使用者指南

AWS AppConfig

AWS AppConfig: 使用者指南

Table of Contents

什麼是 AWS AppConfig？

AWS AppConfig功能旗標和動態設定可協助軟體建置人員快速安全地調整生產環境中的應用程式行為，而無需完整的程 AWS AppConfig加速軟體發行頻率、改善應用程式恢復能力，並協助您更快解決突發的問題。使用功能旗標，您可以逐步向使用者發佈新功能，並評估這些變更的影響，然後再將新功能完全部署給所有使用者。透過操作旗標和動態設定，您可以更新封鎖清單、允許清單、節流限制、記錄詳細資訊，以及執行其他作業調整，以快速回應生產環境中的問題。

> **Note**
>
> AWS AppConfig 是 AWS Systems Manager 的功能。

提高效率並更快地發布更改

搭配新功能使用功能旗標可加速發行生產環境變更的程序。功能旗標可讓您使用以主幹為基礎的開發來撰寫軟體，而不是仰賴需要在發行前進行複雜合併的長期開發分支。功能旗標可讓您在使用者隱藏的 CI/CD 管線中安全地推出預先發行程式碼。當您準備好發行變更時，您可以更新功能旗標，而不需要部署新的程式碼。啟動完成之後，旗標仍可做為區塊參數運作，以停用新功能或功能，而不需要回復程式碼部署。

透過內建的安全功能，避免意外變更或故障

AWS AppConfig提供下列安全功能，協助您避免啟用功能旗標或更新可能導致應用程式失敗的組態資料。

- 驗證器：驗證器可確保您的配置數據在語法和語義上是正確的，然後再將更改部署到生產環境。
- 部署策略：部署策略可讓您在數分鐘或數小時內緩慢發行生產環境的變更。
- 監控和自動復原：與 Amazon AWS AppConfig 整合 CloudWatch 以監控應用程式的變更。如果您的應用程式因為組態變更不良而變得不正常，而且變更會觸發警示 CloudWatch，則AWS AppConfig會自動回復變更，將對應用程式使用者的影響降到最低。

安全且可擴充的功能旗標部署

AWS AppConfig與 AWS Identity and Access Management (IAM) 整合，以提供精細、以角色為基礎的服務存取。 AWS AppConfig還與AWS Key Management Service（AWS KMS）集成以進行加密和AWS CloudTrail審計。在向外部客戶發佈之前，所有AWS AppConfig安全控制措施最初都是由大規模使用該服務的內部客戶開發並驗證。

AWS AppConfig 使用案例

儘管應用程式組態內容因應用程式而異很大，但仍AWS AppConfig支援下列使用案例，這些使用案例涵蓋了廣泛的客戶需求：

- 功能旗標與切換 — 在受控環境中安全地發佈新功能給您的客戶。如果遇到問題，請立即復原變更。
- 應用程式調整 — 仔細介紹應用程式變更，同時測試這些變更對生產環境中使用者造成的影響。
- 允許列表或阻止列表-控制對高級功能的訪問或立即阻止特定用戶，而無需部署新的代碼。
- 集中式組態儲存 — 在所有工作負載中保持組態資料井然有序且一致。您可以用AWS AppConfig來部署存放在AWS AppConfig託管組態存放區AWS Secrets Manager、Systems Manager 參數存放區或 Amazon S3 中的組態資料。

使用 AWS AppConfig 的優勢

AWS AppConfig為您的組織提供下列優點：

- 減少客戶的意外停機時間

  AWS AppConfig 藉由讓您建立規則來驗證組態，以 減少應用程式停機時間。無效的組態無法部署。AWS AppConfig提供下列兩個用於驗證組態的選項：

  - 對於語法驗證，您可以使用 JSON 模式。AWS AppConfig 會使用 JSON 結構描述來驗證您的組態，以確保組態變更會遵循應用程式需求。
  - 對於語義驗證，AWS AppConfig可以調用您擁有的AWS Lambda函數來驗證配置中的數據。
- 在一組目標之間快速部署變更

  AWS AppConfig透過從中央位置部署組態變更，簡化大規模應用程式的管理。 AWS AppConfig支援存放在AWS AppConfig託管組態存放區、Systems Manager 參數存放區、Systems Manager (SSM) 文件和 Amazon S3 中存放的組態。AWS AppConfig 可以與 EC2 執行個體、AWS Lambda、容器、行動應用程式或 IoT 裝置上託管的應用程式一起使用。

  目標不需要使用系統管理員 SSM 代理程式或其他系統管理員功能所需的 IAM 執行個體設定檔來設定。這表示 AWS AppConfig 可搭配未受管執行個體運作。
- 更新應用程式而不中斷

  AWS AppConfig 會在執行時間將組態變更部署到您的目標，而不需要繁重的建置程序或將您的目標停止服務。

- 控制應用程式中變更的部署

  將組態變更部署到目標時，AWS AppConfig可讓您使用部署策略將風險降至最低。部署策略可讓您緩慢地對叢集推出組態變更。如果您在部署期間遇到問題，您可以在組態變更到達大多數主機之前復原。

AWS AppConfig 的運作方式

本節提供有關如何AWS AppConfig工作以及如何開始的高級描述。

1. 識別您要在雲端中管理的程式碼中的組態值

   在開始建立AWS AppConfig成品之前，建議您在程式碼中識別要使用動態管理的組態資料AWS AppConfig。很好的例子包括功能標誌或切換，允許和阻止列表，日誌詳細信息，服務限制和節流規則，僅舉幾例。

   如果您的組態資料已存在於雲端中，您可以利用AWS AppConfig驗證、部署和延伸功能來進一步簡化組態資料管理。

2. 創建一個應用程序名

   若要建立命名空間，您可以建立稱為應用程式的AWS AppConfig成品。一個應用程序只是一個組織結構像一個文件夾。

3. 建立環境

   您可以為每個AWS AppConfig應用程式定義一或多個環境。環境是目標的邏輯分組，例如`Production`環境中的`Beta`應用程式、AWS Lambda函數或容器。您也可以定義應用程式子元件的環境，例如`WebMobile`、和`Back-end`。

   您可以為每個環境設定 Amazon CloudWatch 警示。系統會在組態部署期間監控警示。如果觸發了警示，系統會回復組態。

4. 建立組態描述檔

   配置描述檔包括可讓AWS AppConfig您在其儲存位置尋找組態資料的 URI 以及設定檔類型。 AWS AppConfig支援兩種組態設定檔類型：功能旗標和自由格式組態。功能標誌配置文件將其數據存儲在AWS AppConfig託管的配置存儲中，並且 URI 很簡單`hosted`。對於自由格式組態設定檔，您可以將資料儲存在AWS AppConfig主控組態存放區或任何與整合的AWS服務中AWS AppConfig，如中[在中建立任意格式組態設定檔 AWS AppConfig](#)所述。

組態描述檔還可以包括選用的驗證器，以確保您的組態資料在語法和語義上是正確的。當您開始部署時，AWS AppConfig 會使用驗證器執行檢查。如果偵測到任何錯誤，部署會復原至先前的組態資料。

5. 部署組態資料

   建立新部署時，請指定下列項目：

   - 應用程式識別碼
   - 設定描述檔識別碼
   - 配置版本
   - 您要在其中部署組態資料的環境 ID
   - 部署策略 ID，定義您希望變更生效的速度

   當您呼叫 StartDeploymentAPI 動作時，請AWS AppConfig執行下列工作：

   1. 使用設定描述檔中的位置 URI，從基礎資料存放區擷取組態資料。
   2. 使用您在建立組態設定檔時指定的驗證程式，驗證組態資料在語法和語意上是否正確。
   3. 緩存數據的副本，以便您的應用程序可以檢索它。此快取副本稱為已部署的資料。

6. 擷取組態

   您可以將AWS AppConfig代理程式設定為本機主機，並讓代理程式輪詢以AWS AppConfig進行組態更新。代理程式會呼叫StartConfigurationSession和 GetLatestConfigurationAPI 動作，並在本機快取您的組態資料。若要擷取資料，您的應用程式會對本機主機伺服器進行 HTTP 呼叫。 AWS AppConfigAgent 支援數個使用案例，如中所述簡化的擷取方法。

   如果您的使用案例不支援 AWS AppConfig Agent，您可以透過直接呼叫StartConfigurationSession和 GetLatestConfigurationAPI 動作，將應用程式設定AWS AppConfig為輪詢設定更新。

AWS AppConfig入門

下列資源可協助您直接使用 AWS AppConfig。

在 Amazon Web Services AWS 頻道上查看更多視 YouTube 頻。

下列部落格可協助您深入瞭解AWS AppConfig及其功能：

- 使用AWS AppConfig功能旗標

- 驗證AWS AppConfig功能旗標和規劃資料的最佳作法

軟體開發套件

如需AWS AppConfig語言特定 SDK 的相關資訊，請參閱下列資源：

- AWS Command Line Interface
- 適用於 .NET 的 AWS 開發套件
- 適用於 C++ 的 AWS 開發套件
- 適用於 Go 的 AWS 開發套件
- 適用於 Java 的 AWS 開發套件第 2 版
- AWS適用於的 SDK JavaScript
- 適用於 PHP 的 AWS 開發套件第 3 版
- 適用於 Python 的 AWS 開發套件
- 適用於 Ruby 的 AWS 開發套件第 3 版

AWS AppConfig 的定價

的定價AWS AppConfig是以組態資料和功能旗標擷取為 pay-as-you-go 基礎。我們建議您使用AWS AppConfig代理程式來協助最佳化成本。如需詳細資訊，請參閱 AWS Systems Manager 定價。

AWS AppConfig 配額

有關AWS AppConfig端點和服務配額以及其他 Systems Manager 配額的資訊，請參閱 Amazon Web Services 一般參考.

> **Note**
>
> 如需存放 AWS AppConfig 組態之服務配額的相關資訊，請參閱關於組態存放區配額和限制。

設定 AWS AppConfig

如果您尚未這麼做，請註冊 AWS 帳戶 並建立系統管理使用者。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

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

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

建立具有管理權限的使用者

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

保護您的 AWS 帳戶根使用者

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

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

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

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

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

2. 在 IAM 身分中心中，將管理存取權授予使用者。

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

以具有管理權限的使用者身分登入

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

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

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

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

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

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

授與程式設計存取權

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

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

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

(選擇性) 根據 CloudWatch 警示設定復原的權限

您可以設定 AWS AppConfig 為回復到先前版本的組態，以回應一個或多個 Amazon CloudWatch 警示。當您設定部署以回應 CloudWatch 警示時，您可以指定 AWS Identity and Access Management (IAM) 角色。 AWS AppConfig 需要此角色才能監控 CloudWatch 警報。

> **Note**
>
> IAM 角色必須屬於目前帳戶。默認情況下，只 AWS AppConfig 能監視當前帳戶擁有的警報。如果您要設定為 AWS AppConfig 回復部署以回應來自不同帳戶的指標，則必須設定跨帳戶警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的跨帳戶跨區域 CloudWatch 主控台。

使用下列程序建立 IAM 角色，以根據 CloudWatch 警示 AWS AppConfig 進行復原。本節包括下列程序。

1. 步驟 1：根據 CloudWatch 警示建立復原的權限原則
2. 步驟 2：根據 CloudWatch 警示建立用於復原的 IAM 角色
3. 步驟 3：新增信任關係

步驟 1：根據 CloudWatch 警示建立復原的權限原則

使用下列程序建立 IAM 政策，以 AWS AppConfig 授予呼叫 `DescribeAlarms` API 動作的權限。

若要根據 CloudWatch警示建立用於復原的 IAM 權限政策

1. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
3. 在建立政策頁面上，選擇 JSON 標籤。
4. 將 JSON 索引標籤上的預設內容取代為下列權限原則，然後選擇 [下一步:標記]。

> **Note**
>
> 若要傳回有關 CloudWatch 複合警示的資訊，必須將*權限指派給 DescribeAlarmsAPI 作業，如下所示。如果範圍較窄，則無法傳回`DescribeAlarms`有關複合警報的資訊。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:DescribeAlarms"
                ],
                "Resource": "*"
            }
        ]
    }
```

5. 輸入此角色的標籤，然後選擇 Next: Review (下一步：檢閱)。
6. 在「複查」頁面的「名稱」欄位**SSMCloudWatchAlarmDiscoveryPolicy**中輸入。
7. 選擇建立政策。系統會讓您返回 Policies (政策) 頁面。

步驟 2：根據 CloudWatch 警示建立用於復原的 IAM 角色

使用下列程序建立 IAM 角色，並將您在上一個程序中建立的政策指派給該角色。

若要根據 CloudWatch 警示建立用於復原的 IAM 角色

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 Select type of trusted entity (選擇可信任執行個體類型) 下，選擇 AWS service (服務)。
4. 立即在選擇將使用此角色的服務下，選擇 EC2：允許 EC2 執行個體代表您呼叫 AWS 服務，然後選擇下一步：許可。
5. 在 [連結的權限原則] 頁面上，搜尋 SSM CloudWatchAlarmDiscoveryPolicy。
6. 選擇此政策，然後選擇 Next: Tags (下一步：標籤)。
7. 輸入此角色的標籤，然後選擇 Next: Review (下一步：檢閱)。
8. 在 [建立角色] 頁面上，輸**SSMCloudWatchAlarmDiscoveryRole**入 [角色名稱] 欄位，然後選擇 [建立角色]。
9. 在 Roles (角色) 頁面上，選擇您剛建立的角色。Summary (摘要) 頁面隨即開啟。

步驟 3：新增信任關係

使用下列程序來設定您剛建立的角色以信任 AWS AppConfig。

若要新增信任關係 AWS AppConfig

1. 在您剛建立之角色的 Summary (摘要) 頁面中，選擇 Trust Relationships (信任關係) 索引標籤，然後著選擇 Edit Trust Relationship (編輯信任關係)。
2. 編輯政策以僅包含 "`appconfig.amazonaws.com`"，如下列範例所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. 選擇 Update Trust Policy (更新信任政策)。

在中建立特徵旗標和任意格式組態資料 AWS AppConfig

本節中的主題可協助您完成中的下列工作 AWS AppConfig。這些工作會建立用於部署組態資料的重要成品。

1. 創建一個應用程序名

   若要建立應用程式命名空間，您可以建立稱為應用程式的 AWS AppConfig 成品。一個應用程序只是一個組織結構像一個文件夾。

2. 建立環境

   您可以為每個 AWS AppConfig 應用程式定義一或多個環境。環境是 AWS AppConfig 目標的邏輯部署群組，例如Beta或Production環境中的應用程式。您也可以定義應用程式子元件的環境，例如AWS Lambda functionsContainers、Web、Mobile、和Back-end。

   您可以為每個環境設定 Amazon CloudWatch 警示，以自動復原有問題的組態變更。系統會在組態部署期間監控警示。如果觸發了警示，系統會回復組態。

3. 建立組態設定描述檔

   配置描述檔包括可讓 AWS AppConfig 您在其儲存位置尋找組態資料的 URI 以及設定檔類型。AWS AppConfig 支援兩種組態設定檔類型：功能旗標和自由格式組態。功能標誌配置文件將其數據存儲在 AWS AppConfig 託管的配置存儲中，並且 URI 很簡單hosted。對於自由格式組態設定檔，您可以將資料儲存在 AWS AppConfig 裝載的組態存放區或與其他整合的 Systems Manager 功能或 AWS 服務中 AWS AppConfig，如中在中建立任意格式組態設定檔 AWS AppConfig所述。

   組態設定檔也可以包含選用的驗證程式，以確保您的組態資料在語法和語義上都是正確的。 AWS AppConfig 當您啟動部署時，會使用驗證程式執行檢查。如果偵測到任何錯誤，部署會在對組態的目標進行任何變更之前停止。

   > **Note**
   >
   > 除非您在 Amazon 簡單儲存服務 (Amazon S3) 中存放機密 AWS Secrets Manager 或管理資料有特定需求，否則我們建議您在託管組態存放區中 AWS AppConfig 託管組態資料，因為它提供了最多的功能和增強功能。

主題

範例組態

使用AWS AppConfig的功能來建立 AWS Systems Manager、管理及快速部署應用程式組態。組態是影響您應用程式行為的設定集合。請見下方範例。

功能旗標組態

下列功能旗標設定會針對每個區域啟用或停用行動付款與預設付款。

JSON

```
{
  "allow_mobile_payments": {
    "enabled": false
  },
  "default_payments_per_region": {
    "enabled": true
  }
}
```

YAML

```
---
allow_mobile_payments:
  enabled: false
default_payments_per_region:
  enabled: true
```

操作配置

下列自由格式組態會強制限制應用程式處理要求的方式。

JSON

```
{
  "throttle-limits": {
    "enabled": "true",
    "throttles": [
      {
        "simultaneous_connections": 12
      },
      {
        "tps_maximum": 5000
      }
    ],
    "limit-background-tasks": [
      true
    ]
  }
}
```

YAML

```
---
throttle-limits:
  enabled: 'true'
  throttles:
  - simultaneous_connections: 12
  - tps_maximum: 5000
  limit-background-tasks:
  - true
```

存取控制清單組態

下列存取控制清單自由格式組態指定哪些使用者或群組可以存取應用程式。

JSON

```
{
  "allow-list": {
    "enabled": "true",
    "cohorts": [
      {
        "internal_employees": true
```

```
      },
      {
        "beta_group": false
      },
      {
        "recent_new_customers": false
      },
      {
        "user_name": "Jane_Doe"
      },
      {
        "user_name": "John_Doe"
      }
    ]
  }
}
```

YAML

```
---
allow-list:
  enabled: 'true'
  cohorts:
  - internal_employees: true
  - beta_group: false
  - recent_new_customers: false
  - user_name: Jane_Doe
  - user_name: Ashok_Kumar
```

關於組態設定檔 IAM 角色

您可以使用建立可存取設定資料的 IAM 角色 AWS AppConfig。或者，您可以自行建立 IAM 角色。如果您使用建立角色 AWS AppConfig，系統會根據您選擇的組態來源類型，建立角色並指定下列權限原則之一。

組態來源是 Secrets Manager 密碼

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
             ],
            "Resource": [
                "arn:aws:secretsmanager:AWS ##:account_ID:secret:secret_name-a1b2c3"
            ]
        }
    ]
}
```

組態來源是參數存放區參數

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
                "arn:aws:ssm:AWS ##:account_ID:parameter/parameter_name"
            ]
        }
    ]
    }
```

組態來源是 SSM 文件

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument"
            ],
            "Resource": [
                "arn:aws:ssm:AWS ##:account_ID:document/document_name"
            ]
        }
```

```
    ]
}
```

如果您使用建立角色 AWS AppConfig，系統也會為角色建立下列信任關係。

```
{

  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

為您的應用程式建立命名空間 AWS AppConfig

本節中的程序可協助您建立稱為應用程式的 AWS AppConfig 人工因素。應用程序只是一個組織結構，如標識應用程序的命名空間的文件夾。這種組織建構與某些可執行程式碼之間存在關係。例如，您可以建立名為的應用程式， MyMobileApp 以組織和管理使用者所安裝之行動應用程式的組態資料。您必須先建立這些人工因素，才能用 AWS AppConfig 來部署和擷取功能旗標或任意格式組態資料。

> **Note**
>
> 您可以用 AWS CloudFormation 來建立 AWS AppConfig 成品，包括應用程式、環境、組態設定檔、部署、部署策略和託管組態版本。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS AppConfig 資源類型參考。

主題

- 建立 AWS AppConfig 應用程式 (主控台)
- 建立 AWS AppConfig 應用程式 (命令列)

建立 AWS AppConfig 應用程式 (主控台)

使用下列程序來使用 AWS Systems Manager 主控台建立 AWS AppConfig 應用程式。

建立應用程式

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 Applications (應用程式)，然後選擇 Create application (建立應用程式)。
3. 對於 Name (名稱)，輸入應用程式的名稱。
4. 對於 Description (描述)，輸入有關應用程式的資訊。
5. (選擇性) 在「副檔名」區段中，從清單中選擇副檔名。如需詳細資訊，請參閱 關於 AWS AppConfig 擴展。
6. (選擇性) 在「標籤」區段中，輸入金鑰和選用值。您可以為資源指定最多 50 個標籤。
7. 選擇建立應用程式。

AWS AppConfig 會建立應用程式，然後顯示「環境」頁籤。繼續執行「為您的應用程式建立環境 AWS AppConfig」。

建立 AWS AppConfig 應用程式 (命令列)

下列程序說明如何使用 AWS CLI (在 Linux 或 Windows 上) 或建 AWS Tools for PowerShell 立 AWS AppConfig 應用程式。

若要逐步建立應用程式

1. 開啟 AWS CLI.
2. 執行下列命令以建立應用程式。

   Linux

```
aws appconfig create-application \
  --name A_name_for_the_application \
  --description A_description_of_the_application \
  --tags User_defined_key_value_pair_metadata_for_the_application
```

Windows

```
aws appconfig create-application ^
  --name A_name_for_the_application ^
  --description A_description_of_the_application ^
  --tags User_defined_key_value_pair_metadata_for_the_application
```

PowerShell

```
New-APPCApplication `
  -Name Name_for_the_application `
  -Description Description_of_the_application `
  -Tag Hashtable_type_user_defined_key_value_pair_metadata_for_the_application
```

系統會傳回相關資訊，如下所示。

Linux

```
{
   "Id": "Application ID",
   "Name": "Application name",
   "Description": "Description of the application"
}
```

Windows

```
{
   "Id": "Application ID",
   "Name": "Application name",
   "Description": "Description of the application"
}
```

PowerShell

```
ContentLength    : Runtime of the command
Description      : Description of the application
HttpStatusCode   : HTTP Status of the runtime
Id               : Application ID
Name             : Application name
```

```
ResponseMetadata : Runtime Metadata
```

為您的應用程式建立環境 AWS AppConfig

您可以為每個 AWS AppConfig 應用程式定義一或多個環境。環境是 AppConfig 目標的邏輯部署群組，例如`Production`環境中的`Beta`應用程式、 AWS Lambda 函數或容器。您也可以定義應用程式子元件的環境，例如`WebMobile`、和`Back-end`。您可以為每個環境設定 Amazon CloudWatch 警示。系統會在組態部署期間監控警示。如果觸發了警示，系統會回復組態。

開始之前

如果您 AWS AppConfig 要啟用以回復組態以回應 CloudWatch 警示，則必須設定具有權限的 AWS Identity and Access Management (IAM) 角色，以啟用 AWS AppConfig 以回應 CloudWatch 警示。您可以在下列程序中選擇此角色。如需詳細資訊，請參閱 (選擇性) 根據 CloudWatch 警示設定復原的權限。

主題

- 建立 AWS AppConfig 環境 (主控台)
- 建立 AWS AppConfig 環境 (指令行)

建立 AWS AppConfig 環境 (主控台)

使用下列程序來使用 AWS Systems Manager 主控台建立 AWS AppConfig 環境。

建立環境

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在瀏覽窗格中，選擇 [應用程式]，然後選擇要開啟詳細資料頁面的應用程式名稱。
3. 選擇 [環境] 索引標籤，然後選擇 [建立環境]。
4. 對於 Name (名稱)，輸入環境的名稱。
5. 對於 Description (描述)，輸入有關環境的資訊。
6. (選擇性) 在 [監視器] 區段中，選擇 IAM 角色欄位，然後選擇具有權限的 IAM 角色，以在觸發警示時復原組態。
7. 在CloudWatch 鬧鐘清單中，選擇一或多個要監控的鬧鐘。 AWS AppConfig 如果其中一個警示進入警示狀態，請復原您的組態部署。

8. (選擇性) 在「關聯副檔名」區段中，從清單中選擇副檔名。如需詳細資訊，請參閱 關於 AWS AppConfig 擴展。
9. (選擇性) 在「標籤」區段中，輸入金鑰和選用值。您可以為資源指定最多 50 個標籤。
10. 選擇 Create environment (建立環境)。

AWS AppConfig 會建立環境，然後顯示「環境詳細資訊」頁面。繼續執行「在中建立組態設定描述檔 AWS AppConfig」。

建立 AWS AppConfig 環境 (指令行)

下列程序說明如何使用 AWS CLI (在 Linux 或 Windows 上) 或 AWS Tools for PowerShell 建立 AWS AppConfig 環境。

若要逐步建立環境

1. 開啟 AWS CLI.
2. 執行下列命令以建立環境。

Linux

```
aws appconfig create-environment \
  --application-id The_application_ID \
  --name A_name_for_the_environment \
  --description A_description_of_the_environment \
  --monitors
 "AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
 role_for_AWS AppConfig_to_monitor_AlarmArn" \
  --tags User_defined_key_value_pair_metadata_of_the_environment
```

Windows

```
aws appconfig create-environment ^
  --application-id The_application_ID ^
  --name A_name_for_the_environment ^
  --description A_description_of_the_environment ^
  --monitors
 "AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
 role_for_AWS AppConfig_to_monitor_AlarmArn" ^
  --tags User_defined_key_value_pair_metadata_of_the_environment
```

PowerShell

```
New-APPCEnvironment `
  -Name Name_for_the_environment `
  -ApplicationId The_application_ID
  -Description Description_of_the_environment `
  -Monitors
 @{"AlarmArn=ARN_of_the_Amazon_CloudWatch_alarm,AlarmArnRole=ARN_of_the_IAM
 role_for_AWS AppConfig_to_monitor_AlarmArn"} `
  -Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_environment
```

系統會傳回相關資訊，如下所示。

Linux

```
{
   "ApplicationId": "The application ID",
   "Id": "The_environment ID",
   "Name": "Name of the environment",
   "State": "The state of the environment",
   "Description": "Description of the environment",

   "Monitors": [
      {
         "AlarmArn": "ARN of the Amazon CloudWatch alarm",
         "AlarmRoleArn": "ARN of the IAM role for AppConfig to monitor AlarmArn"
      }
   ]
}
```

Windows

```
{
   "ApplicationId": "The application ID",
   "Id": "The environment ID",
   "Name": "Name of the environment",
   "State": "The state of the environment"
   "Description": "Description of the environment",

   "Monitors": [
      {
```

```
            "AlarmArn": "ARN of the Amazon CloudWatch alarm",
            "AlarmRoleArn": "ARN of the IAM role for AppConfig to monitor AlarmArn"
        }
    ]
}
```

PowerShell

```
ApplicationId     : The application ID
ContentLength     : Runtime of the command
Description       : Description of the environment
HttpStatusCode    : HTTP Status of the runtime
Id                : The environment ID
Monitors          : {ARN of the Amazon CloudWatch alarm, ARN of the IAM role for
 AppConfig to monitor AlarmArn}
Name              : Name of the environment
Response Metadata : Runtime Metadata
State             : State of the environment
```

繼續執行「在中建立組態設定描述檔 AWS AppConfig」。

在中建立組態設定描述檔 AWS AppConfig

配置描述檔包括可 AWS AppConfig 在其儲存位置中尋找組態資料的 URI 以及配置類型。 AWS AppConfig 支援兩種類型的組態設定檔：功能旗標和自由格式組態。功能標誌配置將數據存儲在 AWS AppConfig 託管配置存儲中，URI 很簡單hosted。自由格式組態可以將資料儲存在裝載的組態存放區、各種系統 AWS AppConfig 管理員功能，或與之整合的服務中儲存資料。 AWS AWS AppConfig如需詳細資訊，請參閱 在中建立任意格式組態設定檔 AWS AppConfig。

組態設定檔也可以包含選用的驗證程式，以確保您的組態資料在語法和語義上都是正確的。 AWS AppConfig 當您啟動部署時，會使用驗證程式執行檢查。如果偵測到任何錯誤，部署會在對組態的目標進行任何變更之前停止。

> **Note**
>
> 如果可能，我們建議您在託管的組態存放區中 AWS AppConfig 託管您的組態資料，因為它提供了最多的功能和增強功能。

主題

關於驗證器

當您建立組態設定檔時，您可以選擇指定最多兩個驗證程式。驗證器可確保您的組態資料在語法和語義上是正確的。如果您打算使用驗證程式，您必須先建立驗證程式，然後再建立組態設定檔。 AWS AppConfig 支持以下類型的驗證器：

- AWS Lambda 函數：支援功能旗標和自由格式組態。
- JSON 結構描述：支援任意格式組態。 （根據 JSON 結構描述AWS AppConfig 自動驗證功能標誌。）

主題

AWS Lambda 函數驗證器

Lambda 函數驗證器必須使用下列事件結構描述來設定。 AWS AppConfig 使用此結構描述來叫用 Lambda 函數。內容是一個 base64 編碼的字串，而 URI 是字串。

```
{
    "applicationId": "The application ID of the configuration profile being
 validated",
    "configurationProfileId": "The ID of the configuration profile being validated",
    "configurationVersion": "The version of the configuration profile being validated",
    "content": "Base64EncodedByteString",
    "uri": "The configuration uri"
}
```

AWS AppConfig 驗證 Lambda `X-Amz-Function-Error` 標頭是否已在回應中設定。如果函數拋出異常，Lambda 設置此報頭。如需有關的詳細資訊`X-Amz-Function-Error`，請參閱AWS Lambda 開發人員指南 AWS Lambda中的錯誤處理和自動重試。

以下是成功驗證的 Lambda 回應程式碼的簡單範例。

```
import json

def handler(event, context):
     #Add your validation logic here
     print("We passed!")
```

以下是驗證失敗的 Lambda 回應程式碼的簡單範例。

```
def handler(event, context):
     #Add your validation logic here
     raise Exception("Failure!")
```

這是另一個只有在組態參數為質數時才會驗證的例子。

```
function isPrime(value) {
    if (value < 2) {
        return false;
    }

    for (i = 2; i < value; i++) {
        if (value % i === 0) {
            return false;
        }
    }

    return true;
}

exports.handler = async function(event, context) {
    console.log('EVENT: ' + JSON.stringify(event, null, 2));
    const input = parseInt(Buffer.from(event.content, 'base64').toString('ascii'));
    const prime = isPrime(input);
    console.log('RESULT: ' + input + (prime ? ' is' : ' is not') + ' prime');
    if (!prime) {
        throw input + "is not prime";
    }
}
```

AWS AppConfig 呼叫`StartDeployment`和 `ValidateConfigurationActivity` API 作業時，會呼叫您的驗證 Lambda。您必須提供`appconfig.amazonaws.com`許可才能叫用 Lambda。如需詳細

資訊，請參閱授與 AWS 服務的函數存取權限。 AWS AppConfig 將驗證 Lambda 執行時間限制為 15 秒，包括啟動延遲。

結構定義驗證器

如果您在 SSM 文件中建立組態，則必須指定或建立該組態的 JSON 結構描述。JSON 結構描述可定義每個應用程式組態設定的允許屬性。JSON 結構描述的功能就像一組規則，以確保新的或更新後的組態設定符合您應用程式所需的最佳實務。請見此處範例。

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "$id$",
  "description": "BasicFeatureToggle-1",
  "type": "object",
  "additionalProperties": false,
  "patternProperties": {
      "[^\\s]+$": {
          "type": "boolean"
      }
  },
  "minProperties": 1
}
```

當您從 SSM 文件建立組態時，系統會自動驗證組態是否符合結構描述需求。如果不符合， AWS AppConfig 會傳回驗證錯誤。

Important

請注意下列有關 JSON 結構描述驗證程式的重要資訊：

- 儲存在 SSM 文件中的組態資料必須先對相關的 JSON 結構描述進行驗證，才能將組態新增至系統。SSM 參數不需要驗證方法，但建議您使用建立新的或更新的 SSM 參數組態的驗證檢查。 AWS Lambda
- SSM 文件中的組態會使用文`ApplicationConfiguration`件類型。相應的 JSON 結構描述會使用`ApplicationConfigurationSchema`文件類型。
- AWS AppConfig 支援內嵌結構描述的 JSON 結構描述 4.X 版本。如果您的應用程式設定需要不同版本的 JSON 結構描述，則您必須建立 Lambda 驗證程式。

在中建立功能旗標組態設定檔 AWS AppConfig

您可以使用功能旗標來啟用或停用應用程式中的功能，或使用旗標屬性來設定應用程式功能的不同特性。 AWS AppConfig 以功能旗標格式將功能旗標組態儲存在 AWS AppConfig 託管組態存放區中，該格式包含有關旗標和旗標屬性的資料和中繼資料。如需有關 AWS AppConfig 託管組態存放區的詳細資訊，請參閱一關於 AWS AppConfig 託管組態存放區節。

主題

- 建立功能旗標組態設定檔 (主控台)
- 建立功能旗標和功能旗標組態設定檔 (指令行)
- 的類型參考 AWS.AppConfig.FeatureFlags

開始之前

在下列程序中，您可以在選擇性的加密區段中選擇 AWS Key Management Service (AWS KMS) 金鑰。此客戶受管金鑰可讓您加密 AWS AppConfig 代管組態存放區中的新組態資料版本。如需有關此金鑰的詳細資訊，請參閱中的AWS AppConfig 支援客戶管理員金鑰AWS AppConfig 中的安全性。

下列程序也提供了將擴充功能與功能旗標組態設定檔相關聯的選項。擴充功能可增強您在建立或部署組態的 AWS AppConfig 工作流程期間，在不同點插入邏輯或行為的能力。如需詳細資訊，請參閱 關於 AWS AppConfig 擴展。

最後，在「特徵旗標屬性」區段中，當您輸入新特徵旗標的屬性詳細資訊時，可以指定限制。條件約束可確保任何未預期的屬性值不會部署到您的應用程式。 AWS AppConfig 支援下列類型的旗標屬性及其對應條件約束。

Type	限制條件	描述
：字串	規則運算式	字符串的正則表達式模式
	列舉	字串可接受的值清單
數字	下限	屬性的最小數值
	最大	屬性的最大數值
：布林值	無	無
字符串數組	規則運算式	數組元素的正則表達式模式

Type	限制條件	描述
	列舉	陣列元素的可接受值清單
數字陣列	下限	數組元素的最小數值
	最大	數組元素的最大數值

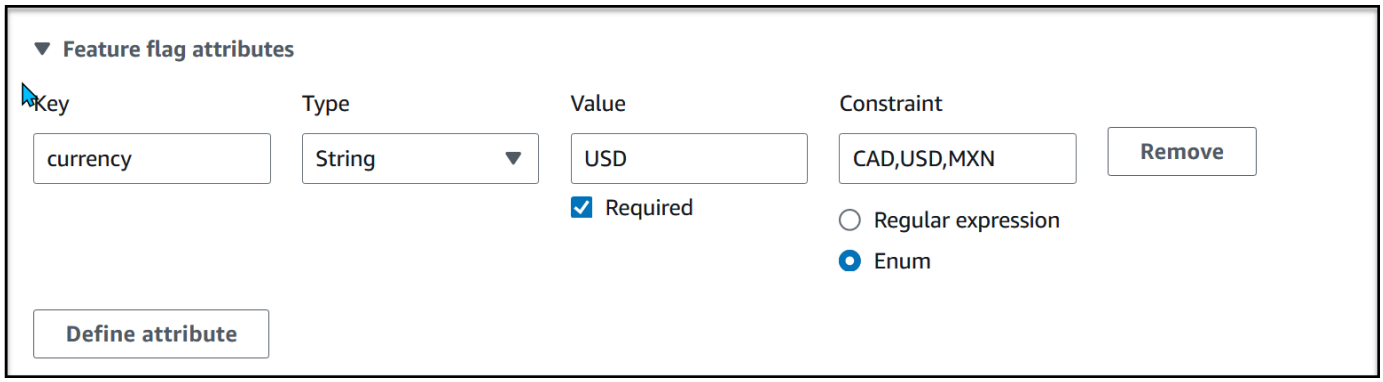
建立功能旗標組態設定檔 (主控台)

