們目前不支援將請求來源 IP 資料轉送至與 WAF 相關聯的 App Runner 私有服務。如果您的 App Runner 應用程式需要來源 IP/CIDR 傳入流量控制規則，您必須對私有端點使用安全群組規則，而不是 WAF Web ACLs。

使用組態檔案設定 App Runner 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。您無法搭配映像型服務使用組態檔案。

當您使用原始碼儲存庫建立 AWS App Runner 服務時， AWS App Runner 需要有關建置和啟動服務的資訊。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，您可以使用組態檔案來設定服務選項。您在檔案中指定的選項會成為來源儲存庫的一部分，而這些選項的任何變更都會受到追蹤，類似於追蹤原始碼變更的方式。您可以使用 App Runner 組態檔案來指定比 API 支援更多的選項。如果您只需要 API 支援的基本選項，則不需要提供組態檔案。

App Runner 組態檔案是在應用程式儲存庫`apprunner.yaml`來源目錄中命名的 YAML 檔案。它為您的服務提供建置和執行時間選項。此檔案中的值會指示 App Runner 如何建置和啟動您的服務，並提供執行時間內容，例如網路設定和環境變數。

App Runner 組態檔案不包含操作設定，例如 CPU 和記憶體。

如需 App Runner 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

- App Runner 組態檔案範例
- App Runner 組態檔案參考

App Runner 組態檔案範例

> **Note**
>
> 組態檔案僅適用於以原始程式碼為基礎的服務。您無法搭配映像型服務使用組態檔案。

下列範例示範 AWS App Runner 組態檔案。有些是最小的，且僅包含必要的設定。其他則已完成，包括所有組態檔案區段。如需 App Runner 組態檔案的概觀，請參閱 App Runner 組態檔案。

組態檔案範例

最小組態檔案

透過最少的組態檔案，App Runner 會做出下列假設：

- 在建置或執行期間，不需要自訂環境變數。
- 使用最新的執行時間版本。
- 使用預設連接埠號碼和連接埠環境變數。

Example apprunner.yaml

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

完成組態檔案

此範例顯示使用具有受管執行時間的`apprunner.yaml`原始格式的所有組態金鑰。

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-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"
```

完整組態檔案 — （使用修訂的組建）

此範例顯示使用 中所有組態金鑰`apprunner.yaml`搭配受管執行時間。

只有修訂後的 App Runner 組建才支援 `pre-run` 參數。如果您的應用程式使用原始 App Runner 組建支援的執行時間版本，請勿將此參數插入您的組態檔案中。如需詳細資訊，請參閱受管執行期版本和 App Runner 組建。

> **Note**
>
> 由於此範例適用於 Python 3.11，因此我們使用 `pip3`和 `python3`命令。如需詳細資訊，請參閱 Python 平台主題特定執行時間版本的呼叫中的 。

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
```

```
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-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"
```

如需特定受管執行期組態檔案的範例，請參閱 下的特定執行期子主題程式碼型服務。

App Runner 組態檔案參考

> **Note**
>
> 組態檔案僅適用於以原始程式碼為基礎的服務。您無法搭配映像型服務使用組態檔案。

本主題是 AWS App Runner 組態檔案語法和語意的完整參考指南。如需 App Runner 組態檔案的概觀，請參閱 App Runner 組態檔案。

App Runner 組態檔案是 YAML 檔案。將其命名為 `apprunner.yaml`，並將其放置在應用程式儲存庫的來源目錄中。

結構概觀

App Runner 組態檔案是 YAML 檔案。將其命名為 `apprunner.yaml`，並將其放置在應用程式儲存庫的來源目錄中。

App Runner 組態檔案包含下列主要部分：

- 頂端區段 – 包含頂層金鑰
- 建置區段 – 設定建置階段
- 執行區段 – 設定執行階段

頂端區段

檔案頂端的金鑰提供有關檔案和服務執行時間的一般資訊。可使用下列金鑰：

- `version` – 必要。App Runner 組態檔案版本。理想情況下，請使用最新版本。

  語法

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime` – 必要。您的應用程式使用的執行時間名稱。若要了解 App Runner 提供的不同程式設計平台的可用執行時間，請參閱 程式碼型服務。

  > **Note**
  >
  > 受管執行時間的命名慣例為 *<language-name><major-version>*。

語法

```
runtime: runtime-name
```

Example

```
runtime: python3
```

組建區段

建置區段會設定 App Runner 服務部署的建置階段。您可以指定建置命令和環境變數。組建命令是必要的。

區段以 `build:`金鑰開頭，並具有下列子金鑰：

- `commands` – 必要。指定 App Runner 在各種建置階段執行的命令。包括下列子索引鍵：
  - `pre-build` – 選用。App Runner 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。App Runner 執行以建置應用程式的命令。例如，使用 pipenv。
  - `post-build` – 選用。App Runner 在建置之後執行的命令。例如，使用 Maven 將建置成品封裝到 JAR 或 WAR 檔案中，或執行測試。

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 – 選用。指定建置階段的自訂環境變數。定義為名稱值純量映射。您可以在建置命令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。一組位於建置區段，另一組位於執行區段。
>
> - 在建置過程中， pre-build、post-build、 build和 pre-run命令可以參考建置區段中的env集合。
>
> 重要 - 請注意，pre-run命令位於此檔案的執行區段中，即使它們只能存取組建區段中定義的環境變數。
>
> - 執行階段環境中的 run命令可以參考執行階段區段中的 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

執行區段會設定 App Runner 應用程式部署的容器執行階段。您可以指定執行時間版本、執行前命令（僅限修訂版格式）、啟動命令、網路連接埠和環境變數。

區段以 `run:`金鑰開頭，並具有下列子金鑰：

- `runtime-version` – 選用。指定您要為 App Runner 服務鎖定的執行時間版本。

  根據預設，只會鎖定主要版本。App Runner 會在每次部署或服務更新時，使用可供執行時間使用的最新次要和修補程式版本。如果您指定主要和次要版本，這兩個版本都會遭到鎖定，而 App Runner 只會更新修補程式版本。如果您指定主要、次要和修補程式版本，您的服務會鎖定在特定執行時間版本，而且 App Runner 永遠不會更新它。

  語法

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  > 某些平台的執行時間具有不同的版本元件。如需詳細資訊，請參閱特定平台主題。

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run` – 選用。 僅修訂組建用量。指定 App Runner 在將應用程式從建置映像複製到執行映像之後執行的命令。您可以在此處輸入命令，以修改 `/app`目錄外的執行映像。例如，如果您需要安裝

位於 /app目錄以外的其他全域相依性，請在此子區段中輸入所需的命令來執行此操作。如需 App Runner 建置程序的詳細資訊，請參閱 受管執行期版本和 App Runner 組建。

> **Note**
>
> - 重要 – 即使pre-run命令列在執行區段中，它們只能參考此組態檔案的建置區段中定義的環境變數。他們無法參考此執行區段中定義的環境變數。
> - 只有修訂後的 App Runner 組建才支援 pre-run 參數。如果您的應用程式使用原始 App Runner 組建支援的執行時間版本，請勿將此參數插入您的組態檔案中。如需詳細資訊，請參閱受管執行期版本和 App Runner 組建。

