aws

Hooks 使用者指南

CloudFormation

CloudFormation: Hooks 使用者指南

Table of Contents

什麼是 CloudFormation 勾點？

CloudFormation 勾點功能可協助確保您的 CloudFormation 資源、堆疊和變更集符合組織的安全性、營運和成本最佳化最佳實務。CloudFormation Hooks 也可以確保您的 AWS 雲端控制 API 資源具有相同層級的合規。使用 CloudFormation Hooks，您可以提供程式碼，在佈建之前主動檢查 AWS 資源的組態。如果找到不合規的資源，則 CloudFormation 操作會失敗，並防止資源佈建或發出警告，並允許佈建操作繼續。

您可以使用勾點來強制執行各種需求和指導方針。例如，與安全相關的勾點可以驗證安全群組是否具有適用於 Amazon VPC 的傳入和傳出流量規則。成本相關的勾點可以限制開發環境只使用較小的 Amazon EC2 執行個體類型。專為資料可用性而設計的勾點可以強制執行 Amazon RDS 的自動備份。

勾點實作選項

CloudFormation 提供多種實作勾點的選項，讓您靈活地選擇最適合您需求的方法。

AWS Control Tower 主動控制

Control AWS Control Tower Catalog 提供標準化的主動控制，您可以實作為勾點。此方法可節省設定時間，並協助您根據整個組織的 AWS 最佳實務驗證資源組態，而無需撰寫程式碼。

防護規則

AWS CloudFormation Guard 是一種policy-as-code評估工具，提供用於撰寫 Hooks 自訂評估邏輯的網域特定語言。此方法可讓您使用 Guard 的宣告式語法來定義合規檢查，讓您輕鬆地建立和維護評估邏輯，而無需廣泛的程式設計知識。

Lambda 函式

您也可以使用 Lambda 函數實作勾點，讓您將 Lambda 的完整功能和彈性用於評估邏輯。您可以使用任何 Lambda 支援的執行時間語言，並視需要與其他 AWS 服務整合。

自訂勾點

對於進階使用案例，您可以使用 CloudFormation CLI 支援的程式設計語言撰寫自己的評估邏輯。此方法可為實作組織特定的控管要求提供最大的彈性。做為 CloudFormation 登錄檔中支援的延伸類型，您的自訂勾點可以公開和私密地分發和啟用。

建立和管理 CloudFormation 勾點

CloudFormation 勾點提供一種機制，可在允許堆疊建立、修改或刪除之前評估 CloudFormation 資源。此功能可協助您確保 CloudFormation 資源符合組織的安全性、營運和成本最佳化最佳實務。

若要建立勾點，您有四個選項。

- 做為勾點的主動控制 – 使用 Control AWS Control Tower Catalog 中的主動控制評估資源。
- Guard Hook – 使用 AWS CloudFormation Guard 規則評估資源。
- Lambda Hook – 將資源評估的請求轉送至 AWS Lambda 函數。
- Custom Hook – 使用您手動開發的自訂 Hook 處理常式。

Proactive controls as Hooks

若要從主動控制建立勾點，請遵循下列步驟：

1. 導覽至 CloudFormation 主控台並開始建立勾點。
2. 從控制目錄中選擇您希望勾點評估資源的特定控制項。

   這些控制項會在建立或更新指定的資源時自動套用。您的選擇會決定勾點將評估哪些資源類型。
3. 將勾點模式設定為警告使用者不合規或防止不合規的操作。
4. 設定選用篩選條件，依堆疊名稱或堆疊角色來包含或排除堆疊。
5. 完成組態後，請啟用勾點以開始強制執行。

Guard Hook

若要建立 Guard Hook，請遵循下列步驟：

1. 使用 Guard 網域特定語言 (DSL)，將資源評估邏輯撰寫為 Guard 政策規則。
2. 將 Guard 政策規則存放在 Amazon S3 儲存貯體中。
3. 導覽至 CloudFormation 主控台並開始建立 Guard Hook。
4. 提供您 Guard 規則的 Amazon S3 路徑。
5. 選擇勾點將評估的特定目標類型。

   - CloudFormation 資源 (RESOURCE)

- 整個堆疊範本 (STACK)
- 變更集 (CHANGE_SET)
- Cloud Control API 資源 (CLOUD_CONTROL)

6. 選擇將調用勾點的部署動作 （建立、更新、刪除）。
7. 選擇勾點在評估失敗時如何回應。
8. 設定選用篩選條件以指定掛鉤應評估的資源類型
9. 設定選用篩選條件，依堆疊名稱或堆疊角色來包含或排除堆疊。
10. 完成組態後，請啟用勾點以開始強制執行。

Lambda Hook

若要建立 Lambda 勾點，請遵循下列步驟：

1. 將您的資源評估邏輯撰寫為 Lambda 函數。
2. 導覽至 CloudFormation 主控台並開始建立 Lambda Hook。
3. 為您的 Lambda 函數提供 Amazon Resource Name (ARN)。
4. 選擇勾點將評估的特定目標類型。

   - CloudFormation 資源 (RESOURCE)
   - 整個堆疊範本 (STACK)
   - 變更集 (CHANGE_SET)
   - Cloud Control API 資源 (CLOUD_CONTROL)

5. 選擇將調用勾點的部署動作 （建立、更新、刪除）。
6. 選擇勾點在評估失敗時如何回應。
7. 設定選用篩選條件以指定掛鉤應評估的資源類型
8. 設定選用篩選條件，依堆疊名稱或堆疊角色來包含或排除堆疊。
9. 完成組態後，請啟用勾點以開始強制執行。

Custom Hook

自訂勾點是您使用 CloudFormation 命令列界面 (CFN-CLI) 在 CloudFormation 登錄檔中註冊的延伸模組。

若要建立自訂勾點，請遵循這些主要步驟：

1. 啟動專案 – 產生開發自訂勾點所需的檔案。
2. 勾點模型 – 撰寫定義勾點的結構描述，以及指定可叫用勾點之操作的處理常式。
3. 註冊並啟用勾點 – 建立勾點後，您需要在要使用勾點的帳戶和區域中註冊它，這會啟用它。

下列主題提供建立和管理勾點的詳細資訊。

主題

CloudFormation 勾點概念

下列術語和概念是了解和使用 CloudFormation Hooks 的核心。

勾點

勾點包含在 CloudFormation 建立、更新或刪除堆疊或特定資源之前立即調用的程式碼。它也可以在建立變更集操作期間叫用。勾點可以檢查 CloudFormation 即將佈建的範本、資源或變更集。此外，您可以在 Cloud Control API 建立、更新或刪除特定資源之前立即叫用勾點。

如果勾點識別不符合勾點邏輯中定義之組織準則的任何組態，則您可以選擇WARN使用者或 FAIL，以防止 CloudFormation 佈建資源。

勾點具有下列特性：

- 主動驗證 – 在建立、更新或刪除不合規資源之前，透過識別這些資源來降低風險、營運開銷和成本。
- 自動強制執行 – 在 中提供強制執行 AWS 帳戶 ，以防止 CloudFormation 佈建不合規的資源。

失敗模式

您的勾點邏輯可以傳回成功或失敗。成功回應將允許操作繼續。不合規資源的失敗可能會導致下列情況：

- FAIL – 停止佈建操作。
- WARN – 允許佈建繼續出現警告訊息。

在WARN模式下建立勾點是監控勾點行為的有效方式，而不會影響堆疊操作。首先，在 WARN 模式中啟用勾點，以了解哪些操作會受到影響。評估潛在影響之後，您可以將勾點切換到 FAIL 模式，以開始防止不合規的操作。

掛鉤目標

勾點目標指定勾點將評估的操作。這些可以是 上的操作：

- CloudFormation 支援的資源 (RESOURCE)
- 堆疊範本 (STACK)
- 變更集 (CHANGE_SET)
- Cloud Control API (CLOUD_CONTROL) 支援的資源

您可以定義一或多個目標，指定勾點將評估的最廣泛操作。例如，您可以撰寫 Hook 目標RESOURCE以鎖定所有 AWS 資源STACK，並鎖定所有堆疊範本。

目標動作

目標動作會定義將叫用勾點的特定動作 CREATE(UPDATE、 或 DELETE)。對於 RESOURCE、 STACK和CLOUD_CONTROL目標，所有目標動作都適用。對於CHANGE_SET目標，只有 CREATE動作適用。

註釋

GetHookResult 回應可以傳回註釋，提供每個評估資源的詳細合規檢查結果和修補指導。如需 API 註釋結構的詳細資訊，請參閱 AWS CloudFormation API 參考中的註釋。如需檢視這些驗證結果的說明，請參閱 檢視 CloudFormation 勾點的調用結果。

您可以在設定勾點時指定自己的 KMS 金鑰，視需要加密敏感合規資訊的註釋。如需詳細資訊，請參閱勾點組態結構描述語法參考。如需設定您為 Hooks 指定 KMS 金鑰時所需金鑰政策的資訊，請參閱 AWS KMS 加密靜態 CloudFormation Hooks 結果的金鑰政策和許可。

> **Important**
>
> 請注意，指定客戶受管金鑰`KmsKeyId`的選項目前只有在您使用 AWS CLI 來設定勾點時才能使用。

勾點處理常式

對於自訂勾點，這是處理評估的程式碼。它與目標叫用點和目標動作相關聯，該動作會標記勾點的確切執行點。您可以撰寫處理常式，以託管這些特定點的邏輯。例如，具有`PRE`目標動作`CREATE`的目標調用點會成為`preCreate`勾點處理常式。當相符的目標叫用點和服務執行相關聯的目標動作時，Hook 處理常式內的程式碼會執行。

有效值：`(preCreate | preUpdate | preDelete)`

> **Important**
>
> 導致 狀態的堆疊操作`UpdateCleanup`不會叫用勾點。例如，在以下兩個案例中，不會叫用 Hook 的`preDelete`處理常式：
>
> - 從範本移除一個資源後，堆疊會更新。
> - 會刪除具有替代更新類型的資源。

逾時和重試限制

勾點每次調用都有 30 秒的逾時限制，且限制為 3 次重試。如果調用超過逾時，我們會傳回錯誤訊息，指出勾點執行逾時。第三次重試後，CloudFormation 會將勾點執行標記為失敗。

AWS Control Tower 做為勾點的主動控制

Control AWS Control Tower Catalog 提供預先建置的合規規則（主動控制），您可以將這些規則實作為勾點。此方法可節省設定時間，並協助您根據整個組織的 AWS 最佳實務驗證資源組態，而無需撰寫程式碼。

主動控制會在部署之前評估 AWS 資源，防止建立不合規的資源，而不是稍後偵測到問題。他們會根據既定的安全、操作和控管標準來檢查組態。

若要開始使用，只需在所需的帳戶和區域中啟用主動控制型勾點。這些勾點接著會評估特定目標類型，以確保符合您選取的控制項。

如需可用主動控制的詳細資訊，請參閱 AWS Control Tower Control Catalog。

主題

AWS CLI 使用 Hooks 的 命令

使用主動式控制型勾點的 AWS CLI 命令包括：

- activate-type 啟動主動控制型勾點的啟用程序。
- set-type-configuration 指定要套用至帳戶中主動控制型勾點的控制項。
- list-types 列出您帳戶中的勾點。
- describe-type 傳回特定 Hook 或特定 Hook 版本的詳細資訊，包括目前的組態資料。
- deactivate-type 從您的帳戶中移除先前啟用的勾點。

在您的帳戶中啟用主動控制型勾點

下列主題說明如何在您的 帳戶中啟用主動控制型勾點，使其可在其啟用的帳戶和區域中使用。

> **Important**
>
> 在繼續之前，請確認您具有使用勾點所需的許可，並從 CloudFormation 主控台檢視主動控制。如需詳細資訊，請參閱授予 CloudFormation Hooks 的 IAM 許可。

主題

啟用主動控制型勾點（主控台）

啟用主動控制型勾點，以便在您的帳戶中使用

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇您要建立連接所在的 AWS 區域 。
3. 在左側導覽窗格中，選擇勾點。
4. 在勾點頁面上，選擇建立勾點，然後選擇使用控制目錄。
5. 在選取控制項頁面上，針對主動控制項，選取要使用的一或多個主動控制項。

   這些控制項會在建立或更新指定的資源時自動套用。您的選擇會決定勾點將評估哪些資源類型。
6. 選擇下一步。
7. 針對勾點名稱，選擇下列其中一個選項：

   - 提供將在 之後新增的簡短描述性名稱`Private::Controls::`。例如，如果您輸入 *`MyTestHook`*，則完整的勾點名稱會變成 `Private::Controls::`*`MyTestHook`*。
   - 使用此格式提供完整的勾點名稱（也稱為別名）：*`Provider`*`::`*`ServiceName`*`::`*`HookName`*。
8. 針對勾點模式，選擇當控制項評估失敗時勾點如何回應：

   - 警告 — 向使用者發出警告，但允許動作繼續。這適用於非關鍵驗證或資訊檢查。
   - 失敗 — 防止動作繼續。這有助於強制執行嚴格的合規或安全政策。
9. 選擇下一步。
10. （選用）對於勾點篩選條件，請執行下列動作：

    a. 針對篩選條件，選擇套用堆疊名稱和堆疊角色篩選條件的邏輯：

       - 所有堆疊名稱和堆疊角色 – 只有在所有指定的篩選條件相符時，才會叫用勾點。
       - 任何堆疊名稱和堆疊角色 – 如果至少一個指定的篩選條件相符，則會叫用勾點。

    b. 對於堆疊名稱，請在勾點調用中包含或排除特定堆疊。

       - 針對包含，指定要包含的堆疊名稱。當您有一小組想要鎖定的特定堆疊時，請使用此選項。只有此清單中指定的堆疊會叫用勾點。
       - 針對排除，指定要排除的堆疊名稱。當您想要在大多數堆疊上叫用勾點，但排除一些特定堆疊時，請使用此選項。除了此處列出的堆疊之外，所有堆疊都會叫用勾點。

    c. 對於堆疊角色，請根據特定堆疊相關聯的 IAM 角色，從勾點調用中包含或排除特定堆疊。

- 針對包含，指定一或多個 IAM 角色 ARNs 至與這些角色相關聯的目標堆疊。只有這些角色啟動的堆疊操作才會叫用勾點。
- 針對排除，為您要排除的堆疊指定一或多個 IAM 角色 ARNs。所有堆疊都會叫用勾點，但由指定角色啟動的堆疊除外。

11. 選擇下一步。
12. 在檢閱和啟用頁面上，檢閱您的選擇。選擇編輯以對相關區段進行變更。
13. 當您準備好繼續時，請選擇啟用勾點。

啟用主動控制型勾點 (AWS CLI)

在繼續之前，請確認您已識別要與此勾點搭配使用的主動控制。如需詳細資訊，請參閱 AWS Control Tower Control Catalog。

啟用主動控制型勾點以用於您的帳戶 (AWS CLI)

1. 若要開始啟用勾點，請使用下列activate-type命令，將預留位置取代為您的特定值。

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::ControlTower::Hook  \
  --publisher-id aws-hooks \
  --type-name-alias MyOrg::Security::ComplianceHook \
  --region us-west-2
```

2. 若要完成啟用勾點，您必須使用 JSON 組態檔案進行設定。

   使用 cat命令建立具有下列結構的 JSON 檔案。如需詳細資訊，請參閱勾點組態結構描述語法參考。

   下列範例會設定在 CREATE和 UPDATE操作期間調用特定 IAM、Amazon EC2 和 Amazon S3 資源的勾點。它套用三個主動控制 (CT.IAM.PR.5、CT.EC2.PR.17、CT.S3.PR.12)，以根據合規標準驗證這些資源。勾點會在 WARN 模式下運作，這表示它會使用警告標記不合規的資源，但不會封鎖部署。

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
```

```
      "TargetOperations": ["RESOURCE"],
      "FailureMode": "WARN",
      "Properties": {
        "ControlsToApply": "CT.IAM.PR.5,CT.EC2.PR.17,CT.S3.PR.12"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE"
        ]
      }
    }
  }
}
```

- HookInvocationStatus：設定為 ENABLED 以啟用勾點。
- TargetOperations：設定為 ，RESOURCE因為這是主動控制型勾點唯一支援的值。
- FailureMode：設為 FAIL 或 WARN。
- ControlsToApply：指定要使用的主動控制控制 IDs。如需詳細資訊，請參閱 AWS Control Tower Control Catalog。
- （選用）TargetFilters：對於 Actions，您可以指定 CREATE或 UPDATE或兩者（預設），以控制何時叫用勾點。CREATE 僅指定 會將勾點限制為僅限 CREATE 操作。其他TargetFilters屬性沒有效果。

3. 使用下列set-type-configuration命令以及您建立的 JSON 檔案來套用組態。將預留位置取代為您的特定值。

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-ComplianceHook" \
  --region us-west-2
```

刪除您帳戶中的主動控制型勾點

當您不再需要已啟用的主動式控制型勾點時，請使用下列程序在帳戶中將其刪除。

若要暫時停用勾點，而不是將其刪除，請參閱 停用和啟用 CloudFormation 勾點。

主題

- 在帳戶中刪除主動控制型勾點（主控台）
- 在帳戶中刪除主動控制型勾點 (AWS CLI)

在帳戶中刪除主動控制型勾點（主控台）

在帳戶中刪除主動控制型勾點

1. 登入 AWS 管理主控台，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 勾點所在的 。
3. 從導覽窗格中，選擇勾點。
4. 在勾點頁面上，尋找您要刪除的主動控制型勾點。
5. 選取勾點旁的核取方塊，然後選擇刪除。
6. 出現確認提示時，輸入勾點名稱以確認刪除指定的勾點，然後選擇刪除。

在帳戶中刪除主動控制型勾點 (AWS CLI)

> **Note**
>
> 您必須先停用勾點，才能刪除勾點。如需詳細資訊，請參閱在帳戶中停用和啟用勾點 (AWS CLI)。

使用下列deactivate-type命令來停用勾點，該勾點會從您的帳戶中移除它。將預留位置取代為您的特定值。

```
aws cloudformation deactivate-type \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-ComplianceHook" \
  --region us-west-2
```

Guard 勾點

若要在帳戶中使用 AWS CloudFormation Guard 勾點，您必須為要使用該勾點的帳戶和區域啟用 勾點。啟用勾點可在啟用勾點的帳戶和區域中的堆疊操作中使用。

當您啟用 Guard Hook 時，CloudFormation 會在您帳戶的登錄檔中為已啟用的 Hook 建立項目，做為私有 Hook。這可讓您設定勾點包含的任何組態屬性。組態屬性會定義如何為指定 AWS 帳戶 和 區域設定勾點。

主題

AWS CLI 使用 Guard Hooks 的 命令

使用 Guard Hooks 的 AWS CLI 命令包括：

- activate-type 啟動 Guard Hook 的啟用程序。
- set-type-configuration 指定帳戶中勾點的組態資料。
- list-types 列出您帳戶中的勾點。
- describe-type 傳回特定 Hook 或特定 Hook 版本的詳細資訊，包括目前的組態資料。
- deactivate-type 從您的帳戶中移除先前啟用的勾點。

撰寫 Guard 規則來評估 Guard Hooks 的資源

AWS CloudFormation Guard 是一種開放原始碼和一般用途網域特定語言 (DSL)，可用來撰寫 policy-as-code。本主題說明如何使用 Guard 撰寫可在 Guard Hook 中執行的範例規則，以自動評估 CloudFormation 和 AWS 雲端控制 API 操作。視您的 Guard Hook 執行時間而定，它也會專注於您的 Guard 規則可用的不同輸入類型。Guard Hook 可設定為在下列操作類型期間執行：

- 資源操作
- 堆疊操作
- 變更集操作

如需撰寫 Guard 規則的詳細資訊，請參閱撰寫 AWS CloudFormation Guard 規則

主題

資源操作 Guard 規則

每當您建立、更新或刪除資源時，即視為資源操作。例如，如果您執行更新建立新資源的 CloudFormation 堆疊，表示您已完成資源操作。當您使用 Cloud Control API 建立、更新或刪除資源時，這也被視為資源操作。您可以在 Guard Hook 的`TargetOperations`組態中，將 Guard Hook 設定為目標`RESOURCE`和`CLOUD_CONTROL`操作。當您的 Guard Hook 評估資源操作時，Guard 引擎會評估資源輸入。

主題

Guard 資源輸入語法

Guard 資源輸入是可供 Guard 規則評估的資料。

以下是資源輸入的範例形狀：

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: String
  TargetType: RESOURCE
  TargetLogicalId: String
  ChangeSetId: String
Resources:
  {ResourceLogicalID}:
    ResourceType: {ResourceType}
    ResourceProperties:
```

```
        {ResourceProperties}
Previous:
  ResourceLogicalID:
    ResourceType: {ResourceType}
    ResourceProperties:
        {PreviousResourceProperties}
```

HookContext

AWSAccountID

AWS 帳戶 包含要評估之資源的 ID。

StackId

屬於資源操作一部分的 CloudFormation 堆疊的堆疊 ID。如果發起人是雲端控制 API，則為空白。

HookTypeName

正在執行的勾點名稱。

HookTypeVersion

正在執行的 Hook 版本。

InvocationPoint

佈建邏輯中 Hook 執行的確切點。

有效值：(CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

TargetName

正在評估的目標類型，例如 AWS::S3::Bucket。

TargetType

正在評估的目標類型，例如 AWS::S3::Bucket。對於使用 Cloud Control API 佈建的資源，此值將為 RESOURCE。

TargetLogicalId

正在評估TargetLogicalId的資源的 。如果 Hook 的原始伺服器是 CloudFormation，這將是資源的邏輯 ID （也稱為邏輯名稱）。如果勾點的原始伺服器是雲端控制 API，這會是建構值。

ChangeSetId

為造成勾點調用而執行的變更集 ID。如果資源變更是由 Cloud Control API 或 create-stack、 update-stack或 delete-stack操作啟動，則此值為空。

Resources

ResourceLogicalID

當 CloudFormation 啟動操作時， ResourceLogicalID是 CloudFormation 範本中資源的邏輯 ID。

操作由 Cloud Control API 啟動時， ResourceLogicalID是資源類型、名稱、操作 ID 和請求 ID 的組合。

ResourceType

資源的類型名稱（範例：AWS::S3::Bucket)。

ResourceProperties

正在修改之資源的提議屬性。當 Guard Hook 針對 CloudFormation 資源執行時，任何函數、參數和轉換都會完全解析。如果正在刪除資源，則此值將為空。

Previous

ResourceLogicalID

當 CloudFormation 啟動操作時， ResourceLogicalID是 CloudFormation 範本中資源的邏輯 ID。

操作由 Cloud Control API 啟動時， ResourceLogicalID是資源類型、名稱、操作 ID 和請求 ID 的組合。

ResourceType

資源的類型名稱（範例：AWS::S3::Bucket)。

ResourceProperties

目前與要修改之資源相關聯的屬性。如果正在刪除資源，則此值將為空。

Guard 資源操作輸入範例

下列範例輸入顯示將接收要更新之AWS::S3::Bucket資源定義的 Guard Hook。這是可供 Guard 評估的資料。

```
HookContext:
  AwsAccountId: "123456789012"
  StackId: "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000"
  HookTypeName: org::s3policy::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: AWS::S3::Bucket
  TargetType: RESOURCE
  TargetLogicalId: MyS3Bucket
  ChangeSetId: ""
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: true
Previous:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: false
```

若要查看資源類型可用的所有屬性，請參閱 AWS::S3::Bucket。

資源變更的防護規則

當 Guard Hook 評估資源變更時，首先會下載使用 Hook 設定的所有規則。然後，這些規則會根據資源輸入進行評估。如果任何規則的評估失敗，勾點將會失敗。如果沒有失敗，勾點將會通過。

下列範例是 Guard 規則，評估 `ObjectLockEnabled` 屬性是否`true`適用於任何`AWS::S3::Bucket`資源類型。

```
let s3_buckets_default_lock_enabled = Resources.*[ Type == 'AWS::S3::Bucket']

rule S3_BUCKET_DEFAULT_LOCK_ENABLED when %s3_buckets_default_lock_enabled !empty {
  %s3_buckets_default_lock_enabled.Properties.ObjectLockEnabled exists
  %s3_buckets_default_lock_enabled.Properties.ObjectLockEnabled == true
  <<
    Violation: S3 Bucket ObjectLockEnabled must be set to true.
    Fix: Set the S3 property ObjectLockEnabled parameter to true.
  >>
```

```
}
```

當此規則針對下列輸入執行時，將會失敗，因為 ObjectLockEnabled 屬性未設定為 true。

```
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: false
```

當此規則針對下列輸入執行時，將會傳遞，因為 ObjectLockEnabled 設定為 true。

```
Resources:
  MyS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      ObjectLockEnabled: true
```

當勾點失敗時，失敗的規則將傳播回 CloudFormation 或 Cloud Control API。如果已為 Guard Hook 設定記錄儲存貯體，則會在該處提供額外的規則意見回饋。此額外的意見回饋包括 Violation和 Fix資訊。

堆疊操作 Guard 規則

建立、更新或刪除 CloudFormation 堆疊時，您可以將 Guard Hook 設定為從評估新範本開始，並可能封鎖堆疊操作以繼續。您可以在 Guard Hook 的TargetOperations組態中將 Guard Hook 設定為目標STACK操作。

主題

- Guard 堆疊輸入語法
- Guard 堆疊操作輸入範例
- 堆疊變更的防護規則

Guard 堆疊輸入語法

Guard 堆疊操作的輸入提供整個 CloudFormation 範本，供您的 Guard 規則評估。

以下是堆疊輸入的範例形狀：

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: String
  TargetType:STACK
  ChangeSetId: String
{Proposed CloudFormation Template}
Previous:
    {CloudFormation Template}
```

HookContext

AWSAccountID

AWS 帳戶 包含 資源的 ID。

StackId

屬於堆疊操作一部分的 CloudFormation 堆疊的堆疊 ID。

HookTypeName

正在執行的勾點名稱。

HookTypeVersion

正在執行的 Hook 版本。

InvocationPoint

佈建邏輯中 Hook 執行的確切點。

有效值：(CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

TargetName

正在評估的堆疊名稱。

TargetType

作為堆疊層級掛鉤執行STACK時，此值將為 。

`ChangeSetId`

為造成勾點調用而執行的變更集 ID。如果堆疊操作是由 `create-stack`、 或 `delete-stack`操作啟動`update-stack`，則此值為空。

`Proposed CloudFormation Template`

傳遞給 CloudFormation `create-stack`或 `update-stack`操作的完整 CloudFormation 範本值。這包括 `Resources`、 `Outputs`和 等物件`Properties`。根據提供給 CloudFormation 的內容，它可以是 JSON 或 YAML 字串。

在 `delete-stack` 操作中，此值將為空。

`Previous`

上次成功部署的 CloudFormation 範本。如果正在建立或刪除堆疊，則此值為空。

在 `delete-stack` 操作中，此值將為空。

> **Note**
>
> 提供的範本是傳遞至 `create`或 `update`堆疊操作的內容。刪除堆疊時，不會提供任何範本值。

Guard 堆疊操作輸入範例

下列範例輸入顯示將接收完整範本和先前部署範本的 Guard Hook。此範例中的範本使用 JSON 格式。

```
HookContext:
  AwsAccountId: 123456789012
  StackId: "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000"
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: MyStack
  TargetType: CHANGE_SET
  TargetLogicalId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
  ChangeSetId: arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
Resources: {
```

```
   "S3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
           "BucketEncryption": {
               "ServerSideEncryptionConfiguration": [
                {"ServerSideEncryptionByDefault":
                    {"SSEAlgorithm": "aws:kms",
                      "KMSMasterKeyID": "KMS-KEY-ARN" },
                      "BucketKeyEnabled": true }
                ]
           }
        }
}
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {}
        }
    }
}
```

堆疊變更的防護規則

當 Guard Hook 評估堆疊變更時，一開始會下載使用 Hook 設定的所有規則。然後，這些規則會根據資源輸入進行評估。如果任何規則的評估失敗，勾點將會失敗。如果沒有失敗，勾點將會通過。

下列範例是 Guard 規則，評估是否有任何`AWS::S3::Bucket`資源類型包含名為 的屬性`BucketEncryption`，並將 `SSEAlgorithm`設定為 `aws:kms`或 `AES256`。

```
let s3_buckets_s3_default_encryption = Resources.*[ Type == 'AWS::S3::Bucket']

rule S3_DEFAULT_ENCRYPTION_KMS when %s3_buckets_s3_default_encryption !empty {
  %s3_buckets_s3_default_encryption.Properties.BucketEncryption exists

 %s3_buckets_s3_default_encryption.Properties.BucketEncryption.ServerSideEncryptionConfiguratio
 in ["aws:kms","AES256"]
  <<
    Violation: S3 Bucket default encryption must be set.
    Fix: Set the S3 Bucket property
 BucketEncryption.ServerSideEncryptionConfiguration.ServerSideEncryptionByDefault.SSEAlgorithm
 to either "aws:kms" or "AES256"
```

```
  >>
}
```

當規則針對下列範本執行時，它會 `fail`。

```
AWSTemplateFormatVersion: 2010-09-09
Description: S3 bucket without default encryption
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
```

當規則針對下列範本執行時，它會 `pass`。

```
AWSTemplateFormatVersion: 2010-09-09
Description: S3 bucket with default encryption using SSE-KMS with an S3 Bucket Key
Resources:
  EncryptedS3Bucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub 'encryptedbucket-${AWS::Region}-${AWS::AccountId}'
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: KMS-KEY-ARN
            BucketKeyEnabled: true
```

變更集操作 Guard 規則

建立 CloudFormation 變更集時，您可以設定 Guard Hook 來評估範本和變更集中提議的變更，以封鎖變更集執行。

主題

- 防護變更集輸入語法
- 範例 Guard 變更集操作輸入
- 變更集操作的防護規則

防護變更集輸入語法

Guard 變更集輸入是可供 Guard 規則評估的資料。

以下是變更集輸入的範例形狀：

```
HookContext:
  AWSAccountID: String
  StackId: String
  HookTypeName: String
  HookTypeVersion: String
  InvocationPoint: [CREATE_PRE_PROVISION, UPDATE_PRE_PROVISION, DELETE_PRE_PROVISION]
  TargetName: CHANGE_SET
  TargetType:CHANGE_SET
  TargetLogicalId:ChangeSet ID
  ChangeSetId: String
{Proposed CloudFormation Template}
Previous:
  {CloudFormation Template}
Changes: [{ResourceChange}]
```

ResourceChange 模型語法為：