使用下列程序來使用 AWS AppConfig 主控台建立 AWS AppConfig 功能旗標組態設定檔。

建立組態描述檔

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在功能窗格中，選擇 [應用程式]，然後選擇您在其中建立的應用程式為您的應用程式建立命名空間 AWS AppConfig。
3. 選擇 [組態設定檔和功能旗標] 索引標籤，然後選擇 [建立組態]。
4. 在 [組態選項] 區段中，選擇 [功能旗標]。
5. 向下捲動。在「組態設定檔」區段中，對於「組態設定檔名稱」，輸入名稱。
6. (選擇性) 展開說明並輸入說明。
7. (選擇性) 展開其他選項，並視需要完成下列項目。
   a. 在「加密」清單中，從清單中選擇 AWS Key Management Service (AWS KMS) 金鑰。
   b. 在「關聯副檔名」區段中，從清單中選擇擴充功能。
   c. 在「標籤」區段中，選擇「新增標籤」，然後指定機碼和選用值。
8. 選擇下一步。
9. 在「功能旗標定義」區段中，輸入名稱做為「旗標名稱」。
10. 針對旗標金鑰，請輸入旗標識別碼，以區分相同組態設定檔中的旗標。相同設定描述檔中的旗標不能有相同的金鑰。建立旗標之後，您可以編輯旗標名稱，但無法編輯旗標金鑰。
11. (選擇性) 展開說明並輸入有關此旗標的資訊。
12. 選取這是短期旗標，並選擇性地選擇停用或刪除旗標的日期。請注意， AWS AppConfig 不會停用旗標。
13. 在「旗標屬性」段落中，選擇定義屬性。屬性可讓您在旗標內提供其他值。

14. 對於「機碼」，請指定旗標機碼，然後從「類型」清單中選擇其類型。您可以選擇性地根據指定的限制來驗證屬性值。下圖顯示範例。



選擇「定義屬性」以新增其他屬性。

> **Note**
>
> 記下以下資訊。
>
> - 對於屬性名稱，「啟用」一詞是保留的。您無法建立名為「已啟用」的功能旗標屬性。沒有其他保留字。
> - 功能標誌的屬性僅在啟用該標誌的情況下包含在`GetLatestConfiguration`響應中。
> - 指定旗標的旗標屬性索引鍵必須是唯一的。
> - 選取必要值以指定是否需要屬性值。

15. 在 [功能旗標值] 區段中，選擇 [啟用] 以啟用旗標。如果適用，請使用相同的切換來停用旗標到達指定的取代日期時停用該旗標。
16. 選擇下一步。
17. 在 [檢閱並儲存] 頁面上，確認旗標的詳細資料，然後儲存並繼續部署。

繼續執行「在中部署功能旗標和組態資料 AWS AppConfig」。

建立功能旗標和功能旗標組態設定檔 (指令行)

下列程序說明如何使用 AWS Command Line Interface (在 Linux 或 Windows 上) 或 Windows 專用工具 PowerShell 來建立 AWS AppConfig 功能旗標組態設定設定檔。如果您願意，您可以 AWS CloudShell 使用執行下列命令。如需詳細資訊，請參閱《 AWS CloudShell使用者指南》中的什麼是 AWS CloudShell ？。

逐步建立特徵旗標組態

1. 開啟 AWS CLI.
2. 建立將其類型指定為的功能旗標組態設定檔AWS.AppConfig.FeatureFlags。設定描述檔必須用hosted於位置 URI。

   Linux

   ```
   aws appconfig create-configuration-profile \
     --application-id The_application_ID \
     --name A_name_for_the_configuration_profile \
     --location-uri hosted \
     --type AWS.AppConfig.FeatureFlags
   ```

   Windows

   ```
   aws appconfig create-configuration-profile ^
     --application-id The_application_ID ^
     --name A_name_for_the_configuration_profile ^
     --location-uri hosted ^
     --type AWS.AppConfig.FeatureFlags
   ```

   PowerShell

   ```
   New-APPCConfigurationProfile `
     -Name A_name_for_the_configuration_profile `
     -ApplicationId The_application_ID `
     -LocationUri hosted `
     -Type AWS.AppConfig.FeatureFlags
   ```

3. 建立功能旗標組態資料。您的資料必須為 JSON 格式，且符合 AWS.AppConfig.FeatureFlags JSON 結構定義。如需資料架構的詳細資訊，請參閱的類型參考 AWS.AppConfig.FeatureFlags。
4. 使用 CreateHostedConfigurationVersion API 將功能旗標設定資料儲存至 AWS AppConfig。

Linux

```
aws appconfig create-hosted-configuration-version \
  --application-id The_application_ID \
  --configuration-profile-id The_configuration_profile_id \
  --content-type "application/json" \
  --content file://path/to/feature_flag_configuration_data \
  file_name_for_system_to_store_configuration_data
```

Windows

```
aws appconfig create-hosted-configuration-version ^
  --application-id The_application_ID ^
  --configuration-profile-id The_configuration_profile_id ^
  --content-type "application/json" ^
  --content file://path/to/feature_flag_configuration_data ^
  file_name_for_system_to_store_configuration_data
```

PowerShell

```
New-APPCHostedConfigurationVersion `
  -ApplicationId The_application_ID `
  -ConfigurationProfileId The_configuration_profile_id `
  -ContentType "application/json" `
  -Content file://path/to/feature_flag_configuration_data `
  file_name_for_system_to_store_configuration_data
```

這裡有一個 Linux 示例命令。

```
aws appconfig create-hosted-configuration-version \
  --application-id 1a2b3cTestApp \
  --configuration-profile-id 4d5e6fTestConfigProfile \
  --content-type "application/json" \
  --content Base64Content
```

content參數使用下列base64編碼資料。

```
{
  "flags": {
```

```
    "flagkey": {
      "name": "WinterSpecialBanner"
    }
  },
  "values": {
    "flagkey": {
      "enabled": true
    }
  },
  "version": "1"
}
```

系統會傳回相關資訊，如下所示。

Linux

```
{
   "ApplicationId"          : "1a2b3cTestApp",
   "ConfigurationProfileId" : "4d5e6fTestConfigProfile",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
}
```

Windows

```
{
   "ApplicationId"          : "1a2b3cTestApp",
   "ConfigurationProfileId" : "4d5e6fTestConfigProfile",
   "VersionNumber"          : "1",
   "ContentType"            : "application/json"
}
```

PowerShell

```
ApplicationId          : 1a2b3cTestApp
ConfigurationProfileId : 4d5e6fTestConfigProfile
VersionNumber          : 1
ContentType            : application/json
```

包service_returned_content_file含您的組態資料，其中包含一些 AWS AppConfig 產生的中繼資料。

> **Note**
>
> 建立託管組態版本時，請 AWS AppConfig 驗證資料是否符合 AWS.AppConfig.FeatureFlags JSON 結構描述。 AWS AppConfig 另外驗證資料中的每個特徵旗標屬性是否符合您為這些屬性定義的限制。

的類型參考 AWS.AppConfig.FeatureFlags

使用 AWS.AppConfig.FeatureFlags JSON 結構定義作為參考，以建立功能旗標組態資料。

```
{
      "$schema": "http://json-schema.org/draft-07/schema#",
      "definitions": {
        "flagSetDefinition": {
          "type": "object",
          "properties": {
            "version": {
              "$ref": "#/definitions/flagSchemaVersions"
            },
            "flags": {
              "$ref": "#/definitions/flagDefinitions"
            },
            "values": {
              "$ref": "#/definitions/flagValues"
            }
          },
          "required": ["version", "flags"],
          "additionalProperties": false
        },
        "flagDefinitions": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-]{0,63}$": {
              "$ref": "#/definitions/flagDefinition"
            }
          },
```

```
      "maxProperties": 100,
      "additionalProperties": false
    },
    "flagDefinition": {
      "type": "object",
      "properties": {
        "name": {
          "$ref": "#/definitions/customerDefinedName"
        },
        "description": {
          "$ref": "#/definitions/customerDefinedDescription"
        },
        "_createdAt": {
          "type": "string"
        },
        "_updatedAt": {
          "type": "string"
        },
        "_deprecation": {
          "type": "object",
          "properties": {
            "status": {
              "type": "string",
              "enum": ["planned"]
            }
          },
         "additionalProperties": false
        },
        "attributes": {
          "$ref": "#/definitions/attributeDefinitions"
        }
      },
      "additionalProperties": false
    },
    "attributeDefinitions": {
      "type": "object",
      "patternProperties": {
        "^[a-z][a-zA-Z\\d-_]{0,63}$": {
          "$ref": "#/definitions/attributeDefinition"
        }
      },
      "maxProperties": 25,
      "additionalProperties": false
    },
```

```
        "attributeDefinition": {
          "type": "object",
          "properties": {
            "description": {
              "$ref": "#/definitions/customerDefinedDescription"
            },
            "constraints": {
              "oneOf": [
                { "$ref": "#/definitions/numberConstraints" },
                { "$ref": "#/definitions/stringConstraints" },
                { "$ref": "#/definitions/arrayConstraints" },
                { "$ref": "#/definitions/boolConstraints" }
              ]
            }
          },
          "additionalProperties": false
        },
        "flagValues": {
          "type": "object",
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/flagValue"
            }
          },
          "maxProperties": 100,
          "additionalProperties": false
        },
        "flagValue": {
          "type": "object",
          "properties": {
            "enabled": {
              "type": "boolean"
            },
            "_createdAt": {
              "type": "string"
            },
            "_updatedAt": {
              "type": "string"
            }
          },
          "patternProperties": {
            "^[a-z][a-zA-Z\\d-_]{0,63}$": {
              "$ref": "#/definitions/attributeValue",
              "maxProperties": 25
```

```
            }
          },
          "required": ["enabled"],
          "additionalProperties": false
        },
        "attributeValue": {
          "oneOf": [
            { "type": "string", "maxLength": 1024 },
            { "type": "number" },
            { "type": "boolean" },
            {
              "type": "array",
              "oneOf": [
                {
                  "items": {
                    "type": "string",
                    "maxLength": 1024
                  }
                },
                {
                  "items": {
                    "type": "number"
                  }
                }
              ]
            }
          ],
          "additionalProperties": false
        },
        "stringConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["string"]
      },
            "required": {
              "type": "boolean"
            },
            "pattern": {
              "type": "string",
              "maxLength": 1024
            },
            "enum": {
```

```
            "type": "array",
            "maxLength": 100,
            "items": {
              "oneOf": [
                {
                  "type": "string",
                  "maxLength": 1024
                },
                {
                  "type": "integer"
                }
              ]
            }
          }
        },
        "required": ["type"],
        "not": {
          "required": ["pattern", "enum"]
        },
        "additionalProperties": false
      },
      "numberConstraints": {
        "type": "object",
        "properties": {
          "type": {
            "type": "string",
            "enum": ["number"]
    },
          "required": {
            "type": "boolean"
          },
          "minimum": {
            "type": "integer"
          },
          "maximum": {
            "type": "integer"
          }
        },
        "required": ["type"],
        "additionalProperties": false
      },
      "arrayConstraints": {
        "type": "object",
        "properties": {
```

```
            "type": {
              "type": "string",
              "enum": ["array"]
      },
            "required": {
              "type": "boolean"
            },
            "elements": {
              "$ref": "#/definitions/elementConstraints"
      }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "boolConstraints": {
          "type": "object",
          "properties": {
            "type": {
              "type": "string",
              "enum": ["boolean"]
      },
            "required": {
              "type": "boolean"
            }
          },
          "required": ["type"],
          "additionalProperties": false
        },
        "elementConstraints": {
          "oneOf": [
            { "$ref": "#/definitions/numberConstraints" },
            { "$ref": "#/definitions/stringConstraints" }
          ]
        },
        "customerDefinedName": {
          "type": "string",
          "pattern": "^[^\\n]{1,64}$"
        },
        "customerDefinedDescription": {
          "type": "string",
          "maxLength": 1024
        },
        "flagSchemaVersions": {
          "type": "string",
```

```
          "enum": ["1"]
        }
      },
      "type": "object",
      "$ref": "#/definitions/flagSetDefinition",
      "additionalProperties": false
    }
```

> **Important**
>
> 若要擷取功能旗標設定資料，您的應用程式必須呼叫 GetLatestConfiguration API。您無法通過調GetConfiguration用（已棄用）來檢索功能標誌配置數據。如需詳細資訊，請參閱 AWS AppConfig API 參考中的GetLatest組態。

當您的應用程式呼叫 GetLatestConfiguration 並接收新部署的組態時，會移除定義功能旗標和屬性的資訊。簡化的 JSON 包含與您指定的每個標誌鍵匹配的鍵的映射鍵。簡化的 JSON 也包含enabled屬性的true或false對應值。如果標誌設enabled定為true，則旗標的任何屬性也會出現。下列 JSON 結構描述了 JSON 輸出的格式。

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "patternProperties": {
    "^[a-z][a-zA-Z\\d-_]{0,63}$": {
      "$ref": "#/definitions/attributeValuesMap"
    }
  },
  "maxProperties": 100,
  "additionalProperties": false,
  "definitions": {
    "attributeValuesMap": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
        }
      },
      "required": ["enabled"],
      "patternProperties": {
        "^[a-z][a-zA-Z\\d-_]{0,63}$": {
          "$ref": "#/definitions/attributeValue"
```

```
        }
      },
      "maxProperties": 25,
      "additionalProperties": false
    },
    "attributeValue": {
      "oneOf": [
        { "type": "string","maxLength": 1024 },
        { "type": "number" },
        { "type": "boolean" },
        {
          "type": "array",
          "oneOf": [
            {
              "items": {
                "oneOf": [
                  {
                    "type": "string",
                    "maxLength": 1024
                  }
                ]
              }
            },
            {
              "items": {
                "oneOf": [
                  {
                    "type": "number"
                  }
                ]
              }
            }
          ]
        }
      ],
      "additionalProperties": false
    }
  }
}
```

在中建立任意格式組態設定檔 AWS AppConfig

配置描述檔包括可讓 AWS AppConfig 您在其儲存位置尋找組態資料的 URI 以及設定檔類型。 AWS AppConfig 支援兩種組態設定檔類型：功能旗標和自由格式組態。功能標誌配置文件將其數據存儲在 AWS AppConfig 託管的配置存儲中，並且 URI 很簡單`hosted`。對於自由格式組態設定檔，您可以將資料儲存在 AWS AppConfig 託管組態存放區或下列任何一項 AWS 服務和 Systems Manager 功能中：

位置	支援的檔案類型
AWS AppConfig 託管組態存放區	YAML、JSON 和文字 (如果使用 AWS Management Console. 使用 AWS AppConfig CreateHostedConfigurationVersionAPI 動作新增的任何檔案類型。
Amazon Simple Storage Service (Amazon S3)	任何
AWS CodePipeline	管道（由服務定義）
AWS Secrets Manager	密碼（由服務定義）
AWS Systems Manager 參數存放區	標準和安全字串參數 (由參數存放區定義)
AWS Systems Manager 文件存放區 (SSM 文件)	矢量, JSON, 文本

組態設定檔也可以包含選用的驗證程式，以確保您的組態資料在語法和語義上都是正確的。 AWS AppConfig 當您啟動部署時，會使用驗證程式執行檢查。如果偵測到任何錯誤，部署會在對組態的目標進行任何變更之前停止。

> **Note**
>
> 如果可能，我們建議您在託管的組態存放區中 AWS AppConfig 託管您的組態資料，因為它提供了最多的功能和增強功能。

對於儲存在 AWS AppConfig 主控組態存放區或 SSM 文件中的自由格式組態，您可以在建立組態設定檔時使用 Systems Manager 主控台來建立自由格式組態。本主題稍後將說明此程序。

對於存放在參數存放區、Secrets Manager 或 Amazon S3 中的自由格式組態，您必須先建立參數、密碼或物件，然後將其存放在相關組態存放區中。儲存組態資料後，請使用本主題中的程序來建立組態設定檔。

主題

關於組態存放區配額和限制

支援的組態存放區 AWS AppConfig 具有下列配額和限制。

	AWS AppConfig 託管組態存放區	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager 文件存儲	AWS CodePipeline
組態大小限制	預設值為 2 MB，最大值為 4 MB	2 MB 由 S3 強制執行 AWS AppConfig，而不是 S3	4 KB (免費方案)/8 KB (進階參數)	64 KB	64 KB	2 MB 由強制執行 AWS AppConfig，而不是 CodePipeline
資源儲存限制	1 GB	無限制	10,000 個參數 (免費方案)/100,000 個參數 (進階參數)	500,000	500 份文件	受每個應用程式的設定描述檔數量限制 (每個應用程式 100 個設定檔)

	AWS AppConfig 託管組態存放區	Amazon S3	Systems Manager Parameter Store	AWS Secrets Manager	Systems Manager 文件存儲	AWS CodePipeline
伺服器端加密	是	SSE-S3, 三公里	是	是	否	是
AWS CloudFormation 支持	是	不適用於建立或更新資料	是	是	否	是
定價	免費	查看 Amazon S3 定價	查看AWS Systems Manager 定價	查看AWS Secrets Manager 定價	免費	查看AWS CodePipeline 定價

關於 AWS AppConfig 託管組態存放區

AWS AppConfig 包含內部或託管的組態存放區。組態必須為 2 MB 或更小。與其他組態存放區選項相比， AWS AppConfig 託管組態存放區提供下列優點。

- 您不需要設定其他服務，例如 Amazon Simple Storage Service (Amazon S3) 或參數存放區。
- 您不需要設定 AWS Identity and Access Management (IAM) 許可即可使用組態存放區。
- 您可以將組態存放為 YAML、JSON 或文字文件。
- 使用存放區無需付費。
- 您可以建立組態，並在建立組態描述檔時將其新增至存放區。

關於存放在 Amazon S3 中的組態

您可以將組態存放在亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體中。在您建立組態描述檔時，您可以指定指向儲存貯體中單一 S3 物件的 URI。您也可以指定 (IAM) 角色的 Amazon 資源名稱 AWS Identity and Access Management (ARN)，以 AWS AppConfig 授予取得物件的權限。在為 Amazon S3 物件建立組態設定檔之前，請注意以下限制。

限制	詳細資訊
大小	存放為 S3 物件的組態大小上限為 1 MB。
物件加密	組態設定檔可以鎖定 SSE-S3 和 SSE-KMS 加密物件的目標。
儲存類別	AWS AppConfig 支援下列 S3 儲存類別：STANDARDINTELLIGENT_TIERING 、REDUCED_REDUNDANCY 、STANDARD_IA 、和ONEZONE_IA 。不支援下列類別：所有 S3 Glacier 類別 (GLACIER 和 DEEP_ARCHIVE)。
版本控制	AWS AppConfig 要求 S3 物件使用版本控制。

為存放為 Amazon S3 物件的組態設定許可

當您為存放為 S3 物件的組態建立組態設定檔時，必須為 IAM 角色指定 ARN，以提供取得物件的 AWS AppConfig 權限。角色必須包含下列許可。

存取 S3 物件的許可

- S3：GetObject
- S3：GetObject版本

列出 S3 儲存貯體的許可

S3：ListAllMyBuckets

存放物件的 S3 儲存貯體的存取許可

- S3：GetBucket位置
- s3：GetBucket版本控制
- S3：ListBucket
- S3：ListBucket版本

請完成下列程序來建立可取 AWS AppConfig 得存放在 S3 物件中組態的角色。

建立存取 S3 物件的 IAM 政策

使用下列程序建立 IAM 政策，以取 AWS AppConfig 得存放在 S3 物件中的組態。

若要建立存取 S3 物件的 IAM 政策

1. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
3. 在建立政策頁面上，選擇 JSON 標籤。
4. 使用 S3 儲存貯體及組態物件的相關資訊，更新以下範例政策。然後將政策貼入 JSON 標籤上的文字欄位中。以您自己的資訊取代*#####*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/my-configurations/my-
configuration.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
```

```
    }
  ]
}
```

5. 選擇檢閱政策。
6. 在 Review policy (檢閱政策) 頁面上，於 Name (名稱) 方塊中輸入名稱，然後輸入描述。
7. 選擇建立政策。系統會讓您回到 Roles (角色) 頁面。

建立存取 S3 物件的 IAM 角色

使用下列程序建立 IAM 角色，以取 AWS AppConfig 得存放在 S3 物件中的組態。

若要建立存取 Amazon S3 物件的 IAM 角色

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 [選取信任實體的類型] 區段中，選擇 [AWS 服務]。
4. 在 Choose a use case (選擇使用案例) 區段中，選擇位於 Common use cases (常見使用案例) 下方的 EC2，然後選擇 Next: Permissions (下一步：許可)。
5. 在 Attach permissions policy (連接許可政策) 頁面上，於搜尋方塊中輸入您在上一個程序中建立的政策名稱。
6. 選擇政策，然後選擇 Next: Tags (下一步：標籤)。
7. 在 [新增標記 (選擇性)] 頁面上，輸入金鑰和選用值，然後選擇 [下一步：複查]。
8. 在 Review (檢閱) 頁面上，在 Role name (角色名稱) 欄位中輸入名稱，然後輸入描述。
9. 選擇 Create role (建立角色)。系統會讓您回到 Roles (角色) 頁面。
10. 在 Roles (角色) 頁面，選擇您剛建立的角色，以開啟 Summary (摘要) 頁面。請記下 Role Name (角色名稱) 和 Role ARN (角色 ARN)。您將會在本主題中稍後建立組態描述檔時指定角色 ARN。

建立信任關係

使用下列程序來設定您剛建立的角色以信任 AWS AppConfig。

新增信任關係

1. 在您剛建立之角色的 Summary (摘要) 頁面中，選擇 Trust Relationships (信任關係) 索引標籤，然後著選擇 Edit Trust Relationship (編輯信任關係)。

2. 刪除 "ec2.amazonaws.com"，然後新增 "appconfig.amazonaws.com"，如以下範例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. 選擇 Update Trust Policy (更新信任政策)。

建立自由格式組態與組態設定描述檔

本節說明如何建立自由格式組態與組態設定檔。開始之前，請注意下列資訊。

- 下列程序要求您指定 IAM 服務角色， AWS AppConfig 以便存取您選擇的組態存放區中的組態資料。如果您使用 AWS AppConfig 託管組態存放區，則不需要此角色。如果您選擇 S3、參數存放區或 Systems Manager 文件存放區，則必須選擇現有的 IAM 角色，或選擇讓系統自動為您建立角色的選項。如需此角色的詳細資訊，請參閱關於組態設定檔 IAM 角色。
- 下列程序也提供了將擴充功能與功能旗標組態設定檔相關聯的選項。擴充功能可增強您在建立或部署組態的 AWS AppConfig 工作流程期間，在不同點插入邏輯或行為的能力。如需詳細資訊，請參閱關於 AWS AppConfig 擴展。
- 如果要為存放在 S3 中的組態建立組態描述檔，則必須設定許可。如需使用 S3 做為組態存放區的許可和其他要求的詳細資訊，請參閱關於存放在 Amazon S3 中的組態。
- 如果您想使用驗證程式，請檢閱使用它們的詳細資訊和要求。如需詳細資訊，請參閱 關於驗證器。

主題

- 建立 AWS AppConfig 自由格式組態設定檔 (主控台)
- 建立 AWS AppConfig 自由格式組態設定檔 (命令列)

建立 AWS AppConfig 自由格式組態設定檔 (主控台)

使用下列程序來使用主控台建立 AWS AppConfig 自由格式組態設定檔和 (選擇性) 自由式組態。 AWS Systems Manager

建立自由格式組態設定檔

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在功能窗格中，選擇 [應用程式]，然後選擇您在其中建立的應用程式為您的應用程式建立命名空間 AWS AppConfig。
3. 選擇 [組態設定檔和功能旗標] 索引標籤，然後選擇 [建立組態]。
4. 在 [組態選項] 區段中，選擇 [自由格式組態]。
5. 在「組態設定檔名稱」中，輸入組態設定檔的名稱。
6. (選擇性) 展開說明並輸入說明。
7. (選擇性) 展開其他選項，並視需要完成下列項目。
   a. 在「關聯副檔名」區段中，從清單中選擇擴充功能。
   b. 在「標籤」區段中，選擇「新增標籤」，然後指定機碼和選用值。
8. 選擇下一步。
9. 在 [指定組態資料] 頁面的 [組態取消] 區段中，選擇一個選項。
10. 完成所選選項的欄位，如下表所述。

選擇的選項	詳細資訊
AWS AppConfig 託管配置	選擇 [文字]、[JSON] 或 [YAML]，然後在欄位中輸入您的設定。轉到此程序中的步驟 12。
Amazon S3 對象	在 S3 物件來源欄位中輸入物件 URI，然後移至此程序中的步驟 11。
AWS CodePipeline	選擇 [下一步]，然後移至此程序中的步驟 12。
Secrets Manager 秘密	從清單中選擇密碼，前往此程序中的步驟 11。

選擇的選項	詳細資訊
AWS Systems Manager parameter	從清單中選擇參數，然後移至此程序中的步驟 11。
AWS Systems Manager 文件	1. 從清單中選擇文件，或選擇「建立新文件」。 2. 如果您選擇「建立新文件」，請輸入名稱做為「文件名稱」。選擇性地展開版本名稱，然後輸入文件版本的名稱。 3. 對於應用程式組態結構描述，請從清單中選擇 JSON 結構描述或選擇 [建立結構描述]。如果您選擇建立結構描述，Systems Manager 會開啟 [建立結構描述] 頁 輸入結構描述詳細資訊，然後選擇 [建立應用程式組態結構描述]。 4. 在 Content (內容) 區段中，選擇 YAML 或 JSON，然後在欄位中輸入組態資料。

11. 在 [服務角色] 區段中，選擇 [新增服務角色] 以 AWS AppConfig 建立可存取設定資料的 IAM 角色。 AWS AppConfig 根據您先前輸入的名稱，自動填入「角色名稱」欄位。或者，選擇現有的服務角色。使用 Role ARN (角色 ARN) 清單選擇角色。
12. 或者，在 [新增驗證程式] 頁面上，選擇 [JSON 結構描述] 或。AWS Lambda如果您選擇 JSON Schema (JSON 結構描述)，請在欄位中輸入 JSON 結構描述。如果您選擇 AWS Lambda，請從清單中選擇 Amazon Resource Name (ARN) 和版本。

> **Important**
>
> 儲存在 SSM 文件中的組態資料必須先對相關的 JSON 結構描述進行驗證，才能將組態新增至系統。SSM 參數不需要驗證方法，但建議您使用建立新的或更新的 SSM 參數組態的驗證檢查。 AWS Lambda

13. 選擇下一步。
14. 在 [檢閱並儲存] 頁面上，選擇 [儲存並繼續部署]。

> **Important**
>
> 如果您建立的組態設定檔 AWS CodePipeline，則必須在中建立指 CodePipeline 定 AWS AppConfig 為部署提供者的管線。您不需要執行在中部署功能旗標和組態資料 AWS AppConfig。不過，您必須設定用戶端來接收應用程式組態更新，如中所述透過直接呼叫 API 擷取組態。如需有關建立指定 AWS AppConfig 為部署提供者的管線的資訊，請參閱《使用指南》中的教學課程：建立用 AWS AppConfig 作部署提供AWS CodePipeline 者的管線。

繼續執行「在中部署功能旗標和組態資料 AWS AppConfig」。

建立 AWS AppConfig 自由格式組態設定檔 (命令列)

下列程序說明如何使用 AWS CLI (在 Linux 或 Windows 上) 或 AWS Tools for PowerShell 建立 AWS AppConfig 自由格式組態設定檔。如果您願意，您可以 AWS CloudShell 使用執行下列命令。如需詳細資訊，請參閱《 AWS CloudShell使用者指南》中的什麼是AWS CloudShell ？。

> **Note**
>
> 對於託管組態存放區中託 AWS AppConfig 管的自由格式組態，您可以`hosted`為位置 URI 指定。

若要使用建立組態設定檔 AWS CLI

1. 開啟 AWS CLI.
2. 執行下列命令來建立自由格式組態設定檔。

   Linux

```
aws appconfig create-configuration-profile \
  --application-id The_application_ID \
  --name A_name_for_the_configuration_profile \
  --description A_description_of_the_configuration_profile \
  --location-uri A_URI_to_locate_the_configuration or hosted \
  --retrieval-role-
arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified
 \
  --tags User_defined_key_value_pair_metadata_of_the_configuration_profile \
```

```
  --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Windows

```
aws appconfig create-configuration-profile ^
  --application-id The_application_ID ^
  --name A_name_for_the_configuration_profile ^
  --description A_description_of_the_configuration_profile ^
  --location-uri A_URI_to_locate_the_configuration or hosted  ^
  --retrieval-role-
arn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_the_specified
 ^
  --tags User_defined_key_value_pair_metadata_of_the_configuration_profile ^
  --validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

PowerShell

```
New-APPCConfigurationProfile `
  -Name A_name_for_the_configuration_profile `
  -ApplicationId The_application_ID `
  -Description Description_of_the_configuration_profile `
  -LocationUri A_URI_to_locate_the_configuration or hosted `
  -
RetrievalRoleArn The_ARN_of_the_IAM_role_with_permission_to_access_the_configuration_at_
 `
  -
Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_configuration_profile
 `
  -Validators "Content=JSON_Schema_content_or_the_ARN_of_an_AWS
 Lambda_function,Type=JSON_SCHEMA or LAMBDA"
```

Important

記下以下重要資訊。

- 如果您建立的組態設定檔 AWS CodePipeline，則必須在中建立指 CodePipeline 定 AWS AppConfig 為部署提供者的管線。您不需要執行在中部署功能旗標和組態資料 AWS AppConfig。不過，您必須設定用戶端來接收應用程式組態更新，如中所述透過直接呼叫

API 擷取組態。如需有關建立指定 AWS AppConfig 為部署提供者的管線的資訊，請參閱《使用指南》中的教學課程：建立用 AWS AppConfig 作部署提供AWS CodePipeline 者的管線。

- 如果您在 AWS AppConfig 託管組態存放區中建立組態，則可以使用 CreateHostedConfigurationVersionAPI 作業建立新版本的組態。若要檢視此 API 作業的 AWS CLI 詳細資訊和範例命令，請參閱《命令參考》中的建立託管配置版本。AWS CLI

繼續執行「在中部署功能旗標和組態資料 AWS AppConfig」。

其他組態資料來源

本主題包括與之整合的其他 AWS 服務的相關資訊 AWS AppConfig。

AWS AppConfig 與整合 AWS Secrets Manager

Secrets Manager 可協助您安全地加密、儲存和擷取資料庫和其他服務的認證。除了在應用程式中對憑證進行硬式編碼，您可以在需要時呼叫 Secrets Manager 以擷取憑證。Secrets Manager 可讓您輪換和管理密碼的存取權，協助您保護對 IT 資源和資料的存取。

建立自由格式組態設定檔時，您可以選擇 Secrets Manager 作為組態資料的來源。在建立設定描述檔之前，您必須使用 Secrets Manager 加載並建立密碼。如需有關 Secrets Manager 的詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。如需建立使用 Secrets Manager 之組態設定描述檔的相關資訊，請參閱在中建立特徵旗標和任意格式組態資料 AWS AppConfig。

在中部署功能旗標和組態資料 AWS AppConfig

建立使用功能旗標和自由格式組態資料所需的人工因素後，您可以建立新部署。建立新部署時，請指定下列資訊：

- 應用程式識別碼
- 設定描述檔識別碼
- 配置版本
- 您要在其中部署組態資料的環境 ID
- 部署策略 ID，定義您希望變更生效的速度
- 使用客戶管理的金鑰加密資料的 AWS Key Management Service (AWS KMS) 金鑰 ID。

當您呼叫 StartDeploymentAPI 動作時，請 AWS AppConfig 執行下列工作：

1. 使用設定描述檔中的位置 URI，從基礎資料存放區擷取組態資料。
2. 使用您在建立組態設定檔時指定的驗證程式，驗證組態資料在語法和語意上是否正確。
3. 緩存數據的副本，以便您的應用程序可以檢索它。此快取副本稱為已部署的資料。

AWS AppConfig 與 Amazon 集成 CloudWatch 以監控部署。如果部署在中觸發警示 CloudWatch，則 AWS AppConfig 會自動復原部署，將對應用程式使用者的影響降到最低。

主題

- 使用部署策略
- 部署組態
- AWS AppConfig 部署整合 CodePipeline

使用部署策略

部署策略可讓您在數分鐘或數小時內緩慢發行生產環境的變更。 AWS AppConfig 部署策略會定義組態部署的下列重要層面。

<table>
<tr><th>設定</th><th>描述</th></tr>
<tr><td>部署類型</td><td>部署類型會定義組態部署或推出的方式。 AWS AppConfig 支援線性和指數部署類型。

• 線性：針對此類型，會以平均分佈在部署中的成長因子遞增來 AWS AppConfig 處理部署。以下是使用 20% 線性成長的 10 小時部署時間表範例：

<table>
<tr><th>經過時間</th><th>部署進度</th></tr>
<tr><td>0 小時</td><td>0%</td></tr>
<tr><td>2 小時</td><td>20%</td></tr>
<tr><td>4 小時</td><td>40%</td></tr>
<tr><td>6 小時</td><td>60%</td></tr>
<tr><td>8 小時</td><td>80%</td></tr>
<tr><td>10 小時</td><td>100%</td></tr>
</table>
• 指數：對於此類型， AWS AppConfig 會使用下列公式以指數方式處理部署：<code>G*(2^N)</code>。在此公式中，<code>G</code> 是使用者指定的步驟百分比，而 <code>N</code> 是將組態部署至所有目標的步驟數目。例如，如果您指定的成長係數為 2，則系統將如下所示推出組態：

<pre>2*(2^0)
2*(2^1)
2*(2^2)</pre>
以數字表示，部署的推出方式如下：2% 的目標、4% 的目標、8% 的目標，並持續執行，直到組態已部署至所有目標為止。</td></tr>
</table>

設定	描述
步驟百分比 (成長係數)	這個設定會指定要在部署的每個步驟期間鎖定的呼叫端百分比。 **Note** 在開發套件和 AWS AppConfig API 參考中，`step percentage` 稱為 `growth factor`。
部署時間	此設定指定部 AWS AppConfig 署至主機的時間量。這不是逾時值。它是以間隔處理部署的時段。
封裝時間	此設定指定將組態部署到 100% 的目標後，Amazon CloudWatch 警示的 AWS AppConfig 監控時間長度，然後再考慮部署完成。如果在此期間觸發警示， AWS AppConfig 會復原部署。您必須設定權限，才 AWS AppConfig 能根據 CloudWatch 警示回復。如需詳細資訊，請參閱 (選擇性) 根據 CloudWatch 警示設定復原的權限。

您可以選擇隨附的預先定義策略， AWS AppConfig 也可以自行建立策略。

主題

- 預先定義的部署策略
- 建立部署策略

預先定義的部署策略

AWS AppConfig 包含預先定義的部署策略，可協助您快速部署組態。您不用建立自己的策略，而可以在部署組態時選擇下列其中一項。

部署策略	描述
AppConfig. 線性 20 6 分鐘 PercentEvery	AWS 推薦： 此策略每六分鐘會將組態部署到 20% 的所有目標，以進行 30 分鐘的部署。系統會監控 Amazon CloudWatch 警報 30 分鐘。如果此時沒有收到任何警示，表示部署已完成。如果在此期間觸發警示，請 AWS AppConfig 復原部署。 我們建議您將此策略用於生產部署，因為它與 AWS 最佳實務保持一致，並且由於部署安全性的長時間和烘烤時間，因此還加強了部署安全性。
AppConfig. 加拿大百分比 20 分鐘	AWS 推薦： 此策略使用 10% 成長係數超過 20 分鐘，以指數方式處理部署。系統會監控 CloudWatch 警報 10 分鐘。如果此時沒有收到任何警示，表示部署已完成。如果在此期間觸發警示，請 AWS AppConfig 復原部署。 我們建議將此策略用於生產部署，因為它符合組態部署的 AWS 最佳實務。
AppConfig.AllAtOnce	快速： 此策略會立即將組態部署到所有目標。系統會監控 CloudWatch 警報 10 分鐘。如果此時沒有收到任何警示，表示部署已完成。如果在此期間觸發警示， AWS AppConfig 會復原部署。
AppConfig. 線性線性 50 秒 PercentEvery	測試/演示： 此策略每隔 30 秒會將組態部署到所有目標的一半，以進行一分鐘部署。系統會監控 Amazon CloudWatch 警報 1 分鐘。如果此時沒有收到任