語法

```
run:
  pre-run:
      - command
      - …
```

- command – 必要。App Runner 在完成應用程式建置後用來執行應用程式的命令。

語法

```
run:
  command: command
```

- network – 選用。指定應用程式接聽的連接埠。其包括以下內容：
  - port – 選用。如果指定，這是您的應用程式接聽的連接埠號碼。預設值為 8080。
  - env – 選用。如果指定，App Runner 會將連接埠號碼傳遞至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 – 選用。執行階段的自訂環境變數定義。定義為名稱值純量映射。您可以在執行時間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。一組位於建置區段，另一組位於執行區段。
>
> - 在建置過程中， pre-build、post-build、 build和 pre-run命令可以參考建置區段中的env集合。
>
> 重要 - 請注意，pre-run命令位於此檔案的執行區段中，即使它們只能存取組建區段中定義的環境變數。
>
> - 執行階段環境中的 run命令可以參考執行階段區段中的 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"
```

App Runner API

AWS App Runner 應用程式程式設計界面 (API) 是一種 RESTful API，用於向 App Runner 服務提出請求。您可以使用 API 在 中建立、列出、描述、更新和刪除 App Runner 資源 AWS 帳戶。

您可以直接在應用程式碼中呼叫 API，也可以使用其中一個 AWS SDKs。

如需完整的 API 參考資訊，請參閱 AWS App Runner API 參考。

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 AWS CLI 搭配 App Runner
- 使用 AWS CloudShell 來使用 AWS App Runner

使用 AWS CLI 搭配 App Runner

對於命令列指令碼，請使用 AWS CLI 呼叫 App Runner 服務。如需完整的 AWS CLI 參考資訊，請參閱《 AWS CLI 命令參考》中的 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 管理主控台 。除了避免安裝之外，您不需要設定登入資料，也不需要指定區域。您的 AWS 管理主控台 工作階段會將此內容提供給 AWS CLI。如需 CloudShell 的詳細資訊，以及使用範例，請參閱 the section called “使用 AWS CloudShell”。

使用 AWS CloudShell 來使用 AWS App Runner

AWS CloudShell 是以瀏覽器為基礎的預先驗證 Shell，您可以直接從 啟動 AWS 管理主控台。您可以使用您偏好的 shell (Bash、PowerShell 或 Z shell AWS App Runner) 對 AWS 服務 （ 包括 ） 執行 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。

您可以從 AWS CloudShell 啟動 AWS 管理主控台，而且您用來登入主控台的 AWS 登入資料會自動在新的 shell 工作階段中使用。此預先驗證 AWS CloudShell 使用者可讓您在使用第 2 AWS CLI 版 （ 預先安裝在 Shell 的運算環境中 ） 與 App Runner 等 AWS 服務互動時，略過設定登入資料。

主題

- 取得 的 IAM 許可 AWS CloudShell
- 使用 與 App Runner 互動 AWS CloudShell

- 使用 驗證您的 App Runner 服務 AWS CloudShell

取得 的 IAM 許可 AWS CloudShell

AWS Identity and Access Management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，讓他們可以存取 AWS CloudShell 和使用環境的功能。

管理員授予使用者存取權的最快速方式是透過 AWS 受管政策。AWS 受管政策是由 AWS建立並管理的獨立政策。下列適用於 CloudShell 的 AWS 受管政策可以連接到 IAM 身分：

- `AWSCloudShellFullAccess`：授予許可，以 AWS CloudShell 使用 並完整存取所有功能。

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。如需限制 CloudShell 中使用者可用的動作的詳細資訊，請參閱AWS CloudShell 《 使用者指南》中的使用 IAM 政策管理 AWS CloudShell 存取和用量。

> **Note**
>
> 您的 IAM 身分也需要政策，授予對 App Runner 進行呼叫的許可。如需詳細資訊，請參閱the section called “App Runner 和 IAM”。

使用 與 App Runner 互動 AWS CloudShell

AWS CloudShell 從 啟動後 AWS 管理主控台，您可以立即開始使用命令列界面與 App Runner 互動。

在下列範例中，您會使用 CloudShell AWS CLI 中的 擷取其中一個 App Runner 服務的相關資訊。

> **Note**
>
> 在 AWS CLI 中使用 時 AWS CloudShell，您不需要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 擷取 App Runner 服務資訊 AWS CloudShell

1. 從 中 AWS 管理主控台，您可以選擇導覽列上可用的下列選項來啟動 CloudShell：

- 選擇 CloudShell 圖示。
- 開始**cloudshell**在搜尋方塊中輸入 ，然後在搜尋結果中看到 CloudShell 選項時選擇該選項。

2. 若要在主控台工作階段的 AWS 區域中列出您 AWS 帳戶中所有目前的 App Runner 服務，請在 CloudShell 命令列中輸入下列命令：

```
$ 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，請使用上一個步驟中擷取的其中一個 ARNs，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
```

輸出會列出您指定服務的詳細描述。

