開發人員指南

Amazon Lookout for Vision

Amazon Lookout for Vision: 開發人員指南

Table of Contents

Amazon Lookout out in Vision sion sion sion sion sion sion sion

您可以使用 Amazon Lookout for Vision 視覺，準確且大規模地尋找工業產品中的視覺缺陷。它使用電腦視覺來識別工業產品中遺失的元件、車輛或結構的損壞、生產線上的不規則性，甚至是矽晶圓中的微小缺陷 — 或任何其他品質很重要的實體項目，例如印刷電路板上的電容器遺失。

主要優點

Amazon Lookout for Vision

- 快速有效地改善流程 — 您可以使用 Amazon Lookout 視覺版，在工業流程中快速有效地大規模地實作以電腦視覺為基礎的檢測。您可以提供最少 30 個基準良好的圖像，Lookout to Vision 可以自動構建用於缺陷檢測的自定義 ML 模型。然後，您可以批量或實時處理來自 IP 攝影機的影像，以快速準確地識別凹痕、裂縫和刮痕等異常情況。
- 快速提高生產品質 — 使用 Amazon Lookout for Vision 功能，您可以即時減少生產流程中的缺陷。它可識別並報告儀表板中的視覺異常，以便您可以快速採取行動來阻止更多缺陷發生，從而提高生產品質並降低成本。
- 降低營運成本 — Amazon Lookout for Vision 報告視覺檢測資料中的趨勢，例如識別缺陷率最高的程序，或標示最近的瑕疵變化。使用此資訊，您可以決定是否要在處理生產線上排程維護，或是在發生昂貴、意外的停機時間之前將生產線重新路由到另一台機器。

您第一次使用 Lookout for Vision

若是 Lookout in Vision sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion sion

1. 建立 Amazon Lookout for Vision— 在本節中，您可以設置您的帳戶詳細信息。
2. Amazon Lookout for Vision— 在本節中，您將了解如何創建第一個亞馬遜 Lookout for Vision 模型。

建立 Amazon Lookout for Vision

在本節中，您註冊一個 AWS 帳戶並設置 Amazon Lookout for Vision。

如需支援 Amazon Lookout for Vision 景的 AWS 區域的相關資訊，請參閱 Amazon Lookout for Vision 端點和配額。

主題

步驟 1：創建一個 AWS 帳戶

在此步驟中，您會註冊 AWS 帳戶並建立系統管理使用者。

主題

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

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

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

建立具有管理權限的使用者

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

保護您的 AWS 帳戶根使用者

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

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

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

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。
2. 在 IAM 身分中心中，將管理存取權授予使用者。

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

以具有管理權限的使用者身分登入

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

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

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

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

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

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

步驟 2：設定權限

若要使用 Amazon Lookout for Vision，您需要存取用於模型訓練的瞭望視覺主控台、 AWS SDK 操作和 Amazon S3 儲存貯體的許可。

> **Note**
>
> 如果您僅使用 AWS SDK 操作，則可以使用範圍為 AWS SDK 操作的策略。如需詳細資訊，請參閱 設定 SDK 權限。

主題

- 使用 AWS 受管政策設定主控台存取
- 設置 Amazon S3 存儲桶許可
- 指派權限

使用 AWS 受管政策設定主控台存取

使用下列 AWS 受管政策，為視覺版 Amazon Lookout 主控台和 SDK 操作套用適當的存取許可。

- AmazonLookoutVisionConsoleFullAccess— 允許完全訪問 Amazon Lookout for Vision 控制台和 SDK 操作. 您需要建立主控台值區的`AmazonLookoutVisionConsoleFullAccess`權限。如需詳細資訊，請參閱 步驟 3：建立主控台儲存貯體。
- AmazonLookoutVisionConsoleReadOnlyAccess— 允許只讀訪問 Amazon Lookout for Vision 控制台和 SDK 操作。

如要指派權限,請參閱 指派權限。

如需 AWS 受管政策的相關資訊，請參閱 AWS 受管政策。

設置 Amazon S3 存儲桶許可

Amazon Lookout for Vision 使用 Amazon S3 存儲桶來存儲以下文件：

- 資料集影像 — 用來訓練模型的影像。如需詳細資訊，請參閱 建立資料集。
- Amazon SageMaker Ground Truth 格式清單文件。例如， SageMaker GroundTruth 工作的資訊清單檔案輸出。如需詳細資訊，請參閱 使用 Amazon SageMaker Ground Truth 清單文件創建數據集。
- 模型訓練的輸出。

如果您使用主控台，Lookout for Vision 會建立 Amazon S3 儲存貯體 (主控台儲存貯體) 來管理您的專案。`LookoutVisionConsoleReadOnlyAccess`和受`LookoutVisionConsoleFullAccess`管政策包括主控台儲存貯體的 Amazon S3 存取許可。

您可以使用控制台存儲桶來存儲數據集映像和 SageMaker Ground Truth 格式清單文件。或者，您可以使用不同的 Amazon S3 儲存貯體。儲存貯體必須由您的 AWS 帳戶擁有，且必須位於您正在使用 Lookout for Vision AWS 區域。

若要使用不同的值區，請將下列政策新增至所需的使用者或群組。`my-bucket`以所需儲存貯體的名稱取代。如需新增 IAM 政策的相關資訊，請參閱建立 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        },
        {
            "Sid": "LookoutVisionS3ObjectAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
```

```
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ]
        }
    ]
}
```

如要指派權限,請參閱 指派權限。

指派權限

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南 的 新增權限到使用者 (主控台) 中的指示。

步驟 3：建立主控台儲存貯體

要使用亞馬遜 Lookout for Vision 控制台，您需要一個稱為控制台存儲桶的 Amazon S3 存儲桶。控制台存儲桶存儲以下內容：

- 您使用主控台上傳至資料集的影像。
- 您從主控台開始的模型訓練的訓練結果。
- 試驗檢測結果。
- 當您使用主控台在 S3 儲存貯體中自動標記映像來建立資料集時，主控台建立的暫時資訊清單檔案。控制台不會刪除清單文件。

當您第一次在新 AWS 區域中開啟 Amazon Lookout for Vision 主控台時，Lookout for Vision 會代表您建立主控台儲存貯體。請記下主控台值區名稱，因為您可能需要在 AWS SDK 作業或主控台工作 (例如建立資料集) 中使用值區名稱。

或者，您也可以使用 Amazon S3 建立主控台儲存貯體。如果 Amazon S3 儲存貯體政策無法讓 Amazon Lookout for Vision 主控台成功建立主控台儲存貯體，請使用此方法。例如，禁止自動建立 Amazon S3 儲存貯體的政策。

> **Note**
>
> 如果您只使用 AWS SDK 而不是 Lookout for Vision 控制台，則不需要創建控制台存儲桶。您可以使用具有您選擇的名稱的不同 S3 儲存貯體。

控制台存儲桶名稱的格式為 `lookoutvision-` <region>`-` <random value>。隨機值可確保儲存貯體名稱之間不會發生衝突。

主題

使用 Amazon Lookout for Vision 控制台創建控制台存儲桶

使用下列程序，使用 Amazon Lookout for Vision 察主控台為 AWS 區域建立主控台儲存貯體。如需我們啟用的 S3 儲存貯體設定的相關資訊，請參閱主控台值區設定。

使用 Amazon Lookout for Vision 控制台創建控制台存儲桶

1. 確定您正在使用的使用者或群組具有`AmazonLookoutVisionConsoleFullAccess`權限。如需詳細資訊，請參閱 步驟 2：設定權限。
2. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
3. 在導覽列上，選擇 [選取區域]。然後選擇您要為其建立主控台值 AWS 區的「區域」。
4. 選擇 開始使用。
5. 如果這是您第一次在目前 AWS 區域開啟主控台，請在「首次設定」對話方塊中執行以下操作：

   a. 將顯示的 Amazon S3 儲存貯體名稱複製下來。稍後您將需要此資訊。

b. 選擇建立 S3 儲存貯體，讓 Amazon Lookout for Vision 代表您建立主控台儲存貯體。

如果目前 [AWS 區域] 的主控台儲存貯體已經存在，則不會顯示 [首次設定] 對話方塊。

6. 關閉瀏覽器視窗。

使用 Amazon S3 創建控制台存儲桶

您可以使用 Amazon S3 建立主控台儲存貯體。您必須在啟用 Amazon S3 版本控制的情況下建立儲存貯體。我們建議您使用 Amazon S3 生命週期組態來移除物件的非目前 (先前) 版本，並刪除不完整的分段上傳。我們不建議使用刪除物件目前版本的生命週期組態。如需我們為您使用 Amazon Lookout 視覺主控台建立的主控台儲存貯體啟用的 S3 儲存貯體設定的相關資訊，請參閱主控台值區設定。

1. 決定您要在其中建立主控台值 AWS 區的區域。如需支援區域的相關資訊，請參閱 Amazon Lookout for Vision 端點和配額。
2. 使用建立儲存貯體中的 S3 主控台指示建立儲存貯體。請執行下列操作：

   a. 對於步驟 3，請指定前面加上的值區名稱。`lookoutvision-`*`region`*`-`*`your-identifier`*變更`region`為您在上一個步驟中選擇的區域代碼。更改`your-identifier`為您選擇的唯一標識符。例如：`lookoutvision-us-east-1-my-console-bucket-1`

   b. 對於步驟 4，選擇您要使用的 AWS 區域。

3. 按照在儲存貯體上啟用版本控制中的 S3 主控台指示啟用儲存貯體的版本控制。
4. (選擇性) 依照在儲存貯體上設定生命週期組態中的 S3 主控台指示，指定儲存貯體的生命週期組態。執行下列動作以移除物件的非目前 (先前) 版本，並刪除不完整的分段上傳。您不需要執行步驟 6、8、9、10。

   a. 在步驟 5 中，選擇「套用至值區中的所有物件」。

   b. 在步驟 7 中，選取 [永久刪除物件的非目前版本] 和 [刪除過期物件刪除標記] 或 [不完整的分段上傳]。

   c. 在步驟 11 中，請輸入刪除物件的封存版本之前要等待的天數。

   d. 在步驟 12 中，請輸入刪除不完整的分段上傳之前的等待天數。

主控台值區設定

如果您使用 Amazon Lookout for Vision 主控台建立主控台儲存貯體，我們會在主控台儲存貯體上啟用以下設定。

- 主控台值區中物件的版本化。
- 主控台值區中物件的伺服器端加密。
- 刪除非目前物件 (30 天) 與不完整分段上傳 (3 天) 的生命週期組態。
- 封鎖主控台值區的公用存取權。

步驟 4：設定 AWS CLI 和 AWS 軟體開發套件

以下步驟說明如何安裝 AWS Command Line Interface (AWS CLI) 和 AWS SDK。本文件中的範例使用 AWS CLI Python 和 Java AWS 軟體開發套件。

主題

- 安裝 AWS 軟體開發套件
- 授與程式設計存取權
- 設定 SDK 權限
- 致電 Amazon Lookout for Vision 操作

安裝 AWS 軟體開發套件

請按照以下步驟下載和配置 AWS SDK。

若要設定 AWS CLI 和 AWS 軟體開發套件

- 下載AWS CLI並安裝您要使用的和 AWS SDK。本指南提供了有關 AWS CLI、Java 和 Python 的範例。如需安裝 AWS 開發套件的相關資訊，請參閱 Amazon Web Services 的工具。

授與程式設計存取權

您可以在本機電腦或其他 AWS 環境 (例如 Amazon 彈性運算雲端執行個體) 上執行本指南中的 AWS CLI 和程式碼範例。若要執行範例，您必須授與範例所使用之 AWS SDK 作業的存取權。

主題

- 在本機電腦執行程式碼
- 在 AWS 環境中執行程式碼

在本機電腦執行程式碼

若要在本機電腦上執行程式碼，建議您使用短期認證授與使用者存取 AWS SDK 作業。如需有關在本機電腦上執行 AWS CLI 和程式碼範例的特定資訊，請參閱在本機電腦上使用設定檔。

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

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

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

哪個使用者需要程式設計存取權？	到	By
		• 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

在本機電腦上使用設定檔

您可以使用您在中建立的短期認證來執行本指南中的 AWS CLI 和程式碼範例在本機電腦執行程式碼。若要取得認證和其他設定資訊，範例會使用名為 `lookoutvision-access` 的設定檔，例如：

```
session = boto3.Session(profile_name='lookoutvision-access')
lookoutvision_client = session.client("lookoutvision")
```

設定檔所代表的使用者必須具有呼叫偵 Lookout for Vision SDK 作業的權限，以及範例所需的其他 AWS SDK 作業。如需詳細資訊，請參閱 設定 SDK 權限。如要指派權限，請參閱 指派權限。

若要建立可搭配 AWS CLI 和程式碼範例使用的描述檔，請選擇下列其中一項。請確定您建立的設定檔名稱是 `lookoutvision-access`。

- 由 IAM 管理的使用者 — 請按照 切換到 IAM 角色 (AWS CLI) 中的指示進行操作。
- 員工身分 (由管理的使用者 AWS IAM Identity Center) — 按照設定 AWS CLI 中的說明進行操作 AWS IAM Identity Center。對於程式碼範例，我們建議使用整合式開發環境 (IDE)，該環境支援透過 IAM Identity Center 啟用身分驗證的 AWS 工具組。如需 Java 範例，請參閱 使用 Java 開始建置。如需 Python 範例，請參閱 使用 Python 開始建置。如需更多詳細資訊，請參閱 什麼是 IAM Identity Center 憑證。

> **Note**
>
> 您可以使用程式碼取得短期憑證。如需更多相關資訊，請參閱 切換到 IAM 角色 (AWS API)。對於 IAM Identity Center，請遵循 取得 CLI 存取的 IAM 角色憑證 中的指示，獲得某個角色的短期憑證。

在 AWS 環境中執行程式碼

您不應該使用用戶憑據在 AWS 環境中簽署 AWS SDK 調用，例如在 AWS Lambda 函數中運行的生產代碼。相反地，您可以一個角色來定義您的程式碼所需的權限。然後，您可以將該角色附加到程式碼執行的環境。如何附加角色並提供臨時憑證取決於程式碼執行的環境：

- AWS Lambda 函數 — 當 Lambda 擔任 Lambda 函數的執行角色時，請使用 Lambda 自動提供給函數的臨時登入資料。這些憑證可在 Lambda 環境變數中使用。您不需要指定設定檔。如需更多詳細資訊，請參閱 Lambda 執行角色。
- Amazon EC2 — 使用 Amazon EC2 執行個體中繼資料端點憑證提供者。提供者會使用您附加到 Amazon EC2 執行個體的 Amazon EC2 執行個體設定檔，為您自動產生和重新整理憑證。如需更多詳細資訊，請參閱 使用 IAM 角色向在 Amazon EC2 執行個體上執行的應用程式授予權限。
- Amazon Elastic Container Service — 使用容器憑證提供者。Amazon ECS 會將憑證傳送並重新整理到中繼資料端點。您指定的 任務 IAM 角色 提供了用於管理應用程式使用的憑證的策略。如需更多詳細資訊，請參閱 與 AWS 服務互動。
- 核心裝置 — 使用 X.509 憑證，使用 TLS 相互驗證通訊協定連線至 AWS IoT 核心。這些憑證讓裝置在沒有 AWS 登入資料的情況下與 AWS IoT 互動。AWS IoT 登入資料供應商會使用 X.509 憑證對裝置進行驗證，並以臨時、有限權限的安全權杖形式核發 AWS 登入資料。如需更多詳細資訊，請參閱 與 AWS 服務互動。

如需憑證提供者的更多詳細資訊，請參閱 標準化憑證提供者。

設定 SDK 權限

若要使用 Amazon Lookout for Vision SDK 操作，您需要存取用於模型訓練的瞭望視覺 API 和 Amazon S3 儲存貯體的許可。

主題

- 授予 SDK 操作權限
- 授予 Amazon S3 存儲桶許可

- 指派權限

授予 SDK 操作權限

我們建議您只授與執行任務所需的權限 (最低權限許可)。例如，要打電話 DetectAnomalies，您需要執行許可`lookoutvision:DetectAnomalies`。若要尋找操作的權限，請檢查 API 參考。

當您剛開始使用應用程式時，您可能不知道所需的特定權限，因此您可以從更廣泛的權限開始。 AWS 託管策略提供權限來幫助您入門。

- AmazonLookoutVisionFullAccess— 允許完全訪問 Amazon Lookout for Vision SDK 操作.
- AmazonLookoutVisionReadOnlyAccess— 允許存取唯讀 SDK 作業。

主控台的受管理原則也提供 SDK 作業的存取權限。如需詳細資訊，請參閱 步驟 2：設定權限。

如需 AWS 受管政策的相關資訊，請參閱 AWS 受管政策。

當您知道應用程式所需的權限時，可以透過定義特定於您的用例的客戶管理政策來進一步減少權限。如需更多詳細資訊，請參閱 客戶管理政策。

> **Note**
>
> 入門指示需要`s3:PutObject`權限。如需詳細資訊，請參閱 步驟 1：建立清單檔案並上傳圖片。

如要指派權限，請參閱 指派權限。

授予 Amazon S3 存儲桶許可

若要訓練模型，您需要具有適當許可的 Amazon S3 儲存貯體來存放映像、資訊清單檔案和訓練輸出。儲存貯體必須由您的 AWS 帳戶擁有，且必須位於您正在使用 Amazon 遠景 Lookout for Vision AWS 區域。

僅限 SDK 的受管政策 (`AmazonLookoutVisionFullAccess`和`AmazonLookoutVisionReadOnlyAccess`) 不包含 Amazon S3 儲存貯體許可，您需要套用下列權限政策才能存取您使用的儲存貯體，包括現有的主控台儲存貯體。

主控台受管理的策略（AmazonLookoutVisionConsoleFullAccess和AmazonLookoutVisionConsoleReadOnlyAccess）包括控制台值區的存取權限。如果您透過 SDK 作業存取主控台值區，且具有主控台受管理原則權限，則不需要使用下列原則。如需詳細資訊，請參閱 步驟 2：設定權限。

決定工作權限

使用下列資訊來決定您要執行的工作需要哪些權限。

建立資料集

若要使用建立資料集 CreateDataset，您需要下列權限。

- s3:GetBucketLocation-允許 Lookout for Vision 驗證您的存儲桶是否位於您使用 Lookout for Vision 同一區域。
- s3:GetObject— 允許存取DatasetSource輸入參數中指定的資訊清單檔案。如果您想要指定資訊清單檔案的確切 S3 物件版本，您還需要s3:GetObjectVersion在資訊清單檔案上。如需詳細資訊，請參閱在 S3 儲存貯體中使用版本控制

建立新模型

若要使用建立模型 CreateModel，您需要下列權限。

- s3:GetBucketLocation-允許 Lookout for Vision 驗證您的存儲桶是否位於您使用 Lookout for Vision 同一區域。
- s3:GetObject— 允許存取專案訓練和測試資料集中指定的影像。
- s3:PutObject— 允許將訓練輸出儲存在指定值區中的權限。您可以在OutputConfig參數中指定輸出值區位置。或者，您可以將權限範圍縮小到僅在S3Location輸入Prefix欄位中指定的物件索引鍵。如需詳細資訊，請參閱OutputConfig。

存取影像、資訊清單檔案和訓練輸出

查看 Amazon Lookout for Vision 操作響應不需要 Amazon S3 存儲桶許可。如果您想要存取作業回應中參照的影像、資訊清單檔案和訓練輸出，則需要s3:GetObject權限。如果您要存取版本控制的 Amazon S3 物件，則需要s3:GetObjectVersion許可。

設置 Amazon S3 存儲桶政策

您可以使用下列政策指定建立資料集 (CreateDataset)、建立模型 (CreateModel) 以及存取映像、資訊清單檔案和訓練輸出所需的 Amazon S3 儲存貯體許可。將 *my-bucket* 的值變更為您要使用的值區的名稱。

您可以根據自己的需求調整政策。如需詳細資訊，請參閱 決定工作權限。將策略新增至所需的使用者。如需詳細資訊，請參閱建立 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccess",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        },
        {
            "Sid": "LookoutVisionS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
```

```
}
```