部署策略	描述
	何警示，表示部署已完成。如果在此期間觸發警示，請 AWS AppConfig 復原部署。 我們建議您僅將此策略用於測試或示範目的，因為它的持續時間和封裝時間很短。

建立部署策略

如果您不想使用其中一種預先定義的部署策略，您可以建立自己的部署策略。您最多可以建立 20 個部署策略。部署組態時，您可以選擇最適合應用程式和環境的部署策略。

建立 AWS AppConfig 部署策略 (主控台)

使用下列程序，使用 AWS Systems Manager 主控台建立 AWS AppConfig 部署策略。

建立部署策略

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在瀏覽窗格中，選擇 [部署策略]，然後選擇 [建立部署策略]。
3. 對於 Name (名稱)，輸入部署策略的名稱。
4. 對於 Description (描述)，輸入有關部署策略的資訊。
5. 對於 Deployment type (部署類型)，請選擇類型。
6. 對於 Step percentage (步驟百分比)，選擇要在部署的每個步驟期間鎖定的呼叫端百分比。
7. 對於 Deployment time (部署時間)，輸入部署的總持續時間 (以分鐘或小時為單位)。
8. 對於烘焙時間，請輸入總時間 (以分鐘或小時為單位) 以監控 Amazon CloudWatch 警示，然後再繼續部署的下一個步驟或考慮完成部署。
9. 在 Tags (標籤) 區段中，輸入一個鍵和一個選用值。您可以為資源指定最多 50 個標籤。
10. 選擇 Create deployment strategy (建立部署策略)。

> **Important**
>
> 如果您建立的組態設定檔 AWS CodePipeline，則必須在中建立指 CodePipeline 定 AWS AppConfig 為部署提供者的管線。您不需要執行部署組態。不過，您必須設定用戶端來接收應

用程式組態更新，如中所述透過直接呼叫 API 擷取組態。如需有關建立指定 AWS AppConfig 為部署提供者的管線的資訊，請參閱《使用指南》中的教學課程：建立用 AWS AppConfig 作部署提供AWS CodePipeline 者的管線。

繼續執行「部署組態」。

建立 AWS AppConfig 部署策略 (命令列)

下列程序說明如何使用 AWS CLI (在 Linux 或 Windows 上) 或 AWS Tools for PowerShell 建立 AWS AppConfig 部署策略。

若要逐步建立部署策略

1. 開啟 AWS CLI.
2. 執行下列命令以建立部署策略。

   Linux

```
aws appconfig create-deployment-strategy \
  --name A_name_for_the_deployment_strategy \
  --description A_description_of_the_deployment_strategy \
  --deployment-duration-in-minutes Total_amount_of_time_for_a_deployment_to_last
 \
  --final-bake-time-in-minutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 \
  --growth-
factor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_interva
 \
  --growth-
type The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_time
 \
  --replicate-
to To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document \
  --tags User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

   Windows

```
aws appconfig create-deployment-strategy ^
  --name A_name_for_the_deployment_strategy ^
```

```
  --description A_description_of_the_deployment_strategy ^
  --deployment-duration-in-minutes Total_amount_of_time_for_a_deployment_to_last
 ^
  --final-bake-time-in-minutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 ^
  --growth-
factor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_interva
 ^
  --growth-
type The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_time
 ^
  --name A_name_for_the_deployment_strategy ^
  --replicate-
to To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document ^
  --tags User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

PowerShell

```
New-APPCDeploymentStrategy `
  --Name A_name_for_the_deployment_strategy `
  --Description A_description_of_the_deployment_strategy `
  --DeploymentDurationInMinutes Total_amount_of_time_for_a_deployment_to_last `
  --FinalBakeTimeInMinutes Amount_of_time_AWS
 AppConfig_monitors_for_alarms_before_considering_the_deployment_to_be_complete
 `
  --
GrowthFactor The_percentage_of_targets_to_receive_a_deployed_configuration_during_each_i
 `
  --
GrowthType The_linear_or_exponential_algorithm_used_to_define_how_percentage_grows_over_
 `
  --
ReplicateTo To_save_the_deployment_strategy_to_a_Systems_Manager_(SSM)_document
 `
  --
Tag Hashtable_type_User_defined_key_value_pair_metadata_of_the_deployment_strategy
```

系統會傳回相關資訊，如下所示。

Linux

```
{
   "Id": "Id of the deployment strategy",
   "Name": "Name of the deployment strategy",
   "Description": "Description of the deployment strategy",
   "DeploymentDurationInMinutes": "Total amount of time the deployment lasted",
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
   "GrowthFactor": "The percentage of targets that received a deployed
 configuration during each interval",
   "FinalBakeTimeInMinutes": "The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete",
   "ReplicateTo": "The Systems Manager (SSM) document where the deployment
 strategy is saved"
}
```

Windows

```
{
   "Id": "Id of the deployment strategy",
   "Name": "Name of the deployment strategy",
   "Description": "Description of the deployment strategy",
   "DeploymentDurationInMinutes": "Total amount of time the deployment lasted",
   "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
   "GrowthFactor": "The percentage of targets that received a deployed
 configuration during each interval",
   "FinalBakeTimeInMinutes": "The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete",
   "ReplicateTo": "The Systems Manager (SSM) document where the deployment
 strategy is saved"
}
```

PowerShell

```
ContentLength               : Runtime of the command
DeploymentDurationInMinutes : Total amount of time the deployment lasted
Description                 : Description of the deployment strategy
FinalBakeTimeInMinutes      : The amount of time AWS AppConfig monitored for
 alarms before considering the deployment to be complete
```

```
GrowthFactor                : The percentage of targets that received a deployed
 configuration during each interval
GrowthType                  : The linear or exponential algorithm used to define
 how percentage grew over time
HttpStatusCode              : HTTP Status of the runtime
Id                          : The deployment strategy ID
Name                        : Name of the deployment strategy
ReplicateTo                 : The Systems Manager (SSM) document where the
 deployment strategy is saved
ResponseMetadata            : Runtime Metadata
```

部署組態

建立使用功能旗標和自由格式組態資料所需的成品後，您可以使用 AWS Management Console AWS CLI、或 SDK 建立新部署。在中啟動部署會 AWS AppConfig 呼叫 StartDeploymentAPI 作業。此呼叫包含 AWS AppConfig 應用程式、環境、組態描述檔的 ID，以及要 (選擇性) 部署的組態資料版本。呼叫也包含要使用的部署策略 ID，這會決定組態資料如何部署。

如果您部署存放在中的密碼 AWS Secrets Manager、使用客戶受管金鑰加密的 Amazon 簡單儲存服務 (Amazon S3) 物件，或使用客戶受管金鑰加密存放在 AWS Systems Manager 參數存放區中存放的安全字串參數，則必須為`KmsKeyIdentifier`參數指定值。如果您的組態未加密或使用加密 AWS 受管金鑰，則不需要為`KmsKeyIdentifier`參數指定值。

> **Note**
>
> 您指定的值`KmsKeyIdentifier`必須是客戶管理的金鑰。這不一定與您用來加密配置的密鑰相同。
> 使用開始部署時`KmsKeyIdentifier`，附加至 AWS Identity and Access Management (IAM) 主體的權限政策必須允許該`kms:GenerateDataKey`作業。

AWS AppConfig 監視所有主機的分發並報告狀態。如果發行版失敗，則回 AWS AppConfig 復組態。

> **Note**
>
> 您一次只能將一個組態部署至一個環境。不過，您可以同時將一個組態分別部署到不同的環境。

部署設定 (主控台)

使用下列程序來使用 AWS Systems Manager 主控台部署 AWS AppConfig 組態。

使用主控台部署組態

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在功能窗格中，選擇 [應用程式]，然後選擇您在其中建立的應用程式為您的應用程式建立命名空間 AWS AppConfig。
3. 在 [環境] 索引標籤上，填入環境的選項按鈕，然後選擇 [檢視詳細資料]。
4. 選擇 Start deployment (啟動部署)。
5. 對於 Configuration (組態)，請從清單中選擇一個組態。
6. 根據組態的來源，使用版本清單來選擇您要部署的版本。
7. 對於 Deployment strategy (部署策略)，請從清單中選擇策略。
8. (選擇性) 對於部署說明，請輸入說明。
9. 對於其他加密選項，請從清單中選擇 AWS Key Management Service 金鑰。
10. (選擇性) 在「標籤」區段中，選擇「新增標籤」，然後輸入金鑰和選用值。您可以為資源指定最多 50 個標籤。
11. 選擇 Start deployment (啟動部署)。

部署配置（命令行）

下列程序說明如何使用 AWS CLI (在 Linux 或 Windows 上) 或 AWS Tools for PowerShell 部署組 AWS AppConfig 態。

若要逐步部署組態

1. 開啟 AWS CLI.
2. 執行下列命令以部署組態。

   Linux

```
aws appconfig start-deployment \
  --application-id The_application_ID \
  --environment-id The_environment_ID \
  --deployment-strategy-id The_deployment_strategy_ID \
```

```
  --configuration-profile-id The_configuration_profile_ID \
  --configuration-version The_configuration_version_to_deploy \
  --description A_description_of_the_deployment \
  --tags User_defined_key_value_pair_metadata_of_the_deployment
```

Windows

```
aws appconfig start-deployment ^
  --application-id The_application_ID ^
  --environment-id The_environment_ID ^
  --deployment-strategy-id The_deployment_strategy_ID ^
  --configuration-profile-id The_configuration_profile_ID ^
  --configuration-version The_configuration_version_to_deploy ^
  --description A_description_of_the_deployment ^
  --tags User_defined_key_value_pair_metadata_of_the_deployment
```

PowerShell

```
Start-APPCDeployment `
  -ApplicationId The_application_ID `
  -ConfigurationProfileId The_configuration_profile_ID `
  -ConfigurationVersion The_configuration_version_to_deploy `
  -DeploymentStrategyId The_deployment_strategy_ID `
  -Description A_description_of_the_deployment `
  -EnvironmentId The_environment_ID `
  -Tag Hashtable_type_user_defined_key_value_pair_metadata_of_the_deployment
```

系統會傳回相關資訊，如下所示。

Linux

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
   "DeploymentNumber": The sequence number of the deployment,
   "ConfigurationName": "The name of the configuration",
```

```
  "ConfigurationLocationUri": "Information about the source location of the
 configuration",
  "ConfigurationVersion": "The configuration version that was deployed",
  "Description": "The description of the deployment",
  "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
  "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
  "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
  "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
  "State": "The state of the deployment",

  "EventLog": [
     {
        "Description": "A description of the deployment event",
        "EventType": "The type of deployment event",
        "OccurredAt": The date and time the event occurred,
        "TriggeredBy": "The entity that triggered the deployment event"
     }
  ],

  "PercentageComplete": The percentage of targets for which the deployment is
 available,
  "StartedAt": The time the deployment started,
  "CompletedAt": The time the deployment completed
}
```

Windows

```
{
   "ApplicationId": "The ID of the application that was deployed",
   "EnvironmentId" : "The ID of the environment",
   "DeploymentStrategyId": "The ID of the deployment strategy that was
 deployed",
   "ConfigurationProfileId": "The ID of the configuration profile that was
 deployed",
   "DeploymentNumber": The sequence number of the deployment,
   "ConfigurationName": "The name of the configuration",
   "ConfigurationLocationUri": "Information about the source location of the
 configuration",
   "ConfigurationVersion": "The configuration version that was deployed",
   "Description": "The description of the deployment",
```

```
  "DeploymentDurationInMinutes": Total amount of time the deployment lasted,
  "GrowthType": "The linear or exponential algorithm used to define how
 percentage grew over time",
  "GrowthFactor": The percentage of targets to receive a deployed configuration
 during each interval,
  "FinalBakeTimeInMinutes": Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete,
  "State": "The state of the deployment",

  "EventLog": [
     {
        "Description": "A description of the deployment event",
        "EventType": "The type of deployment event",
        "OccurredAt": The date and time the event occurred,
        "TriggeredBy": "The entity that triggered the deployment event"
     }
  ],

  "PercentageComplete": The percentage of targets for which the deployment is
 available,
  "StartedAt": The time the deployment started,
  "CompletedAt": The time the deployment completed
}
```

PowerShell

```
ApplicationId               : The ID of the application that was deployed
CompletedAt                 : The time the deployment completed
ConfigurationLocationUri    : Information about the source location of the
 configuration
ConfigurationName           : The name of the configuration
ConfigurationProfileId      : The ID of the configuration profile that was
 deployed
ConfigurationVersion        : The configuration version that was deployed
ContentLength               : Runtime of the deployment
DeploymentDurationInMinutes : Total amount of time the deployment lasted
DeploymentNumber            : The sequence number of the deployment
DeploymentStrategyId        : The ID of the deployment strategy that was
 deployed
Description                 : The description of the deployment
EnvironmentId               : The ID of the environment that was deployed
```

```
EventLog                    : {Description : A description of the deployment
 event, EventType : The type of deployment event, OccurredAt : The date and time
 the event occurred,
        TriggeredBy : The entity that triggered the deployment event}
FinalBakeTimeInMinutes      : Time AWS AppConfig monitored for alarms before
 considering the deployment to be complete
GrowthFactor                : The percentage of targets to receive a deployed
 configuration during each interval
GrowthType                  : The linear or exponential algorithm used to define
 how percentage grew over time
HttpStatusCode              : HTTP Status of the runtime
PercentageComplete          : The percentage of targets for which the deployment
 is available
ResponseMetadata            : Runtime Metadata
StartedAt                   : The time the deployment started
State                       : The state of the deployment
```

AWS AppConfig 部署整合 CodePipeline

AWS AppConfig 是 AWS CodePipeline (CodePipeline) 的整合式部署動作。 CodePipeline 是一項全受管的持續交付服務，可協助您將發行管道自動化，進行快速可靠的應用程式和基礎結構更新。CodePipeline 每次發生程式碼變更時，都會根據您定義的發行模型，自動執行發行程序的建置、測試和部署階段。如需詳細資訊，請參閱什麼是 AWS CodePipeline？

AWS AppConfig 與的整合提 CodePipeline 供下列優點：

- 使用 CodePipeline 管理協調流程的客戶現在可以輕量化方式將組態變更部署到應用程式，而不需要部署整個程式碼庫。
- 想要用 AWS AppConfig 來管理組態部署，但因為 AWS AppConfig 不支援目前的程式碼或組態存放區而受到限制的客戶，現在有其他選項。 CodePipeline 支持 AWS CodeCommit GitHub，，和 BitBucket（僅舉幾例）。

> **Note**
>
> AWS AppConfig 只有在可 CodePipeline 用的 AWS 區域 地方 CodePipeline 才支援與整合。

整合的運作方式

您可以通過設置和配置開始 CodePipeline。這包括將您的組態新增至 CodePipeline支援的程式碼存放區。接下來，您可以執行下列工作來設定 AWS AppConfig 環境：

- 建立命名空間和設定描述檔
- 選擇預先定義的部署策略或自行建立

完成這些工作後，您可以在中建立指定 CodePipeline AWS AppConfig 為部署提供者的管道。然後，您可以變更組態，並將其上傳至 CodePipeline 程式碼存放區。上傳新組態會自動在中啟動新部署 CodePipeline。部署完成後，您可以驗證您的變更。如需有關建立指定 AWS AppConfig 為部署提供者的管線的資訊，請參閱《使用指南》中的教學課程：建立用 AWS AppConfig 作部署提供AWS CodePipeline 者的管線。

擷取特徵旗標和模型組態資料 AWS AppConfig

您的應用程式會使用 Data 服務建立組態工作階段，擷取功能旗標和任意格式組態 AWS AppConfig 資料。如果您使用本節所述的其中一種簡化擷取方法，則代理 AWS AppConfig 程式 Lambda 延伸模組或 AWS AppConfig 代理程式會代表您管理一系列 API 呼叫和工作階段權杖。您可以將 AWS AppConfig 代理程式設定為本機主機，並讓代理程式輪詢以 AWS AppConfig 進行組態更新。代理程式會呼叫工作StartConfiguration階段和GetLatest設定 API 動作，並在本機快取您的組態資料。若要擷取資料，您的應用程式會對本機主機伺服器進行 HTTP 呼叫。 AWS AppConfig Agent 支援數個使用案例，如中所述簡化的擷取方法。

如果您願意，您可以手動呼叫這些 API 動作來擷取設定。API 程序的運作方式如下：

您的應用程式會使用 `StartConfigurationSession` API 動作建立組態工作階段。然後，您的會話的客戶端定期調用`GetLatestConfiguration`以檢查和檢索可用的最新數據。

呼叫時`StartConfigurationSession`，您的程式碼會傳送工作階段追蹤之 AWS AppConfig 應用程式、環境和組態設定檔的識別碼 (ID 或名稱)。

作`GetLatestConfiguration`為回應， AWS AppConfig 提供`InitialConfigurationToken`給工作階段的用戶端，並在第一次呼叫該工作階段時使用。

在調用時`GetLatestConfiguration`，您的客戶端代碼發送它具有並接收的最新`ConfigurationToken`值作為響應：

- `NextPollConfigurationToken`：要在下次呼叫時使用的`ConfigurationToken`值`GetLatestConfiguration`。
- 配置：用於會話的最新數據。如果用戶端已經擁有最新版本的組態，則此選項可能是空的。

此區段包含下列資訊：

目錄

關於資 AWS AppConfig 料平面服務

2021 年 11 月 18 日， AWS AppConfig 發布了一項新的數據平面服務。此服務會使用 GetConfiguration API 動作取代先前擷取設定資料的程序。資料平面服務使用兩個新的 API 動作：工作StartConfiguration階段和GetLatest組態。資料平面服務也會使用新的端點。

如果您在 2022 年 1 月 28 日 AWS AppConfig 之前開始使用，服務可能會直接呼叫 GetConfiguration API 動作，或者可能正在使用由 AWS代理程式 Lambda 延伸模組提供的用戶端 (例如 AWS AppConfig 代理程式 Lambda 延伸模組) 來呼叫此 API 動作。如果您直接呼叫 GetConfiguration API 動作，請採取步驟使用StartConfigurationSession和 GetLatestConfiguration API 動作。如果您使用的是 AWS AppConfig 代理程式 Lambda 延伸模組，請參閱本主題稍後標題為此變更如何影響 AWS AppConfig 代理程式 Lambda 延伸模組的一節。

與 API 動作相比，新的資料平面 API 動作提供了以下優點，該動作現已淘汰。GetConfiguration

1. 您不需要管理ClientID參數。使用資料平面服務，ClientID由建立的工作階段權杖在內部管理StartConfigurationSession。
2. 您不再需要包含ClientConfigurationVersion來表示組態資料的快取版本。使用資料平面服務，ClientConfigurationVersion由建立的工作階段權杖在內部管理StartConfigurationSession。
3. 用於資料平面 API 呼叫的全新專用端點可將控制平面和資料平面呼叫分隔開來改善程式碼結構。
4. 新的資料平面服務可改善資料平面作業的 future 擴充性。透過使用管理組態資料擷取的組態工作階段，該 AWS AppConfig 小組可以在 future 建立更強大的增強功能。

從 **GetConfiguration** 遷移到 **GetLatestConfiguration**

若要開始使用新的資料平面服務，您需要更新呼叫 GetConfiguration API 動作的程式碼。使用 StartConfigurationSession API 動作啟動設定工作階段，然後呼叫 GetLatestConfiguration API 動作以擷取設定資料。若要改善效能，建議您在本機快取設定資料。如需詳細資訊，請參閱 透過直接呼叫 API 擷取組態。

這項變更如何影響 AWS AppConfig 代理程式 Lambda 延伸

此變更對 AWS AppConfig 代理程式 Lambda 延伸模組的運作方式沒有直接影響。舊版的 AWS AppConfig 代理程式 Lambda 擴充功能會代表您呼叫 GetConfiguration API 動作。較新版本會呼叫資料平面 API 動作。如果您使用 AWS AppConfig Lambda 擴充功能，建議您將擴充功能更新為最新的 Amazon 資源名稱 (ARN)，並更新新 API 呼叫的許可。如需詳細資訊，請參閱 使用 AWS AppConfig 代理程式 Lambda 延伸模組擷取組態。

簡化的擷取方法

AWS AppConfig 提供了數種擷取組態資料的簡化方法。如果您在 AWS Lambda 函數中使用 AWS AppConfig 功能旗標或任意格式組態資料，則可以使用 AWS AppConfig Agent Lambda 延伸模組來擷取組態。如果您的應用程式在 Amazon EC2 執行個體上執行，則可以使用 AWS AppConfig 代理程式擷取組態。 AWS AppConfig 代理程式也支援在 Amazon Elastic Kubernetes Service (Amazon EKS) 或 Amazon Elastic Container Service (Amazon ECS) 容器映像上執行的應用程式。

完成初始設定之後，這些擷取設定資料的方法比直接呼叫 AWS AppConfig API 更簡單。它們會自動實施最佳實踐，並可能會降低您的使用成本， AWS AppConfig 因為更少的 API 調用來檢索配置。

主題

使用 AWS AppConfig 代理程式 Lambda 延伸模組擷取組態

AWS Lambda 擴充功能是增強 Lambda 函數功能的伴隨程序。擴展可以在調用函數之前啟動，與函數 parallel 運行，並在處理函數調用後繼續運行。實質上，Lambda 擴充功能就像是與 Lambda 叫用 parallel 執行的用戶端。此平行用戶端可在其生命週期中的任何時間點與您的功能連接。

如果您在 Lambda 函數中使用 AWS AppConfig 功能旗標或其他動態組態資料，建議您將 AWS AppConfig 代理程式 Lambda 延伸模組新增為 Lambda 函數的層。這使得調用功能標誌更簡單，擴展本身包括簡化使用， AWS AppConfig 同時降低成本的最佳實踐。由於對 AWS AppConfig 服務的 API 呼叫次數減少，並縮短 Lambda 函數處理時間，可降低成本。如需 Lambda 擴充的詳細資訊，請參閱AWS Lambda 開發人員指南中的 Lambda 擴充功能

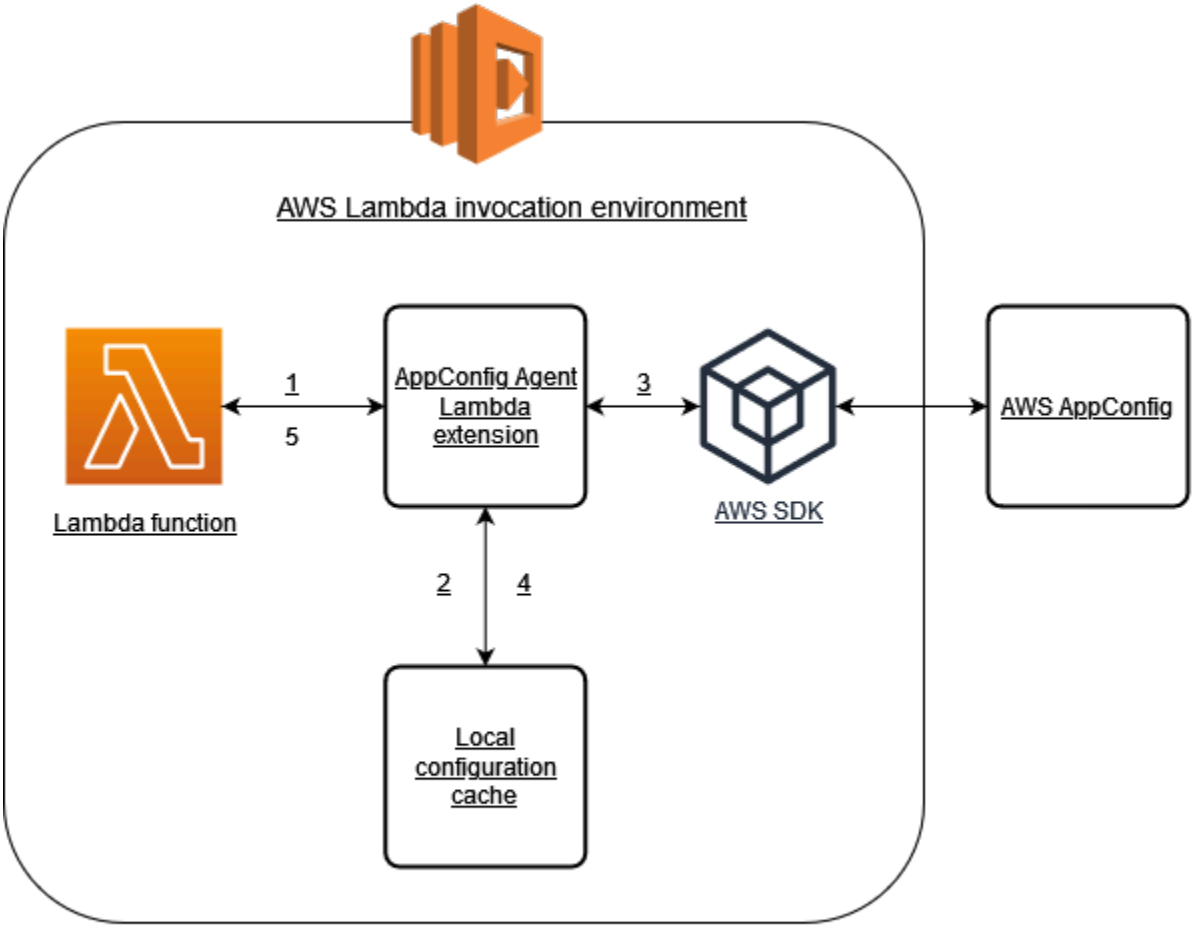
> **Note**
>
> AWS AppConfig 定價是根據呼叫和接收組態的次數而定。如果 Lambda 執行多次冷啟動並經常擷取新的組態資料，您的成本就會增加。

本主題包括 AWS AppConfig 代理程式 Lambda 延伸模組的相關資訊，以及如何設定擴充功能以使用 Lambda 函數的程序。

運作方式

如果您使用 AWS AppConfig 不含 Lambda 擴充功能的 Lambda 函數管理組態，則必須將 Lambda 函數設定為透過與工作StartConfiguration階段和組態 API 動作整合來接收GetLatest組態更新。

將 AWS AppConfig 代理程式 Lambda 延伸模組與 Lambda 函數整合，可簡化此程序 擴充功能負責呼叫 AWS AppConfig 服務、管理擷取資料的本機快取、追蹤下一次服務呼叫所需的組態 Token，以及定期在背景中檢查組態更新。下圖顯示了它是如何工作的。



1. 您可以將 AWS AppConfig 代理程式 Lambda 延伸模組設定為 Lambda 函數的一層。
2. 若要存取其組態資料，您的函數會在上執行的 HTTP 端點呼叫 AWS AppConfig 擴充功能`localhost:2772`。

3. 延伸功能會維護組態資料的本機快取。如果資料不在快取中，擴充功能會呼叫 AWS AppConfig 以取得設定資料。
4. 從服務接收組態後，擴充功能會將其儲存在本機快取中，並將其傳遞至 Lambda 函數。
5. AWS AppConfig 代理程式 Lambda 延伸模組會定期在背景檢查組態資料的更新。每次叫用 Lambda 函數時，擴充功能都會檢查擷取組態後經過的時間。如果經歷時間大於設定的輪詢間隔，則擴充功能會呼叫 AWS AppConfig 以檢查新部署的資料、在發生變更時更新本機快取，並重設經過時間。

> **Note**
>
> - Lambda 會將與函數所要求並行層級相符的另外執行個體具現化。每個執行個體都彼此隔離，並維護自己組態資料的本機快取。如需 Lambda 執行個體和並行的詳細資訊，請參閱管理 Lambda 函數的並行處理。
> - 在您部署更新的組態後，Lambda 函數中顯示組態變更所需的時間 AWS AppConfig，取決於您用於部署的部署策略以及您為擴充功能設定的輪詢間隔。

開始之前

啟用 AWS AppConfig 代理程式 Lambda 延伸模組之前，請執行下列動作：

- 在 Lambda 函數中組織組織組態，以便您可以將它們外部化到 AWS AppConfig.
- 建立 AWS AppConfig 人工因素和組態資料，包括功能旗標或自由格式組態資料。如需詳細資訊，請參閱 在中建立特徵旗標和任意格式組態資料 AWS AppConfig。
- 將其新增`appconfig:GetLatestConfiguration`至 Lambda 函數執行角色所使用的 AWS Identity and Access Management (IAM) 政策，`appconfig:StartConfigurationSession`並加入該政策。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda 執行角色。如需有關 AWS AppConfig 權限的詳細資訊，請參閱服務授權參考 AWS AppConfig中的動作、資源和條件索引鍵。

新增 AWS AppConfig 代理程式 Lambda 延伸

若要使用 AWS AppConfig 代理程式 Lambda 延伸模組，您需要將擴充功能新增至您的 Lambda。這可以透過將 AWS AppConfig 代理程式 Lambda 延伸模組新增至 Lambda 函數作為層，或在 Lambda 函數上啟用擴充功能做為容器映像來完成。

> **Note**
>
> AWS AppConfig 擴充功能不受執行階段限制，並支援所有執行階段。

使用層和 ARN 新增 AWS AppConfig 代理程式 Lambda 延伸模組

若要使用 AWS AppConfig 代理程式 Lambda 延伸模組，您可以將擴充功能新增至 Lambda 函數做為一個層。有關如何向函數添加圖層的詳細信息，請參閱AWS Lambda 開發人員指南中的配置擴展。AWS Lambda 控制台中擴展名的名稱是 AWS AppConfig-擴展名。另請注意，當您將擴充功能新增為 Lambda 的層時，必須指定 Amazon 資源名稱 (ARN)。從下列其中一個與平台對應的清單中選擇 ARN，以及您建立 Lambda 的 AWS 區域 位置。

- 64 平台
- ARM64 平台

如果您想在將擴充功能新增至函式之前先測試擴充功能，您可以使用下列程式碼範例來驗證其運作正常。

```
import urllib.request


def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name'
    config = urllib.request.urlopen(url).read()
    return config
```

要對其進行測試，請為 Python 創建一個新的 Lambda 函數，添加擴展名，然後運行 Lambda 函數。執行 Lambda 函數之後， AWS AppConfig Lambda 函數會傳回您為 http://localhost:2772 路徑指定的組態。如需建立 Lambda 函數的詳細資訊，請參閱AWS Lambda 開發人員指南中的使用主控台建立 Lambda 函數。

若要將 AWS AppConfig 代理程式 Lambda 延伸模組新增為容器映像檔，請參閱使用容器映像檔新增 AWS AppConfig 代理程式 Lambda 延伸模組。

設定 AWS AppConfig 代理程式 Lambda 延伸

您可以透過變更下列 AWS Lambda 環境變數來設定擴充功能。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的使用 AWS Lambda 環境變數。

預取組態資料

環境變數AWS_APPCONFIG_EXTENSION_PREFETCH_LIST可以改善函數的啟動時間。初始化 AWS AppConfig 代理程式 Lambda 延伸模組時，它會 AWS AppConfig 在 Lambda 開始初始化函數並叫用您的處理常式之前，擷取指定的組態。在某些情況下，在函數請求之前，配置數據已經在本地緩存中可用。

若要使用預先擷取功能，請將環境變數的值設定為對應於組態資料的路徑。例如，如果您的配置對應於分別名為「我的應用程序」，「我的環境」和「my_configation_data」的應用程序，環境和配置配置文件，則路徑將是。/applications/my_application/environments/my_environment/configurations/my_configuration_data您可以將多個組態項目列為逗號分隔清單來指定多個組態項目 (如果您的資源名稱包含逗號，請使用資源的 ID 值而非名稱)。

從另一個帳戶存取組態資料

AWS AppConfig 代理程式 Lambda 延伸模組可透過指定授與資料許可的 IAM 角色，從另一個帳戶擷取組態資料。若要進行設定，請依照下列步驟執行：

1. 在用 AWS AppConfig 於管理組態資料的帳戶中，建立具有信任政策的角色，該角色會授與執行 Lambda 函數的帳戶存取appconfig:StartConfigurationSession和appconfig:GetLatestConfiguration動作，以及與 AWS AppConfig 組態資源對應的部分或完整 ARN。
2. 在執行 Lambda 函數的帳戶中，使用在步驟 1 中建立之角色的 ARN，將AWS_APPCONFIG_EXTENSION_ROLE_ARN環境變數新增至 Lambda 函數。
3. (選擇性) 如有需要，可以使用AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID環境變數指定外部 ID。同樣地，可以使用AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME環境變數來配置工作階段名稱。

> **Note**
>
> 記下以下資訊。
>
> - AWS AppConfig 代理程式 Lambda 擴充功能只能從一個帳戶擷取資料。如果您指定 IAM 角色，則擴充功能將無法從執行 Lambda 函數的帳戶擷取組態資料。
> - AWS Lambda 使用 Amazon CloudWatch 日誌記錄 AWS AppConfig 代理程式 Lambda 延伸模組和 Lambda 函數的相關資訊。

環境變數	詳細資訊	預設值
AWS_APPCONFIG_EXTENSION_HTTP_PORT	此環境變數指定主控擴充功能的本機 HTTP 伺服器執行的連接埠。	2772
AWS_APPCONFIG_EXTENSION_LOG_LEVEL	此環境變數指定要針對 AWS AppConfig 某個函數傳送至 Amazon Logs 的延伸特定 CloudWatch 日誌。有效且不區分大小寫的值為：debug、info、warnerro:和。none調試包括有關擴展程序的詳細信息，包括計時信息。	info
AWS_APPCONFIG_EXTENSION_MAX_CONNECTIONS	此環境變數會設定延伸模組用來擷取組態的連線數目上限。AWS AppConfig	3
AWS_APPCONFIG_EXTENSION_POLL_INTERVAL_SECONDS	此環境變數可控制擴充功能輪詢更新組態的頻率 (以秒 AWS AppConfig 為單位)。	45
AWS_APPCONFIG_EXTENSION_POLL_TIMEOUT_MILLIS	此環境變數控制重新整理快取中的資料時，擴充功能會等待回應的最大 AWS AppConfig 時間量 (以毫秒為單位)。如果在指定的時間量內 AWS AppConfig 沒有回應，擴充功能會略過此輪詢間隔，並傳回先前更新的快取資料。	3000
AWS_APPCONFIG_EXTENSION_PREFETCH_LIST	此環境變數會指定擴充功能在函數初始化並執行處理常式之前開始擷取的組態資料。它	無

環境變數	詳細資訊	預設值
	可以顯著減少功能的冷啟動時間。	
AWS_APPCONFIG_EXTENSION_PROXY_HEADERS	此環境變數會指定環 AWS_APPCONFIG_EXTENSION_PROXY_URL 境變數中參照之 Proxy 所需的標頭。值是以逗號分隔的標頭清單。每個標題使用以下形式： `"header: value"`	無
AWS_APPCONFIG_EXTENSION_PROXY_URL	此環境變數指定要用於從 AWS AppConfig 擴充功能到的連線的 Proxy URL AWS 服務。HTTPS和HTTP網址被支持。	無
AWS_APPCONFIG_EXTENSION_ROLE_ARN	此環境變數會指定 IAM 角色 ARN 對應至 AWS AppConfig 擴充功能假設以擷取組態的角色。	無
AWS_APPCONFIG_EXTENSION_ROLE_EXTERNAL_ID	此環境變數會指定要與假定角色 ARN 搭配使用的外部 ID。	無
AWS_APPCONFIG_EXTENSION_ROLE_SESSION_NAME	此環境變數會指定要與假設 IAM 角色的登入資料相關聯的工作階段名稱。	無
AWS_APPCONFIG_EXTENSION_SERVICE_REGION	此環境變數會指定擴充功能用來呼叫 AWS AppConfig 服務的替代區域。未定義時，擴充功能會使用目前「區域」中的端點。	無

環境變數	詳細資訊	預設值
AWS_APPCONFIG_EXTENSION_MANIFEST	此環境變數會設定 AWS AppConfig 代理程式，以利用額外的每個組態功能，例如多帳戶擷取，以及將組態儲存至磁碟。您可以輸入下列其中一個值： • "app:env:manifest-config" • "file:/fully/qualified/path/to/manifest.json" 如需這些功能的詳細資訊，請參閱其他擷取功能。	true
AWS_APPCONFIG_EXTENSION_WAIT_ON_MANIFEST	此環境變數會設定 AWS AppConfig 代理程式等到資訊清單處理完成後才完成啟動。	true

從功能旗標組態擷取一或多個旗標

對於功能旗標組態 (類型的組態AWS.AppConfig.FeatureFlags)，Lambda 擴充功能可讓您擷取組態中的單一旗標或旗標子集。如果 Lambda 只需要使用組態設定檔中的幾個旗標，擷取一或兩個旗標就很有用。下面的實例使用 Python。

> **Note**
>
> 只有在 AWS AppConfig 代理程式 Lambda 延伸版本 2.0.45 及更新版本中，才能呼叫組態中的單一功能旗標或旗標子集。

您可以從本機 HTTP 端點擷取 AWS AppConfig 組態資料。若要存取特定旗標或旗標清單，請使用 AWS AppConfig 組態設定檔的 ?flag=*flag_name* query 參數。

若要存取單一旗標及其屬性