```
{
  "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
    }
  }
}
```

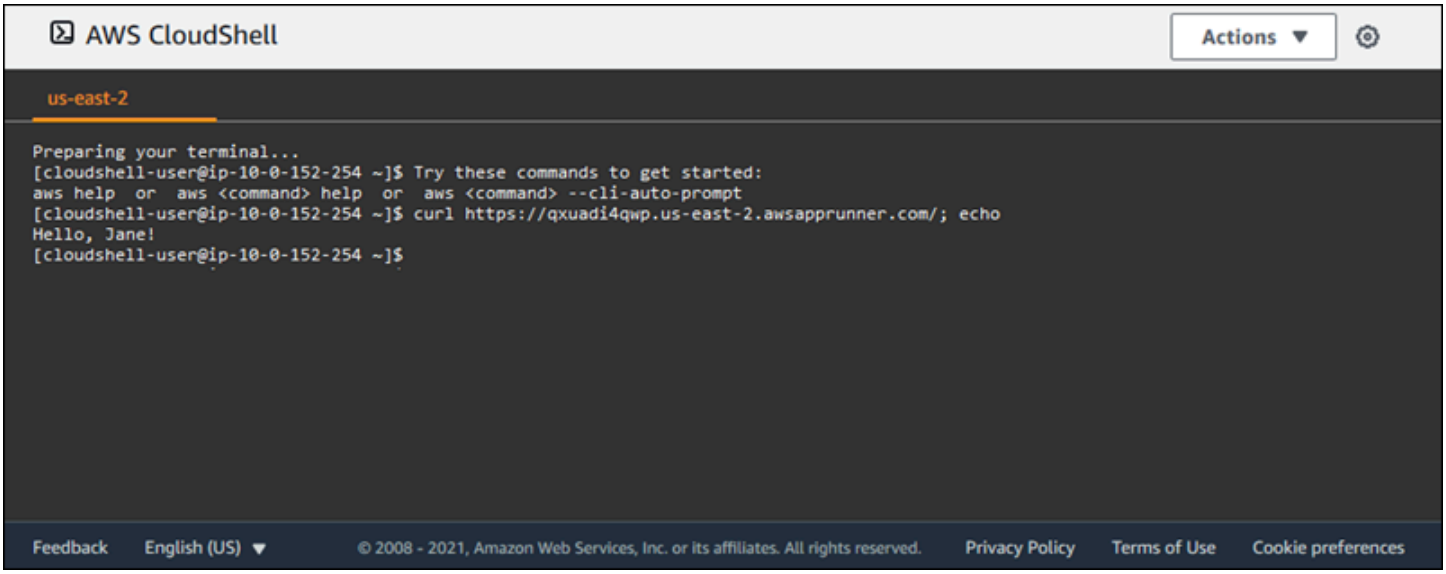
使用 驗證您的 App Runner 服務 AWS CloudShell

當您建立 App Runner 服務時，App Runner 會為您的服務網站建立預設網域，並在主控台中顯示它（或在 API 呼叫結果中傳回它）。您可以使用 CloudShell 對網站進行呼叫，並確認其正常運作。

例如，在如 中所述建立 App Runner 服務之後開始使用，請在 CloudShell 中執行下列命令：

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

輸出應會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。錯誤訊息會出現在 主控台、API 或服務頁面的日誌索引標籤上。

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

- 當服務無法建立時
- 自訂網域名稱
- HTTP/HTTPS 請求路由錯誤
- 當服務無法連線至 Amazon RDS 或下游服務時
- 當沒有足夠的 IP 地址來啟動執行個體或擴展時

當服務無法建立時

如果您嘗試建立 App Runner 服務失敗，服務會進入 `CREATE_FAILED` 狀態。此狀態在主控台上顯示為建立失敗。由於與下列一或多個項目相關的問題，服務可能無法建立 ：

- 您的應用程式程式碼
- 建置程序
- Configuration
- 資源配額
- 服務使用之基礎 AWS 服務 的暫時性問題

若要對無法建立的服務進行故障診斷，建議您執行下列動作。

1. 閱讀服務事件和日誌，了解導致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. 如果您達到服務配額，請刪除一或多個 服務。
4. 如果您達到另一個資源配額，如果可調整，則可能可以增加。
5. 完成上述所有步驟後，請嘗試重新建置服務。如需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 App Runner 可提供給服務的執行個體數量。這是服務中 Fargate 隨需 vCPU 資源計數的可調整配額值 AWS Fargate 。若要檢視您帳戶的 vCPU 配額設定或請求提高配額，請使用 中的 Service Quotas 主控台 AWS 管理主控台。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的AWS Fargate 服務配額。

> **Important**
>
> 除了失敗服務的初始建立嘗試之外，您不會產生任何額外費用。即使故障的服務無法使用，它仍會計入您的服務配額。App Runner 不會自動刪除失敗的服務，因此請務必在完成故障分析時將其刪除。

自訂網域名稱

本節涵蓋如何對連結至自訂網域時可能遇到的各種錯誤進行故障診斷和解決。

> **Note**
>
> 為了增強 App Runner 應用程式的安全性，在公有尾碼清單 (PSL) 中註冊 *.awsapprunner.com 網域。為了進一步提高安全性，如果您需要在 App Runner 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用具有`__Host-`字首的 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

- 檢查此錯誤是否為 CAA 記錄的問題所致。如果 DNS 樹狀目錄中沒有 CAA 記錄，您會收到訊息 `fail open`，並 AWS Certificate Manager 發出憑證來驗證自訂網域。這可讓 App Runner 接受自訂網域。如果您在 DNS 記錄中使用 CAA 憑證，請確定至少一個網域的 CAA 記錄包含 `amazon.com`。否則，ACM 無法發出憑證。因此，無法建立 App Runner 的自訂網域。

  下列範例使用 DNS 查詢工具 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，並確保至少一個 CAA 記錄包含 `amazon.com`。
- 重試將自訂網域與 App Runner 連結。

如需如何解決 CAA 錯誤的說明，請參閱以下內容：

- 憑證授權機構授權 (CAA) 問題
- 如何解決發行或續約 ACM 憑證的 CAA 錯誤？

取得自訂網域的 DNS 憑證驗證擱置錯誤

- 檢查您是否略過自訂網域設定中的重要步驟。此外，請檢查您是否使用 DiG 等 DNS 查詢工具來設定不正確的 DNS 記錄。特別是，請檢查下列錯誤：
  - 任何遺漏的步驟。
  - 不支援的字元，例如 DNS 記錄中的雙引號。
- 更正錯誤。
- 重試將自訂網域與 App Runner 連結。

如需如何解決 CAA 驗證錯誤的說明，請參閱以下內容。

- DNS 驗證
- the section called “自訂網域名稱”

基本疑難排解命令

- 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
```

- 列出所有進行中的操作。

```
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 會為您提供一組您新增至 DNS 伺服器的 CNAME 記錄。這些 CNAME 記錄包含憑證記錄。App Runner 使用 AWS Certificate Manager (ACM) 驗證網域。App Runner 會驗證這些 DNS 記錄，以確保持續擁有此網域。如果您從 DNS 區域移除 CNAME 記錄，App Runner 就無法再驗證 DNS 記錄，而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 已從 DNS 伺服器移除”.
- the section called “憑證已過期”.

CNAME 已從 DNS 伺服器移除

- 使用 DescribeCustomDomains API 或從 App Runner 主控台的自訂網域設定擷取 CNAME 記錄。如需已儲存 CNAMEs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，App Runner 可以驗證您擁有該網域。新增 CNAME 記錄後，最多可能需要 30 分鐘才能傳播 DNS 記錄。App Runner 和 ACM 可能需要幾個小時才能重試憑證續約程序。如需如何新增 CNAME 記錄的說明，請參閱 the section called “管理自訂網域”。

憑證已過期

- 使用 App Runner 主控台或 API 取消關聯（取消連結），然後關聯（連結）App Runner 服務的自訂網域。App Runner 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）和關聯（連結）自訂網域的指示，請參閱 the section called “管理自訂網域”。

如何驗證憑證已成功續約

您可以檢查憑證記錄的狀態，以確認您的憑證已成功續約。您可以使用 curl 等工具來檢查憑證的狀態。

如需憑證續約的詳細資訊，請參閱下列連結：

- 為什麼我的 ACM 憑證標示為不符合續約資格？
- ACM 憑證的受管續約
- 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 流量路由至 App Runner 服務端點時可能遇到的錯誤。

404 將 HTTP/HTTPS 流量傳送至 App Runner 服務端點時發生錯誤

- 確認 `Host Header` 指向 HTTP 請求中的服務 URL，因為 App Runner 會使用主機標頭資訊來路由請求。cURL和 Web 瀏覽器等大多數用戶端會自動將主機標頭指向服務 URL。如果您的用戶端未將服務 URL 設定為 `Host Header`，您會收到`404 Not Found`錯誤。

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
```

- 確認您的用戶端已針對路由至公有或私有服務的請求正確設定伺服器名稱指標 (SNI)。對於 TLS 終止和請求路由，App Runner 會在 HTTPS 連線中使用 SNI 集。

當服務無法連線至 Amazon RDS 或下游服務時

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，則可能存在網路組態問題。本主題會逐步解說一些步驟，以判斷您的網路組態是否有任何問題，以及更正這些組態的選項。若要進一步了解 App Runner 的傳出流量組態，請參閱 啟用傳出流量的 VPC 存取。

> **Note**
>
> 若要檢視 VPC Connector 組態，請從 App Runner 主控台左側導覽窗格中，選取網路組態。然後選取傳出流量索引標籤。選取 VPC 連接器。下一頁顯示 VPC Connector 的詳細資訊。在此頁面上，您可以檢視並深入了解下列項目：使用 VPC 的子網路、安全群組和 App Runner 服務。