```
logicalResourceId: String
resourceType: String
##: CREATE, UPDATE, DELETE
lineNumber: Number
beforeContext: JSON String
afterContext: JSON String
```

HookContext

AWSAccountID

AWS 帳戶 包含 資源的 ID。

StackId

屬於堆疊操作一部分的 CloudFormation 堆疊的堆疊 ID。

HookTypeName

正在執行的勾點名稱。

HookTypeVersion

正在執行的 Hook 版本。

InvocationPoint

佈建邏輯中 Hook 執行的確切點。

有效值：(CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

TargetName

正在評估的堆疊名稱。

TargetType

作為變更集層級掛鉤執行CHANGE_SET時，此值將為 。

TargetLogicalId

此值將是變更集的 ARN。

ChangeSetId

為造成勾點調用而執行的變更集 ID。如果堆疊操作是由 create-stack、 或 delete-stack操作啟動update-stack，則此值為空。

Proposed CloudFormation Template

提供給 create-change-set 操作的完整 CloudFormation 範本。根據提供給 CloudFormation 的內容，它可以是 JSON 或 YAML 字串。

Previous

上次成功部署的 CloudFormation 範本。如果正在建立或刪除堆疊，則此值為空。

Changes

Changes 模型。這會列出資源變更。

變更

logicalResourceId

已變更資源的邏輯資源名稱。

resourceType

要變更的資源類型。

動作

在資源上執行的操作類型。

有效值：(CREATE | UPDATE | DELETE)

lineNumber

範本中與變更相關聯的行號。

beforeContext

變更前資源屬性的 JSON 字串：

```
{"properties": {"property1": "value"}}
```

afterContext

變更後資源屬性的 JSON 字串：

```
{"properties": {"property1": "new value"}}
```

範例 Guard 變更集操作輸入

下列範例輸入顯示將接收完整範本的 Guard Hook、先前部署的範本，以及資源變更清單。此範例中的範本使用 JSON 格式。

```
HookContext:
  AwsAccountId: "00000000"
  StackId: MyStack
  HookTypeName: org::templatechecker::hook
  HookTypeVersion: "00001"
  InvocationPoint: UPDATE_PRE_PROVISION
  TargetName: my-example-stack
  TargetType:STACK
  TargetLogicalId: arn...:changeSet/change-set
  ChangeSetId: ""
Resources: {
    "S3Bucket": {
       "Type": "AWS::S3::Bucket",
       "Properties": {
           "BucketName": "amzn-s3-demo-bucket",
           "VersioningConfiguration":{
```

```
              "Status": "Enabled"
            }
         }
    }
Previous: {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket",
                "VersioningConfiguration":{
                  "Status": "Suspended"
                }
            }
        }
    }
}
Changes: [
  {
    "logicalResourceId": "S3Bucket",
    "resourceType": "AWS::S3::Bucket",
    "action": "UPDATE",
    "lineNumber": 5,
    "beforeContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":
\"Suspended\"}}}",
    "afterContext": "{\"Properties\":{\"VersioningConfiguration\":{\"Status\":\"Enabled
\"}}}"
  }
]
```

變更集操作的防護規則

下列範例是 Guard 規則，可評估 Amazon S3 儲存貯體的變更，並確保 `VersionConfiguration` 不會停用。

```
let s3_buckets_changing = Changes[resourceType == 'AWS::S3::Bucket']

rule S3_VERSIONING_STAY_ENABLED when %s3_buckets_changing !empty {
    let afterContext = json_parse(%s3_buckets_changing.afterContext)
    when %afterContext.Properties.VersioningConfiguration.Status !empty {
        %afterContext.Properties.VersioningConfiguration.Status == 'Enabled'
```

```
    }
}
```

準備建立 Guard Hook

建立 Guard Hook 之前，您必須完成下列先決條件：

- 您必須已建立 Guard 規則。如需更多資訊，請參閱勾點的寫入防護規則。
- 建立勾點的使用者或角色必須具有足夠的許可，才能啟用勾點。如需詳細資訊，請參閱授予 CloudFormation Hooks 的 IAM 許可。
- 若要使用 AWS CLI 或 開發套件建立 Guard Hook，您必須手動建立具有 IAM 許可和信任政策的執行角色，以允許 CloudFormation 叫用 Guard Hook。

建立 Guard Hook 的執行角色

Hook 會使用 執行角色來取得在 中叫用該 Hook 所需的許可 AWS 帳戶。

如果您從 建立 Guard Hook，則可以自動建立此角色 AWS 管理主控台；否則，您必須自行建立此角色。

下一節說明如何設定建立 Guard Hook 的許可。

所需的許可

遵循《IAM 使用者指南》中的使用自訂信任政策建立角色的指引，建立具有自訂信任政策的角色。

然後，完成下列步驟以設定您的許可：

1. 將下列最低權限政策連接至您要用來建立 Guard Hook 的 IAM 角色。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
```

```
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::my-guard-output-bucket/*",
        "arn:aws:s3:::my-guard-rules-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-guard-output-bucket/*"
      ]
    }
  ]
}
```

2. 將信任政策新增至角色，授予您的 Hook 擔任角色的許可。以下顯示您可以使用的範例信任政策。

   JSON

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

在您的帳戶中啟用 Guard Hook

下列主題說明如何在您的 帳戶中啟用 Guard Hook，使其可在其啟用的帳戶和區域中使用。

主題

啟用 Guard Hook （主控台）

啟用 Guard Hook 以在您的帳戶中使用

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇您要建立連接所在的 AWS 區域 。
3. 在左側導覽窗格中，選擇勾點。
4. 在勾點頁面上，選擇建立勾點，然後選擇使用 Guard。
5. 如果您尚未建立任何 Guard 規則，請建立您的 Guard 規則，將其存放在 Amazon S3 中，然後返回此程序。請參閱 中的範例規則撰寫 Guard 規則來評估 Guard Hooks 的資源以開始使用。

   如果您已建立 Guard 規則並將其存放在 S3 中，請繼續下一個步驟。

   > **Note**
   >
   > 存放在 S3 中的物件必須具有下列其中一個副檔名：`.guard`、 `.zip`或 `.tar.gz`。

6. 對於 Guard Hook 來源，請將您的 Guard 規則存放在 S3 中，請執行下列動作：
   - 對於 S3 URI，指定規則檔案的 S3 路徑，或使用瀏覽 S3 按鈕開啟對話方塊來瀏覽並選取 S3 物件。
   - （選用）對於物件版本，如果您的 S3 儲存貯體已啟用版本控制，您可以選取 S3 物件的特定版本。

   每次叫用勾點時，Guard Hook 都會從 S3 下載您的規則。為了防止意外變更或刪除，我們建議您在設定 Guard Hook 時使用 版本。
7. （選用）針對 Guard 輸出報告的 S3 儲存貯體，指定要存放 Guard 輸出報告的 S3 儲存貯體。此報告包含 Guard 規則驗證的結果。

   若要設定輸出報告目的地，請選擇下列其中一個選項：

- 選取使用與 Guard 規則存放相同的儲存貯體核取方塊，以使用 Guard 規則所在的相同儲存貯體。
- 選擇不同的 S3 儲存貯體名稱來存放 Guard 輸出報告。

8. （選用）展開 Guard 規則輸入參數，然後在 S3 中存放 Guard 規則輸入參數下提供以下資訊：
   - 對於 S3 URI，指定參數檔案的 S3 路徑，或使用瀏覽 S3 按鈕開啟對話方塊來瀏覽並選取 S3 物件。
   - （選用）對於物件版本，如果您的 S3 儲存貯體已啟用版本控制，您可以選取 S3 物件的特定版本。
9. 選擇下一步。
10. 針對勾點名稱，選擇下列其中一個選項：
    - 提供將在 之後新增的簡短描述性名稱`Private::Guard::`。例如，如果您輸入 *`MyTestHook`*，則完整的勾點名稱會變成 `Private::Guard::`*`MyTestHook`*。
    - 使用此格式提供完整的勾點名稱（也稱為別名）：*`Provider`*`::`*`ServiceName`*`::`*`HookName`*
11. 對於勾點目標，選擇要評估的內容：
    - 堆疊 — 在使用者建立、更新或刪除堆疊時評估堆疊範本。
    - 資源 — 評估使用者更新堆疊時的個別資源變更。
    - 變更集 — 評估使用者建立變更集時的計劃更新。
    - 雲端控制 API — 評估 Cloud Control API 啟動的建立、更新或刪除操作。
12. 針對動作，選擇哪些動作（建立、更新、刪除）會叫用您的勾點。
13. 對於勾點模式，選擇當規則評估失敗時勾點如何回應：
    - 警告 — 向使用者發出警告，但允許動作繼續。這適用於非關鍵驗證或資訊檢查。
    - 失敗 — 防止動作繼續。這有助於強制執行嚴格的合規或安全政策。
14. 針對執行角色，選擇 Hook 從 S3 擷取您的 Guard 規則所擔任的 IAM 角色，並選擇性地撰寫詳細的 Guard 輸出報告。您可以允許 CloudFormation 為您自動建立執行角色，也可以指定您已建立的角色。
15. 選擇下一步。
16. （選用）對於勾點篩選條件，請執行下列動作：
    a. 針對資源篩選條件，指定哪些資源類型可以叫用勾點。這可確保僅針對相關資源叫用勾點。
    b. 針對篩選條件，選擇套用堆疊名稱和堆疊角色篩選條件的邏輯：

- 所有堆疊名稱和堆疊角色 – 只有在所有指定的篩選條件相符時，才會叫用勾點。
- 任何堆疊名稱和堆疊角色 – 如果至少一個指定的篩選條件相符，則會叫用勾點。

> **Note**
>
> 對於雲端控制 API 操作，會忽略所有堆疊名稱和堆疊角色篩選條件。

c. 對於堆疊名稱，請在勾點調用中包含或排除特定堆疊。

- 針對包含，指定要包含的堆疊名稱。當您有一小組想要鎖定的特定堆疊時，請使用此選項。只有此清單中指定的堆疊才會叫用勾點。
- 針對排除，指定要排除的堆疊名稱。當您想要在大多數堆疊上叫用勾點，但排除幾個特定堆疊時，請使用此選項。除了此處列出的堆疊之外，所有堆疊都會叫用勾點。

d. 對於堆疊角色，請根據特定堆疊相關聯的 IAM 角色，從勾點調用中包含或排除特定堆疊。

- 針對包含，指定一或多個 IAM 角色 ARNs 至與這些角色相關聯的目標堆疊。只有這些角色啟動的堆疊操作才會叫用勾點。
- 針對排除，為您要排除的堆疊指定一或多個 IAM 角色 ARNs。所有堆疊都會叫用勾點，但由指定角色啟動的堆疊除外。

17. 選擇下一步。
18. 在檢閱和啟用頁面上，檢閱您的選擇。選擇編輯以對相關區段進行變更。
19. 當您準備好繼續時，請選擇啟用勾點。

啟用 Guard Hook (AWS CLI)

在繼續之前，請確認您已建立 Guard 規則和您將與此勾點搭配使用的執行角色。如需詳細資訊，請參閱撰寫 Guard 規則來評估 Guard Hooks 的資源及建立 Guard Hook 的執行角色。

啟用 Guard Hook 以用於您的帳戶 (AWS CLI)

1. 若要開始啟用勾點，請使用下列activate-type命令，將預留位置取代為您的特定值。此命令會授權 Hook 從您的 使用指定的執行角色 AWS 帳戶。

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::Hooks::GuardHook \
  --publisher-id aws-hooks \
```

```
  --type-name-alias Private::Guard::MyTestHook \
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --region us-west-2
```

2. 若要完成啟用勾點，您必須使用 JSON 組態檔案進行設定。

   使用 cat命令建立具有下列結構的 JSON 檔案。如需詳細資訊，請參閱勾點組態結構描述語法參考。

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE",
        "CHANGE_SET"
      ],
      "FailureMode": "WARN",
      "Properties": {
        "ruleLocation": "s3://amzn-s3-demo-bucket/MyGuardRules.guard",
        "logBucket": "amzn-s3-demo-logging-bucket"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

   - `HookInvocationStatus`：將 設定為 `ENABLED` 以啟用 勾點。
   - `TargetOperations`：指定勾點將評估的操作。
   - `FailureMode`：設為 `FAIL` 或 `WARN`。
   - `ruleLocation`：將 取代為儲存規則的 S3 URI。存放在 S3 中的物件必須具有下列其中一個副檔名：`.guard`、 `.zip`和 `.tar.gz`。
   - `logBucket`：（選用）指定 Guard JSON 報告的 S3 儲存貯體名稱。

- `TargetFilters`：指定將調用勾點的動作類型。

3. 使用下列set-type-configuration命令以及您建立的 JSON 檔案來套用組態。將預留位置取代為您的特定值。

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

相關資源

我們提供範本範例，您可以用來了解如何在 CloudFormation 堆疊範本中宣告 Guard Hook。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS::CloudFormation::GuardHook。

檢視您帳戶中 Guard Hooks 的日誌

當您啟用 Guard Hook 時，您可以將 Amazon S3 儲存貯體指定為 Hook 輸出報告的目的地。啟用後，Hook 會自動將 Guard 規則驗證的結果存放在指定的儲存貯體中。然後，您可以在 Amazon S3 主控台中檢視這些結果。

在 Amazon S3 主控台中檢視 Guard Hook 日誌

檢視 Guard Hook 輸出日誌檔案

1. 登入 https：//https://console.aws.amazon.com/s3/。
2. 在畫面上方的導覽列上，選擇 AWS 區域。
3. 選擇儲存貯體。
4. 選擇您為 Guard 輸出報告選取的儲存貯體。
5. 選擇所需的驗證輸出報告日誌檔案。
6. 選擇您要下載檔案或開啟檔案以檢視。

刪除您帳戶中的 Guard Hook

當您不再需要已啟用的 Guard Hook 時，請使用下列程序在帳戶中將其刪除。

若要暫時停用勾點，而不是將其刪除，請參閱 停用和啟用 CloudFormation 勾點。

主題

刪除您帳戶中的 Guard Hook （主控台）

刪除您帳戶中的 Guard Hook

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 勾點所在的 。
3. 從導覽窗格中，選擇勾點。
4. 在勾點頁面上，尋找您要刪除的 Guard Hook。
5. 選取勾點旁的核取方塊，然後選擇刪除。
6. 出現確認提示時，輸入勾點名稱以確認刪除指定的勾點，然後選擇刪除。

刪除您帳戶中的 Guard Hook (AWS CLI)

> **Note**
>
> 您必須先停用勾點，才能刪除勾點。如需詳細資訊，請參閱在帳戶中停用和啟用勾點 (AWS CLI)。

使用下列deactivate-type命令來停用勾點，該勾點會從您的帳戶中移除它。將預留位置取代為您的特定值。

```
aws cloudformation deactivate-type \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

Lambda 勾點

若要在帳戶中使用 AWS Lambda 勾點，您必須先為要使用該勾點的帳戶和區域啟用 勾點。啟用勾點可在啟用勾點的帳戶和區域中的堆疊操作中使用。

當您啟用 Lambda Hook 時，CloudFormation 會在您帳戶的登錄檔中為已啟用的 Hook 建立項目，做為私有 Hook。這可讓您設定勾點包含的任何組態屬性。組態屬性會定義如何為指定 AWS 帳戶 和 區域設定勾點。

主題

AWS CLI 使用 Lambda Hooks 的 命令

使用 Lambda Hooks 的 AWS CLI 命令包括：

- activate-type 啟動 Lambda Hook 的啟用程序。
- set-type-configuration 指定帳戶中勾點的組態資料。
- list-types 列出您帳戶中的勾點。
- describe-type 傳回特定 Hook 或特定 Hook 版本的詳細資訊，包括目前的組態資料。
- deactivate-type 從您的帳戶中移除先前啟用的勾點。

建立 Lambda 函數來評估 Lambda Hooks 的資源

CloudFormation Lambda Hooks 可讓您根據自己的自訂程式碼評估 CloudFormation 和 AWS 雲端控制 API 操作。您的勾點可以封鎖操作以繼續，或向發起人發出警告，並允許操作繼續。當您建立 Lambda Hook 時，您可以將其設定為攔截和評估下列 CloudFormation 操作：

- 資源操作
- 堆疊操作
- 變更集操作

主題

- 使用 Lambda Hooks 評估資源操作
- 使用 Lambda Hooks 評估堆疊操作
- 使用 Lambda Hooks 評估變更集操作

開發 Lambda 勾點

當 Hooks 調用 Lambda 時，Lambda 最多會等待 30 秒來評估輸入。Lambda 將傳回 JSON 回應，指出勾點是否成功或失敗。

主題

- 請求輸入
- 回應輸入
- 範例

請求輸入

傳遞至 Lambda 函數的輸入取決於勾點目標操作（範例：堆疊、資源或變更集）。

回應輸入

為了在請求成功或失敗時與 Hooks 通訊，您的 Lambda 函數需要傳回 JSON 回應。

以下是 Hooks 預期回應的範例形狀：

```
{
  "hookStatus": "SUCCESS" or "FAILED" or "IN_PROGRESS",
  "errorCode": "NonCompliant" or "InternalFailure"
  "message": String,
  "clientRequestToken": String,
  "callbackContext": None,
  "callbackDelaySeconds": Integer,
  "##": [
    {
      "annotationName": String,
      "status": "PASSED" or "FAILED" or "SKIPPED",
      "statusMessage": String,
      "remediationMessage": String,
      "remediationLink": String,
      "severityLevel": "INFORMATIONAL" or "LOW" or "MEDIUM" or "HIGH" or "CRITICAL"
```

```
    }
  ]
}
```

hookStatus

勾點的狀態。此為必要欄位。

有效值：(SUCCESS | FAILED | IN_PROGRESS)

> **Note**
>
> 勾點可以傳回 IN_PROGRESS 3 次。如果未傳回任何結果，勾點將會失敗。對於 Lambda 勾點，這表示您的 Lambda 函數最多可叫用 3 次。

errorCode

顯示操作是否已評估並判斷為無效，或勾點內是否發生錯誤，以防止評估。如果勾點失敗，則此欄位為必要欄位。

有效值：(NonCompliant | InternalFailure)

message

呼叫者的訊息，說明勾點成功或失敗的原因。

> **Note**
>
> 評估 CloudFormation 操作時，此欄位會截斷為 4096 個字元。
> 評估 Cloud Control API 操作時，此欄位會截斷為 1024 個字元。

clientRequestToken

做為 Hook 請求輸入而提供的請求字符。此為必要欄位。

callbackContext

如果您指出 hookStatus 是 IN_PROGRESS ，則會傳遞在叫用 Lambda 函數時作為輸入提供的額外內容。

callbackDelaySeconds

Hooks 應等待多久才再次叫用此 Hook。

註釋

註釋物件陣列，可提供進一步的詳細資訊和修補指引。

annotationName

註釋的識別符。

status

勾點調用狀態。當註釋代表具有類似 Guard 規則的通過/失敗評估的邏輯時，這很有幫助。

有效值：(PASSED | FAILED | SKIPPED)

StatusMessage

特定狀態的說明。

remediationMessage

修正FAILED狀態的建議。例如，如果資源缺少加密，您可以陳述如何將加密新增至資源組態。

remediationLink

用於其他修補指導的 HTTP URL。

severityLevel

定義與此類型的任何違規相關的相對風險。將嚴重性等級指派給 Hook 調用結果時，您可以參考 AWS Security Hub CSPM 嚴重性架構做為如何建構有意義的嚴重性類別的範例。

有效值：(INFORMATIONAL | LOW | MEDIUM | HIGH | CRITICAL)

範例

以下是成功回應的範例：

```
{
  "hookStatus": "SUCCESS",
  "message": "compliant",
  "clientRequestToken": "123avjdjk31"
```

```
}
```

以下是失敗回應的範例：

```
{
  "hookStatus": "FAILED",
  "errorCode": "NonCompliant",
  "message": "S3 Bucket Versioning must be enabled.",
  "clientRequestToken": "123avjdjk31"
 }
```

使用 Lambda Hooks 評估資源操作

每當您建立、更新或刪除資源時，即視為資源操作。例如，如果您執行更新建立新資源的 CloudFormation 堆疊，表示您已完成資源操作。當您使用 Cloud Control API 建立、更新或刪除資源時，這也被視為資源操作。您可以將 CloudFormation Lambda Hook 設定為 Hook `TargetOperations`組態中的目標`RESOURCE`和`CLOUD_CONTROL`操作。

> **Note**
>
> 只有在使用來自 Cloud Control API `delete-resource`或 CloudFormation 的操作觸發程序刪除資源時，才會叫用 `delete` Hook 處理常式`delete-stack`。

主題

- Lambda Hook 資源輸入語法
- Lambda Hook 資源變更輸入範例
- 資源操作的 Lambda 函數範例

Lambda Hook 資源輸入語法

當您的 Lambda 被叫用進行資源操作時，您將會收到 JSON 輸入，其中包含資源屬性、提議屬性，以及勾點叫用的相關內容。

以下是 JSON 輸入的範例形狀：

```
{
```

```
    "awsAccountId": String,
    "stackId": String,
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction": String
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION" or "UPDATE_PRE_PROVISION" or
 "DELETE_PRE_PROVISION"
    "requestData": {
        "targetName": String,
        "targetType": String,
        "targetLogicalId": String,
        "targetModel": {
            "resourceProperties": {...},
            "previousResourceProperties": {...}
        }
    },
    "requestContext": {
        "##": 1,
        "callbackContext": null
    }
}
```

`awsAccountId`

AWS 帳戶 包含要評估之資源的 ID。

`stackId`

此操作所屬 CloudFormation 堆疊的堆疊 ID。如果發起人是雲端控制 API，則此欄位為空白。

`changeSetId`

啟動勾點調用的變更集 ID。如果資源變更是由 Cloud Control API 或 `create-stack`、 `update-stack`或 `delete-stack`操作啟動，則此值為空白。

`hookTypeName`

正在執行的勾點名稱。

`hookTypeVersion`

正在執行的 Hook 版本。

hookModel

LambdaFunction

Hook 調用的目前 Lambda ARN。

actionInvocationPoint

Hook 執行所在佈建邏輯中的確切點。

有效值：(CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

requestData

targetName

正在評估的目標類型，例如 AWS::S3::Bucket。

targetType

正在評估的目標類型，例如 AWS::S3::Bucket。對於使用 Cloud Control API 佈建的資源，此值將為 RESOURCE。

targetLogicalId

要評估之資源的邏輯 ID。如果勾點呼叫的原始伺服器是 CloudFormation，這將是 CloudFormation 範本中定義的邏輯資源 ID。如果此勾點調用的原始伺服器是 Cloud Control API，則會是建構值。

targetModel

resourceProperties

正在修改之資源的提議屬性。如果正在刪除資源，則此值將為空。

previousResourceProperties

目前與正在修改的資源相關聯的屬性。如果正在建立資源，則此值將為空。

requestContext

調用

目前執行勾點的嘗試。

callbackContext

如果 Hookwas 設定為 IN_PROGRESS，且callbackContext傳回 ，則會在叫用後出現。

Lambda Hook 資源變更輸入範例

下列範例輸入顯示將接收要更新之AWS::DynamoDB::Table資源定義的 Lambda 勾點，其中ReadCapacityUnitsProvisionedThroughput的 已從 3 變更為 10。這是可供 Lambda 評估的資料。

```
{
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "hookTypeName": "my::lambda::resourcehookfunction",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    },
    "actionInvocationPoint": "UPDATE_PRE_PROVISION",
    "requestData": {
        "targetName": "AWS::DynamoDB::Table",
        "targetType": "AWS::DynamoDB::Table",
        "targetLogicalId": "DDBTable",
        "targetModel": {
            "resourceProperties": {
                "AttributeDefinitions": [
                    {
                        "AttributeType": "S",
                        "AttributeName": "Album"
                    },
                    {
                        "AttributeType": "S",
                        "AttributeName": "Artist"
                    }
                ],
                "ProvisionedThroughput": {
                    "WriteCapacityUnits": 5,
                    "ReadCapacityUnits": 10
                },
                "KeySchema": [
                    {
                        "KeyType": "HASH",
                        "AttributeName": "Album"
                    },
                    {
                        "KeyType": "RANGE",
```

```
                        "AttributeName": "Artist"
                    }
                ]
            },
            "previousResourceProperties": {
                "AttributeDefinitions": [
                    {
                        "AttributeType": "S",
                        "AttributeName": "Album"
                    },
                    {
                        "AttributeType": "S",
                        "AttributeName": "Artist"
                    }
                ],
                "ProvisionedThroughput": {
                    "WriteCapacityUnits": 5,
                    "ReadCapacityUnits": 5
                },
                "KeySchema": [
                    {
                        "KeyType": "HASH",
                        "AttributeName": "Album"
                    },
                    {
                        "KeyType": "RANGE",
                        "AttributeName": "Artist"
                    }
                ]
            }
        }
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

若要查看資源類型可用的所有屬性，請參閱 AWS::DynamoDB::Table。

資源操作的 Lambda 函數範例

以下是 DynamoDB 的任何資源更新失敗的簡單函數，它會嘗試將 ReadCapacity 的 設定為大於 10 ProvisionedThroughput的 。如果勾點成功，訊息「ReadCapacity 已正確設定」將顯示給發起人。如果請求驗證失敗，勾點將會失敗，狀態為「ReadCapacity 不能超過 10。」

Node.js

```
export const handler = async (event, context) => {
    var targetModel = event?.requestData?.targetModel;
    var targetName = event?.requestData?.targetName;
    var response = {
        "hookStatus": "SUCCESS",
        "message": "ReadCapacity is correctly configured.",
        "clientRequestToken": event.clientRequestToken
    };

    if (targetName == "AWS::DynamoDB::Table") {
        var readCapacity =
 targetModel?.resourceProperties?.ProvisionedThroughput?.ReadCapacityUnits;
        if (readCapacity > 10) {
            response.hookStatus = "FAILED";
            response.errorCode = "NonCompliant";
            response.message = "ReadCapacity must be cannot be more than 10.";
        }
    }
    return response;
};
```

Python

```
import json

def lambda_handler(event, context):
    # Using dict.get() for safe access to nested dictionary values
    request_data = event.get('requestData', {})
    target_model = request_data.get('targetModel', {})
    target_name = request_data.get('targetName', '')

    response = {
        "hookStatus": "SUCCESS",
        "message": "ReadCapacity is correctly configured.",
        "clientRequestToken": event.get('clientRequestToken')
```

```
    }

    if target_name == "AWS::DynamoDB::Table":
        # Safely navigate nested dictionary
        resource_properties = target_model.get('resourceProperties', {})
        provisioned_throughput = resource_properties.get('ProvisionedThroughput',
{})
        read_capacity = provisioned_throughput.get('ReadCapacityUnits')

        if read_capacity and read_capacity > 10:
            response['hookStatus'] = "FAILED"
            response['errorCode'] = "NonCompliant"
            response['message'] = "ReadCapacity must be cannot be more than 10."

    return response
```

使用 Lambda Hooks 評估堆疊操作

每當您使用新範本建立、更新或刪除堆疊時，都可以透過評估新範本來設定 CloudFormation Lambda Hook，並可能封鎖堆疊操作以繼續。您可以在 Hook `TargetOperations`組態中將 CloudFormation Lambda Hook 設定為目標`STACK`操作。

主題

- Lambda Hook 堆疊輸入語法
- Lambda Hook 堆疊變更輸入範例
- 堆疊操作的範例 Lambda 函數

Lambda Hook 堆疊輸入語法

當您的 Lambda 調用堆疊操作時，您將會收到 JSON 請求，其中包含勾點調用內容、`actionInvocationPoint`和 請求內容。由於 CloudFormation 範本的大小，以及 Lambda 函數接受的有限輸入大小，實際範本會存放在 Amazon S3 物件中。的輸入`requestData`包含另一個物件的 Amazon S3 已簽章 URL，其中包含目前和先前的範本版本。

以下是 JSON 輸入的範例形狀：

```
{
    "clientRequesttoken": String,
    "awsAccountId": String,
```

```
    "stackID": String,
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction":String
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION" or "UPDATE_PRE_PROVISION" or
 "DELETE_PRE_PROVISION"
    "requestData": {
        "targetName": "STACK",
        "targetType": "STACK",
        "targetLogicalId": String,
        "payload": String (S3 Presigned URL)
    },
    "requestContext": {
        "invocation": Integer,
        "callbackContext": String
    }
}
```

`clientRequesttoken`

做為 Hook 請求輸入而提供的請求字符。此為必要欄位。

`awsAccountId`

AWS 帳戶 包含要評估之堆疊的 ID。

`stackID`

CloudFormation 堆疊的堆疊 ID。

`changeSetId`

啟動勾點調用的變更集 ID。如果堆疊變更是由 Cloud Control API 或 `create-stack`、 `update-stack`或 `delete-stack`操作啟動，則此值為空白。

`hookTypeName`

正在執行的勾點名稱。

`hookTypeVersion`

正在執行的 Hook 版本。

hookModel

LambdaFunction

Hook 調用的目前 Lambda ARN。

actionInvocationPoint

佈建邏輯中 Hook 執行的確切點。

有效值：(CREATE_PRE_PROVISION | UPDATE_PRE_PROVISION | DELETE_PRE_PROVISION)

requestData

targetName

此值將為 STACK。

targetType

此值將為 STACK。

targetLogicalId

堆疊名稱。

payload

Amazon S3 預先簽章的 URL，其中包含具有目前和先前範本定義的 JSON 物件。

requestContext

如果正在重新叫用勾點，則會設定此物件。

invocation

目前執行勾點的嘗試。

callbackContext

如果勾點設定為 callbackContext IN_PROGRESS並傳回，則會在叫用時出現。

請求資料中的 payload 屬性是您程式碼需要擷取的 URL。收到 URL 後，您會取得具有下列結構描述的物件：