```
import urllib.request

def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name'
    config = urllib.request.urlopen(url).read()
    return config
```

若要存取多個旗標及其屬性

```
import urllib.request

def lambda_handler(event, context):
    url = f'http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two'
    config = urllib.request.urlopen(url).read()
    return config
```

檢視 AWS AppConfig 代理程式 Lambda 延伸

您可以在記錄中檢視 AWS AppConfig 代理程式 Lambda 延伸模組的 AWS Lambda 記錄資料。記錄項目前面加上。`appconfig agent`範例如下。

```
[appconfig agent] 2024/05/07 04:19:01 ERROR retrieve failure for
 'SourceEventConfig:SourceEventConfigEnvironment:SourceEventConfigProfile':
 StartConfigurationSession: api error AccessDenied: User:
 arn:aws:sts::0123456789:assumed-role/us-east-1-LambdaRole/extension1 is not authorized
 to perform: sts:AssumeRole on resource: arn:aws:iam::0123456789:role/test1 (retry in
 60s)
```

AWS AppConfig 代理程式 Lambda 延伸模組的可用版

本主題包含 AWS AppConfig 代理程式 Lambda 延伸模組版本的相關 AWS AppConfig 代理程式 Lambda 延伸模組支援專為 x86-64 和 ARM64 (重力數 2) 平台開發的 Lambda 函數。若要正常運作，您的 Lambda 函數必須設定為使用目前託管 AWS 區域 位置的特定 Amazon 資源名稱 (ARN)。您可以在本節稍後檢視 AWS 區域 和 ARN 詳細資訊。

> **Important**
>
> 請注意下列 AWS AppConfig 代理程式 Lambda 延伸模組的重要詳細資料
>
> - `GetConfiguration`API 動作已於 2022 年 1 月 28 日棄用。接收設定資料的呼叫應改用`StartConfigurationSession`和 `GetLatestConfiguration` API。如果您使用的是在 2022 年 1 月 28 日之前建立的 AWS AppConfig 代理程式 Lambda 擴充功能版本，您可能必須設定新 API 的權限。如需詳細資訊，請參閱 關於資 AWS AppConfig 料平面服務。
> - AWS AppConfig 支援中列出的所有版本較舊的延伸版本。我們建議您定期更新至最新版本，以利用延伸功能增強功能。

主題

AWS AppConfig 代理程式 Lambda 擴充功能

下表說明對 AWS AppConfig Lambda 擴充功能最新版本所做的變更。

版本	啟動日期	備註
2.0.358	12/01/2023	已新增對下列擷取功能的支援： • 多帳戶擷取：使用主帳戶或擷取 AWS 帳戶 中的 AWS AppConfig 代理程式，從多個廠商帳戶擷取組態資料。 • 將組態複本寫入磁碟：使用「 AWS AppConfig 代理程式」將組態資料寫入磁碟。此功能可讓客戶擁有從磁碟

版本	啟動日期	備註
		讀取組態資料的應用程式以進行整合 AWS AppConfig。
2.0.181	08/14/2023	增加了對以色列（特拉維夫）il- AWS 區域 central-1 的支持。
2.0.165	02/21/2023	次要錯誤修正。不再透過 AWS Lambda 主控台將擴充功能使用限制為特定執行階段版本。增加了對以下內容的支持 AWS 區域： • 中東（阿聯酋）me-central-1 • 亞太區域 (海德拉巴)，ap-south-2 • 亞太區域 (墨爾本), ap-southeast-4 • 歐洲（西班牙），eu-south-2 • 歐洲（蘇黎世），eu-central-2
2.0.122	08/23/2022	增加了對通道代理的支持，該代理可以使用`AWS_APPCONFIG_EXTENSION_PROXY_URL` 和`AWS_APPCONFIG_EXTENSION_PROXY_HEADERS` 環境變量進行配置。已新增 .NET 6 作為執行階段。如需環境變數的詳細資訊，請參閱設定 AWS AppConfig 代理程式 Lambda 延伸。

版本	啟動日期	備註
2.0.58	2022 年 3 月 5 日	改進了對 Lambda 中重力 2（ARM64）處理器的支持。
2.0.45	03/15/2022	增加了對調用單個功能標誌的支持。先前，客戶呼叫功能旗標分組到組態設定檔中，並且必須剖析回應用戶端。在此版本中，客戶可以在呼叫 HTTP localhost 端點時使用`flag=<flag-name>` 參數來取得單一旗標的值。還增加了對重力 2（ARM64）處理器的初始支持。

尋找您的擴 Lambda 功能版本號碼

使用下列程序來尋找目前設定的 AWS AppConfig 代理程式 Lambda 延伸模組的版本號碼。若要正常運作，您的 Lambda 函數必須設定為使用目前託管 AWS 區域 位置的特定 Amazon 資源名稱 (ARN)。

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 選擇您要在其中新增`AWS-AppConfig-Extension`圖層的 Lambda 函數。
3. 在 圖層 部份，選擇 新增圖層。
4. 在 「選擇AWS 圖層」區段中，從「圖層」清單中選擇「AWS AppConfig-延伸」。
5. 使用「版本」清單來選擇版本號碼。
6. 選擇新增。
7. 使用「測試」頁籤來測試函數。
8. 測試完成後，檢視記錄輸出。在 [執行的詳細資料] 區段中找出 AWS AppConfig 代理程式 Lambda 延伸模組版本。此版本必須符合該版本所需的 URL。

64 平台

當您將擴充功能新增為 Lambda 的圖層時，您必須指定 ARN。從下表中選擇與您建立 Lambda 的 AWS 區域 位置相對應的 ARN。這些 ARN 適用於為 x86-64 平台開發的 Lambda 函數。

版本二點三五

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:93
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:141
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:161
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:93
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:106
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:47
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:125
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:93

區域	ARN
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:98
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:159
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:83
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:44
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:76
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:76
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:83
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:98
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:108

區域	ARN
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:101
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:106
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:106
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:79
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:20
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:107
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:47
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:128
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:83

區域	ARN
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:22
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:49
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:85
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:54
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:54

ARM64 平台

當您將擴充功能新增為 Lambda 的圖層時，您必須指定 ARN。從下表中選擇與您建立 Lambda 的 AWS 區域 位置相對應的 ARN。這些 ARN 適用於針對 ARM64 平台開發的 Lambda 函數。

版本二點三五

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:61
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:45

區域	ARN
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:18
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:63
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:13
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:49
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension-Arm64:5
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:63
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:45
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:17
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:18

區域	ARN
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:11
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension-Arm64:5
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:11
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:51
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:16
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:16
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:58
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:49
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:16

區域	ARN
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension-Arm64:5
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:49
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension-Arm64:5
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:16
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:11
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension-Arm64:5
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:13
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension-Arm64:5

較舊的延伸版本

本節列出 ARN 以及 AWS 區域 較舊版本的 AWS AppConfig Lambda 擴充功能。此清單不包含所有舊版 AWS AppConfig Agent Lambda 延伸模組的資訊，但會在發行新版本時更新。

較舊的擴充功能版本 (x86-64 平台)

下表列出 ARN 以及 AWS 區域 針對 x86-64 平台開發之 AWS AppConfig 代理程式 Lambda 延伸模組的舊版本。

由較新的副檔名取代的日期：2023 年 12 月 1 日

版本二零一八

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:113
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:81
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:124
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:146
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:81
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:93
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:32

區域	ARN
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:110
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:81
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:82
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:142
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:73
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:29
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:68
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:68
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:73

區域	ARN
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:84
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:93
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:86
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:91
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:93
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:64
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:5
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:94
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:32

區域	ARN
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:113
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:73
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:895787185223:layer:AWS-AppConfig-Extension:7
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:34
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:73
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:46
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:46

由較新的副檔名取代的日期：2023 年 8 月 14 日

版本

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:110
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:79
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:121
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:143
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:79
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:91
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:758369105281:layer:AWS-AppConfig-Extension:29
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:108
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:79

區域	ARN
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:80
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:139
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:71
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:586093569114:layer:AWS-AppConfig-Extension:26
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:66
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:66
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:71
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:82
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:91

區域	ARN
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:84
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:89
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:91
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:60
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:307021474294:layer:AWS-AppConfig-Extension:2
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:92
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:489524808438:layer:AWS-AppConfig-Extension:29
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:110
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:71

區域	ARN
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:662846165436:layer:AWS-AppConfig-Extension:31
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:71
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:44
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:44

由較新的副檔名取代的日期：2023 年 2 月 2 日

版本

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:82
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:59
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:93

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:114
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:59
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:70
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:82
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:59
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:60
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:111
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:54
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:52

區域	ARN
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:52
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:54
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:62
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:70
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:59
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:64
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:70
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:37
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:71

區域	ARN
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:82
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:54
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:54
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:29
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:29

由較新的副檔名取代的日期：2022 年 8 月 23 日

版本二點五八

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:69
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:50

區域	ARN
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:78
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:101
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:50
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:59
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:69
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:50
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:51
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:98
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:47

區域	ARN
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:46
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:46
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:47
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:49
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:59
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:46
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:51
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:59
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:24

區域	ARN
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:60
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:69
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:47
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:47
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:23
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:23

由較新的副檔名取代的日期：2022 年 4 月 21 日

版本二點四五

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:68

區域	ARN
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:49
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:77
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:100
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:49
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:58
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:68
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:49
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:50
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:97

區域	ARN
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:46
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:45
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:45
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:46
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:48
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:58
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:45
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:50
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:58

區域	ARN
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:23
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:59
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:68
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:46
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:46
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:22
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:22

由較新的副檔名取代的日期：2022 年 3 月 15 日

版本二點三十

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:61
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension:47
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension:61
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension:89
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension:47
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension:54
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension:59
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension:47
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension:48

區域	ARN
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension:86
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension:44
中國 (北京)	arn:aws-cn:lambda:cn-north-1:615057806174:layer:AWS-AppConfig-Extension:43
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:615084187847:layer:AWS-AppConfig-Extension:43
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension:44
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension:45
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension:42
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension:54
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension:45

區域	ARN
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension:54
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension:13
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension:55
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension:61
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension:44
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension:44
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946561847325:layer:AWS-AppConfig-Extension:20
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946746059096:layer:AWS-AppConfig-Extension:20

較舊的擴充功能版本 (ARM64 平台)

下表列出 ARN 以及 AWS 區域 針對 ARM64 平台開發之 AWS AppConfig 代理程式 Lambda 延伸模組的舊版本。

由較新的副檔名取代的日期：2023 年 12 月 1 日

版本二零一八

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:46
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:33
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:958113053741:layer:AWS-AppConfig-Extension-Arm64:1
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:48
加拿大 (中部)	arn:aws:lambda:ca-central-1:039592058896:layer:AWS-AppConfig-Extension-Arm64:1
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:36
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:48
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:33

區域	ARN
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:493207061005:layer:AWS-AppConfig-Extension-Arm64:1
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:646970417810:layer:AWS-AppConfig-Extension-Arm64:1
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:203683718741:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (香港)	arn:aws:lambda:ap-east-1:630222743974:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:37
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:826293736237:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:706869817123:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:43
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:36

區域	ARN
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:418787028745:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:36
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:000010852771:layer:AWS-AppConfig-Extension-Arm64:1
非洲 (開普敦)	arn:aws:lambda:af-south-1:574348263942:layer:AWS-AppConfig-Extension-Arm64:1
Middle East (Bahrain)	arn:aws:lambda:me-south-1:559955524753:layer:AWS-AppConfig-Extension-Arm64:1

由較新的副檔名取代的日期：2023 年 3 月 30 日

版本

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:43
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:31

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:45
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:34
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:46
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:31
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:35
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:41
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:34
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:34

由較新的副檔名取代的日期：2023 年 2 月 2 日

版本

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:15
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:11
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:16
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:13
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:20
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:11
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:15
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:16
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:13

區域	ARN
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:13

由較新的副檔名取代的日期：2022 年 8 月 23 日

版本二點五八

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:2
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:2
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:3
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:2
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:7
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:2

區域	ARN
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:2
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:3
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:2
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:2

由較新的副檔名取代的日期：2022 年 4 月 21 日

版本二點四五

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension-Arm64:1
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:728743619870:layer:AWS-AppConfig-Extension-Arm64:1
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:359756378197:layer:AWS-AppConfig-Extension-Arm64:2

區域	ARN
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:066940009817:layer:AWS-AppConfig-Extension-Arm64:1
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:434848589818:layer:AWS-AppConfig-Extension-Arm64:6
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:282860088358:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:980059726660:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:421114256042:layer:AWS-AppConfig-Extension-Arm64:2
亞太區域 (悉尼)	arn:aws:lambda:ap-southeast-2:080788657173:layer:AWS-AppConfig-Extension-Arm64:1
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:554480029851:layer:AWS-AppConfig-Extension-Arm64:1

使用容器映像檔新增 AWS AppConfig 代理程式 Lambda 延伸模組

您可以將 AWS AppConfig 代理程式 Lambda 延伸模組封裝為容器映像，以便將其上傳到託管在 Amazon Elastic Container Registry (Amazon ECR) 上的容器登錄。

將 AWS AppConfig 代理程式 Lambda 延伸模組新增為 Lambda 容器映像

1. 在（ ）中輸入 AWS 區域 和 Amazon 資源名稱（ARN AWS CLI），如下所示。 AWS Command Line Interface 將區域和 ARN 值取代為您的區域和相符的 ARN，以擷取 Lambda 層的副本。 AWS AppConfig 提供適用於 64 和 ARM64 平台的 ARN。

```
aws lambda get-layer-version-by-arn \
  --region AWS ## \
  --arn extension ARN
```

範例如下。

```
aws lambda get-layer-version-by-arn \
  --region us-east-1 \
  --arn arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-Extension:128
```

回傳的結果如下所示：

```
{
  "LayerVersionArn": "arn:aws:lambda:us-east-1:027255383542:layer:AWS-AppConfig-
Extension:128",
  "Description": "AWS AppConfig extension: Use dynamic configurations deployed via
 AWS AppConfig for your AWS Lambda functions",
  "CreatedDate": "2021-04-01T02:37:55.339+0000",
  "LayerArn": "arn:aws:lambda::layer:AWS-AppConfig-Extension",

  "Content": {
    "CodeSize": 5228073,
    "CodeSha256": "8otOgbLQbexpUm3rKlMhvcE6Q5TvwcLCKrc4Oe+vmMY=",
    "Location" : "S3-Bucket-Location-URL"
  },

  "Version": 30,
  "CompatibleRuntimes": [
    "python3.8",
    "python3.7",
    "nodejs12.x",
    "ruby2.7"
  ],
}
```

2. 在上述回應中，傳回的值`Location`是包含 Lambda 擴充功能之亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體的 URL。將網址貼到您的網頁瀏覽器中，以下載 Lambda 擴充功能 .zip 檔案。

> **Note**
>
> Amazon S3 儲存貯體 URL 只能使用 10 分鐘。

(選擇性) 或者，您也可以使用下列`curl`命令來下載 Lambda 擴充功能。

```
curl -o extension.zip "S3-Bucket-Location-URL"
```

(選擇性) 或者，您也可以結合步驟 1 和步驟 2 來擷取 ARN 並一次下載 .zip 副檔名檔案。

```
aws lambda get-layer-version-by-arn \
  --arn extension ARN \
  | jq -r '.Content.Location' \
  | xargs curl -o extension.zip
```

3. 在中新增下列行，以`Dockerfile`將副檔名新增至容器映像檔。

```
COPY extension.zip extension.zip
RUN yum install -y unzip \
  && unzip extension.zip /opt \
  && rm -f extension.zip
```

4. 確定 Lambda 函數執行角色具有應用程式設定：GetConfiguration權限集。

範例

本節包含在容器映像型 Python Lambda 函數上啟用 AWS AppConfig 代理程式 Lambda 延伸模組的範例。

1. 創建一`Dockerfile`個類似於以下內容。

```
FROM public.ecr.aws/lambda/python:3.8 AS builder
COPY extension.zip extension.zip
RUN yum install -y unzip \
  && unzip extension.zip -d /opt \
  && rm -f extension.zip
```

```
FROM public.ecr.aws/lambda/python:3.8
COPY --from=builder /opt /opt
COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. 將延伸層下載至與Dockerfile.

```
aws lambda get-layer-version-by-arn \
  --arn extension ARN \
  | jq -r '.Content.Location' \
  | xargs curl -o extension.zip
```

3. 在相同的目錄index.py中建立一個名為的 Python 檔案Dockerfile。

```
import urllib.request

def handler(event, context):
    return {
       # replace parameters here with your application, environment, and
 configuration names
       'configuration': get_configuration('myApp', 'myEnv', 'myConfig'),
    }

def get_configuration(app, env, config):
    url = f'http://localhost:2772/applications/{app}/environments/{env}/
configurations/{config}'
    return urllib.request.urlopen(url).read()
```

4. 執行下列步驟以建立映docker像並將其上傳到 Amazon ECR。

```
// set environment variables
export ACCOUNT_ID = <YOUR_ACCOUNT_ID>
export REGION = <AWS_REGION>

// create an ECR repository
aws ecr create-repository --repository-name test-repository

// build the docker image
docker build -t test-image .

// sign in to AWS
aws ecr get-login-password \
```

```
  | docker login \
  --username AWS \
  --password-stdin "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com"

// tag the image
docker tag test-image:latest "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/test-
repository:latest"

// push the image to ECR
docker push "$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/test-repository:latest"
```

5. 使用您在上面建立的 Amazon ECR 映像檔建立 Lambda 函數。如需 Lambda 函數做為容器的詳細資訊，請參閱建立定義為容器映像的 Lambda 函數。
6. 確定 Lambda 函數執行角色具有應用程式設定：GetConfiguration權限集。

與應用 OpenAPI 整合

您可以使用下面的 YAML 規範的 OpenAPI 創建使用工具，如 Open API 生成器的 SDK。您可以更新此規格，以包含「應用程式」、「環境」或「組態」的硬式編碼值。您還可以添加其他路徑（如果您有多個配置類型）並包含配置模式，以為 SDK 客戶端生成特定於配置的類型模型。如需 OpenAPI (也稱為施瓦格) 的詳細資訊，請參閱 OpenAPI 規格。

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: AppConfig Agent Lambda extension API
  description: An API model for the AppConfig Agent Lambda extension.
servers:
  - url: https://localhost:{port}/
    variables:
      port:
        default:
          '2772'
paths:
  /applications/{Application}/environments/{Environment}/configurations/
{Configuration}:
    get:
      operationId: getConfiguration
      tags:
        - configuration
      parameters:
```

```
        - in: path
          name: Application
          description: The application for the configuration to get. Specify either the
application name or the application ID.
          required: true
          schema:
            type: string
        - in: path
          name: Environment
          description: The environment for the configuration to get. Specify either the
environment name or the environment ID.
          required: true
          schema:
            type: string
        - in: path
          name: Configuration
          description: The configuration to get. Specify either the configuration name
or the configuration ID.
          required: true
          schema:
            type: string
      responses:
        200:
          headers:
            ConfigurationVersion:
              schema:
                type: string
          content:
            application/octet-stream:
              schema:
                type: string
                format: binary
          description: successful config retrieval
        400:
          description: BadRequestException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        404:
          description: ResourceNotFoundException
          content:
            application/text:
              schema:
```

```
                $ref: '#/components/schemas/Error'
        500:
          description: InternalServerException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        502:
          description: BadGatewayException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'
        504:
          description: GatewayTimeoutException
          content:
            application/text:
              schema:
                $ref: '#/components/schemas/Error'

components:
  schemas:
    Error:
      type: string
      description: The response error
```

從 Amazon EC2 執行個體擷取組態資料

您可以使用 AWS AppConfig 代理程式 AWS AppConfig 與在 Amazon 彈性運算雲端 (Amazon EC2) Linux 執行個體上執行的應用程式整合。代理程式可透過下列方式加強應用程式處理與管理：

- 代理程式 AWS AppConfig 會使用 AWS Identity and Access Management (IAM) 角色並管理設定資料的本機快取，代表您呼叫。藉由從本機快取提取組態資料，您的應用程式需要較少的程式碼更新來管理組態資料、擷取設定資料 (以毫秒為單位)，而且不會受到可能中斷此類資料呼叫的網路問題影響。 *
- 代理程式提供擷取和解析 AWS AppConfig 功能旗標的原生體驗。
- 代理程式現成可提供快取策略、輪詢間隔以及本機組態資料可用性的最佳作法，同時追蹤後續服務呼叫所需的組態 Token。
- 在背景執行時，代理程式會定期輪詢 AWS AppConfig 資料平面以進行組態資料更新。您的應用程序可以通過連接到端口 2772 上的本地主機（可定制的默認端口值）並調用 HTTP GET 來檢索數據來檢索數據。

*AWS AppConfig 代理程式會在服務第一次擷取您的組態資料時快取資料。因此，第一次檢索數據的調用比後續調用慢。

主題

步驟 1：(必要) 建立資源和設定權限

若要 AWS AppConfig 與 Amazon EC2 執行個體上執行的應用程式整合，您必須建立成 AWS AppConfig 品和組態資料，包括功能旗標或自由格式組態資料。如需詳細資訊，請參閱 在中建立特徵旗標和任意格式組態資料 AWS AppConfig。

若要擷取由託管的組態資料 AWS AppConfig，您的應用程式必須設定為具有 AWS AppConfig 資料平面的存取權。若要為應用程式提供存取權限，請更新指派給 Amazon EC2 執行個體角色的 IAM 許可政策。具體而言，您必須將`appconfig:StartConfigurationSession`和`appconfig:GetLatestConfiguration`動作新增至策略。請見此處範例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appconfig:StartConfigurationSession",
                "appconfig:GetLatestConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

如需將許可新增至政策的詳細資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

步驟 2：(必要) 在 Amazon EC2 執行個體上安裝和啟動 AWS AppConfig 代理程式

AWS AppConfig 代理程式託管在由管理的 Amazon Simple Storage Service (Amazon S3) 儲存貯體中 AWS。使用下列程序在 Linux 執行個體上安裝最新版本的代理程式。如果您的應用程式分散在多個執行個體，則您必須在裝載應用程式的每個執行個體上執行此程序。

> **Note**
>
> 記下以下資訊：
>
> - AWS AppConfig 代理程式適用於執行核心版本 4.15 或更新版本的 Linux 作業系統。不支援以 Debian 為基礎的系統，例如 Ubuntu。
> - 此代理程式支援 x86_64 和 ARM64 架構。
> - 對於分散式應用程式，我們建議將安裝和啟動命令新增至 Auto Scaling 群組的 Amazon EC2 使用者資料。如果這樣做，每個執行個體都會自動執行這些指令。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的啟動時在 Linux 執行個體上執行命令。此外，請參閱 Amazon EC2 Auto Scaling 使用者指南中的教學：設定使用者資料以透過執行個體中繼資料擷取目標生命週期狀態。
> - 本主題中的程序說明如何透過登入執行個體來執行命令來執行安裝代理程式之類的動作。您可以從本機用戶端電腦執行命令，並使用 Run Command (功能) 來鎖定一或多個執行個體 AWS Systems Manager。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的 AWS Systems Manager 執行命令。
> - AWS AppConfig Amazon EC2 Linux 實例上的代理是一種`systemd`服務。

在執行個體上安裝和啟動 AWS AppConfig 代理程式

1. 登入您的 Linux 執行個體。
2. 開啟終端機，並以 x86_64 架構的系統管理員權限執行下列命令：

```
sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/x86_64/latest/aws-appconfig-agent.rpm
```

對於 ARM64 架構，請執行下列命令：

```
sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/arm64/latest/aws-appconfig-agent.rpm
```

如果您要安裝特定版本的 AWS AppConfig 代理程式，請在 URL `latest` 中以特定的版本號碼取代。以下是 x86_64 的一個例子：

```
sudo yum install https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/linux/x86_64/2.0.2/aws-appconfig-agent.rpm
```

3. 執行下列命令以啟動代理程式：

```
sudo systemctl start aws-appconfig-agent
```

4. 執行下列命令以確認代理程式是否正在執行：

```
sudo systemctl status aws-appconfig-agent
```

如果成功，命令會傳回如下資訊：

```
aws-appconfig-agent.service - aws-appconfig-agent
  ...
  Active: active (running) since Mon 2023-07-26 00:00:00 UTC; 0s ago
  ...
```

> **Note**
>
> 若要停用代理程式，請執行下列命令：
>
> ```
> sudo systemctl stop aws-appconfig-agent
> ```

步驟 3：(選擇性，但建議使用) 將記錄檔傳送至 CloudWatch 記錄

根據預設， AWS AppConfig 代理程式會將記錄檔發佈至 STDERR。Systemd 會將 Linux 執行個體上執行的所有服務的標準輸出和標準錯誤重新導向至系統日誌。如果您只在一或兩個執行個體上執行 AWS AppConfig 代理程式，則可以在 systemd 日誌中檢視和管理記錄資料。更好的解決方案是我們強烈推薦用於分散式應用程式的解決方案，是將日誌檔寫入磁碟，然後使用 Amazon CloudWatch Agent 將日誌資料上傳到 AWS 雲端。此外，您可以設定 CloudWatch 代理程式刪除執行個體中的舊記錄檔，以防止執行個體磁碟空間不足。

若要啟用記錄到磁碟，您必須設定LOG_PATH環境變數，如中所述步驟 4：(選用) 使用環境變數設定 Amazon EC2 的 AWS AppConfig 代理程式。

若要開始使用 CloudWatch 代理程式，請參閱 Amazon 使用者指南中的代理 CloudWatch 程式從 Amazon EC2 執行個體和現場部署伺服器收集 CloudWatch 指標和日誌。您可以使用「快速設定」，這是一種「Systems Manager」的功能來快速安裝 CloudWatch 代理程式 如需詳細資訊，請參閱AWS Systems Manager 使用指南中的快速設定主機管理。

> **Warning**
>
> 如果您選擇在不使用 CloudWatch 代理程式的情況下將記錄檔寫入磁碟，則必須刪除舊的記錄檔。 AWS AppConfig 代理程式會每小時自動輪換記錄檔。如果您未刪除舊的記錄檔，執行個體可能會耗盡磁碟空間。

在執行個體上安裝 CloudWatch 代理程式之後，請建立 CloudWatch 代理程式設定檔。組態檔會指示 CloudWatch 代理程式如何使用 AWS AppConfig 代理程式記錄檔。如需有關建立 CloudWatch 代理程式組態檔的詳細資訊，請參閱建立 CloudWatch 代理程式組態檔。

將下列logs區段新增至執行個體上的 CloudWatch 代理程式設定檔，並儲存變更：

```
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/path_you_specified_for_logging",
          "log_group_name": "${YOUR_LOG_GROUP_NAME}/aws-appconfig-agent.log",
          "auto_removal": true
        },
        ...
      ]
    },
    ...
  },
  ...
}
```

如果的值auto_removal為true， CloudWatch 代理程式會自動刪除輪換的 AWS AppConfig 代理程式記錄檔。

步驟 4：(選用) 使用環境變數設定 Amazon EC2 的 AWS AppConfig 代理程式

您可以使用環境變數為 Amazon EC2 設定 AWS AppConfig 代理程式。若要設定systemd服務的環境變數，您可以建立嵌入式單位檔案。下列範例顯示如何建立插入式單元檔案，以將 AWS AppConfig 代理程式記錄層級設定為DEBUG。

如何為環境變數建立下拉式單位檔案的範例

1. 登入您的 Linux 執行個體。
2. 開啟終端機並以系統管理員權限執行下列命令。該命令創建一個配置目錄：

```
sudo mkdir /etc/systemd/system/aws-appconfig-agent.service.d
```

3. 執行下列指令以建立置入式單元檔案。以*#####*取代檔案名稱。副檔名必須是.conf：

```
sudo touch /etc/systemd/system/aws-appconfig-agent.service.d/file_name.conf
```

4. 在下拉式單位檔案中輸入資訊。下列範例會新增定義環境變數的Service區段。此範例會將 AWS AppConfig 代理程式記錄層級設定為DEBUG。

```
[Service]
Environment=LOG_LEVEL=DEBUG
```

5. 執行下列命令以重新載入 systemd 組態：

```
sudo systemctl daemon-reload
```

6. 執行下列命令以重新啟動 AWS AppConfig 代理程式：

```
sudo systemctl restart aws-appconfig-agent
```

您可以在插入式單元檔案中指定下列環境變數，以設定 Amazon EC2 的 AWS AppConfig 代理程式。

環境變數	詳細資訊	預設值
ACCESS_TOKEN	此環境變數定義了從代理程式 HTTP 伺服器要求組態資料時必須提供的權杖。權杖的值必須在具有授權類型的 HTTP 要	無

環境變數	詳細資訊	預設值
	求授權標頭中設定Bearer。請見此處範例。 `GET /applications/my_app/... Host: localhost:2772 Authorization: Bearer <token value>`	
BACKUP_DIRECTORY	此環境變數可讓 AWS AppConfig 代理程式將擷取之每個組態的備份儲存至指定目錄。 **Important** 備份到磁碟的組態不會加密。如果您的配置包含敏感數據，AWS AppConfig 建議您使用文件系統權限實踐最低權限原則。如需詳細資訊，請參閱 AWS AppConfig 中的安全性。	無
HTTP_PORT	此環境變數指定代理程式之 HTTP 伺服器執行所在的連接埠。	2772

環境變數	詳細資訊	預設值
LOG_LEVEL	此環境變數指定代理程式記錄的詳細資料層級。每個級別包括當前級別和所有更高級別。這些變量是區分大小寫的。從最詳細到最不詳細，記錄層級為：debuginfowarn、error、和none。 Debug包括有關代理程式的詳細資訊，包括計時資訊。	info
LOG_PATH	寫入記錄檔的磁碟位置。如果未指定，則會將記錄寫入 stderr。	無
MANIFEST	此環境變數會設定 AWS AppConfig 代理程式，以利用額外的每個組態功能，例如多帳戶擷取，以及將組態儲存至磁碟。您可以輸入下列其中一個值： • "app:env:manifest-config" • "file:/fully/qualified/path/to/manifest.json" 如需這些功能的詳細資訊，請參閱其他擷取功能。	true
MAX_CONNECTIONS	此環境變數會設定代理程式用來擷取組態的連線數目上限。AWS AppConfig	3

環境變數	詳細資訊	預設值
POLL_INTERVAL	此環境變數可控制代理程式輪詢 AWS AppConfig 更新組態資料的頻率。您可以指定間隔的秒數。您也可以指定帶有時間單位的數字：s 表示秒，m 表示分鐘，h 表示小時。如果未指定單位，則代理程式預設為秒。例如，60、60 和 1m 會產生相同的輪詢間隔。	四十五秒
PREFETCH_LIST	此環境變數會指定代理程式啟動時立即 AWS AppConfig 要求的組態資料。	無
PRELOAD_BACKUPS	如果設為`true`，則 AWS AppConfig 代理程式會將在記憶體中找到的組態備份載BACKUP_DIRECTORY 入記憶體，並立即檢查服務是否存在較新的版本。如果設定為`false`，則只有當 AWS AppConfig 代理程式無法從服務擷取組態資料時，才會從組態備份載入內容，例如網路發生問題。	true
PROXY_HEADERS	此環境變數會指定環 PROXY_URL 境變數中所參考 Proxy 所需的標頭。值是以逗號分隔的標頭清單。每個標題使用下面的形式。 `"header: value"`	無

環境變數	詳細資訊	預設值
PROXY_URL	此環境變數會指定用於從代理程式到的連線的 Proxy URL AWS 服務，包括 AWS AppConfig。 HTTPS和HTTP網址被支持。	無
REQUEST_TIMEOUT	此環境變數可控制代理程式等待回應的時間量。 AWS AppConfig如果服務沒有回應，請求就會失敗。 如果要求是用於初始資料擷取，則代理程式會將錯誤傳回給您的應用程式。 如果在背景檢查更新資料期間發生逾時，代理程式會記錄錯誤，並在短暫的延遲後再試一次。 您可以指定逾時的毫秒數。您也可以指定具有時間單位的數字：ms 表示毫秒，s 表示秒。如果未指定單位，則代理程式預設為毫秒。例如，5000、5000 毫秒和 5 秒會產生相同的要求逾時值。	三千毫秒
ROLE_ARN	此環境變數指定 IAM 角色的 Amazon 資源名稱 (ARN)。AWS AppConfig 代理程式會假設此角色來擷取組態資料。	無
ROLE_EXTERNAL_ID	此環境變數會指定要與假定角色 ARN 搭配使用的外部 ID。	無

環境變數	詳細資訊	預設值
ROLE_SESSION_NAME	此環境變數會指定要與假設 IAM 角色的登入資料相關聯的工作階段名稱。	無
SERVICE_REGION	此環境變數會指定 AWS AppConfig 代理程式 AWS 區域 用來呼叫 AWS AppConfig 服務的替代方案。如果未定義，代理程式會嘗試判斷目前的「區域」。如果無法啟動，代理程式將無法啟動。	無
WAIT_ON_MANIFEST	此環境變數會設定 AWS AppConfig 代理程式等到資訊清單處理完成後才完成啟動。	true

步驟 5：(必要) 擷取組態資料

您可以使用 HTTP 本機主機呼叫，從 AWS AppConfig 代理程式擷取組態資料。下列範例會`curl`搭配 HTTP 用戶端使用。您可以使用應用程式語言或可用程式庫 (包括 AWS SDK) 支援的任何可用 HTTP 用戶端來呼叫代理程式。

擷取任何已部署組態的完整內容

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name"
```

若要從類型的 AWS AppConfig 組態擷取單一旗標及其屬性 **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name"
```

若要從類型的 AWS AppConfig 組態存取多個旗標及其屬性 **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two"
```

步驟 6 (選擇性，但建議使用)：自動更新代理程式 AWS AppConfig

AWS AppConfig 代理程式會定期更新。為確保您在執行個體上執行最新版本的 AWS AppConfig 代理程式，建議您將下列命令新增至 Amazon EC2 使用者資料。您可以將命令新增至執行個體或 EC2 Auto Scaling 群組上的使用者資料。每次執行個體啟動或重新啟動時，指令碼都會安裝並啟動最新版本的代理程式。

```
#!/bin/bash
# install the latest version of the agent
yum install -y https://s3.amazonaws.com/aws-appconfig-downloads/aws-appconfig-agent/
linux/x86_64/latest/aws-appconfig-agent.rpm
# optional: configure the agent
mkdir /etc/systemd/system/aws-appconfig-agent.service.d
echo "${MY_AGENT_CONFIG}" > /etc/systemd/system/aws-appconfig-agent.service.d/
overrides.conf
systemctl daemon-reload
# start the agent
systemctl start aws-appconfig-agent
```