如要指派權限,請參閱 指派權限。

指派權限

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

致電 Amazon Lookout for Vision 操作

執行下列程式碼，以確認您可以呼叫 Amazon Lookout for Vision API。此代碼會在目前 AWS 區域中列出您 AWS 帳戶中的專案。如果您之前尚未建立專案，則回應為空白，但確認您可以呼叫該`ListProjects`操作。

一般而言，呼叫範例函數需要 AWS SDK Lookout for Vision 戶端和任何其他必要參數。AWS 開發套件 Lookout for Vision 戶端在主要功能中宣告。

如果程式碼失敗，請檢查您使用的使用者是否擁有正確的權限。另外，請檢查您使用的 AWS 區域 Amazon Lookout for Vision 並非所有區 AWS 域都可用。

致電 Amazon Lookout for Vision 操作

1. 如果您尚未這樣做，請安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用以下的範例程式碼來檢視您的專案。

CLI

使用 `list-projects` 指令列出您帳戶中的專案。

```
aws lookoutvision list-projects \
--profile lookoutvision-access
```

Python

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0


from botocore.exceptions import ClientError
import boto3

class GettingStarted:

    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("ARN: " + project["ProjectArn"])
                print()
            print("Done!")
        except ClientError:
            raise
def main():
    session = boto3.Session(profile_name='lookoutvision-access')
    lookoutvision_client = session.client("lookoutvision")
```

```
    GettingStarted.list_projects(lookoutvision_client)

if __name__ == "__main__":
    main()
```

Java V2

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.ProjectMetadata;
import
 software.amazon.awssdk.services.lookoutvision.paginators.ListProjectsIterable;
import software.amazon.awssdk.services.lookoutvision.model.ListProjectsRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class GettingStarted {

    public static final Logger logger =
 Logger.getLogger(GettingStarted.class.getName());

    /**
     * Lists the Amazon Lookoutfor Vision projects in the current AWS account
 and
     * AWS Region.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
```

```
     * @return List<ProjectMetadata> Metadata for each project.
     */
    public static List<ProjectMetadata> listProjects(LookoutVisionClient
lfvClient)
            throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                .maxResults(100)
                .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();

        ListProjectsIterable projects =
lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                .forEach(project -> {
                    projectMetadata.add(project);
                    logger.log(Level.INFO, project.projectName());
                });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

    }

    public static void main(String[] args) throws Exception {

        try {

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

.credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                    .build();

            List<ProjectMetadata> projects = Projects.listProjects(lfvClient);

            System.out.printf("Projects%n--------%n");

            for (ProjectMetadata project : projects) {
                System.out.printf("Name: %s%n", project.projectName());
```

```
                System.out.printf("ARN: %s%n", project.projectArn());
                System.out.printf("Date: %s%n%n",
    project.creationTimestamp().toString());
            }

        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Could not list projects: {0}: {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("Could not list projects: %s",
    lfvError.getMessage()));
            System.exit(1);
        }

    }

}
```

步驟 5：(選用) 使用您自己的 AWS Key Management Service 金鑰

您可以使用 AWS Key Management Service (KMS) (KMS) 來管理存放在 Amazon S3 儲存貯體中的輸入映像的加密。

根據預設，您的映像會使用 AWS 擁有和管理的金鑰加密。您也可以選擇使用自己的 AWS Key Management Service (KMS) (KMS) 金鑰。如需更多詳細資訊，請參閱 AWS 金鑰管理服務概念。

如果您想要使用自己的 KMS 金鑰，請使用下列原則來指定 KMS 金鑰。將 *kms_key_arn* 變更為您要使用的 KMS 金鑰 (或 KMS 別名 ARN) 的 ARN。或者，指定*使用任何 KMS 金鑰。如需將政策新增至使用者或角色的相關資訊，請參閱建立 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionKmsDescribeAccess",
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "kms_key_arn"
        },
        {
            "Sid": "LookoutVisionKmsCreateGrantAccess",
```

```
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "kms_key_arn",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "lookoutvision.*.amazonaws.com"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

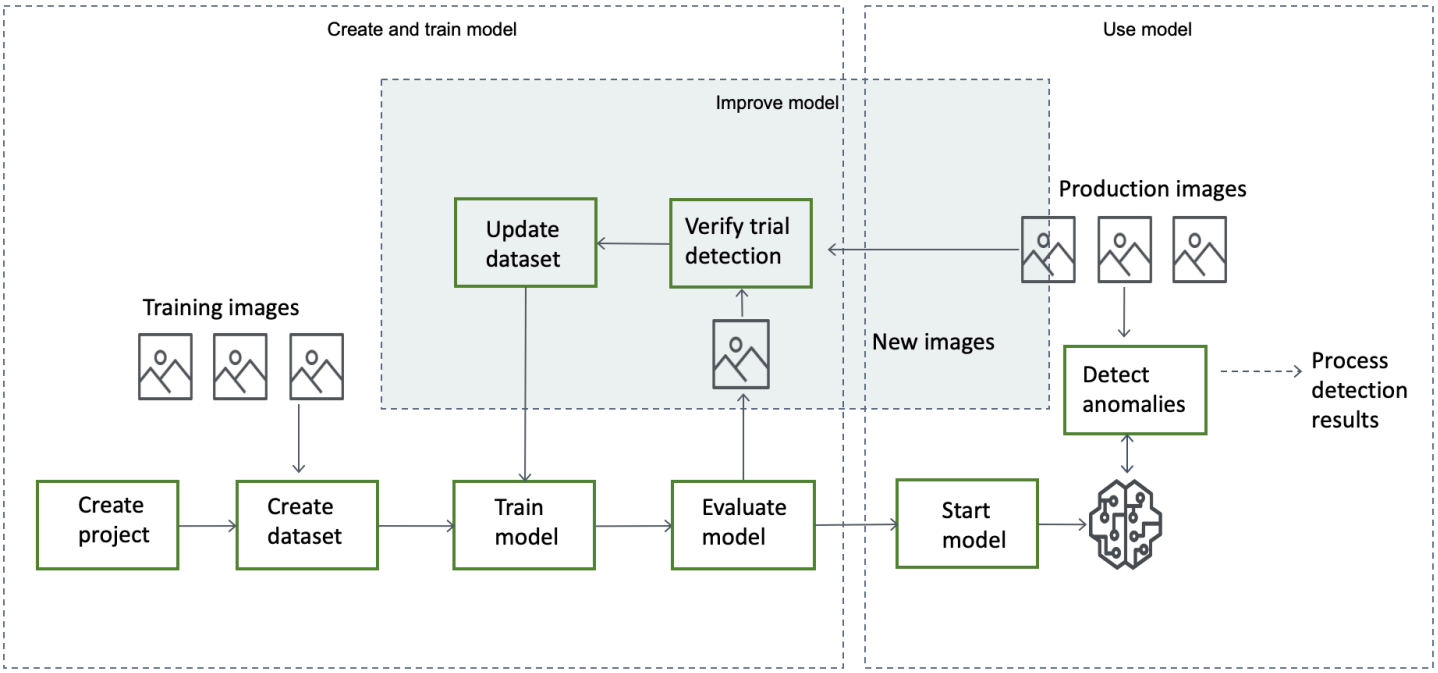
瞭解 Amazon Lookout for Vision

您可以使用 Amazon Lookout for Vision 找工業產品中的視覺瑕疵，準確且大規模地尋找工業產品中的視覺缺陷，以執行下列任務：

- 檢測損壞的零件 — 在製造和組裝過程中，發現產品表面質量，顏色和形狀的損壞。
- 識別遺失的元件 — 根據物件的存在、存在或位置來判斷遺失的元件。例如，印刷電路板上缺少的電容器。
- 揭露製程問題 — 利用重複圖案偵測瑕疵，例如矽晶圓上同一點的重複刮痕。

透過 Lookout for Vision，您可以建立一個電腦視覺模型，以預測影像中異常的存在。您提供亞馬遜 Lookout for Vision 來訓練和測試您的模型的圖像。Amazon Lookout for Vision 提供指標，您可以用來評估和改善訓練後的模型。您可以在AWS雲端託管訓練過的模型，也可以將模型部署到邊緣裝置。一個簡單的 API 操作返回您的模型做出的預測。

建立、評估和使用模型的一般工作流程如下：



主題

- 選擇您的型號
- 建立您的模型
- 評估您的模型

- 使用您的模型
- 在邊緣裝置上使用您的模型
- 使用您的儀表板

選擇您的型號

在建立模型之前，您必須先決定所要的模型類型。您可以建立兩種類型的模型：影像分類和影像分割。您可以根據您的使用案例來決定建立哪種模型類型。

圖像分類模型

如果您只需要知道影像是否包含異常，但不需要知道其位置，請建立影像分類模型。影像分類模型可預測影像是否包含異常。預測包括模型對預測準確性的信心。模型不會提供有關影像上發現任何異常位置的任何資訊。

圖像分割模型

如果您需要知道異常的位置 (例如刮痕的位置)，請建立影像分割模型。Amazon Lookout for Vision 模型使用語義分割來識別存在異常類型 (例如刮痕或遺失部分) 的影像上的像素。

> **Note**
>
> 語義分割模型定位不同類型的異常。它不會針對個別異常提供執行個體資訊。例如，如果影像包含兩個凹痕，Lookout for Vision 會傳回代表凹痕異常類型的單一實體中兩個凹痕的相關資訊。

亞馬遜 Lookout for Vision 分段模型可預測以下情況：

分類

模型會傳回已分析影像的分類 (正常/異常)，其中包括模型對預測的信賴度。分類資訊是與區段資訊分開計算的，您不應該假設它們之間的關係。

分段

模型會傳回影像遮色片，標記影像上發生異常的像素。不同類型的異常會根據指派給資料集中異常標籤的顏色進行顏色編碼。異常標籤表示異常的類型。例如，以下影像中的藍色遮色片會標記在汽車上發現的刮痕異常類型的位置。

模型會傳回遮色片中每個異常標籤的顏色代碼。該模型還返回異常標籤具有的圖像覆蓋率的百分比。

使用 LoLookout for Vision 分割模型，您可以使用各種標準來分析模型中的分析結果。例如：

- 異常位置 — 如果您需要知道異常的位置，請使用區段資訊查看涵蓋異常的遮罩。
- 異常類型 — 使用分段資訊來決定影像是否包含超過可接受數量的異常類型。
- 涵蓋範圍 — 使用分段資訊來決定異常類型是否涵蓋的範圍超過影像的可接受區域。
- 影像分類 — 如果您不需要知道異常的位置，請使用分類資訊來判斷影像是否包含異常。

如需範例程式碼，請參閱偵測影像中的異常。

決定所需的模型類型後，您可以建立專案和資料集來管理模型。使用標籤，您可以將圖像分類為正常或異常。標籤也可識別區段資訊，例如遮罩和異常類型。如何標記資料集中的影像，會決定 Lookout for Vision 您建立的模型類型。

標示影像分割模型比標示影像分類模型更為複雜。若要訓練區段模型，您必須將訓練影像分類為正常或異常。您還必須為每個異常圖像定義異常遮色片和異常類型。分類模型只需要您將訓練圖像識別為正常或異常。

建立您的模型

創建模型的步驟是創建一個項目，創建一個數據集，並培訓模型如下：

建立專案

建立專案以管理資料集和您建立的模型。專案應該用於單一使用案例，例如偵測單一機器零件類型中的異常。

您可以用儀表板取得您專案的概觀。如需詳細資訊，請參閱使用 Amazon LoLookout for Vision 儀表板。

其他資訊：建立您的專案。

建立資料集

若要訓練模型，Amazon Lookout 視覺需要適用於您的使用案例的正常物件和異常物件的影像。您可以在資料集中提供這些影像。

資料集是描述這些影像的一組影像和標籤。影像應代表可能發生異常的單一類型物件。如需詳細資訊，請參閱為資料集準備影像。

使用 Amazon Lookout for Vision 版，您可以擁有使用單一資料集的專案，或是具有獨立訓練和測試資料集的專案。我們建議您使用單一資料集專案，除非您想要更精細地控制訓練、測試和效能調整。

您可以透過匯入影像來建立資料集。視您匯入影像的方式而定，影像可能也會加上標籤。如果沒有，您可以使用控制台標記圖像。

匯入影像

如果您用 Lookout for Vision 建立資料集，您可以用下列其中一種方法匯入影像：

- 從您本機電腦匯入影像。圖像沒有標記。
- 從 S3 儲存貯體匯入映像。亞馬遜 Lookout for Vision 可以使用包含圖像的文件夾名稱對圖像進行分類。用`normal`於一般影像。用`anomaly`於異常影像。您無法自動指派區段標籤。
- 導入亞馬遜 SageMaker Ground Truth 清單文件。清單文件中的圖像被標記。您可以創建和導入自己的清單文件。如果您有很多圖像，請考慮使用「 SageMaker Ground Truth」標籤服務。然後，您可以從 Amazon SageMaker Ground Truth 工作匯入輸出資訊清單檔案。

標記影像

標籤描述資料集中的影像。標籤指定圖像是正常還是異常（分類）。標籤也會描述影像上異常的位置(分割)。

如果您的圖像沒有標籤，您可以使用控制台標記它們。

您指派給資料集中影像的標籤會決定 Lookout the Vision 所建立的模型類型：

影像分類

若要建立影像分類模型，請使用 Lookout for Vision 主控台將資料集中的影像分類為正常或異常狀況。

您也可以使用此CreateDataset作業，從包含分類資訊的資訊清單檔案建立資料集。

圖像分割

若要建立影像分割模型，請使用 Lookout for Vision 主控台將資料集中的影像分類為正常或異常狀況。您也可以為影像上的異常區域 (如果存在) 指定像素遮色片，以及個別異常遮色片的異常標籤。

您也可以使用此CreateDataset作業，從包含區段和分類資訊的資訊清單檔案建立資料集。

如果您的專案有個別的訓練和測試資料集，Lookout for Vision 會使用訓練資料集來學習和判斷模型類型。您應該以相同的方式標記測試數據集中的圖像。

其他資訊：建立資料集。

訓練您的模型

訓練會建立模型並對其進行訓練，以預測影像中異常的存在。每次訓練時都會建立模型的新版本。

在訓練開始時，Amazon Lookout for Vision 會選擇最合適的演算法來訓練您的模型。模型經過訓練，然後進行測試。在中Amazon Lookout for Vision，您訓練單一資料集專案，資料集會在內部分割，以建立訓練資料集和測試資料集。您也可以建立具有獨立訓練和測試資料集的專案。在此組態中，Amazon Lookout 視覺會使用訓練資料集來訓練您的模型，並使用測試資料集測試模型。

> **Important**
>
> 您需要為成功訓練模型所需的時間收費。

其他資訊：訓練您的模型。

評估您的模型

使用在測試期間建立的效能指標來評估模型的效能。

使用效能指標，您可以更好地瞭解訓練模型的效能，並決定是否已準備好在生產環境中使用該模型。

其他資訊：改善您的模型。

如果效能指標指出需要改善，您可以透過使用新影像執行試驗偵測工作，以新增更多訓練資料。工作完成後，您可以驗證結果，並將已驗證的映像新增至訓練資料集。或者，您也可以將新的訓練影像直接新增至資料集。接下來，您重新訓練模型並重新檢查效能指標。

其他資訊：使用試驗偵測工作來驗證模型。

使用您的模型

在您可以在AWS雲端中使用模型之前，請先使用StartModel作業啟動模型。您可以從控制台獲取模型的`StartModel` CLI 命令。

其他資訊：啟動模型。

訓練有素的 Amazon Lookout for Vision 模型可預測輸入影像是否包含正常或異常內容。如果您的模型是分割模型，則預測會包含一個異常遮色片，用來標記發現異常的像素。

要使用模型進行預測，請調用DetectAnomalies操作並從本地計算機傳遞輸入圖像。您可以獲取從控制台調用`DetectAnomalies`的 CLI 命令。

其他資訊：偵測影像中的異常。

> **Important**
>
> 您需支付模型執行時間的費用。

如果您不再使用模型，請使用此StopModel操作停止模型。您可以從主控台得到 CLI 命令。

其他資訊：停止模型。

在邊緣裝置上使用您的模型

您可以在AWS IoT Greengrass Version 2核心裝置上使用 Lookout for Vision 模型。

其他資訊：在邊緣裝置上使用 Amazon Lookout for Vision 模型。

使用您的儀表板

您可以使用儀表板來取得所有專案的概觀，以及個別專案的概觀資訊。

其他資訊：使用儀表板。

Amazon Lookout for Vision

在開始這些入門指示之前，我們建議您閱讀瞭解 Amazon Lookout for Vision。

入門指示會示範如何使用建立範例影像分割模型。如果您要建立範例影像分類模型，請參閱影像分類資料集。

如果您想快速嘗試示例模型，我們提供示例訓練圖像和蒙版圖像。我們還提供了一個 Python 腳本，用於創建圖像分割清單文件。您可以使用資訊清單檔案為專案建立資料集，而且不需要為資料集中的影像加上標籤。當您使用自己的影像建立模型時，必須為資料集中的影像加上標籤。如需詳細資訊，請參閱建立資料集。

我們提供的圖像是正常和異常的 cookie。異常的餅乾在餅乾形狀上有裂縫。您使用影像訓練的模型會預測分類 (正常或異常)，並在異常 Cookie 中尋找裂縫區域 (遮罩)，如下列範例所示。

主題

步驟 1：建立清單檔案並上傳圖片

在此程序中，您可以將 Amazon Lookout for Vision 文件存放庫複製到您的電腦。然後，您可以使用 Python (3.7 版或更高版本) 指令碼建立資訊清單檔案，並將訓練映像和遮罩映像上傳到您指定的 Amazon S3 位置。您可以使用資訊清單檔案來建立模型。稍後，您可以在本地存儲庫中使用測試映像來嘗試您的模型。

若要建立資訊清單檔案並上傳影像

1. 按照設置亞馬遜 Lookout for Vision 察中的說明設置亞馬遜 Lookout for Vision. 請務必安裝適用於 Python 的AWS軟件開發套件。
2. 在您想要使用 Lookout for Vision 的AWS區域中，建立 S3 儲存貯體。
3. 在 Amazon S3 儲存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存 `getting-started`
4. 請注意該資料夾的 Amazon S3 URI 和 Amazon S3 URI 和 Amazon S3 URI 和 Amazon S3 URI 和 Amazon Resource Name (您可以使用它們來設定權限並執行指令碼。
5. 請確定呼叫指令碼的使用者具有呼叫`s3:PutObject`作業的權限。您可以使用下列政策。若要指派權限，請參閱指派權限。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Sid": "Statement1",
  "Effect": "Allow",
  "Action": [
   "s3:PutObject"
  ],
  "Resource": [
   "arn:aws:s3::: ARN for S3 folder in step 4/*"
  ]
 }]
}
```

6. 請確定您有名為的本機設定檔，`lookoutvision-access`且設定檔使用者具有上一個步驟的權限。如需詳細資訊，請參閱在本機電腦上使用設定檔。
7. 下載壓縮文件，getting-started.zip。zip 檔案包含入門資料集和設定指令碼。
8. 解壓縮 `getting-started.zip` 檔案。
9. 在命令提示中，執行下列動作：

a. 導覽至 `getting-started` 資料夾。

b. 執行下列命令以建立資訊清單檔案，並將訓練映像和映像遮罩上傳到您在步驟 4 中記下的 Amazon S3 路徑。