```
{
    "template": String,
    "previousTemplate": String
```

```
}
```

`template`

提供給 `create-stack`或 的完整 CloudFormation 範本`update-stack`。根據提供給 CloudFormation 的內容，它可以是 JSON 或 YAML 字串。

在 `delete-stack` 操作中，此值將為空。

`previousTemplate`

先前的 CloudFormation 範本。根據提供給 CloudFormation 的內容，它可以是 JSON 或 YAML 字串。

在 `delete-stack` 操作中，此值將為空。

Lambda Hook 堆疊變更輸入範例

以下是堆疊變更輸入的範例。Hook 正在評估將 更新`ObjectLockEnabled`為 true 的變更，並新增 Amazon SQS 佇列：

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "changeSetId": null,
    "hookTypeName": "my::lambda::stackhook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    },
    "actionInvocationPoint": "UPDATE_PRE_PROVISION",
    "requestData": {
        "targetName": "STACK",
        "targetType": "STACK",
        "targetLogicalId": "my-cloudformation-stack",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
```

```
}
```

這是 payload的範例requestData：

```
{
    "template": "{\"Resources\":{\"S3Bucket\":{\"Type\":\"AWS::S3::Bucket\",
\"Properties\":{\"ObjectLockEnabled\":true}},\"SQSQueue\":{\"Type\":\"AWS::SQS::Queue
\",\"Properties\":{\"QueueName\":\"NewQueue\"}}}}",
    "previousTemplate": "{\"Resources\":{\"S3Bucket\":{\"Type\":\"AWS::S3::Bucket\",
\"Properties\":{\"ObjectLockEnabled\":false}}}}"
}
```

堆疊操作的範例 Lambda 函數

下列範例是一個簡單的 函數，可下載堆疊操作承載、剖析範本 JSON，並傳回 SUCCESS。

Node.js

```
export const handler = async (event, context) => {
    var targetType = event?.requestData?.targetType;
    var payloadUrl = event?.requestData?.payload;

    var response = {
        "hookStatus": "SUCCESS",
        "message": "Stack update is compliant",
        "clientRequestToken": event.clientRequestToken
    };
    try {
        const templateHookPayloadRequest = await fetch(payloadUrl);
        const templateHookPayload = await templateHookPayloadRequest.json()
        if (templateHookPayload.template)  {
            // Do something with the template templateHookPayload.template
            // JSON or YAML
        }
        if (templateHookPayload.previousTemplate) {
            // Do something with the template templateHookPayload.previousTemplate
            // JSON or YAML
        }
    } catch (error) {
        console.log(error);
        response.hookStatus = "FAILED";
        response.message = "Failed to evaluate stack operation.";
        response.errorCode = "InternalFailure";
```

```
    }
    return response;
};
```

Python

若要使用 Python，您需要匯入程式`requests`庫。若要這樣做，您需要在建立 Lambda 函數時，在部署套件中包含 程式庫。如需詳細資訊，請參閱《 AWS Lambda 開發人員指南》中的建立具有相依性的 .zip 部署套件。

```
import json
import requests

def lamnbda_handler(event, context):
    # Safely access nested dictionary values
    request_data = event.get('requestData', {})
    target_type = request_data.get('targetType')
    payload_url = request_data.get('payload')

    response = {
        "hookStatus": "SUCCESS",
        "message": "Stack update is compliant",
        "clientRequestToken": event.get('clientRequestToken')
    }

    try:
        # Fetch the payload
        template_hook_payload_request = requests.get(payload_url)
        template_hook_payload_request.raise_for_status()  # Raise an exception for
 bad responses
        template_hook_payload = template_hook_payload_request.json()

        if 'template' in template_hook_payload:
            # Do something with the template template_hook_payload['template']
            # JSON or YAML
            pass

        if 'previousTemplate' in template_hook_payload:
            # Do something with the template
 template_hook_payload['previousTemplate']
            # JSON or YAML
            pass
```

```
    except Exception as error:
        print(error)
        response['hookStatus'] = "FAILED"
        response['message'] = "Failed to evaluate stack operation."
        response['errorCode'] = "InternalFailure"

    return response
```

使用 Lambda Hooks 評估變更集操作

每當您建立變更集時，都可以設定 CloudFormation Lambda Hook 來先評估新的變更集，並可能封鎖其執行。您可以在 Hook `TargetOperations`組態中將 CloudFormation Lambda Hook 設定為目標`CHANGE_SET`操作。

主題

- Lambda Hook 變更集輸入語法
- Lambda Hook 變更集變更輸入範例
- 變更集操作的範例 Lambda 函數

Lambda Hook 變更集輸入語法

變更集操作的輸入類似於堆疊操作，但 的承載`requestData`也包含變更集引入的資源變更清單。

以下是 JSON 輸入的範例形狀：

```
{
    "clientRequesttoken": String,
    "awsAccountId": String,
    "stackID": String,
    "changeSetId": String,
    "hookTypeName": String,
    "hookTypeVersion": String,
    "hookModel": {
        "LambdaFunction":String
    },
    "requestData": {
        "targetName": "CHANGE_SET",
        "targetType": "CHANGE_SET",
        "targetLogicalId": String,
        "payload": String (S3 Presigned URL)
```

```
    },
    "requestContext": {
        "invocation": Integer,
        "callbackContext": String
    }
}
```

clientRequesttoken

做為 Hook 請求輸入而提供的請求字符。此為必要欄位。

awsAccountId

AWS 帳戶 包含要評估之堆疊的 ID。

stackID

CloudFormation 堆疊的堆疊 ID。

changeSetId

啟動勾點調用的變更集 ID。

hookTypeName

正在執行的勾點名稱。

hookTypeVersion

正在執行的 Hook 版本。

hookModel

LambdaFunction

Hook 調用的目前 Lambda ARN。

requestData

targetName

此值將為 CHANGE_SET。

targetType

此值將為 CHANGE_SET。

targetLogicalId

變更集 ARN。

`payload`

Amazon S3 預先簽章的 URL，其中包含具有目前範本的 JSON 物件，以及此變更集引入的變更清單。

`requestContext`

如果正在重新叫用勾點，則會設定此物件。

`invocation`

目前執行勾點的嘗試。

`callbackContext`

如果勾點設定為 `callbackContext` `IN_PROGRESS`並傳回，則會在叫用時出現。

請求資料中的 `payload` 屬性是您程式碼需要擷取的 URL。收到 URL 後，您會取得具有下列結構描述的物件：

```
{
    "template": String,
    "changedResources": [
        {
            "action": String,
            "beforeContext": JSON String,
            "afterContext": JSON String,
            "lineNumber": Integer,
            "logicalResourceId": String,
            "resourceType": String
        }
    ]
}
```

`template`

提供給 `create-stack`或 的完整 CloudFormation 範本`update-stack`。根據提供給 CloudFormation 的內容，它可以是 JSON 或 YAML 字串。

`changedResources`

已變更資源的清單。

`action`

套用至資源的變更類型。

有效值： (CREATE | UPDATE | DELETE)

`beforeContext`

變更前資源屬性的 JSON 字串。建立資源時，此值為 null。此 JSON 字串中的所有布林值和數值都是 STRINGS。

`afterContext`

如果執行此變更集，資源屬性的 JSON 字串。刪除資源時，此值為 null。此 JSON 字串中的所有布林值和數值都是 STRINGS。

`lineNumber`

範本中造成此變更的行號。如果動作是 ，則DELETE此值將為 null。

`logicalResourceId`

要變更之資源的邏輯資源 ID。

`resourceType`

正在變更的資源類型。

Lambda Hook 變更集變更輸入範例

以下是變更集變更輸入的範例。在下列範例中，您可以查看變更集引進的變更。第一個變更是刪除名為 的佇列CoolQueue。第二個變更是新增名為 的新佇列NewCoolQueue。上次變更是 的更新DynamoDBTable。

```
{
    "clientRequestToken": "f8da6d11-b23f-48f4-814c-0fb6a667f50e",
    "awsAccountId": "123456789012",
    "stackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "changeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000",
    "hookTypeName": "my::lambda::changesethook",
    "hookTypeVersion": "00000008",
    "hookModel": {
        "LambdaFunction": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

```
    },
    "actionInvocationPoint": "CREATE_PRE_PROVISION",
    "requestData": {
        "targetName": "CHANGE_SET",
        "targetType": "CHANGE_SET",
        "targetLogicalId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000",
        "payload": "https://s3......"
    },
    "requestContext": {
        "invocation": 1,
        "callbackContext": null
    }
}
```

這是 payload的範例requestData.payload：

```
{
  template: 'Resources:\n' +
    '  DynamoDBTable:\n' +
    '    Type: AWS::DynamoDB::Table\n' +
    '    Properties:\n' +
    '      AttributeDefinitions:\n' +
    '        - AttributeName: "PK"\n' +
    '          AttributeType: "S"\n' +
    '      BillingMode: "PAY_PER_REQUEST"\n' +
    '      KeySchema:\n' +
    '        - AttributeName: "PK"\n' +
    '          KeyType: "HASH"\n' +
    '      PointInTimeRecoverySpecification:\n' +
    '        PointInTimeRecoveryEnabled: false\n' +
    '  NewSQSQueue:\n' +
    '    Type: AWS::SQS::Queue\n' +
    '    Properties:\n' +
    '      QueueName: "NewCoolQueue"',
  changedResources: [
    {
      logicalResourceId: 'SQSQueue',
      resourceType: 'AWS::SQS::Queue',
      action: 'DELETE',
      lineNumber: null,
      beforeContext: '{"Properties":{"QueueName":"CoolQueue"}}',
      afterContext: null
```

```
    },
    {
      logicalResourceId: 'NewSQSQueue',
      resourceType: 'AWS::SQS::Queue',
      action: 'CREATE',
      lineNumber: 14,
      beforeContext: null,
      afterContext: '{"Properties":{"QueueName":"NewCoolQueue"}}'
    },
    {
      logicalResourceId: 'DynamoDBTable',
      resourceType: 'AWS::DynamoDB::Table',
      action: 'UPDATE',
      lineNumber: 2,
      beforeContext: '{"Properties":
{"BillingMode":"PAY_PER_REQUEST","AttributeDefinitions":
[{"AttributeType":"S","AttributeName":"PK"}],"KeySchema":
[{"KeyType":"HASH","AttributeName":"PK"}]}}',
      afterContext: '{"Properties":
{"BillingMode":"PAY_PER_REQUEST","PointInTimeRecoverySpecification":
{"PointInTimeRecoveryEnabled":"false"},"AttributeDefinitions":
[{"AttributeType":"S","AttributeName":"PK"}],"KeySchema":
[{"KeyType":"HASH","AttributeName":"PK"}]}}'
    }
  ]
}
```

變更集操作的範例 Lambda 函數

下列範例是一個簡單的 函數，可下載變更集操作承載、逐一查看每個變更，然後在傳回 之前列印屬性前後的 SUCCESS。

Node.js

```
export const handler = async (event, context) => {
    var payloadUrl = event?.requestData?.payload;
    var response = {
        "hookStatus": "SUCCESS",
        "message": "Change set changes are compliant",
        "clientRequestToken": event.clientRequestToken
    };
    try {
        const changeSetHookPayloadRequest = await fetch(payloadUrl);
```

```
        const changeSetHookPayload = await changeSetHookPayloadRequest.json();
        const changes = changeSetHookPayload.changedResources || [];
        for(const change of changes) {
            var beforeContext = {};
            var afterContext = {};
            if(change.beforeContext) {
                beforeContext = JSON.parse(change.beforeContext);
            }
            if(change.afterContext) {
                afterContext = JSON.parse(change.afterContext);
            }
            console.log(beforeContext)
            console.log(afterContext)
            // Evaluate Change here
        }
    } catch (error) {
        console.log(error);
        response.hookStatus = "FAILED";
        response.message = "Failed to evaluate change set operation.";
        response.errorCode = "InternalFailure";
    }
    return response;
};
```

Python

若要使用 Python，您需要匯入程式`requests`庫。若要這樣做，您需要在建立 Lambda 函數時，在部署套件中包含 程式庫。如需詳細資訊，請參閱《 AWS Lambda 開發人員指南》中的建立具有相依性的 .zip 部署套件。

```
import json
import requests

def lambda_handler(event, context):
    payload_url = event.get('requestData', {}).get('payload')
    response = {
        "hookStatus": "SUCCESS",
        "message": "Change set changes are compliant",
        "clientRequestToken": event.get('clientRequestToken')
    }

    try:
        change_set_hook_payload_request = requests.get(payload_url)
```

```
        change_set_hook_payload_request.raise_for_status()  # Raises an HTTPError
for bad responses
        change_set_hook_payload = change_set_hook_payload_request.json()

        changes = change_set_hook_payload.get('changedResources', [])

        for change in changes:
            before_context = {}
            after_context = {}

            if change.get('beforeContext'):
                before_context = json.loads(change['beforeContext'])

            if change.get('afterContext'):
                after_context = json.loads(change['afterContext'])

            print(before_context)
            print(after_context)
            # Evaluate Change here

    except requests.RequestException as error:
        print(error)
        response['hookStatus'] = "FAILED"
        response['message'] = "Failed to evaluate change set operation."
        response['errorCode'] = "InternalFailure"
    except json.JSONDecodeError as error:
        print(error)
        response['hookStatus'] = "FAILED"
        response['message'] = "Failed to parse JSON payload."
        response['errorCode'] = "InternalFailure"

    return response
```

準備建立 Lambda 勾點

建立 Lambda Hook 之前，您必須完成下列先決條件：

- 您必須已建立 Lambda 函數。如需更多資訊，請參閱為勾點建立 Lambda 函數。
- 建立勾點的使用者或角色必須具有足夠的許可，才能啟用勾點。如需詳細資訊，請參閱授予 CloudFormation Hooks 的 IAM 許可。

- 若要使用 AWS CLI 或 開發套件建立 Lambda Hook，您必須手動建立具有 IAM 許可和信任政策的執行角色，以允許 CloudFormation 叫用 Lambda Hook。

建立 Lambda Hook 的執行角色

Hook 會使用 執行角色來取得在 中叫用該 Hook 所需的許可 AWS 帳戶。

如果您從 建立 Lambda Hook，則可以自動建立此角色 AWS 管理主控台；否則，您必須自行建立此角色。

下一節說明如何設定建立 Lambda Hook 的許可。

所需的許可

遵循《IAM 使用者指南》中的使用自訂信任政策建立角色的指引，建立具有自訂信任政策的角色。

然後，完成下列步驟以設定您的許可：

1. 將下列最低權限政策連接至您要用來建立 Lambda Hook 的 IAM 角色。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
    }
  ]
}
```

2. 將信任政策新增至角色，授予您的勾點擔任角色的許可。以下顯示您可以使用的範例信任政策。

   JSON

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "hooks.cloudformation.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

在您的帳戶中啟用 Lambda Hook

下列主題說明如何在您的 帳戶中啟用 Lambda Hook，這使其可在其啟用的帳戶和區域中使用。

主題

- 啟用 Lambda 勾點 （ 主控台 ）
- 啟用 Lambda 勾點 (AWS CLI)
- 相關資源

啟用 Lambda 勾點 （ 主控台 ）

啟用 Lambda Hook 以在您的帳戶中使用

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇您要建立連接所在的 AWS 區域 。
3. 如果您尚未為 勾點建立 Lambda 函數，請執行下列動作：

   - 開啟 Lambda 主控台中的 Functions (函數) 頁面。
   - 建立您將與此勾點搭配使用的 Lambda 函數，然後返回此程序。如需詳細資訊，請參閱建立 Lambda 函數來評估 Lambda Hooks 的資源。

   如果您已建立 Lambda 函數，請繼續下一個步驟。
4. 在左側導覽窗格中，選擇勾點。
5. 在勾點頁面上，選擇建立勾點，然後選擇使用 Lambda。

6. 針對勾點名稱，選擇下列其中一個選項：

   - 提供將在 之後新增的簡短描述性名稱`Private::Lambda::`。例如，如果您輸入 *`MyTestHook`*，則完整的勾點名稱會變成 `Private::Lambda::`*`MyTestHook`*。
   - 使用此格式提供完整的勾點名稱（也稱為別名）：*`Provider`*`::`*`ServiceName`*`::`*`HookName`*

7. 對於 Lambda 函數，請提供要與此勾點搭配使用的 Lambda 函數。您可以使用：

   - 不含尾碼的完整 Amazon Resource Name (ARN)。
   - 具有版本或別名尾碼的合格 ARN。

8. 針對勾點目標，選擇要評估的內容：

   - 堆疊 — 在使用者建立、更新或刪除堆疊時評估堆疊範本。
   - 資源 — 評估使用者更新堆疊時的個別資源變更。
   - 變更集 — 評估使用者建立變更集時的計劃更新。
   - 雲端控制 API — 評估 Cloud Control API 啟動的建立、更新或刪除操作。

9. 針對動作，選擇哪些動作（建立、更新、刪除）將調用您的勾點。

10. 針對勾點模式，選擇勾點在勾點叫用 Lambda 函數傳回回應時的勾點`FAILED`回應方式：

    - 警告 — 向使用者發出警告，但允許動作繼續。這對於非關鍵驗證或資訊檢查非常有用。
    - 失敗 — 防止動作繼續。這有助於強制執行嚴格的合規或安全政策。

11. 針對執行角色，選擇 Hook 用來叫用 Lambda 函數的 IAM 角色。您可以允許 CloudFormation 為您自動建立執行角色，也可以指定您已建立的角色。

12. 選擇下一步。

13. （選用）對於勾點篩選條件，請執行下列動作：

    a. 針對資源篩選條件，指定哪些資源類型可以叫用勾點。這可確保僅針對相關資源叫用勾點。

    b. 針對篩選條件，選擇套用堆疊名稱和堆疊角色篩選條件的邏輯：

       - 所有堆疊名稱和堆疊角色 – 只有在所有指定的篩選條件相符時，才會叫用勾點。
       - 任何堆疊名稱和堆疊角色 – 如果至少一個指定的篩選條件相符，則會叫用勾點。

> **Note**
>
> 對於雲端控制 API 操作，會忽略所有堆疊名稱和堆疊角色篩選條件。

c. 對於堆疊名稱，請在勾點調用中包含或排除特定堆疊。

- 針對包含，指定要包含的堆疊名稱。當您有一小組想要鎖定的特定堆疊時，請使用此選項。只有此清單中指定的堆疊才會叫用勾點。
- 針對排除，指定要排除的堆疊名稱。當您想要在大多數堆疊上叫用勾點，但排除幾個特定堆疊時，請使用此選項。除了此處列出的堆疊之外，所有堆疊都會叫用勾點。

d. 對於堆疊角色，請根據特定堆疊相關聯的 IAM 角色，從勾點調用中包含或排除特定堆疊。

- 針對包含，指定一或多個 IAM 角色 ARNs 至與這些角色相關聯的目標堆疊。只有這些角色啟動的堆疊操作才會叫用勾點。
- 針對排除，為您要排除的堆疊指定一或多個 IAM 角色 ARNs。所有堆疊都會叫用勾點，但由指定角色啟動的堆疊除外。

14. 選擇下一步。
15. 在檢閱和啟用頁面上，檢閱您的選擇。選擇編輯以對相關區段進行變更。
16. 當您準備好繼續時，請選擇啟用勾點。

啟用 Lambda 勾點 (AWS CLI)

在繼續之前，請確認您已建立 Lambda 函數和您將與此勾點搭配使用的執行角色。如需詳細資訊，請參閱建立 Lambda 函數來評估 Lambda Hooks 的資源及建立 Lambda Hook 的執行角色。

啟用 Lambda Hook 以用於您的帳戶 (AWS CLI)

1. 若要開始啟用勾點，請使用下列activate-type命令，將預留位置取代為您的特定值。此命令授權 勾點從您的 使用指定的執行角色 AWS 帳戶。

```
aws cloudformation activate-type --type HOOK \
  --type-name AWS::Hooks::LambdaHook \
  --publisher-id aws-hooks \
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --type-name-alias Private::Lambda::MyTestHook \
  --region us-west-2
```

2. 若要完成啟用勾點，您必須使用 JSON 組態檔案進行設定。

   使用 cat命令建立具有下列結構的 JSON 檔案。如需詳細資訊，請參閱勾點組態結構描述語法參考。

```
$ cat > config.json
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "CLOUD_CONTROL"
      ],
      "FailureMode": "WARN",
      "Properties": {
        "LambdaFunction": "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
      },
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

- `HookInvocationStatus`：將 設定為 `ENABLED` 以啟用 勾點。
- `TargetOperations`：指定勾點將評估的操作。
- `FailureMode`：設為 `FAIL` 或 `WARN`。
- `LambdaFunction`：指定 Lambda 函數的 ARN。
- `TargetFilters`：指定將叫用勾點的動作類型。

3. 使用下列set-type-configuration命令以及您建立的 JSON 檔案來套用組態。將預留位置取代為您的特定值。

```
aws cloudformation set-type-configuration \
  --configuration file://config.json \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

相關資源

我們提供範本範例，供您用來了解如何在 CloudFormation 堆疊範本中宣告 Lambda Hook。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS::CloudFormation::LambdaHook。

檢視您帳戶中 Lambda Hooks 的日誌

使用 Lambda Hook 時，您可以在 Lambda 主控台中找到驗證輸出報告日誌檔案。

在 Lambda 主控台中檢視 Lambda Hook 日誌

檢視 Lambda Hook 輸出日誌檔案

1. 登入 Lambda 主控台。
2. 在畫面上方的導覽列上，選擇 AWS 區域。
3. 選擇 函數。
4. 選擇所需的 Lambda 函數。
5. 選擇測試標籤。
6. 選擇 CloudWatch Logs Live Trail
7. 選擇下拉式選單，然後選取您要檢視的日誌群組。
8. 選擇 開始使用。日誌會顯示在 CloudWatch Logs Live Trail 視窗中。根據您的偏好，選擇在資料欄中檢視或以純文字檢視。

    - 您可以將更多篩選條件新增至結果，方法是在新增篩選條件模式欄位中新增這些篩選條件。此欄位可讓您篩選結果，只包含符合指定模式的事件。

如需檢視 Lambda 函數日誌的詳細資訊，請參閱檢視 Lambda 函數的 CloudWatch Logs。

刪除您帳戶中的 Lambda 勾點

當您不再需要已啟用的 Lambda Hook 時，請使用下列程序在帳戶中將其刪除。

若要暫時停用勾點，而不是將其刪除，請參閱 停用和啟用 CloudFormation 勾點。

主題

- 刪除您帳戶中的 Lambda 勾點 （ 主控台 ）

- 刪除您帳戶中的 Lambda 勾點 (AWS CLI)

刪除您帳戶中的 Lambda 勾點 （主控台）

刪除您帳戶中的 Lambda 勾點

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 勾點所在的 。
3. 從導覽窗格中，選擇勾點。
4. 在勾點頁面上，尋找您要刪除的 Lambda 勾點。
5. 選取勾點旁的核取方塊，然後選擇刪除。
6. 出現確認提示時，輸入勾點名稱以確認刪除指定的勾點，然後選擇刪除。

刪除您帳戶中的 Lambda 勾點 (AWS CLI)

> **Note**
>
> 您必須先停用勾點，才能刪除勾點。如需詳細資訊，請參閱在帳戶中停用和啟用勾點 (AWS CLI)。

使用下列deactivate-type命令來停用勾點，該勾點會從您的帳戶中移除它。將預留位置取代為您的特定值。

```
aws cloudformation deactivate-type \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

使用 CloudFormation CLI 開發自訂勾點

本節適用於想要開發自訂勾點並在 CloudFormation 登錄檔中註冊的客戶。它提供 CloudFormation 勾點結構的概觀，以及使用 Python 或 Java 開發、註冊、測試、管理和發佈您自己的勾點的指南。

開發自訂勾點有三個主要步驟：

1. 啟動

   若要開發自訂勾點，您必須設定和使用 CloudFormation CLI。若要啟動 Hook 的專案及其必要檔案，請使用 CloudFormation CLI init命令並指定您要建立 Hook。如需詳細資訊，請參閱啟動自訂 CloudFormation 勾點專案。

2. 模型

   若要建立、撰寫和驗證您的勾點結構描述，請定義勾點、其屬性及其屬性。

   CloudFormation CLI 會建立對應至特定勾點的空處理常式函數。將您自己的邏輯新增至這些處理常式，以控制 Hook 調用期間在其目標生命週期的每個階段發生的情況。如需詳細資訊，請參閱建立自訂 CloudFormation 勾點的模型。

3. 註冊

   若要註冊勾點，請提交您的勾點以註冊為私有或公有第三方延伸模組。向 `submit`操作註冊您的勾點。如需詳細資訊，請參閱向 註冊自訂勾點 CloudFormation。

   下列任務與註冊您的勾點相關聯：

   a. 發佈 – 勾點會發佈至登錄檔。

   b. 設定 – 當類型組態調用堆疊時，會設定勾點。

   > **Note**
   >
   > 勾點會在 30 秒後逾時，並重試最多 3 次。如需詳細資訊，請參閱逾時和重試限制。

主題

- 開發自訂 CloudFormation 勾點的先決條件
- 啟動自訂 CloudFormation 勾點專案
- 建立自訂 CloudFormation 勾點的模型
- 向 註冊自訂勾點 CloudFormation
- 在 中測試自訂勾點 AWS 帳戶
- 更新自訂勾點
- 從 CloudFormation 登錄檔取消註冊自訂掛鉤
- 發佈勾點以供公開使用
- CloudFormation 勾點的結構描述語法參考

開發自訂 CloudFormation 勾點的先決條件

您可以使用 Java 或 Python 開發自訂勾點。以下是開發自訂勾點的先決條件：

Java 先決條件

- Apache Maven
- JDK 17

> **Note**
>
> 如果您想要使用 CloudFormation 命令列界面 (CLI) 啟動 Java 的 Hooks 專案，您也必須安裝 Python 3.8 或更新版本。CloudFormation CLI 的 Java 外掛程式可以透過 `pip`(Python 的套件管理員） 安裝，該程式使用 Python 解散。

若要為 Java Hooks 專案實作 Hook 處理常式，您可以下載 Java Hook 處理常式範例檔案。

Python 先決條件

- Python 3.8 版或更新版本。

若要為 Python Hooks 專案實作 Hook 處理常式，您可以下載 Python Hook 處理常式範例檔案。

開發勾點的許可

除了 CloudFormation `Create`、 `Update`和 `Delete`堆疊許可之外，您還需要存取下列 AWS CloudFormation 操作。存取這些操作是透過 IAM 角色的 CloudFormation 政策來管理。

- register-type
- list-types
- deregister-type
- set-type-configuration

如需詳細資訊，請參閱授予 CloudFormation Hooks 的 IAM 許可。

設定 Hooks 的開發環境

若要開發勾點，您應該熟悉 CloudFormation 範本，以及 Python 或 Java。

若要安裝 CloudFormation CLI 和相關聯的外掛程式：

1. 使用 Python 套件管理員 pip，安裝 CloudFormation CLI。

```
pip3 install cloudformation-cli
```

2. 安裝 CloudFormation CLI 的 Python 或 Java 外掛程式。

   Python

```
pip3 install cloudformation-cli-python-plugin
```

   Java

```
pip3 install cloudformation-cli-java-plugin
```

若要升級 CloudFormation CLI 和外掛程式，您可以使用升級選項。

Python

```
pip3 install --upgrade cloudformation-cli cloudformation-cli-python-plugin
```

Java

```
pip3 install --upgrade cloudformation-cli cloudformation-cli-java-plugin
```

啟動自訂 CloudFormation 勾點專案

建立自訂 Hooks 專案的第一步是啟動專案。您可以使用 CloudFormation CLI `init`命令來啟動自訂 Hooks 專案。

`init` 命令會啟動精靈，引導您設定專案，包括 Hooks 結構描述檔案。使用此結構描述檔案作為定義勾點形狀和語意的起點。如需詳細資訊，請參閱結構描述語法。

若要初始化 Hook 專案：

1. 建立專案的目錄。

```
mkdir ~/mycompany-testing-mytesthook
```

2. 導覽至新目錄。

```
cd ~/mycompany-testing-mytesthook
```

3. 使用 CloudFormation CLI `init`命令啟動專案。

```
cfn init
```

命令會傳回下列輸出：

```
Initializing new project
```

4. `init` 命令會啟動精靈，引導您設定專案。出現提示時，輸入 h 以指定 Hooks 專案。

```
Do you want to develop a new resource(r) a module(m) or a hook(h)?
```

```
h
```

5. 輸入勾點類型的名稱。

```
What's the name of your hook type?
(Organization::Service::Hook)
```

```
MyCompany::Testing::MyTestHook
```

6. 如果只安裝一種語言外掛程式，預設會選取該外掛程式。如果已安裝多個語言外掛程式，您可以選擇所需的語言。為您選擇的語言輸入數字選擇。

```
Select a language for code generation:
[1] java
[2] python38
[3] python39
(enter an integer):
```

7. 根據所選的開發語言設定封裝。

Python

(選用) 針對與平台無關的封裝選擇 Docker。雖然不需要 Docker，但強烈建議您更輕鬆地封裝。

```
Use docker for platform-independent packaging (Y/n)?
This is highly recommended unless you are experienced with cross-platform Python
 packaging.
```

Java

設定 Java 套件名稱，然後選擇 codegen 模型。您可以使用預設套件名稱，或建立新的套件名稱。

```
Enter a package name (empty for default 'com.mycompany.testing.mytesthook'):
```

```
Choose codegen model - 1 (default) or 2 (guided-aws):
```

結果：您已成功啟動專案，並產生開發勾點所需的檔案。以下是構成 Python 3.8 的 Hooks 專案的目錄和檔案範例。

```
mycompany-testing-mytesthook.json
rpdk.log
README.md
requirements.txt
hook-role.yaml
template.yml
docs
    README.md
src
    __init__.py
    handlers.py
    models.py
    target_models
        aws_s3_bucket.py
```

> **Note**
>
> `src` 目錄中的檔案會根據您的語言選擇建立。產生的檔案中有一些有用的註解和範例。當您執行 `generate`命令來新增處理常式的執行時間程式碼時`models.py`，某些檔案，例如 ，會在稍後的步驟中自動更新。

建立自訂 CloudFormation 勾點的模型

建立自訂 CloudFormation 勾點的模型需要建立定義勾點、其屬性及其屬性的結構描述。當您使用 `cfn init`命令建立自訂 Hook 專案時，Hook 結構描述範例會建立為 JSON 格式的文字檔案 *hook-name*`.json`。

目標調用點和目標動作會指定叫用勾點的確切點。勾點處理常式託管這些點的可執行自訂邏輯。例如，`CREATE`操作的目標動作會使用`preCreate`處理常式。當 Hook 目標和服務執行相符動作時，您在處理常式中寫入的程式碼將會叫用 。勾點目標是調用勾點的目的地。您可以指定目標，例如 CloudFormation 公有資源、私有資源或自訂資源。勾點支援無限數量的勾點目標。

結構描述包含勾點所需的許可。編寫勾點需要您為每個勾點處理常式指定許可。CloudFormation 鼓勵作者撰寫遵循授予最低權限或僅授予執行任務所需許可之標準安全建議的政策。決定使用者 （和角色） 需要執行的動作，然後制定政策，允許他們僅執行勾點操作的任務。CloudFormation 使用這些許可來縮小範圍勾點使用者提供的許可。這些許可會傳遞至 勾點。勾點處理常式使用這些許可來存取 AWS 資源。

您可以使用下列結構描述檔案做為定義勾點的起點。使用勾點結構描述來指定您要實作的處理常式。如果您選擇不實作特定處理常式，請從勾點結構描述的處理常式區段中移除它。如需結構描述的詳細資訊，請參閱 結構描述語法。

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies S3 bucket and SQS queues properties before create and
 update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
```

```
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string"
            }
        },
        "required":[

        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preUpdate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preDelete":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetEncryptionConfiguration",
                "sqs:ListQueues",
```

```
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ]
        }
    },
    "additionalProperties":false
}
```