從 Amazon ECS 和 Amazon EKS 擷取組態資料

您可以通過使用代理 AWS AppConfig 與 Amazon 彈性容器服務（Amazon ECS）和亞馬遜彈性 Kubernetes 服務（亞馬遜 EKS）集成。 AWS AppConfig 此代理程式可作為與 Amazon ECS 和 Amazon EKS 容器應用程式一起執行的附屬容器運作。代理程式以下列方式增強容器化應用程式的處理與管理：

- 代理程式 AWS AppConfig 會使用 AWS Identity and Access Management (IAM) 角色並管理設定資料的本機快取，代表您呼叫。藉由從本機快取提取組態資料，您的應用程式需要較少的程式碼更新來管理組態資料、擷取設定資料 (以毫秒為單位)，而且不會受到可能中斷此類資料呼叫的網路問題影響。 *
- 代理程式提供擷取和解析 AWS AppConfig 功能旗標的原生體驗。
- 代理程式現成可提供快取策略、輪詢間隔和本機組態資料可用性的最佳作法，同時追蹤後續服務呼叫所需的組態 Token。

- 在背景執行時，代理程式會定期輪詢 AWS AppConfig 資料平面以進行組態資料更新。您的容器化應用程序可以通過連接到端口 2772 上的本地主機（可自定義的默認端口值）並調用 HTTP GET 來檢索數據來檢索數據。
- AWS AppConfig 代理程式會更新容器中的組態資料，而不必重新啟動或回收這些容器。

*AWS AppConfig 代理程式會在服務第一次擷取您的組態資料時快取資料。因此，第一次檢索數據的調用比後續調用慢。

主題

開始之前

若要 AWS AppConfig 與容器應用程式整合，您必須建立成 AWS AppConfig 品和組態資料，包括功能旗標或自由格式組態資料。如需詳細資訊，請參閱 在中建立特徵旗標和任意格式組態資料 AWS AppConfig。

若要擷取由託管的組態資料 AWS AppConfig，您的容器應用程式必須設定為具有 AWS AppConfig 資料平面的存取權。若要授予應用程式存取權限，請更新容器服務 IAM 角色所使用的 IAM 許可政策。具體而言，您必須將`appconfig:StartConfigurationSession`和`appconfig:GetLatestConfiguration`動作新增至策略。容器服務 IAM 角色包括下列各項：

- Amazon ECS 任務角色
- Amazon EKS 節點角色
- AWS Fargate (Fargate) 網繭執行角色 (如果您的 Amazon EKS 容器使用 Fargate 進行運算處理)

如需將許可新增至政策的詳細資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

啟動 Amazon ECS 集成的 AWS AppConfig 代理

AWS AppConfig 代理程式邊車容器會自動在您的 Amazon ECS 環境中使用。若要使用 AWS AppConfig 代理程式並行容器，您必須啟動它。

要啟動 Amazon ECS（控制台）

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇應用程式的作業定義，然後選取最新的修訂版本。
4. 選擇建立新修訂、建立新修訂版本。
5. 選擇 [新增更多容器]。
6. 在名稱中，輸入 AWS AppConfig 代理程式容器的唯一名稱。
7. 對於影像 URI，請輸入：**`public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x`**
8. 針對「基本容器」，選擇「是」。
9. 在連接埠對應區段中，選擇新增連接埠對應。
10. 針對「容器連接埠」，輸入**2772**。

> **Note**
>
> AWS AppConfig 根據預設，代理程式會在連接埠 2772 上執行。您可以指定不同的連接埠。

11. 選擇建立。Amazon ECS 會建立新的容器修訂版本並顯示詳細資訊。
12. 在瀏覽窗格中，選擇 [叢集]，然後在清單中選擇您的應用程式叢集。
13. 在 [服務] 索引標籤上，選取您應用程式的服務。
14. 選擇更新。
15. 在「部署規劃」下，對於「修訂」，選擇最新的修訂版。
16. 選擇更新。Amazon ECS 部署了最新的任務定義。
17. 部署完成後，您可以在 [組態和工作] 索引標籤上確認 AWS AppConfig 代理程式是否正在執行。在 [工作] 索引標籤上，選擇執行中的工作。
18. 在 [容器] 區段中，確認已列出 AWS AppConfig 代理程式容器。

19. 如果要確認 AWS AppConfig 代理程式已啟動，請選擇記錄檔索引標籤 為 AWS AppConfig 代理程式容器找出類似下列的陳述式：`[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> 您可以輸入或變更環境變數，以調整 AWS AppConfig 代理程式的預設行為。如需有關可用環境變數的資訊，請參閱使用環境變數設定 Amazon ECS 和 Amazon EKS 的 AWS AppConfig 代理程式。如需如何在 Amazon ECS 中變更環境變數的相關資訊，請參閱 Amazon 彈性容器服務開發人員指南中的將環境變數傳遞至容器。

啟動 Amazon EKS 集成的 AWS AppConfig 代理

AWS AppConfig 代理程式邊車容器會自動在您的 Amazon EKS 環境中使用。若要使用 AWS AppConfig 代理程式並行容器，您必須啟動它。下列程序說明如何使用 Amazon EKS `kubectl` 命令列工具在容器應用程式的`kubeconfig`檔案中進行變更。如需有關建立或編輯`kubeconfig`檔案的詳細資訊，請參閱 Amazon EKS 使用者指南中的為 Amazon EKS 叢集建立或更新 Kubeconfig 檔案。

若要啟動 AWS AppConfig 代理程式 (kubectl 命令列工具)

1. 開啟`kubeconfig`檔案並確認 Amazon EKS 應用程式是以單一容器部署的形式執行。檔案的內容看起來應類似下列內容。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-application-label
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-application-label
  template:
    metadata:
      labels:
        app: my-application-label
```

```
    spec:
      containers:
      - name: my-app
        image: my-repo/my-image
        imagePullPolicy: IfNotPresent
```

2. 將 AWS AppConfig 代理程式容器定義詳細資料新增至您的 YAML 部署檔案。

```
- name: appconfig-agent
        image: public.ecr.aws/aws-appconfig/aws-appconfig-agent:2.x
        ports:
        - name: http
          containerPort: 2772
          protocol: TCP
        env:
        - name: SERVICE_REGION
          value: region
        imagePullPolicy: IfNotPresent
```

> **Note**
>
> 記下以下資訊。
>
> - AWS AppConfig 根據預設，代理程式會在連接埠 2772 上執行。您可以指定不同的連接埠。
> - 您可以輸入環境變數來調整 AWS AppConfig 代理程式的預設行為。如需詳細資訊，請參閱 使用環境變數設定 Amazon ECS 和 Amazon EKS 的 AWS AppConfig 代理程式。
> - 針對 *SERVICE_REGION*，指定 AWS 區域 代 AWS AppConfig 理程式擷取組態資料的程式碼 (例如，us-west-1)。

3. 在kubectl工具中執行下列命令，將變更套用至叢集。

```
kubectl apply -f my-deployment.yml
```

4. 部署完成後，請確認 AWS AppConfig 代理程式是否正在執行。使用下列命令來檢視應用程式網繭記錄檔。

```
kubectl logs -n my-namespace -c appconfig-agent my-pod
```

為 AWS AppConfig 代理程式容器找出類似下列的陳述式：`[appconfig agent] 1970/01/01 00:00:00 INFO serving on localhost:2772`

> **Note**
>
> 您可以輸入或變更環境變數，以調整 AWS AppConfig 代理程式的預設行為。如需有關可用環境變數的資訊，請參閱使用環境變數設定 Amazon ECS 和 Amazon EKS 的 AWS AppConfig 代理程式。

使用環境變數設定 Amazon ECS 和 Amazon EKS 的 AWS AppConfig 代理程式

您可以透過變更 AWS AppConfig 代理程式容器的下列環境變數來設定代理程式。

環境變數	詳細資訊	預設值

ACCESS_TOKEN	此環境變數定義了從代理程式 HTTP 伺服器要求組態資料時必須提供的權杖。權杖的值必須在具有授權類型的 HTTP 要求授權標頭中設定`Bearer`。請見此處範例。
	`GET /applications/my_app/...`
	`Host: localhost:2772`
	`Authorization: Bearer <token value>`
		無

BACKUP_DIRECTORY	此環境變數可讓 AWS AppConfig 代理程式將擷取之每個組態的備份儲存至指定目錄。	無

環境變數	詳細資訊	預設值
	Important 備份到磁碟的組態不會加密。如果您的配置包含敏感數據， AWS AppConfig 建議您使用文件系統權限實踐最低權限原則。如需詳細資訊，請參閱 AWS AppConfig 中的安全性。	
HTTP_PORT	此環境變數指定代理程式之 HTTP 伺服器執行所在的連接埠。	2772
LOG_LEVEL	此環境變數指定代理程式記錄的詳細資料層級。每個級別包括當前級別和所有更高級別。這些變量是區分大小寫的。從最詳細到最不詳細，記錄層級為：debuginfowarn、error、和none。 Debug包括有關代理程式的詳細資訊，包括計時資訊。	info

環境變數	詳細資訊	預設值
MANIFEST	此環境變數會設定 AWS AppConfig 代理程式，以利用額外的每個組態功能，例如多帳戶擷取，以及將組態儲存至磁碟。您可以輸入下列其中一個值： • "app:env:manifest-config" • "file:/fully/qualified/path/to/manifest.json" 如需這些功能的詳細資訊，請參閱其他擷取功能。	true
MAX_CONNECTIONS	此環境變數會設定代理程式用來擷取組態的連線數目上限。AWS AppConfig	3
POLL_INTERVAL	此環境變數可控制代理程式輪詢 AWS AppConfig 更新組態資料的頻率。您可以指定間隔的秒數。您也可以指定帶有時間單位的數字：s 表示秒，m 表示分鐘，h 表示小時。如果未指定單位，則代理程式預設為秒。例如，60、60 和 1m 會產生相同的輪詢間隔。	四十五秒
PREFETCH_LIST	此環境變數會指定代理程式啟動時立即 AWS AppConfig 要求的組態資料。	無

環境變數	詳細資訊	預設值
PRELOAD_BACKUPS	如果設為`true`，則 AWS AppConfig 代理程式會將在記憶體中找到的組態備份載BACKUP_DIRECTORY 入記憶體，並立即檢查服務是否存在較新的版本。如果設定為`false`，則只有當 AWS AppConfig 代理程式無法從服務擷取組態資料時，才會從組態備份載入內容，例如網路發生問題。	true
PROXY_HEADERS	此環境變數會指定環 PROXY_URL 境變數中所參考 Proxy 所需的標頭。值是以逗號分隔的標頭清單。每個標題使用下面的形式。 `"header: value"`	無
PROXY_URL	此環境變數會指定用於從代理程式到的連線的 Proxy URL AWS 服務，包括 AWS AppConfig。 HTTPS和HTTP網址被支持。	無

環境變數	詳細資訊	預設值
REQUEST_TIMEOUT	此環境變數可控制代理程式等待回應的時間量。 AWS AppConfig如果服務沒有回應，請求就會失敗。 如果要求是用於初始資料擷取，則代理程式會將錯誤傳回給您的應用程式。 如果在背景檢查更新資料期間發生逾時，代理程式會記錄錯誤，並在短暫的延遲後再試一次。 您可以指定逾時的毫秒數。您也可以指定具有時間單位的數字：ms 表示毫秒，s 表示秒。如果未指定單位，則代理程式預設為毫秒。例如，5000、5000 毫秒和 5 秒會產生相同的要求逾時值。	三千毫秒
ROLE_ARN	此環境變數指定 IAM 角色的 Amazon 資源名稱 (ARN)。AWS AppConfig 代理程式會假設此角色來擷取組態資料。	無
ROLE_EXTERNAL_ID	此環境變數會指定要與假定角色 ARN 搭配使用的外部 ID。	無
ROLE_SESSION_NAME	此環境變數會指定要與假設 IAM 角色的登入資料相關聯的工作階段名稱。	無

環境變數	詳細資訊	預設值
SERVICE_REGION	此環境變數會指定 AWS AppConfig 代理程式 AWS 區域 用來呼叫 AWS AppConfig 服務的替代方案。如果未定義，代理程式會嘗試判斷目前的「區域」。如果無法啟動，代理程式將無法啟動。	無
WAIT_ON_MANIFEST	此環境變數會設定 AWS AppConfig 代理程式等到資訊清單處理完成後才完成啟動。	true

擷取組態資料

您可以使用 HTTP 本機主機呼叫，從 AWS AppConfig 代理程式擷取組態資料。下列範例會curl搭配 HTTP 用戶端使用。您可以使用應用程式語言或可用程式庫支援的任何可用 HTTP 用戶端來呼叫代理程式。

> **Note**
>
> 如果您的應用程序使用正斜杠（例如「測試後端/測試服務」）來檢索配置數據，則需要使用 URL 編碼。

擷取任何已部署組態的完整內容

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name"
```

若要從類型的 AWS AppConfig 組態擷取單一旗標及其屬性 **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?flag=flag_name"
```

若要從類型的 AWS AppConfig 組態存取多個旗標及其屬性 **`Feature Flag`**

```
$ curl "http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name?
flag=flag_name_one&flag=flag_name_two"
```

其他擷取功能

AWS AppConfig 代理程式提供下列其他功能，協助您擷取應用程式的組態。

- 多賬戶檢索：使用主要帳戶或擷 AWS 帳戶 取中的 AWS AppConfig 代理程式，從多個廠商帳戶擷取組態資料。
- 將組態複製寫入磁碟：使用「 AWS AppConfig 代理程式」將組態資料寫入磁碟。此功能可讓客戶擁有從磁碟讀取組態資料的應用程式以進行整合 AWS AppConfig。

關於代理程式清單

若要啟用這些 AWS AppConfig 代理程式功能，您可以建立資訊清單。資訊清單是您提供用來控制代理程式可執行動作的一組組態資料。清單是用 JSON 編寫的。它包含一組頂級密鑰，對應於您使用部署的不同配置 AWS AppConfig。

資訊清單可以包含多個組態。此外，資訊清單中的每個組態都可以識別要用於指定組態的一或多個代理程式功能。資訊清單的內容使用下列格式：

```
{
    "application_name:environment_name:configuration_name": {
        "agent_feature_to_enable_1": {
            "feature-setting-key": "feature-setting-value"
        },
        "agent_feature_to_enable_2": {
            "feature-setting-key": "feature-setting-value"
        }
    }
}
```

以下是具有兩個配置的清單的示例 JSON。第一個組態 (*MyApp*) 不使用任何 AWS AppConfig 代理程式功能。第二個配置（ *my2ndApp* ）使用寫入配置複製到磁盤和多帳戶檢索功能：

```
{
        "MyApp:Test:MyAllowListConfiguration": {},
```

```
        "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
            "credentials": {
                "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
                "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
                "roleSessionName": "AwsAppConfigAgent",
                "credentialsDuration": "2h"
            },
            "writeTo": {
                "path": "/tmp/aws-appconfig/my-2nd-app/beta/my-enable-payments-feature-
flag-configuration.json"
            }
        }
    }
```

如何提供代理清單

您可以將資訊清單儲存為檔案， AWS AppConfig 代理程式可以讀取資訊清單的位置。或者，您可以將資訊清單儲存為 AWS AppConfig 組態，並將代理程式指向它。若要提供代理程式資訊清單，您必須使用下列其中一個值來設定MANIFEST環境變數：

清單位置	環境變數值	使用案例
檔案	文件：/路徑/到代理清單 .json	如果您的資訊清單不會經常變更，請使用此方法。
AWS AppConfig 配置	*######:####:####*	使用此方法進行動態更新。您可以使用與儲存其他組態相同的方式，更新和部署儲存 AWS AppConfig 為組 AWS AppConfig 態中的資訊清單。
環境變數	資訊清單內容	如果您的資訊清單不會經常變更，請使用此方法。在容器環境中，設定環境變數比公開檔案更容易，此方法非常有用。

如需有關為 AWS AppConfig 代理程式設定變數的詳細資訊，請參閱您使用案例的相關主題：

- 設定 AWS AppConfig 代理程式 Lambda 延伸

- 使用 AWS AppConfig 代理與 Amazon EC2
- 使用 AWS AppConfig 代理與 Amazon ECS 和 Amazon EKS

多賬戶檢索

您可以在 AWS AppConfig 代理程式資訊清單中輸入認證覆寫，將 AWS AppConfig 代理程式設定為從多個 AWS 帳戶 擷取組態。登入資料覆寫包括 (IAM) 角色的 Amazon 資源名稱 AWS Identity and Access Management (ARN)、角色 ID、工作階段名稱，以及代理程式可擔任該角色的持續時間。

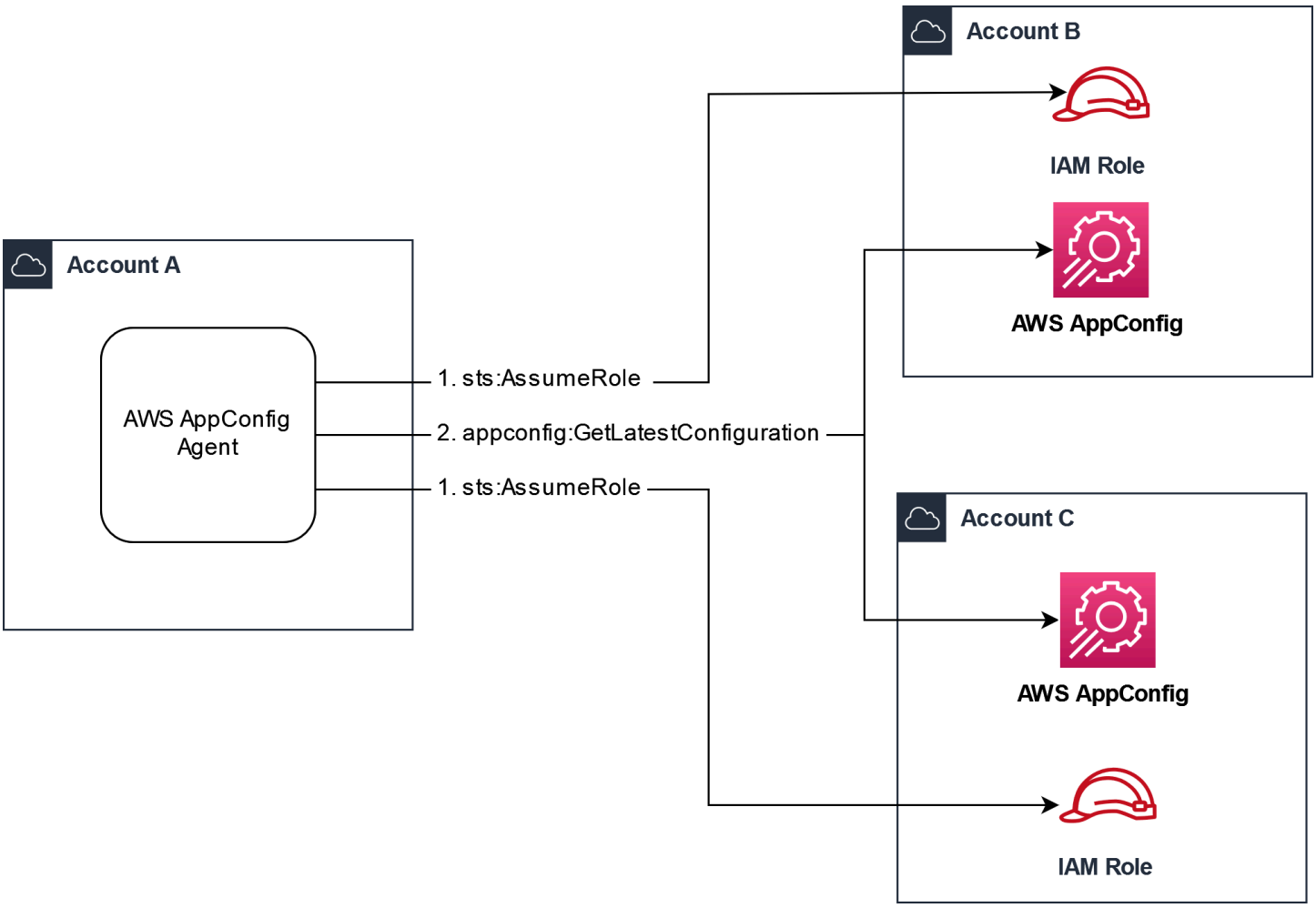
您可以在資訊清單的「認證」區段中輸入這些詳細資料。「認證」區段使用下列格式：

```
{
    "application_name:environment_name:configuration_name": {
        "credentials": {
            "roleArn": "arn:partition:iam::account_ID:role/roleName",
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
        }
    }
}
```

請見此處範例：

```
{
    "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
        "credentials": {
            "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
            "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
            "roleSessionName": "AWSAppConfigAgent",
            "credentialsDuration": "2h"
        }
    }
}
```

在擷取組態之前，代理程式會從資訊清單讀取組態的認證詳細資料，然後假設為該組態指定的 IAM 角色。您可以在單一資訊清單中為不同的組態指定一組不同的認證覆寫。下圖顯示 AWS AppConfig 代理程式在帳戶 A (擷取帳戶) 中執行時，如何假設為帳戶 B 和 C (廠商帳戶) 指定的個別角色，然後呼叫 GetLatestConfiguration API 作業以擷取在這些帳戶中 AWS AppConfig 執行的組態資料：



配置從供應商帳戶檢索配置數據的權限

AWS AppConfig 在擷取帳戶中執行的代理程式需要從廠商帳戶擷取組態資料的權限。您可以透過在每個供應商帳戶中建立 AWS Identity and Access Management (IAM) 角色來授予代理程式權限。 AWS AppConfig 擷取帳戶中的代理程式會擔任此角色，可從廠商帳戶取得資料。完成本節中的程序，以建立 IAM 許可政策、IAM 角色，並將代理程式覆寫新增至資訊清單。

開始之前

在 IAM 中建立權限政策和角色之前，請先收集下列資訊。

- 每個識別碼 AWS 帳戶。擷取帳戶是將呼叫其他帳戶以取得設定資料的帳戶。廠商帳戶是將組態資料傳送至擷取帳戶的帳戶。
- 擷取帳戶中使用 AWS AppConfig 的 IAM 角色名稱。以下是預設使用的 AWS AppConfig角色清單：
  - 對於亞馬遜彈性運算雲端 (Amazon EC2)，請 AWS AppConfig 使用執行個體角色。
  - 對於 AWS Lambda， AWS AppConfig 使用 Lambda 執行角色。

- 對於 Amazon Elastic Container Service (Amazon ECS) 和 Amazon Elastic Kubernetes Service（Amazon EKS），AWS AppConfig 使用容器角色。

如果您透過指定ROLE_ARN環境變數將 AWS AppConfig Agent 設定為使用不同的 IAM 角色，請記下該名稱。

建立權限原則

使用下列程序，使用 IAM 主控台建立許可政策。完成每個 AWS 帳戶 將為擷取帳戶顯示組態資料的程序。

建立 IAM 政策

1. 登入供應商帳戶。 AWS Management Console
2. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
3. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
4. 選擇「JSON」選項。
5. 在 [原則編輯器] 中，以下列原則陳述式取代預設 JSON。使用供應商帳號詳細############

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "appconfig:StartConfigurationSession",
            "appconfig:GetLatestConfiguration"
        ],
        "Resource":
 "arn:partition:appconfig:region:vendor_account_ID:application/
vendor_application_ID/environment/vendor_environment_ID/
configuration/vendor_configuration_ID"
    }
   ]
}
```

範例如下：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "appconfig:StartConfigurationSession",
            "appconfig:GetLatestConfiguration"
        ],
        "Resource": "arn:aws:appconfig:us-east-2:111122223333:application/abc123/
environment/def456/configuration/hij789"
    }
   ]
}
```

6. 選擇下一步。
7. 在「策略名稱」欄位中，輸入名稱。
8. (選擇性) 對於 [新增標籤]，請新增一或多個標籤金鑰值組，以組織、追蹤或控制此原則的存取權。
9. 選擇建立政策。系統會讓您返回 Policies (政策) 頁面。
10. 在每個將為擷取帳戶顯 AWS 帳戶 示組態資料的每個程序中重複此程序。

建立 IAM 角色

使用下列程序，使用 IAM 主控台建立 IAM 角色。完成每個 AWS 帳戶 將為擷取帳戶顯示組態資料的程序。

若要建立一個 IAM 角色

1. 登入供應商帳戶。 AWS Management Console
2. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
3. 在瀏覽窗格中，選擇 [角色]，然後選擇 [建立原則]。
4. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 帳戶。
5. 在AWS 帳戶區段中，選擇 [其他] AWS 帳戶。
6. 在「帳戶 ID」欄位中，輸入擷取帳號 ID。
7. (選擇性) 做為此假設角色的安全性最佳作法，請選擇需要外部 ID 並輸入字串。
8. 選擇下一步。
9. 在 [新增權限] 頁面上，使用 [搜尋] 欄位尋找您在上一個程序中建立的原則。選取其名稱旁的核取方塊。
10. 選擇下一步。

11. 在 Role name (角色名稱) 中，輸入名稱。
12. 在描述，請輸入描述。
13. 對於步驟 1：選取信任的實體，請選擇編輯。以下列原則取代預設 JSON 信任原則。使用擷取帳號中的*###############*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::retrieval_account_ID:role/appconfig_role_in_retrieval_account"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

14. (選用) 針對 Tags (標籤)，新增一個或多個標籤鍵值組來組織、追蹤或控制對此角色的存取。
15. 選擇 Create role (建立角色)。系統會讓您回到 Roles (角色) 頁面。
16. 搜尋您剛建立的角色。請選擇此群組。在 ARN 區段中，複製 ARN。您將在下一個程序中指定此資訊。

將認證覆寫新增至資訊清單

在供應商帳戶中建立 IAM 角色後，請在擷取帳戶中更新資訊清單。具體而言，新增登入資料區塊和 IAM 角色 ARN，以從供應商帳戶擷取組態資料。以下是 JSON 格式：

```
{
    "vendor_application_name:vendor_environment_name:vendor_configuration_name": {
        "credentials": {
            "roleArn":
 "arn:partition:iam::vendor_account_ID:role/name_of_role_created_in_vendor_account",
            "roleExternalId": "string",
            "roleSessionName": "string",
            "credentialsDuration": "time_in_hours"
        }
    }
}
```

請見此處範例：

```
{
    "My2ndApp:Beta:MyEnableMobilePaymentsFeatureFlagConfiguration": {
        "credentials": {
            "roleArn": "arn:us-west-1:iam::123456789012:role/MyTestRole",
            "roleExternalId": "00b148e2-4ea4-46a1-ab0f-c422b54d0aac",
            "roleSessionName": "AwsAppConfigAgent",
            "credentialsDuration": "2h"
        }
    }
}
```

驗證多帳戶擷取是否正常運作

您可以檢視代理程式記錄，驗證該代理程式是否能夠從多個帳戶擷取組態資料。 AWS AppConfig 擷取 'YourApplicationNameYourEnvironmentName:YourConfigurationName' 初始資料的INFO層級記錄是成功擷取的最佳指標。如果擷取失敗，您應該會看到指出失敗原因的ERROR層級記錄。以下是從供應商帳戶成功檢索的示例：

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MyTestApplication:MyTestEnvironment:MyDenyListConfiguration' in XX.Xms
```

將組態複製寫入磁碟

您可以將 AWS AppConfig 代理程式設定為以純文字自動將組態複本儲存到磁碟。此功能可讓客戶擁有從磁碟讀取組態資料的應用程式以進行整合 AWS AppConfig。

此功能的設計不是用來做為組態備份功能。 AWS AppConfig 代理程式不會從複製到磁碟的組態檔讀取。如果您想要將組態備份到磁碟，請參閱將AWS AppConfig 代理程式與 Amazon EC2 搭配使用或搭配 Amazon ECS 和 Amazon EKS 使用 AWS AppConfig 代理程式的PRELOAD_BACKUP環境變數。BACKUP_DIRECTORY

> **Warning**
>
> 請注意下列有關此功能的重要資訊：
>
> - 儲存至磁碟的組態會以純文字格式儲存，且可供人類讀取。對於包含敏感資料的組態，請勿啟用此功能。

- 此功能會寫入本機磁碟。使用檔案系統權限的最小權限原則。如需詳細資訊，請參閱 實作最低權限存取。

啟用寫入組態複製到磁碟

1. 編輯資訊清單。
2. 選擇您要 AWS AppConfig 寫入磁碟的組態並新增`writeTo`元素。請見此處範例：

```
{
    "application_name:environment_name:configuration_name": {
        "writeTo": {
            "path": "path_to_configuration_file"
        }
    }
}
```

請見此處範例：

```
{
    "MyTestApp:MyTestEnvironment:MyNewConfiguration": {
        "writeTo": {
            "path": "/tmp/aws-appconfig/mobile-app/beta/enable-mobile-payments"
        }
    }
}
```

3. 儲存您的變更。每次部署新的組態資料時，組態 .json 檔案都會更新。

驗證將組態複本寫入磁碟是否正常運作

您可以檢視 AWS AppConfig 代理程式記錄，以驗證組態的副本是否正在寫入磁碟。*###INFO #INFO################ file_path######## AWS AppConfig ##############*

請見此處範例：

```
[appconfig agent] 2023/11/13 11:33:27 INFO AppConfig Agent 2.0.x
[appconfig agent] 2023/11/13 11:33:28 INFO serving on localhost:2772
[appconfig agent] 2023/11/13 11:33:28 INFO retrieved initial data for
 'MobileApp:Beta:EnableMobilePayments' in XX.Xms
```

```
[appconfig agent] 2023/11/13 17:05:49 INFO wrote configuration
 'MobileApp:Beta:EnableMobilePayments' to /tmp/configs/your-app/your-env/your-
config.json
```

AWS AppConfig 代理當地開發

AWS AppConfig 代理程式支援本機開發模式。如果您啟用本機開發模式，代理程式會從磁碟上的指定目錄讀取組態資料。它不會從中檢索配置數據 AWS AppConfig。您可以透過更新指定目錄中的檔案來模擬組態部署。對於下列使用案例，我們建議使用本機開發模式：

- 在使用部署不同的配置版本之前測試它們 AWS AppConfig。
- 在將變更送至程式碼儲存庫之前，請先測試新功能的不同組態選項。
- 測試不同的組態案例，以確認它們如預期般運作。

> **Warning**
>
> 不要在生產環境中使用本地開發模式。此模式不支援重要的 AWS AppConfig 安全功能，例如部署驗證和自動復原。

使用下列程序來為本機開發模式設定 AWS AppConfig 代理程式。

若要設定本機開發模式的 AWS AppConfig 代理程式

1. 使用針對您的計算環境所描述的方法安裝代理程式。 AWS AppConfig 代理程式適用於下列項目 AWS 服務：
   - AWS Lambda
   - Amazon EC2
   - Amazon ECS 和 Amazon EKS
2. 如果代理程式正在執行，請停止代理程式。
3. 新增LOCAL_DEVELOPMENT_DIRECTORY至環境變數清單。在檔案系統上指定提供代理程式讀取權限的目錄。例如 `/tmp/local_configs`。
4. 在目錄中創建一個文件。檔案名稱必須使用下列格式：

   ```
   application_name:environment_name:configuration_profile_name
   ```

請見此處範例：

```
Mobile:Development:EnableMobilePaymentsFeatureFlagConfiguration
```

> **Note**
>
> (選擇性) 您可以根據您提供的檔案副檔名，控制代理程式針對組態資料傳回的內容類型。例如，如果您以 .json 副檔名命名檔案，則代理程式會在您的應用程式要求`application/json`時傳回內容類型。如果您省略擴充功能，代理程式會用`application/octet-stream`於內容類型。如果您需要精確控制，則可以提供格式的擴展名`.type%subtype`。代理程式會傳回的內容類型`.type/subtype`。

5. 執行下列命令以重新啟動代理程式並要求組態資料。

```
curl http://localhost:2772/applications/application_name/
environments/environment_name/configurations/configuration_name
```

代理程式會按照為代理程式指定的輪詢間隔檢查本機檔案的變更。如果未指定輪詢間隔，則代理程式會使用預設間隔 45 秒。輪詢間隔時的這項檢查可確保代理程式在本機開發環境中的行為與設定為與 AWS AppConfig 服務互動時的行為相同。

> **Note**
>
> 若要部署新版本的本機開發設定檔，請使用新資料更新檔案。

透過直接呼叫 API 擷取組態

您的應用程式會先使用工作階段 API 作業建立組態工作階StartConfiguration段來擷取組態資料。然後，您的會話的客戶端定期調用 GetLatestConfiguration 以檢查和檢索可用的最新數據。

呼叫時`StartConfigurationSession`，您的程式碼會傳送下列資訊：

- 工作階段追蹤之 AWS AppConfig 應用程式、環境和組態設定檔的識別碼 (ID 或名稱)。
- (選擇性) 工作階段用戶端在呼叫之間必須等待的最短時間量`GetLatestConfiguration`。

作GetLatestConfiguration為回應， AWS AppConfig 提供InitialConfigurationToken給工作階段的用戶端，並在第一次呼叫該工作階段時使用。

> **Important**
>
> 此令牌只能在您第一次呼叫時使用一次GetLatestConfiguration。您必須在每次後續呼叫的 GetLatestConfiguration response (NextPollConfigurationToken) 中使用新的權杖GetLatestConfiguration。為了支持長輪詢用例，令牌的有效期長達 24 小時。如果GetLatestConfiguration調用使用過期令牌，則系統返回BadRequestException。

在調用時GetLatestConfiguration，您的客戶端代碼發送它具有並接收的最新ConfigurationToken值作為響應：

- NextPollConfigurationToken：要在下次呼叫時使用的ConfigurationToken值GetLatestConfiguration。
- NextPollIntervalInSeconds：用戶端在下次呼叫之前應等待的持續時間GetLatestConfiguration。
- 配置：用於會話的最新數據。如果用戶端已經擁有最新版本的組態，則此選項可能是空的。

> **Important**
>
> 記下以下重要資訊。
>
> - 每個應用程式、環境、組態設定檔和用戶端只能呼叫工作階段 API 一次，以建立與服務的工作階段。StartConfiguration這通常是在應用程式啟動時完成，或在第一次擷取組態之前完成。
> - 如果您的組態是使用部署KmsKeyIdentifier，則接收組態的請求必須包含呼叫的權限kms:Decrypt。如需詳細資訊，請參閱 AWS Key Management Service API 參考中的解密。
> - 先前用來擷取設定資料的 API 作業已取代。GetConfigurationGetConfigurationAPI 操作不支持加密配置。

擷取組態範例

下列 AWS CLI 範例示範如何使用 Data StartConfigurationSession 和 GetLatestConfiguration API 作業擷取設定 AWS AppConfig 資料。第一個指令會啟動組態工作階段。此呼叫包括 AWS AppConfig 應用程式的 ID (或名稱)、環境和組態設定檔。該 API 返回InitialConfigurationToken用於獲取配置數據。

```
aws appconfigdata start-configuration-session \
    --application-identifier application_name_or_ID \
    --environment-identifier environment_name_or_ID \
    --configuration-profile-identifier configuration_profile_name_or_ID
```

系統會以下列格式回應相關資訊。

```
{
   "InitialConfigurationToken": initial configuration token
}
```

啟動會話後，使用InitialConfiguration令牌調用GetLatest配置以獲取配置數據。組態資料會儲存到mydata.json檔案中。

```
aws appconfigdata get-latest-configuration \
    --configuration-token initial configuration token mydata.json
```

第一次調用GetLatestConfiguration使用從ConfigurationToken獲得的StartConfigurationSession。會傳回下列資訊。

```
{
    "NextPollConfigurationToken" : next configuration token,
    "ContentType" : content type of configuration,
    "NextPollIntervalInSeconds" : 60
}
```

後續呼叫必GetLatestConfiguration須NextPollConfigurationToken從先前的回應中提供。

```
aws appconfigdata get-latest-configuration \
    --configuration-token next configuration token mydata.json
```