```
python getting_started.py S3-URI-from-step-4
```

c. 指令碼完成時，請記下指令碼在其後顯示之`train.manifest`檔案的路徑`Create dataset using manifest file:`。路徑應類似`s3://*path to getting started folder*/manifests/train.manifest`。

步驟 2：建立模型

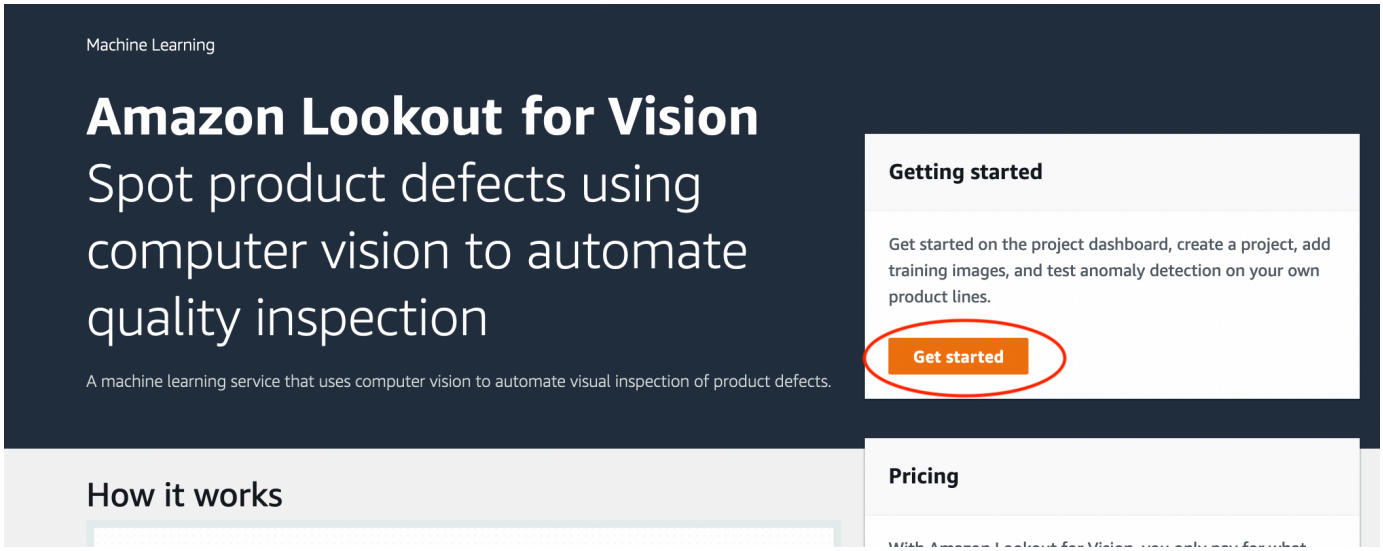
在此程序中，您可以使用先前上傳到 Amazon S3 儲存貯體的映像和資訊清單檔案建立專案和資料集。然後，您可以建立模型並檢視模型訓練中的評估結果。

由於您是從入門資訊清單檔案建立資料集，因此不需要為資料集的映像加上標籤。當您使用自己的影像建立資料集時，您需要為影像加上標籤。如需詳細資訊，請參閱標記檔案。

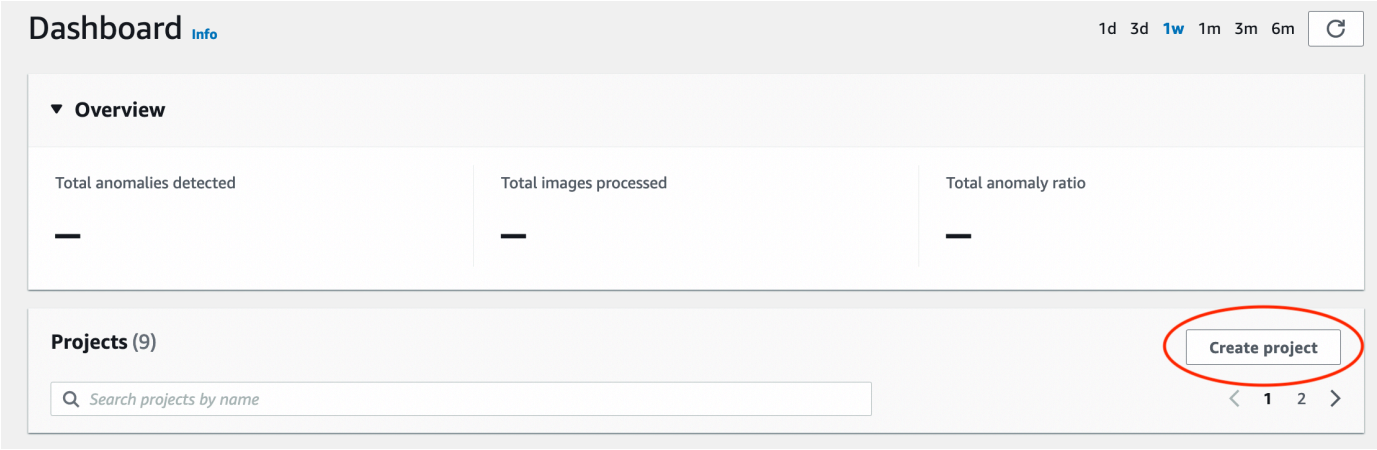
> Important
>
> 您需要支付成功訓練模型的費用。

建立模型

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 請確保您位於 Amazon S3 儲存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯存貯 AWS 步驟 1：建立清單檔案並上傳圖片 若要變更區域，請在導覽列中選擇目前顯示區域的名稱。然後選擇您要切換的區域。
3. 選擇 Get started (開始使用)。



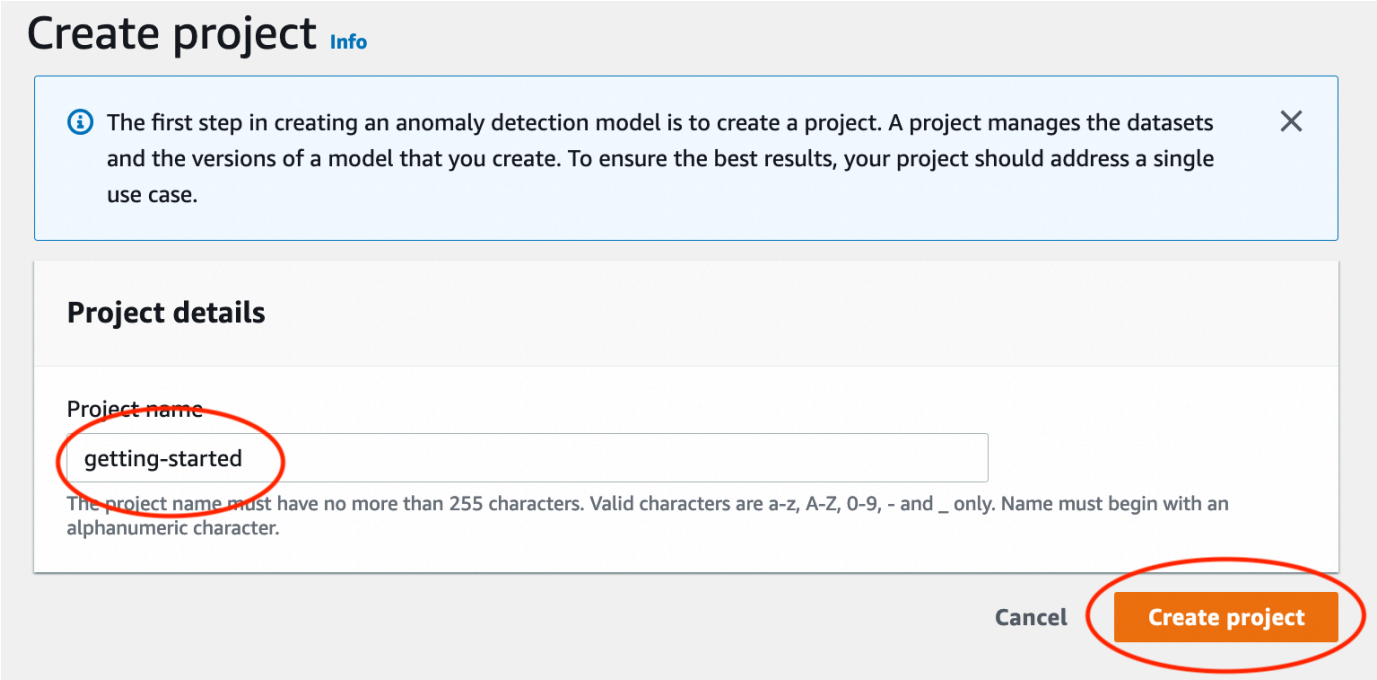
4. 在「專案」區段中，選擇「建立專案」。



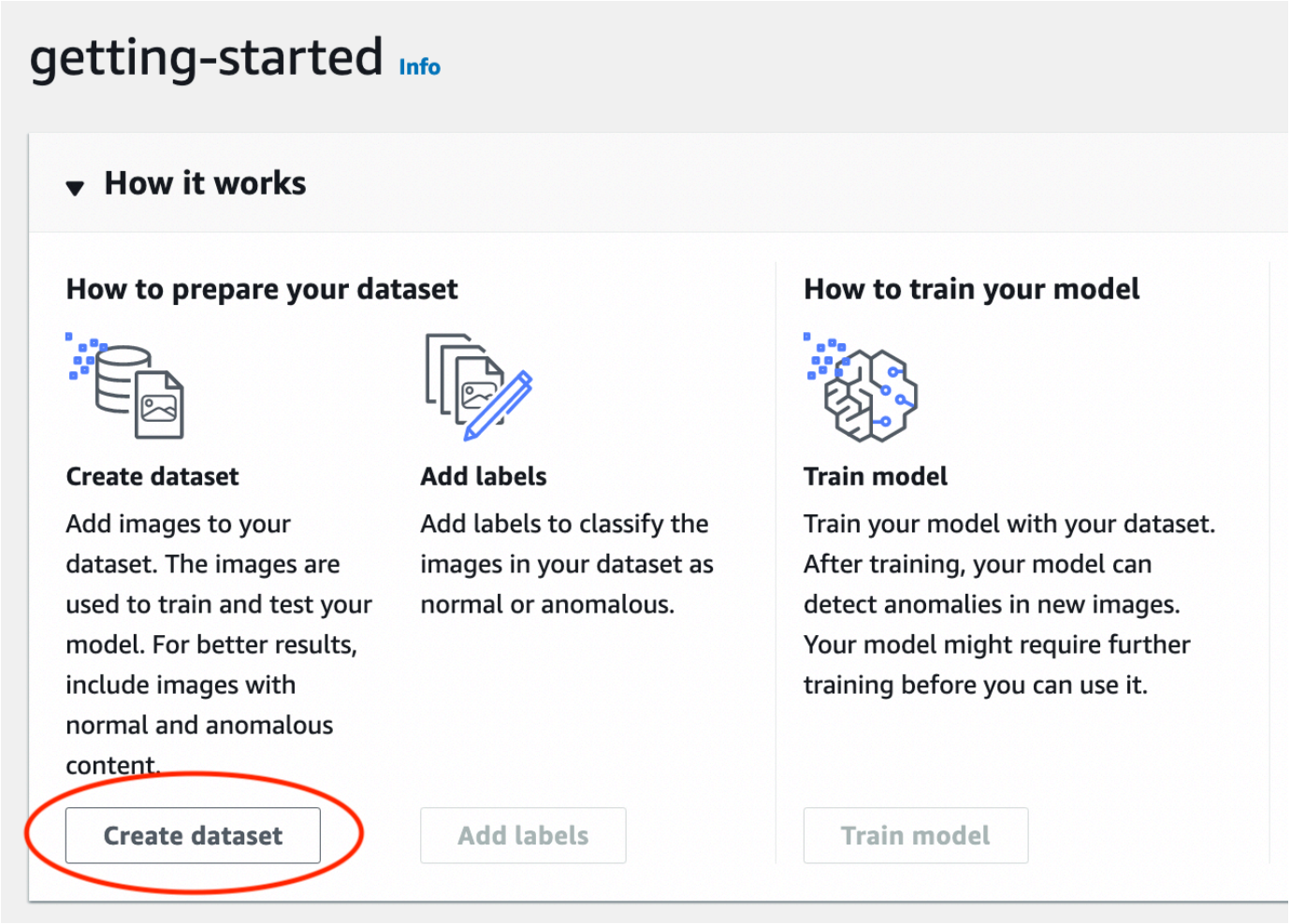
5. 在建立專案頁面上，執行下列動作：

   a. 在專案名稱中，輸入`getting-started`。

   b. 選擇 Create project (建立專案)。



6. 在專案頁面的 [運作方式] 區段中，選擇 [建立資料集]。



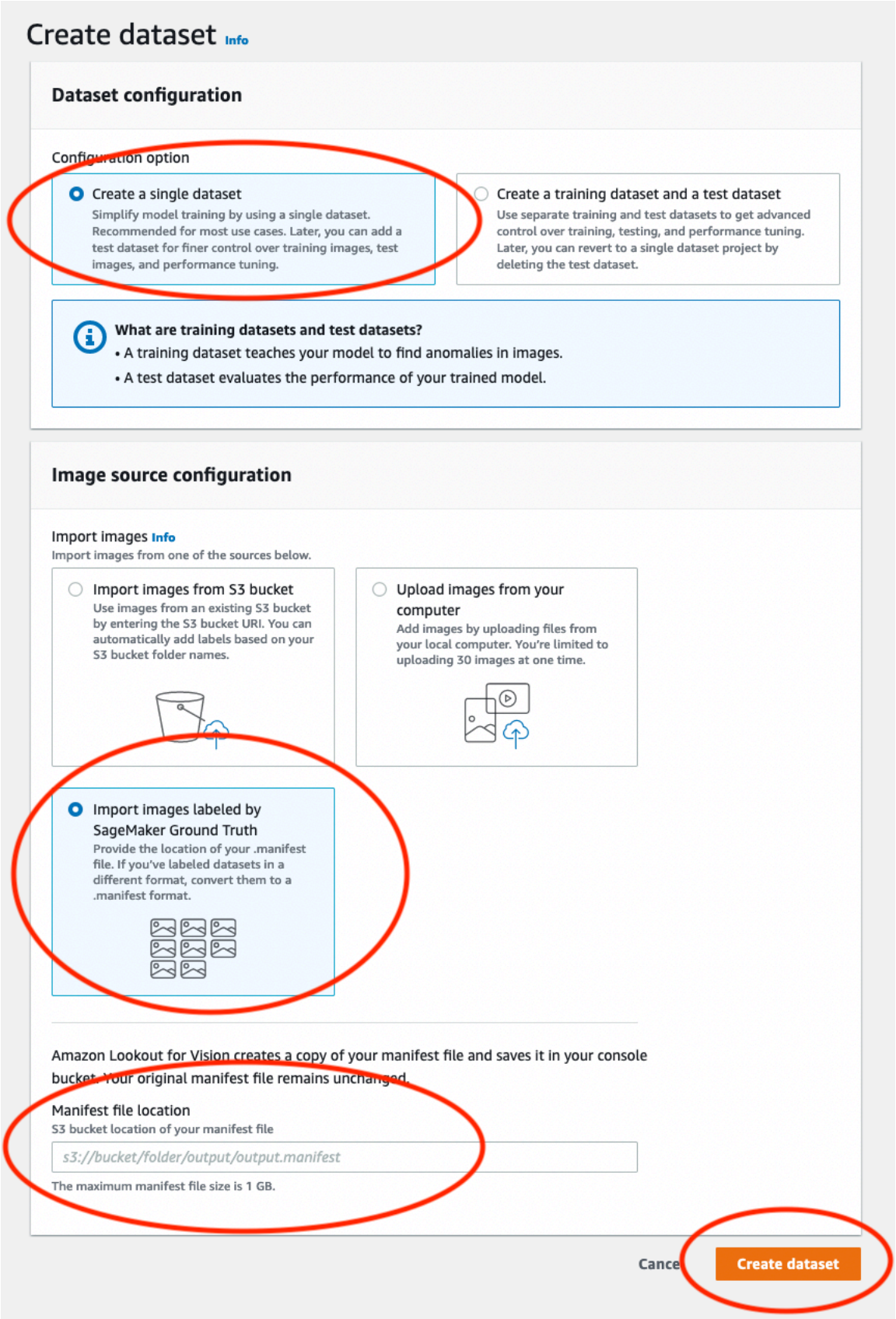
7. 在建立資料集頁面上，執行下列動作：

   a. 選擇 [建立單一資料集]。

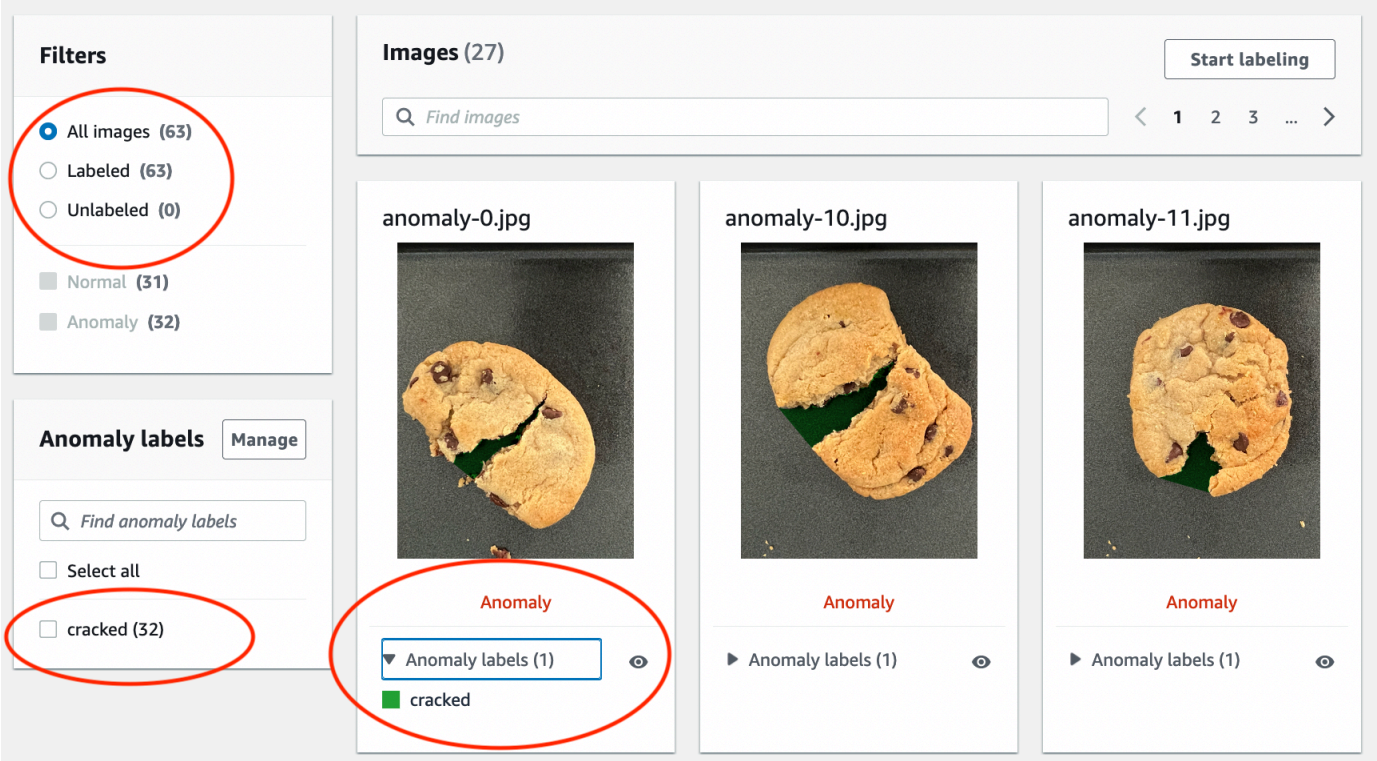
   b. 在 [影像來源設定] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像]。

   c. 對於 .manifest 檔案位置，請輸入您在步驟 6.c 中記下的資訊清單檔案的 Amazon S3 位置。步驟 1：建立清單檔案並上傳圖片Amazon S3 位置應該類似於 `s3://path to getting started folder/manifests/train.manifest`

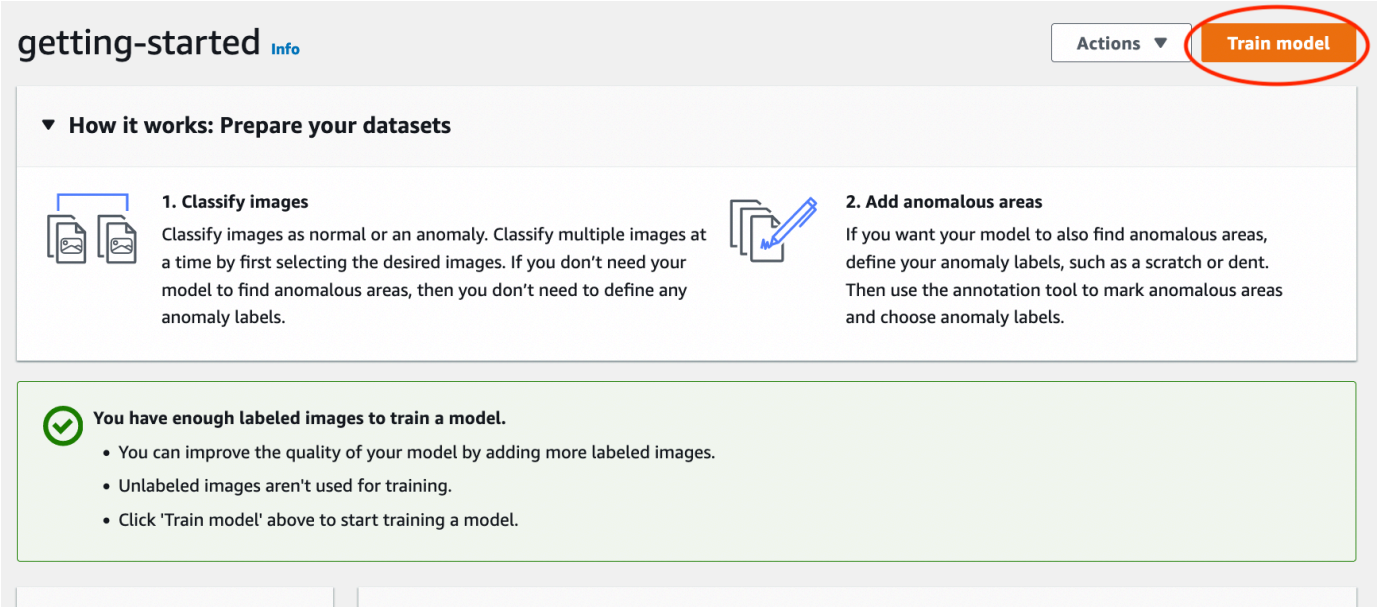
   d. 選擇建立資料集。


Create dataset Info
Dataset configuration
Configuration option
Create a single dataset
Simplify model training by using a single dataset. Recommended for most use cases. Later, you can add a test dataset for finer control over training images, test images, and performance tuning.
Create a training dataset and a test dataset
Use separate training and test datasets to get advanced control over training, testing, and performance tuning. Later, you can revert to a single dataset project by deleting the test dataset.
What are training datasets and test datasets?
• A training dataset teaches your model to find anomalies in images.
• A test dataset evaluates the performance of your trained model.
Image source configuration
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URI. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Import images labeled by SageMaker Ground Truth
Provide the location of your .manifest file. If you've labeled datasets in a different format, convert them to a .manifest format.
Amazon Lookout for Vision creates a copy of your manifest file and saves it in your console bucket. Your original manifest file remains unchanged.
Manifest file location
S3 bucket location of your manifest file
s3://bucket/folder/output/output.manifest
The maximum manifest file size is 1 GB.
Cancel
Create dataset

8. 在專案詳細資訊頁面的 [映像] 區段中，檢視資料集影像。您可以檢視每個資料集影像的分類和影像分割資訊 (遮罩和異常標籤)。您也可以搜尋影像、依標籤狀態篩選影像 (已標記/未標記)，或依指派給影像的異常標籤篩選影像。



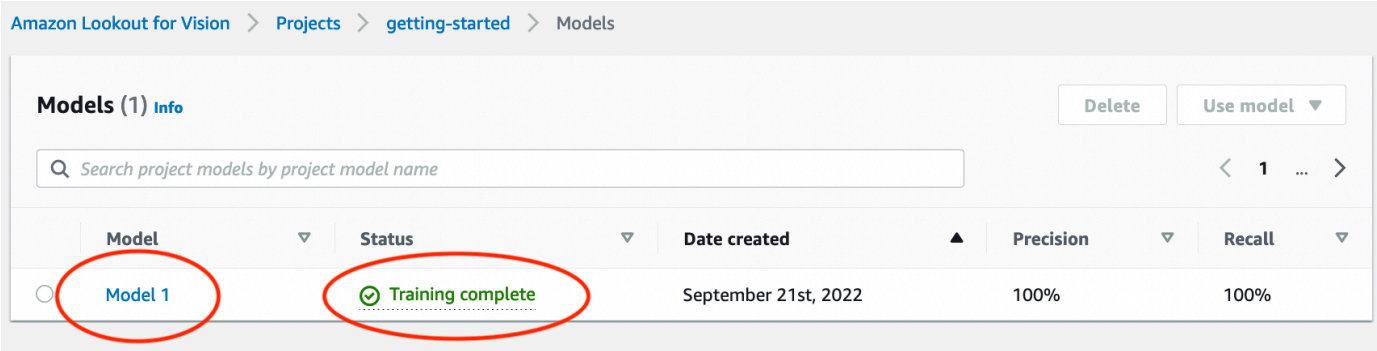
9. 在專案詳細資訊頁面上，選擇 [訓練模型]。



10. 在 [訓練模型詳細資料] 頁面上，選擇 [訓練模型]

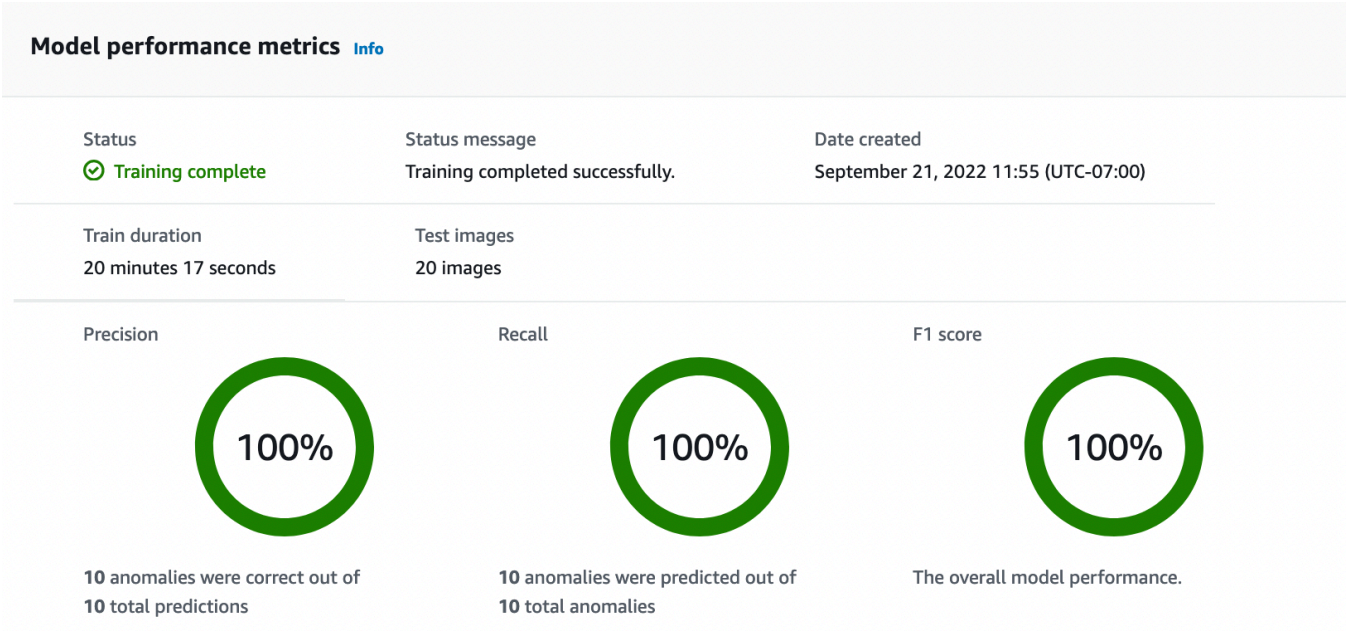
11. 在 [您要訓練您的模型嗎？ 」對話方塊中選擇「訓練模型」。

12. 在「專案模型」頁面中，您可以看到訓練已經開始。檢視模型版本的「狀態」(S tatus) 欄，以檢查目前的狀態。模型的訓練時間至少為 30 分鐘。當狀態變更為 [訓練完成] 時，訓練已成功完成。
13. 訓練結束後，請在「模型」頁面中選擇模型 1。

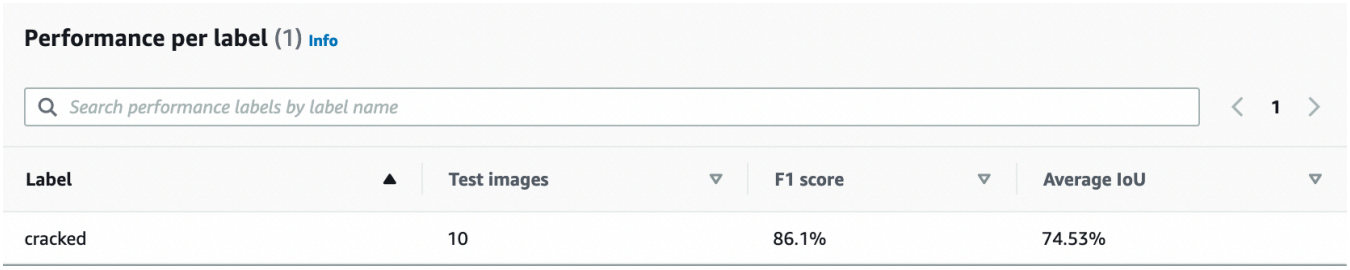


14. 在模型的詳細資訊頁面中，檢視「效能測量結果」標籤中的評估結果。有下列測量結果：

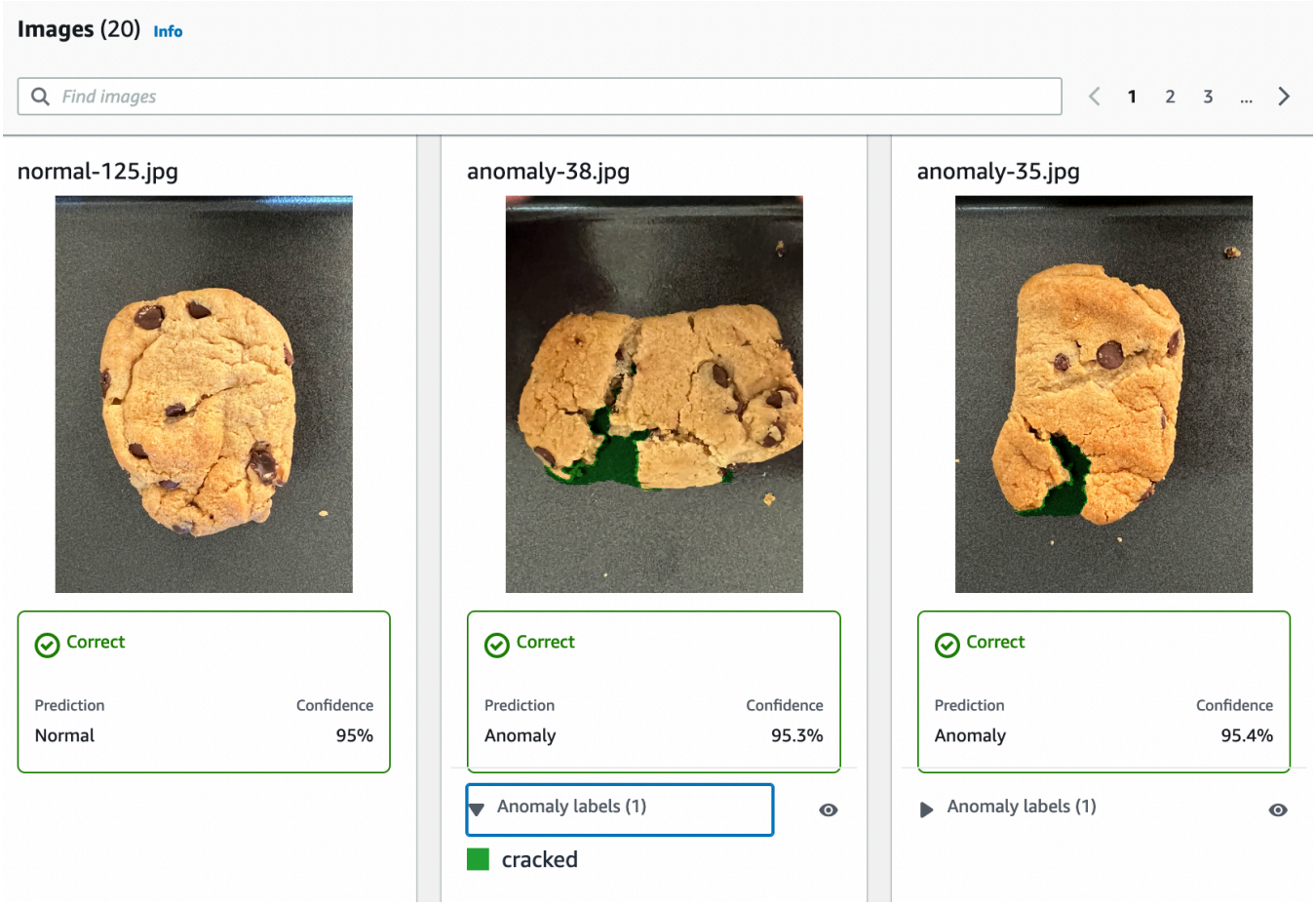
- 模型所做的分類預測的整體模型效能指標 (精確度、召回和 F1 分數)。



- 測試影像中發現異常標籤的效能指標 (平均 IoU、F1 分數)



- 測試影像的預測 (分類、分段遮罩和異常標籤)



由於模型訓練不具決定性，因此您的評估結果可能與此頁面上顯示的結果不同。如需詳細資訊，請參閱改善您的 Amazon Lookout for Vision 模型。

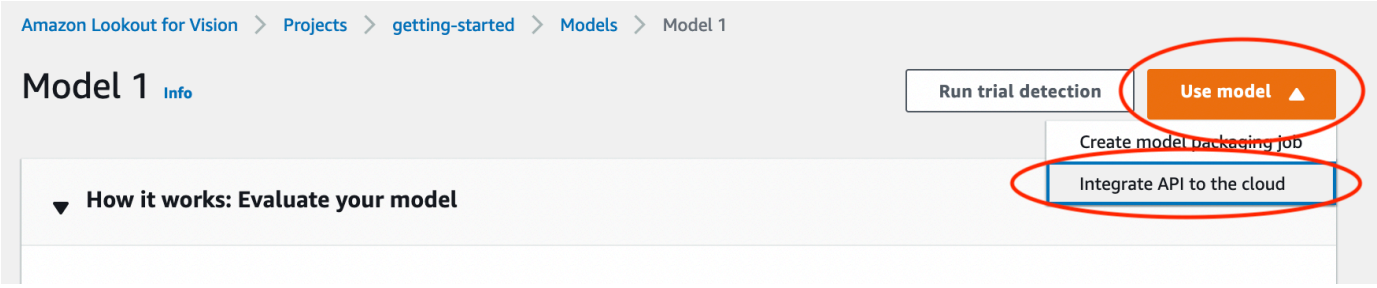
步驟 3：啟動模型

在此步驟中，您將開始主體模型，以便可以分析影像。如需詳細資訊，請參閱運行訓練有素的亞馬遜 Lookout for Vision 模型。

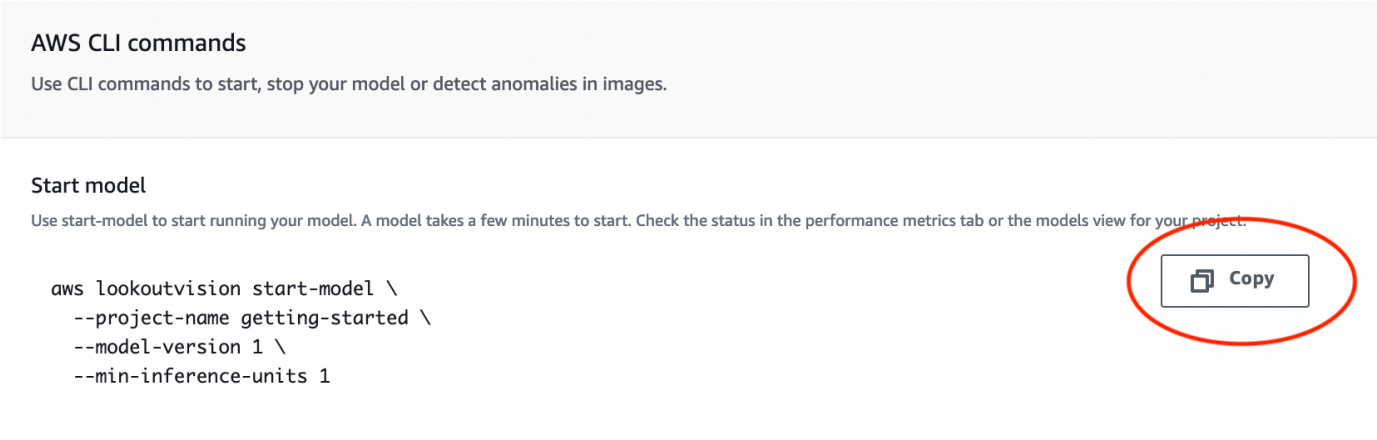
> **Note**
>
> 系統會依模型執行的時間量來對您進行測試。你停止你的模型步驟 5：停止模型。

要啟動模型。

1. 在模型的詳細資料頁面上，選擇 [使用模型]，然後選擇 [將 API 整合至雲端]。



2. 在「指AWS CLI令」區段中，複製指`start-model`AWS CLI令。



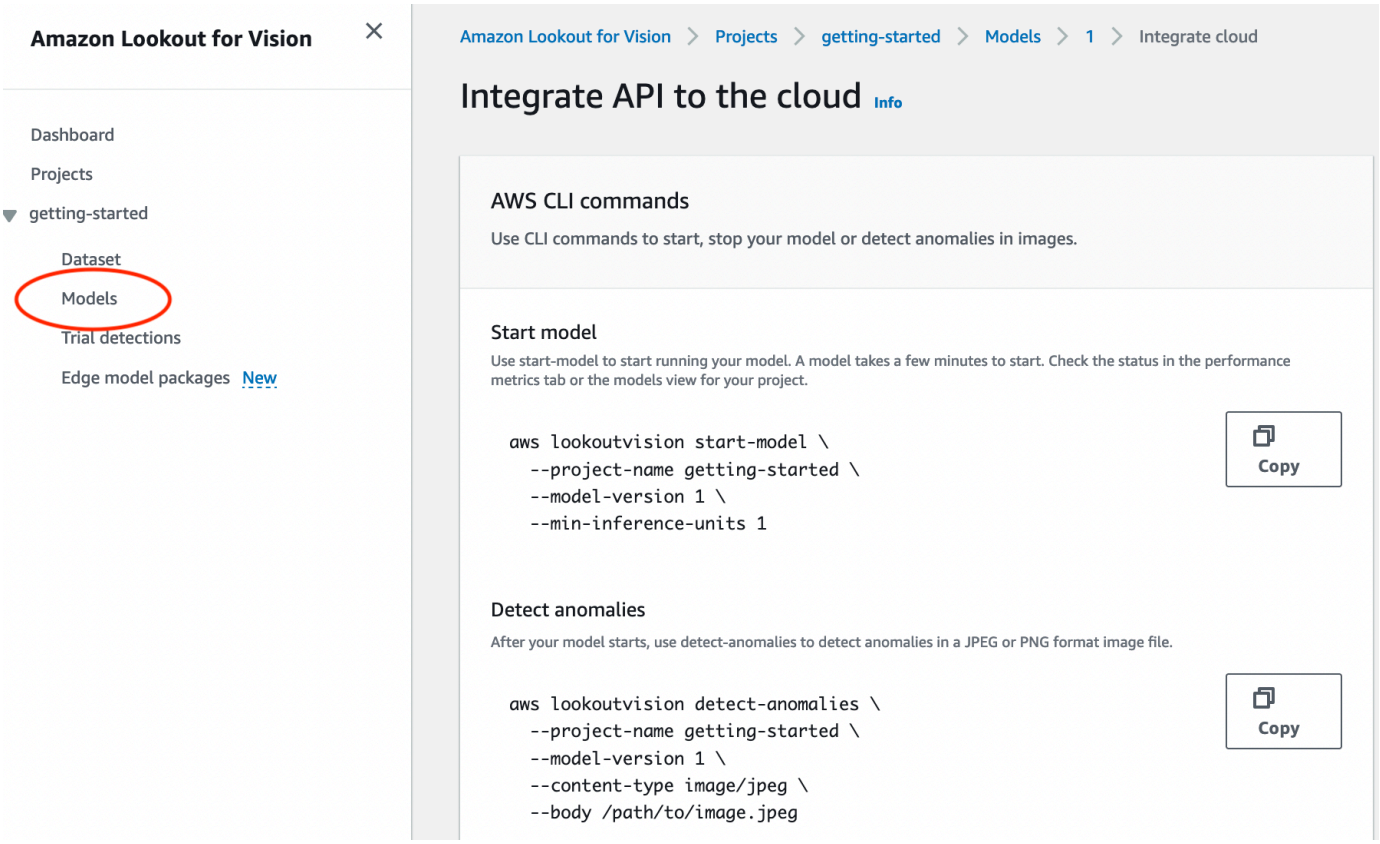
3. 請確定已設定AWS CLI為在您使用 Amazon Lookout for Vision 主控台的相同AWS區域中執行。若要變更AWS CLI使用的「AWS區域」，請參閱安裝 AWS 軟體開發套件。
4. 於指令提示下，透過輸入`start-model`指令啟動模型。如果您使用`lookoutvision`設定檔取得認證，請新增`--profile lookoutvision-access`參數。例如：