縮小應用程式無法連線至另一個下游服務的原因

1. 確定 VPC 連接器中使用的子網路是私有子網路。如果使用公有子網路設定連接器，您的服務將會發生錯誤，因為每個子網路的基礎 Hyperplane ENIs（彈性網路介面）沒有公有 IP 空間。

   如果您的 VPC 連接器使用公有子網路，您有下列選項可更正此組態：

a. 建立新的私有子網路，並使用它來取代 VPC Connector 的公有子網路。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 的子網路。

b. 透過 NAT 閘道路由現有的公有子網路。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT 閘道。

2. 驗證 VPC Connector 的安全群組輸入和輸出規則是否正確。從 App Runner 主控台左側導覽窗格中，選取網路組態 > 傳出流量。從清單中選擇 VPC Connector。下一頁列出您可以選擇檢查的安全群組。

3. 確認您嘗試連線的 RDS 執行個體或其他下游服務的安全群組傳入和傳出規則正確無誤。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. 若要確認 App Runner 組態之外沒有其他類型的網路設定問題，請嘗試連線至 App Runner 之外的 RDS 或下游服務：

   a. 從相同 VPC 中的 Amazon EC2 執行個體，嘗試連線至 RDS 執行個體或服務。

   b. 如果您嘗試連線到服務 VPC 端點，請從相同 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 中的任一連線測試失敗，很可能在 App Runner 組態之外存在您 AWS帳戶中另一個資源的問題。如需進一步隔離和修正其他網路組態問題的協助，請聯絡 AWS Support。

6. 如果您透過執行步驟 4 中的指示成功連接到 RDS 執行個體或下游服務，則繼續此步驟中的指示。我們會啟用並檢查 Hyperplane ENI 流程日誌，以檢查流量是否進入 ENI。

> **Note**
>
> 若要能夠完成這些步驟並取得所需的 ENI 流程日誌資訊，必須在 App Runner 服務成功啟動後嘗試連線到 RDS 或下游服務。您的應用程式必須在處於執行中狀態時，執行 RDS 或下游服務的連線操作。否則，可以在 App Runner 的復原工作流程中清除 ENIs。此方法可確保 ENIs仍可供進一步調查。

   a. 從 AWS 主控台啟動 EC2 主控台。

   b. 從左側導覽窗格中，在網路與安全群組中，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尋找 ENIs。它們會有下列命名模式。

      - 介面類型： fargate

- 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適用的 ENIs。

e. 從動作功能表中選取建立流程日誌。

f. 在提示中輸入資訊，然後選取頁面底部的建立流程日誌。

g. 檢查產生的流程日誌。

- 如果流量在您測試連線時進入 ENI，則問題與 ENI 設定無關。除了 App Runner 服務之外，您 AWS 帳戶中的另一個資源可能會發生網路組態問題。如需 AWS 進一步協助，請聯絡 Support。
- 如果您在測試連線時流量未進入 ENI，建議您聯絡 AWS Support，以查看 Fargate 服務是否有任何已知問題。

h. 使用網路 Reachability Analyzer 工具。當虛擬網路路徑中的來源無法連線時，此工具可透過識別封鎖元件，協助判斷網路組態錯誤。如需詳細資訊，請參閱什麼是 Reachability Analyzer？ Amazon VPC Reachability Analyzer 指南中的 。

輸入 App Runner ENI 做為來源，然後輸入 RDS ENI 做為目的地。

7. 如果您無法進一步縮小問題範圍，或在完成先前步驟後仍無法連線至 RDS 或下游服務，建議您聯絡 AWS Support 以取得進一步協助。

當沒有足夠的 IP 地址來啟動執行個體或擴展時

> **Note**
>
> 對於公有服務，App Runner 不會在您的 VPCs中建立彈性網路界面 (ENI)，因此您的公有服務不受此變更影響。

本指南可協助您解決在啟用傳出流量的 VPC 存取的 App Runner 服務上可能遇到的 IP 耗盡錯誤。

App Runner 將在與您的 VPC 連接器相關聯的子網路中啟動執行個體。App Runner 會在啟動執行個體的子網路中，為每個執行個體建立 1 個 ENI。每個 ENI 在該子網路中使用私有 IP。子網路有固定數量 IPs，取決於與該子網路相關聯的 CIDR 區塊。如果 App Runner 找不到有足夠的 IPs來建立 ENI 的子網路 (App Runner)，則無法為您的 App Runner 服務啟動新的執行個體。這可能會導致擴展服務的問題。在這種情況下，您會看到 App Runner 事件日誌，指出 App Runner 找不到具有可用 IPs子網路。您可以使用以下指示來更新您的服務，以解決此類錯誤。

如何更新您的服務以擁有更多可用的 IPs

子網路中可用的 IP 地址數量取決於與該子網路相關聯的 CIDR 區塊。與子網路相關聯的 CIDR 區塊無法在建立後更新。App Runner VPC 連接器建立後也無法更新。若要為啟用傳出流量的 VPC 存取提供更多 IPs 給 App Runner 服務：

1. 使用較大的 CIDR 區塊建立新的子網路（子網路）。
2. 使用新的子網路（含）建立新的 VPC 連接器。
3. 更新您的 App Runner 服務以使用新的 VPC 連接器。

計算服務所需的 IPs

嘗試使用較大的 CIDR 區塊建立新子網路之前，請判斷您在 App Runner 服務中所需的 IPs 數量。建議您計算連接器中所需的 IPs 數量，如下所示：

1. 對於已啟用傳出流量的 VPC 存取的每個服務，請注意自動擴展組態中的大小上限（執行個體上限）。
2. 將所有服務的值加總。
3. 將此總和加倍，以考量藍綠部署期間啟動的新執行個體。

範例

考慮使用相同 VPC 連接器的兩個服務 A 和 B。

1. 服務 A 的大小上限為 25。
2. 服務 B 的大小上限為 15。

必要的 IPs = 2 × (25 + 15) = 80

確保您的子網路至少合併 80 IPs。

建立新的子網路 (s)

1. 使用此公式判斷 IPv4 所需的 CIDR 區塊大小（請注意，AWS 會保留 5 IPs：子網路大小)

```
Number of available IP addresses = 2^(32 - prefix length) - 5
```

```
Example :
For 192.168.1.0/24:
Prefix length is 24
Number of available IP addresses = 2^(32 - 24) - 5 = 2^8-5 = 251 IP addresses

For 10.0.0.0/16:
Prefix length is 16
Number of available IP addresses = 2^(32 - 16) - 5 = 2^16-5 = 65,531 IP addresses

Quick reference:
/24 = 251 IP addresses
/16 = 65,531 IP addresses
```

2. 使用 AWS EC2 CLI 建立新的子網路。

```
aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

範例（建立具有 4，096 個 IPs子網路）：