Important

請注意下列有關 GetLatestConfiguration API 作業的重要詳細資訊：

- 回GetLatestConfiguration應包括顯示Configuration組態資料的區段。只有在系統找到新的或更新的組態資料時，Configuration區段才會出現。如果系統找不到新的或更新的組態資料，則Configuration資料為空。
- 您會ConfigurationToken在每個回覆中收到新的GetLatestConfiguration。
- 建議您根據預算、組態部署的預期頻率，以及組態的目標數目，調整 GetLatestConfiguration API 呼叫的輪詢頻率。

使用擴展擴展工作流

擴充功能可增強您在建立或部署組態的 AWS AppConfig 工作流程期間，在不同點插入邏輯或行為的能力。例如，您可以使用擴展來執行以下類型的任務（僅舉幾例）：

- 部署組態設定檔時，傳送通知至 Amazon 簡單通知服務 (Amazon SNS) 主題。
- 在部署開始之前，清除敏感資料的組態設定檔內容。
- 每當對功能標誌進行變更時，都會建立或更新 Atlassian Jira 問題。
- 啟動部署時，將服務或資料來源的內容合併到您的組態資料中。
- 每當部署組態時，將組態備份到 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體。

您可以將這些類型的工作與 AWS AppConfig 應用程式、環境和組態設定檔建立關聯。

目錄

關於 AWS AppConfig 擴展

本主題介紹 AWS AppConfig 擴充功能概念和術語。這些資訊會在設定和使用 AWS AppConfig 延伸模組所需的每個步驟的前後關聯中討論。

主題

步驟 1：確定您要使用擴展程序執行的操作

您是否要收到 Webhook 的通知，該通知會在 AWS AppConfig 部署完成時向 Slack 傳送訊息？ 是否要在部署組態之前將組態設定檔備份到 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體？ 是否要在部署組

態之前清除機密資訊的組態資料？ 您可以使用擴充功能來執行這些類型的工作等。您可以建立自訂擴充功能，或使用隨附的 AWS 編寫擴充功能。 AWS AppConfig

> **Note**
>
> 對於大多數使用案例，若要建立自訂擴充功能，您必須建立 AWS Lambda 函數來執行擴充功能中定義的任何計算和處理。如需詳細資訊，請參閱 逐步解說：建立自訂 AWS AppConfig 延伸。

下列 AWS 撰寫的延伸功能可協助您快速整合組態部署與其他服務。您可以在 AWS AppConfig 主控台中使用這些擴充功能，或直接從 AWS CLI AWS Tools for PowerShell、或 SDK 呼叫擴充功能 API 動作。

延伸	描述
Amazon 明 CloudWatch 显 A/B測試	此擴充功能可讓您的應用程式在本機指派變體給使用者工作階段，而不是呼叫EvaluateFeature作業。如需詳細資訊，請參閱 與 Amazon CloudWatch 顯然擴展工作。
AWS AppConfig 部署事件 EventBridge	此擴充功能會在部署組態時，將事件傳送至 EventBridge 預設事件匯流排。
AWS AppConfig 部署事件至 Amazon Simple Notification Service (Amazon SNS)	此擴充功能會將訊息傳送至您在部署組態時指定的 Amazon SNS 主題。
AWS AppConfig 部署事件至 Amazon Simple Queue Service (Amazon SQS)	此延伸模組會在部署組態時將訊息排入 Amazon SQS 佇列中。
整合延伸模組 — 阿特拉西亞吉拉	此擴充功能可 AWS AppConfig 讓您在變更功能旗標時建立和更新問題。

步驟 2：確定何時要執行擴充功能

擴充功能會定義它在工作流程期間執行的一或多個動 AWS AppConfig 作。例如， AWS 編寫的AWS AppConfig deployment events to Amazon SNS擴充功能包含一個動作，可將通知傳送至

Amazon SNS 主題。當您與之互動 AWS AppConfig 或代表您執行處理程序 AWS AppConfig 時，都會叫用每個動作。這些被稱為行動點。 AWS AppConfig 擴充功能支援下列動作要點：

- PRE_CREATE_HOSTED_CONFIGURATION_VERSION
- PRE_START_DEPLOYMENT
- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

在動作點上設定的延伸PRE_*動作會在請求驗證後套用，但在 AWS AppConfig 執行與動作點名稱相對應的活動之前。這些動作呼叫會與要求同時處理。如果發出一個以上的請求，動作調用按順序運行。另外請注意，PRE_*動作點會接收並且可以變更組態的內容。 PRE_*動作點也可以回應錯誤並防止動作發生。

擴充功能也可以使用ON_*動作點與 AWS AppConfig 工作流程 parallel 執行。 ON_*動作點是異步調用的。 ON_*動作點不會收到配置的內容。如果延伸功能在ON_*動作點期間遇到錯誤，服務會忽略錯誤並繼續工作流程。

步驟 3：建立擴充功能關聯

若要建立擴充功能或設定 AWS 已編寫的擴充功能，您可以定義在使用特定 AWS AppConfig 資源時叫用擴充功能的動作點。例如，您可以在針對特定應用程式啟動組態部署時，選擇執行AWS AppConfig deployment events to Amazon SNS擴充功能並接收 Amazon SNS 主題的通知。定義哪些動作點叫用特定 AWS AppConfig 資源的擴充功能稱為擴充功能關聯。擴充功能關聯是擴充功能與 AWS AppConfig 資源 (例如應用程式或組態設定檔) 之間的指定關係。

單一 AWS AppConfig 應用程式可以包含多個環境和組態設定檔。如果您將擴充功能與應用程式或環境產生關聯，則會針對與應用程式或環境資源相關的任何工作流程 (如果適用) AWS AppConfig 叫用擴充程式。

例如，假設您有一個名為的 AWS AppConfig 應用程式 MobileApps ，其中包含名為的設定描述檔 AccessList。並假設該 MobileApps 應用程序包括 Beta，集成和生產環境。您可以為 AWS 編寫的 Amazon SNS 通知延伸模組建立擴充功能關聯，並將擴充功能與 MobileApps 應用程式相關聯。只要將應用程式的組態部署到三個環境中的任何一個，就會叫用 Amazon SNS 通知延伸模組。

> **Note**
>
> 您不需要建立擴充功能即可使用已 AWS 編寫的擴充功能，但您必須建立擴充功能關聯。

步驟 4：部署設定並確認已執行擴充動作

建立關聯之後，建立託管組態或部署組態時，會 AWS AppConfig 叫用擴充功能並執行指定的動作。呼叫擴充功能時，如果系統在PRE-*動作點期間遇到錯誤，則會 AWS AppConfig 傳回該錯誤的相關資訊。

使用 AWS 已編寫的擴充功能

AWS AppConfig 包括以下 AWS 編寫的擴充功能。這些擴充功能可協助您將 AWS AppConfig 工作流程與其他服務整合。您可以直接從、 AWS Management Console 或 SDK 呼叫擴充 API 動作 AWS CLI， AWS Tools for PowerShell在或中使用這些擴充功能。

延伸	描述
Amazon 顯示的 CloudWatch A/B測試	此擴充功能可讓您的應用程式在本機指派變體給使用者工作階段，而不是呼叫EvaluateFeature作業。如需詳細資訊，請參閱 與 Amazon CloudWatch 顯然擴展工作。
AWS AppConfig 部署事件 EventBridge	此擴充功能會在部署組態時，將事件傳送至 EventBridge 預設事件匯流排。
AWS AppConfig 部署事件至 Amazon Simple Notification Service (Amazon SNS)	此擴充功能會將訊息傳送至您在部署組態時指定的 Amazon SNS 主題。
AWS AppConfig 部署事件至 Amazon Simple Queue Service (Amazon SQS)	此延伸模組會在部署組態時將訊息排入 Amazon SQS 佇列中。
整合延伸模組 — 阿特拉西亞吉拉	此擴充功能可 AWS AppConfig 讓您在變更功能旗標時建立和更新問題。

與 Amazon CloudWatch 顯然擴展工作

在推出此功能時，您可以使用 Amazon CloudWatch Evitivity 將新功能提供給指定百分比的使用者，以安全地驗證新功能。您可以監控新功能的效能，以協助您決定何時要提升使用者的流量。這可協助您在完全啟動功能之前降低風險並識別意外的後果。您也可以執行 A/B 實驗，根據證據和資料作出功能設計決策。

CloudWatch Evitional 的 AWS AppConfig 擴展允許您的應用程序在本地分配變體給用戶會話，而不是通過調用EvaluateFeature操作。本機工作階段可降低 API 呼叫所帶來的延遲和可用性風險。有關如何設定和使用擴充功能的資訊，請參閱 Amazon 使用 CloudWatch 者指南中的「 CloudWatch 明顯地執行啟動和 A/B 實驗」。

使用AWS AppConfig deployment events to Amazon EventBridge擴充功能

AWS AppConfig deployment events to Amazon EventBridge延伸功能是 AWS 編寫的擴充功能，可協助您監視 AWS AppConfig 組態部署工作流程並採取行動。每當部署配置時，擴展程序都會將事件通知發送到 EventBridge 默認事件總線。將擴充功能與其中一個 AWS AppConfig 應用程式、環境或組態設定檔產生關聯之後，請在每次設定部署開始、結束和復原之後，將事件通知 AWS AppConfig 傳送至事件匯流排。

如果您想進一步控制哪些動作點傳送 EventBridge 通知，可以建立自訂擴充功能，並在 URI 欄位中輸入 EventBridge 預設事件匯流排 Amazon 資源名稱 (ARN)。如需有關建立延伸功能的資訊，請參閱逐步解說：建立自訂 AWS AppConfig 延伸。

> **Important**
>
> 此擴充功能僅支援 EventBridge 預設事件匯流排。

使用擴充功能

若要使用AWS AppConfig deployment events to Amazon EventBridge擴充功能，請先建立擴充功能關聯，將擴充功能附加至其中一個 AWS AppConfig 資源。您可以使用 AWS AppConfig 主控台或 CreateExtensionAssociationAPI 動作來建立關聯。建立關聯時，您可以指定 AWS AppConfig 應用程式、環境或組態設定檔的 ARN。如果您將擴充功能與應用程式或環境產生關聯，則會針對指定應用程式或環境中包含的任何組態設定檔傳送事件通知。

建立關聯之後，當部署指定 AWS AppConfig 資源的組態時，會 AWS AppConfig 呼叫擴充功能，並根據擴充功能中指定的動作點傳送通知。

> **Note**
>
> 此擴充功能由下列動作點叫用：
>
> - ON_DEPLOYMENT_START
> - ON_DEPLOYMENT_COMPLETE
> - ON_DEPLOYMENT_ROLLED_BACK
>
> 您無法自訂此擴充功能的動作點。若要叫用不同的動作點，您可以建立自己的擴充功能。如需詳細資訊，請參閱 逐步解說：建立自訂 AWS AppConfig 延伸。

使用下列程序，透過使用 AWS Systems Manager 主控台或建立 AWS AppConfig 擴充功能關聯 AWS CLI。

若要建立擴充功能關聯 (主控台)

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 AWS AppConfig。
3. 在 [擴充功能] 索引標籤上，選擇 [新增至資源
4. 在 [擴充功能資源詳細資訊] 區段中，選擇資源類型做為 [AWS AppConfig 資源類型]。根據您選擇的資源， AWS AppConfig 提示您選擇其他資源。
5. 選擇建立與資源的關聯。

以下是呼叫擴充功能 EventBridge 時傳送至的範例事件。

```
{
   "version":"0",
   "id":"c53dbd72-c1a0-2302-9ed6-c076e9128277",
   "detail-type":"On Deployment Complete",
   "source":"aws.appconfig",
   "account":"111122223333",
   "time":"2022-07-09T01:44:15Z",
   "region":"us-east-1",
```

```
   "resources":[
      "arn:aws:appconfig:us-east-1:111122223333:extensionassociation/z763ff5"
   ],
   "detail":{
      "InvocationId":"5tfjcig",
       "Parameters":{

      },
      "Type":"OnDeploymentComplete",
      "Application":{
         "Id":"ba8toh7",
         "Name":"MyApp"
      },
      "Environment":{
         "Id":"pgil2o7",
         "Name":"MyEnv"
      },
      "ConfigurationProfile":{
         "Id":"ga3tqep",
         "Name":"MyConfigProfile"
      },
      "DeploymentNumber":1,
      "ConfigurationVersion":"1"
   }
}
```

使用AWS AppConfig deployment events to Amazon SNS擴充功能

AWS AppConfig deployment events to Amazon SNS延伸功能是 AWS 編寫的擴充功能，可協助您監視 AWS AppConfig 組態部署工作流程並採取行動。只要部署組態，擴充功能就會將訊息發佈到 Amazon SNS 主題。將擴充功能與其中一個 AWS AppConfig 應用程式、環境或組態設定檔產生關聯後，在每次組態部署開始、結束和復原之後，將訊息 AWS AppConfig 發佈至主題。

如果您想進一步控制哪些動作點傳送 Amazon SNS 通知，可以建立自訂擴充功能，並在 URI 欄位中輸入 Amazon SNS 主題 Amazon 資源名稱 (ARN)。如需有關建立延伸功能的資訊，請參閱逐步解說：建立自訂 AWS AppConfig 延伸。

使用擴充功能

本節說明如何使用AWS AppConfig deployment events to Amazon SNS擴充功能。

步驟 1：設定 AWS AppConfig 將訊息發佈至主題

將存取控制政策新增至 Amazon SNS 主題授與 AWS AppConfig (`appconfig.amazonaws.com`) 發佈許可 (`sns:Publish`)。如需詳細資訊，請參閱 Amazon SNS 存取控制的範例案例。

步驟 2：建立擴充功能關聯

建立擴充功能關聯，將擴充功能附加至其中一個 AWS AppConfig 資源。您可以使用 AWS AppConfig 主控台或 CreateExtensionAssociationAPI 動作來建立關聯。建立關聯時，您可以指定 AWS AppConfig 應用程式、環境或組態設定檔的 ARN。如果您將擴充功能與應用程式或環境產生關聯，則會針對指定應用程式或環境中包含的任何組態設定檔傳送通知。建立關聯時，必須為包含要使用之 Amazon SNS 主題的 ARN 的`topicArn`參數輸入值。

建立關聯之後，當部署指定 AWS AppConfig 資源的組態時，會 AWS AppConfig 呼叫擴充功能，並根據擴充功能中指定的動作點傳送通知。

> **Note**
>
> 此擴充功能由下列動作點叫用：
>
> - `ON_DEPLOYMENT_START`
> - `ON_DEPLOYMENT_COMPLETE`
> - `ON_DEPLOYMENT_ROLLED_BACK`
>
> 您無法自訂此擴充功能的動作點。若要叫用不同的動作點，您可以建立自己的擴充功能。如需詳細資訊，請參閱 逐步解說：建立自訂 AWS AppConfig 延伸。

使用下列程序，透過使用 AWS Systems Manager 主控台或建立 AWS AppConfig 擴充功能關聯 AWS CLI。

若要建立擴充功能關聯 (主控台)

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 AWS AppConfig。
3. 在 [擴充功能] 索引標籤上，選擇 [新增至資源
4. 在 [擴充功能資源詳細資訊] 區段中，選擇資源類型做為 [AWS AppConfig 資源類型]。根據您選擇的資源， AWS AppConfig 提示您選擇其他資源。

5. 選擇建立與資源的關聯。

以下是叫用擴充功能時傳送至 Amazon SNS 主題的訊息範例。

```
{
    "Type": "Notification",
    "MessageId": "ae9d702f-9a66-51b3-8586-2b17932a9f28",
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic",
    "Message": {
        "InvocationId": "7itcaxp",
        "Parameters": {
            "topicArn": "arn:aws:sns:us-east-1:111122223333:MySNSTopic"
        },
        "Application": {
            "Id": "1a2b3c4d",
            "Name": MyApp
        },
        "Environment": {
            "Id": "1a2b3c4d",
            "Name": MyEnv
        },
        "ConfigurationProfile": {
            "Id": "1a2b3c4d",
            "Name": "MyConfigProfile"
        },
        "Description": null,
        "DeploymentNumber": "3",
        "ConfigurationVersion": "1",
        "Type": "OnDeploymentComplete"
    },
    "Timestamp": "2022-06-30T20:26:52.067Z",
    "SignatureVersion": "1",
    "Signature": "<...>",
    "SigningCertURL": "<...>",
    "UnsubscribeURL": "<...>",
    "MessageAttributes": {
        "MessageType": {
            "Type": "String",
            "Value": "OnDeploymentStart"
        }
    }
}
```

使用AWS AppConfig deployment events to Amazon SQS擴充功能

AWS AppConfig deployment events to Amazon SQS延伸功能是 AWS 編寫的擴充功能，可協助您監視 AWS AppConfig 組態部署工作流程並採取行動。每當部署組態時，延伸模組都會將訊息排入 Amazon 簡單佇列服務 (Amazon SQS) 佇列中。將擴充功能與其中一個 AWS AppConfig 應用程式、環境或組態設定檔產生關聯之後，請在每次組態部署開始、結束和復原之後，將訊息排入 AWS AppConfig 佇列。

如果您想進一步控制哪些動作點傳送 Amazon SQS 通知，可以建立自訂擴充功能，並在 URI 欄位中輸入 Amazon SQS 佇列 Amazon 資源名稱 (ARN)。如需有關建立延伸功能的資訊，請參閱逐步解說：建立自訂 AWS AppConfig 延伸。

使用擴充功能

本節說明如何使用AWS AppConfig deployment events to Amazon SQS擴充功能。

步驟 1：設定 AWS AppConfig 將訊息排入佇列

將 Amazon SQS 政策新增至您的 Amazon SQS 佇列授予 AWS AppConfig (appconfig.amazonaws.com) 傳送訊息許可 (sqs:SendMessage)。如需詳細資訊，請參閱 Amazon SQS 政策的基本範例。

步驟 2：建立擴充功能關聯

建立擴充功能關聯，將擴充功能附加至其中一個 AWS AppConfig 資源。您可以使用 AWS AppConfig 主控台或 CreateExtensionAssociationAPI 動作來建立關聯。建立關聯時，您可以指定 AWS AppConfig 應用程式、環境或組態設定檔的 ARN。如果您將擴充功能與應用程式或環境產生關聯，則會針對指定應用程式或環境中包含的任何組態設定檔傳送通知。建立關聯時，您必須輸入包含要使用之 Amazon SQS 佇列之 ARN 的Here參數。

建立關聯之後，當建立或部署指定 AWS AppConfig 資源的組態時，會 AWS AppConfig 呼叫擴充功能，並根據擴充功能中指定的動作點傳送通知。

> **Note**
>
> 此擴充功能由下列動作點叫用：
>
> - ON_DEPLOYMENT_START
> - ON_DEPLOYMENT_COMPLETE
> - ON_DEPLOYMENT_ROLLED_BACK

您無法自訂此擴充功能的動作點。若要叫用不同的動作點，您可以建立自己的擴充功能。如需詳細資訊，請參閱 逐步解說：建立自訂 AWS AppConfig 延伸。

使用下列程序，透過使用 AWS Systems Manager 主控台或建立 AWS AppConfig 擴充功能關聯 AWS CLI。

若要建立擴充功能關聯 (主控台)

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 AWS AppConfig。
3. 在 [擴充功能] 索引標籤上，選擇 [新增至資源
4. 在 [擴充功能資源詳細資訊] 區段中，選擇資源類型做為 [AWS AppConfig 資源類型]。根據您選擇的資源， AWS AppConfig 提示您選擇其他資源。
5. 選擇建立與資源的關聯。

以下是叫用擴充功能時傳送至 Amazon SQS 佇列的訊息範例。

```
{
   "InvocationId":"7itcaxp",
   "Parameters":{
      "queueArn":"arn:aws:sqs:us-east-1:111122223333:MySQSQueue"
   },
   "Application":{
      "Id":"1a2b3c4d",
      "Name":MyApp
   },
   "Environment":{
      "Id":"1a2b3c4d",
      "Name":MyEnv
   },
   "ConfigurationProfile":{
      "Id":"1a2b3c4d",
      "Name":"MyConfigProfile"
   },
   "Description":null,
   "DeploymentNumber":"3",
```

```
    "ConfigurationVersion":"1",
    "Type":"OnDeploymentComplete"
}
```

與阿特拉西亞吉拉擴展工作 AWS AppConfig

通過與 Atlassian Jira 集成，每當您更改指定的功能標誌時，都 AWS AppConfig 可以在 Atlassian 控制台中創建和更新問題。 AWS 帳戶 AWS 區域每個 Jira 問題都包含旗標名稱、應用程式 ID、組態設定檔 ID 和旗標值。在您更新、儲存和部署旗標變更之後，Jira 會使用變更的詳細資訊來更新現有問題。

> **Note**
>
> 每當您建立或更新功能旗標時，Jira 都會更新問題。當您從父層級旗標刪除子層級旗標屬性時，Jira 也會更新問題。當您刪除父層級旗標時，Jira 不會記錄資訊。

若要設定整合，您必須執行下列動作：

- 設定 AWS AppConfig Jira 整合的權限
- 設定 AWS AppConfig Jira 整合應用程式

設定 AWS AppConfig Jira 整合的權限

當您設定與 Jira 的 AWS AppConfig 整合時，您可以指定使用者的認證。具體而言，您可以在 for Jira 應用程式中輸入使用者的存取AWS AppConfig 金鑰 ID 和秘密金鑰。此使用者授予 Jira 與之通訊的 AWS AppConfig權限。 AWS AppConfig 使用這些憑證一次，在 AWS AppConfig 和 Jira 之間建立關聯。不會儲存認證。您可以通過卸載 Jira 應用程序 AWS AppConfig 來刪除關聯。

使用者帳號需要包含下列動作的權限原則：

- `appconfig:CreateExtensionAssociation`
- `appconfig:GetConfigurationProfile`
- `appconfig:ListApplications`
- `appconfig:ListConfigurationProfiles`
- `appconfig:ListExtensionAssociations`
- `sts:GetCallerIdentity`

完成以下任務以建立 IAM 許可政策和用於 AWS AppConfig 和 Jira 整合的使用者：

工作

- 工作 1：建立 AWS AppConfig 和 Jira 整合的 IAM 權限政策
- 工作 2：建立 AWS AppConfig 和 Jira 整合的使用者

工作 1：建立 AWS AppConfig 和 Jira 整合的 IAM 權限政策

使用下列程序建立 IAM 權限政策，以允許 Atlassian Jira 進行通訊。 AWS AppConfig建議您建立新政策，並將此政策附加到新的 IAM 角色。將必要的權限新增至現有 IAM 政策和角色會違反最低權限原則，因此不建議使用。

若要建立 AWS AppConfig 和 Jira 整合的 IAM 政策

1. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
3. 在 [建立原則] 頁面上，選擇 [JSON] 索引標籤，並以下列原則取代預設內容。在下列原則中，使用功能旗*##################*識別碼」、「*#### ID*」和「*####* ID」。 AWS AppConfig

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                      "appconfig:CreateExtensionAssociation",
                      "appconfig:ListExtensionAssociations",
                      "appconfig:GetConfigurationProfile"
             ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID",

 "arn:aws:appconfig:Region:account_ID:application/application_ID/
configurationprofile/configuration_profile_ID"
             ]
        },
       {
            "Effect": "Allow",
            "Action": [
```

```
                    "appconfig:ListApplications"

            ],
            "Resource": [
                    "arn:aws:appconfig:Region:account_ID:*"
            ]
        },
         {
            "Effect": "Allow",
            "Action": [
                    "appconfig:ListConfigurationProfiles"
            ],
            "Resource": [

 "arn:aws:appconfig:Region:account_ID:application/application_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
            "Resource": "*"
        }
    ]
}
```

4. 選擇下一步：標籤。
5. (選用) 新增一個或多個標籤鍵值組來組織、追蹤或控制存取此政策，然後選擇 Next: Review (下一步：檢閱)。
6. 在 Review policy (檢閱政策) 頁面，在 Name (名稱) 方塊中輸入名稱 (如 **AppConfigJiraPolicy**)，接著輸入選用描述。
7. 選擇建立政策。

工作 2：建立 AWS AppConfig 和 Jira 整合的使用者

請使用下列程序來建立 Atlassian Jira 整合 AWS AppConfig 的使用者。建立使用者之後，您可以複製存取金鑰 ID 和秘密金鑰，這些 ID 和秘密金鑰會在您完成整合時指定。

若要為 AWS AppConfig 和 Jira 整合建立使用者

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Users (使用者)，然後選擇 Add users (新增使用者)。

3. 在「使用者名稱」欄位中，輸入名稱，例如**AppConfigJiraUser**。
4. 針對 [選取 AWS 認證類型]，選擇 [存取鍵]-[程式設計存取]
5. 選擇下一步：許可。
6. 在 [設定權限] 頁面下，選擇 [直接附加現有原則] 搜尋並選取您在其中建立之原則的核取方塊工作 1：建立 AWS AppConfig 和 Jira 整合的 IAM 權限政策，然後選擇 [下一步:標記]。
7. 在「新增標籤 (選用)」頁面上，新增一或多個標籤金鑰值組，以組織、追蹤或控制此使用者的存取權。選擇下一步：檢閱。
8. 在「檢閱」頁面上，確認使用者詳細資訊。
9. 選擇 Create user (建立使用者)。系統會顯示使用者的存取金鑰 ID 和秘密金鑰。您可以下載 .csv 檔案，或將這些認證複製到不同的位置。您將在設定整合時指定這些認證。

設定 AWS AppConfig Jira 整合應用程式

使用下列程序來設定 Jira 應用程式中 AWS AppConfig 的必要選項。完成此程序後，Jira 會針對指定 AWS 區域的中的每個功能旗標建立新問題。 AWS 帳戶 如果您對中的功能旗標進行變更 AWS AppConfig，Jira 會記錄現有問題中的詳細資訊。

> **Note**
>
> AWS AppConfig 功能旗標可以包含多個子層級旗標屬性。Jira 會為每個父層級特徵旗標建立一個問題。如果您變更子層級旗標屬性，您可以在父層級旗標的 Jira 問題中檢視該變更的詳細資訊。

若要設定整合

1. 登錄到阿特拉西亞 Marketplace。
2. **AWS AppConfig**在搜尋欄位中輸入，然後按 Enter 鍵。
3. 在 Jira 實例上安裝應用程序。
4. 在 Atlassian 控制台中，選擇管理應用程序，然後選擇AWS AppConfig Jira。
5. 選擇設定。
6. 在 [組態詳細資料] 下，選擇 [Jira 專案]，然後選擇要與 AWS AppConfig 功能旗標相關聯的專案。
7. 選擇 AWS 區域，然後選擇 AWS AppConfig 功能旗標所在的區域。
8. 在「應用程式 ID」欄位中，輸入包含功能旗標的 AWS AppConfig 應用程式名稱。

9. 在「組態設定檔 ID」欄位中，輸入功能旗標的 AWS AppConfig 組態設定檔名稱。
10. 在「存取金鑰 ID」和「秘密金鑰」欄位中，輸入您複製的認證工作 2：建立 AWS AppConfig 和 Jira 整合的使用者。或者，您也可以指定階段作業權杖。
11. 選擇提交。
12. 在 Atlassian 主控台中，選擇 [專案]，然後選擇您選取要進行整合的專案。 AWS AppConfig 「問題」頁面會針對指定的 AWS 帳戶 和中的每個功能旗標顯示問題 AWS 區域。

刪除 Jira 應用程式和 AWS AppConfig 資料

如果您不想再將 Jira 整合與 AWS AppConfig 功能旗標搭配使用，您可以在 Atlassian 主控台中刪除 AWS AppConfig for Jira 應用程式。刪除整合應用程式會執行下列作業：

- 刪除您的 Jira 實例和 AWS AppConfig
- 從中刪除您的 Jira 實例詳細信息 AWS AppConfig

若要刪除 Jira AWS AppConfig 應用程式的

1. 在 Atlassian 主控台中，選擇 [管理應用程式]。
2. 選擇吉拉AWS AppConfig 。
3. 選擇解除安裝。

逐步解說：建立自訂 AWS AppConfig 延伸

若要建立自訂 AWS AppConfig 擴充功能，請完成下列工作。每項工作都會在後面的主題中更詳細地說明。

> **Note**
>
> 您可以在以下位置查看自定義 AWS AppConfig 擴展的示例 GitHub：
>
> - 範例擴充功能，可防止使用 Systems Manager `blocked day` 變更行事曆暫停行事曆進行部署
> - 使用 git-secrets 防止密碼洩漏到配置數據中的示例擴展
> - 使用 Amazon Comprehend 防止個人識別資訊 (PII) 洩漏到組態資料中的範例擴充功能

1. 創建一個 AWS Lambda 函數

   對於大多數使用案例，若要建立自訂擴充功能，您必須建立 AWS Lambda 函數來執行擴充功能中定義的任何計算和處理。此規則的例外情況是，如果您建立AWS 已編寫的通知延伸模組的自訂版本以新增或移除動作點。如需此例外狀況的詳細資訊，請參閱建立自訂 AWS AppConfig 擴充功能。

2. 設定自訂擴充功能的權限

   若要設定自訂擴充功能的權限，您可以執行下列其中一項操作：

   - 建立包含`InvokeFunction`許可的 AWS Identity and Access Management (IAM) 服務角色。
   - 使用 Lambda AddPermissionAPI 動作建立資源政策。

   本逐步解說說明如何建立 IAM 服務角色。

3. 建立擴充功能

   您可以使用主 AWS AppConfig 控台或從 AWS CLI、 AWS Tools for PowerShell或 SDK 呼叫 CreateExtensionAPI 動作來建立擴充功能。逐步解說會使用主控台。

4. 建立擴充功能關聯

   您可以使用 AWS AppConfig 主控台或從 AWS CLI、 AWS Tools for PowerShell或 SDK 呼叫 CreateExtensionAssociationAPI 動作來建立擴充功能關聯。逐步解說會使用主控台。

5. 執行呼叫擴充功能的動作

   建立關聯之後，當擴充功能定義的動作點發生該資源時，會 AWS AppConfig 叫用擴充功能。例如，如果您關聯包含`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`動作的擴充功能，則每次您建立新的託管組態版本時，都會叫用該擴充功能。

本節中的主題說明建立自訂 AWS AppConfig 擴充功能時所涉及的每項工作。在使用案例中描述了每個任務，其中客戶想要建立可自動將組態備份到 Amazon Simple Storage Service (Amazon S3) 貯體的擴充功能。只要建立託管組態 (`PRE_CREATE_HOSTED_CONFIGURATION_VERSION`) 或已部署 (`PRE_START_DEPLOYMENT`)，擴充功能就會執行。

主題

- 為自訂 AWS AppConfig 擴充功能建立 Lambda 函數
- 設定自訂 AWS AppConfig 擴充功能的權限
- 建立自訂 AWS AppConfig 擴充功能
- 為自訂擴充功能建立擴 AWS AppConfig 充功能關聯

- 執行呼叫自訂 AWS AppConfig 擴充功能的動作

為自訂 AWS AppConfig 擴充功能建立 Lambda 函數

對於大多數用例，要創建自定義擴展，必須創建一個 AWS Lambda 函數來執行擴展中定義的任何計算和處理。本節包含自訂擴充功能的 Lambda 函數範例程式 AWS AppConfig 碼。本節還包括有效負載請求和響應參考詳細信息。如需建立 Lambda 函數的相關資訊，請參閱開AWS Lambda 發人員指南中的開始使用 Lambda。

範本程式碼

下列 Lambda 函數的範例程式碼在叫用時，會自動將組 AWS AppConfig 態備份到 Amazon S3 儲存貯體。每當建立或部署新組態時，都會備份組態。該示例使用擴展參數，因此存儲桶名稱不必在 Lambda 函數中進行硬編碼。通過使用擴展參數，用戶可以將擴展程序附加到多個應用程序，並將配置備份到不同的存儲桶。程式碼範例包含可進一步說明函式的註解。

AWS AppConfig 擴充功能的範例 Lambda 函數

```
from datetime import datetime
import base64
import json

import boto3


def lambda_handler(event, context):
    print(event)

    # Extensions that use the PRE_CREATE_HOSTED_CONFIGURATION_VERSION and
 PRE_START_DEPLOYMENT
    # action points receive the contents of AWS AppConfig configurations in Lambda
 event parameters.
    # Configuration contents are received as a base64-encoded string, which the lambda
 needs to decode
    # in order to get the configuration data as bytes. For other action points, the
 content
    # of the configuration isn't present, so the code below will fail.
    config_data_bytes = base64.b64decode(event["Content"])

    # You can specify parameters for extensions. The CreateExtension API action lets
 you define
```

```
    # which parameters an extension supports. You supply the values for those
 parameters when you
    # create an extension association by calling the CreateExtensionAssociation API
 action.
    # The following code uses a parameter called S3_BUCKET to obtain the value
 specified in the
    # extension association. You can specify this parameter when you create the
 extension
    # later in this walkthrough.
    extension_association_params = event.get('Parameters', {})
    bucket_name = extension_association_params['S3_BUCKET']
    write_backup_to_s3(bucket_name, config_data_bytes)

    # The PRE_CREATE_HOSTED_CONFIGURATION_VERSION and PRE_START_DEPLOYMENT action
 points can
    # modify the contents of a configuration. The following code makes a minor change
    # for the purposes of a demonstration.
    old_config_data_string = config_data_bytes.decode('utf-8')
    new_config_data_string = old_config_data_string.replace('hello', 'hello!')
    new_config_data_bytes = new_config_data_string.encode('utf-8')

    # The lambda initially received the configuration data as a base64-encoded string
    # and must return it in the same format.
    new_config_data_base64string =
 base64.b64encode(new_config_data_bytes).decode('ascii')

    return {
        'statusCode': 200,
        # If you want to modify the contents of the configuration, you must include the
 new contents in the
        # Lambda response. If you don't want to modify the contents, you can omit the
 'Content' field shown here.
        'Content': new_config_data_base64string
    }


def write_backup_to_s3(bucket_name, config_data_bytes):
    s3 = boto3.resource('s3')
    new_object = s3.Object(bucket_name,
 f"config_backup_{datetime.now().isoformat()}.txt")
    new_object.put(Body=config_data_bytes)
```

如果您想要在本逐步解說中使用此範例，請使用名稱儲存該範例，**MyS3ConfigurationBackUpExtension**然後複製該函數的 Amazon 資源名稱 (ARN)。您可以在下一節中建立 AWS Identity and Access Management (IAM) 假設角色時指定 ARN。您可以在建立副檔名時指定 ARN 和名稱。

有效載荷參

本節包含使用自訂 AWS AppConfig 延伸模組的承載要求和回應參考詳細資料。

請求結構

PreCreateHostedConfigurationVersion

```
{
    'InvocationId': 'vlns753', // id for specific invocation
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh', // Base64 encoded content
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'Description': '',
    'Type': 'PreCreateHostedConfigurationVersion',
    'PreviousContent': {
        'ContentType': 'text/plain',
        'ContentVersion': '1',
        'Content': 'SGVsbG8gd29ybGQh'
    }
}
```

PreStartDeployment

```
{
    'InvocationId': '765ahdm',
```

```
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'ContentType': 'text/plain',
    'ContentVersion': '2',
    'Content': 'SGVsbG8gZWFydGgh',
    'Application': {
        'Id': 'abcd123',
        'Name': 'ApplicationName'
    },
    'Environment': {
        'Id': 'ibpnqlq',
        'Name': 'EnvironmentName'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'Type': 'PreStartDeployment'
}
```

非同步事件

OnStartDeployment, OnDeploymentStep, OnDeployment

```
{
    'InvocationId': 'o2xbtm7',
    'Parameters': {
        'ParameterOne': 'ValueOne',
        'ParameterTwo': 'ValueTwo'
    },
    'Type': 'OnDeploymentStart',
    'Application': {
        'Id': 'abcd123'
    },
    'Environment': {
        'Id': 'efgh456'
    },
    'ConfigurationProfile': {
        'Id': 'ijkl789',
```