```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
  --min-inference-units 1 \
  --profile lookoutvision-access
```

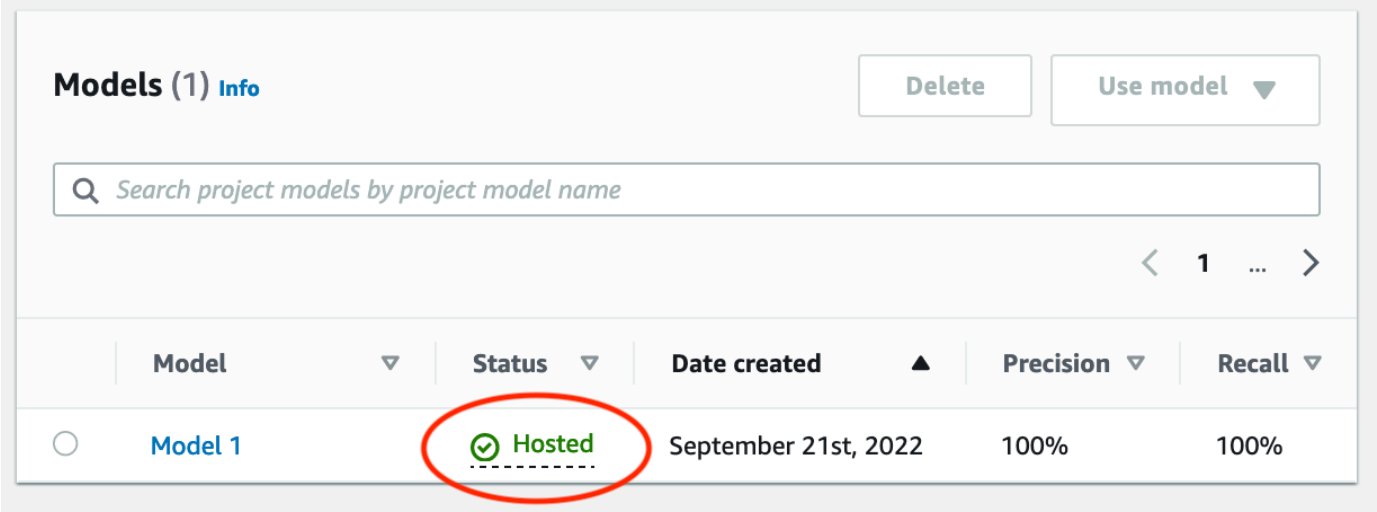
如果呼叫成功，則會顯示下列輸出：