```
aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
```

3. 建立新的 VPC 連接器。請參閱：管理 VPC 存取
4. 在啟用 VPC 傳出流量的情況下更新服務，以使用此新的 VPC 連接器。更新服務後，App Runner 將開始使用新的子網路。

> **Note**
>
> VPCs也會受限於 CIDR 區塊可配置給子網路的可用 IPs 數量。如果您無法使用較大的 CIDR 區塊建立子網路，您可能需要先使用次要 CIDR 區塊更新 VPC（然後再建立新的子網路）。

將次要 CIDR 區塊連接至 VPC

將次要 CIDR 區塊與此 VPC 建立關聯。

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

範例：

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

驗證

更新服務之後。您可以使用下列項目來執行修正的驗證

1. 監控事件日誌 ：監控您的 App Runner 服務事件日誌，以驗證沒有出現新的 IP 或 ENI 無法使用錯誤
2. 檢查服務擴展：
    1. 變更自動擴展組態中的執行個體計數下限，以完全擴展服務
    2. 確認所有新執行個體都已啟動，沒有任何 IP 相關錯誤
    3. 透過數個擴展事件進行監控，以確保一致的效能
3. 主控台橫幅：如果您使用的是 AWS 管理主控台，請確認 App Runner 不再顯示有關 IPs橫幅警告。
4. VPC 和子網路 IP 使用率：
    1. 使用 VPC 儀表板或 CLI 命令來檢查新子網路中的 IP 地址使用率。
    2. 確認您的服務擴展後，可用 IPs仍有正常的邊界

常見缺陷

在 App Runner 服務中解決 IP 耗盡問題時，請注意下列潛在問題：

1. IP 地址規劃不足：低估未來的 IP 需求可能會導致經常性耗盡問題。進行徹底的容量規劃，考慮潛在的服務成長和尖峰使用案例。
2. 覆寫整個 VPC 的 IP 用量：請記住，相同 VPC 中的其他 AWS 服務也會使用 IP 地址。規劃 VPC 和子網路組態時，請考慮所有 服務的 IP 需求。
3. 忽略更新服務：建立新的子網路或 VPC 連接器之後，請務必更新您的 App Runner 服務以使用新的組態。否則，將會繼續使用耗盡的 IP 範圍。
4. 錯誤的 CIDR 區塊重疊：將次要 CIDR 區塊新增至 VPC 時，請確保它們不會與現有區塊重疊。重疊 CIDR 區塊可能會導致路由衝突和 IP 地址模棱兩可。
5. 超過 VPC 限制：請注意，VPC 最多可有 5 個 CIDR 區塊 (1 個主要區塊和 4 個次要區塊）。在這些限制範圍內規劃您的 IP 地址空間擴展。

6. 忽略子網路可用區域分佈：建立新的子網路時，請確保它們分佈在多個可用區域，以實現高可用性和容錯能力。
7. 覆寫 ENI 限制：請記住，可以連接到執行個體ENIs 數量有限制。確認您的 AWS 帳戶限制符合您計劃的網路介面用量。

透過了解這些陷阱，您可以更有效地管理您的 VPC 資源，並避免 App Runner 服務中的 IP 耗盡問題。

其他資源

1. AWS VPC 文件
2. 了解 CIDR 區塊
3. App Runner VPC 連接器

詞彙表

1. ENI：Elastic Network Interface，AWS 中的虛擬網路介面。
2. CIDR：Classless 網域間路由，一種配置 IP 地址的方法。
3. VPC Connector：可讓 App Runner 連線至 VPC 的資源。

App Runner 的安全性

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

安全性是 AWS 與您之間的共同責任。共同責任模式將其描述為雲端的安全性，和雲端中的安全性：

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

本文件可協助您了解如何在使用 App Runner 時套用共同責任模型。下列主題說明如何設定 App Runner 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 App Runner 資源。

主題

- App Runner 中的資料保護
- App Runner 的身分和存取管理
- 在 App Runner 中記錄和監控
- App Runner 的合規驗證
- App Runner 中的彈性
- 中的基礎設施安全性 AWS App Runner
- 搭配 VPC 端點使用 App Runner
- App Runner 中的組態和漏洞分析
- App Runner 的安全最佳實務

App Runner 中的資料保護

共同責任模型適用於 AWS 中的資料保護 AWS App Runner。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 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。 如需歐洲資料保護的相關資訊，請參閱「一般資料保護法規 (GDPR) 中心」。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 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。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，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) 140-3。

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 App Runner 或使用主控台、API AWS CLI、 AWS SDKs的其他 AWS 服務 時。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 App Runner 安全主題，請參閱 安全。

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 會從您指定的儲存庫讀取您的應用程式來源 （ 來源映像或來源碼 ），並將其存放以部署到您的服務。如需詳細資訊，請參閱架構和概念。

資料保護是指在傳輸中 （ 往返於 App Runner 時 ） 和靜態 （ 存放於 AWS 資料中心時 ） 時保護資料。

如需資料保護的詳細資訊，請參閱 the section called “資料保護”。

如需其他 App Runner 安全主題，請參閱 安全。

傳輸中加密

您可以透過兩種方式實現傳輸中的資料保護：使用 Transport Layer Security (TLS) 加密連線，或使用用戶端加密（在傳送物件之前對其進行加密）。這兩種方法都能保護您的應用程式資料。為了保護連線，每當您的應用程式、開發人員和管理員及其最終使用者傳送或接收任何物件時，請使用 TLS 加密連線。App Runner 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 部署的來源映像或程式碼的有效方法。App Runner 需要存取您的應用程式來源，因此無法加密。因此，請務必保護開發或部署環境與 App Runner 之間的連線。

靜態加密和金鑰管理

為了保護應用程式的靜態資料，App Runner 會加密應用程式原始碼映像或原始碼套件的所有儲存複本。當您建立 App Runner 服務時，您可以提供 AWS KMS key。如果您提供金鑰，App Runner 會使用您提供的金鑰來加密來源。如果您未提供，App Runner 會 AWS 受管金鑰 改用 。

如需 App Runner 服務建立參數的詳細資訊，請參閱 CreateService。如需 AWS Key Management Service (AWS KMS) 的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》。

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邊界，並控制它們、內部部署網路和網際網路之間的流量。如需 Amazon VPC 安全性的詳細資訊，請參閱《Amazon VPC 使用者指南》中的 Amazon VPC 中的網際網路流量隱私權。

如需將 App Runner 應用程式與自訂 Amazon VPC 建立關聯的資訊，請參閱 the section called “傳出流量”。

如需使用 VPC 端點保護 App Runner 請求的詳細資訊，請參閱 the section called “VPC 端點”。

如需資料保護的詳細資訊，請參閱 the section called “資料保護”。

如需其他 App Runner 安全主題，請參閱 安全。

App Runner 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證（登入）和授權（具有許可），以使用 App Runner 資源。IAM 是您可以免費使用 AWS 服務 的 。

如需其他 App Runner 安全主題，請參閱 安全。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- App Runner 如何與 IAM 搭配使用
- App Runner 身分型政策範例
- 使用 App Runner 的服務連結角色
- AWS 的 受管政策 AWS App Runner
- 疑難排解 App Runner 身分和存取

目標對象

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

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 疑難排解 App Runner 身分和存取)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 App Runner 如何與 IAM 搭配使用)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 App Runner 身分型政策範例)

使用身分驗證

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

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

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

AWS 帳戶 根使用者

建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分可完整存取所有 AWS 服務 和 資源。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

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

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

IAM 角色

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

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

使用政策管理存取權

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

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

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

身分型政策

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

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

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。您必須在資源型政策中指定主體。

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

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 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。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 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

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

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

App Runner 如何與 IAM 搭配使用

在您使用 IAM 管理對 的存取之前 AWS App Runner，您應該了解哪些 IAM 功能可與 App Runner 搭配使用。若要全面了解 App Runner 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的與 IAM AWS 搭配使用的 服務。

如需其他 App Runner 安全主題，請參閱 安全。

主題

- App Runner 身分型政策
- App Runner 資源型政策
- 以 App Runner 標籤為基礎的授權
- App Runner 使用者許可
- App Runner IAM 角色

App Runner 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。App Runner 支援特定動作、資源和條件索引鍵。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內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

App Runner 中的政策動作在動作之前使用下列字首：`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 會定義自己的一組動作，描述您可以使用此服務執行的任務。

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*"
```