```
        'Name': 'ConfigurationName'
    },
    'DeploymentNumber': 2,
    'Description': 'Deployment description',
    'ConfigurationVersion': '2'
}
```

響應結構

下列範例顯示您的 Lambda 功能回應自訂 AWS AppConfig 擴充功能的要求所傳回的內容。

同步事件-成功回應

如果要轉換內容，請使用以下內容：

```
"Content": "SomeBase64EncodedByteArray"
```

如果您不想轉換內容，則不返回任何內容。

異步事件-成功響應

什麼都不返回。

所有錯誤事件

```
{
        "Error": "BadRequestError",
        "Message": "There was malformed stuff in here",
        "Details": [{
            "Type": "Malformed",
            "Name": "S3 pointer",
            "Reason": "S3 bucket did not exist"
        }]
    }
```

設定自訂 AWS AppConfig 擴充功能的權限

使用下列程序來建立和設定 AWS Identity and Access Management (IAM) 服務角色 (或擔任角色)。AWS AppConfig 使用此角色來叫用 Lambda 函數。

建立 IAM 服務角色並 AWS AppConfig 允許假設

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。

2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 [選取信任實體的類型] 下，選擇 [自訂信任原則]。
4. 將下列 JSON 原則貼到 [自訂信任原則] 欄位中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

選擇下一步。

5. 在 [新增權限] 頁面上，選擇 [建立原則]。Create policy (建立政策) 頁面隨即在新標籤中開啟。
6. 選擇 [JSON] 索引標籤，然後將下列權限原則貼到編輯器中。動`lambda:InvokeFunction`作用於`PRE_*`動作點。動`lambda:InvokeAsync`作用於`ON_*`動作點。將*`## Lambda ARN`* 取代為您的 Amazon 資源名稱 (ARN)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:InvokeAsync"
      ],
      "Resource": "Your Lambda ARN"
    }
  ]
}
```

7. 選擇下一步：標籤。
8. 在 [新增標記 (選擇性)] 頁面上，新增一或多個機碼值配對，然後選擇 [下一步：複查]。

9. 在 [檢閱策略] 頁面上輸入名稱和說明，然後選擇 [建立策略]。
10. 在自訂信任原則的瀏覽器索引標籤上，選擇 [重新整理] 圖示，然後搜尋您剛建立的權限原則。
11. 選取權限原則的核取方塊，然後選擇 [下一步]。
12. 在 [名稱、檢閱和建立] 頁面上的 [角色名稱] 方塊中輸入名稱，然後輸入說明。
13. 選擇 Create role (建立角色)。系統會讓您回到 Roles (角色) 頁面。在橫幅中選擇 [檢視角色]。
14. 複製 ARN。您可以在建立擴充功能時指定此 ARN。

建立自訂 AWS AppConfig 擴充功能

擴充功能會定義它在工作流程期間執行的一或多個動 AWS AppConfig 作。例如， AWS 編寫的AWS AppConfig deployment events to Amazon SNS擴充功能包含一個動作，可將通知傳送至 Amazon SNS 主題。當您與之互動 AWS AppConfig 或代表您執行處理程序 AWS AppConfig 時，都會叫用每個動作。這些被稱為行動點。 AWS AppConfig 擴充功能支援下列動作要點：

- PRE_CREATE_HOSTED_CONFIGURATION_VERSION
- PRE_START_DEPLOYMENT
- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_STEP
- ON_DEPLOYMENT_BAKING
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

在動作點上設定的延伸PRE_*動作會在請求驗證後套用，但在 AWS AppConfig 執行與動作點名稱相對應的活動之前。這些動作呼叫會與要求同時處理。如果發出一個以上的請求，動作調用按順序運行。另外請注意，PRE_*動作點會接收並且可以變更組態的內容。 PRE_*動作點也可以回應錯誤並防止動作發生。

擴充功能也可以使用ON_*動作點與 AWS AppConfig 工作流程 parallel 執行。 ON_*動作點是異步調用的。 ON_*動作點不會收到配置的內容。如果延伸功能在ON_*動作點期間遇到錯誤，服務會忽略錯誤並繼續工作流程。

下列範例擴充功能會定義一個呼叫PRE_CREATE_HOSTED_CONFIGURATION_VERSION動作點的動作。在Uri欄位中，動作會指定本逐步解說稍早建立之 MyS3ConfigurationBackUpExtension

Lambda 函數的 Amazon 資源名稱 (ARN)。此動作也會指定 AWS Identity and Access Management (IAM) 假設在本逐步解說稍早建立的角色 ARN。

範例 AWS AppConfig 擴充

```
{
    "Name": "MySampleExtension",
    "Description": "A sample extension that backs up configurations to an S3 bucket.",
    "Actions": {
        "PRE_CREATE_HOSTED_CONFIGURATION_VERSION": [
            {
                "Name": "PreCreateHostedConfigVersionActionForS3Backup",
                "Uri": "arn:aws:lambda:aws-
region:111122223333:function:MyS3ConfigurationBackUpExtension",
                "RoleArn": "arn:aws:iam::111122223333:role/ExtensionsTestRole"
            }
        ]
    },
    "Parameters" : {
        "S3_BUCKET": {
            "Required": false
        }
    }
}
```

> **Note**
>
> 若要在建立擴充功能時檢視要求語法和欄位說明，請參閱 AWS AppConfig API 參考中的CreateExtension主題。

若要建立擴充功能 (主控台)

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 AWS AppConfig。
3. 在 [擴充功能] 索引標籤上選擇 [建立擴充
4. 在擴充功能名稱中，輸入唯一的名稱。對於本穿越的目的，請輸入**MyS3ConfigurationBackUpExtension**。您可以選擇性地輸入說明。
5. 在「動作」區段中，選擇「新增動作」。

6. 在「動作名稱」 中，輸入唯一的名稱。對於本穿越的目的，請輸入**PreCreateHostedConfigVersionActionForS3Backup**。此名稱描述動作和延伸用途所使用的動作點。
7. 在「動作點」清單中，選擇「預先建立 _ 主機 _ 組態 _ 版本」。
8. 對於 Uri，請選擇 Lambda 函數，然後在 Lambda 函數清單中選擇函數。如果您沒有看到您的函數，請確認您與建立函數的 AWS 區域 位置相同。
9. 對於 IAM 角色，請選擇您先前在本逐步解說中建立的角色。
10. 在「擴充功能參數 (選用)」區段中，選擇「新增參數」。
11. 在「參數名稱」 中，輸入名稱。對於本穿越的目的，請輸入**S3_BUCKET**。
12. 重複步驟 5-11，為動作點建立第二個PRE_START_DEPLOYMENT動作。
13. 選擇建立擴充功能。

自訂 AWS 編寫的通知延伸模組

您不需要建立 Lambda 或擴充功能即可使用已AWS 編寫的通知延伸模組。您可以簡單地創建擴展關聯，然後執行調用其中一個支持操作點的操作。根據預設， AWS 編寫的通知延伸模組支援下列動作點：

- ON_DEPLOYMENT_START
- ON_DEPLOYMENT_COMPLETE
- ON_DEPLOYMENT_ROLLED_BACK

如果您建立擴充功能和AWS AppConfig deployment events to Amazon SNSAWS AppConfig deployment events to Amazon SQS擴充功能的自訂版本，您可以指定要接收通知的動作點。

> **Note**
>
> AWS AppConfig deployment events to EventBridge擴充功能不支援行PRE_*動點。如果您要移除指派給已 AWS 編寫版本的某些預設動作點，您可以建立自訂版本。

如果您建立 AWS 已編寫的通知延伸模組的自訂版本，則不需要建立 Lambda 函數。您只需要在新擴充功能版本的Uri欄位中指定 Amazon 資源名稱 (ARN) 即可。

- 對於自訂 EventBridge 通知延伸模組，請在Uri欄位中輸入 EventBridge 預設事件的 ARN。

- 如需自訂 Amazon SNS 通知延伸功能，請在Uri欄位中輸入 Amazon SNS 主題的 ARN。
- 如需自訂 Amazon SQS 通知延伸模組，請在欄位中輸入 Amazon SQS 訊息佇列的 ARN。Uri

為自訂擴充功能建立擴 AWS AppConfig 充功能關聯

若要建立擴充功能或設定 AWS 已編寫的擴充功能，您可以定義在使用特定 AWS AppConfig 資源時叫用擴充功能的動作點。例如，您可以在針對特定應用程式啟動組態部署時，選擇執行AWS AppConfig deployment events to Amazon SNS擴充功能並接收 Amazon SNS 主題的通知。定義哪些動作點叫用特定 AWS AppConfig 資源的擴充功能稱為擴充功能關聯。擴充功能關聯是擴充功能與 AWS AppConfig 資源 (例如應用程式或組態設定檔) 之間的指定關係。

單一 AWS AppConfig 應用程式可以包含多個環境和組態設定檔。如果您將擴充功能與應用程式或環境產生關聯，則會針對與應用程式或環境資源相關的任何工作流程 (如果適用) AWS AppConfig 叫用擴充程式。

例如，假設您有一個名為的 AWS AppConfig 應用程式 MobileApps ，其中包含名為的設定描述檔 AccessList。並假設該 MobileApps 應用程序包括 Beta，集成和生產環境。您可以為 AWS 編寫的 Amazon SNS 通知延伸模組建立擴充功能關聯，並將擴充功能與 MobileApps 應用程式相關聯。只要將應用程式的組態部署到三個環境中的任何一個，就會叫用 Amazon SNS 通知延伸模組。

使用下列程序，使用 AWS AppConfig 主控台建立 AWS AppConfig 擴充功能關聯。

若要建立擴充功能關聯 (主控台)

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/appconfig/
2. 在導覽窗格中，選擇 AWS AppConfig。
3. 在 [擴充功能] 索引標籤上，選擇擴充功能的選項按鈕，然後選擇 [新增至資源]。針對本逐步解說的目的，請選擇 MyS3 ConfigurationBackUpExtension。
4. 在 [擴充功能資源詳細資訊] 區段中，選擇資源類型做為 [AWS AppConfig 資源類型]。根據您選擇的資源， AWS AppConfig 提示您選擇其他資源。針對本逐步解說的目的，請選擇「應用程式」。
5. 在清單中選擇一個應用程式。
6. 在「參數」區段中，確認「機碼」欄位中列出了 S3_BUCKET。在「值」欄位中，貼上 Lambda 擴充功能的 ARN。例如：arn:aws:lambda:*aws-region*:111122223333:function:MyS3ConfigurationBackUpExtension。
7. 選擇建立與資源的關聯。

執行呼叫自訂 AWS AppConfig 擴充功能的動作

建立關聯之後，您可以建立hosted為其指定的新組態設定檔來叫用MyS3ConfigurationBackUpExtension擴充功能SourceUri。作為建立新組態的工作流程的一部分， AWS AppConfig 會遇到PRE_CREATE_HOSTED_CONFIGURATION_VERSION動作點。遇到此動作點會叫用擴充功能，該MyS3ConfigurationBackUpExtension擴充功能會自動將新建立的組態備份到延伸功能關聯Parameter區段中指定的 S3 儲存貯體。

AWS AppConfig 擴展集成與阿特拉西亞吉拉

AWS AppConfig 與阿特拉西亞吉拉集成. 每當您 AWS 帳戶 針 AWS AppConfig 對指定的功能標誌進行變更時，整合都可以在 Atlassian 主控台中建立和更新問題。 AWS 區域每個 Jira 問題都包含旗標名稱、應用程式 ID、組態設定檔 ID 和旗標值。在您更新、儲存和部署旗標變更之後，Jira 會使用變更的詳細資訊來更新現有問題。如需更多詳細資訊，請參閱 與阿特拉西亞吉拉擴展工作 AWS AppConfig。

AWS AppConfig 程式碼範例

本節包含程式設計方式執行常見AWS AppConfig動作的程式碼範例。我們建議您將這些範例與 Java、Python 和 JavaScriptSDK 搭配使用，以便在測試環境中執行動作。本節包括一個代碼示例，用於在完成後清理測試環境。

主題

建立或更新儲存在主控組態存放區中的自由格式組態

下列每個範例都包含有關程式碼所執行動作的註解。本節中的範例會呼叫下列 API：

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion

Java

```
public CreateHostedConfigurationVersionResponse createHostedConfigVersion() {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
```

```
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));

        return hcv;
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a hosted, freeform configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
    Content=b'my config data',
    ContentType='text/plain')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
```

```
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a hosted, freeform configuration profile
const profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: profile.Id,
    ContentType: "text/plain",
    Content: "my config data",
  })
);
```

為秘密管理員中儲存的密碼建立組態設定描述檔

下列每個範例都包含有關程式碼所執行動作的註解。本節中的範例會呼叫下列 API：

- CreateApplication
- CreateConfigurationProfile

Java

```
private void createSecretsManagerConfigProfile() {
```

```
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a configuration profile for Secrets Manager Secret
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("secretsmanager://MySecret")
            .retrievalRoleArn("arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret")
            .type("AWS.Freeform"));
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create a configuration profile for Secrets Manager Secret
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='secretsmanager://MySecret',
    RetrievalRoleArn='arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret',
    Type='AWS.Freeform')
```

JavaScript

```
import {
  AppConfigClient,
  CreateConfigurationProfileCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();
```

```
// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create a configuration profile for Secrets Manager Secret
await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "secretsmanager://MySecret",
    RetrievalRoleArn: "arn:aws:iam::000000000000:role/
RoleTrustedByAppConfigThatCanRetrieveSecret",
    Type: "AWS.Freeform",
  })
);
```

部署設定描述檔

下列每個範例都包含有關程式碼所執行動作的註解。本節中的範例會呼叫下列 API：

- CreateApplication
- CreateConfigurationProfile
- CreateHostedConfigurationVersion
- CreateEnvironment
- StartDeployment
- GetDeployment

Java

```
private void createDeployment() throws InterruptedException {
        AppConfigClient appconfig = AppConfigClient.create();

        // Create an application
        CreateApplicationResponse app = appconfig.createApplication(req ->
 req.name("MyDemoApp"));

        // Create a hosted, freeform configuration profile
```

```
        CreateConfigurationProfileResponse configProfile =
 appconfig.createConfigurationProfile(req -> req
            .applicationId(app.id())
            .name("MyConfigProfile")
            .locationUri("hosted")
            .type("AWS.Freeform"));

        // Create a hosted configuration version
        CreateHostedConfigurationVersionResponse hcv =
 appconfig.createHostedConfigurationVersion(req -> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .contentType("text/plain; charset=utf-8")
            .content(SdkBytes.fromUtf8String("my config data")));


        // Create an environment
        CreateEnvironmentResponse env = appconfig.createEnvironment(req -> req
            .applicationId(app.id())
            .name("Beta")
            // If you have CloudWatch alarms that monitor the health of your
 service, you can add them here and they
            // will trigger a rollback if they fire during an appconfig deployment
            //.monitors(Monitor.builder().alarmArn("arn:aws:cloudwatch:us-
east-1:520900602629:alarm:MyAlarm")
            //
  .alarmRoleArn("arn:aws:iam::520900602629:role/MyAppConfigAlarmRole").build())
        );

        // Start a deployment
        StartDeploymentResponse deploymentResponse = appconfig.startDeployment(req -
> req
            .applicationId(app.id())
            .configurationProfileId(configProfile.id())
            .environmentId(env.id())
            .configurationVersion(hcv.versionNumber().toString())
            .deploymentStrategyId("AppConfig.Linear50PercentEvery30Seconds")
        );

        // Wait for deployment to complete
        List<DeploymentState> nonFinalDeploymentStates = Arrays.asList(
            DeploymentState.DEPLOYING,
            DeploymentState.BAKING,
            DeploymentState.ROLLING_BACK,
```

```
            DeploymentState.VALIDATING);
        GetDeploymentRequest getDeploymentRequest =
 GetDeploymentRequest.builder().applicationId(app.id())

 .environmentId(env.id())

 .deploymentNumber(deploymentResponse.deploymentNumber()).build();
        GetDeploymentResponse deployment =
 appconfig.getDeployment(getDeploymentRequest);
        while (nonFinalDeploymentStates.contains(deployment.state())) {
            System.out.println("Waiting for deployment to complete: " + deployment);
            Thread.sleep(1000L);
            deployment = appconfig.getDeployment(getDeploymentRequest);
        }

        System.out.println("Deployment complete: " + deployment);
    }
```

Python

```
import boto3

appconfig = boto3.client('appconfig')

# create an application
application = appconfig.create_application(Name='MyDemoApp')

# create an environment
environment = appconfig.create_environment(
    ApplicationId=application['Id'],
    Name='MyEnvironment')

# create a configuration profile
config_profile = appconfig.create_configuration_profile(
    ApplicationId=application['Id'],
    Name='MyConfigProfile',
    LocationUri='hosted',
    Type='AWS.Freeform')

# create a hosted configuration version
hcv = appconfig.create_hosted_configuration_version(
    ApplicationId=application['Id'],
    ConfigurationProfileId=config_profile['Id'],
```

```
    Content=b'my config data',
    ContentType='text/plain')

# start a deployment
deployment = appconfig.start_deployment(
    ApplicationId=application['Id'],
    EnvironmentId=environment['Id'],
    ConfigurationProfileId=config_profile['Id'],
    ConfigurationVersion=str(hcv['VersionNumber']),
    DeploymentStrategyId='AppConfig.Linear20PercentEvery6Minutes')
```

JavaScript

```
import {
  AppConfigClient,
  CreateApplicationCommand,
  CreateEnvironmentCommand,
  CreateConfigurationProfileCommand,
  CreateHostedConfigurationVersionCommand,
  StartDeploymentCommand,
} from "@aws-sdk/client-appconfig";

const appconfig = new AppConfigClient();

// create an application
const application = await appconfig.send(
  new CreateApplicationCommand({ Name: "MyDemoApp" })
);

// create an environment
const environment = await appconfig.send(
  new CreateEnvironmentCommand({
    ApplicationId: application.Id,
    Name: "MyEnvironment",
  })
);

// create a configuration profile
const config_profile = await appconfig.send(
  new CreateConfigurationProfileCommand({
    ApplicationId: application.Id,
    Name: "MyConfigProfile",
    LocationUri: "hosted",
```

```
    Type: "AWS.Freeform",
  })
);

// create a hosted configuration version
const hcv = await appconfig.send(
  new CreateHostedConfigurationVersionCommand({
    ApplicationId: application.Id,
    ConfigurationProfileId: config_profile.Id,
    Content: "my config data",
    ContentType: "text/plain",
  })
);

// start a deployment
await appconfig.send(
  new StartDeploymentCommand({
    ApplicationId: application.Id,
    EnvironmentId: environment.Id,
    ConfigurationProfileId: config_profile.Id,
    ConfigurationVersion: hcv.VersionNumber.toString(),
    DeploymentStrategyId: "AppConfig.Linear20PercentEvery6Minutes",
  })
);
```

使用AWS AppConfig代理程式讀取自由格式組態設定描述檔

下列每個範例都包含有關程式碼所執行動作的註解。

Java

```
public void retrieveConfigFromAgent() throws Exception {
        /*
        In this sample, we will retrieve configuration data from the AWS AppConfig
 Agent.
        The agent is a sidecar process that handles retrieving configuration data
 from AppConfig
        for you in a way that implements best practices like configuration caching.

        For more information about the agent, see Simplified retrieval methods
        */
```

```
        // The agent runs a local HTTP server that serves configuration data
        // Make a GET request to the agent's local server to retrieve the
 configuration data
        URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyConfigProfile");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        StringBuilder content;
        try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
            content = new StringBuilder();
            int ch;
            while ((ch = in.read()) != -1) {
                content.append((char) ch);
            }
        }
        con.disconnect();
        System.out.println("Configuration from agent via HTTP: " + content);
    }
```

Python

```
# in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
# the agent is a sidecar process that handles retrieving configuration data from AWS
 AppConfig
# for you in a way that implements best practices like configuration caching.
#
# for more information about the agent, see
# Simplified retrieval methods
#

import requests

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

# the agent runs a local HTTP server that serves configuration data
# make a GET request to the agent's local server to retrieve the configuration data
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}")
config = response.content
```

JavaScript

```
// in this sample, we will retrieve configuration data from the AWS AppConfig Agent.
// the agent is a sidecar process that handles retrieving configuration data from
 AppConfig
// for you in a way that implements best practices like configuration caching.

// for more information about the agent, see
// Simplified retrieval methods

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// the agent runs a local HTTP server that serves configuration data
// make a GET request to the agent's local server to retrieve the configuration data
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}`;
const response = await fetch(url);
const config = await response.text(); // (use `await response.json()` if your config
 is json)
```

使用AWS AppConfig代理程式讀取特定功能旗標

下列每個範例都包含有關程式碼所執行動作的註解。

Java

```
public void retrieveSingleFlagFromAgent() throws Exception {
        /*
          You can retrieve a single flag's data from the agent by providing the
 "flag" query string parameter.
          Note: the configuration's type must be AWS.AppConfig.FeatureFlags
        */

        URL url = new URL("http://localhost:2772/applications/MyDemoApp/
environments/Beta/configurations/MyFlagsProfile?flag=myFlagKey");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        StringBuilder content;
        try (BufferedReader in = new BufferedReader(new
 InputStreamReader(con.getInputStream()))) {
```

```
            content = new StringBuilder();
            int ch;
            while ((ch = in.read()) != -1) {
                content.append((char) ch);
            }
        }
        con.disconnect();
        System.out.println("MyFlagName from agent: " + content);
    }
```

Python

```
import requests

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'
flag_key = 'MyFlag'

# retrieve a single flag's data by providing the "flag" query string parameter
# note: the configuration's type must be AWS.AppConfig.FeatureFlags
response = requests.get(f"http://localhost:2772/applications/{application_name}/
environments/{environment_name}/configurations/{config_profile_name}?
flag={flag_key}")
config = response.content
```

JavaScript

```
const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";
const flag_name = "MyFlag";

// retrieve a single flag's data by providing the "flag" query string parameter
// note: the configuration's type must be AWS.AppConfig.FeatureFlags
const url = `http://localhost:2772/applications/${application_name}/environments/
${environment_name}/configurations/${config_profile_name}?flag=${flag_name}`;
const response = await fetch(url);
const flag = await response.json(); // { "enabled": true/false }
```

使用 GetLatestConfig API 動作讀取自由格式組態設定描述檔

下列每個範例都包含有關程式碼所執行動作的註解。本節中的範例會呼叫下列 API：

- GetLatestConfiguration
- StartConfigurationSession

Java

```
public void retrieveConfigFromApi() {
        /*
        The example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.
        For more information on these APIs, see AWS AppConfig Data         */
        AppConfigDataClient appConfigData = AppConfigDataClient.create();

        /*
        Start a new configuration session using the StartConfigurationSession API.
 This operation does not return configuration data.
        Rather, it returns an initial configuration token that should be passed to
 GetLatestConfiguration.
        IMPORTANT: This operation should only be performed once (per configuration),
 prior to the first GetLatestConfiguration
        call you preform. Each GetLatestConfiguration will return a new
 configuration token that you should then use in the
        next GetLatestConfiguration call.
        */
        StartConfigurationSessionResponse session =
            appConfigData.startConfigurationSession(req -> req
                .applicationIdentifier("MyDemoApp")
                .configurationProfileIdentifier("MyConfigProfile")
                .environmentIdentifier("Beta"));

        /*
        Retrieve configuration data using the GetLatestConfiguration API. The first
 time you call this API your configuration
        data will be returned. You should cache that data (and the configuration
 token) and update that cache asynchronously
        by regularly polling the GetLatestConfiguration API in a background thread.
 If you already have the latest configuration
        data, subsequent GetLatestConfiguration calls will return an empty response.
 If you then deploy updated configuration
```

```
        data the next time you call GetLatestConfiguration it will return that
 updated data.

        You can also avoid all the complexity around writing this code yourself by
 leveraging our agent instead.
        For more information about the agent, see Simplified retrieval methods
        */

        // The first getLatestConfiguration call uses the token from
 StartConfigurationSession
        String configurationToken = session.initialConfigurationToken();
        GetLatestConfigurationResponse configuration =

 appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

        System.out.println("Configuration retrieved via API: " +
 configuration.configuration().asUtf8String());

        // You'll want to hold on to the token in the getLatestConfiguration
 response because you'll need to use it
        // the next time you call
        configurationToken = configuration.nextPollConfigurationToken();
        configuration =

 appConfigData.getLatestConfiguration(GetLatestConfigurationRequest.builder().configurationT

        // Try creating a new deployment at this point to see how the output below
 changes.
        if (configuration.configuration().asByteArray().length != 0) {
            System.out.println("Configuration contents have changed
 since the last GetLatestConfiguration call, new contents = " +
 configuration.configuration().asUtf8String());
        } else {
            System.out.println("GetLatestConfiguration returned an empty response
 because we already have the latest configuration");
        }
    }
```

Python

```
# the example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.
#
```

```
# for more information on these APIs, see
# AWS AppConfig Data
#

import boto3

application_name = 'MyDemoApp'
environment_name = 'MyEnvironment'
config_profile_name = 'MyConfigProfile'

appconfigdata = boto3.client('appconfigdata')

# start a new configuration session.
# this operation does not return configuration data.
# rather, it returns an initial configuration token that should be passed to
 GetLatestConfiguration.
#
# note: this operation should only be performed once (per configuration).
#   all subsequent calls to AppConfigData should be via GetLatestConfiguration.
scs = appconfigdata.start_configuration_session(
    ApplicationIdentifier=application_name,
    EnvironmentIdentifier=environment_name,
    ConfigurationProfileIdentifier=config_profile_name)
initial_token = scs['InitialConfigurationToken']

# retrieve configuration data from the session.
# this operation returns your configuration data.
# each invocation of this operation returns a unique token that should be passed to
 the subsequent invocation.
#
# note: this operation does not always return configuration data after the first
 invocation.
#   data is only returned if the configuration has changed within AWS AppConfig
 (i.e. a deployment occurred).
#   therefore, you should cache the data returned by this call so that you can use
 it later.
glc = appconfigdata.get_latest_configuration(ConfigurationToken=initial_token)
config = glc['Configuration'].read()
```

JavaScript

```
// the example below uses two AppConfigData APIs: StartConfigurationSession and
 GetLatestConfiguration.
```

```
// for more information on these APIs, see
// AWS AppConfig Data

import {
  AppConfigDataClient,
  GetLatestConfigurationCommand,
  StartConfigurationSessionCommand,
} from "@aws-sdk/client-appconfigdata";

const appconfigdata = new AppConfigDataClient();

const application_name = "MyDemoApp";
const environment_name = "MyEnvironment";
const config_profile_name = "MyConfigProfile";

// start a new configuration session.
// this operation does not return configuration data.
// rather, it returns an initial configuration token that should be passed to
 GetLatestConfiguration.
//
// note: this operation should only be performed once (per configuration).
//   all subsequent calls to AppConfigData should be via GetLatestConfiguration.
const scs = await appconfigdata.send(
  new StartConfigurationSessionCommand({
    ApplicationIdentifier: application_name,
    EnvironmentIdentifier: environment_name,
    ConfigurationProfileIdentifier: config_profile_name,
  })
);
const { InitialConfigurationToken } = scs;

// retrieve configuration data from the session.
// this operation returns your configuration data.
// each invocation of this operation returns a unique token that should be passed to
 the subsequent invocation.
//
// note: this operation does not always return configuration data after the first
 invocation.
//   data is only returned if the configuration has changed within AWS AppConfig
 (i.e. a deployment occurred).
//   therefore, you should cache the data returned by this call so that you can use
 it later.
const glc = await appconfigdata.send(
```

```
  new GetLatestConfigurationCommand({
    ConfigurationToken: InitialConfigurationToken,
  })
);
const config = glc.Configuration.transformToString();
```

清理您的環境

如果您執行本節中的一或多個程式碼範例，建議您使用下列其中一個範例來尋找並刪除這些程式碼範例所建立的AWS AppConfig資源。本節中的範例會呼叫下列 API：

- ListApplications
- DeleteApplication
- ListEnvironments
- DeleteEnvironments
- ListConfigurationProfiles
- DeleteConfigurationProfile
- ListHostedConfigurationVersions
- DeleteHostedConfigurationVersion

Java

```
/*
    This sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.

    WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
    configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
    an application that you may need in the future.
    */

    public void cleanUpDemoResources() {
        AppConfigClient appconfig = AppConfigClient.create();

        // The name of the application to delete
```

```
        // IMPORTANT: verify this name corresponds to the application you wish to
 delete
        String applicationToDelete = "MyDemoApp";


 appconfig.listApplicationsPaginator(ListApplicationsRequest.builder().build()).items().forE
 -> {
            if (app.name().equals(applicationToDelete)) {
                System.out.println("Deleting App: " + app);
                appconfig.listConfigurationProfilesPaginator(req ->
 req.applicationId(app.id())).items().forEach(cp -> {
                    System.out.println("Deleting Profile: " + cp);
                    appconfig
                        .listHostedConfigurationVersionsPaginator(req -> req
                            .applicationId(app.id())
                            .configurationProfileId(cp.id()))
                        .items()
                        .forEach(hcv -> {
                            System.out.println("Deleting HCV: " + hcv);
                            appconfig.deleteHostedConfigurationVersion(req -> req
                                .applicationId(app.id())
                                .configurationProfileId(cp.id())
                                .versionNumber(hcv.versionNumber()));
                        });
                    appconfig.deleteConfigurationProfile(req -> req
                        .applicationId(app.id())
                        .configurationProfileId(cp.id()));
                });

                appconfig.listEnvironmentsPaginator(req-
>req.applicationId(app.id())).items().forEach(env -> {
                    System.out.println("Deleting Environment: " + env);
                    appconfig.deleteEnvironment(req-
>req.applicationId(app.id()).environmentId(env.id()));
                });

                appconfig.deleteApplication(req -> req.applicationId(app.id()));
            }
        });
    }
```

Python

```
# this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.
#
# WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
#   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
#   an application that you may need in the future.
#

import boto3

# the name of the application to delete
# IMPORTANT: verify this name corresponds to the application you wish to delete
application_name = 'MyDemoApp'

# create and iterate over a list paginator such that we end up with a list of pages,
 which are themselves lists of applications
# e.g. [ [{'Name':'MyApp1',...},{'Name':'MyApp2',...}], [{'Name':'MyApp3',...}] ]
list_of_app_lists = [page['Items'] for page in
 appconfig.get_paginator('list_applications').paginate()]
# retrieve the target application from the list of lists
application = [app for apps in list_of_app_lists for app in apps if app['Name'] ==
 application_name][0]
print(f"deleting application {application['Name']} (id={application['Id']})")

# delete all configuration profiles
list_of_config_lists = [page['Items'] for page in
 appconfig.get_paginator('list_configuration_profiles').paginate(ApplicationId=application['
for config_profile in [config for configs in list_of_config_lists for config in
 configs]:
    print(f"\tdeleting configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

    # delete all hosted configuration versions
    list_of_hcv_lists = [page['Items'] for page in
 appconfig.get_paginator('list_hosted_configuration_versions').paginate(ApplicationId=applic
 ConfigurationProfileId=config_profile['Id'])]
    for hcv in [hcv for hcvs in list_of_hcv_lists for hcv in hcvs]:

 appconfig.delete_hosted_configuration_version(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'], VersionNumber=hcv['VersionNumber'])
```

```
        print(f"\t\tdeleted hosted configuration version {hcv['VersionNumber']}")

    # delete the config profile itself
    appconfig.delete_configuration_profile(ApplicationId=application['Id'],
 ConfigurationProfileId=config_profile['Id'])
    print(f"\tdeleted configuration profile {config_profile['Name']}
 (Id={config_profile['Id']})")

# delete all environments
list_of_env_lists = [page['Items'] for page in
 appconfig.get_paginator('list_environments').paginate(ApplicationId=application['Id'])]
for environment in [env for envs in list_of_env_lists for env in envs]:
    appconfig.delete_environment(ApplicationId=application['Id'],
 EnvironmentId=environment['Id'])
    print(f"\tdeleted environment {environment['Name']} (Id={environment['Id']})")

# delete the application itself
appconfig.delete_application(ApplicationId=application['Id'])
print(f"deleted application {application['Name']} (id={application['Id']})")
```

JavaScript

```
// this sample provides cleanup code that deletes all the AWS AppConfig resources
 created in the samples above.

// WARNING: this code will permanently delete the given application and all of its
 sub-resources, including
//   configuration profiles, hosted configuration versions, and environments. DO NOT
 run this code against
//   an application that you may need in the future.

import {
  AppConfigClient,
  paginateListApplications,
  DeleteApplicationCommand,
  paginateListConfigurationProfiles,
  DeleteConfigurationProfileCommand,
  paginateListHostedConfigurationVersions,
  DeleteHostedConfigurationVersionCommand,
  paginateListEnvironments,
  DeleteEnvironmentCommand,
} from "@aws-sdk/client-appconfig";
```

```
const client = new AppConfigClient();

// the name of the application to delete
// IMPORTANT: verify this name corresponds to the application you wish to delete
const application_name = "MyDemoApp";

// iterate over all applications, deleting ones that have the name defined above
for await (const app_page of paginateListApplications({ client }, {})) {
  for (const application of app_page.Items) {

    // skip applications that dont have the name thats set
    if (application.Name !== application_name) continue;

    console.log( `deleting application ${application.Name} (id=${application.Id})`);

    // delete all configuration profiles
    for await (const config_page of paginateListConfigurationProfiles({ client },
 { ApplicationId: application.Id })) {
      for (const config_profile of config_page.Items) {
        console.log(`\tdeleting configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`);

        // delete all hosted configuration versions
        for await (const hosted_page of
 paginateListHostedConfigurationVersions({ client },
          { ApplicationId: application.Id, ConfigurationProfileId:
 config_profile.Id }
        )) {
          for (const hosted_config_version of hosted_page.Items) {
            await client.send(
              new DeleteHostedConfigurationVersionCommand({
                ApplicationId: application.Id,
                ConfigurationProfileId: config_profile.Id,
                VersionNumber: hosted_config_version.VersionNumber,
              })
            );
            console.log(`\t\tdeleted hosted configuration version
 ${hosted_config_version.VersionNumber}`);
          }
        }

        // delete the config profile itself
        await client.send(
          new DeleteConfigurationProfileCommand({
```

```
            ApplicationId: application.Id,
            ConfigurationProfileId: config_profile.Id,
          })
        );
        console.log(`\tdeleted configuration profile ${config_profile.Name} (Id=
${config_profile.Id})`)
      }

      // delete all environments
      for await (const env_page of paginateListEnvironments({ client },
 { ApplicationId: application.Id })) {
        for (const environment of env_page.Items) {
          await client.send(
            new DeleteEnvironmentCommand({
              ApplicationId: application.Id,
              EnvironmentId: environment.Id,
            })
          );
          console.log(`\tdeleted environment ${environment.Name} (Id=
${environment.Id})`)
        }
      }
    }

    // delete the application itself
    await client.send(
      new DeleteApplicationCommand({ ApplicationId: application.Id })
    );
    console.log(`deleted application ${application.Name} (id=${application.Id})`)
  }
}
```

AWS AppConfig 中的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同的責任模式將其稱為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護執行 AWS 雲端 內 AWS 服務的基礎設施。AWS 提供的服務，也可讓您安全使用。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計畫的一部分。若要了解適用於 AWS Systems Manager 的合規計畫，請參閱合規計畫的 AWS 服務範圍。
- 雲端內部的安全 – 您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的請求和適用法律和法規。

AWS AppConfig 是 AWS Systems Manager 的功能。若要瞭解如何在使用時套用共用責任模型AWS AppConfig，請參閱中的安全性AWS Systems Manager。本節說明如何設定 Systems Manager 以符合的安全性和合規性目標AWS AppConfig。