```
{
    "Status": "STARTING_HOSTING"
}
```

5. 回到主控台中，選擇導覽窗格中的模型。



6. 等待「狀態」欄中模型 (模型 1) 的狀態顯示為「主體」。如果您先前已在專案中訓練過模型，請等待最新的模型版本完成。

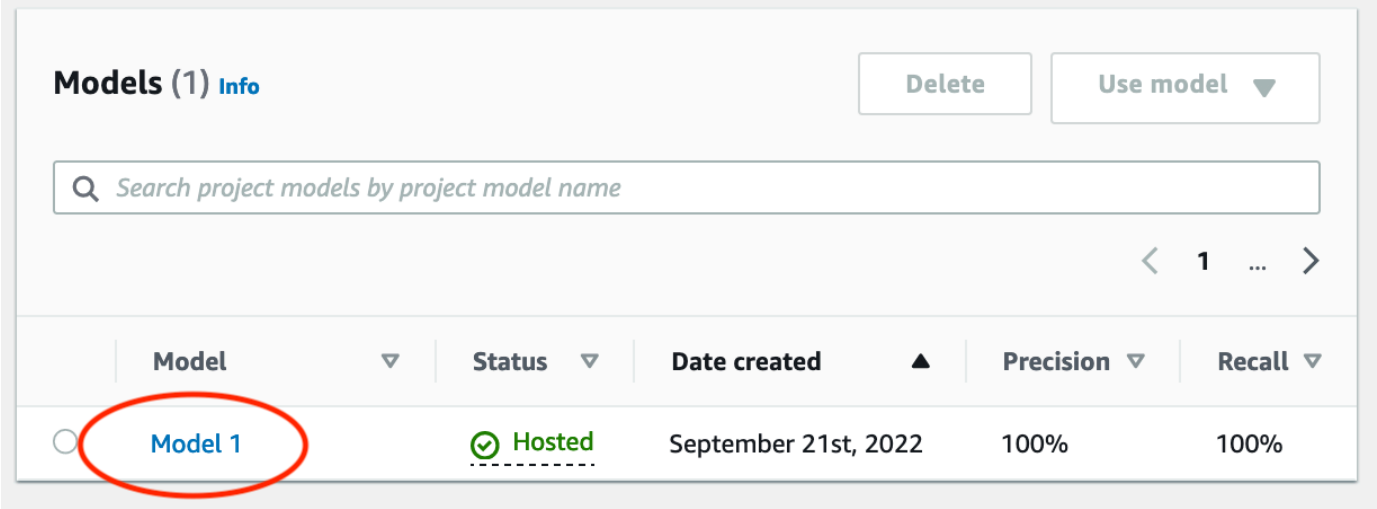


步驟 4：分析影像

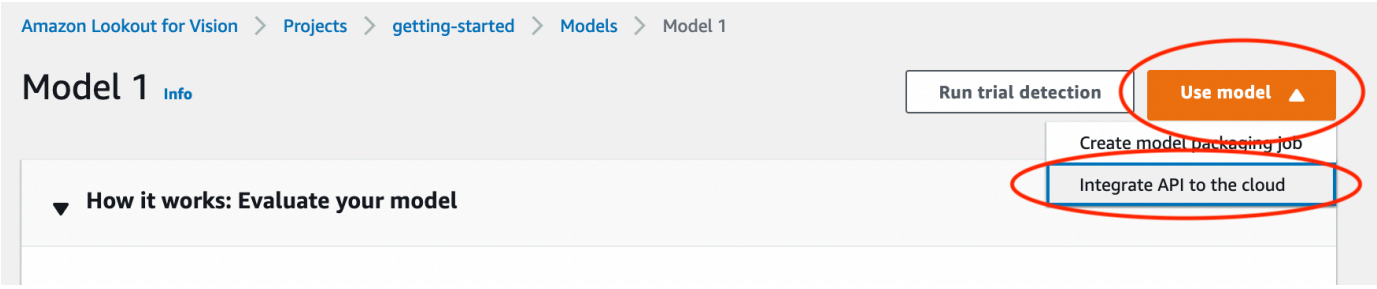
在此步驟中，將會分析模型的影像。我們提供範例影像，您可以在電腦上的 Lookout Vision 文件儲存庫中的入門`test-images`資料夾中使用。如需詳細資訊，請參閱偵測影像中的異常。

分析影像的步驟

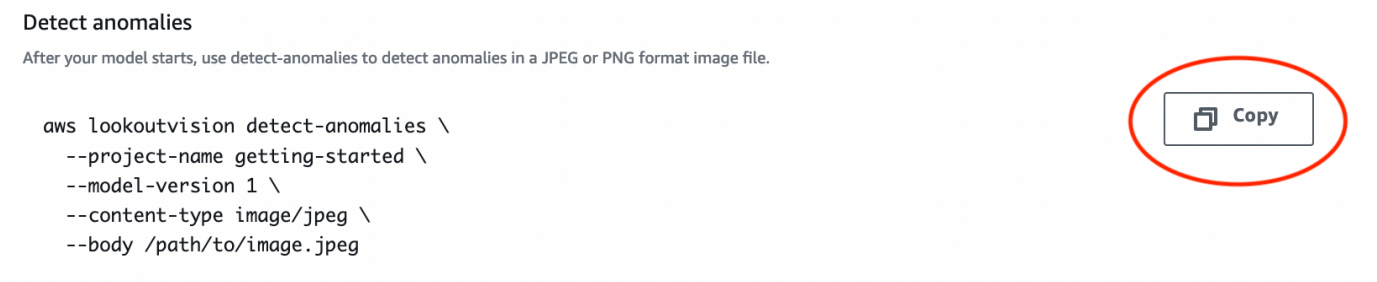
1. 在「型號」頁面上，選擇型號 1。



2. 在模型的詳細資料頁面上，選擇 [使用模型]，然後選擇 [將 API 整合至雲端]。



3. 在「指AWS CLI令」區段中，複製指`detect-anomalies`AWS CLI令。



4. 於指令提示下，透過輸入上一個步驟中的`detect-anomalies`指令來分析異常影像。對於`--body`參數，請從電腦上的入門`test-images`資料夾中指定異常影像。如果您使用`lookoutvision`設定檔取得認證，請新增`--profile lookoutvision-access`參數。例如：

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
```

```
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-anomaly-1.jpg \
  --profile lookoutvision-access
```

輸出格式應類似以下內容：

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.983975887298584,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.9818974137306213,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "cracked",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.018102575093507767,
                    "Color": "#23A436"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAAMACA......"
    }
}
```

5. 在輸出中，注意下列事項：

   - IsAnomalous是預測分類的布林值。 true如果圖像是異常的，否則false。
   - Confidence是一個浮點值，代表亞馬遜 Lookout for Vision 在預測中的信心。0 是最低的置信度，1 是最高的信心。
   - Anomalies是在影像中找到的異常清單。 Name是異常標籤。 PixelAnomaly包括異常 (TotalPercentageArea) 的總百分比面積和異常標籤的顏色 (Color)。該清單還包括「背景」異常，該異常覆蓋了在圖像上發現的異常之外的區域。

- AnomalyMask是顯示分析影像上異常位置的遮色片影像。

您可以使用回應中的資訊來顯示分析影像和異常遮色片的混合，如下列範例所示。如需範例程式碼，請參閱顯示分類和區段資訊。


Classification:
Prediction: Anomalous
Confidence: 99.9%
Segmentation:
Anomaly: cracked. Area: 6.2%

6. 在指令提示下，分析入門test-images資料夾中的一般影像。如果您使用lookoutvision設定檔取得認證，請新增--profile lookoutvision-access參數。例如：

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-normal-1.jpg \
  --profile lookoutvision-access
```

輸出格式應類似以下內容：

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": false,
        "Confidence": 0.9916400909423828,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 1.0,
                    "Color": "#FFFFFF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAA....."
    }
}
```

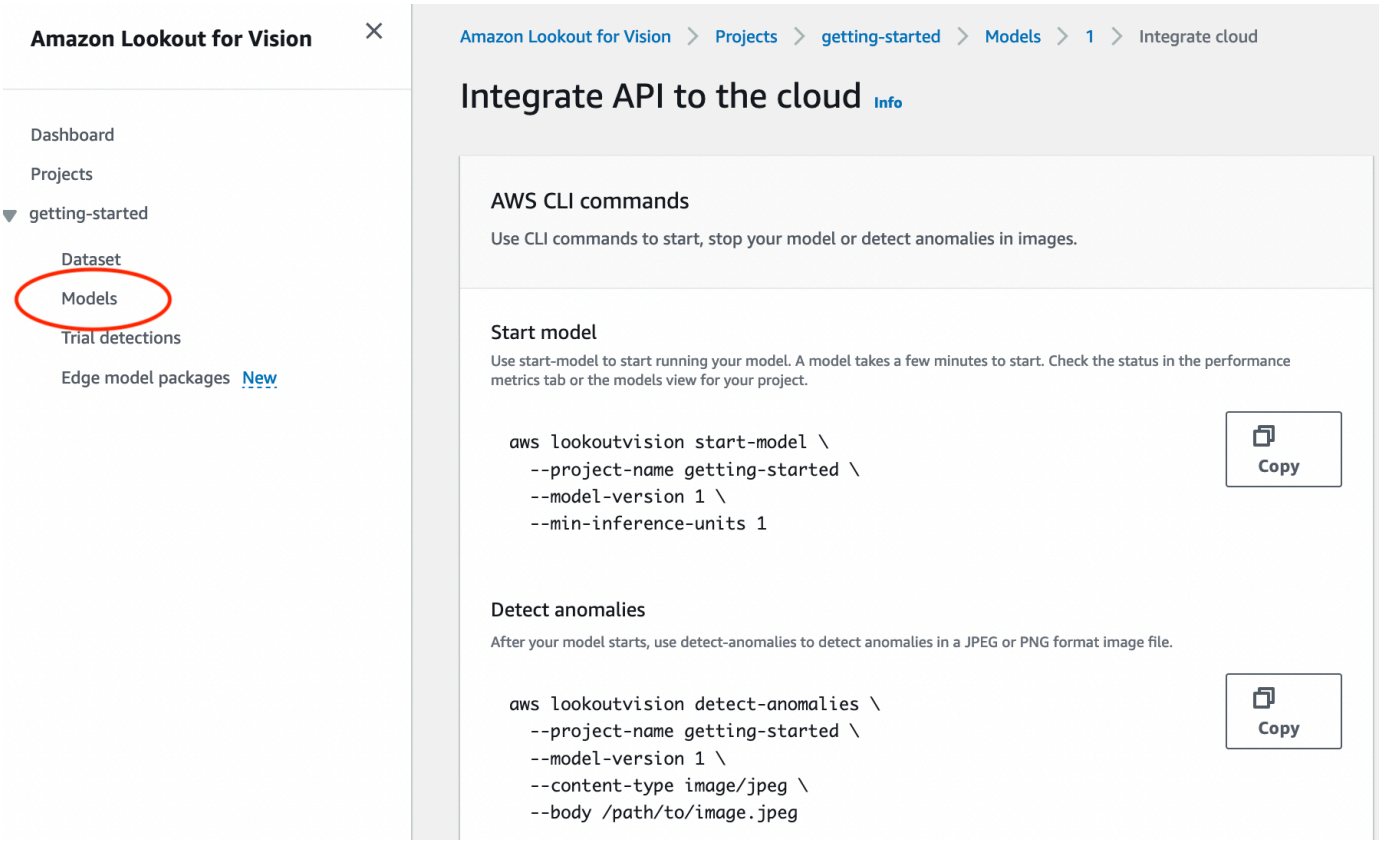
7. 在輸出中，請注意，用於將影像IsAnomalous分類為沒有異常的false值。用Confidence來協助決定您對分類的信心。此外，Anomalies陣列只有background異常標籤。

步驟 5：停止模型

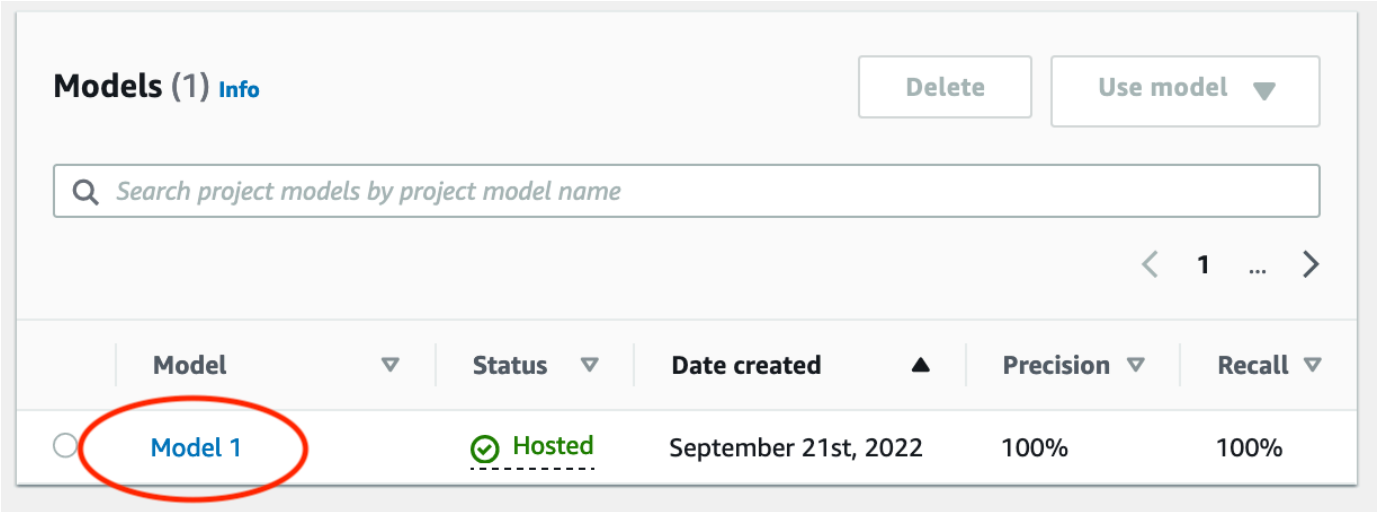
在此步驟中，將會停止主控模型。您需支付模型執行時間的費用。如果您不使用模型，則應停止它。下次需要時，將會重新啟動模型。如需詳細資訊，請參閱開始您的亞馬遜 Lookout for Vision 模型。

停止模型。

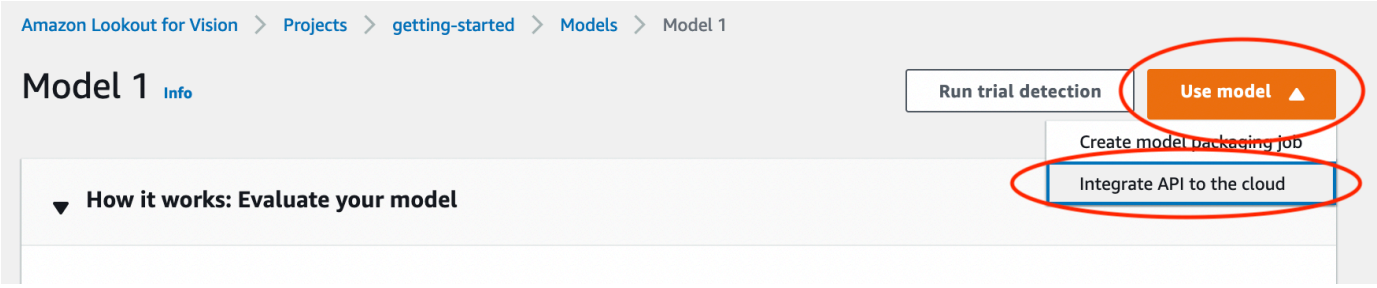
1. 在導覽窗格中，選擇模型。



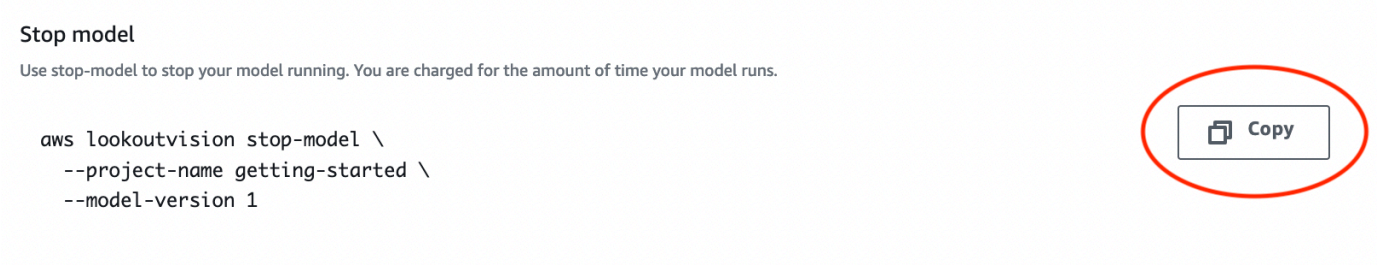
2. 在「型號」頁面中，選擇型號 1。



3. 在模型的詳細資料頁面上，選擇 [使用模型]，然後選擇 [將 API 整合至雲端]。



4. 在「指AWS CLI令」區段中，複製指stop-modelAWS CLI令。



5. 於指令提示下，透過輸入上一個步驟中的stop-modelAWS CLI指令來停止模型。如果您使用lookoutvision設定檔取得認證，請新增--profile lookoutvision-access參數。例如：