若要查看 App Runner 動作的清單，請參閱《服務授權參考》中的 定義的動作 AWS App Runner。

Resources

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

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

App Runner 資源具有下列 ARN 結構：

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

如需 ARNs 格式的詳細資訊，請參閱 中的 Amazon Resource Name (ARNs) AWS 和服務命名空間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/*"
```

有些 App Runner 動作無法在特定資源上執行，例如用於建立資源的動作。在這些情況下，您必須使用萬用字元 (*)。

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

若要查看 App Runner 資源類型及其 ARNs，請參閱《服務授權參考》中的 定義的資源 AWS App Runner。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內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素會根據定義的條件，指定陳述式的執行時機。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

App Runner 支援使用一些全域條件金鑰。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。

App Runner 定義一組服務特定的條件金鑰。此外，App Runner 支援標籤型存取控制，其使用條件金鑰實作。如需詳細資訊，請參閱the section called “以 App Runner 標籤為基礎的授權”。

若要查看 App Runner 條件索引鍵的清單，請參閱《服務授權參考》中的 的條件索引鍵 AWS App Runner。若要了解您可以使用條件金鑰的動作和資源，請參閱 定義的動作 AWS App Runner。

範例

若要檢視 App Runner 身分型政策的範例，請參閱 App Runner 身分型政策範例。

App Runner 資源型政策

App Runner 不支援以資源為基礎的政策。

以 App Runner 標籤為基礎的授權

您可以將標籤連接至 App Runner 資源，或在請求中將標籤傳遞至 App Runner。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 App Runner 資源的詳細資訊，請參閱the section called “Configuration”。

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 App Runner 服務的存取。

App Runner 使用者許可

若要使用 App Runner，IAM 使用者需要 App Runner 動作的許可。授予許可給使用者的常見方法是將政策連接到 IAM 使用者或群組。如需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 提供兩個可連接至使用者的受管政策。

- `AWSAppRunnerReadOnlyAccess` – 准許列出和檢視 App Runner 資源的詳細資訊。
- `AWSAppRunnerFullAccess`– 授予所有 App Runner 動作的許可。

如需更精細地控制使用者許可，您可以建立自訂政策並將其連接到您的使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需使用者政策的範例，請參閱 the section called “使用者政策”。

App Runner IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 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。

App Runner 支援服務連結角色。如需建立或管理 App Runner 服務連結角色的資訊，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。

App Runner 支援幾個服務角色。

存取角色

存取角色是 App Runner 用來存取您帳戶中 Amazon Elastic Container Registry (Amazon ECR) 中映像的角色。在 Amazon ECR 中存取映像是必要的，而 Amazon ECR Public 則不需要。

在 Amazon ECR 中根據映像建立服務之前，請使用 IAM 建立服務角色。在您的服務角色`AWSAppRunnerServicePolicyForECRAccess`中使用 受管政策。然後，當您在 SourceConfiguration 參數的 AuthenticationConfiguration 成員中呼叫 CreateService API，或使用 App Runner 主控台建立服務時，您可以將此角色傳遞給 App Runner。

> **Note**
>
> 如果您為存取角色建立自己的自訂政策，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 登錄檔。

當您建立存取角色時，請務必新增信任政策，將 App Runner 服務主體宣告`build.apprunner.amazonaws.com`為信任的實體。

存取角色的信任政策

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "build.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果您使用 App Runner 主控台來建立服務，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，您可以視需要選取不同的角色。

執行個體角色

執行個體角色是選用角色，App Runner 會用來提供許可給您 AWS 服務的運算執行個體所需的服務動作。如果您的應用程式程式碼呼叫 AWS 動作 (APIs)，您需要提供執行個體角色給 App Runner。您可

以在執行個體角色中嵌入必要的許可，或建立您自己的自訂政策，並在執行個體角色中使用它。我們無法預測您的程式碼使用哪些 呼叫。因此，我們不會為此目的提供受管政策。

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，當您在 InstanceConfiguration 參數`InstanceRoleArn`的成員中呼叫 CreateService API，或使用 App Runner 主控台建立服務時，您可以將此角色傳遞給 App Runner 做為執行個體角色。

當您建立執行個體角色時，請務必新增信任政策，將 App Runner 服務主體宣告`tasks.apprunner.amazonaws.com`為信任的實體。

執行個體角色的信任政策

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tasks.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果您使用 App Runner 主控台來建立服務，主控台會列出帳戶中的角色，而且您可以選取為此目的建立的角色。

如需建立服務的資訊，請參閱 the section called “建立”。

App Runner 身分型政策範例

根據預設，IAM 使用者和角色沒有建立或修改 AWS App Runner 資源的許可。他們也無法使用 AWS 管理主控台 AWS CLI、 或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 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 App Runner 安全主題，請參閱 安全。

主題

- 政策最佳實務
- 使用者政策
- 根據標籤控制對 App Runner 服務的存取

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 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 受管政策並邁向最低權限許可 – 若要開始將許可授予使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱《IAM 使用者指南》中的 IAM 安全最佳實務。

使用者政策

若要存取 App Runner 主控台，IAM 使用者必須擁有一組最低許可。這些許可必須允許您列出和檢視中 App Runner 資源的詳細資訊 AWS 帳戶。如果您建立比最低必要許可更嚴格的身分型政策，主控台將無法對具有該政策的使用者如預期般運作。

App Runner 提供兩個可連接至使用者的受管政策。

- `AWSAppRunnerReadOnlyAccess` – 准許列出和檢視 App Runner 資源的詳細資訊。
- `AWSAppRunnerFullAccess` – 授予所有 App Runner 動作的許可。

為了確保使用者可以使用 App Runner 主控台，請至少將 `AWSAppRunnerReadOnlyAccess`受管政策連接到使用者。您可以改為連接 `AWSAppRunnerFullAccess` 受管政策，或新增特定的額外許可，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。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只允許存取符合您要允許使用者執行之 API 操作的動作。

下列範例示範自訂使用者政策。您可以使用它們做為起點，來定義您自己的自訂使用者政策。複製範例，和/或移除動作、縮小資源範圍和新增條件。

範例：主控台和連線管理使用者政策

此範例政策可啟用主控台存取，並允許連線建立和管理。它不允許 App Runner 服務建立和管理。它可以連接到其角色為管理來源碼資產的 App Runner 服務存取權的使用者。

JSON

```
{
  "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

本節中的範例示範條件式許可，這些許可取決於某些資源屬性或動作參數。

此範例政策允許建立 App Runner 服務，但拒絕使用名為 的連線prod。

JSON

```
  { "Version":"2012-10-17",
  "Statement":
     [ { "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
         "Effect": "Allow",
         "Action": "apprunner:CreateService",
         "Resource": "*",
         "Condition":
            { "ArnNotLike":
               {"apprunner:ConnectionArn":"arn:aws:apprunner:*:*:connection/prod/
*"}
            }
       }
     ]
   }
```

此範例政策僅允許preprod使用名為 的自動擴展組態來更新名為 的 App Runner 服務preprod。

JSON

```
{
    "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:us-east-1:*:autoscalingconfiguration/preprod/*"
                }
              }
        }
    ]
}
```