實作最低權限存取

安全性最佳作法是授與身分識別在特定條件下對特定資源執行特定動作所需的最低必要權限。 AWS AppConfig代理程式提供兩種功能，可讓代理程式存取執行個體或容器的檔案系統：備份和寫入磁碟。如果您啟用這些功能，請確認只有AWS AppConfig代理程式具有寫入檔案系統上指定組態檔的權限。此外，請確認只有從這些組態檔案讀取所需的程序才能執行此動作。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。

如需有關實作最低權限存取的詳細資訊，請參閱《AWS Well-Architected Tool使用指南》中的 SEC03-BP02 授與最少權限存取權。如需本節所述AWS AppConfig代理程式功能的詳細資訊，請參閱其他擷取功能。

靜態資料加密 AWS AppConfig

AWS AppConfig默認情況下提供加密，以使用保護靜態客戶數據AWS 擁有的金鑰。

AWS 擁有的金鑰— 預設情況下，AWS AppConfig使用這些金鑰來自動加密服務部署並託管在AWS AppConfig資料存放區中的資料。您無法檢視、管理或使用AWS 擁有的金鑰，或稽核其使用情況。不過，您不需要採取任何動作或變更任何程式，即可保護加密您資料的金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS 擁有的金鑰。

雖然您無法停用此層加密或選取替代加密類型，但您可以指定在儲存資料存放區中託管的組態資料以及部署組態AWS AppConfig資料時使用的客戶管理金鑰。

客戶管理金鑰 — AWS AppConfig 支援使用您建立、擁有和管理的對稱客戶管理金鑰，以針對現有 AWS 擁有的金鑰金鑰新增第二層加密。您可以完全控制此層加密，因此能執行以下任務：

- 建立和維護關鍵政策和補助金
- 建立和維護 IAM 政策
- 啟用和停用金鑰政策
- 輪換金鑰密碼編譯資料
- 新增標籤
- 建立金鑰別名
- 安排金鑰供刪除

如需詳細資訊，請參閱AWS Key Management Service開發人員指南中的客戶管理金鑰。

AWS AppConfig支援客戶管理的金鑰

AWS AppConfig為配置數據提供客戶管理的密鑰加密支持。對於儲存至AWS AppConfig託管資料存放區的組態版本，客戶可以在對應的組態設定檔`KmsKeyIdentifier`上設定。每次使用 `CreateHostedConfigurationVersion` API 作業建立新版本的組態資料時，都會從中AWS AppConfig產生AWS KMS資料金鑰，以`KmsKeyIdentifier`便在儲存資料之前加密資料。稍後在`GetHostedConfigurationVersion`或 `StartDeployment` API 作業期間存取資料時，會使用產生的資料金鑰相關資訊AWS AppConfig解密組態資料。

AWS AppConfig還為部署的組態資料提供客戶管理的金鑰加密支援。若要加密組態資料，客戶可以提供`KmsKeyIdentifier`給其部署。 AWS AppConfig使用此生成AWS KMS數據密鑰`KmsKeyIdentifier`以加密 `StartDeployment` API 操作的數據。

AWS AppConfig加密存取

建立客戶管理的金鑰時，請使用下列金鑰原則來確保金鑰可以使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "Allow use of the key",
```

```
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::account_ID:role/role_name"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
            ],
        "Resource": "*"
        }
 ]
```

若要使用客戶受管金鑰加密託管組態資料，身分呼叫CreateHostedConfigurationVersion需要下列可指派給使用者、群組或角色的原則陳述式：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

如果您使用 Secrets Manager 密碼或任何其他使用客戶管理金鑰加密的設定資料，您retrievalRoleArn將需kms:Decrypt要解密並擷取資料。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:Region:account_ID:configuration source/object"
    }
  ]
}
```

呼叫 AWS AppConfig StartDeploymentAPI 作業時，身分呼叫StartDeployment需要下列可指派給使用者、群組或角色的 IAM 政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey*"
      ],
      "Resource": "arn:aws:kms:Region:account_ID:key_ID"
    }
  ]
}
```

呼叫 AWS AppConfig GetLatestConfigurationAPI 作業時，身分呼叫GetLatestConfiguration需要下列可指派給使用者、群組或角色的原則：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt,
            "Resource": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ]
}
```

加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 會使用加密內容作為其他驗證資料，以支援驗證的加密。在加密資料的請求中包含加密內容時，AWS KMS 會將加密內容繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

AWS AppConfig加密內容：在所有密AWS KMS碼編譯作業中AWS AppConfig使用加密內容，用於加密的託管組態資料和部署。上下文包含對應於數據類型和標識特定數據項的值的鍵。

監控您的加密金鑰 AWS

將AWS KMS客戶受管金鑰搭配使用時AWS AppConfig，您可以使用AWS CloudTrail或 Amazon CloudWatch Logs 追蹤AWS AppConfig傳送至的請求AWS KMS。

下列範例是一個 CloudTrail 事件，用Decrypt於AWS KMS監視呼叫者存AWS AppConfig取由客戶管理金鑰加密之資料的作業：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "appconfig.amazonaws.com"
    },
    "eventTime": "2023-01-03T02:22:28z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "Region",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:appconfig:deployment:arn":
 "arn:aws:appconfig:Region:account_ID:application/application_ID/
environment/environment_ID/deployment/deployment_ID"
        },
        "keyId": "arn:aws:kms:Region:account_ID:key/key_ID",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account_ID",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:Region:account_ID:key_ID"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "account_ID",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

使AWS AppConfig用介面端點存取 (AWS PrivateLink)

您可以使AWS PrivateLink用在 VPC 和AWS AppConfig. 之間建立私人連線。您可以AWS AppConfig像在 VPC 中一樣進行存取，而無需使用網際網路閘道、NAT 裝置、VPN 連線或AWS Direct Connect連線。VPC 中的執行個體不需要公用 IP 位址即可存取AWS AppConfig。

您可以建立由 AWS PrivateLink 提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 AWS AppConfig 之流量的進入點。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的透過 AWS PrivateLink 存取 AWS 服務。

AWS AppConfig 的考量

設定的介面端點之前AWS AppConfig，請先檢閱AWS PrivateLink指南中的考量事項。

AWS AppConfig支援透過介面端點對appconfig和appconfigdata服務進行呼叫。

為 AWS AppConfig 建立介面端點

您可以建立介面端點以AWS AppConfig使用 Amazon VPC 主控台或 AWS Command Line Interface (AWS CLI)。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的建立介面端點。

建立AWS AppConfig使用下列服務名稱的介面端點：

```
com.amazonaws.region.appconfig
```

```
com.amazonaws.region.appconfigdata
```

如果您為介面端點啟用私有 DNS，您可以AWS AppConfig使用其預設的區域 DNS 名稱向 API 要求。例如，`appconfig.us-east-1.amazonaws.com` 和 `appconfigdata.us-east-1.amazonaws.com`。

為您的介面端點建立端點政策

端點政策為 IAM 資源，您可將其連接至介面端點。預設端點策略允許AWS AppConfig透過介面端點進行完整存取。若要控制允許AWS AppConfig從您的 VPC 存取，請將自訂端點原則附加到介面端點。

端點政策會指定以下資訊：

- 可執行動作 (AWS 帳戶、IAM 使用者和 IAM 角色) 的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制對服務的存取。

範例：AWS AppConfig 動作的 VPC 端點政策

以下是自訂端點政策的範例。將此政策附加至介面端點後，此政策會針對所有資源上的所有主體，授予列出的 AWS AppConfig 動作的存取權限。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "appconfig:CreateApplication",
            "appconfig:CreateEnvironment",
            "appconfig:CreateConfigurationProfile",
            "appconfig:StartDeployment",
            "appconfig:GetLatestConfiguration"
            "appconfig:StartConfigurationSession"
         ],
         "Resource":"*"
      }
   ]
}
```

密鑰管理器密鑰旋轉

本節說明與 Secrets Manager AWS AppConfig 整合的重要安全性資訊。如需 Secrets Manager 的相關資訊，請參閱什麼是AWS Secrets Manager？ 在《AWS Secrets Manager使用者指南》中。

設定由部署的 Secrets Manager 密碼的自動輪替 AWS AppConfig

輪換是定期更新秘密管理員中儲存的密碼的程序。當您輪換秘密時，會更新秘密和資料庫或服務中的憑證。您可以使用更新密碼和資料庫的AWS Lambda功能，在 Secrets Manager 中設定自動密碼輪換。如需詳細資訊，請參閱AWS Secrets Manager使用指南中的旋轉AWS Secrets Manager密碼。

若要啟用由部署的 Secrets Manager 密鑰的金鑰輪替AWS AppConfig，請更新您的輪換 Lambda 函數並部署輪換的密碼。

> **Note**
>
> 在您的密碼輪替並完全更新到新版本後，部署您的AWS AppConfig配置描述檔。您可以決定密碼是否因為狀態從`VersionStage`變更為而旋`AWSPENDING`轉`AWSCURRENT`。秘密輪換完成會在 Secrets Manager 輪換範本`finish_secret`功能中發生。

這是一個示例函數，它在秘密旋轉後啟動AWS AppConfig部署。

```
import time
import boto3
client = boto3.client('appconfig')

def finish_secret(service_client, arn, new_version):
    """Finish the rotation by marking the pending secret as current
    This method finishes the secret rotation by staging the secret staged AWSPENDING
 with the AWSCURRENT stage.
    Args:
        service_client (client): The secrets manager service client
        arn (string): The secret ARN or other identifier
        new_version (string): The new version to be associated with the secret
    """
    # First describe the secret to get the current version
    metadata = service_client.describe_secret(SecretId=arn)
    current_version = None
    for version in metadata["VersionIdsToStages"]:
        if "AWSCURRENT" in metadata["VersionIdsToStages"][version]:
            if version == new_version:
                # The correct version is already marked as current, return
                logger.info("finishSecret: Version %s already marked as AWSCURRENT for
 %s" % (version, arn))
                return
            current_version = version
            break

    # Finalize by staging the secret version current
    service_client.update_secret_version_stage(SecretId=arn, VersionStage="AWSCURRENT",
 MoveToVersionId=new_version, RemoveFromVersionId=current_version)
```

```
    # Deploy rotated secret
    response = client.start_deployment(
            ApplicationId='TestApp',
            EnvironmentId='TestEnvironment',
            DeploymentStrategyId='TestStrategy',
            ConfigurationProfileId='ConfigurationProfileId',
            ConfigurationVersion=new_version,
            KmsKeyIdentifier=key,
            Description='Deploy secret rotated at ' + str(time.time())
        )

    logger.info("finishSecret: Successfully set AWSCURRENT stage to version %s for
secret %s." % (new_version, arn))
```

監控 AWS AppConfig

監控是維護 AWS AppConfig 及其他 AWS 解決方案的可靠性、可用性和效能的重要部分。AWS 提供以下監控工具，可讓您監看 AWS AppConfig、在發現錯誤時回報，並適時採取自動動作。

- AWS CloudTrail 擷取您 AWS 帳戶發出或代表發出的 API 呼叫和相關事件，並傳送記錄檔案至您指定的 Simple Storage Service (Amazon S3) 儲存貯體。您可以找出哪些使用者和帳戶呼叫 AWS、發出呼叫的來源 IP 地址，以及呼叫的發生時間。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/。
- Amazon CloudWatch 日誌可讓您從 Amazon EC2 執行個體和其他來源監控 CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。

主題

使用 AWS CloudTrail 記錄 AWS AppConfig API 呼叫

AWS AppConfig與 (提供中的使用者AWS CloudTrail、角色或服務所採取的動作記錄) 的AWS服務整合AWS AppConfig。 CloudTrail 擷取AWS AppConfig作為事件的所有 API 呼叫。擷取的呼叫包括從 AWS AppConfig 主控台進行的呼叫，以及針對 AWS AppConfig API 操作的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail事件到 Amazon S3 儲存貯體，包括AWS AppConfig. 如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷提出的要求AWS AppConfig、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

AWS AppConfig中的資訊 CloudTrail

CloudTrail 在您創建帳戶AWS 帳戶時啟用。當活動發生在中時AWS AppConfig，該活動會與事件歷史記錄中的其他AWS服務 CloudTrail 事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶 的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

如需您 AWS 帳戶 帳戶中正在進行事件的記錄 (包含 AWS AppConfig 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail 日誌文件

所有AWS AppConfig動作均由「API 參考」記錄 CloudTrail 並記錄在「AWS AppConfigAPI 參考」中。例如，呼叫`GetApplication`和`ListApplications`動作會`CreateApplication`在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

AWS AppConfig資料事件 CloudTrail

資料事件提供在資源上或在資源中執行之資源作業的相關資訊 (例如，透過呼叫擷取最新部署的組態 GetLatestConfiguration)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。依預設，CloudTrail 不會記錄資料事件。 CloudTrail 事件歷史記錄不會記錄數據事件。

資料事件需支付額外的費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail定價。

您可以使用 CloudTrail 主控台或 CloudTrail API 作業記錄AWS AppConfig資源類型的資料事件。AWS CLI此段落中的表格顯示可用的資源類型AWS AppConfig。

- 若要使用 CloudTrail 主控台記錄資料事件，請建立追蹤或事件資料存放區以記錄資料事件，或更新現有追蹤或事件資料存放區以記錄資料事件。

  1. 選擇 [資料事件] 以記錄資料事件。

2. 從「資料」事件類型清單中選擇AWS AppConfig。
3. 選擇您要使用的記錄選取器範本。您可以記錄資源類型的所有資料事件、記錄所有`readOnly`事件、記錄所有`writeOnly`事件，或建立自訂記錄選取器範本以篩選`readOnly`、`eventName`、和`resources.ARN`欄位。
4. 輸入做為「選取器名稱」AppConfigDataEvents。如需為資料事件追蹤啟用 Amazon CloudWatch 日誌的相關資訊，請參閱記錄資AWS AppConfig料平面呼叫的指標。

- 若要使用記錄資料事件AWS CLI，請配置`--advanced-event-selectors`參數以將`eventCategory`欄位設定為等於，`Data`且`resources.type`欄位等於資源類型值 (請參閱表格)。您可以加入條件以篩選`readOnly`、`eventName`、和`resources.ARN`欄位的值。
  - 若要設定追蹤以記錄資料事件，請執行put-event-selectors命令。如需詳細資訊，請參閱使用 AWS CLI.
  - 若要規劃事件資料倉庫以記錄資料事件，請執行create-event-data-store指令以建立新的事件資料倉庫以記錄資料事件，或執行update-event-data-store指令來更新現有的事件資料倉庫。如需詳細資訊，請參閱使用 AWS CLI.

下表列出 AWS AppConfig 資源類型。[資料事件類型 (主控台)] 欄顯示可從主控台的 [資料事件類型 CloudTrail] 清單中選擇的值。resource .type 值欄會顯示**`resources.type`**值，您可以在使用或 API 設定進階事件選取器時指定這個值。AWS CLI CloudTrail 記錄到資料 CloudTrail欄中的資料 API 會顯示 CloudTrail針對資源類型記錄的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	資料 API 已記錄至 CloudTrail *
AWS AppConfig	`AWS::AppConfig::Configuration`	• GetLatestConfiguration • StartConfigurationSession

* 您可以設定進階事件選取器來篩選`eventName`、`readOnly`、和`resources.ARN`欄位，以僅記錄對您很重要的事件。如需有關這些欄位的詳細資訊，請參閱 AdvancedFieldSelector。

AWS AppConfig管理事件 CloudTrail

管理事件提供在您的 AWS 帳戶中的資源上執行的管理作業的相關資訊。這些也稱為控制平面操作。依預設，會 CloudTrail 記錄管理事件。

AWS AppConfig將所有AWS AppConfig控制平面作業記錄為管理事件。如需記AWS AppConfig錄到的AWS AppConfig控制平面作業清單 CloudTrail，請參閱 AWS AppConfigAPI 參考。

了解 AWS AppConfig 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範StartConfigurationSession動作的 CloudTrail 記錄項目。

```
{
      "eventVersion": "1.09",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Administrator",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {},
          "attributes": {
            "creationDate": "2024-01-11T14:37:02Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2024-01-11T14:45:15Z",
      "eventSource": "appconfig.amazonaws.com",
      "eventName": "StartConfigurationSession",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "Boto3/1.34.11 md/Botocore#1.34.11 ua/2.0 os/macos#22.6.0
 md/arch#x86_64 lang/python#3.11.4 md/pyimpl#CPython cfg/retry-mode#legacy
 Botocore/1.34.11",
      "requestParameters": {
        "applicationIdentifier": "rrfexample",
        "environmentIdentifier": "mexampleqe0",
        "configurationProfileIdentifier": "3eexampleu1"
      },
      "responseElements": null,
      "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
      "readOnly": false,
      "resources": [
```

```
        {
          "accountId": "123456789012",
          "type": "AWS::AppConfig::Configuration",
          "ARN": "arn:aws:appconfig:us-east-1:123456789012:application/rrfexample/
environment/mexampleqe0/configuration/3eexampleu1"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "123456789012",
      "eventCategory": "Data",
      "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "appconfigdata.us-east-1.amazonaws.com"
      }
    }
```

記錄資AWS AppConfig料平面呼叫的指標

如果您設定AWS CloudTrail為記錄AWS AppConfig資料事件，則可以啟用 Amazon CloudWatch Logs 來記錄呼叫AWS AppConfig資料平面的指標。然後，您可以建立一或多個量度篩選器，在 CloudWatch 記錄檔中搜尋和篩選記錄資料。量度篩選器會定義傳送至記錄檔時要在 CloudWatch 記錄檔資料中尋找的術語和模式。 CloudWatch 日誌使用指標過濾器將日誌數據轉換為數字 CloudWatch 指標。您可以繪製指標圖形或使用警示進行設定。

開始之前

啟用中的AWS AppConfig資料事件記錄AWS CloudTrail。下列程序說明如何啟用中現有AWS AppConfig軌跡的測量結果記錄日誌 CloudTrail。如需如何為資AWS AppConfig料計劃呼叫啟用 CloudTrail 記錄的相關資訊，請參閱AWS AppConfig資料事件 CloudTrail。

使用下列程序啟用「 CloudWatch 記錄」，以記錄呼叫AWS AppConfig資料平面的測量結果。

啟用 CloudWatch Logs 以記錄AWS AppConfig資料平面呼叫的度量

1. 請在以下位置開啟 CloudTrail 主控台。 https://console.aws.amazon.com/cloudtrail/
2. 在儀表板上，選擇您的AWS AppConfig路線。
3. 在「CloudWatch 記錄檔」區段中，選擇 「編輯」。
4. 選擇 Enable (啟用)。

5. 對於「記錄群組名稱」，請保留預設名稱或輸入名稱。記下名稱。您稍後會在「記錄」主控台中選擇 CloudWatch 記錄群組。
6. 在 Role name (角色名稱) 中，輸入名稱。
7. 選擇儲存變更。

請使用下列程序AWS AppConfig在「 CloudWatch 記錄檔」中建立測量結果和測量結果篩選。此程序說明如何針對呼叫建立operation與 (選擇性) 呼叫的測量結果篩選Amazon Resource Name (ARN)。operation

AWS AppConfig在 CloudWatch 記錄檔中建立測量結果和量度篩選

1. 在開啟 CloudWatch 主控台https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Logs (日誌)，然後選擇 Log groups (日誌群組)。
3. 選擇記AWS AppConfig錄群組旁邊的核取方塊。
4. 選擇 Actions (動作)，然後選擇 Create metric filter (建立指標篩選條件)。
5. 對於「篩選器名稱」，輸入名稱。
6. 針對「篩選器模式」，輸入下列內容：

   ```
   { $.eventSource = "appconfig.amazonaws.com" }
   ```

7. （選擇性）在 「測試模式」區段中，從「選取要測試的記錄資料」清單中選擇您的記錄群組。如果 CloudTrail 尚未記錄任何通話，則可以跳過此步驟。
8. 選擇下一步。
9. 針對測量結果命名空間，輸入 **AWS AppConfig**
10. 對於 Metric name (指標名稱)，輸入 **Calls**。
11. 針對 Metric value (指標值)，輸入 **1**。
12. 略過預設值和單位。
13. 對於「維度名稱」，輸入**operation**。
14. 對於「標註」值，請輸入**$.eventName**。

    (選擇性) 您可以輸入第二個維度，其中包括撥打電話的 Amazon 資源名稱 (ARN)。若要新增第二個維度，請輸入做為「維度名稱」 **resource**。對於「標註」值，請輸入**$.resources[0].ARN**。

    選擇下一步。

15. 檢閱篩選器的詳細資訊和建立量度篩選器。

(選擇性) 您可以重複此程序，為特定錯誤代碼 (例如) 建立新的量度篩選器AccessDenied。如果您這樣做，請輸入下列詳細資訊：

1. 對於「篩選器名稱」，輸入名稱。
2. 針對「篩選器模式」，輸入下列內容：

   ```
   { $.errorCode = "codename" }
   ```

   例如

   ```
   { $.errorCode = "AccessDenied" }
   ```

3. 針對測量結果命名空間，輸入 **AWS AppConfig**
4. 對於 Metric name (指標名稱)，輸入 **Errors**。
5. 針對 Metric value (指標值)，輸入 **1**。
6. 在「預設值」中，輸入零 (0)。
7. 略過單位、尺寸和警報。

CloudTrail 記錄 API 呼叫後，您可以在中檢視指標 CloudWatch。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的在主控台中檢視指標和日誌。如需如何尋找您建立之測量結果的相關資訊，請參閱搜尋可用的測量結果。

> **Note**
>
> 如果您設定沒有維度的錯誤測量結果 (如此處所述)，您可以在「沒有維度的測量結果」頁面上檢視這些測量結果。

建立 CloudWatch量度的警示

建立指標後，您可以在中建立量度警示 CloudWatch。例如，您可以為在上一個程序中建立的AWS AppConfig呼叫量度建立警示。具體而言，您可以針對超過閾值的 AWS AppConfig `StartConfigurationSession` API 動作的呼叫建立警示。有關如何為指標建立警示的資訊，請參閱 Amazon CloudWatch 使用者指南中的根據靜態閾值建立 CloudWatch 警示。如需有關呼叫資AWS

AppConfig料平面之預設限制的詳細資訊，請參閱《》中的資料平面預設限制Amazon Web Services 一般參考。

AWS AppConfig 使用者指南文件歷史

下表說明自上次發行版本以來文件的重要變更 AWS AppConfig。

目前應用程式介面版本:

變更	描述	日期
AWS AppConfig 自定義擴展示例	逐步解說：建立自訂 AWS AppConfig 擴充功能主題現在包含下列範例擴充功能的連結 GitHub： • 範例擴充功能，可防止使用 Systems Manager blocked day 變更行事曆暫停行事曆進行部署 • 使用 git-secret 防止密碼洩漏到配置數據中的示例擴展 • 使用 Amazon Comprehend 防止個人識別資訊 (PII) 洩漏到組態資料中的範例擴充功能	2024年2月28日
新主題：使用記錄 AWS AppConfig API 呼叫 AWS CloudTrail	AWS AppConfig 與 (提供中的使用者 AWS CloudTrail、角色或服務所採取的動作記錄) 的 AWS 服務整合 AWS AppConfig。 CloudTrail 擷取 AWS AppConfig 作為事件的所有 API 呼叫。此新主題提供 AWS AppConfig特定內容，而不是連結至《AWS Systems Manager 使用指南》中的對應內容。如需詳細資訊，請 AWS	2024年1月18日

	AppConfig 參閱使用 AWS CloudTrail.	
AWS AppConfig 現在支持 AWS PrivateLink	您可以使 AWS PrivateLink 用在 VPC 和 AWS AppConfig. 之間建立私人連線。您可以 AWS AppConfig 像在 VPC 中一樣進行存取，而無需使用網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。VPC 中的執行個體不需要公用 IP 位址即可存取 AWS AppConfig。如需詳細資訊，請參閱 AWS AppConfig 使用介面端點存取 (AWS PrivateLink)。	2023 年 12 月 6 日

其他 AWS AppConfig 代理程式擷取功能和新的本機開發模式	AWS AppConfig 代理程式提供下列其他功能，協助您擷取應用程式的組態。 其他擷取功能 • 多帳戶擷取：使用主帳戶或擷取 AWS 帳戶 中的 AWS AppConfig 代理程式，從多個廠商帳戶擷取組態資料。 • 將組態複本寫入磁碟：使用「 AWS AppConfig 代理程式」將組態資料寫入磁碟。此功能可讓客戶擁有從磁碟讀取組態資料的應用程式以進行整合 AWS AppConfig。 **Note** 將組態寫入磁碟並非設計為組態備份功能。AWS AppConfig 代理程式不會從複製到磁碟的組態檔讀取。如果您想要將組態備份到磁碟，請參閱將AWS AppConfig 代理程式與 Amazon EC2 搭配使用或搭配 Amazon ECS 和 Amazon EKS 使用 AWS AppConfig 代理程式的PRELOAD_BACKUP 環境變數。BACKUP_DIRECTORY	2023 年 12 月 1 日

	本機開發模式 AWS AppConfig 代理程式支援本機開發模式。如果您啟用本機開發模式，代理程式會從磁碟上的指定目錄讀取組態資料。它不會從中檢索配置數據 AWS AppConfig。您可以透過更新指定目錄中的檔案來模擬組態部署。針對下列使用案例，我們建議使用本機開發模式： • 在使用部署不同的配置版本之前測試它們 AWS AppConfig。 • 在將變更提交至程式碼儲存庫之前，請先測試新功能的不同組態選項。 • 測試不同的組態案例，以確認它們如預期般運作。	
新的程式碼範例主題	在本指南中新增了新的程式碼範例主題。本主題包括 Java、Python 中的範例，以及 JavaScript 以程式設計方式執行六個常見 AWS AppConfig 動作的範例。	2023 年 11 月 17 日
修訂的目錄以更好地反映 AWS AppConfig 工作流程	本使用者指南中的內容現在分組在「建立」、「部署」、「擷取」和「延伸工作流程」標題下。這個組織可以更好地反映使用的工作流程， AWS AppConfig 並旨在幫助使內容更容易被發現。	2023 年 11 月 7 日

已新增有效載荷	為自訂 AWS AppConfig 擴充功能建立 Lambda 函數主題現在包含請求和回應承載參考。	2023 年 11 月 7 日
全新 AWS 預先定義的部署	AWS AppConfig 現在提供並建議`AppConfig.Linear20PercentEvery6Minutes` 預先定義的部署策略。有關詳情，請參閱預先定義的部署策略。	2023 年 8 月 11 日
AWS AppConfig 與 Amazon EC2 集成	您可以使用 AWS AppConfig 代理程式 AWS AppConfig 與在 Amazon 彈性運算雲端 (Amazon EC2) Linux 執行個體上執行的應用程式整合。此代理程式支援適用於 Amazon EC2 的 x86_64 和 ARM64 架構。如需詳細資訊，請參閱與 Amazon EC2 AWS AppConfig 整合。	2023 年 7 月 20 日
AWS CloudFormation 支援新 AWS AppConfig 資源和功能旗標範例	AWS CloudFormation 現在支援AWS::AppConfig::Extension和AWS::AppConfig::ExtensionAssociation資源，可協助您開始使用 AWS AppConfig 擴充功能。 AWS::AppConfig::ConfigurationProfile和AWS::AppConfig::HostedConfiguration版本資源現在包括在 AWS AppConfig 託管組態存放區中建立功能旗標組態設定檔的範例。	2023 年 4 月 12 日

AWS AppConfig 與整合 AWS Secrets Manager	AWS AppConfig 與 AWS Secrets Manager. Secrets Manager 可協助您安全地加密、儲存和擷取資料庫和其他服務的認證。除了在應用程式中對憑證進行硬式編碼，您可以在需要時呼叫 Secrets Manager 以擷取憑證。Secrets Manager 可讓您輪換和管理密碼的存取權，協助您保護對 IT 資源和資料的存取。 建立自由格式組態設定檔時，您可以選擇 Secrets Manager 作為組態資料的來源。在建立設定描述檔之前，您必須使用 Secrets Manager 加載並建立密碼。如需有關 Secrets Manager 的詳細資訊，請參閱什麼是 AWS Secrets Manager？ 在《AWS Secrets Manager 使用者指南》中。如需有關建立組態設定檔的資訊，請參閱建立自由格式組態設定檔。	2023 年 2 月 2 日

AWS AppConfig 與 Amazon ECS 和 Amazon EKS 集成	您可以通過使用代理 AWS AppConfig 與 Amazon 彈性容器服務（Amazon ECS）和亞馬遜彈性 Kubernetes 服務（亞馬遜 EKS）集成。 AWS AppConfig 此代理程式可作為與 Amazon ECS 和 Amazon EKS 容器應用程式一起執行的附屬容器運作。代理程式以下列方式增強容器化應用程式的處理與管理： • 代理程式 AWS AppConfig 會使用 AWS Identity and Access Management (IAM) 角色並管理設定資料的本機快取，代表您呼叫。藉由從本機快取提取組態資料，您的應用程式需要較少的程式碼更新來管理組態資料、擷取設定資料 (以毫秒為單位)，而且不會受到可能中斷此類資料呼叫的網路問題影響。 • 代理程式提供擷取和解析 AWS AppConfig 功能旗標的原生體驗。 • 代理程式現成可提供快取策略、輪詢間隔和本機組態資料可用性的最佳作法，同時追蹤後續服務呼叫所需的組態 Token。 • 在背景執行時，代理程式會定期輪詢 AWS AppConfig 資料平面以進行組態資料更新。您的容器化應用程序可	2022 年 12 月 2 日

以通過連接到端口 2772 上的本地主機（可自定義的默認端口值）並調用 HTTP GET 來檢索數據來檢索數據。

- AWS AppConfig 代理程式會更新容器中的組態資料，而不必重新啟動或回收這些容器。

如需詳細資訊，請參閱AWS AppConfig 與 Amazon ECS 和 Amazon E KS 整合。

新的擴展： AWS AppConfig 擴展 CloudWatch 顯然	在推出此功能時，您可以使用 Amazon CloudWatch Evitic 將新功能提供給指定百分比的使用者，以安全地驗證新功能。您可以監控新功能的效能，以協助您決定何時要提升使用者的流量。這可協助您在完全啟動功能之前降低風險並識別意外的後果。您也可以執行 A/B 實驗，根據證據和資料作出功能設計決策。 CloudWatch Evitic 的 AWS AppConfig 擴展允許您的應用程序在本地分配變體給用戶會話，而不是通過調用EvaluateFeature操作。本機工作階段可降低 API 呼叫所帶來的延遲和可用性風險。有關如何設定和使用擴充功能的資訊，請參閱 Amazon 使用 CloudWatch 者指南中的「 CloudWatch明顯地執行啟動和 A/B 實驗」。	2022 年 9 月 13 日
取代 `GetConfiguration` API 動作	2021 年 11 月 18 日， AWS AppConfig 發布了一項新的數據平面服務。此服務會使用 `GetConfiguration` API 動作取代先前擷取設定資料的程序。資料平面服務會使用兩個新的 API 動作StartConfigurationSession和 GetLatestConfiguration. 資料平面服務也會使用新的端點。 如需詳細資訊，請參閱關於資 AWS AppConfig 料平面服務。	2022 年 9 月 13 日

新版 AWS AppConfig 代理程式 Lambda 擴充功能	AWS AppConfig 代理程式 Lambda 擴充功能的 2.0.122 版現已推出。新的擴充功能使用不同的 Amazon 資源名稱 (ARN)。如需詳細資訊，請參閱AWS AppConfig 代理程式 Lambda 擴充功能版本	2022 年 8 月 23 日
啟動 AWS AppConfig 擴充功能	擴充功能可增強您在建立或部署組態的 AWS AppConfig 工作流程期間，在不同點插入邏輯或行為的能力。您可以使用 AWS-authded 擴充功能或建立自己的擴充功能。如需詳細資訊，請參閱使用 AWS AppConfig 擴充功能。	2022 年 7 月 12 日
新版 AWS AppConfig 代理程式 Lambda 擴充功能	AWS AppConfig 代理程式 Lambda 擴充功能的 2.0.58 版現已推出。新的擴充功能使用不同的 Amazon 資源名稱 (ARN)。如需詳細資訊，請參閱 AWS AppConfig Lambda 擴充功能的可用版本。	2022 年 5 月 3 日

AWS AppConfig 與阿特拉西揚吉拉整合	與 Atlassian Jira 整合時，每當您變更指 AWS AppConfig 定的功能旗標時，都可以在 Atlassian 主控台中建立和更新問題。 AWS 帳戶 AWS 區域 每個 Jira 問題都包含旗標名稱、應用程式 ID、組態設定檔 ID 和旗標值。更新、儲存和部署旗標變更後，Jira 會使用變更的詳細資訊更新現有問題。如需詳細資訊，請參閱與阿特拉西揚 Jira AWS AppConfig 整合。	2022 年 4 月 7 日

ARM64 (重力 2) 處理器的功能旗標和 Lambda 擴充支援正式推出	使用 AWS AppConfig 功能標誌，您可以開發新功能並將其部署到生產環境，同時向使用者隱藏該功能。首先，您可以將旗標加入 AWS AppConfig 為組態資料。一旦功能準備好發布，您可以更新標誌配置數據，而無需部署任何代碼。此功能可改善您 dev-ops 環境的安全性，因為您不需要部署新程式碼即可發行此功能。如需詳細資訊，請參閱建立功能旗標組態設定檔。 中功能旗標的一般可用性 AWS AppConfig 包括下列增強功能： • 控制台包括一個選項，用於將標誌指定為短期標誌。您可以篩選和排序短期旗標上的旗標清單。 • 對於在中使用功能旗標的客戶 AWS Lambda，新的 Lambda 擴充功能可讓您使用 HTTP 端點呼叫個別功能旗標。如需詳細資訊，請參閱從功能旗標組態擷取一或多個旗標。 此更新也提供針對 ARM64 (重力 on2) 處理器開發的 AWS Lambda 擴充功能的支援。如需詳細資訊，請參閱 AWS AppConfig Lambda 擴充功能的可用版本。	2022 年 3 月 15 日

GetConfiguration API 動作已被取代	`GetConfiguration` API 動作已取代。接收設定資料的呼叫應改用`StartConfigurationSession` 和`GetLatestConfiguration` API。如需這些 API 及其使用方式的詳細資訊，請參閱擷取設定。	2022 年 1 月 28 日
用於 AWS AppConfig Lambda 擴展的新區域 ARN	AWS AppConfig Lambda 擴充功能在新的亞太區域 (大阪) 區域提供。Amazon 資源名稱 (ARN) 需要在該地區創建一個 Lambda。如需有關亞太區域 (大阪) 區域 ARN 的詳細資訊，請參閱新增 AWS AppConfig Lambda 擴充功能。	2021 年 3 月 4 日
AWS AppConfig Lambda 擴展	如果您使用 AWS AppConfig 來管理 Lambda 函數的組態，建議您新增 AWS AppConfig Lambda 擴充功能。此擴充功能包含可簡化使用 AWS AppConfig 同時降低成本的最佳實務。降低成本，因為對 AWS AppConfig 服務的 API 呼叫次數減少，而且另外減少 Lambda 函數處理時間的成本。如需詳細資訊，請參閱與 Lambda 擴充功能AWS AppConfig 整合。	2020 年 10 月 8 日
新增章節	已新增提供設定指示的新區段 AWS AppConfig。如需詳細資訊，請參閱設定 AWS AppConfig。	2020 年 9 月 30 日

添加了命令行程序	本使用指南中的程序現在包括 AWS Command Line Interface (AWS CLI) 和 Windows PowerShell 工具的指令行步驟。如需詳細資訊，請參閱使用 AWS AppConfig。	2020 年 9 月 30 日
推出用 AWS AppConfig 戶指南	使用 AWS AppConfig的功能來建立 AWS Systems Manager、管理及快速部署應用程式組態。 AWS AppConfig 支援任何規模應用程式的受控部署，並包含內建的驗證檢查和監控。您可以搭 AWS AppConfig 配託管在 EC2 執行個體、容器 AWS Lambda、行動應用程式或 IoT 裝置上的應用程式搭配使用。	2020 年 7 月 31 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。