主題

- 使用 Java 建立自訂 CloudFormation 勾點的模型
- 使用 Python 建立自訂 CloudFormation 勾點的模型

使用 Java 建立自訂 CloudFormation 勾點的模型

建立自訂 CloudFormation 勾點的模型需要建立定義勾點、其屬性及其屬性的結構描述。本教學課程會逐步引導您使用 Java 建立自訂勾點的模型。

步驟 1：新增專案相依性

Java 型 Hooks 專案依賴 Maven `pom.xml` 的檔案做為相依性。展開以下區段，並將原始程式碼複製到專案根目錄中的 `pom.xml` 檔案。

勾點專案相依性 (pom.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<project
    xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.testing.mytesthook</groupId>
    <artifactId>mycompany-testing-mytesthook-handler</artifactId>
    <name>mycompany-testing-mytesthook-handler</name>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <aws.java.sdk.version>2.16.1</aws.java.sdk.version>
        <checkstyle.version>8.36.2</checkstyle.version>
        <commons-io.version>2.8.0</commons-io.version>
        <jackson.version>2.11.3</jackson.version>
        <maven-checkstyle-plugin.version>3.1.1</maven-checkstyle-plugin.version>
        <mockito.version>3.6.0</mockito.version>
        <spotbugs.version>4.1.4</spotbugs.version>
        <spotless.version>2.5.0</spotless.version>
        <maven-javadoc-plugin.version>3.2.0</maven-javadoc-plugin.version>
        <maven-source-plugin.version>3.2.1</maven-source-plugin.version>
        <cfn.generate.args/>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.16.1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-
cloudformation-rpdk-java-plugin -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>aws-cloudformation-rpdk-java-plugin</artifactId>
            <version>[2.0.0,3.0.0)</version>
        </dependency>

        <!-- AWS Java SDK v2 Dependencies -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sdk-core</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>cloudformation</artifactId>
        </dependency>
```

```
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>utils</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sqs</artifactId>
        </dependency>

        <!-- Test dependency for Java Providers -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>cloudformation-cli-java-plugin-testing-support</artifactId>
            <version>1.0.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-s3 -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-s3</artifactId>
            <version>1.12.85</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>${commons-io.version}</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.9</version>
        </dependency>
```

```
        <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-collections4
 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-collections4</artifactId>
            <version>4.4</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.google.guava/guava -->
        <dependency>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
            <version>29.0-jre</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-
cloudformation -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-cloudformation</artifactId>
            <version>1.11.555</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.14</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/software.amazon.cloudformation/aws-
cloudformation-resource-schema -->
        <dependency>
            <groupId>software.amazon.cloudformation</groupId>
            <artifactId>aws-cloudformation-resource-schema</artifactId>
            <version>[2.0.5, 3.0.0)</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-databind -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>${jackson.version}</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/
jackson-dataformat-cbor -->
```

```
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-cbor</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.datatype</groupId>
            <artifactId>jackson-datatype-jsr310</artifactId>
            <version>${jackson.version}</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.module/jackson-
modules-java8 -->
        <dependency>
            <groupId>com.fasterxml.jackson.module</groupId>
            <artifactId>jackson-modules-java8</artifactId>
            <version>${jackson.version}</version>
            <type>pom</type>
            <scope>runtime</scope>
        </dependency>

         <!-- https://mvnrepository.com/artifact/org.json/json -->
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20180813</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-core -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-core</artifactId>
            <version>1.11.1034</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-lambda-java-core -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-lambda-java-core</artifactId>
            <version>1.2.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-lambda-java-log4j2 --
>
        <dependency>
            <groupId>com.amazonaws</groupId>
```

```
            <artifactId>aws-lambda-java-log4j2</artifactId>
            <version>1.2.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.8</version>
        </dependency>


        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.4</version>
            <scope>provided</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.17.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core --
>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.17.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-slf4j-
impl -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j-impl</artifactId>
            <version>2.17.1</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.assertj/assertj-core -->
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
```

```
            <version>3.12.2</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.5.0-M1</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>3.6.0</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-junit-jupiter -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-junit-jupiter</artifactId>
            <version>3.6.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <compilerArgs>
                        <arg>-Xlint:all,-options,-processing</arg>
                    </compilerArgs>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>2.3</version>
                <configuration>
                    <createDependencyReducedPom>false</createDependencyReducedPom>
```

```
                    <filters>
                        <filter>
                            <artifact>*:*</artifact>
                            <excludes>
                                <exclude>**/Log4j2Plugins.dat</exclude>
                            </excludes>
                        </filter>
                    </filters>
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.6.0</version>
                <executions>
                    <execution>
                        <id>generate</id>
                        <phase>generate-sources</phase>
                        <goals>
                            <goal>exec</goal>
                        </goals>
                        <configuration>
                            <executable>cfn</executable>
                            <commandlineArgs>generate ${cfn.generate.args}</
commandlineArgs>
                            <workingDirectory>${project.basedir}</workingDirectory>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>build-helper-maven-plugin</artifactId>
                <version>3.0.0</version>
                <executions>
                    <execution>
```

```
                        <id>add-source</id>
                        <phase>generate-sources</phase>
                        <goals>
                            <goal>add-source</goal>
                        </goals>
                        <configuration>
                            <sources>
                                <source>${project.basedir}/target/generated-sources/
rpdk</source>
                            </sources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>2.4</version>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M3</version>
            </plugin>
            <plugin>
                <groupId>org.jacoco</groupId>
                <artifactId>jacoco-maven-plugin</artifactId>
                <version>0.8.4</version>
                <configuration>
                    <excludes>
                        <exclude>**/BaseHookConfiguration*</exclude>
                        <exclude>**/BaseHookHandler*</exclude>
                        <exclude>**/HookHandlerWrapper*</exclude>
                        <exclude>**/ResourceModel*</exclude>
                        <exclude>**/TypeConfigurationModel*</exclude>
                        <exclude>**/model/**/*</exclude>
                    </excludes>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>prepare-agent</goal>
                        </goals>
                    </execution>
                    <execution>
```

```
                    <id>report</id>
                    <phase>test</phase>
                    <goals>
                        <goal>report</goal>
                    </goals>
                </execution>
                <execution>
                    <id>jacoco-check</id>
                    <goals>
                        <goal>check</goal>
                    </goals>
                    <configuration>
                        <rules>
                            <rule>
                                <element>PACKAGE</element>
                                <limits>
                                    <limit>
                                        <counter>BRANCH</counter>
                                        <value>COVEREDRATIO</value>
                                        <minimum>0.8</minimum>
                                    </limit>
                                    <limit>
                                        <counter>INSTRUCTION</counter>
                                        <value>COVEREDRATIO</value>
                                        <minimum>0.8</minimum>
                                    </limit>
                                </limits>
                            </rule>
                        </rules>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
    <resources>
        <resource>
            <directory>${project.basedir}</directory>
            <includes>
                <include>mycompany-testing-mytesthook.json</include>
            </includes>
        </resource>
        <resource>
            <directory>${project.basedir}/target/loaded-target-schemas</directory>
            <includes>
```

```
                    <include>**/*.json</include>
                </includes>
            </resource>
        </resources>
    </build>
</project>
```

步驟 2：產生勾點專案套件

產生您的 Hook 專案套件。CloudFormation CLI 會建立空的處理常式函數，對應至目標生命週期中的特定 Hook 動作，如 Hook 規格所定義。

```
cfn generate
```

命令會傳回下列輸出：

```
Generated files for MyCompany::Testing::MyTestHook
```

> **Note**
>
> 請確定您的 Lambda 執行時間是up-to-date，以避免使用已棄用版本。如需詳細資訊，請參閱更新資源類型和勾點的 Lambda 執行時間。

步驟 3：新增勾點處理常式

將您自己的 Hook 處理常式執行時間程式碼新增至您選擇實作的處理常式。例如，您可以新增下列程式碼進行記錄。

```
logger.log("Internal testing Hook triggered for target: " +
 request.getHookContext().getTargetName());
```

CloudFormation CLI 會產生純舊 Java 物件 (Java POJO)。以下是從 產生的輸出範例`AWS::S3::Bucket`。

Example AwsS3BucketTargetModel.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;
```

```
import...


@Data
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class AwsS3BucketTargetModel extends ResourceHookTargetModel<AwsS3Bucket> {

    @JsonIgnore
    private static final TypeReference<AwsS3Bucket> TARGET_REFERENCE =
        new TypeReference<AwsS3Bucket>() {};

    @JsonIgnore
    private static final TypeReference<AwsS3BucketTargetModel> MODEL_REFERENCE =
        new TypeReference<AwsS3BucketTargetModel>() {};

    @JsonIgnore
    public static final String TARGET_TYPE_NAME = "AWS::S3::Bucket";


    @JsonIgnore
    public TypeReference<AwsS3Bucket> getHookTargetTypeReference() {
        return TARGET_REFERENCE;
    }

    @JsonIgnore
    public TypeReference<AwsS3BucketTargetModel> getTargetModelTypeReference() {
        return MODEL_REFERENCE;
    }
}
```

Example AwsS3Bucket.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
```

```
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
@ToString(callSuper = true)
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class AwsS3Bucket extends ResourceHookTarget {
    @JsonIgnore
    public static final String TYPE_NAME = "AWS::S3::Bucket";

    @JsonIgnore
    public static final String IDENTIFIER_KEY_ID = "/properties/Id";

    @JsonProperty("InventoryConfigurations")
    private List<InventoryConfiguration> inventoryConfigurations;

    @JsonProperty("WebsiteConfiguration")
    private WebsiteConfiguration websiteConfiguration;

    @JsonProperty("DualStackDomainName")
    private String dualStackDomainName;

    @JsonProperty("AccessControl")
    private String accessControl;

    @JsonProperty("AnalyticsConfigurations")
    private List<AnalyticsConfiguration> analyticsConfigurations;

    @JsonProperty("AccelerateConfiguration")
    private AccelerateConfiguration accelerateConfiguration;

    @JsonProperty("PublicAccessBlockConfiguration")
    private PublicAccessBlockConfiguration publicAccessBlockConfiguration;

    @JsonProperty("BucketName")
    private String bucketName;

    @JsonProperty("RegionalDomainName")
    private String regionalDomainName;

    @JsonProperty("OwnershipControls")
    private OwnershipControls ownershipControls;

    @JsonProperty("ObjectLockConfiguration")
```

```
    private ObjectLockConfiguration objectLockConfiguration;

    @JsonProperty("ObjectLockEnabled")
    private Boolean objectLockEnabled;

    @JsonProperty("LoggingConfiguration")
    private LoggingConfiguration loggingConfiguration;

    @JsonProperty("ReplicationConfiguration")
    private ReplicationConfiguration replicationConfiguration;

    @JsonProperty("Tags")
    private List<Tag> tags;

    @JsonProperty("DomainName")
    private String domainName;

    @JsonProperty("BucketEncryption")
    private BucketEncryption bucketEncryption;

    @JsonProperty("WebsiteURL")
    private String websiteURL;

    @JsonProperty("NotificationConfiguration")
    private NotificationConfiguration notificationConfiguration;

    @JsonProperty("LifecycleConfiguration")
    private LifecycleConfiguration lifecycleConfiguration;

    @JsonProperty("VersioningConfiguration")
    private VersioningConfiguration versioningConfiguration;

    @JsonProperty("MetricsConfigurations")
    private List<MetricsConfiguration> metricsConfigurations;

    @JsonProperty("IntelligentTieringConfigurations")
    private List<IntelligentTieringConfiguration> intelligentTieringConfigurations;

    @JsonProperty("CorsConfiguration")
    private CorsConfiguration corsConfiguration;

    @JsonProperty("Id")
    private String id;
```

```
    @JsonProperty("Arn")
    private String arn;

    @JsonIgnore
    public JSONObject getPrimaryIdentifier() {
        final JSONObject identifier = new JSONObject();
        if (this.getId() != null) {
            identifier.put(IDENTIFIER_KEY_ID, this.getId());
        }

        // only return the identifier if it can be used, i.e. if all components are
 present
        return identifier.length() == 1 ? identifier : null;
    }

    @JsonIgnore
    public List<JSONObject> getAdditionalIdentifiers() {
        final List<JSONObject> identifiers = new ArrayList<JSONObject>();
        // only return the identifiers if any can be used
        return identifiers.isEmpty() ? null : identifiers;
    }
}
```

Example BucketEncryption.java

```
package software.amazon.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class BucketEncryption {
    @JsonProperty("ServerSideEncryptionConfiguration")
    private List<ServerSideEncryptionRule> serverSideEncryptionConfiguration;

}
```

Example ServerSideEncryptionRule.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class ServerSideEncryptionRule {
    @JsonProperty("BucketKeyEnabled")
    private Boolean bucketKeyEnabled;

    @JsonProperty("ServerSideEncryptionByDefault")
    private ServerSideEncryptionByDefault serverSideEncryptionByDefault;

}
```

Example ServerSideEncryptionByDefault.java

```
package com.mycompany.testing.mytesthook.model.aws.s3.bucket;

import ...


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@JsonAutoDetect(fieldVisibility = Visibility.ANY, getterVisibility = Visibility.NONE,
 setterVisibility = Visibility.NONE)
public class ServerSideEncryptionByDefault {
    @JsonProperty("SSEAlgorithm")
    private String sSEAlgorithm;

    @JsonProperty("KMSMasterKeyID")
    private String kMSMasterKeyID;

}
```

產生 POJOs後，您現在可以撰寫實際實作 Hook 功能的處理常式。在此範例中，請實作處理常式的 `preCreate`和 `preUpdate` 叫用點。

步驟 4：實作勾點處理常式

主題

編碼 API 用戶端建置器

1. 在您的 IDE 中，開啟位於 `src/main/java/com/mycompany/testing/mytesthook` 資料夾的 `ClientBuilder.java` 檔案。
2. 使用下列程式碼取代`ClientBuilder.java`檔案的整個內容。

   Example ClientBuilder.java

```
package com.awscommunity.kms.encryptionsettings;

import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.cloudformation.HookLambdaWrapper;

/**
 * Describes static HTTP clients (to consume less memory) for API calls that
```

```
 * this hook makes to a number of AWS services.
 */
public final class ClientBuilder {

    private ClientBuilder() {
    }

    /**
     * Create an HTTP client for Amazon EC2.
     *
     * @return Ec2Client An {@link Ec2Client} object.
     */
    public static Ec2Client getEc2Client() {
        return
 Ec2Client.builder().httpClient(HookLambdaWrapper.HTTP_CLIENT).build();
    }
}
```

編碼 API 請求建構器

1. 在您的 IDE 中，開啟位於 `src/main/java/com/mycompany/testing/mytesthook` 資料夾的 `Translator.java` 檔案。
2. 使用下列程式碼取代`Translator.java`檔案的整個內容。

   Example Translator.java

```
package com.mycompany.testing.mytesthook;


import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

/**
 * This class is a centralized placeholder for
 *  - api request construction
 *  - object translation to/from aws sdk
 */

public class Translator {
```

```
    static ListBucketsRequest translateToListBucketsRequest(final HookTargetModel
 targetModel) {
        return ListBucketsRequest.builder().build();
    }

    static ListQueuesRequest translateToListQueuesRequest(final String nextToken) {
        return ListQueuesRequest.builder().nextToken(nextToken).build();
    }

    static ListBucketsRequest createListBucketsRequest() {
        return ListBucketsRequest.builder().build();
    }

    static ListQueuesRequest createListQueuesRequest() {
        return createListQueuesRequest(null);
    }

    static ListQueuesRequest createListQueuesRequest(final String nextToken) {
        return ListQueuesRequest.builder().nextToken(nextToken).build();
    }

    static GetBucketEncryptionRequest createGetBucketEncryptionRequest(final String
 bucket) {
        return GetBucketEncryptionRequest.builder().bucket(bucket).build();
    }
}
```

實作協助程式程式碼

1. 在您的 IDE 中，開啟位於 `src/main/java/com/mycompany/testing/mytesthook` 資料夾的 `AbstractTestBase.java` 檔案。
2. 使用下列程式碼取代`AbstractTestBase.java`檔案的整個內容。

   Example Translator.java

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import org.mockito.Mockito;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsSessionCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
```

```
import software.amazon.awssdk.awscore.AwsRequest;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.awscore.AwsResponse;
import software.amazon.awssdk.core.SdkClient;
import software.amazon.awssdk.core.pagination.sync.SdkIterable;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.Credentials;
import software.amazon.cloudformation.proxy.LoggerProxy;
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import javax.annotation.Nonnull;
import java.time.Duration;
import java.util.concurrent.CompletableFuture;
import java.util.function.Function;
import java.util.function.Supplier;

import static org.assertj.core.api.Assertions.assertThat;

@lombok.Getter
public class AbstractTestBase {
    protected final AwsSessionCredentials awsSessionCredential;
    protected final AwsCredentialsProvider v2CredentialsProvider;
    protected final AwsRequestOverrideConfiguration configuration;
    protected final LoggerProxy loggerProxy;
    protected final Supplier<Long> awsLambdaRuntime = () ->
 Duration.ofMinutes(15).toMillis();
    protected final AmazonWebServicesClientProxy proxy;
    protected final Credentials mockCredentials =
        new Credentials("mockAccessId", "mockSecretKey", "mockSessionToken");

    @lombok.Setter
    private SdkClient serviceClient;

    protected AbstractTestBase() {
        loggerProxy = Mockito.mock(LoggerProxy.class);
        awsSessionCredential =
 AwsSessionCredentials.create(mockCredentials.getAccessKeyId(),
            mockCredentials.getSecretAccessKey(),
 mockCredentials.getSessionToken());
        v2CredentialsProvider =
 StaticCredentialsProvider.create(awsSessionCredential);
```

```
        configuration = AwsRequestOverrideConfiguration.builder()
            .credentialsProvider(v2CredentialsProvider)
            .build();
        proxy = new AmazonWebServicesClientProxy(
            loggerProxy,
            mockCredentials,
            awsLambdaRuntime
        ) {
            @Override
            public <ClientT> ProxyClient<ClientT> newProxy(@Nonnull
Supplier<ClientT> client) {
                return new ProxyClient<ClientT>() {
                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
AwsResponse>
                        ResponseT injectCredentialsAndInvokeV2(RequestT request,
                                                               Function<RequestT,
ResponseT> requestFunction) {
                        return proxy.injectCredentialsAndInvokeV2(request,
requestFunction);
                    }

                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
AwsResponse> CompletableFuture<ResponseT>
                        injectCredentialsAndInvokeV2Async(RequestT request,
Function<RequestT, CompletableFuture<ResponseT>> requestFunction) {
                        return proxy.injectCredentialsAndInvokeV2Async(request,
requestFunction);
                    }

                    @Override
                    public <RequestT extends AwsRequest, ResponseT extends
AwsResponse, IterableT extends SdkIterable<ResponseT>>
                        IterableT
                        injectCredentialsAndInvokeIterableV2(RequestT request,
Function<RequestT, IterableT> requestFunction) {
                        return proxy.injectCredentialsAndInvokeIterableV2(request,
requestFunction);
                    }

                    @SuppressWarnings("unchecked")
                    @Override
                    public ClientT client() {
```

```
                        return (ClientT) serviceClient;
                    }
                };
            }
        };
    }

    protected void assertResponse(final ProgressEvent<HookTargetModel,
 CallbackContext> response, final OperationStatus expectedStatus, final String
 expectedMsg) {
        assertThat(response).isNotNull();
        assertThat(response.getStatus()).isEqualTo(expectedStatus);
        assertThat(response.getCallbackContext()).isNull();
        assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
        assertThat(response.getMessage()).isNotNull();
        assertThat(response.getMessage()).isEqualTo(expectedMsg);
    }

    protected HookTargetModel createHookTargetModel(final Object
 resourceProperties) {
        return HookTargetModel.of(ImmutableMap.of("ResourceProperties",
 resourceProperties));
    }

    protected HookTargetModel createHookTargetModel(final Object
 resourceProperties, final Object previousResourceProperties) {
        return HookTargetModel.of(
                ImmutableMap.of(
                    "ResourceProperties", resourceProperties,
                    "PreviousResourceProperties", previousResourceProperties
                )
        );
    }
}
```

實作基本處理常式

1. 在您的 IDE 中，開啟位於 src/main/java/com/mycompany/testing/mytesthook 資料夾的 BaseHookHandlerStd.java 檔案。
2. 使用下列程式碼取代BaseHookHandlerStd.java檔案的整個內容。

Example Translator.java

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;


public abstract class BaseHookHandlerStd extends BaseHookHandler<CallbackContext,
 TypeConfigurationModel> {
    public static final String HOOK_TYPE_NAME = "MyCompany::Testing::MyTestHook";

    protected Logger logger;

    @Override
    public ProgressEvent<HookTargetModel, CallbackContext> handleRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final Logger logger,
            final TypeConfigurationModel typeConfiguration
    ) {
        this.logger = logger;

        final String targetName = request.getHookContext().getTargetName();

        final ProgressEvent<HookTargetModel, CallbackContext> result;
        if (AwsS3Bucket.TYPE_NAME.equals(targetName)) {
            result = handleS3BucketRequest(
                    proxy,
                    request,
                    callbackContext != null ? callbackContext : new
 CallbackContext(),
                    proxy.newProxy(ClientBuilder::createS3Client),
```

```
                    typeConfiguration
            );
        } else if (AwsSqsQueue.TYPE_NAME.equals(targetName)) {
            result = handleSqsQueueRequest(
                    proxy,
                    request,
                    callbackContext != null ? callbackContext : new
CallbackContext(),
                    proxy.newProxy(ClientBuilder::createSqsClient),
                    typeConfiguration
            );
        } else {
            throw new UnsupportedTargetException(targetName);
        }

        log(
            String.format(
                "Result for [%s] invocation for target [%s] returned status [%s]
with message [%s]",
                request.getHookContext().getInvocationPoint(),
                targetName,
                result.getStatus(),
                result.getMessage()
            )
        );

        return result;
    }

    protected abstract ProgressEvent<HookTargetModel, CallbackContext>
handleS3BucketRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<S3Client> proxyClient,
            final TypeConfigurationModel typeConfiguration
    );

    protected abstract ProgressEvent<HookTargetModel, CallbackContext>
handleSqsQueueRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<SqsClient> proxyClient,
```

```
            final TypeConfigurationModel typeConfiguration
    );

    protected void log(final String message) {
        if (logger != null) {
            logger.log(message);
        } else {
            System.out.println(message);
        }
    }
}
```

實作**preCreate**處理常式

preCreate 處理常式會驗證 AWS::S3::Bucket或 AWS::SQS::Queue 資源的伺服器端加密設定。

- 對於 AWS::S3::Bucket 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - Amazon S3 儲存貯體加密已設定。
  - 已啟用儲存貯體的 Amazon S3 儲存貯體金鑰。
  - Amazon S3 儲存貯體的加密演算法集是所需的正確演算法。
  - AWS Key Management Service 金鑰 ID 已設定。
- 對於 AWS::SQS::Queue 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - AWS Key Management Service 金鑰 ID 已設定。

編碼**preCreate**處理常式

1. 在您的 IDE 中，開啟位於 src/main/java/software/mycompany/testing/mytesthook 資料夾的 PreCreateHookHandler.java 檔案。
2. 使用下列程式碼取代PreCreateHookHandler.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
```

```
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueueTargetModel;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.List;

public class PreCreateHookHandler extends BaseHookHandler<TypeConfigurationModel,
 CallbackContext> {

    @Override
    public HookProgressEvent<CallbackContext> handleRequest(
        final AmazonWebServicesClientProxy proxy,
        final HookHandlerRequest request,
        final CallbackContext callbackContext,
        final Logger logger,
        final TypeConfigurationModel typeConfiguration) {

        final String targetName = request.getHookContext().getTargetName();
        if ("AWS::S3::Bucket".equals(targetName)) {
            final ResourceHookTargetModel<AwsS3Bucket> targetModel =
 request.getHookContext().getTargetModel(AwsS3BucketTargetModel.class);

            final AwsS3Bucket bucket = targetModel.getResourceProperties();
            final String encryptionAlgorithm =
 typeConfiguration.getEncryptionAlgorithm();

            return validateS3BucketEncryption(bucket, encryptionAlgorithm);

        } else if ("AWS::SQS::Queue".equals(targetName)) {
            final ResourceHookTargetModel<AwsSqsQueue> targetModel =
 request.getHookContext().getTargetModel(AwsSqsQueueTargetModel.class);
```

```
            final AwsSqsQueue queue = targetModel.getResourceProperties();
            return validateSQSQueueEncryption(queue);
        } else {
            throw new UnsupportedTargetException(targetName);
        }
    }

    private HookProgressEvent<CallbackContext> validateS3BucketEncryption(final
AwsS3Bucket bucket, final String requiredEncryptionAlgorithm) {
        HookStatus resultStatus = null;
        String resultMessage = null;

        if (bucket != null) {
            final BucketEncryption bucketEncryption = bucket.getBucketEncryption();
            if (bucketEncryption != null) {
                final List<ServerSideEncryptionRule> serverSideEncryptionRules =
bucketEncryption.getServerSideEncryptionConfiguration();
                if (CollectionUtils.isNotEmpty(serverSideEncryptionRules)) {
                    for (final ServerSideEncryptionRule rule :
serverSideEncryptionRules) {
                        final Boolean bucketKeyEnabled =
rule.getBucketKeyEnabled();
                        if (bucketKeyEnabled) {
                            final ServerSideEncryptionByDefault
serverSideEncryptionByDefault = rule.getServerSideEncryptionByDefault();

                            final String encryptionAlgorithm =
serverSideEncryptionByDefault.getSSEAlgorithm();
                            final String kmsKeyId =
serverSideEncryptionByDefault.getKMSMasterKeyID(); // "KMSMasterKeyID" is name of
the property for an AWS::S3::Bucket;

                            if (!StringUtils.equals(encryptionAlgorithm,
requiredEncryptionAlgorithm) && StringUtils.isBlank(kmsKeyId)) {
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "KMS Key ID not set
and SSE Encryption Algorithm is incorrect for bucket with name: " +
bucket.getBucketName();
                            } else if (!StringUtils.equals(encryptionAlgorithm,
requiredEncryptionAlgorithm)) {
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "SSE Encryption Algorithm is
incorrect for bucket with name: " + bucket.getBucketName();
                            } else if (StringUtils.isBlank(kmsKeyId)) {
```

```
                                resultStatus = HookStatus.FAILED;
                                resultMessage = "KMS Key ID not set for bucket with
name: " + bucket.getBucketName();
                            } else {
                                resultStatus = HookStatus.SUCCESS;
                                resultMessage = "Successfully invoked
PreCreateHookHandler for target: AWS::S3::Bucket";
                            }
                        } else {
                            resultStatus = HookStatus.FAILED;
                            resultMessage = "Bucket key not enabled for bucket with
name: " + bucket.getBucketName();
                        }

                        if (resultStatus == HookStatus.FAILED) {
                            break;
                        }
                    }
                } else {
                    resultStatus = HookStatus.FAILED;
                    resultMessage = "No SSE Encryption configurations for bucket
with name: " + bucket.getBucketName();
                }
            } else {
                resultStatus = HookStatus.FAILED;
                resultMessage = "Bucket Encryption not enabled for bucket with
name: " + bucket.getBucketName();
            }
        } else {
            resultStatus = HookStatus.FAILED;
            resultMessage = "Resource properties for S3 Bucket target model are
empty";
        }

        return HookProgressEvent.<CallbackContext>builder()
                .status(resultStatus)
                .message(resultMessage)
                .errorCode(resultStatus == HookStatus.FAILED ?
HandlerErrorCode.ResourceConflict : null)
                .build();
    }

    private HookProgressEvent<CallbackContext> validateSQSQueueEncryption(final
AwsSqsQueue queue) {
```