根據標籤控制對 App Runner 服務的存取

您可以在身分型政策中使用條件，根據標籤控制對 App Runner 資源的存取。此範例示範如何建立允許刪除 App Runner 服務的政策。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。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:us-east-1:*:service/*",
      "Condition": {
        "StringEquals": {"aws: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，則該服務必須加上標籤 `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。

使用 App Runner 的服務連結角色

AWS App Runner use AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 App Runner 的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

主題

- 使用 角色進行管理
- 使用角色進行聯網

使用 角色進行管理

AWS App Runner use AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 App Runner 的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 App Runner，因為您不必手動新增必要的許可。App Runner 定義其服務連結角色的許可，除非另有定義，否則只有 App Runner 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 App Runner 資源，因為您不會不小心移除存取資源的許可。

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。

App Runner 的服務連結角色許可

App Runner 使用名為 AWSServiceRoleForAppRunner 的服務連結角色。

此角色允許 App Runner 執行下列任務：

- 將日誌推送至 Amazon CloudWatch Logs 日誌群組。

- 建立 Amazon CloudWatch Events 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。

AWSServiceRoleForAppRunner 服務連結角色信任下列服務擔任該角色：

- `apprunner.amazonaws.com`

AWSServiceRoleForAppRunner 服務連結角色的許可政策包含 App Runner 代表您完成動作所需的所有許可：

- 受管政策 AppRunnerServiceRolePolicy
- X-Ray 追蹤的政策 – 請參閱下列政策內容。

X-Ray 追蹤的政策

JSON

```
{
  "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 App Runner 建立服務連結角色

您不需要手動建立服務連結角色，當您在 、 或 API 中建立 App Runner 服務時，App Runner 會為您建立服務連結角色。 AWS 管理主控台 AWS CLI AWS

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 App Runner 服務時，App Runner 會再次為您建立服務連結角色。

編輯 App Runner 的服務連結角色

App Runner 不允許您編輯 AWSServiceRoleForAppRunner 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 App Runner 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，這表示刪除您帳戶中的所有 App Runner 服務。若要了解如何刪除 App Runner 服務，請參閱 the section called “刪除”。

> **Note**
>
> 如果 App Runner 服務在您嘗試刪除資源時使用角色，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 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForAppRunner 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

App Runner 服務連結角色支援的區域

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。

使用角色進行聯網

AWS App Runner use AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 App Runner 的唯一 IAM 角色類型。服務連結角色由 App Runner 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 App Runner，因為您不必手動新增必要的許可。App Runner 定義其服務連結角色的許可，除非另有定義，否則只有 App Runner 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 App Runner 資源，因為您不會不小心移除存取資源的許可。

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。

App Runner 的服務連結角色許可

App Runner 使用名為 AWSServiceRoleForAppRunnerNetworking 的服務連結角色。

此角色允許 App Runner 執行下列任務：

- 將 VPC 連接至 App Runner 服務並管理網路介面。

AWSServiceRoleForAppRunnerNetworking 服務連結角色信任下列服務擔任該角色：

- `networking.apprunner.amazonaws.com`

名為 的角色許可政策`AppRunnerNetworkingServiceRolePolicy`包含 App Runner 代表您完成動作所需的所有許可。

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 App Runner 建立服務連結角色

您不需要手動建立服務連結角色，當您在 AWS 管理主控台、 AWS CLI或 AWS API 中建立 VPC 連接器時，App Runner 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立 VPC 連接器時，App Runner 會再次為您建立服務連結角色。

編輯 App Runner 的服務連結角色

App Runner 不允許您編輯 AWSServiceRoleForAppRunnerNetworking 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 App Runner 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，這表示取消 VPC 連接器與您帳戶中所有 App Runner 服務的關聯，並刪除 VPC 連接器。如需詳細資訊，請參閱the section called “傳出流量”。

> **Note**
>
> 如果 App Runner 服務在您嘗試刪除資源時使用角色，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 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForAppRunnerNetworking 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

App Runner 服務連結角色支援的區域

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 的 受管政策 AWS App Runner

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常見使用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 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 AWS 受管政策中定義的許可。如果 AWS 更新 受管政策中 AWS 定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可用於現有 服務時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

受 AWS 管政策的 App Runner 更新

檢視自此服務開始追蹤這些變更以來，App Runner AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱 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 新增了新的政策，允許使用者列出和檢視有關 App Runner 資源的詳細資訊。	2022 年 2 月 24 日
AWSAppRunnerFullAccess – 更新至現有政策	App Runner 已更新 `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

變更	描述	日期
AppRunnerNetworkingServiceRolePolicy – 新政策	App Runner 新增了一項政策，以允許 App Runner 呼叫 Amazon Virtual Private Cloud，將 VPC 連接至您的 App Runner 服務，並代表 App Runner 服務管理網路介面。此政策用於`AWSServiceRoleForAppRunnerNetworking` 服務連結角色。	2022 年 2 月 8 日
AWSAppRunnerFullAccess – 新政策	App Runner 新增了新的政策，允許使用者執行所有 App Runner 動作。	2022 年 1 月 10 日
AppRunnerServiceRolePolicy – 新政策	App Runner 新增了一項政策，以允許 App Runner 代表 App Runner 服務呼叫 Amazon CloudWatch Logs 和 Amazon CloudWatch Events。此政策用於`AWSServiceRoleForAppRunner` 服務連結角色。	2021 年 3 月 1 日
AWSAppRunnerServicePolicyForECRAccess – 新政策	App Runner 新增了一項政策，以允許 App Runner 存取您帳戶中的 Amazon Elastic Container Registry (Amazon ECR) 映像。	2021 年 3 月 1 日
App Runner 已開始追蹤變更	App Runner 開始追蹤其 AWS 受管政策的變更。	2021 年 3 月 1 日

疑難排解 App Runner 身分和存取

使用以下資訊來協助您診斷和修正使用 AWS App Runner 和 IAM 時可能遇到的常見問題。

如需其他 App Runner 安全主題，請參閱 安全。

主題

- 我未獲授權在 App Runner 中執行動作

- 我想要允許 以外的人員 AWS 帳戶 存取我的 App Runner 資源

我未獲授權在 App Runner 中執行動作

如果 AWS 管理主控台 告訴您無權執行 動作，請聯絡您的管理員尋求協助。您的管理員是為您提供 AWS 登入憑證的人員。

當名為 的 IAM `marymajor` 使用者嘗試使用主控台檢視 App Runner 服務的詳細資訊，但沒有`apprunner:DescribeService`許可時，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
```

在這種情況下，Mary 要求她的管理員更新她的政策，以允許她使用`apprunner:DescribeService`動作存取*`my-example-service`*資源。

我想要允許 以外的人員 AWS 帳戶 存取我的 App Runner 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 App Runner 是否支援這些功能，請參閱 App Runner 如何與 IAM 搭配使用。
- 若要了解如何提供您擁有 AWS 帳戶 的資源存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

在 App Runner 中記錄和監控

監控是維護 AWS App Runner 服務可靠性、可用性和效能的重要部分。從 AWS 解決方案的所有部分收集監控資料，可讓您在發生故障時更輕鬆地偵錯故障。App Runner 與數個 AWS 工具整合，用於監控 App Runner 服務和回應潛在事件。