```
aws lookoutvision stop-model \
  --project-name getting-started \
  --model-version 1 \
  --profile lookoutvision-access
```

如果呼叫成功，則會顯示下列輸出：

```
{
    "Status": "STOPPING_HOSTING"
}
```

6. 返回控制台，在左側導航頁面中選擇模型。
7. 當「狀態」(Status) 欄中的模型狀態為「訓練完成」時，模型已停止。

後續步驟

當您準備好使用自己的影像建立模型時，請按照中的指示開始建立您的專案。這些說明包括使用 Amazon Lookout for Vision 主控台和 AWS SDK 建立模型的步驟。

如果您想嘗試其他範例資料集，請參閱範例程式碼和資料集。

創建您的 Amazon Lookout for Vision 模型

Amazon Lookout for Vision 模型是一種機器學習模型，可透過在用於訓練模型的影像中尋找模型的模式，來預測新影像中異常的存在。本節說明如何建立和訓練模型。訓練模型之後，您可以評估模型的效能。如需詳細資訊，請參閱 改善您的 Amazon Lookout for Vision 模型。

建立第一個模型之前，建議您先閱讀瞭解 Amazon Lookout for Vision和Amazon Lookout for Vision。如果您使用的是 AWS SDK，請閱讀致電 Amazon Lookout for Vision 操作。

主題

建立您的專案

Amazon Lookout for Vision 項目是創建和管理 Lookout for Vision 模型所需的資源分組。專案會管理下列項目：

- 資料集 — 用來訓練模型的影像和影像標籤。如需詳細資訊，請參閱 建立資料集。
- 模型 — 您訓練來偵測異常的軟體。一個模型可以有多個版本。如需詳細資訊，請參閱 培訓您的模型。

建議您針對單一使用案例使用專案，例如偵測單一機器零件類型中的異常情況。

> **Note**
>
> 您可以使 AWS CloudFormation 用佈建和設定視覺專案的 Amazon 瞭望台。如需詳細資訊，請參閱 使用記錄 Amazon Lookout for Vision 資源AWS CloudFormation。

若要檢視專案，請參閱檢視您的專案或開啟使用 Amazon LoLookout for Vision 儀表板. 若要刪除模型，請參閱刪除模型。

主題

創建項目（控制台）

下列程序說明如何使用主控台建立專案。

建立專案 (主控台)

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 在左側導覽視窗中，選擇專案。
3. 選擇建立專案。
4. 在 專案名稱 中，輸入您的專案名稱。
5. 選擇建立專案。專案的詳細資訊頁面隨即顯示。
6. 依照中的步驟建立資料集建立資料集。

建立專案 (SDK)

您可以使用該CreateProject操作來創建一個 Amazon Lookout for Vision 項目。來自的響應CreateProject包括項目名稱和項目的 Amazon 資源名稱（ARN）。之後，CreateDataset請打電話將訓練和測試數據集添加到您的項目中。如需詳細資訊，請參閱 使用資訊清單檔案 (SDK) 建立資料集。

若要檢視您在專案中建立的專案，請呼叫ListProjects。如需詳細資訊，請參閱 檢視您的專案。

若要建立專案 (SDK)

1. 如果您尚未這樣做，請安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來建立模型。

   CLI

   project-name將的值變更為您要用於專案的名稱。

```
aws lookoutvision create-project --project-name project name \
```

```
  --profile lookoutvision-access
```

Python

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def create_project(lookoutvision_client, project_name):
        """
        Creates a new Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param  project_name: The name for the new project.
        :return project_arn: The ARN of the new project.
        """
        try:
            logger.info("Creating project: %s", project_name)
            response =
 lookoutvision_client.create_project(ProjectName=project_name)
            project_arn = response["ProjectMetadata"]["ProjectArn"]
            logger.info("project ARN: %s", project_arn)
        except ClientError:
            logger.exception("Couldn't create project %s.", project_name)
            raise
        else:
            return project_arn
```

Java V2

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Creates an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return ProjectMetadata Metadata information about the created project.
 */
public static ProjectMetadata createProject(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {
```

```
        logger.log(Level.INFO, "Creating project: {0}", projectName);
        CreateProjectRequest createProjectRequest =
 CreateProjectRequest.builder().projectName(projectName)
                        .build();

        CreateProjectResponse response =
 lfvClient.createProject(createProjectRequest);

        logger.log(Level.INFO, "Project created. ARN: {0}",
 response.projectMetadata().projectArn());

        return response.projectMetadata();

}
```

3. 依照中的步驟使用 Amazon SageMaker Ground Truth 清單文件創建數據集建立資料集。

建立資料集

資料集包含您用來訓練和測試模型的影像和指派的標籤。您可以使用 Amazon Lookout 視覺主控台或使用CreateDataset作業建立專案的資料集。資料集影像必須根據您要建立的模型類型加上標籤 (影像分類或影像分割)。

主題

- 為資料集準備影像
- 建立資料集
- 使用儲存在本機電腦上的影像建立資料集
- 使用存放在 Amazon S3 儲存貯體中的映像建立資料集
- 使用 Amazon SageMaker Ground Truth 清單文件創建數據集

為資料集準備影像

您需要一組影像才能建立資料集。您的影像必須是 PNG 或 JPEG 格式的檔案。您需要的影像數量和類型取決於您的專案是否有單一資料集，還是單獨的訓練和測試資料集。

單一資料集專案

若要建立映像分類模型，您需要下列項目才能開始訓練：

- 普通對象的至少 20 圖像。
- 至少 10 個異常物體的圖像。

若要建立影像分割模型，您需要下列項目才能開始訓練：

- 每種異常類型至少 20 張圖像。
- 每個異常影像 (存在異常類型的影像) 必須只有一種異常類型。
- 普通對象的至少 20 圖像。

獨立的訓練和測試資料集專案

若要建立影像分類模型，您需要下列項目：

- 訓練資料集中至少有 10 個一般物件的影像。
- 測試資料集中至少有 10 個一般物件的影像。
- 測試資料集中至少 10 個異常物件的影像。

若要建立影像分割模型，您需要下列項目：

- 每個資料集至少需要 10 個每種異常類型的影像。
- 每個異常影像 (存在異常類型的影像) 只能包含一種異常類型。
- 每個資料集必須至少有 10 個一般物件的影像。

若要建立較高品質的模型，請使用大於最小數目的影像。如果您要建立區段模型，我們建議您加入具有多種異常類型的影像，但這些影像不會計入 Lookout for Vision 開始訓練所需的最低限度。

您的圖像應該是單一類型的對象。此外，您應該擁有一致的影像拍攝條件，例如相機定位、照明和物件姿勢。

訓練和測試資料集中的所有影像都必須具有相同的維度。稍後，您使用訓練的模型分析的影像必須具有與訓練和測試資料集影像相同的維度。如需詳細資訊，請參閱 偵測影像中的異常。

所有訓練和測試影像都必須是唯一的影像，最好是獨特的物件。一般影像應該會擷取被分析物件的正常變化。異常影像應擷取異常的多樣化取樣。

Amazon 視覺瞭望提供您可以使用的示例圖像。如需詳細資訊，請參閱 影像分類資料集。

如需影像限制，請參閱配額。

建立資料集

當您為專案建立資料集時，您可以選擇專案的初始資料集設定。您還可以選擇「Lookout for Vision」從中導入圖像的位置。

選擇專案的資料集設定

當您在專案中建立第一個資料集時，請選擇下列其中一個資料集設定：

- 單一資料集 — 單一資料集專案使用單一資料集來訓練和測試您的模型。使用單一資料集可讓 Amazon Lookout for Vision 選擇訓練和測試影像，從而簡化訓練。在訓練期間，Amazon Lookout for Vision 會將資料集內部分割為訓練資料集和測試資料集。您無法存取分割的資料集。我們建議在大多數情況下使用單一資料集專案。
- 單獨的訓練和測試資料集 — 如果您想要更精細地控制訓練、測試和效能調整，您可以將專案設定為具有獨立的訓練和測試資料集。如果您想要控制用於測試的影像，或者您已經有一組要使用的基準測試影像，請使用個別的測試資料集。

您可以將測試資料集新增至現有的單一資料集專案。然後，單一資料集會成為訓練資料集。如果您從具有不同訓練和測試資料集的專案中移除測試資料集，該專案就會變成單一資料集專案。如需詳細資訊，請參閱 刪除資料集。

匯入影像

建立資料集時，您可以選擇從何處匯入影像。視您匯入影像的方式而定，影像可能已經加上標籤。如果影像在建立資料集之後沒有標示，請參閱標記檔案。

您可以使用下列其中一種方式建立資料集並匯入其影像：

- 從本機電腦匯入影像。圖像沒有標記。您可以使用 Lookout for Vision 控制台添加或標籤。
- 從 S3 儲存貯體匯入映像。Amazon Lookout for Vision 察可以通過使用文件夾名稱來標記圖像進行分類。用`normal`於一般影像。用`anomaly`於異常影像。您無法自動指派區段標籤。
- 導入 Amazon SageMaker Ground Truth 清單文件，其中包括標記的圖像。您可以創建和導入自己的清單文件。如果您有很多圖像，請考慮使用「 SageMaker Ground Truth」標籤服務。然後，您可以從 Amazon SageMaker Ground Truth 工作匯入輸出資訊清單檔案。如有必要，您可以使用 LoLookout for Vision 主控台來新增或變更標籤。

如果您使用的是 AWS SDK，您可以使用 Amazon SageMaker Ground Truth 資訊清單檔案建立資料集。如需詳細資訊，請參閱 使用 Amazon SageMaker Ground Truth 清單文件創建數據集。

如果在建立資料集之後，您的影像已加上標籤，您可以訓練模型。如果圖像沒有標記，請根據您要創建的模型類型添加標籤。如需詳細資訊，請參閱 標記檔案。

您可以將更多影像新增至現有的資料集。如需詳細資訊，請參閱 將影像新增至資料集。

使用儲存在本機電腦上的影像建立資料集

您可以使用直接從電腦載入的影像來建立資料集。您一次最多可以上傳 30 個影像。在此程序中，您可以建立單一資料集專案，或建立具有不同訓練和測試資料集的專案。

> **Note**
>
> 如果您剛完成建立您的專案，主控台應該會顯示您的專案儀表板，而您不需要執行步驟 1-3。

使用本機電腦 (主控台) 上的影像建立資料集

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 在左側導覽視窗中，選擇專案。
3. 在專案 頁面，選擇您要新增資料集的專案。
4. 在專案的詳細頁面上，選擇 建立資料集。
5. 選擇 [單一資料集] 索引標籤或 [個別訓練與測試資料集] 索引標籤，然後依照步驟

   Single dataset

   a. 在 [資料集設定] 區段中，選擇 [建立單一資料集]。
   b. 在 [映像來源設定] 區段中，選擇 [從您的電腦上傳影像]。
   c. 選擇建立資料集。
   d. 在資料集頁面上，選擇 [新增影像]。
   e. 從電腦檔案中選擇要上傳至資料集的影像。您可以拖曳影像或從本機電腦選擇要上傳的影像。
   f. 選擇上傳影像。

Separate training and test datasets

a. 在 [資料集設定] 區段中，選擇 [建立訓練資料集和測試資料集]。

b. 在 培訓資料集詳細資料 的區段中，選擇 從電腦上傳影像。

c. 在測試資料集詳細資訊區段中，選擇從您的電腦上傳影像。

> **Note**
>
> 您的訓練和測試資料集可以有不同的影像來源。

d. 選擇建立資料集。隨即會顯示資料集頁面，其中包含對應資料集的培訓索引標籤和測試索引標籤。

e. 選擇動作，然後選擇新增影像至訓練資料集。

f. 選擇要上傳至資料集的影像。您可以拖曳影像或從本機電腦選擇要上傳的影像。

g. 選擇上傳影像。

h. 重複步驟 5e 至 5g。對於步驟 5e，請選擇動作，然後選擇新增影像至測試資料集。

6. 請遵循 標記檔案 中的步驟標記影像。
7. 請遵循 培訓您的模型 中的步驟訓練模型。

使用存放在 Amazon S3 儲存貯體中的映像建立資料集

您可以使用存放在 Amazon S3 儲存貯體中的映像建立資料集。使用此選項，您可以使用 Amazon S3 儲存貯體中的資料夾結構自動分類映像。您可以將映像存放在主控台儲存貯體或帳戶中的其他 Amazon S3 儲存貯體中。

設定用於自動標籤的資料夾

在建立資料集期間，您可以選擇根據包含影像的資料夾名稱，為影像分配標籤名稱。當您建立資料集時，這些資料夾必須是您在 S3 URI 中指定的 Amazon S3 資料夾路徑的子系。

以下是入門範例影像的`train`資料夾。如果您將 Amazon S3 資料夾位置指定為`S3-bucket/circuitboard/train/`，則資料夾中的映像`normal`會指派標籤`Normal`。資料夾中的影像`anomaly`會指定標籤`Anomaly`。較深的子資料夾的名稱不會用於標記影像。

```
S3-bucket
  ### circuitboard
    ### train
        ### anomaly
            ### train-anomaly_1.jpg
            ### train-anomaly_2.jpg
            ### .
            ### .
        ### normal
            ### train-normal_1.jpg
            ### train-normal_2.jpg
            ### .
            ### .
```

使用 Amazon S3 儲存貯體中的映像檔建立資料集

下列程序使用存放在 Amazon S3 儲存貯體中的分類範例映像建立資料集。若要使用您自己的影像，請建立中所述的資料夾結構設定用於自動標籤的資料夾。

此程序也會示範如何建立單一資料集專案，或使用個別訓練與測試資料集的專案。

如果您不選擇自動為影像加上標籤，則需要在建立資料集之後為影像加上標籤。如需詳細資訊，請參閱分類映像（控制台）。

> **Note**
>
> 如果您剛完成建立您的專案，主控台應該會顯示您的專案儀表板，而您不需要執行步驟 1-4。

使用存放在 Amazon S3 儲存貯體中的映像建立資料集

1. 如果您尚未這樣做，請將入門映像上傳到 Amazon S3 儲存貯體。如需詳細資訊，請參閱 影像分類資料集。
2. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
3. 在左側導覽視窗中，選擇專案。
4. 在專案 頁面，選擇您要新增資料集的專案。專案的詳細資訊頁面隨即顯示。
5. 選擇建立資料集。建立資料集頁面即會顯示。

> **Tip**
>
> 如果您遵循入門指示，請選擇 [建立訓練資料集和測試資料集]。

6. 選擇 [單一資料集] 索引標籤或 [個別訓練與測試資料集] 索引標籤，然後依照步驟

   Single dataset

   a. 在 [資料集設定] 區段中，選擇 [建立單一資料集]。

   b. 在 [映像來源設定] 區段中輸入步驟 7-9 的資訊。

   Separate training and test datasets

   a. 在 [資料集設定] 區段中，選擇 [建立訓練資料集和測試資料集]。

   b. 對於訓練資料集，請在訓練資料集詳細資料區段中輸入步驟 7-9 的資訊。

   c. 對於測試資料集，請在「測試資料集詳細資料」區段中輸入步驟 7 至 9 的資訊。

   > **Note**
   >
   > 您的訓練和測試資料集可以有不同的影像來源。

7. 選擇從 Amazon S3 儲存貯體匯入影像。

8. 在 S3 URI 中，輸入 Amazon S3 儲存貯體位置和資料夾路徑。將 `bucket` 變更為 Amazon S3 儲存貯體的名稱。

   a. 如果您要建立單一資料集專案或訓練資料集，請輸入下列內容：

   ```
   s3://bucket/circuitboard/train/
   ```

   b. 如果您要建立測試資料集，請輸入下列內容：

   ```
   s3://bucket/circuitboard/test/
   ```

9. 選擇根據資料夾自動將標籤連接至影像。

10. 選擇建立資料集。隨即開啟資料集頁面，其中包含已標記的影

11. 請遵循 培訓您的模型 中的步驟訓練模型。

使用 Amazon SageMaker Ground Truth 清單文件創建數據集

資訊清單檔案包含可用來訓練和測試模型的影像和影像標籤的相關資訊。您可以將資訊清單檔案存放在 Amazon S3 儲存貯體中，並使用它來建立資料集。您可以建立自己的資訊清單檔案，也可以使用現有的資訊清單檔案，例如 Amazon SageMaker Ground Truth 任務的輸出。

主題

- 使用 Amazon Ground Truth 工作
- 建立清單檔案

使用 Amazon Ground Truth 工作

標記圖像可能需要大量時間。例如，在異常周圍精確繪製遮色片可能需要 10 秒鐘的時間。如果您有 100 張圖像，則可能需要幾個小時才能標記它們。作為自己標記圖像的替代方法，請考慮使用 Amazon SageMaker Ground Truth。

使用 Amazon SageMaker Ground Truth，您可以使用來自您選擇的廠商公司 Amazon Mechanical Turk 的工作人員或內部私人員工來建立一組已標記的影像。如需詳細資訊，請參閱使用 Amazon SageMaker Ground Truth 來標記資料。

有使用 Amazon Mechanical Turk 的成本。另外，完成 Amazon Ground Truth 標籤任務可能需要幾天的時間。如果成本有問題，或者您需要快速訓練模型，建議您使用 Amazon Lookout 視覺主控台為影像加上標籤。

您可以使用 Amazon SageMaker Ground Truth 標籤任務來標記適用於影像分類模型和影像分割模型的影像。任務完成後，您可以使用輸出資訊清單檔案來建立視覺版 Amazon 瞭望資料集。

Image classification

若要標示影像分類模型的影像，請為影像分類 (單一標籤) 工作建立標籤工作。

圖像分割

若要為影像分割模型標示影像，請為影像分類 (單一標籤) 工作建立標籤工作。然後，鏈結工作以建立「影像語意分割」工作的標籤工作。

您也可以使用標籤工作，為影像分割模型建立部分資訊清單檔案。例如，您可以使用「影像分類 (單一標籤)」工作來分類影像。使用任務輸出建立 Lookout for Vision 資料集之後，請使用 Amazon Lookout for Vision 主控台將區段遮罩和異常標籤新增至資料集映像。

使用 Amazon SageMaker Ground Truth 標記圖像

下列程序說明如何使用 Amazon SageMaker Ground Truth 影像標籤任務來標記影像。此程序會建立映像分類資訊清單檔案，並選擇性地鏈結映像標籤工作以建立映像分割資訊清單檔案。如果您希望專案具有單獨的測試資料集，請重複此程序以建立測試資料集的資訊清單檔案。

使用 Amazon SageMaker Ground Truth 標記圖像（控制台）

1. 依照建立標籤 Job (主控台) 中的指示，為映像分類 (單一標籤) 工作建立「Ground Truth」工作。
   a. 在步驟 10 中，從 [工作類別] 下拉式功能表中選擇 [影像]，然後選擇 [影像分類 (單一標籤)] 做為工作類型。
   b. 在步驟 16 中，在影像分類 (單一標籤) 標籤工具區段中，新增兩個標籤：正常標籤和異常。
2. 等到工作人員完成您的圖像分類。
3. 如果您要為影像分割模型建立資料集，請執行下列動作。否則，請轉到步驟 4。
   a. 在 Amazon SageMaker Ground Truth 主控台中，開啟「標籤任務」頁面。
   b. 選擇您先前建立的工作。這樣會啟用動作功能表。
   c. 從動作功能表中，選擇串連。工作詳細資訊頁面隨即開啟。
   d. 在工作類型中，選擇語意分割。
   e. 選擇下一步。
   f. 在 [語意分割標籤工具] 區段中，為您希望模型尋找的每種異常類型新增異常標籤。
   g. 選擇建立。
   h. 等到工作人員標記您的圖像。
4. 開啟「Ground Truth」主控台並開啟「標籤工作」頁面。
5. 如果要建立影像分類模型，請選擇您在步驟 1 中建立的工作。如果要建立影像分割模型，請選擇步驟 3 中建立的工作。
6. 在標籤工作摘要中，在輸出資料集位置中開啟 S3 位置。請注意資訊清單檔案的位置，應該是`s3://`*`output-dataset-location`*`/manifests/output/output.manifest`。
7. 如果您想要為測試資料集建立資訊清單檔案，請重複此程序。否則，請按照中的說明建立資料集建立包含資訊清單檔案的資料集。

建立資料集

使用此程序，在 Lookout Vision 專案中使用您在的步驟 6 中記下的資訊清單檔案建立資料集 使用 Amazon SageMaker Ground Truth 標記圖像。資訊清單檔案會為單一資料集專案建立訓練資料集。如

果您希望專案擁有單獨的測試資料集，可以執行另一個 Amazon SageMaker Ground Truth 任務，為測試資料集建立資訊清單檔案。或者，您可以自己創建清單文件。您也可以從 Amazon S3 儲存貯體或本機電腦將映像匯入測試資料集。（圖像可能需要標籤才能訓練模型）。

此程序假設您的專案沒有任何資料集。

若要使用 Lookout for Vision 建立資料集 (主控台)

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇開始使用。
3. 在左側導覽視窗中，選擇專案。
4. 選擇您要新增與資訊清單檔案搭配使用的專案。
5. 在 [運作方式] 區段中，選擇 [建立資料集]。
6. 選擇 [單一資料集] 索引標籤或 [個別訓練與測試資料集] 索引標籤，然後依照步驟

   Single dataset

   1. 選擇 [建立單一資料集]。
   2. 在 [影像來源設定] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像]。
   3. 對於 .manifest 檔案位置，請輸入您在的步驟 6 中記下的資訊清單檔案的 使用 Amazon SageMaker Ground Truth 標記圖像位置。

   Separate training and test datasets

   1. 選擇 [建立訓練資料集和測試資料集]。
   2. 在 [訓練資料集詳細資料] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像
   3. 在 .manifest 檔案位置中，您在的步驟 6 中記下的資訊清單檔案的 使用 Amazon SageMaker Ground Truth 標記圖像位置。
   4. 在 [測試資料集詳細資料] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像
   5. 在 .manifest 檔案位置中，您在的步驟 6 中記下的資訊清單檔案的 使用 Amazon SageMaker Ground Truth 標記圖像位置。請記住，測試資料集需要個別的資訊清單檔案。
7. 選擇提交。
8. 請遵循 培訓您的模型 中的步驟訓練模型。

建立清單檔案

您可以透過匯入 SageMaker Ground Truth 格式資訊清單檔案來建立資料集。如果您的圖像標記的格式不是 SageMaker Ground Truth 清單文件，請使用以下信息創建 SageMaker Ground Truth 格式的清單文件。

清單檔案採用 JSON Lines 格式，其中每行都是一個完整的 JSON 物件，代表影像的標記資訊。影像分類和影像分割有不同的格式。清單文件必須使用 UTF-8 編碼進行編碼。

> **Note**
>
> 本區段中的 JSON Line 範例已格式化以提高可讀性。

清單檔案所參考的影像必須位於同一個 Amazon S3 儲存貯體中。清單文件可以位於不同的存儲桶中。您可以在 JSON Line 的 `source-ref` 欄位中指定影像的位置。

您可以通過使用代碼創建清單文件。Amazon 觀察視覺實驗室 Python 筆記本顯示瞭如何為電路板示例圖像創建圖像分類清單文件。或者，您也可以使用程式碼範例儲存庫中的資料集範例 AWS 程式碼。您可以使用逗號分隔值 (CSV) 檔案輕鬆建立資訊清單檔案。如需詳細資訊，請參閱 從 CSV 檔案建立分類資訊清單檔案。

主題

- 定義圖像分類的 JSON 行
- 定義用於圖像分割的 JSON 行
- 從 CSV 檔案建立分類資訊清單檔案
- 使用清單文件（控制台）創建數據集
- 使用資訊清單檔案 (SDK) 建立資料集

定義圖像分類的 JSON 行

您可以為想要在 Amazon 觀察視覺資訊清單檔案中使用的每個映像定義一個 JSON 行。如果您要建立分類模型，JSON 行必須包含正常或異常的影像分類。JSON 行是 SageMaker Ground Truth 分類 Job 輸出格式。清單檔案由一個或多個 JSON Lines 組成，每個要匯入的影像都有一個。

建立已分類影像的資訊清單檔案

1. 建立空白文字檔案。

2. 針對要匯入的每個影像新增 JSON Line。每個 JSON 行看起來應類似於以下內容：