```
        if (queue == null) {
            return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.FAILED)
                    .message("Resource properties for SQS Queue target model are
 empty")
                    .errorCode(HandlerErrorCode.ResourceConflict)
                    .build();
        }

        final String kmsKeyId = queue.getKmsMasterKeyId(); // "KmsMasterKeyId" is
 name of the property for an AWS::SQS::Queue
        if (StringUtils.isBlank(kmsKeyId)) {
            return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.FAILED)
                    .message("Server side encryption turned off for queue with
 name: " + queue.getQueueName())
                    .errorCode(HandlerErrorCode.ResourceConflict)
                    .build();
        }

        return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.SUCCESS)
                    .message("Successfully invoked PreCreateHookHandler for target:
 AWS::SQS::Queue")
                    .build();
    }
}
```

更新**preCreate**測試

1. 在您的 IDE 中，開啟位於 src/test/java/software/mycompany/testing/mytesthook 資料夾的 PreCreateHandlerTest.java 檔案。
2. 使用下列程式碼取代PreCreateHandlerTest.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
```

```
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Collections;
import java.util.Map;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.assertThatExceptionOfType;
import static org.mockito.Mockito.mock;

@ExtendWith(MockitoExtension.class)
public class PreCreateHookHandlerTest {

    @Mock
    private AmazonWebServicesClientProxy proxy;

    @Mock
    private Logger logger;

    @BeforeEach
    public void setup() {
        proxy = mock(AmazonWebServicesClientProxy.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsSqsQueueSuccess() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();
```

```
        final AwsSqsQueue queue = buildSqsQueue("MyQueue", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(queue);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::SQS::Queue").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
PreCreateHookHandler for target: AWS::SQS::Queue");
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"AES256", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(bucket);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
PreCreateHookHandler for target: AWS::S3::Bucket");
    }

    @Test
    public void handleRequest_awsS3BucketFail_bucketKeyNotEnabled() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", false,
"AES256", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(bucket);
```

```
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "Bucket key not enabled for
bucket with name: amzn-s3-demo-bucket");
    }

    @Test
    public void handleRequest_awsS3BucketFail_incorrectSSEEncryptionAlgorithm() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"SHA512", "KmsKey");
        final HookTargetModel targetModel = createHookTargetModel(bucket);
       final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "SSE Encryption Algorithm is
incorrect for bucket with name: amzn-s3-demo-bucket");
    }

    @Test
    public void handleRequest_awsS3BucketFail_kmsKeyIdNotSet() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsS3Bucket bucket = buildAwsS3Bucket("amzn-s3-demo-bucket", true,
"AES256", null);
        final HookTargetModel targetModel = createHookTargetModel(bucket);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();
```

```
        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "KMS Key ID not set for bucket
with name: amzn-s3-demo-bucket");
    }

    @Test
    public void handleRequest_awsSqsQueueFail_serverSideEncryptionOff() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final AwsSqsQueue queue = buildSqsQueue("MyQueue", null);
        final HookTargetModel targetModel = createHookTargetModel(queue);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::SQS::Queue").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "Server side encryption turned
off for queue with name: MyQueue");
    }

    @Test
    public void handleRequest_unsupportedTarget() {
        final PreCreateHookHandler handler = new PreCreateHookHandler();

        final Map<String, Object> unsupportedTarget =
ImmutableMap.of("ResourceName", "MyUnsupportedTarget");
        final HookTargetModel targetModel =
createHookTargetModel(unsupportedTarget);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
```

```
.hookContext(HookContext.builder().targetName("AWS::Unsupported::Target").targetModel(targ
            .build();

        assertThatExceptionOfType(UnsupportedTargetException.class)
                .isThrownBy(() -> handler.handleRequest(proxy, request, null,
logger, typeConfiguration))
                .withMessageContaining("Unsupported target")
                .withMessageContaining("AWS::Unsupported::Target")
                .satisfies(e ->
assertThat(e.getErrorCode()).isEqualTo(HandlerErrorCode.InvalidRequest));
    }

    private void assertResponse(final HookProgressEvent<CallbackContext> response,
final HookStatus expectedStatus, final String expectedErrorMsg) {
        assertThat(response).isNotNull();
        assertThat(response.getStatus()).isEqualTo(expectedStatus);
        assertThat(response.getCallbackContext()).isNull();
        assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
        assertThat(response.getMessage()).isNotNull();
        assertThat(response.getMessage()).isEqualTo(expectedErrorMsg);
    }

    private HookTargetModel createHookTargetModel(final Object resourceProperties)
{
        return HookTargetModel.of(ImmutableMap.of("ResourceProperties",
resourceProperties));
    }

    @SuppressWarnings("SameParameterValue")
    private AwsSqsQueue buildSqsQueue(final String queueName, final String
kmsKeyId) {
        return AwsSqsQueue.builder()
                .queueName(queueName)
                .kmsMasterKeyId(kmsKeyId) // "KmsMasterKeyId" is name of the
property for an AWS::SQS::Queue
                .build();
    }

    @SuppressWarnings("SameParameterValue")
    private AwsS3Bucket buildAwsS3Bucket(
            final String bucketName,
            final Boolean bucketKeyEnabled,
            final String sseAlgorithm,
```

```
            final String kmsKeyId
    ) {
        return AwsS3Bucket.builder()
                .bucketName(bucketName)
                .bucketEncryption(
                    BucketEncryption.builder()
                        .serverSideEncryptionConfiguration(
                            Collections.singletonList(
                                ServerSideEncryptionRule.builder()
                                    .bucketKeyEnabled(bucketKeyEnabled)
                                    .serverSideEncryptionByDefault(
                                        ServerSideEncryptionByDefault.builder()
                                            .sSEAlgorithm(sseAlgorithm)
                                            .kMSMasterKeyID(kmsKeyId) //
 "KMSMasterKeyID" is name of the property for an AWS::S3::Bucket
                                            .build()
                                    ).build()
                            )
                        ).build()
                ).build();
    }
}
```

實作**preUpdate**處理常式

實作preUpdate處理常式，其會在處理常式中所有指定目標的更新操作之前啟動。preUpdate 處理常式會完成下列項目：

- 對於 AWS::S3::Bucket 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - Amazon S3 儲存貯體的儲存貯體加密演算法尚未修改。

編碼**preUpdate**處理常式

1. 在您的 IDE 中，開啟位於 src/main/java/software/mycompany/testing/mytesthook 資料夾的 PreUpdateHookHandler.java 檔案。
2. 使用下列程式碼取代PreUpdateHookHandler.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
```

```
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import org.apache.commons.lang3.StringUtils;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.List;

public class PreUpdateHookHandler extends BaseHookHandler<TypeConfigurationModel,
 CallbackContext> {

    @Override
    public HookProgressEvent<CallbackContext> handleRequest(
        final AmazonWebServicesClientProxy proxy,
        final HookHandlerRequest request,
        final CallbackContext callbackContext,
        final Logger logger,
        final TypeConfigurationModel typeConfiguration) {

        final String targetName = request.getHookContext().getTargetName();
        if ("AWS::S3::Bucket".equals(targetName)) {
            final ResourceHookTargetModel<AwsS3Bucket> targetModel =
 request.getHookContext().getTargetModel(AwsS3BucketTargetModel.class);

            final AwsS3Bucket bucketProperties =
 targetModel.getResourceProperties();
            final AwsS3Bucket previousBucketProperties =
 targetModel.getPreviousResourceProperties();

            return validateBucketEncryptionRulesNotUpdated(bucketProperties,
 previousBucketProperties);
        } else {
            throw new UnsupportedTargetException(targetName);
        }
    }
```

```
   private HookProgressEvent<CallbackContext>
validateBucketEncryptionRulesNotUpdated(final AwsS3Bucket resourceProperties,
final AwsS3Bucket previousResourceProperties) {
       final List<ServerSideEncryptionRule> bucketEncryptionConfigs =
resourceProperties.getBucketEncryption().getServerSideEncryptionConfiguration();
       final List<ServerSideEncryptionRule> previousBucketEncryptionConfigs =
previousResourceProperties.getBucketEncryption().getServerSideEncryptionConfiguration();

       if (bucketEncryptionConfigs.size() !=
previousBucketEncryptionConfigs.size()) {
           return HookProgressEvent.<CallbackContext>builder()
                   .status(HookStatus.FAILED)
                   .errorCode(HandlerErrorCode.NotUpdatable)
                   .message(
                       String.format(
                           "Current number of bucket encryption configs does not
match previous. Current has %d configs while previously there were %d configs",
                           bucketEncryptionConfigs.size(),
                           previousBucketEncryptionConfigs.size()
                       )
                   ).build();
       }

       for (int i = 0; i < bucketEncryptionConfigs.size(); ++i) {
           final String currentEncryptionAlgorithm =
bucketEncryptionConfigs.get(i).getServerSideEncryptionByDefault().getSSEAlgorithm();
           final String previousEncryptionAlgorithm =
previousBucketEncryptionConfigs.get(i).getServerSideEncryptionByDefault().getSSEAlgorithm(

           if (!StringUtils.equals(currentEncryptionAlgorithm,
previousEncryptionAlgorithm)) {
               return HookProgressEvent.<CallbackContext>builder()
                   .status(HookStatus.FAILED)
                   .errorCode(HandlerErrorCode.NotUpdatable)
                   .message(
                       String.format(
                           "Bucket Encryption algorithm can not be changed once
set. The encryption algorithm was changed to '%s' from '%s'.",
                           currentEncryptionAlgorithm,
                           previousEncryptionAlgorithm
                       )
                   )
                   .build();
           }
```

```
        }

        return HookProgressEvent.<CallbackContext>builder()
                    .status(HookStatus.SUCCESS)
                    .message("Successfully invoked PreUpdateHookHandler for target:
 AWS::SQS::Queue")
                    .build();
    }
}
```

更新**preUpdate**測試

1. 在您的 IDE 中，開啟 src/main/java/com/mycompany/testing/mytesthook 資料夾中PreUpdateHandlerTest.java的檔案。
2. 使用下列程式碼取代PreUpdateHandlerTest.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.BucketEncryption;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionByDefault;
import
 com.mycompany.testing.mytesthook.model.aws.s3.bucket.ServerSideEncryptionRule;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.cloudformation.exceptions.UnsupportedTargetException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.Logger;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.HookProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookStatus;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Arrays;
import java.util.stream.Stream;
```

```
import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.assertThatExceptionOfType;
import static org.mockito.Mockito.mock;

@ExtendWith(MockitoExtension.class)
public class PreUpdateHookHandlerTest {

    @Mock
    private AmazonWebServicesClientProxy proxy;

    @Mock
    private Logger logger;

    @BeforeEach
    public void setup() {
        proxy = mock(AmazonWebServicesClientProxy.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final ServerSideEncryptionRule serverSideEncryptionRule =
 buildServerSideEncryptionRule("AES256");
        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", serverSideEncryptionRule);
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", serverSideEncryptionRule);
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.SUCCESS, "Successfully invoked
 PreUpdateHookHandler for target: AWS::SQS::Queue");
```

```
    }

    @Test
    public void handleRequest_awsS3BucketFail_bucketEncryptionConfigsDontMatch() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final ServerSideEncryptionRule[] serverSideEncryptionRules =
 Stream.of("AES256", "SHA512", "AES32")
                .map(this::buildServerSideEncryptionRule)
                .toArray(ServerSideEncryptionRule[]::new);

        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", serverSideEncryptionRules[0]);
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", serverSideEncryptionRules);
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

 .hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, "Current number of bucket
 encryption configs does not match previous. Current has 1 configs while previously
 there were 3 configs");
    }

    @Test
    public void
 handleRequest_awsS3BucketFail_bucketEncryptionAlgorithmDoesNotMatch() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final AwsS3Bucket resourceProperties = buildAwsS3Bucket("amzn-s3-demo-
bucket", buildServerSideEncryptionRule("SHA512"));
        final AwsS3Bucket previousResourceProperties = buildAwsS3Bucket("amzn-s3-
demo-bucket", buildServerSideEncryptionRule("AES256"));
        final HookTargetModel targetModel =
 createHookTargetModel(resourceProperties, previousResourceProperties);
```

```
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::S3::Bucket").targetModel(targetModel).
            .build();

        final HookProgressEvent<CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);
        assertResponse(response, HookStatus.FAILED, String.format("Bucket
Encryption algorithm can not be changed once set. The encryption algorithm was
changed to '%s' from '%s'.", "SHA512", "AES256"));
    }

    @Test
    public void handleRequest_unsupportedTarget() {
        final PreUpdateHookHandler handler = new PreUpdateHookHandler();

        final Object resourceProperties = ImmutableMap.of("FileSizeLimit", 256);
        final Object previousResourceProperties = ImmutableMap.of("FileSizeLimit",
512);
        final HookTargetModel targetModel =
createHookTargetModel(resourceProperties, previousResourceProperties);
        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder().encryptionAlgorithm("AES256").build();

        final HookHandlerRequest request = HookHandlerRequest.builder()

.hookContext(HookContext.builder().targetName("AWS::Unsupported::Target").targetModel(targ
            .build();

        assertThatExceptionOfType(UnsupportedTargetException.class)
                .isThrownBy(() -> handler.handleRequest(proxy, request, null,
logger, typeConfiguration))
                .withMessageContaining("Unsupported target")
                .withMessageContaining("AWS::Unsupported::Target")
                .satisfies(e ->
assertThat(e.getErrorCode()).isEqualTo(HandlerErrorCode.InvalidRequest));
    }

    private void assertResponse(final HookProgressEvent<CallbackContext> response,
final HookStatus expectedStatus, final String expectedErrorMsg) {
        assertThat(response).isNotNull();
```

```
        assertThat(response.getStatus()).isEqualTo(expectedStatus);
        assertThat(response.getCallbackContext()).isNull();
        assertThat(response.getCallbackDelaySeconds()).isEqualTo(0);
        assertThat(response.getMessage()).isNotNull();
        assertThat(response.getMessage()).isEqualTo(expectedErrorMsg);
    }

    private HookTargetModel createHookTargetModel(final Object resourceProperties,
 final Object previousResourceProperties) {
        return HookTargetModel.of(
                ImmutableMap.of(
                    "ResourceProperties", resourceProperties,
                    "PreviousResourceProperties", previousResourceProperties
                )
        );
    }

    @SuppressWarnings("SameParameterValue")
    private AwsS3Bucket buildAwsS3Bucket(
            final String bucketName,
            final ServerSideEncryptionRule ...serverSideEncryptionRules
            ) {
        return AwsS3Bucket.builder()
                .bucketName(bucketName)
                .bucketEncryption(
                    BucketEncryption.builder()
                        .serverSideEncryptionConfiguration(
                                Arrays.asList(serverSideEncryptionRules)
                        ).build()
                ).build();
    }

    private ServerSideEncryptionRule buildServerSideEncryptionRule(final String
 encryptionAlgorithm) {
        return ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .serverSideEncryptionByDefault(
                        ServerSideEncryptionByDefault.builder()
                                .sSEAlgorithm(encryptionAlgorithm)
                                .build()
                ).build();
    }
}
```

實作**preDelete**處理常式

實作preDelete處理常式，其會在處理常式中所有指定目標的刪除操作之前啟動。preDelete 處理常式會完成下列項目：

- 對於 AWS::S3::Bucket 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - 驗證刪除資源後，帳戶是否將存在最低必要投訴資源。
  - 最低必要投訴資源數量是在 Hook 的類型組態中設定。

編碼**preDelete**處理常式

1. 在您的 IDE 中，開啟 src/main/java/com/mycompany/testing/mytesthook 資料夾中PreDeleteHookHandler.java的檔案。
2. 使用下列程式碼取代PreDeleteHookHandler.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.google.common.annotations.VisibleForTesting;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3BucketTargetModel;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueue;
import com.mycompany.testing.mytesthook.model.aws.sqs.queue.AwsSqsQueueTargetModel;
import org.apache.commons.lang3.StringUtils;
import org.apache.commons.lang3.math.NumberUtils;
import software.amazon.awssdk.services.cloudformation.CloudFormationClient;
import
 software.amazon.awssdk.services.cloudformation.model.CloudFormationException;
import
 software.amazon.awssdk.services.cloudformation.model.DescribeStackResourceRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.cloudformation.exceptions.CfnGeneralServiceException;
import software.amazon.cloudformation.proxy.AmazonWebServicesClientProxy;
```

```
import software.amazon.cloudformation.proxy.HandlerErrorCode;
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.ProxyClient;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;
import
 software.amazon.cloudformation.proxy.hook.targetmodel.ResourceHookTargetModel;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;

public class PreDeleteHookHandler extends BaseHookHandlerStd {

    private ProxyClient<S3Client> s3Client;
    private ProxyClient<SqsClient> sqsClient;

    @Override
    protected ProgressEvent<HookTargetModel, CallbackContext>
 handleS3BucketRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<S3Client> proxyClient,
            final TypeConfigurationModel typeConfiguration
    ) {
        final HookContext hookContext = request.getHookContext();
        final String targetName = hookContext.getTargetName();
        if (!AwsS3Bucket.TYPE_NAME.equals(targetName)) {
            throw new RuntimeException(String.format("Request target type [%s] is
 not 'AWS::S3::Bucket'", targetName));
        }
        this.s3Client = proxyClient;

        final String encryptionAlgorithm =
 typeConfiguration.getEncryptionAlgorithm();
        final int minBuckets =
 NumberUtils.toInt(typeConfiguration.getMinBuckets());
```

```
        final ResourceHookTargetModel<AwsS3Bucket> targetModel =
hookContext.getTargetModel(AwsS3BucketTargetModel.class);
        final List<String> buckets = listBuckets().stream()
                .filter(b -> !StringUtils.equals(b,
targetModel.getResourceProperties().getBucketName()))
                .collect(Collectors.toList());

        final List<String> compliantBuckets = new ArrayList<>();
        for (final String bucket : buckets) {
            if (getBucketSSEAlgorithm(bucket).contains(encryptionAlgorithm)) {
                compliantBuckets.add(bucket);
            }

            if (compliantBuckets.size() >= minBuckets) {
                return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                        .status(OperationStatus.SUCCESS)
                        .message("Successfully invoked PreDeleteHookHandler for
target: AWS::S3::Bucket")
                        .build();
            }
        }

        return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                .status(OperationStatus.FAILED)
                .errorCode(HandlerErrorCode.NonCompliant)
                .message(String.format("Failed to meet minimum of [%d] encrypted
buckets.", minBuckets))
                .build();
    }

    @Override
    protected ProgressEvent<HookTargetModel, CallbackContext>
handleSqsQueueRequest(
            final AmazonWebServicesClientProxy proxy,
            final HookHandlerRequest request,
            final CallbackContext callbackContext,
            final ProxyClient<SqsClient> proxyClient,
            final TypeConfigurationModel typeConfiguration
    ) {
        final HookContext hookContext = request.getHookContext();
        final String targetName = hookContext.getTargetName();
        if (!AwsSqsQueue.TYPE_NAME.equals(targetName)) {
            throw new RuntimeException(String.format("Request target type [%s] is
not 'AWS::SQS::Queue'", targetName));
```

```
        }
        this.sqsClient = proxyClient;
        final int minQueues = NumberUtils.toInt(typeConfiguration.getMinQueues());

        final ResourceHookTargetModel<AwsSqsQueue> targetModel =
hookContext.getTargetModel(AwsSqsQueueTargetModel.class);

        final String queueName =
Objects.toString(targetModel.getResourceProperties().get("QueueName"), null);

        String targetQueueUrl = null;
        if (queueName != null) {
            try {
                targetQueueUrl = sqsClient.injectCredentialsAndInvokeV2(
                        GetQueueUrlRequest.builder().queueName(
                                queueName
                        ).build(),
                        sqsClient.client()::getQueueUrl
                ).queueUrl();
            } catch (SqsException e) {
                log(String.format("Error while calling GetQueueUrl API for queue
name [%s]: %s", queueName, e.getMessage()));
            }
        } else {
            log("Queue name is empty, attempting to get queue's physical ID");
            try {
                final ProxyClient<CloudFormationClient> cfnClient =
proxy.newProxy(ClientBuilder::createCloudFormationClient);
                targetQueueUrl = cfnClient.injectCredentialsAndInvokeV2(
                        DescribeStackResourceRequest.builder()
                                .stackName(hookContext.getTargetLogicalId())

.logicalResourceId(hookContext.getTargetLogicalId())
                                .build(),
                        cfnClient.client()::describeStackResource
                ).stackResourceDetail().physicalResourceId();
            } catch (CloudFormationException e) {
                log(String.format("Error while calling DescribeStackResource API
for queue name: %s", e.getMessage()));
            }
        }

        // Creating final variable for the filter lambda
        final String finalTargetQueueUrl = targetQueueUrl;
```

```
        final List<String> compliantQueues = new ArrayList<>();

        String nextToken = null;
        do {
            final ListQueuesRequest req =
Translator.createListQueuesRequest(nextToken);
            final ListQueuesResponse res =
sqsClient.injectCredentialsAndInvokeV2(req, sqsClient.client()::listQueues);
            final List<String> queueUrls = res.queueUrls().stream()
                    .filter(q -> !StringUtils.equals(q, finalTargetQueueUrl))
                    .collect(Collectors.toList());

            for (final String queueUrl : queueUrls) {
                if (isQueueEncrypted(queueUrl)) {
                    compliantQueues.add(queueUrl);
                }

                if (compliantQueues.size() >= minQueues) {
                    return ProgressEvent.<HookTargetModel,
CallbackContext>builder()
                            .status(OperationStatus.SUCCESS)
                            .message("Successfully invoked PreDeleteHookHandler for
target: AWS::SQS::Queue")
                            .build();
                }
                nextToken = res.nextToken();
            }
        } while (nextToken != null);

        return ProgressEvent.<HookTargetModel, CallbackContext>builder()
                .status(OperationStatus.FAILED)
                .errorCode(HandlerErrorCode.NonCompliant)
                .message(String.format("Failed to meet minimum of [%d] encrypted
queues.", minQueues))
                .build();
    }

    private List<String> listBuckets() {
        try {
            return
s3Client.injectCredentialsAndInvokeV2(Translator.createListBucketsRequest(),
s3Client.client()::listBuckets)
                    .buckets()
```

```
                    .stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());
        } catch (S3Exception e) {
            throw new CfnGeneralServiceException("Error while calling S3
ListBuckets API", e);
        }
    }

    @VisibleForTesting
    Collection<String> getBucketSSEAlgorithm(final String bucket) {
        try {
            return
s3Client.injectCredentialsAndInvokeV2(Translator.createGetBucketEncryptionRequest(bucket),
s3Client.client()::getBucketEncryption)
                    .serverSideEncryptionConfiguration()
                    .rules()
                    .stream()
                    .filter(r ->
Objects.nonNull(r.applyServerSideEncryptionByDefault()))
                    .map(r ->
r.applyServerSideEncryptionByDefault().sseAlgorithmAsString())
                    .collect(Collectors.toSet());
        } catch (S3Exception e) {
            return new HashSet<>();
        }
    }

    @VisibleForTesting
    boolean isQueueEncrypted(final String queueUrl) {
        try {
            final GetQueueAttributesRequest request =
GetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributeNames(QueueAttributeName.KMS_MASTER_KEY_ID)
                    .build();
            final String kmsKeyId = sqsClient.injectCredentialsAndInvokeV2(request,
sqsClient.client()::getQueueAttributes)
                    .attributes()
                    .get(QueueAttributeName.KMS_MASTER_KEY_ID);

            return StringUtils.isNotBlank(kmsKeyId);
        } catch (SqsException e) {
```

```
            throw new CfnGeneralServiceException("Error while calling SQS
 GetQueueAttributes API", e);
        }
    }
}
```

更新**preDelete**處理常式

1. 在您的 IDE 中，開啟 src/main/java/com/mycompany/testing/mytesthook 資料夾中PreDeleteHookHandler.java的檔案。
2. 使用下列程式碼取代PreDeleteHookHandler.java檔案的整個內容。

```
package com.mycompany.testing.mytesthook;

import com.google.common.collect.ImmutableList;
import com.google.common.collect.ImmutableMap;
import com.mycompany.testing.mytesthook.model.aws.s3.bucket.AwsS3Bucket;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.cloudformation.proxy.Logger;
```

```
import software.amazon.cloudformation.proxy.OperationStatus;
import software.amazon.cloudformation.proxy.ProgressEvent;
import software.amazon.cloudformation.proxy.hook.HookContext;
import software.amazon.cloudformation.proxy.hook.HookHandlerRequest;
import software.amazon.cloudformation.proxy.hook.targetmodel.HookTargetModel;

import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.never;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

@ExtendWith(MockitoExtension.class)
public class PreDeleteHookHandlerTest extends AbstractTestBase {

    @Mock private S3Client s3Client;
    @Mock private SqsClient sqsClient;
    @Mock private Logger logger;

    @BeforeEach
    public void setup() {
        s3Client = mock(S3Client.class);
        sqsClient = mock(SqsClient.class);
        logger = mock(Logger.class);
    }

    @Test
    public void handleRequest_awsS3BucketSuccess() {
        final PreDeleteHookHandler handler = Mockito.spy(new
 PreDeleteHookHandler());

        final List<Bucket> bucketList = ImmutableList.of(
                Bucket.builder().name("bucket1").build(),
                Bucket.builder().name("bucket2").build(),
                Bucket.builder().name("toBeDeletedBucket").build(),
                Bucket.builder().name("bucket3").build(),
                Bucket.builder().name("bucket4").build(),
```

```
                Bucket.builder().name("bucket5").build()
        );
        final ListBucketsResponse mockResponse =
ListBucketsResponse.builder().buckets(bucketList).build();

when(s3Client.listBuckets(any(ListBucketsRequest.class))).thenReturn(mockResponse);
        when(s3Client.getBucketEncryption(any(GetBucketEncryptionRequest.class)))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256", "aws:kms"))
                .thenThrow(S3Exception.builder().message("No Encrypt").build())
                .thenReturn(buildGetBucketEncryptionResponse("aws:kms"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"));
        setServiceClient(s3Client);

        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder()
                .encryptionAlgorithm("AES256")
                .minBuckets("3")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::S3::Bucket")
                        .targetModel(
                            createHookTargetModel(
                                AwsS3Bucket.builder()
                                    .bucketName("toBeDeletedBucket")
                                    .build()
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(s3Client,
times(5)).getBucketEncryption(any(GetBucketEncryptionRequest.class));
        verify(handler, never()).getBucketSSEAlgorithm("toBeDeletedBucket");

        assertResponse(response, OperationStatus.SUCCESS, "Successfully invoked
PreDeleteHookHandler for target: AWS::S3::Bucket");
    }
```

```
    @Test
    public void handleRequest_awsSqsQueueSuccess() {
        final PreDeleteHookHandler handler = Mockito.spy(new
 PreDeleteHookHandler());

        final List<String> queueUrls = ImmutableList.of(
                "https://queue1.queue",
                "https://queue2.queue",
                "https://toBeDeletedQueue.queue",
                "https://queue3.queue",
                "https://queue4.queue",
                "https://queue5.queue"
        );

        when(sqsClient.getQueueUrl(any(GetQueueUrlRequest.class)))
                .thenReturn(GetQueueUrlResponse.builder().queueUrl("https://
toBeDeletedQueue.queue").build());
        when(sqsClient.listQueues(any(ListQueuesRequest.class)))

 .thenReturn(ListQueuesResponse.builder().queueUrls(queueUrls).build());
        when(sqsClient.getQueueAttributes(any(GetQueueAttributesRequest.class)))

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build());
        setServiceClient(sqsClient);

        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder()
                .minQueues("3")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
```

```
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::SQS::Queue")
                        .targetModel(
                            createHookTargetModel(
                                ImmutableMap.of("QueueName", "toBeDeletedQueue")
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(sqsClient,
times(5)).getQueueAttributes(any(GetQueueAttributesRequest.class));
        verify(handler, never()).isQueueEncrypted("toBeDeletedQueue");

        assertResponse(response, OperationStatus.SUCCESS, "Successfully invoked
PreDeleteHookHandler for target: AWS::SQS::Queue");
    }

    @Test
    public void handleRequest_awsS3BucketFailed() {
        final PreDeleteHookHandler handler = Mockito.spy(new
PreDeleteHookHandler());

        final List<Bucket> bucketList = ImmutableList.of(
                Bucket.builder().name("bucket1").build(),
                Bucket.builder().name("bucket2").build(),
                Bucket.builder().name("toBeDeletedBucket").build(),
                Bucket.builder().name("bucket3").build(),
                Bucket.builder().name("bucket4").build(),
                Bucket.builder().name("bucket5").build()
        );
        final ListBucketsResponse mockResponse =
ListBucketsResponse.builder().buckets(bucketList).build();

when(s3Client.listBuckets(any(ListBucketsRequest.class))).thenReturn(mockResponse);
        when(s3Client.getBucketEncryption(any(GetBucketEncryptionRequest.class)))
                .thenReturn(buildGetBucketEncryptionResponse("AES256"))
                .thenReturn(buildGetBucketEncryptionResponse("AES256", "aws:kms"))
                .thenThrow(S3Exception.builder().message("No Encrypt").build())
                .thenReturn(buildGetBucketEncryptionResponse("aws:kms"))
```

```
                .thenReturn(buildGetBucketEncryptionResponse("AES256"));
        setServiceClient(s3Client);

        final TypeConfigurationModel typeConfiguration =
TypeConfigurationModel.builder()
                .encryptionAlgorithm("AES256")
                .minBuckets("10")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::S3::Bucket")
                        .targetModel(
                            createHookTargetModel(
                                AwsS3Bucket.builder()
                                    .bucketName("toBeDeletedBucket")
                                    .build()
                            )
                        )
                        .build())
                .build();

        final ProgressEvent<HookTargetModel, CallbackContext> response =
handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(s3Client,
times(5)).getBucketEncryption(any(GetBucketEncryptionRequest.class));
        verify(handler, never()).getBucketSSEAlgorithm("toBeDeletedBucket");

        assertResponse(response, OperationStatus.FAILED, "Failed to meet minimum of
[10] encrypted buckets.");
    }


    @Test
    public void handleRequest_awsSqsQueueFailed() {
        final PreDeleteHookHandler handler = Mockito.spy(new
PreDeleteHookHandler());

        final List<String> queueUrls = ImmutableList.of(
                "https://queue1.queue",
                "https://queue2.queue",
                "https://toBeDeletedQueue.queue",
```

```
                "https://queue3.queue",
                "https://queue4.queue",
                "https://queue5.queue"
        );

        when(sqsClient.getQueueUrl(any(GetQueueUrlRequest.class)))
                .thenReturn(GetQueueUrlResponse.builder().queueUrl("https://
toBeDeletedQueue.queue").build());
        when(sqsClient.listQueues(any(ListQueuesRequest.class)))

 .thenReturn(ListQueuesResponse.builder().queueUrls(queueUrls).build());
        when(sqsClient.getQueueAttributes(any(GetQueueAttributesRequest.class)))

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build())
                .thenReturn(GetQueueAttributesResponse.builder().attributes(new
 HashMap<>()).build())

 .thenReturn(GetQueueAttributesResponse.builder().attributes(ImmutableMap.of(QueueAttribute
 "kmsKeyId")).build());
        setServiceClient(sqsClient);

        final TypeConfigurationModel typeConfiguration =
 TypeConfigurationModel.builder()
                .minQueues("10")
                .build();

        final HookHandlerRequest request = HookHandlerRequest.builder()
                .hookContext(
                    HookContext.builder()
                        .targetName("AWS::SQS::Queue")
                        .targetModel(
                            createHookTargetModel(
                                ImmutableMap.of("QueueName", "toBeDeletedQueue")
                            )
                        )
                        .build())
                .build();
```