Amazon CloudWatch 警示

使用 Amazon CloudWatch 警示，您可以在指定的期間內監看服務指標。如果指標超過指定期間數的特定閾值，您會收到通知。

App Runner 會收集有關整個服務以及執行 Web 服務的執行個體（擴展單位）的各種指標。如需詳細資訊，請參閱指標 (CloudWatch)。

應用程式記錄

App Runner 會收集應用程式程式碼的輸出，並將其串流至 Amazon CloudWatch Logs。此輸出中的內容由您決定。例如，您可以包含對 Web 服務提出請求的詳細記錄。這些日誌記錄在安全和存取稽核中可能很有用。如需詳細資訊，請參閱日誌 (CloudWatch Logs)。

AWS CloudTrail 動作日誌

App Runner 已與 整合 AWS CloudTrail，此服務提供由使用者、角色或 App Runner 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 App Runner 的所有 API 呼叫擷取為事件。您可以在 CloudTrail 主控台中檢視最新的事件，也可以建立追蹤，以將 CloudTrail 事件持續交付至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如需詳細資訊，請參閱API 動作 (CloudTrail)。

App Runner 的合規驗證

在多個合規計畫 AWS App Runner 中，第三方稽核人員會評估 的安全與 AWS 合規。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

若要了解 AWS 服務 是否在特定合規計劃範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS 合規計劃。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 AWS 安全文件。

如需其他 App Runner 安全主題，請參閱 安全。

App Runner 中的彈性

AWS 全球基礎設施是以 AWS 區域 和 可用區域為基礎建置。 AWS 區域 提供多個實體分隔和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。透過可用區域，您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 全球基礎設施。

AWS App Runner 代表您管理和自動化全球 AWS 基礎設施的使用。使用 App Runner 時，您會受益於 AWS 提供的可用性和容錯機制。

如需其他 App Runner 安全主題，請參閱 安全。

中的基礎設施安全性 AWS App Runner

作為受管服務， AWS App Runner 受到 Amazon Web Services：安全程序概觀白皮書中所述的 AWS 全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路管理和操作 App Runner。呼叫 App Runner APIs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。這些要求不適用於來自 App Runner 應用程式的端點。

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 App Runner 安全主題，請參閱 安全。

搭配 VPC 端點使用 App Runner

您的 AWS 應用程式可能會將 AWS App Runner 服務與從 Amazon Virtual Private Cloud (Amazon VPC) 在 VPC 中執行 AWS 服務 的其他 整合。應用程式的一部分可能會從 VPC 內向 App Runner 提出請求。例如，您可以使用 AWS CodePipeline 持續部署到 App Runner 服務。改善應用程式安全性的一種方法是透過 VPC 端點傳送這些 App Runner 請求 （以及其他請求 AWS 服務)。

使用 VPC 端點，您可以將 VPC 私下連線至支援 AWS 服務 且採用 技術的 VPC 端點服務 AWS PrivateLink。您不需要網際網路閘道、NAT 裝置、VPN 連線或 Direct Connect 連線。

VPC 中的資源不會使用公有 IP 地址與 App Runner 資源互動。VPC 和 App Runner 之間的流量不會離開 Amazon 網路。如需 VPC 端點的詳細資訊，請參閱《 AWS PrivateLink 指南》中的 VPC 端點。

> **Note**
>
> 根據預設，App Runner 服務中的 Web 應用程式會在 App Runner 提供和設定的 VPC 中執行。此 VPC 為公有。這表示它已連線到網際網路。您可以選擇性地將應用程式與自訂 VPC 建立關聯。如需詳細資訊，請參閱the section called “傳出流量”。
> 您可以設定您的服務以存取網際網路，包括 AWS APIs，即使您的服務連線到 VPC。如需如何為 VPC 傳出流量啟用公有網際網路存取的說明，請參閱 the section called “選取子網路時的考量事項”。
> App Runner 不支援為您的應用程式建立 VPC 端點。

設定 App Runner 的 VPC 端點

若要在 VPC 中為 App Runner 服務建立介面 VPC 端點，請遵循 AWS PrivateLink 指南中的建立介面端點程序。在 Service Name (服務名稱) 中，選擇 `com.amazonaws.region.apprunner`。

VPC 網路隱私權考量事項

> **Important**
>
> 將 VPC 端點用於 App Runner 並不保證來自 VPC 的所有流量都保持在網際網路之外。VPC 可能是公有的。此外，您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權，請閱讀本節。

為了確保 VPC 中網路流量的隱私權，請考慮下列事項：

- 啟用 DNS 名稱 – 您應用程式的一部分可能仍會使用`apprunner.region.amazonaws.com`公有端點透過網際網路將請求傳送至 App Runner。如果您的 VPC 已設定網際網路存取，則這些請求會成功，而不會對您發出任何指示。您可以透過確保在建立端點時啟用 DNS 名稱來防止這種情況。根據預設，它會設定為 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人 AWS 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，並啟用這些端點上的 DNS 名稱。
- 設定私有 VPC – 如果可能（如果您的解決方案完全不需要網際網路存取），請將 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 端點造成可見錯誤，因此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

App Runner 支援 VPC 端點政策。根據預設，允許透過介面端點完整存取 App Runner。VPC 端點政策可用來控制哪些 AWS 主體可以存取 App Runner 端點。或者，您可以將安全群組與端點網路介面建立關聯，以控制透過介面端點傳送至 App Runner 的流量。

與介面端點整合

App Runner 支援 AWS PrivateLink，可為 App Runner 提供私有連線，並消除網際網路流量的暴露。若要讓您的應用程式使用 傳送請求至 App Runner AWS PrivateLink，請設定稱為介面端點的 VPC 端點類型。如需詳細資訊，請參閱《 AWS PrivateLink 指南》中的界面 VPC 端點 (AWS PrivateLink)。

App Runner 中的組態和漏洞分析

AWS 和客戶共同負責實現高階軟體元件的安全性和合規性。如需詳細資訊，請參閱 AWS 共同責任模型。

修補程式容器映像

修補容器映像是共用安全模型中客戶責任的一部分。映像擁有者負責更新並定期修補容器映像。建議您建立例行排程，以檢查和套用更新至容器映像。如需如何掃描映像是否有漏洞的詳細資訊，請參閱 AWS App Runner 文件

如需其他 App Runner 安全主題，請參閱 安全。

App Runner 的安全最佳實務

AWS App Runner 提供數種安全功能，供您在開發和實作自己的安全政策時考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。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 App Runner 安全主題，請參閱 安全。

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

App Runner 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 App Runner 服務正確操作所需的所有許可。

您的應用程式可能不需要受管政策中的所有許可。您可以自訂它們，並僅授予使用者和 App Runner 服務執行其任務所需的許可。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。

掃描您的映像是否有漏洞

您可以使用 Amazon ECR 的 APIs，協助識別容器映像中的軟體漏洞。如需詳細資訊，請參閱 Amazon ECR 文件。

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 解決方案可靠性、安全性、可用性和效能的重要部分。 AWS 提供數種工具和服務，協助您監控 AWS 服務。

以下是一些要監控的項目範例：

- App Runner 的 Amazon CloudWatch 指標 – 為主要 App Runner 指標和應用程式的自訂指標設定警示。如需詳細資訊，請參閱指標 (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

如需最新的 AWS 術語，請參閱 AWS 詞彙表 參考中的AWS 詞彙表。