```
{"source-ref":"s3://lookoutvision-console-us-east-1-nnnnnnnnnn/gt-job/manifest/
IMG_1133.png","anomaly-label":1,"anomaly-label-metadata":{"confidence":0.95,"job-
name":"labeling-job/testclconsolebucket","class-name":"normal","human-
annotated":"yes","creation-date":"2020-04-15T20:17:23.433061","type":"groundtruth/
image-classification"}}
```

3. 儲存檔案。

> **Note**
>
> 您可以使用副檔名 .manifest，但這不是必要的。

4. 使用您建立的清單檔案建立資料集。如需詳細資訊，請參閱 建立清單檔案。

將 JSON 行分類

在本節中，您將學習如何建立將圖像分類為正常或異常的 JSON 行。

異常行

下列 JSON 行顯示標記為異常的影像。請注意，的值class-name是anomaly。

```
{
    "source-ref": "s3: //bucket/image/anomaly/abnormal-1.jpg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "anomaly",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.600",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 1
}
```

正常的 JSON 行

下面的 JSON 行顯示標記為正常的圖像。請注意，的值class-name是normal。

```
{
    "source-ref": "s3: //bucket/image/normal/2020-10-20_12-14-55_613.jpeg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "normal",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.603",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 0
}
```

JSON 線路索引鍵和值

下列資訊說明 Amazon Lookout for Vision JSON 行中的索引鍵和值。

source-ref

(必要) 影像的 Amazon S3 位置。格式是 "s3://*BUCKET*/*OBJECT_PATH*"。匯入資料集中的影像必須存放在同一個 Amazon S3 儲存貯體中。

異常標籤

(必要) 屬性標籤。使用金鑰anomaly-label或您選擇的其他金鑰名稱。關鍵值（0在前面的例子中）是由 Amazon Lookout for Vision 所需的，但不使用它。Amazon Lookout for Vision 創建的輸出清單將值轉換1為異常圖像和一般圖像0的值。的值class-name決定影像是正常還是異常。

必須有由欄位名稱識別的對應中繼資料，並附加了 -metadata。例如 "anomaly-label-metadata"。

異常標籤中繼資料

(必要) 有關標籤屬性的中繼資料。欄位名稱必須與附加了 -metadata 的標籤屬性相同。

信賴度

（可選）Amazon 觀察目前未使用視覺。如果您確實指定了值，請使用的值1。

job-name

(選用) 您為處理影像的任務選擇的名稱。

class-name

(必要) 如果影像包含一般內容，請指定normal，否則請指定anomaly。如果的值class-name是任何其他值，則會將影像新增至資料集，做為未標籤的影像。若要標示影像，請參閱將影像新增至資料集。

human-annotated

(必要) 如果註釋由人類完成，則指定 "yes"。否則請指定 "no"。

creation-date

(選擇性) 建立標籤的國際標準時間 (UTC) 日期和時間。

type

(必要) 應套用至影像的處理類型。對於影像層級異常標籤，值為。"groundtruth/image-classification"

定義用於圖像分割的 JSON 行

您可以為想要在 Amazon 觀察視覺資訊清單檔案中使用的每個映像定義一個 JSON 行。如果您想要建立區段模型，JSON 行必須包含影像的區段和分類資訊。清單檔案由一個或多個 JSON Lines 組成，每個要匯入的影像都有一個。

建立分段影像的資訊清單檔案

1. 建立空白文字檔案。
2. 針對要匯入的每個影像新增 JSON Line。每個 JSON 行看起來應類似於以下內容：

```
{"source-ref":"s3://path-to-image","anomaly-label":1,"anomaly-label-metadata":
{"class-name":"anomaly","creation-date":"2021-10-12T14:16:45.668","human-
annotated":"yes","job-name":"labeling-job/classification-job","type":"groundtruth/
image-classification","confidence":1},"anomaly-mask-ref":"s3://path-to-
image","anomaly-mask-ref-metadata":{"internal-color-map":{"0":{"class-
name":"BACKGROUND","hex-color":"#ffffff","confidence":0.0},"1":{"class-
name":"scratch","hex-color":"#2ca02c","confidence":0.0},"2":{"class-
name":"dent","hex-color":"#1f77b4","confidence":0.0}},"type":"groundtruth/
semantic-segmentation","human-annotated":"yes","creation-
date":"2021-11-23T20:31:57.758889","job-name":"labeling-job/segmentation-job"}}
```

3. 儲存檔案。

> **Note**
>
> 您可以使用副檔名 .manifest，但這不是必要的。

4. 使用您建立的清單檔案建立資料集。如需詳細資訊，請參閱 建立清單檔案。

分割 JSON 行

在本節中，您將學習如何建立 JSON 行，其中包含影像的區段和分類資訊。

下列 JSON 行顯示含有分割和分類資訊的影像。 anomaly-label-metadata包含分類資訊。anomaly-mask-ref並anomaly-mask-ref-metadata包含區段資訊。

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
```

```
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

JSON 線路索引鍵和值

下列資訊說明 Amazon Lookout for Vision JSON 行中的索引鍵和值。

source-ref

(必要) 影像的 Amazon S3 位置。格式是 "s3://*BUCKET*/*OBJECT_PATH*"。匯入資料集中的影像必須存放在同一個 Amazon S3 儲存貯體中。

異常標籤

(必要) 屬性標籤。使用金鑰anomaly-label或您選擇的其他金鑰名稱。關鍵值（1在前面的例子中）是由 Amazon Lookout for Vision 所需的，但不使用它。Amazon Lookout for Vision 創建的輸出清單將值轉換1為異常圖像和一般圖像0的值。的值class-name決定影像是正常還是異常。

必須有由欄位名稱識別的對應中繼資料，並附加了 -metadata。例如 "anomaly-label-metadata"。

異常標籤中繼資料

(必要) 有關標籤屬性的中繼資料。包含分類資訊。欄位名稱必須與附加了 -metadata 的標籤屬性相同。

信賴度

（可選）Amazon 觀察目前未使用視覺。如果您確實指定了值，請使用的值1。

job-name

(選用) 您為處理影像的任務選擇的名稱。

class-name

(必要) 如果影像包含一般內容，請指定normal，否則請指定anomaly。如果的值class-name是任何其他值，則會將影像新增至資料集，做為未標籤的影像。若要標示影像，請參閱將影像新增至資料集。

human-annotated

(必要) 如果註釋由人類完成，則指定 "yes"。否則請指定 "no"。

creation-date

(選擇性) 建立標籤的國際標準時間 (UTC) 日期和時間。

type

(必要) 應套用至影像的處理類型。使用值"groundtruth/image-classification"。

異常掩碼參考

(必要) 遮罩影像的 Amazon S3 位置。用anomaly-mask-ref於金鑰名稱或使用您選擇的金鑰名稱。金鑰必須以結尾-ref。遮色片影像必須包含每種異常類型internal-color-map的彩色遮色片。格式是 "s3://*BUCKET*/*OBJECT_PATH*"。匯入資料集中的影像必須存放在同一個 Amazon S3 儲存貯體中。遮罩影像必須是可攜式網路圖形 (PNG) 格式的影像。

異常遮罩參考中繼資料

(必要) 分割影像的中繼資料。用anomaly-mask-ref-metadata於金鑰名稱或使用您選擇的金鑰名稱。金鑰名稱必須以結尾-ref-metadata。

內部顏色映射

(必要) 對應至個別異常類型的顏色對應。顏色必須與遮色片影像中的顏色相符 (anomaly-mask-ref)。

金鑰

(必填) 鍵進入地圖。項目0必須包含代表影像異常以外區域的類別名稱 BACKAGE。

class-name

(必要) 異常類型的名稱，例如刮痕或凹痕。

十六進制顏色

(必要) 異常類型的十六進位顏色，例如#2ca02c。顏色必須與中的顏色相符anomaly-mask-ref。BACKGROUND異常類型的值永遠#ffffff為。

信賴度

(必要) Amazon Lookout for Vision 目前尚未使用，但需要浮點值。

human-annotated

(必要) 如果註釋由人類完成，則指定 "yes"。否則請指定 "no"。

creation-date

(選擇性) 建立區段資訊的國際標準時間 (UTC) 日期和時間。

type

(必要) 應套用至影像的處理類型。使用值"groundtruth/semantic-segmentation"。

從 CSV 檔案建立分類資訊清單檔案

這個範例 Python 指令碼使用逗號分隔值 (CSV) 檔案來標記影像，簡化了分類資訊清單檔案的建立。建立 CSV 檔案。

清單檔案會描述用於訓練模型的影像。清單檔案由一或多個 JSON Lines 組成。每個 JSON Line 會描述單一影像。如需詳細資訊，請參閱 定義圖像分類的 JSON 行。

CSV 檔案代表文字檔案中多資料列的表格式資料。資料列中的欄位以逗號分隔。如需詳細資訊，請參閱逗號分隔值。對於此指令碼，CSV 檔案中的每一列都包含映像的 S3 位置以及映像 (normal或anomaly) 的異常分類。每一列對應至資訊清單檔案中的 JSON 行。

例如，下列 CSV 檔案描述範例影像中的某些影像。

```
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train-normal_1.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_10.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_11.jpg,normal
```

該指令碼會為每一資料列產生 JSON Lines。例如，以下是第一資料列 (s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly) 的 JSON Line。

```
{"source-ref": "s3://s3bucket/csv_test/train_anomaly_1.jpg","anomaly-label":
 1,"anomaly-label-metadata": {"confidence": 1,"job-name": "labeling-job/anomaly-
classification","class-name": "anomaly","human-annotated": "yes","creation-date":
 "2022-02-04T22:47:07","type": "groundtruth/image-classification"}}
```

如果您的 CSV 檔案不包含映像檔的 Amazon S3 路徑，請使用`--s3-path`命令列引數來指定影像的 Amazon S3 路徑。

在建立資訊清單檔案之前，指令碼會檢查 CSV 檔案中是否有重複的映像，以及任何不是`normal`或的映像分類`anomaly`。如果發現重複的影像或影像分類錯誤，指令碼會執行下列動作：

- 將所有影像的第一個有效影像項目記錄在刪除重複的 CSV 檔案中。
- 在錯誤檔案中記錄影像的重複出現次數。
- 記錄不在錯誤檔案`anomaly`中`normal`或錯誤檔案中的影像分類。
- 不建立資訊清單檔案。

錯誤檔案包含在輸入 CSV 檔案中發現重複影像或分類錯誤的行號。使用錯誤 CSV 檔案更新輸入 CSV 檔案，然後再次執行指令碼。或者，您也可以使用錯誤 CSV 檔案來更新已刪除重複資料的 CSV 檔案，該檔案僅包含唯一的影像項目和沒有影像分類錯誤的影像。使用已更新的刪除重複資料的 CSV 檔案重新執行指令碼。

如果在輸入的 CSV 檔案中找不到重複項目或錯誤，指令碼會刪除重複的影像 CSV 檔案和錯誤檔案，因為它們是空的。

在此程序中，您可以建立 CSV 檔案並執行 Python 指令碼來建立清單檔案。該腳本已經與 Python 版本 3.7 進行了測試。

從 CSV 檔案建立清單檔案

1. 建立 CSV 檔案，每一資料列中包含以下欄位 (每個影像一個資料列)。請勿將標題資料列新增至 CSV 檔案。

欄位 1	欄位 2
影像名稱或 Amazon S3 路徑影像。例如 `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` 。您不能混合使用具有 Amazon	影像的異常分類 (`normal`或`anomaly`)。

欄位 1	欄位 2
S3 路徑的影像和不具有 Amazon S3 路徑的影像。	

例如：s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly 或 image_11.jpg,normal

2. 儲存 CSV 檔案。
3. 執行下列 Python 指令碼。提供下列引數：

   - csv_file — 您在步驟 1 中建立的 CSV 檔案。
   - (選用) --s3-path *s3://path_to_folder/* — 要新增至影像檔案名稱的 Amazon S3 路徑 (欄位 1)。如果欄位 1 中的影像尚未包含 S3 路徑，請使用 --s3-path。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0
"""
Purpose
Shows how to create an Amazon Lookout for Vision manifest file from a CSV file.
The CSV file format is image location,anomaly classification (normal or anomaly)
For example:
s3://s3bucket/circuitboard/train/anomaly/train_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train_1.jpg,normal

If necessary, use the bucket argument to specify the Amazon S3 bucket folder for
 the images.
"""

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

logger = logging.getLogger(__name__)


def check_errors(csv_file):
```

```
    """
    Checks for duplicate images and incorrect classifications in a CSV file.
    If duplicate images or invalid anomaly assignments are found, an errors CSV
file
    and deduplicated CSV file are created. Only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in the
errors file.
    :param csv_file: The source CSV file
    :return: True if errors or duplicates are found, otherwise false.
    """

    logger.info("Checking %s.", csv_file)

    errors_found = False
    errors_file = f"{os.path.splitext(csv_file)[0]}_errors.csv"
    deduplicated_file = f"{os.path.splitext(csv_file)[0]}_deduplicated.csv"

    with open(csv_file, 'r', encoding="UTF-8") as input_file,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(errors_file, 'w', encoding="UTF-8") as errors:

        reader = csv.reader(input_file, delimiter=',')
        dedup_writer = csv.writer(dedup)
        error_writer = csv.writer(errors)
        line = 1
        entries = set()
        for row in reader:

            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            # Record any incorrect classifications.
            if not row[1].lower() == "normal" and not row[1].lower() == "anomaly":
                error_writer.writerow(
                    [line, row[0], row[1], "INVALID_CLASSIFICATION"])
                errors_found = True

            # Write first image entry to dedup file and record duplicates.
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
```

```
                error_writer.writerow([line, row[0], row[1], "DUPLICATE"])
                errors_found = True
            line += 1

    if errors_found:
        logger.info("Errors found check %s.", errors_file)
    else:
        os.remove(errors_file)
        os.remove(deduplicated_file)

    return errors_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Read a CSV file and create an Amazon Lookout for Vision classification manifest
 file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The Amazon S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s.", csv_file)

    image_count = 0
    anomalous_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
        open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')

        # Process each row (image) in the CSV file.
        for row in image_classifications:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            source_ref = str(s3_path) + row[0]
            classification = 0

            if row[1].lower() == 'anomaly':
                classification = 1
                anomalous_count += 1
```

```
            # Create the JSON line.
             json_line = {}
             json_line['source-ref'] = source_ref
             json_line['anomaly-label'] = str(classification)

             metadata = {}
             metadata['confidence'] = 1
             metadata['job-name'] = "labeling-job/anomaly-classification"
             metadata['class-name'] = row[1]
             metadata['human-annotated'] = "yes"
             metadata['creation-date'] = datetime.now(timezone.utc).strftime('%Y-%m-
%dT%H:%M:%S.%f')
             metadata['type'] = "groundtruth/image-classification"

             json_line['anomaly-label-metadata'] = metadata

             output_file.write(json.dumps(json_line))
             output_file.write('\n')
             image_count += 1

    logger.info("Finished creating manifest file %s.\n"
                "Images: %s\nAnomalous: %s",
                manifest_file,
                image_count,
                anomalous_count)
    return image_count, anomalous_count


def add_arguments(parser):
    """
    Add command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "csv_file", help="The CSV file that you want to process."
    )

    parser.add_argument(
        "--s3_path", help="The Amazon S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the Amazon S3 path.",
 required=False
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()
        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ""

        csv_file = args.csv_file
        csv_file_no_extension = os.path.splitext(csv_file)[0]
        manifest_file = csv_file_no_extension + '.manifest'

        # Create manifest file if there are no duplicate images.
        if check_errors(csv_file):
            print(f"Issues found. Use {csv_file_no_extension}_errors.csv "\
                "to view duplicates and errors.")
            print(f"{csv_file}_deduplicated.csv contains the first"\
                "occurrence of a duplicate.\n"
                  "Update as necessary with the correct information.")
            print(f"Re-run the script with
 {csv_file_no_extension}_deduplicated.csv")
        else:
            print('No duplicates found. Creating manifest file.')

            image_count, anomalous_count = create_manifest_file(csv_file,
 manifest_file, s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n")

            normal_count = image_count-anomalous_count
            print(f"Images processed: {image_count}")
            print(f"Normal: {normal_count}")
            print(f"Anomalous: {anomalous_count}")

    except FileNotFoundError as err:
```

```
        logger.exception("File not found.:%s", err)
        print(f"File not found: {err}. Check your input CSV file.")

if __name__ == "__main__":
    main()
```