```
        final ProgressEvent<HookTargetModel, CallbackContext> response =
 handler.handleRequest(proxy, request, null, logger, typeConfiguration);

        verify(sqsClient,
 times(5)).getQueueAttributes(any(GetQueueAttributesRequest.class));
        verify(handler, never()).isQueueEncrypted("toBeDeletedQueue");

        assertResponse(response, OperationStatus.FAILED, "Failed to meet minimum of
 [10] encrypted queues.");
    }

    private GetBucketEncryptionResponse buildGetBucketEncryptionResponse(final
 String ...sseAlgorithm) {
        return buildGetBucketEncryptionResponse(
                Arrays.stream(sseAlgorithm)
                    .map(a ->
 ServerSideEncryptionRule.builder().applyServerSideEncryptionByDefault(
                        ServerSideEncryptionByDefault.builder()
                            .sseAlgorithm(a)
                            .build()
                        ).build()
                    )
                    .collect(Collectors.toList())
        );
    }

    private GetBucketEncryptionResponse buildGetBucketEncryptionResponse(final
 Collection<ServerSideEncryptionRule> rules) {
        return GetBucketEncryptionResponse.builder()
                .serverSideEncryptionConfiguration(
                    ServerSideEncryptionConfiguration.builder().rules(
                        rules
                    ).build()
                ).build();
    }
}
```

使用 Python 建立自訂 CloudFormation 勾點的模型

建立自訂 CloudFormation 勾點的模型需要建立定義勾點、其屬性及其屬性的結構描述。本教學課程會逐步引導您使用 Python 建立自訂勾點的模型。

步驟 1：產生勾點專案套件

產生您的 Hook 專案套件。CloudFormation CLI 會建立空的處理常式函數，對應至目標生命週期中的特定 Hook 動作，如 Hook 規格所定義。

```
cfn generate
```

命令會傳回下列輸出：

```
Generated files for MyCompany::Testing::MyTestHook
```

> **Note**
>
> 請確定您的 Lambda 執行時間是up-to-date，以避免使用已取代的版本。如需詳細資訊，請參閱更新資源類型和勾點的 Lambda 執行時間。

步驟 2：新增掛鉤處理常式

將您自己的 Hook 處理常式執行時間程式碼新增至您選擇實作的處理常式。例如，您可以新增下列程式碼進行記錄。

```
LOG.setLevel(logging.INFO)
LOG.info("Internal testing Hook triggered for target: " +
 request.hookContext.targetName);
```

CloudFormation CLI 會從 產生 `src/models.py` 檔案組態結構描述。

Example models.py

```
import sys
from dataclasses import dataclass
from inspect import getmembers, isclass
from typing import (
    AbstractSet,
    Any,
    Generic,
    Mapping,
    MutableMapping,
    Optional,
    Sequence,
```

```
    Type,
    TypeVar,
)

from cloudformation_cli_python_lib.interface import (
    BaseModel,
    BaseHookHandlerRequest,
)
from cloudformation_cli_python_lib.recast import recast_object
from cloudformation_cli_python_lib.utils import deserialize_list

T = TypeVar("T")


def set_or_none(value: Optional[Sequence[T]]) -> Optional[AbstractSet[T]]:
    if value:
        return set(value)
    return None


@dataclass
class HookHandlerRequest(BaseHookHandlerRequest):
    pass


@dataclass
class TypeConfigurationModel(BaseModel):
    limitSize: Optional[str]
    cidr: Optional[str]
    encryptionAlgorithm: Optional[str]

    @classmethod
    def _deserialize(
        cls: Type["_TypeConfigurationModel"],
        json_data: Optional[Mapping[str, Any]],
    ) -> Optional["_TypeConfigurationModel"]:
        if not json_data:
            return None
        return cls(
            limitSize=json_data.get("limitSize"),
            cidr=json_data.get("cidr"),
            encryptionAlgorithm=json_data.get("encryptionAlgorithm"),
        )
```

```
_TypeConfigurationModel = TypeConfigurationModel
```

步驟 3：實作勾點處理常式

透過產生的 Python 資料類別，您可以撰寫實際實作 Hook 功能的處理常式。在此範例中，您將實作處理常式的 `preUpdate`、 `preCreate`和 `preDelete` 叫用點。

主題

- 實作 preCreate 處理常式
- 實作 preUpdate 處理常式
- 實作 preDelete 處理常式
- 實作勾點處理常式

實作 preCreate 處理常式

`preCreate` 處理常式會驗證 `AWS::S3::Bucket` 或 `AWS::SQS::Queue` 資源的伺服器端加密設定。

- 對於 `AWS::S3::Bucket` 資源，只有在下列情況為 true 時，勾點才會傳遞。
  - Amazon S3 儲存貯體加密已設定。
  - 已啟用儲存貯體的 Amazon S3 儲存貯體金鑰。
  - Amazon S3 儲存貯體的加密演算法集是所需的正確演算法。
  - AWS Key Management Service 金鑰 ID 已設定。
- 對於 `AWS::SQS::Queue` 資源，只有在下列情況為 true 時，勾點才會傳遞。
  - AWS Key Management Service 金鑰 ID 已設定。

實作 preUpdate 處理常式

實作`preUpdate`處理常式，其會在處理常式中所有指定目標的更新操作之前啟動。`preUpdate` 處理常式會完成下列項目：

- 對於 `AWS::S3::Bucket` 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - Amazon S3 儲存貯體的儲存貯體加密演算法尚未修改。

實作 preDelete 處理常式

實作preDelete處理常式，其會在處理常式中所有指定目標的刪除操作之前啟動。preDelete 處理常式會完成下列項目：

- 對於 AWS::S3::Bucket 資源，只有在下列情況為 true 時，勾點才會傳遞：
  - 驗證刪除資源後，帳戶是否將存在最低必要合規資源。
  - 最低合規資源數量是在 Hook 的組態中設定。

實作勾點處理常式

1. 在您的 IDE 中，開啟位於 src 資料夾的 handlers.py 檔案。
2. 使用下列程式碼取代handlers.py檔案的整個內容。

   Example handlers.py

```
import logging
from typing import Any, MutableMapping, Optional
import botocore

from cloudformation_cli_python_lib import (
    BaseHookHandlerRequest,
    HandlerErrorCode,
    Hook,
    HookInvocationPoint,
    OperationStatus,
    ProgressEvent,
    SessionProxy,
    exceptions,
)

from .models import HookHandlerRequest, TypeConfigurationModel

# Use this logger to forward log messages to CloudWatch Logs.
LOG = logging.getLogger(__name__)
TYPE_NAME = "MyCompany::Testing::MyTestHook"

LOG.setLevel(logging.INFO)

hook = Hook(TYPE_NAME, TypeConfigurationModel)
test_entrypoint = hook.test_entrypoint
```

```
def _validate_s3_bucket_encryption(
    bucket: MutableMapping[str, Any], required_encryption_algorithm: str
) -> ProgressEvent:
    status = None
    message = ""
    error_code = None

    if bucket:
        bucket_name = bucket.get("BucketName")

        bucket_encryption = bucket.get("BucketEncryption")
        if bucket_encryption:
            server_side_encryption_rules = bucket_encryption.get(
                "ServerSideEncryptionConfiguration"
            )
            if server_side_encryption_rules:
                for rule in server_side_encryption_rules:
                    bucket_key_enabled = rule.get("BucketKeyEnabled")
                    if bucket_key_enabled:
                        server_side_encryption_by_default = rule.get(
                            "ServerSideEncryptionByDefault"
                        )

                        encryption_algorithm =
 server_side_encryption_by_default.get(
                            "SSEAlgorithm"
                        )
                        kms_key_id = server_side_encryption_by_default.get(
                            "KMSMasterKeyID"
                        )  # "KMSMasterKeyID" is name of the property for an
 AWS::S3::Bucket

                        if encryption_algorithm == required_encryption_algorithm:
                            if encryption_algorithm == "aws:kms" and not
 kms_key_id:
                                status = OperationStatus.FAILED
                                message = f"KMS Key ID not set for bucket with
 name: f{bucket_name}"
                            else:
                                status = OperationStatus.SUCCESS
                                message = f"Successfully invoked
 PreCreateHookHandler for AWS::S3::Bucket with name: {bucket_name}"
```

```
                        else:
                            status = OperationStatus.FAILED
                            message = f"SSE Encryption Algorithm is incorrect for
 bucket with name: {bucket_name}"
                    else:
                        status = OperationStatus.FAILED
                        message = f"Bucket key not enabled for bucket with name:
 {bucket_name}"

                    if status == OperationStatus.FAILED:
                        break
            else:
                status = OperationStatus.FAILED
                message = f"No SSE Encryption configurations for bucket with name:
 {bucket_name}"
        else:
            status = OperationStatus.FAILED
            message = (
                f"Bucket Encryption not enabled for bucket with name:
 {bucket_name}"
            )
    else:
        status = OperationStatus.FAILED
        message = "Resource properties for S3 Bucket target model are empty"

    if status == OperationStatus.FAILED:
        error_code = HandlerErrorCode.NonCompliant

    return ProgressEvent(status=status, message=message, errorCode=error_code)


def _validate_sqs_queue_encryption(queue: MutableMapping[str, Any]) ->
 ProgressEvent:
    if not queue:
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message="Resource properties for SQS Queue target model are empty",
            errorCode=HandlerErrorCode.NonCompliant,
        )
    queue_name = queue.get("QueueName")

    kms_key_id = queue.get(
        "KmsMasterKeyId"
    )  # "KmsMasterKeyId" is name of the property for an AWS::SQS::Queue
```

```
    if not kms_key_id:
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message=f"Server side encryption turned off for queue with name:
 {queue_name}",
            errorCode=HandlerErrorCode.NonCompliant,
        )

    return ProgressEvent(
        status=OperationStatus.SUCCESS,
        message=f"Successfully invoked PreCreateHookHandler for
 targetAWS::SQS::Queue with name: {queue_name}",
    )


@hook.handler(HookInvocationPoint.CREATE_PRE_PROVISION)
def pre_create_handler(
    session: Optional[SessionProxy],
    request: HookHandlerRequest,
    callback_context: MutableMapping[str, Any],
    type_configuration: TypeConfigurationModel,
) -> ProgressEvent:
    target_name = request.hookContext.targetName
    if "AWS::S3::Bucket" == target_name:
        return _validate_s3_bucket_encryption(
            request.hookContext.targetModel.get("resourceProperties"),
            type_configuration.encryptionAlgorithm,
        )
    elif "AWS::SQS::Queue" == target_name:
        return _validate_sqs_queue_encryption(
            request.hookContext.targetModel.get("resourceProperties")
        )
    else:
        raise exceptions.InvalidRequest(f"Unknown target type: {target_name}")


def _validate_bucket_encryption_rules_not_updated(
    resource_properties, previous_resource_properties
) -> ProgressEvent:
    bucket_encryption_configs = resource_properties.get("BucketEncryption",
 {}).get(
        "ServerSideEncryptionConfiguration", []
    )
    previous_bucket_encryption_configs = previous_resource_properties.get(
```

```
        "BucketEncryption", {}
    ).get("ServerSideEncryptionConfiguration", [])

    if len(bucket_encryption_configs) != len(previous_bucket_encryption_configs):
        return ProgressEvent(
            status=OperationStatus.FAILED,
            message=f"Current number of bucket encryption configs does not
 match previous. Current has {str(len(bucket_encryption_configs))} configs while
 previously there were {str(len(previous_bucket_encryption_configs))} configs",
            errorCode=HandlerErrorCode.NonCompliant,
        )

    for i in range(len(bucket_encryption_configs)):
        current_encryption_algorithm = (
            bucket_encryption_configs[i]
            .get("ServerSideEncryptionByDefault", {})
            .get("SSEAlgorithm")
        )
        previous_encryption_algorithm = (
            previous_bucket_encryption_configs[i]
            .get("ServerSideEncryptionByDefault", {})
            .get("SSEAlgorithm")
        )

        if current_encryption_algorithm != previous_encryption_algorithm:
            return ProgressEvent(
                status=OperationStatus.FAILED,
                message=f"Bucket Encryption algorithm can not be changed once
 set. The encryption algorithm was changed to {current_encryption_algorithm} from
 {previous_encryption_algorithm}.",
                errorCode=HandlerErrorCode.NonCompliant,
            )

    return ProgressEvent(
        status=OperationStatus.SUCCESS,
        message="Successfully invoked PreUpdateHookHandler for target:
 AWS::SQS::Queue",
    )


def _validate_queue_encryption_not_disabled(
    resource_properties, previous_resource_properties
) -> ProgressEvent:
    if previous_resource_properties.get(
```

```
        "KmsMasterKeyId"
    ) and not resource_properties.get("KmsMasterKeyId"):
        return ProgressEvent(
            status=OperationStatus.FAILED,
            errorCode=HandlerErrorCode.NonCompliant,
            message="Queue encryption can not be disable",
        )
    else:
        return ProgressEvent(status=OperationStatus.SUCCESS)


@hook.handler(HookInvocationPoint.UPDATE_PRE_PROVISION)
def pre_update_handler(
    session: Optional[SessionProxy],
    request: BaseHookHandlerRequest,
    callback_context: MutableMapping[str, Any],
    type_configuration: MutableMapping[str, Any],
) -> ProgressEvent:
    target_name = request.hookContext.targetName
    if "AWS::S3::Bucket" == target_name:
        resource_properties =
 request.hookContext.targetModel.get("resourceProperties")
        previous_resource_properties = request.hookContext.targetModel.get(
            "previousResourceProperties"
        )

        return _validate_bucket_encryption_rules_not_updated(
            resource_properties, previous_resource_properties
        )
    elif "AWS::SQS::Queue" == target_name:
        resource_properties =
 request.hookContext.targetModel.get("resourceProperties")
        previous_resource_properties = request.hookContext.targetModel.get(
            "previousResourceProperties"
        )

        return _validate_queue_encryption_not_disabled(
            resource_properties, previous_resource_properties
        )
    else:
        raise exceptions.InvalidRequest(f"Unknown target type: {target_name}")
```

繼續進行下一個主題向 註冊自訂勾點 CloudFormation。

向 註冊自訂勾點 CloudFormation

建立自訂勾點後，您需要向 註冊它 CloudFormation ，才能使用它。在本節中，您將學習如何封裝和註冊您的勾點，以便在 中使用 AWS 帳戶。

封裝勾點 (Java)

如果您已使用 Java 開發 Hook，請使用 Maven 來封裝它。

在 Hook 專案的 目錄中，執行下列命令來建置您的 Hook、執行單元測試，並將您的專案封裝為檔案JAR，可用來將 Hook 提交至 CloudFormation 登錄檔。

```
mvn clean package
```

註冊自訂勾點

註冊勾點

1. （選用）透過提交 configure操作us-west-2，將預設 AWS 區域 名稱設定為 。

   ```
   $ aws configure
   AWS Access Key ID [None]: <Your Access Key ID>
   AWS Secret Access Key [None]: <Your Secret Key>
   Default region name [None]: us-west-2
   Default output format [None]: json
   ```

2. （選用）下列命令會建置和封裝您的 Hook 專案，而不註冊它。

   ```
   $ cfn submit --dry-run
   ```

3. 使用 CloudFormation CLI submit操作註冊您的勾點。

   ```
   $ cfn submit --set-default
   ```

   該命令會傳回下列命令。

   ```
   {‘ProgressStatus’: ‘COMPLETE’}
   ```

   結果：您已成功註冊您的勾點。

驗證勾點可在您的帳戶中存取

確認您的勾點在您提交勾點的 AWS 帳戶 和 區域中可用。

1. 若要驗證您的勾點，請使用 list-types命令列出新註冊的勾點，並傳回其摘要說明。

```
$ aws cloudformation list-types
```

命令會傳回下列輸出，也會顯示您可以在 AWS 帳戶 和 區域中啟用的公開可用勾點。

```
{
    "TypeSummaries": [
        {
            "Type": "HOOK",
            "TypeName": "MyCompany::Testing::MyTestHook",
            "DefaultVersionId": "00000001",
            "TypeArn": "arn:aws:cloudformation:us-west-2:ACCOUNT_ID/type/hook/
MyCompany-Testing-MyTestHook",
            "LastUpdated": "2021-08-04T23:00:03.058000+00:00",
            "Description": "Verifies S3 bucket and SQS queues properties before
 creating or updating"
        }
    ]
}
```

2. TypeArn 從勾點的list-type輸出擷取 並儲存它。

```
export HOOK_TYPE_ARN=arn:aws:cloudformation:us-west-2:ACCOUNT_ID/type/hook/
MyCompany-Testing-MyTestHook
```

若要了解如何發佈勾點以供公開使用，請參閱 發佈勾點以供公開使用。

設定勾點

開發並註冊 Hook 之後，您可以透過將 Hook 發佈到登錄檔 AWS 帳戶 ，在 中設定您的 Hook。

- 若要在帳戶中設定勾點，請使用 SetTypeConfiguration操作。此操作將啟用勾點結構描述 properties 區段中定義的掛鉤屬性。在下列範例中， minBuckets 屬性在組態1中設定為 。

> **Note**
>
> 在帳戶中啟用勾點，即表示您授權勾點使用您 定義的許可 AWS 帳戶。CloudFormation 會在將您的許可傳遞給勾點之前，移除非必要的許可。CloudFormation 建議客戶或勾點使用者檢閱勾點許可，並注意在您的帳戶中啟用勾點之前允許勾點的許可。

在相同的 帳戶和 中，指定已註冊的 Hook 延伸模組的組態資料 AWS 區域。

```
$ aws cloudformation set-type-configuration --region us-west-2
  --configuration '{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus":"ENABLED","FailureMode":"FAIL","Properties":{"minBuckets":
 "1","minQueues": "1", "encryptionAlgorithm": "aws:kms"}}}}'
  --type-arn $HOOK_TYPE_ARN
```

> **Important**
>
> 若要讓 Hook 主動檢查堆疊的組態，您必須在帳戶中註冊和啟用 Hook 之後，在 `HookConfiguration`區段`HookInvocationStatus`中將 `ENABLED` 設定為 。

在處理常式中存取 AWS APIs

如果您的 Hooks 在其任何處理常式中使用 AWS API，CFN-CLI 會自動建立 IAM 執行角色範本 `hook-role.yaml`。`hook-role.yaml` 範本是根據 Hook 結構描述之處理常式區段中每個處理常式指定的許可。如果未在generate操作期間使用 `--role-arn`旗標，則會佈建此堆疊中的角色，並用作勾點的執行角色。

如需詳細資訊，請參閱從資源類型存取 AWS APIs。

hook-role.yaml 範本

> **Note**
>
> 如果您選擇建立自己的執行角色，強烈建議透過僅允許列出 `hooks.cloudformation.amazonaws.com`和 來練習最低權限原則`resources.cloudformation.amazonaws.com`。

下列範本使用 IAM、Amazon S3 和 Amazon SQS 許可。

```
AWSTemplateFormatVersion: 2010-09-09
Description: >
  This CloudFormation template creates a role assumed by CloudFormation during
  Hook operations on behalf of the customer.
Resources:
  ExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      MaxSessionDuration: 8400
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - resources.cloudformation.amazonaws.com
                - hooks.cloudformation.amazonaws.com
            Action: 'sts:AssumeRole'
            Condition:
              StringEquals:
                aws:SourceAccount: !Ref AWS::AccountId
              StringLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:cloudformation:
${AWS::Region}:${AWS::AccountId}:type/hook/MyCompany-Testing-MyTestHook/*
      Path: /
      Policies:
        - PolicyName: HookTypePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 's3:GetEncryptionConfiguration'
                  - 's3:ListBucket'
                  - 's3:ListAllMyBuckets'
                  - 'sqs:GetQueueAttributes'
                  - 'sqs:GetQueueUrl'
                  - 'sqs:ListQueues'
                Resource: '*'
Outputs:
  ExecutionRoleArn:
    Value: !GetAtt
```

```
      - ExecutionRole
      - Arn
```

在 中測試自訂勾點 AWS 帳戶

現在您已將對應至調用點的處理常式函數編碼，是時候在 CloudFormation 堆疊上測試自訂掛接了。

FAIL 如果 CloudFormation 範本未佈建具有下列項目的 S3 儲存貯體，則勾點失敗模式會設為 ：

- Amazon S3 儲存貯體加密已設定。
- 已啟用儲存貯體的 Amazon S3 儲存貯體金鑰。
- Amazon S3 儲存貯體的加密演算法集是所需的正確演算法。
- AWS Key Management Service 金鑰 ID 已設定。

在下列範例中，建立名為 my-failed-bucket-stack.yml 且堆疊名稱為 的範本my-hook-stack，該範本會讓堆疊組態失敗，並在資源佈建之前停止。

透過佈建堆疊來測試勾點

範例 1：佈建堆疊

佈建不合規的堆疊

1. 撰寫指定 S3 儲存貯體的範本。例如 my-failed-bucket-stack.yml。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties: {}
```

2. 建立堆疊，並在 AWS Command Line Interface () 中指定您的範本AWS CLI。在下列範例中，將堆疊名稱指定為 ，my-hook-stack並將範本名稱指定為 my-failed-bucket-stack.yml。

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://my-failed-bucket-stack.yml
```

3. （選用）透過指定堆疊名稱來檢視堆疊進度。在下列範例中，指定堆疊名稱 my-hook-stack。

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

使用 describe-stack-events操作來查看建立儲存貯體時的勾點失敗。以下是 命令的範例輸出。

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
hook-stack/2c693970-f57e-11eb-a0fb-061a2a83f0b9",
            "EventId": "S3Bucket-CREATE_FAILED-2021-08-04T23:47:03.305Z",
            "StackName": "my-hook-stack",
            "LogicalResourceId": "S3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:47:03.305000+00:00",
            "ResourceStatus": "CREATE_FAILED",
            "ResourceStatusReason": "The following hook(s) failed:
 [MyCompany::Testing::MyTestHook]",
            "ResourceProperties": "{}",
            "ClientRequestToken": "Console-CreateStack-abe71ac2-ade4-
a762-0499-8d34d91d6a92"
        },
    ...
    ]
}
```

結果：Hook 調用使堆疊組態失敗，並停止資源佈建。

使用 CloudFormation 範本傳遞勾點驗證

1. 若要建立堆疊並通過勾點驗證，請更新範本，讓您的資源使用加密的 S3 儲存貯體。此範例使用範本 my-encrypted-bucket-stack.yml。

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
```

```
  EncryptedS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub encryptedbucket-${AWS::Region}-${AWS::AccountId}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: !Ref EncryptionKey
            BucketKeyEnabled: true
  EncryptionKey:
    Type: AWS::KMS::Key
    DeletionPolicy: Retain
    Properties:
      Description: KMS key used to encrypt the resource type artifacts
      EnableKeyRotation: true
      KeyPolicy:
        Version: 2012-10-17
        Statement:
          - Sid: Enable full access for owning account
            Effect: Allow
            Principal:
              AWS: !Ref AWS::AccountId
            Action: 'kms:*'
            Resource: '*'
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
```

> **Note**
>
> 不會針對略過的資源叫用勾點。

2. 建立堆疊並指定您的範本。在此範例中，堆疊名稱為 my-encrypted-bucket-stack。

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://my-encrypted-bucket-stack.yml \
```

3. （選用）透過指定堆疊名稱來檢視堆疊進度。

```
$ aws cloudformation describe-stack-events \
```

```
  --stack-name my-encrypted-bucket-stack
```

使用 describe-stack-events命令來檢視回應。以下是 describe-stack-events 命令的範例。

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_COMPLETE-2021-08-04T23:23:20.973Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_IN_PROGRESS-2021-08-04T23:22:59.410Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:59.410000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
```

```
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-6516081f-c1f2-4bfe-a0f0-cefa28679994",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:58.349000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Hook invocations complete.  Resource creation
 initiated",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
    ...
    ]
}
```

結果：CloudFormation 已成功建立堆疊。Hook 的邏輯驗證`AWS::S3::Bucket`資源在佈建資源之前包含伺服器端加密。

範例 2：佈建堆疊

佈建不合規的堆疊

1. 撰寫指定 S3 儲存貯體的範本。例如 `aes256-bucket.yml`。

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
  EncryptedS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub encryptedbucket-${AWS::Region}-${AWS::AccountId}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
```

```
              SSEAlgorithm: AES256
            BucketKeyEnabled: true
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
```

2. 建立堆疊，並在 中指定您的範本 AWS CLI。在下列範例中，將堆疊名稱指定為 ，my-hook-stack並將範本名稱指定為 aes256-bucket.yml。

```
$ aws cloudformation create-stack \
  --stack-name my-hook-stack \
  --template-body file://aes256-bucket.yml
```

3. （選用）透過指定堆疊名稱來檢視堆疊進度。在下列範例中，指定堆疊名稱 my-hook-stack。

```
$ aws cloudformation describe-stack-events \
  --stack-name my-hook-stack
```

使用 describe-stack-events操作來查看建立儲存貯體時的勾點失敗。以下是 命令的範例輸出。

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
hook-stack/2c693970-f57e-11eb-a0fb-061a2a83f0b9",
            "EventId": "S3Bucket-CREATE_FAILED-2021-08-04T23:47:03.305Z",
            "StackName": "my-hook-stack",
            "LogicalResourceId": "S3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:47:03.305000+00:00",
            "ResourceStatus": "CREATE_FAILED",
            "ResourceStatusReason": "The following hook(s) failed:
 [MyCompany::Testing::MyTestHook]",
            "ResourceProperties": "{}",
            "ClientRequestToken": "Console-CreateStack-abe71ac2-ade4-
a762-0499-8d34d91d6a92"
        },
    ...
    ]
```

```
}
```

結果：Hook 調用使堆疊組態失敗，並停止資源佈建。由於 S3 儲存貯體加密設定不正確，堆疊失敗。此儲存貯體使用 aws:kms時，需要勾點類型組態AES256。

使用 CloudFormation 範本傳遞勾點驗證

1. 若要建立堆疊並通過勾點驗證，請更新範本，讓您的資源使用加密的 S3 儲存貯體。此範例使用範本 kms-bucket-and-queue.yml。

```
AWSTemplateFormatVersion: 2010-09-09
Description: |
  This CloudFormation template provisions an encrypted S3 Bucket
Resources:
  EncryptedS3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub encryptedbucket-${AWS::Region}-${AWS::AccountId}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: 'aws:kms'
              KMSMasterKeyID: !Ref EncryptionKey
            BucketKeyEnabled: true
  EncryptedQueue:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: !Sub encryptedqueue-${AWS::Region}-${AWS::AccountId}
      KmsMasterKeyId: !Ref EncryptionKey
  EncryptionKey:
    Type: AWS::KMS::Key
    DeletionPolicy: Retain
    Properties:
      Description: KMS key used to encrypt the resource type artifacts
      EnableKeyRotation: true
      KeyPolicy:
        Version: 2012-10-17
        Statement:
          - Sid: Enable full access for owning account
            Effect: Allow
            Principal:
              AWS: !Ref AWS::AccountId
```

```
            Action: 'kms:*'
            Resource: '*'
Outputs:
  EncryptedBucketName:
    Value: !Ref EncryptedS3Bucket
  EncryptedQueueName:
    Value: !Ref EncryptedQueue
```

> **Note**
>
> 不會針對略過的資源叫用勾點。

2. 建立堆疊並指定您的範本。在此範例中，堆疊名稱為 my-encrypted-bucket-stack。

```
$ aws cloudformation create-stack \
  --stack-name my-encrypted-bucket-stack \
  --template-body file://kms-bucket-and-queue.yml
```

3. （選用）透過指定堆疊名稱來檢視堆疊進度。

```
$ aws cloudformation describe-stack-events \
  --stack-name my-encrypted-bucket-stack
```

使用 describe-stack-events命令來檢視回應。以下是 describe-stack-events 命令的範例。

```
{
    "StackEvents": [
    ...
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_COMPLETE-2021-08-04T23:23:20.973Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:23:20.973000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
```

```
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-
CREATE_IN_PROGRESS-2021-08-04T23:22:59.410Z",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "encryptedbucket-us-west-2-123456789012",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:59.410000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"BucketName\":\"encryptedbucket-us-
west-2-123456789012\",\"BucketEncryption\":{\"ServerSideEncryptionConfiguration\":
[{\"BucketKeyEnabled\":\"true\",\"ServerSideEncryptionByDefault\":{\"SSEAlgorithm
\":\"aws:kms\",\"KMSMasterKeyID\":\"ENCRYPTION_KEY_ARN\"}}]}}",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
encrypted-bucket-stack/82a97150-f57a-11eb-8eb2-06a6bdcc7779",
            "EventId": "EncryptedS3Bucket-6516081f-c1f2-4bfe-a0f0-cefa28679994",
            "StackName": "my-encrypted-bucket-stack",
            "LogicalResourceId": "EncryptedS3Bucket",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2021-08-04T23:22:58.349000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Hook invocations complete.  Resource creation
 initiated",
            "ClientRequestToken": "Console-CreateStack-39df35ac-ca00-
b7f6-5661-4e917478d075"
        },
    ...
    ]
```

```
}
```

結果：CloudFormation 已成功建立堆疊。Hook 的邏輯驗證AWS::S3::Bucket資源在佈建資源之前包含伺服器端加密。

更新自訂勾點

更新自訂勾點允許在 CloudFormation 登錄檔中提供勾點中的修訂。

若要更新自訂勾點，請透過 CloudFormation CLI submit操作將您的修訂提交至 CloudFormation 登錄檔。

```
$ cfn submit
```

若要在帳戶中指定 Hook 的預設版本，請使用 set-type-default-version命令並指定類型、類型名稱和版本 ID。

```
$ aws cloudformation set-type-default-version \
    --type HOOK \
    --type-name MyCompany::Testing::MyTestHook \
    --version-id 00000003
```

若要擷取有關勾點版本的資訊，請使用 list-type-versions。

```
$ aws cloudformation list-type-versions \
  --type HOOK \
  --type-name "MyCompany::Testing::MyTestHook"
```

從 CloudFormation 登錄檔取消註冊自訂掛鉤

取消註冊自訂勾點會將擴充功能或擴充功能版本標記為 CloudFormation 登錄DEPRECATED檔中的 ，這會將其從作用中使用中移除。一旦棄用，自訂勾點就無法在 CloudFormation 操作中使用。

> **Note**
>
> 取消註冊勾點之前，您必須個別取消註冊該延伸模組的所有先前作用中版本。如需詳細資訊，請參閱DeregisterType。

若要取消註冊勾點，請使用 deregister-type操作並指定您的勾點 ARN。

```
$ aws cloudformation deregister-type \
    --arn HOOK_TYPE_ARN
```

此命令不會產生輸出。

發佈勾點以供公開使用

若要開發公有第三方勾點，請將您的勾點開發為私有延伸模組。然後，在您要公開提供延伸項目的每個 AWS 區域 中：

1. 在 CloudFormation 登錄檔中將勾點註冊為私有延伸。
2. 測試您的勾點，以確保它符合在 CloudFormation 登錄檔中發佈的所有必要要求。
3. 將您的勾點發佈至 CloudFormation 登錄檔。

> **Note**
>
> 在指定區域中發佈任何擴充功能之前，您必須先註冊為該區域中的擴充功能發佈者。若要同時在多個區域中執行此操作，請參閱《 CloudFormation CLI 使用者指南》中的使用 StackSets 在多個區域中發佈擴充功能。

開發並註冊 Hook 之後，您可以將它發佈到 CloudFormation 登錄檔做為第三方公有延伸，讓一般 CloudFormation 使用者公開使用它。

公有第三方勾點可讓您提供 CloudFormation 使用者在佈建之前主動檢查 AWS 資源的組態。如同私有勾點，公有勾點與 CloudFormation AWS 內發佈的任何勾點視為相同。

發佈到登錄檔的勾點可供所有 CloudFormation 使用者在發佈它們 AWS 區域 的 中看見。然後，使用者可以在其帳戶中啟用您的擴充功能，以便在其範本中使用。如需詳細資訊，請參閱CloudFormation《 使用者指南》中的從 CloudFormation 登錄檔使用第三方公有擴充功能。

測試自訂勾點以供公開使用

若要發佈已註冊的自訂勾點，它必須通過為其定義的所有測試要求。以下是將自訂勾點發佈為第三方延伸模組之前所需的需求清單。

每個處理常式和目標都會經過兩次測試。一次用於 SUCCESS，一次用於 FAILED。

- 對於SUCCESS回應案例：
  - 狀態必須為 SUCCESS。
  - 不得傳回錯誤碼。
  - 如果指定，回呼延遲應設定為0秒。
- 對於FAILED回應案例：
  - 狀態必須為 FAILED。
  - 必須傳回錯誤代碼。
  - 必須有訊息作為回應。
  - 如果指定，回呼延遲應設定為0秒。
- 對於IN_PROGRESS回應案例：
  - 不得傳回錯誤碼。
  - Result 欄位不得在回應中設定。

指定用於合約測試的輸入資料

根據預設， 會使用您在 Hook 結構描述中定義的模式所產生的輸入屬性 CloudFormation 來執行合約測試。不過，大多數勾點都夠複雜，因此預先建立或預先更新佈建堆疊的輸入屬性需要了解佈建的資源。若要解決此問題，您可以指定 執行其合約測試 CloudFormation 時使用的輸入。

CloudFormation 提供兩種方式，可讓您指定輸入資料，以便在執行合約測試時使用：

- 覆寫檔案

  使用 overrides 檔案提供輕量方式，可指定特定屬性的輸入資料 CloudFormation ，以便在 preCreate、 preUpdate和 preDelete操作測試期間使用。
- 輸入檔案

  在以下情況下，您也可以使用多個input檔案來指定合約測試輸入資料：
  - 您想要或需要為建立、更新和刪除操作指定不同的輸入資料，或測試的無效資料。
  - 您想要指定多個不同的輸入資料集。

使用覆寫檔案指定輸入資料

以下是使用 overrides 檔案的 Amazon S3 Hook 輸入資料範例。

```
{
    "CREATE_PRE_PROVISION": {
        "AWS::S3::Bucket": {
            "resourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            }
        },
        "AWS::SQS::Queue": {
            "resourceProperties": {
                "/QueueName": "MyQueueContract",
                "/KmsMasterKeyId": "hellocontract"
            }
        }
    },
    "UPDATE_PRE_PROVISION": {
        "AWS::S3::Bucket": {
            "resourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "previousResourceProperties": {
                "/BucketName": "encryptedbucket-us-west-2-contractor",
                "/BucketEncryption/ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
```

```
                        "KMSMasterKeyID": "KMS-KEY-ARN",
                        "SSEAlgorithm": "aws:kms"
                    }
                }
            ]
        }
    }
},
"INVALID_UPDATE_PRE_PROVISION": {
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "/BucketName": "encryptedbucket-us-west-2-contractor",
            "/BucketEncryption/ServerSideEncryptionConfiguration": [
                {
                    "BucketKeyEnabled": true,
                    "ServerSideEncryptionByDefault": {
                        "KMSMasterKeyID": "KMS-KEY-ARN",
                        "SSEAlgorithm": "AES256"
                    }
                }
            ]
        },
        "previousResourceProperties": {
            "/BucketName": "encryptedbucket-us-west-2-contractor",
            "/BucketEncryption/ServerSideEncryptionConfiguration": [
                {
                    "BucketKeyEnabled": true,
                    "ServerSideEncryptionByDefault": {
                        "KMSMasterKeyID": "KMS-KEY-ARN",
                        "SSEAlgorithm": "aws:kms"
                    }
                }
            ]
        }
    }
},
"INVALID": {
    "AWS::SQS::Queue": {
        "resourceProperties": {
            "/QueueName": "MyQueueContract",
            "/KmsMasterKeyId": "KMS-KEY-ARN"
        }
    }
}
```

```
}
```

使用輸入檔案指定輸入資料

使用 input 檔案來指定不同類型的輸入資料 CloudFormation ，以供 使用：preCreate輸入、preUpdate輸入和無效的輸入。每種類型的資料都會在個別的檔案中指定。您也可以為合約測試指定多組輸入資料。

若要指定 CloudFormation 要在合約測試中使用的input檔案，請將 inputs 資料夾新增至 Hooks 專案的根目錄。然後新增您的輸入檔案。

使用以下命名慣例指定檔案包含的輸入資料類型，其中 *n* 是整數：

- inputs_*n*_pre_create.json：使用檔案搭配preCreate處理常式來指定用於建立資源的輸入。
- inputs_*n*_pre_update.json：使用檔案搭配preUpdate處理常式來指定用於更新資源的輸入。
- inputs_*n*_pre_delete.json：使用檔案搭配preDelete處理常式來指定用於刪除資源的輸入。
- inputs_*n*_invalid.json：用於指定無效的輸入進行測試。

若要為合約測試指定多組輸入資料，請在檔案名稱中遞增整數，以排序輸入資料集。例如，您的第一組輸入檔案應該命名為 inputs_1_pre_create.json、 inputs_1_pre_update.json和 inputs_1_pre_invalid.json。您的下一個集合會命名為 inputs_2_pre_create.json、inputs_2_pre_update.json和 inputs_2_pre_invalid.json，以此類推。

每個輸入檔案都是 JSON 檔案，只包含用於測試的資源屬性。

以下是 inputs 使用輸入檔案 Amazon S3 指定輸入資料的範例目錄。

```
inputs_1_pre_create.json
```

以下是inputs_1_pre_create.json合約測試的範例。

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "AccessControl": "BucketOwnerFullControl",
            "AnalyticsConfigurations": [],
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
```

```
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    },
    "AWS::SQS::Queue": {
        "resourceProperties": {
            "QueueName": "MyQueue",
            "KmsMasterKeyId": "KMS-KEY-ARN"
        }
    }
}
```

inputs_1_pre_update.json

以下是inputs_1_pre_update.json合約測試的範例。

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        },
        "previousResourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
```

```
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    }
}
```

inputs_1_invalid.json

以下是inputs_1_invalid.json合約測試的範例。

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "AccessControl": "BucketOwnerFullControl",
            "AnalyticsConfigurations": [],
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "ServerSideEncryptionByDefault": {
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    },
    "AWS::SQS::Queue": {
        "resourceProperties": {
            "NotValid": "The property of this resource is not valid."
        }
    }
}
```

inputs_1_invalid_pre_update.json

以下是inputs_1_invalid_pre_update.json合約測試的範例。

```
{
    "AWS::S3::Bucket": {
        "resourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        },
        "previousResourceProperties": {
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "BucketKeyEnabled": true,
                        "ServerSideEncryptionByDefault": {
                            "KMSMasterKeyID": "KMS-KEY-ARN",
                            "SSEAlgorithm": "aws:kms"
                        }
                    }
                ]
            },
            "BucketName": "encryptedbucket-us-west-2"
        }
    }
}
```

如需詳細資訊，請參閱《 CloudFormation CLI 使用者指南》中的發佈擴充功能以供公眾使用。

CloudFormation 勾點的結構描述語法參考

本節說明您用來開發 CloudFormation Hooks 的結構描述語法。

勾點包含由 JSON 結構描述和勾點處理常式表示的勾點規格。建立自訂勾點的第一步是建立定義勾點、其屬性及其屬性的結構描述模型。當您使用 CloudFormation CLI init命令初始化自訂 Hook 專案時，會為您建立 Hook 結構描述檔案。使用此結構描述檔案做為定義自訂勾點形狀和語意的起點。

結構描述語法

下列結構描述是勾點的結構。

```
{
"typeName": "string",
    "description": "string",
    "sourceUrl": "string",
    "documentationUrl": "string",
    "definitions": {
        "definitionName": {
          . . .
        }
    },
    "typeConfiguration": {
        "properties": {
             "propertyName": {
                "description": "string",
                "type": "string",
                 . . .
            },
        },
    "required": [
        "propertyName"
         . . .
            ],
    "additionalProperties": false
    },
    "handlers": {
        "preCreate": {
            "targetNames": [
            ],
            "permissions": [
            ]
        },
        "preUpdate": {
            "targetNames": [
            ],
            "permissions": [
            ]
        },
        "preDelete": {
            "targetNames": [
            ],
```

```
            "permissions": [
            ]
        }
    },
    "additionalProperties": false
}
```

`typeName`

勾點的唯一名稱。為您的勾點指定三個部分的命名空間，建議模式為 `Organization::Service::Hook`。

> **Note**
>
> 下列組織命名空間是預留的，無法用於您的勾點類型名稱：
>
> - `Alexa`
> - `AMZN`
> - `Amazon`
> - `ASK`
> - `AWS`
> - `Custom`
> - `Dev`

必要：是

模式：`^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$`

下限：`10`

上限：`196`

`description`

CloudFormation 主控台中顯示的勾點簡短描述。

必要：是

`sourceUrl`

如果為公有，Hook 的來源碼 URL。

必要：否

上限：4096

`documentationUrl`

為 勾點提供詳細文件的頁面 URL。

必要：是

模式：`^https\:\/\/[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])(\:[0-9]*)*([\?/#].*)?$`

上限：4096

> **Note**
>
> 雖然勾點結構描述應包含完整且準確的屬性描述，但您可以使用 documentationURL 屬性為使用者提供更多詳細資訊，包括範例、使用案例和其他詳細資訊。

`definitions`

使用 `definitions`區塊提供共用的 Hook 屬性結構描述。

最佳實務是使用 `definitions`區段來定義結構描述元素，可用於勾點類型結構描述中的多個點。然後，您可以使用 JSON 指標在勾點類型結構描述中的適當位置參考該元素。

必要：否

`typeConfiguration`

Hook 組態資料的定義。

必要：是

`properties`

勾點的屬性。勾點的所有屬性都必須在結構描述中表示。將勾點結構描述屬性與勾點類型組態屬性對齊。

> **Note**
>
> 不允許巢狀屬性。反之，請在 `definitions`元素中定義任何巢狀屬性，並使用`$ref`指標在所需的屬性中參考它們。

目前支援下列屬性：

- `default` – 屬性的預設值。
- `description` – 屬性的描述。
- `pattern` – 用於驗證輸入的 regex 模式。
- `type` – 屬性的已接受類型。

`additionalProperties`

`additionalProperties` 必須設定為 `false`。勾點的所有屬性都必須以結構描述表示：不允許任意輸入。

必要：是

有效值：`false`

`handlers`

處理常式指定可啟動結構描述中定義的勾點的操作，例如勾點叫用點。例如，`preUpdate`在處理常式中所有指定目標的更新操作之前，會叫用處理常式。

有效值：`preCreate | preUpdate | preDelete`

> **Note**
>
> 必須為處理常式指定至少一個值。

> **Important**
>
> 導致 狀態的堆疊操作`UpdateCleanup`不會叫用勾點。例如，在以下兩個案例中，不會叫用 Hook 的`preDelete`處理常式：
>
> - 堆疊會在從範本中移除一個資源後更新。
> - 會刪除具有替代更新類型的資源。

`targetNames`

Hook 目標的類型名稱字串陣列。例如，如果`preCreate`處理常式有 `AWS::S3::Bucket`目標，則 Hook 會在預先佈建階段執行 Amazon S3 儲存貯體。

- `TargetName`

為每個實作的處理常式指定至少一個目標名稱。

模式：^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$

下限：1

必要：是

> **Warning**
>
> SSM SecureString 和 Secrets Manager 動態參考在傳遞給 Hooks 之前不會解析。

`permissions`

字串陣列，指定叫用處理常式所需的 AWS 許可。

必要：是

`additionalProperties`

`additionalProperties` 必須設定為 `false`。勾點的所有屬性都必須以結構描述表示：不允許任意輸入。

必要：是

有效值：`false`

勾點結構描述範例

範例 1

Java 和 Python 演練使用以下程式碼範例。以下是名為 的勾點的範例結構`mycompany-testing-mytesthook.json`。

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies S3 bucket and SQS queues properties before create and
 update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
```

```
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[

        ],
        "additionalProperties":false
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preUpdate":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
            "permissions":[

            ]
        },
        "preDelete":{
            "targetNames":[
                "AWS::S3::Bucket",
                "AWS::SQS::Queue"
            ],
```

```
            "permissions":[
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetEncryptionConfiguration",
                "sqs:ListQueues",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ]
        }
    },
    "additionalProperties":false
}
```

範例 2

下列範例是使用 STACK和 CHANGE_SET 的結構描述targetNames，以堆疊範本和變更集操作為目標。

```
{
    "typeName":"MyCompany::Testing::MyTestHook",
    "description":"Verifies Stack and Change Set properties before create and update",
    "sourceUrl":"https://mycorp.com/my-repo.git",
    "documentationUrl":"https://mycorp.com/documentation",
    "typeConfiguration":{
        "properties":{
            "minBuckets":{
                "description":"Minimum number of compliant buckets",
                "type":"string"
            },
            "minQueues":{
                "description":"Minimum number of compliant queues",
                "type":"string"
            },
            "encryptionAlgorithm":{
                "description":"Encryption algorithm for SSE",
                "default":"AES256",
                "type":"string",
                "pattern": "[a-zA-Z]*[1-9]"
            }
        },
        "required":[
        ],
        "additionalProperties":false
```

```
    },
    "handlers":{
        "preCreate":{
            "targetNames":[
                "STACK",
                "CHANGE_SET"
            ],
            "permissions":[
            ]
        },
        "preUpdate":{
            "targetNames":[
                "STACK"
            ],
            "permissions":[
            ]
        },
        "preDelete":{
            "targetNames":[
                "STACK"
            ],
            "permissions":[

            ]
        }
    },
    "additionalProperties":false
}
```

停用和啟用 CloudFormation 勾點

本主題說明如何停用再重新啟用勾點，暫時防止其在您的帳戶中處於作用中狀態。當您需要在不受勾點干擾的情況下調查問題時，停用勾點會很有用。

在帳戶中停用和啟用勾點（主控台）

在帳戶中停用勾點

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/cloudformation 的 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 勾點所在的 。
3. 從導覽窗格中，選擇勾點。
4. 選擇您要停用的勾點名稱。
5. 在勾點詳細資訊頁面的勾點名稱右側，選擇停用按鈕。
6. 出現確認提示時，請選擇停用掛鉤。

重新啟用先前停用的勾點

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/cloudformation 的 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇 AWS 區域 勾點所在的 。
3. 從導覽窗格中，選擇勾點。
4. 選擇您要啟用的勾點名稱。
5. 在勾點詳細資訊頁面的勾點名稱右側，選擇啟用按鈕。
6. 出現確認提示時，請選擇啟用勾點。

在帳戶中停用和啟用勾點 (AWS CLI)

> **Important**
>
> 停用和啟用勾點的 AWS CLI 命令會將整個勾點組態取代為 `--configuration`選項中指定的值。若要避免意外變更，您必須包含執行這些命令時要保留的所有現有設定。若要檢視目前的組態資料，請使用 describe-type命令。

停用勾點

使用下列set-type-configuration命令，並將 指定HookInvocationStatus為 DISABLED以停用勾點。將預留位置取代為您的特定值。

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "DISABLED", "FailureMode": "FAIL",
 "TargetOperations": ["STACK","RESOURCE","CHANGE_SET"], "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

重新啟用先前停用的勾點

使用下列set-type-configuration命令，並將 指定HookInvocationStatus為 ENABLED以重新啟用勾點。將預留位置取代為您的特定值。

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL",
 "TargetOperations": ["STACK","RESOURCE","CHANGE_SET"], "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" \
  --region us-west-2
```

如需詳細資訊，請參閱勾點組態結構描述語法參考。

檢視 CloudFormation 勾點的調用結果

本主題說明如何檢視 CloudFormation 勾點的調用結果。檢視調用結果可協助您了解 Hooks 如何評估您的資源，並解決 Hooks 驗證資源時偵測到的任何問題。

當您的驗證邏輯（無論是 AWS Control Tower 主動控制、Guard 規則或 Lambda 函數）在資源的生命週期期間執行時，呼叫是特定的執行個體。

在主控台中檢視調用結果

您可以透過三種方式在主控台中檢視調用結果：透過調用摘要頁面、透過個別勾點的調用歷史記錄，或透過堆疊特定調用的個別堆疊事件。

檢視所有勾點的結果

調用摘要頁面提供過去 90 天內您帳戶和區域中所有 Hook 調用的完整檢視。

檢視所有勾點的結果

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇您要檢視勾點叫用的 AWS 區域 。
3. 從導覽窗格中，選擇叫用摘要。
4. 此頁面會顯示過去 90 天的所有勾點調用清單，包括：
   - 調用 ID
   - 勾點
   - Target
   - 模式 (`Warn` 或 `Fail`)
   - 結果 (`Warning`、`Pass`、`Failed`、`In progress`)
   - 調用時間
   - 結果訊息
5. 您可以使用資料表頂端的搜尋列來篩選清單，以尋找特定的調用。
6. 選取特定的調用，以檢視調用結果的更多新增詳細資訊，包括失敗的勾點調用的修補指引。

檢視個別勾點的調用歷史記錄

您也可以透過個別勾點的調用歷史記錄來檢視調用結果。

檢視特定勾點的勾點調用

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇您要檢視勾點叫用的 AWS 區域 。
3. 從導覽窗格中，選擇勾點。
4. 選擇您要檢視勾點調用的勾點。
5. 選取特定的調用，以檢視調用結果的更多新增詳細資訊，包括失敗的勾點調用的修補指引。

檢視堆疊特定調用的結果

您也可以透過堆疊事件頁面檢視特定堆疊的調用結果。

檢視特定堆疊的勾點叫用

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。
2. 在畫面頂端的導覽列上，選擇堆疊操作發生的 AWS 區域 。
3. 從導覽窗格選擇堆疊。
4. 選取您要檢視勾點調用的堆疊。
5. 選擇堆疊事件索引標籤。
6. 在事件清單中，尋找狀態原因欄中完成的勾點調用事件。
7. 若要檢視特定的勾點調用詳細資訊，請檢閱勾點調用欄，然後選擇加上底線的文字以開啟含有更多詳細資訊的快顯視窗。

> **Note**
>
> 若要顯示隱藏的資料欄，請選擇區段右上角的齒輪圖示以開啟偏好設定模式，視需要更新設定，然後選擇確認。

使用 檢視調用結果 AWS CLI

使用 list-hook-results命令來擷取有關勾點調用的資訊。此命令支援下列篩選選項：

- 取得所有勾點調用結果 （ 不需要參數 ）
- 依勾點 ARN 篩選 （ 使用 `--type-arn`)
- 依勾點 ARN 和狀態篩選 （ 使用 `--type-arn`和 `--status`)
- 搜尋特定目標 （ 使用 `--target-type`和 `--target-id`)

依勾點 ARN 篩選結果

下列命令會列出特定勾點的所有勾點調用結果。

```
aws cloudformation list-hook-results \
  --type-arn arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-
ComplianceHook \
  --region us-west-2
```

輸出範例：

```
{
  "HookResults": [
    {
      "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook",
      "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
      "TypeName": "MyOrg::Security::ComplianceHook",
      "TypeVersionId": "00000001",
      "HookExecutionTarget": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/39f29d10-73ed-11f0-abc1-0affdfe4aebb",
      "InvokedAt": "2025-08-08T00:18:39.651Z",
      "FailureMode": "WARN",
      "HookStatusReason": "...",
      "InvocationPoint": "PRE_PROVISION",
      "Status": "HOOK_COMPLETE_FAILED"
    },
    ...
  ]
}
```

如需回應中欄位的說明，請參閱 AWS CloudFormation API 參考中的 HookResultSummary。

依勾點 ARN 和狀態篩選結果

若要篩選結果之間的常見狀態，請在 命令中指定 `--status`選項。有效的值如下：

- `HOOK_IN_PROGRESS`：Hook 目前正在執行。
- `HOOK_COMPLETE_SUCCEEDED`：勾點已成功完成。
- `HOOK_COMPLETE_FAILED`：勾點已完成但驗證失敗。
- `HOOK_FAILED`：Hook 在執行期間遇到錯誤。

```
aws cloudformation list-hook-results \
  --type-arn arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-
ComplianceHook \
  --status HOOK_COMPLETE_FAILED \
  --region us-west-2
```

輸出範例：

```
{
  "HookResults": [
    {
      "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook",
      "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
      "TypeName": "MyOrg::Security::ComplianceHook",
      "TypeVersionId": "00000001",
      "HookExecutionTarget": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/39f29d10-73ed-11f0-abc1-0affdfe4aebb",
      "InvokedAt": "2025-08-08T00:18:39.651Z",
      "FailureMode": "WARN",
      "HookStatusReason": "...",
      "InvocationPoint": "PRE_PROVISION",
      "Status": "HOOK_COMPLETE_FAILED"
    },
    ...
  ]
}
```

如需回應中欄位的說明，請參閱 AWS CloudFormation API 參考中的 HookResultSummary。

依目標類型和目標 ID 篩選結果

下列命令會列出特定 Cloud Control API 請求的所有勾點調用結果。

```
aws cloudformation list-hook-results \
  --target-type CLOUD_CONTROL \
  --target-id d417b05b-9eff-46ef-b164-08c76aec1801 \
  --region us-west-2
```

輸出範例：

```
{
  "HookResults": [
    {
      "TargetType": "CLOUD_CONTROL",
      "TargetId": "d417b05b-9eff-46ef-b164-08c76aec1801",
      "HookResults": [
        {
          "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-
Security-ComplianceHook",
          "HookResultId": "4e7f4766-d8fe-44e5-8587-5b327a148abe",
          "TypeName": "MyOrg::Security::ComplianceHook",
          "TypeVersionId": "00000001",
          "FailureMode": "WARN",
          "HookStatusReason": "...",
          "InvocationPoint": "PRE_PROVISION",
          "Status": "HOOK_COMPLETE_FAILED"
        },
        ...
      ]
    }
  ]
}
```

如需回應中欄位的說明，請參閱 AWS CloudFormation API 參考中的 HookResultSummary。

取得特定調用的詳細結果

使用 get-hook-result命令來擷取特定勾點調用的詳細資訊，包括合規檢查結果和修補指導的註釋。

```
aws cloudformation get-hook-result \
  --hook-result-id 59ef501c-0ac4-47c0-a193-e071cabf748d \
```

```
  --region us-west-2
```

輸出範例：

```
{
  "HookResultId": "59ef501c-0ac4-47c0-a193-e071cabf748d",
  "InvocationPoint": "PRE_PROVISION",
  "FailureMode": "WARN",
  "TypeName": "MyOrg::Security::ComplianceHook",
  "TypeVersionId": "00000001",
  "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyOrg-Security-
ComplianceHook",
  "Status": "HOOK_COMPLETE_FAILED",
  "HookStatusReason": "Hook completed with failed validations",
  "InvokedAt": "2025-08-08T00:18:39.651Z",
  "Target": {
    "TargetType": "RESOURCE",
    "TargetTypeName": "AWS::S3::Bucket",
    "TargetId": "my-s3-bucket",
    "Action": "CREATE"
  },
  "Annotations": [
    {
      "AnnotationName": "BlockPublicAccessCheck",
      "Status": "FAILED",
      "StatusMessage": "Bucket does not block public access",
      "RemediationMessage": "Enable block public access settings on the S3 bucket",
      "SeverityLevel": "HIGH"
    },
    {
      "AnnotationName": "BucketEncryptionCheck",
      "Status": "PASSED",
      "StatusMessage": "Bucket has encryption configured correctly"
    }
  ]
}
```

如需回應中欄位的說明，請參閱 AWS CloudFormation API 參考中的 GetHookResult。

勾點組態結構描述語法參考

本節概述用於設定勾點的結構描述語法。在 中叫用勾點時，CloudFormation 會在執行時間使用此組態結構描述 AWS 帳戶。

若要讓 Hook 主動檢查堆疊的組態，請在帳戶中註冊和啟用 Hook `HookInvocationStatus`之後，將 設定為 `ENABLED`。

主題

> **Note**
>
> Hook 組態可存放的最大資料量為 300 KB。這是對 SetTypeConfiguration操作的`Configuration`請求參數施加的所有限制之外的。

勾點組態結構描述屬性

下列結構描述是勾點組態結構描述的結構。

```
{
    "CloudFormationConfiguration": {
        "HookConfiguration": {
            "HookInvocationStatus": "ENABLED",
            "TargetOperations": ["STACK"],
            "FailureMode": "FAIL",
            "EncryptionConfiguration": {
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/abc-123"
            },
            "Properties": {
                ...
            }
```

```
        }
    }
}
```

HookConfiguration

勾點組態支援在堆疊層級、失敗模式和勾點屬性值啟用或停用勾點。

勾點組態支援下列屬性。

HookInvocationStatus

指定勾點是 ENABLED還是 DISABLED。

有效值：ENABLED | DISABLED

TargetOperations

指定要執行勾點的操作清單。如需詳細資訊，請參閱掛鉤目標。

有效值：STACK | RESOURCE | CHANGE_SET | CLOUD_CONTROL

TargetStacks

可用於回溯相容性。請*HookInvocationStatus*改用 。

如果模式設定為 ALL，勾點會在 UPDATE、 CREATE或 DELETE 資源操作期間套用至您帳戶中的所有堆疊。

如果模式設定為 NONE，則勾點不會套用至您帳戶中的堆疊。

有效值：ALL | NONE

FailureMode

此欄位會告知服務如何處理勾點故障。

- 如果模式設定為 FAIL，且勾點失敗，則失敗組態會停止佈建資源並復原堆疊。
- 如果模式設定為 WARN且勾點失敗，則警告組態允許佈建繼續出現警告訊息。

有效值：FAIL | WARN

EncryptionConfiguration

指定勾點註釋資料的加密設定。

`KmsKeyId`

用於加密 Hook 註釋資料的對稱加密 AWS KMS 金鑰的別名、別名 ARN、金鑰 ID 或金鑰 ARN。如需詳細資訊，請參閱 AWS KMS 文件中的 KeyId。

在使用客戶受管 AWS KMS 金鑰建立勾點之前，您的使用者或角色必須具有 `DescribeKey`和 的 AWS KMS 許可`GenerateDataKey`。如需詳細資訊，請參閱AWS KMS 加密靜態 CloudFormation Hooks 結果的金鑰政策和許可。

`Properties`

指定勾點執行時間屬性。這些應符合 Hooks 結構描述支援的屬性形狀。

勾點組態範例

如需從 設定勾點的範例 AWS CLI，請參閱下列各節：

- 啟用主動控制型勾點 (AWS CLI)
- 啟用 Guard Hook (AWS CLI)
- 啟用 Lambda 勾點 (AWS CLI)

CloudFormation 勾點堆疊層級篩選條件

您可以將堆疊層級篩選條件新增至 CloudFormation Hooks，以根據堆疊名稱和角色將特定堆疊設為目標。當您有多個具有相同資源類型的堆疊，但勾點適用於特定堆疊時，這非常有用。

本節說明這些篩選條件的運作方式，並提供您可以遵循的範例。

沒有堆疊層級篩選的勾點組態基本結構如下所示：

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
```

```
      "TargetFilters": {
        "Actions": [
          "CREATE",
          "UPDATE",
          "DELETE"
        ]
      }
    }
  }
}
```

如需HookConfiguration語法的詳細資訊，請參閱 勾點組態結構描述語法參考。

若要使用堆疊層級篩選條件，請在 下新增StackFilters金鑰HookConfiguration。

StackFilters 金鑰有一個必要成員，並且有兩個選用成員。

- FilteringCriteria (必要)
- StackNames (選用)
- StackRoles (選用)

StackNames 或 StackRoles 屬性是選用的。但是，您必須指定這些屬性中至少其中一種屬性。

如果您建立以 Cloud Control API 操作為目標的勾點，則會忽略所有堆疊層級篩選條件。

FilteringCriteria

FilteringCriteria 是指定篩選行為的必要參數。它可以設定為 ALL或 ANY。

- ALL 如果所有篩選條件都相符， 會叫用勾點。
- ANY 如果有任何相符的篩選條件， 會叫用勾點。

StackNames

若要在勾點組態中將一或多個堆疊名稱指定為篩選條件，請使用下列 JSON 結構：

```
"StackNames": {
  "Include": [
    "string"
```

```
  ],
  "Exclude": [
    "string"
  ]
}
```

您必須指定下列其中一項：

- `Include`：要包含的堆疊名稱清單。只有此清單中指定的堆疊才會叫用勾點。
  - 類型：字串陣列
  - 項目上限：50
  - 最小項目：1
- `Exclude`：要排除的堆疊名稱清單。除了此處列出的堆疊之外，所有堆疊都會叫用勾點。
  - 類型：字串陣列
  - 項目上限：50
  - 最小項目：1

`Include` 和 `Exclude`陣列中的每個堆疊名稱必須遵循下列模式和長度要求：

- 模式：`^[a-zA-Z][-a-zA-Z0-9]*$`
- 長度上限：128

`StackNames` 支援具體堆疊名稱和完整萬用字元比對。若要查看使用萬用字元的範例，請參閱 使用萬用字元搭配勾點目標名稱。

StackRoles

若要在勾點組態中將一或多個 IAM 角色指定為篩選條件，請使用下列 JSON 結構：

```
"StackRoles": {
  "Include": [
    "string"
  ],
  "Exclude": [
    "string"
  ]
}
```

您必須指定下列其中一項：

- Include：與這些角色相關聯的目標堆疊的 IAM 角色 ARNs 清單。只有這些角色啟動的堆疊操作才會叫用勾點。
  - 類型：字串陣列
  - 項目上限：50
  - 最小項目：1
- Exclude：您要排除之堆疊的 IAM 角色 ARNs 清單。所有堆疊都會叫用勾點，但由指定角色啟動的堆疊除外。
  - 類型：字串陣列
  - 項目上限：50
  - 最小項目：1

Include 和 Exclude陣列中的每個堆疊角色必須遵循下列模式和長度要求：

- 模式：arn:.+:iam::[0-9]{12}:role/.+
- 長度上限：256

StackRoles 在下列 ARN 語法區段中允許萬用字元：

- partition
- account-id
- resource-id

若要在 ARN 語法區段中查看使用萬用字元的範例，請參閱 使用萬用字元搭配勾點目標名稱。

Include 和 Exclude

每個篩選條件 (StackNames 和 StackRoles) 都有Include清單和Exclude清單。使用 StackNames 作為範例，勾點只會在Include清單中指定的堆疊上調用。如果只在Exclude清單中指定堆疊名稱，則只會在不在Exclude清單中的堆疊上叫用掛鉤。如果同時指定 Exclude Include和 ，Hook 會以Include清單中的內容為目標，而不是Exclude清單中的內容為目標。

例如，假設您有四個堆疊：A、B、C 和 D。

- "Include": ["A","B"] 在 A 和 B 上調用勾點。

- "Exclude": ["B"] 在 A、C 和 D 上調用勾點。
- "Include": ["A","B","C"], "Exclude": ["A","D"] 在 B 和 C 上調用勾點。
- "Include": ["A","B","C"], "Exclude": ["A”,"B","C"] 任何堆疊上都不會叫用勾點。

堆疊層級篩選條件的範例

本節提供您可以遵循的範例，為 CloudFormation 勾點建立堆疊層級篩選條件。

範例 1：包含特定堆疊

下列範例會指定Include清單。僅在名為 stack-test-1、 stack-test-2和 的堆疊上叫用勾點stack-test-3。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

範例 2：排除特定堆疊

如果堆疊名稱改為新增至Exclude清單，則會在未命名為 stack-test-1、 stack-test-2或 的任何堆疊上叫用勾點stack-test-3。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

範例 3：合併包含和排除

如果未指定 Include和 Exclude清單，則只會在不在Exclude清單中Include的 堆疊上叫用勾點。在下列範例中，僅在 上叫用勾點stack-test-3。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
```

```
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ],
          "Exclude": [
            "stack-test-1",
            "stack-test-2"
          ]
        }
      }
    }
  }
}
```

範例 4：結合堆疊名稱和角色與**ALL**條件

下列勾點包含三個堆疊名稱和一個堆疊角色。由於 FilteringCriteria 指定為 ALL，因此只會針對同時具有相符堆疊名稱和相符堆疊角色的堆疊叫用勾點。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        },
        "StackRoles": {
          "Include": ["arn:aws:iam::123456789012:role/hook-role"]
        }
      }
    }
```

```
  }
}
```

範例 5：結合堆疊名稱和角色與**ANY**條件

下列勾點包含三個堆疊名稱和一個堆疊角色。因為 FilteringCriteria指定為 ANY，所以針對具有相符堆疊名稱或相符堆疊角色的堆疊叫用勾點。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ANY",
        "StackNames": {
          "Include": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        },
        "StackRoles": {
            "Include": ["arn:aws:iam::123456789012:role/hook-role"]
        }
      }
    }
  }
}
```

CloudFormation 掛鉤目標篩選條件

本主題提供設定 CloudFormation Hooks 目標篩選條件的指引。您可以使用目標篩選條件來更精細地控制您的勾點被調用的時間和資源。您可以設定篩選條件，範圍從簡單的資源類型目標到更複雜的資源類型、動作和調用點組合。

若要在勾點組態中將一或多個堆疊名稱指定為篩選條件，請在 下新增TargetFilters金鑰HookConfiguration。

TargetFilters 支援下列屬性。

Actions

指定要鎖定之動作的字串陣列。如需範例，請參閱 範例 1：基本目標篩選條件。

有效值：CREATE | UPDATE | DELETE

> **Note**
>
> 對於 RESOURCE、 STACK和 CLOUD_CONTROL目標，所有目標動作都適用。對於CHANGE_SET目標，只有 CREATE動作適用。如需詳細資訊，請參閱掛鉤目標。

InvocationPoints

字串陣列，指定對目標的調用點。

有效值：PRE_PROVISION

TargetNames

字串陣列，指定要鎖定的資源類型名稱，例如 AWS::S3::Bucket。

目標名稱支援具體的目標名稱和完整的萬用字元比對。如需詳細資訊，請參閱使用萬用字元搭配勾點目標名稱。

模式：^[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}::[a-zA-Z0-9]{2,64}$

上限：50

Targets

物件陣列，指定要用於目標篩選的目標清單。

目標陣列中的每個目標都有下列屬性。

Actions

指定目標的動作。

有效值：CREATE | UPDATE | DELETE

`InvocationPoints`

指定目標的調用點。

有效值：PRE_PROVISION

`TargetNames`

要鎖定的資源類型名稱。

> **Note**
>
> 您無法同時包含Targets物件陣列和 Actions、 TargetNames或 InvocationPoints陣列。如果您想要使用這三個項目 和 Targets，您必須在Targets物件陣列中包含它們。如需範例，請參閱 範例 2：使用Targets物件陣列。

目標篩選條件的範例

本節提供您可以遵循的範例，為 CloudFormation 勾點建立目標篩選條件。

範例 1：基本目標篩選條件

若要建立著重於特定資源類型的基本目標篩選條件，請使用 TargetFilters 物件搭配 Actions陣列。下列目標篩選條件組態會在指定目標操作的所有 Create、 Update和 Delete動作上叫用 勾點（在此情況下為 RESOURCE和 STACK操作）。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Actions": [
           "Create",
```

```
            "Update",
            "Delete"
          ]
        }
      }
    }
}
```

範例 2：使用**Targets**物件陣列

對於更進階的篩選條件，您可以使用Targets物件陣列來列出特定目標、動作和調用點組合。下列目標篩選條件組態會在 S3 儲存貯體CREATE和 DynamoDB 資料表上叫用 掛鉤之前和UPDATE動作。它同時適用於 STACK和 RESOURCE操作。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "TargetFilters": {
        "Targets": [
          {
             "TargetName": "AWS::S3::Bucket",
             "Action": "CREATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::S3::Bucket",
             "Action": "UPDATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::DynamoDB::Table",
             "Action": "CREATE",
             "InvocationPoint": "PRE_PROVISION"
          },
          {
             "TargetName": "AWS::DynamoDB::Table",
```

```
            "Action": "UPDATE",
            "InvocationPoint": "PRE_PROVISION"
          }
        ]
      }
    }
  }
}
```

使用萬用字元搭配勾點目標名稱

您可以使用萬用字元做為目標名稱的一部分。您可以在 Hook 目標名稱中使用萬用字元 (* 和 ?)。星號 (*) 代表字元的任意組合。問號 (?) 代表任何單一字元。您可以在目標名稱中使用多個 *和 ? 字元。

Example：Hook 結構描述中目標名稱萬用字元的範例

下列範例以 Amazon S3 支援的所有資源類型為目標。

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::S3::*"
            ],
            "permissions": []
       }
    }
...
}
```

下列範例符合名稱中具有「Bucket」的所有資源類型。

```
{
...
    "handlers": {
       "preCreate": {
            "targetNames": [
                "AWS::*::Bucket*"
            ],
            "permissions": []
       }
```

```
    }
...
}
```

AWS::*::Bucket* 可能會解析為下列任何具體資源類型：

- AWS::Lightsail::Bucket
- AWS::S3::Bucket
- AWS::S3::BucketPolicy
- AWS::S3Outpost::Bucket
- AWS::S3Outpost::BucketPolicy

Example：Hook 組態結構描述中目標名稱萬用字元的範例

下列範例組態會叫用勾點，用於所有 Amazon S3 資源類型的CREATE操作，以及所有具名資料表資源類型的UPDATE操作，例如 AWS::DynamobDB::Table或 AWS::Glue::Table。

```
{
   "CloudFormationConfiguration": {
        "HookConfiguration": {
            "TargetStacks": "ALL",
            "FailureMode": "FAIL",
            "Properties": {},
            "TargetFilters":{
                 "Targets": [
                    {
                        "TargetName": "AWS::S3::*",
                        "Action": "CREATE",
                        "InvocationPoint": "PRE_PROVISION"
                    },
                    {
                        "TargetName": "AWS::*::Table",
                        "Action": "UPDATE",
                        "InvocationPoint": "PRE_PROVISION"
                    }
                ]
            }
        }
   }
}
```

下列範例組態會叫用所有 Amazon S3 資源類型的 和 CREATEUPDATE操作的勾點，以及所有具名資料表資源類型的 CREATE和 UPDATE操作，例如 AWS::DynamobDB::Table或 AWS::Glue::Table。

```
{
   "CloudFormationConfiguration": {
        "HookConfiguration": {
            "TargetStacks": "ALL",
            "FailureMode": "FAIL",
            "Properties": {},
            "TargetFilters":{
                "TargetNames": [
                    "AWS::S3::*",
                    "AWS::*::Table"
                ],
                "Actions": [
                    "CREATE",
                    "UPDATE"
                ],
                "InvocationPoints": [
                    "PRE_PROVISION"
                ]
            }
        }
   }
}
```

Example：Include特定堆疊

下列範例指定 Include清單。只有在堆疊名稱以 開頭時，才會叫用勾點stack-test-。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
```

```
          "Include": [
            "stack-test-*"
          ]
        }
      }
    }
  }
}
```

Example：Exclude特定堆疊

下列範例指定 Exclude清單。在未以 開頭的任何堆疊上叫用勾點stack-test-。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Exclude": [
            "stack-test-*"
          ]
        }
      }
    }
  }
}
```

Example：Exclude針對特定堆疊結合 Include和

如果指定 Include和 Exclude 清單，則只會在清單中不相符Include的 中符合的堆疊上叫用勾點Exclude。在下列範例中，除了名為 stack-test-1、 和 的堆疊stack-test-之外stack-test-2，在開頭為 的所有堆疊上叫用勾點stack-test-3。

```
{
  "CloudFormationConfiguration": {
```

```
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ],
          "Exclude": [
            "stack-test-1",
            "stack-test-2",
            "stack-test-3"
          ]
        }
      }
    }
  }
}
```

Example：Include特定角色

下列範例會指定具有兩種萬用字元模式的Include清單。第一個項目將針對任何 partition和hook-role中開頭為 的任何角色執行勾點account-id。第二個項目將為partition屬於 的任何account-id 中的任何角色執行 123456789012。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
```

```
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/hook-role*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
}
```

Example：Exclude特定角色

下列範例會指定具有兩種萬用字元模式的Exclude清單。第一個項目會在角色exempt名稱中包含任何 partition和任何 時略過勾點執行account-id。當屬於 的角色與堆疊操作account-id123456789012搭配使用時，第二個項目會略過勾點執行。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Exclude": [
            "arn:*:iam::*:role/*exempt*",
            "arn:*:iam::123456789012:role/*
          ]
        }
      }
    }
  }
}
```

Example：結合特定角色 ARN 模式Exclude的 Include和

如果指定 Include和 Exclude 清單，則只會在與Exclude清單中不相符的角色相符Include的堆疊上使用勾點。在下列範例中，使用任何 partition、 account-id和 role名稱在堆疊操作上叫用勾點，除非角色屬於 account-id 123456789012。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackRoles": {
          "Include": [
            "arn:*:iam::*:role/*"
          ],
          "Exclude": [
            "arn:*:iam::123456789012:role/*"
          ]
        }
      }
    }
  }
}
```

Example：結合堆疊名稱和角色與所有條件

下列勾點包含一個堆疊名稱萬用字元和一個堆疊角色萬用字元。由於 FilteringCriteria 指定為 ALL，因此只會針對同時具有相符StackName和相符 的堆疊叫用勾點StackRoles。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
```

```
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ALL",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ]
        },
        "StackRoles": {
          "Include": ["arn:*:iam::*:role/hook-role*"]
        }
      }
    }
  }
}
```

Example：結合 StackNames和 StackRoles與任何條件

下列勾點包含一個堆疊名稱萬用字元和一個堆疊角色萬用字元。由於 FilteringCriteria 指定為 ANY，因此會針對具有相符StackNames或相符 的堆疊叫用 勾點StackRoles。

```
{
  "CloudFormationConfiguration": {
    "HookConfiguration": {
      "HookInvocationStatus": "ENABLED",
      "TargetOperations": [
        "STACK",
        "RESOURCE"
      ],
      "FailureMode": "WARN",
      "Properties": {},
      "StackFilters": {
        "FilteringCriteria": "ANY",
        "StackNames": {
          "Include": [
            "stack-test-*"
          ]
        },
        "StackRoles": {
            "Include": ["arn:*:iam::*:role/hook-role*"]
        }
```

```
        }
      }
    }
}
```

使用 CloudFormation 範本建立勾點

此頁面提供 Hooks 的範例 CloudFormation 範本和技術參考主題的連結。

透過使用 CloudFormation 範本建立勾點，您可以重複使用範本來一致且重複地設定勾點。此方法可讓您定義勾點一次，然後在多個 AWS 帳戶 和 區域中逐一佈建相同的勾點。

CloudFormation 提供下列特殊資源類型，用於建立 Guard 和 Lambda Hook。

任務	解決方案	連結
建立 Guard Hook	使用 `AWS::CloudFormation::GuardHook` 資源類型來建立和啟用 Guard Hook。	範本範例 技術參考
建立 Lambda 勾點	使用 `AWS::CloudFormation::LambdaHook` 資源類型來建立和啟用 Lambda Hook。	範本範例 技術參考

CloudFormation 也提供下列資源類型，您可以在堆疊範本中用於建立自訂勾點。

任務	解決方案	連結
註冊勾點	使用 `AWS::CloudFormation::HookVersion` 資源類型，將新版或第一個版本的自訂勾點發佈至 CloudFormation 登錄檔。	範例範本 技術參考
設定勾點的組態	使用 `AWS::CloudFormation::HookTypeConfig` 資源類型來指定自訂勾點的組態。	範例範本 技術參考
設定勾點的預設版本	使用 `AWS::CloudFormation::HookDefaultVersion` 資源類型來指定自訂勾點的預設版本。	範例範本 技術參考
將您的帳戶註冊為發佈者	使用 `AWS::CloudFormation::Publisher` 資源類型，在	技術參考

任務	解決方案	連結
	CloudFormation 登錄檔中將您的帳戶註冊為公有擴充功能（勾點、模組和資源類型）的發佈者。	
公開發佈勾點	使用 AWS::CloudFormation::PublicTypeVersion 資源類型來測試和發佈已註冊的自訂勾點做為公有第三方勾點。	技術參考
啟用公有、第三方勾點	AWS::CloudFormation::TypeActivation 資源類型會與 AWS::CloudFormation::HookTypeConfig 資源類型搭配使用，以在您的帳戶中啟用公有第三方自訂勾點。	技術參考

授予 CloudFormation Hooks 的 IAM 許可

根據預設，您 中的新使用者 AWS 帳戶 沒有使用 AWS 管理主控台、 AWS Command Line Interface (AWS CLI) 或 AWS API 管理勾點的許可。若要授予使用者許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

使用本主題中的政策範例來建立您自己的自訂 IAM 政策，以授予使用者使用 Hooks 的許可。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的使用客戶受管政策定義自訂 IAM 許可。

本主題涵蓋執行下列動作所需的許可：

- 管理勾點 – 在帳戶中建立、修改和停用勾點。
- 公開發佈勾點 – 註冊、測試和發佈您的自訂勾點，以便在 CloudFormation 登錄檔中公開提供。
- 檢視調用結果 – 存取和查詢您帳戶中的 Hook 調用結果。
- 檢視調用結果的詳細資訊 – 存取您帳戶中特定 Hook 調用結果的詳細資訊和修補指導。

建立 IAM 政策時，您可以在`cloudformation`服務授權參考 章節的動作、資源和條件索引鍵中找到與服務字首相關聯的所有動作、資源和條件索引鍵 CloudFormation的文件。

主題

允許使用者管理勾點

如果您需要允許使用者管理延伸模組，包括勾點，而不能在 CloudFormation 登錄檔中公開這些延伸模組，您可以使用下列範例 IAM 政策。

> **Important**
>
> `ActivateType` 和 `SetTypeConfiguration` API 呼叫一起運作，在您的 帳戶中建立勾點。當您授予使用者呼叫 `SetTypeConfiguration` API 的許可時，您會自動授予他們修改和停用現有勾點的能力。您無法使用資源層級許可來限制對此 API 呼叫的存取。因此，請確定您僅將此許可授予帳戶中的授權使用者。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ActivateType",
                "cloudformation:DescribeType",
                "cloudformation:ListTypes",
                "cloudformation:SetTypeConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

管理勾點的使用者可能需要一些相關的許可，例如：

- 若要在 CloudFormation 主控台中從 Control Catalog 檢視主動控制，使用者必須擁有 IAM 政策中的 `controlcatalog:ListControls`許可。
- 若要在 CloudFormation 登錄檔中將自訂勾點註冊為私有延伸，使用者必須在 IAM 政策中擁有 `cloudformation:RegisterType`許可。

允許使用者公開發佈自訂勾點

下列範例 IAM 政策特別著重於發佈功能。如果您需要允許使用者在 CloudFormation 登錄檔中公開使用延伸項目，包括勾點，請使用此政策。

> **Important**
>
> 發佈勾點可公開提供給其他 AWS 帳戶。確保只有授權使用者具有這些許可，且已發佈的延伸符合組織的品質和安全標準。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribePublisher",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:ListTypes",
                "cloudformation:ListTypeVersions",
                "cloudformation:PublishType",
                "cloudformation:RegisterPublisher",
                "cloudformation:RegisterType",
                "cloudformation:TestType"
            ],
            "Resource": "*"
        }
    ]
}
```

允許使用者檢視勾點調用結果

檢視勾點調用結果所需的 IAM 許可會根據請求的資訊類型而變更。

列出勾點調用結果

若要列出勾點調用結果，使用者需要不同的許可，具體取決於提出的 API 請求。

- 若要授予請求所有勾點結果、特定勾點結果或特定勾點和調用狀態結果的許可，您必須授予 `cloudformation:ListAllHookResults` 動作的存取權。

- 若要透過指定勾點目標授予請求結果的許可，您必須授予 cloudformation:ListHookResults 動作的存取權。此許可允許 API 呼叫者在呼叫 時指定 TargetType和 TargetId 參數ListHookResults。

以下顯示列出勾點調用結果的基本許可政策範例。具有此政策的 IAM 身分 （使用者或角色） 具有使用所有可用參數組合來請求所有調用結果的許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListAllHookResults",
                "cloudformation:ListHookResults"
            ],
            "Resource": "*"
        }
    ]
}
```

控制可以指定哪些變更集

下列範例 IAM 政策會指定 勾點的目標，授予 cloudformation:ListHookResults 動作請求結果的許可。不過，如果目標是名為 的變更集，也會拒絕該動作example-changeset。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListHookResults"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudformation:ChangeSetName": "example-changeset"
                }
            }
        }
    ]
}
```

控制可以指定哪些勾點

下列範例 IAM 政策授予 `cloudformation:ListAllHookResults` 動作的許可，僅在請求中提供 Hook 的 ARN 時請求調用結果。它會拒絕指定 Hook ARN 的動作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
```

```
                    "cloudformation:TypeArn": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "cloudformation:ListAllHookResults"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "cloudformation:TypeArn": "arn:aws:cloudformation:us-
east-1:123456789012:type/hook/MyCompany-MyHook"
                }
            }
        }
    ]
}
```

允許使用者檢視詳細的勾點調用結果

若要授予許可來檢視特定 Hook 調用的詳細結果，您必須授予 `cloudformation:GetHookResult` 動作的存取權。此許可允許使用者擷取特定 Hook 調用結果的詳細資訊和修補指導。如需詳細資訊，請參閱《 AWS CloudFormation API 參考》中的 GetHookResult。

下列範例 IAM 政策授予 `cloudformation:GetHookResult`動作的許可。

JSON

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

```
}
```

> **Note**
>
> 您可以設定勾點，使用您自己的 AWS KMS 金鑰加密存放在雲端中的詳細調用結果。如需有關如何在使用客戶受管金鑰進行加密時設定金鑰政策和所需的 IAM 許可的資訊，請參閱 AWS KMS 加密靜態 CloudFormation Hooks 結果的金鑰政策和許可。

AWS KMS 加密靜態 CloudFormation Hooks 結果的金鑰政策和許可

本主題說明如何設定當您指定客戶受管金鑰來加密可從 GetHookResult API 取得的 Hooks 註釋資料時所需的 AWS KMS 金鑰政策和許可。

> **Note**
>
> CloudFormation 勾點不需要額外的授權，即可使用預設值 AWS 擁有的金鑰 來加密帳戶中的註釋資料。

主題

概觀

下列項目 AWS KMS keys 可用於加密 Hook 註釋資料：

- AWS 擁有的金鑰 – 根據預設，CloudFormation 會使用 AWS 擁有的金鑰 來加密資料。您無法檢視、管理或使用 AWS 擁有的金鑰或稽核其使用方式。不過，您不需要執行明確的組態來保護用來加密資料的金鑰。 AWS 擁有的金鑰 是免費的 （無需每月費用或使用費）。除非您需要稽核或控制保護註釋資料的加密金鑰，否則 AWS 擁有的金鑰 是不錯的選擇。

- 客戶受管金鑰 – CloudFormation 支援使用您建立、擁有和管理的對稱客戶受管金鑰，在現有的上新增第二層加密 AWS 擁有的金鑰。需支付 AWS KMS 費用。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的建立金鑰。若要管理您的金鑰，請在 AWS Key Management Service AWS KMS 主控台 AWS CLI、 或 AWS KMS API 中使用 (AWS KMS)。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》https://docs.aws.amazon.com/kms/latest/developerguide/。

您可以在建立和更新勾點時設定客戶受管金鑰。當您提供客戶受管金鑰時，CloudFormation 會使用此金鑰在儲存註釋資料之前對其進行加密。當註釋資料稍後在 `GetHookResult` API 操作期間存取時，CloudFormation 會自動將其解密。如需設定 Hooks 加密金鑰的相關資訊，請參閱 勾點組態結構描述語法參考。

> **Important**
>
> 請注意，指定客戶受管金鑰`KmsKeyId`的選項目前只有在您使用 AWS CLI 來設定勾點時才能使用。

使用加密內容控制對客戶受管金鑰的存取

CloudFormation Hooks 會自動在每個註釋儲存和擷取操作中包含加密內容。這可讓您在金鑰政策中設定加密內容條件，以確保金鑰只能用於特定勾點：

- `kms:EncryptionContext:aws:cloudformation:hooks:service` – 確保金鑰僅供 CloudFormation Hooks 服務使用。
- `kms:EncryptionContext:aws:cloudformation:account-id` – 透過比對您的 AWS 帳戶 ID 來防止跨帳戶金鑰使用。
- `kms:EncryptionContext:aws:cloudformation:arn` – 使用 ARN 模式限制特定勾點的使用。

這些條件透過密碼編譯方式將加密的資料繫結至特定 Hook 內容，提供額外的保護，避免混淆代理人攻擊。

客戶受管 KMS 金鑰政策

建立客戶受管金鑰時，您必須定義其金鑰政策，以允許 CloudFormation Hooks 服務執行 AWS KMS 操作。若要使用下列金鑰政策，請將`#####`取代為您自己的資訊。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnableIAMUserDescribeKey",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "EnableIAMUserGenerateDataKey",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:GenerateDataKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
 "hooks.cloudformation.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:123456789012": "123456789012"
        },
        "ArnLike": {
          "kms:EncryptionContext:aws:cloudformation:arn":
 "arn:aws:cloudformation:*:123456789012:hook/*"
        }
      }
    },
    {
      "Sid": "EnableIAMUserDecrypt",
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExampleRole"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "cloudformation.us-east-1.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AllowHooksServiceDescribeKey",
      "Effect": "Allow",
      "Principal": {
        "Service": "hooks.cloudformation.amazonaws.com"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:*:123456789012:hook/*"
        }
      }
    },
    {
      "Sid": "AllowHooksService",
      "Effect": "Allow",
      "Principal": {
        "Service": "hooks.cloudformation.amazonaws.com"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012",
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
"hooks.cloudformation.amazonaws.com",
```

```
            "kms:EncryptionContext:aws:cloudformation:123456789012": "123456789012"
          },
          "ArnLike": {
            "aws:SourceArn": "arn:aws:cloudformation:*:123456789012:hook/*",
            "kms:EncryptionContext:aws:cloudformation:arn":
 "arn:aws:cloudformation:*:123456789012:hook/*"
          }
        }
      }
    ]
}
```

此政策會將許可授予 IAM 角色（前三個陳述式）和 CloudFormation Hooks 服務（最後兩個陳述式）。`kms:ViaService` 條件金鑰可確保 KMS 金鑰只能透過 CloudFormation 使用，防止直接 KMS API 呼叫。金鑰操作包括：

- `kms:DescribeKey` – 驗證金鑰屬性和中繼資料。此操作位於不同的陳述式中，因為它無法與加密內容條件搭配使用。
- `kms:GenerateDataKey` – 產生資料加密金鑰，以在儲存前加密註釋。此操作包含範圍存取控制的加密內容條件。
- `kms:Decrypt` – 解密先前加密的註釋資料。對於 IAM 角色，這包含 `kms:ViaService`條件。對於服務主體，這包含加密內容條件。

`aws:SourceAccount` 和 `aws:SourceArn`條件索引鍵提供主要保護，防止混淆代理人攻擊。加密內容條件提供額外的驗證層。如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的使用 aws:SourceArn或 aws:SourceAccount條件金鑰。

> **Important**
>
> 勾點執行角色不需要 AWS KMS 許可。CloudFormation Hooks 服務主體會執行所有 AWS KMS 操作。

`SetTypeConfiguration` API 的 KMS 許可

在 SetTypeConfiguration API 呼叫期間，CloudFormation 會驗證使用者使用指定 AWS KMS 金鑰加密註釋資料的許可。將下列 IAM 政策新增至將使用 `SetTypeConfiguration` API 設定加密的使用者或角色。使用您的資訊取代`######`。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudformation:SetTypeConfiguration",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "kms:DescribeKey",
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/abc-123"
    },
    {
      "Effect": "Allow",
      "Action": "kms:GenerateDataKey",
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/abc-123",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:cloudformation:hooks:service":
 "hooks.cloudformation.amazonaws.com",
          "kms:EncryptionContext:aws:cloudformation:123456789012": "123456789012"
        },
        "ArnLike": {
          "kms:EncryptionContext:aws:cloudformation:arn":
 "arn:aws:cloudformation:*:123456789012:hook/*"
        }
      }
    }
  ]
}
```

`GetHookResult` API 的 KMS 許可

若要呼叫使用客戶受管金鑰的 GetHookResult for Hooks，使用者必須擁有該金鑰的`kms:Decrypt`許可。將下列 IAM 政策新增至將呼叫 的使用者或角色`GetHookResult`。*`arn:aws:kms:us-east-1:123456789012:key/abc-123`* 使用客戶受管金鑰的 ARN 取代 。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudformation:GetHookResult",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/abc-123"
    }
  ]
}
```

CloudFormation Hooks 使用者指南的文件歷史記錄

下表說明自上次發行 CloudFormation Hooks 以來文件的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- 文件最近更新時間：2025 年 9 月 4 日。

變更	描述	日期
詳細合規檢查結果	勾點現在支援註釋，提供每個評估資源的詳細合規檢查結果和修復指導。透過 CloudFormation 主控台或 CLI `get-hook-result` 命令檢視這些詳細驗證結果。	2025 年 11 月 13 日
主動控制做為勾點	您現在可以使用 `activate-type` 和 `set-type-configuration` 命令，透過 CloudFormation 主控台或 CLI 啟用主動控制型勾點。您可以設定這些勾點來套用特定的 AWS Control Tower Control Catalog 主動控制，以在 CREATE和 UPDATE操作期間評估資源。	2025 年 9 月 4 日
勾點調用摘要	您現在可以透過 CloudFormation 主控台擷取有關勾點調用的資訊，或使用 `list-hook-results` CLI 命令以程式設計方式擷取調用詳細資訊。您現在可以依勾點或調用狀態篩選`list-hook-results` 結果，以專注於相關調用。	2025 年 9 月 4 日

堆疊層級勾點	掛鉤現在在堆疊層級受到支援，可讓客戶使用 CloudFormation Hooks 來評估新範本，並可能封鎖堆疊操作以繼續。	2024 年 11 月 13 日
AWS 雲端控制 API 勾點整合	勾點現在已與 Cloud Control API 整合，可讓客戶使用 CloudFormation Hooks 在佈建之前主動檢查資源的組態。如果找到不合規的資源，勾點會失敗操作並防止佈建資源，或發出警告並允許佈建操作繼續。	2024 年 11 月 13 日
AWS CloudFormation Guard 勾點	AWS CloudFormation Guard 是一種開放原始碼和一般用途網域特定語言 (DSL)，可用來撰寫policy-as-code。Guard Hooks 可以評估 Cloud Control API 和 CloudFormation 操作，以在佈建之前檢查資源的組態。如果找到不合規的資源，勾點會失敗操作並防止佈建資源，或發出警告並允許佈建操作繼續。	2024 年 11 月 13 日
AWS Lambda 勾點	AWS CloudFormation Lambda Hooks 可讓您針對自訂自訂程式碼評估 CloudFormation 和 Cloud Control API 操作。您的勾點可以封鎖操作以繼續，或向發起人發出警告，並允許操作繼續。	2024 年 11 月 13 日

Hooks 使用者指南	CloudFormation Hooks 使用者指南的初始版本。更新包括新的簡介、入門演練、概念和術語、堆疊層級篩選，以及有關先決條件、設定和 CloudFormation Hooks 開發的更新主題。	2023 年 12 月 8 日

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。