4. 如果發生重複的影像或發生分類錯誤：

   a. 使用錯誤檔案來更新已刪除的 CSV 檔案或輸入的 CSV 檔案。

   b. 使用更新的刪除重複資料的 CSV 檔案或更新的輸入 CSV 檔案再次執行指令碼。

5. 如果您打算使用測試資料集，請重複步驟 1-4，為測試資料集建立資訊清單檔案。

6. 如有必要，請將映像從電腦複製到您在 CSV 檔案欄 1 (或在`--s3-path`命令列中指定) 中指定的 Amazon S3 儲存貯體路徑。若要複製影像，請於指令提示下輸入以下指令。

```
aws s3 cp --recursive your-local-folder/ s3://your-target-S3-location/
```

7. 依照中的指示使用清單文件（控制台）創建數據集建立資料集。如果您正在使用 AWS SDK，請參閱使用資訊清單檔案 (SDK) 建立資料集。

使用清單文件（控制台）創建數據集

下列程序說明如何透過匯入存放在 Amazon S3 儲存貯體的 SageMaker 格式資訊清單檔案來建立訓練或測試資料集。

建立資料集之後，您可以在資料集中新增更多影像，或在影像中新增標籤。如需詳細資訊，請參閱 將影像新增至資料集。

若要使用 SageMaker Ground Truth 格式資訊清單檔案 (主控台) 建立資料集

1. 建立或使用現有的 Amazon Lookout for Vision 相容 SageMaker Ground Truth 格式資訊清單檔案。如需詳細資訊，請參閱 建立清單檔案。
2. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
3. 在 Amazon S3 儲存貯體中，建立一個資料夾來存放資訊清單檔案。
4. 將您的資訊清單檔案上傳到您剛建立的資料夾。
5. 在 Amazon S3 儲存貯體中，建立用於存放映像的資料夾。
6. 將影像上傳至您剛建立的資料夾。

> **Important**
>
> 每個 JSON 行中的`source-ref`欄位值必須對應至資料夾中的影像。

7. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
8. 選擇開始使用。
9. 在左側導覽視窗中，選擇專案。
10. 選擇您要新增與資訊清單檔案搭配使用的專案。
11. 在 [運作方式] 區段中，選擇 [建立資料集]。
12. 選擇 [單一資料集] 索引標籤或 [個別訓練與測試資料集] 索引標籤，然後依照步驟

    Single dataset

    1. 選擇 [建立單一資料集]。
    2. 在 [影像來源設定] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像]。
    3. 對於 .manifest 文件位置，請輸入您的清單文件的位置。

    Separate training and test datasets

    1. 選擇 [建立訓練資料集和測試資料集]。
    2. 在 [訓練資料集詳細資料] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像
    3. 在 .manifest 檔案位置中，輸入訓練資訊清單檔案的位置。
    4. 在 [測試資料集詳細資料] 區段中，選擇 [匯入由 SageMaker Ground Truth 標記的影像
    5. 在 .manifest 文件位置中，輸入測試清單文件的位置。

    > **Note**
    >
    > 您的訓練和測試資料集可以有不同的影像來源。

13. 選擇提交。
14. 請遵循 培訓您的模型 中的步驟訓練模型。

亞馬遜 Lookout for Vision 在 Amazon S3 存儲桶`datasets`文件夾中創建一個數據集。您的原始`.manifest`檔案會保持不變。

使用資訊清單檔案 (SDK) 建立資料集

您可以使用此CreateDataset操作來建立與視覺專案的 Amazon 瞭望台相關聯的資料集。

如果您想要使用單一資料集進行訓練和測試，請建立將DatasetType值設定為的單一資料集train。在訓練期間，資料集會在內部分割，以製作訓練和測試資料集。您無法存取分割訓練和測試資料集。如果您想要個別的測試資料集，請使CreateDataset用設定DatasetType值進行第二個呼叫test。在訓練期間，訓練和測試資料集會用來訓練和測試模型。

您可以選擇性地使用DatasetSource參數來指定用於填入資料集的 SageMaker Ground Truth 格式資訊清單檔案的位置。在這種情況下，呼叫CreateDataset是非同步的。若要檢查目前狀態，請呼叫 DescribeDataset。如需詳細資訊，請參閱 檢視資料集。如果在匯入期間發生驗證錯誤，則會將的值設定Status為 CREATE_FAILED，並設定狀態訊息 (StatusMessage)。

> **Tip**
>
> 如果您要使用入門範例資料集建立資料集，請使用指令碼在其中建立的 manifest 檔案 (getting-started/dataset-files/manifests/train.manifest) 步驟 1：建立清單檔案並上傳圖片。
> 如果您要使用電路板範例影像建立資料集，您有兩種選擇：
>
> 1. 使用代碼創建清單文件。Amazon 觀察視覺實驗室 Python 筆記本顯示瞭如何為電路板示例圖像創建清單文件。或者，使用程式碼範例儲存庫中的資料集範例 AWS 程式碼。
> 2. 如果您已經使用 Amazon Lookout for Vision 主控台建立包含電路板範例映像的資料集，請重複使用 Amazon Lookout for Vision 您建立的資訊清單檔案。訓練和測試資訊清單檔案位置為s3://*bucket*/datasets/*project name*/*train or test*/manifests/output/output.manifest。

如果未指定DatasetSource，則會建立空白資料集。在這種情況下，呼叫CreateDataset是同步的。稍後，您可以呼叫「UpdateDataset項目」，將影像標示為資料集。如需範例程式碼，請參閱 新增更多影像 (SDK)。

如果您想要取代資料集，請先刪除現有資料集，DeleteDataset然後透過呼叫建立相同資料集類型的新資料集CreateDataset。如需詳細資訊，請參閱 刪除資料集。

建立資料集之後，您可以建立模型。如需詳細資訊，請參閱 培訓模型 (SDK)。

您可以呼叫「ListDataset項目」，檢視資料集中已標示的影像 (JSON 行)。您可以通過調用添加標籤的圖像UpdateDatasetEntries。

若要檢視測試和訓練資料集的相關資訊，請參閱檢視資料集。

若要建立資料集 (SDK)

1. 如果您尚未這樣做，請安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼建立資料集。

   CLI

   變更下列值：

   - `project-name`到您要與資料集建立關聯的專案名稱。
   - `dataset-type`到您要創建的數據集類型（`train`或`test`）。
   - `dataset-source`到資訊清單檔案的 Amazon S3 位置。
   - `Bucket`到包含資訊清單檔案的 Amazon S3 儲存貯體的名稱。
   - `Key`到 Amazon S3 儲存貯體中資訊清單檔案的路徑和檔案名稱。

```
aws lookoutvision create-dataset --project-name project\
  --dataset-type train or test\
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Key": "manifest file" } } }' \
  --profile lookoutvision-access
```

   Python

   此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
 dataset_type):
        """
        Creates a new Lookout for Vision dataset

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                             create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
```

```
        :param dataset_type: The type of the dataset (train or test).
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key":
 key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s", response["DatasetMetadata"]
["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
            logger.info("Dataset Type: %s", response["DatasetMetadata"]
["DatasetType"])

            # Wait until either created or failed.
            finished = False
            status = ""
            dataset_description = {}
            while finished is False:
                dataset_description = lookoutvision_client.describe_dataset(
                    ProjectName=project_name, DatasetType=dataset_type
                )
                status = dataset_description["DatasetDescription"]["Status"]

                if status == "CREATE_IN_PROGRESS":
                    logger.info("Dataset creation in progress...")
                    time.sleep(2)
                elif status == "CREATE_COMPLETE":
                    logger.info("Dataset created.")
                    finished = True
                else:
                    logger.info(
                        "Dataset creation failed: %s",
                        dataset_description["DatasetDescription"]
["StatusMessage"],
```

```
                    )
                    finished = True

            if status != "CREATE_COMPLETE":
                message = dataset_description["DatasetDescription"]
["StatusMessage"]
                logger.exception("Couldn't create dataset: %s", message)
                raise Exception(f"Couldn't create dataset: {message}")

        except ClientError:
            logger.exception("Service error: Couldn't create dataset.")
            raise
```

Java V2

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to create a
 *                     dataset.
 * @param datasetType  The type of dataset that you want to create (train or
 *                     test).
 * @param bucket       The S3 bucket that contains the manifest file.
 * @param manifestFile The name and location of the manifest file within the S3
 *                     bucket.
 * @return DatasetDescription The description of the created dataset.
 */
public static DatasetDescription createDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType,
                String bucket,
                String manifestFile)
                throws LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Creating {0} dataset for project {1}",
                        new Object[] { projectName, datasetType });
```

```
        // Build the request. If no bucket supplied, setup for empty dataset
creation.
        CreateDatasetRequest createDatasetRequest = null;

        if (bucket != null && manifestFile != null) {

                InputS3Object s3Object = InputS3Object.builder()
                                .bucket(bucket)
                                .key(manifestFile)
                                .build();

                DatasetGroundTruthManifest groundTruthManifest =
DatasetGroundTruthManifest.builder()
                                .s3Object(s3Object)
                                .build();

                DatasetSource datasetSource = DatasetSource.builder()
                                .groundTruthManifest(groundTruthManifest)
                                .build();

                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .datasetSource(datasetSource)
                                .build();
        } else {
                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
        }

        lfvClient.createDataset(createDatasetRequest);

        DatasetDescription datasetDescription = null;

        boolean finished = false;

        // Wait until dataset is created, or failure occurs.
        while (!finished) {

                datasetDescription = describeDataset(lfvClient, projectName,
datasetType);
```

```
                switch (datasetDescription.status()) {
                        case CREATE_COMPLETE:
                                logger.log(Level.INFO, "{0}dataset created for
project {1}",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;
                        case CREATE_IN_PROGRESS:
                                logger.log(Level.INFO, "{0} dataset creating for
project {1}",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case CREATE_FAILED:
                                logger.log(Level.SEVERE,
                                                "{0} dataset creation failed for
project {1}. Error {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;
                        default:
                                logger.log(Level.SEVERE, "{0} error when
creating {1} dataset for project {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;

                }
        }

        logger.log(Level.INFO, "Dataset info. Status: {0}\n Message: {1} }",
                        new Object[] { datasetDescription.statusAsString(),
                                        datasetDescription.statusMessage() });
```

```
        return datasetDescription;

}
```

3. 依照中的步驟訓練您的模型培訓模型 (SDK)。

標記檔案

您可以使用 Amazon Lookout for Vision 主控台新增或修改指派給資料集中映像的標籤。如果您使用的是 SDK，則標籤是您提供的清單文件的一部分CreateDataset。您可以呼叫「UpdateDataset項目」來更新影像的標籤。如需範例程式碼，請參閱 新增更多影像 (SDK)。

選擇模型類型

您指派給影像的標籤決定 Lookout for Vision 所建立的模型類型。如果您的專案有個別的測試資料集，請以相同的方式標記影像。

圖像分類模型

若要建立影像分類模型，請將每個影像分類為正常或異常。對於每個圖像，請執行分類映像（控制台）。

圖像分割模型

若要建立影像分割模型，請將每個影像分類為正常或異常。對於每個異常影像，您也可以為影像上的每個異常區域指定像素遮色片，並指定像素遮色片中異常類型的異常標籤。例如，下圖中的藍色遮色片會標記車輛上刮痕異常類型的位置。您可以在影像中指定一種以上的異常標籤類型。對於每個圖像，請執行分割圖像（控制台）。

分類映像（控制台）

您可以使 Lookout for Vision 主控台，將資料集中的影像分類為正常或異常狀況。未分類的圖像不會用於訓練您的模型。

如果您要建立影像分割模型，請略過此程序並執行分割圖像（控制台），其中包括分類影像的步驟。

> **Note**
>
> 如果您剛完成建立資料集，主控台目前應該會顯示您的模型儀表板，而您不需要執行步驟 1-4。

若要分類映像檔 (主控台)

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 在左側導覽視窗中，選擇專案。
3. 在所有專案頁面上，選擇您要使用的專案。
4. 在專案的左側導覽窗格中，選擇 [資料集]。
5. 如果您有個別的訓練和測試資料集，請為您要使用的資料集選擇索引標籤。
6. 選擇 [開始標記]。
7. 選擇 [選取此頁面上的所有影像]。
8. 如果影像正常，請選擇「正常分類」，否則選擇「分類為異常」。每張圖片下方都會出現一個標籤。
9. 如果您需要變更影像的標籤，請執行下列動作：
   a. 選擇圖像下的「異常」或「正常」。
   b. 如果無法判斷影像的正確標籤，請在圖庫中選擇影像來放大影像。

   > **Note**
   >
   > 您可以在「篩選器」區段中選擇所需的標籤或標籤狀態來篩選影像標籤。

10. 視需要在每頁重複步驟 7-9，直到資料集中的所有影像都已正確標記為止。
11. 選擇儲存變更。

12. 如果您已完成圖像標籤，則可以訓練您的模型。

分割圖像（控制台）

如果您要建立影像分割模型，則必須將影像分類為正常或異常狀況。您也必須將區段資訊新增至異常影像。若要指定區段資訊，您必須先為您希望模型尋找的每種異常類型 (例如凹痕或刮痕) 指定異常標籤。然後，您可以為資料集中異常影像的每個異常指定異常遮色片和異常標籤。

> **Note**
>
> 如果您要建立影像分類模型，則不需要區段影像，也不需要指定異常標籤。

主題

- 指定異常標籤
- 為影像加上標籤
- 使用註解工具分割影像

指定異常標籤

您可以為資料集影像中的每種異常類型定義異常標籤。

指定異常標籤

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 在左側導覽視窗中，選擇專案。
3. 在所有專案頁面上，選擇您要使用的專案。
4. 在專案的左側導覽窗格中，選擇 [資料集]。
5. 在「異常標籤」中，選擇「新增異常標籤」。如果您之前已新增異常標籤，請選擇 [管理]。
6. 在 對話方塊中，執行下列動作：
   a. 輸入您要新增的異常標籤，然後選擇 [新增異常標籤]。
   b. 重複上一個步驟，直到輸入您希望模型尋找的每個異常標籤為止。
   c. (選擇性) 選擇編輯圖示以變更標籤名稱。
   d. (選擇性) 選擇刪除圖示以刪除新的異常標籤。您無法刪除資料集目前使用的異常類型。

7. 選擇 [確認]，將新的異常標籤新增至資料集。

指定異常標籤後，請執行以下操作為影像加上標籤標記圖像。

為影像加上標籤

若要標示影像以進行影像分割，請將影像分類為正常或異常。然後，使用註解工具繪製遮色片，以緊密覆蓋影像中每種異常類型的區域來劃分影像。

標示影像的步驟

1. 如果您有個別的訓練和測試資料集，請為您要使用的資料集選擇索引標籤。
2. 如果您尚未指定資料集的異常類型，請執指定異常標籤行下列動作來指定資料集的異常類型。
3. 選擇 [開始標記]。
4. 選擇 [選取此頁面上的所有影像]。
5. 如果影像正常，請選擇「正常分類」，否則選擇「分類為異常」。
6. 若要變更單一影像的標籤，請在影像下選擇「正常」或「異常」。

> **Note**
>
> 您可以在「篩選器」區段中選擇所需的標籤或標籤狀態來篩選影像標籤。您可以在 [排序選項] 區段中依信賴度分數排序。

7. 針對每個異常影像，選擇要開啟註解工具的影像。通過執行添加細分信息使用註解工具分割影像。
8. 選擇儲存變更。
9. 如果您已完成圖像標籤，則可以訓練您的模型。

使用註解工具分割影像

您可以使用註解工具，透過使用遮色片標記異常區域來劃分影像。

使用註解工具分割影像

1. 在資料集庫中選取影像，以開啟註解工具。如有必要，請選擇「開始標籤」以進入標籤模式。
2. 在「異常標籤」區段中，選擇您要標記的異常標籤。如有必要，請選擇「新增異常標籤」以新增異常標籤。

3. 選擇頁面底部的繪圖工具，然後繪製遮罩，緊密覆蓋異常標籤的異常區域。下列影像是緊密覆蓋異常的遮色片範例。

以下是不能緊密覆蓋異常狀況的不良遮罩範例。

4. 如果您要分割更多影像，請選擇「下一步」，然後重複步驟 2 和 3。
5. 選擇「提交」並關閉以完成分段影像。

培訓您的模型

建立資料集並標記影像之後，您就可以訓練模型。在訓練過程中，會使用測試資料集。如果您有單一資料集專案，資料集中的影像會自動分割為測試資料集和訓練資料集，做為訓練程序的一部分。如果您的專案有訓練和測試資料集，則會使用這些資料集來分別訓練和測試資料集。

訓練完成後，您可以評估模型的效能並進行任何必要的改進。如需詳細資訊，請參閱 改善您的 Amazon Lookout for Vision 模型。

為了訓練您的模型，Amazon Lookout 視覺會製作一份來源訓練和測試影像的副本。根據預設，複製的映像會使用 AWS 擁有和管理的金鑰加密。您也可以選擇使用自己的 AWS Key Management Service (KMS) (KMS) 金鑰。如需更多詳細資訊，請參閱 AWS 金鑰管理服務概念。您的來源圖像不會受到影響。

您可以以標籤的形式將中繼資料指派給模型。如需詳細資訊，請參閱 標記模型。

每次訓練模型時，都會建立模型的新版本。如果您不再需要模型的版本，可以將其刪除。如需詳細資訊，請參閱 刪除模型。

您需支付成功訓練模型所花費的時間費用。如需詳細資訊，請參閱訓練時數。

若要檢視專案中的現有模型，請執行檢視您的模型。

> **Note**
>
> 如果您剛剛完成建立資料集或將影像新增至資料集. 主控台目前應該會顯示您的模型儀表板，而您不需要執行步驟 1 至 4。

主題

- 訓練模型（控制台）
- 培訓模型 (SDK)

訓練模型（控制台）

下列程序說明如何使用主控台訓練模型。

培訓您的模型（主控台）

1. 打開 Amazon Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.

2. 在左側導覽視窗中，選擇 專案。
3. 在 專案 頁面，選擇包含要培訓的模型的專案。
4. 在專案詳細資訊頁面上，選擇 [訓練模型]。如果您有足夠的標籤影像來訓練模型，則可以使用「訓練模型」按鈕。如果該按鈕不可用，請添加更多圖像，直到您有足夠的標記圖像。
5. (可選) 如果您想要使用自己的 AWS KMS 加密金鑰，請執行以下操作：
   a. 在 圖像資料加密 中選擇 自訂加密配置 (進階)。
   b. 在 encryption.aws_kms_key，輸入您的金鑰的 Amazon Resource Name (ARN)，或選擇現有的 AWS KMS key。若要建立新的金鑰，請選擇 建立 AWS IMS 金鑰。
6. (可選) 如果您要新增標籤到模型，請執行以下操作：
   a. 在 標籤 區域，選擇 新增。
   b. 輸入下列資料：
      i. 金鑰 中的金鑰名稱。
      ii. 值 中的鍵/值。
   c. 若要新增更多標籤，請重複步驟 6a 和 6b。
   d. (選用) 如果您要移除標籤，請選擇要刪除的標籤旁邊的刪除。如果您要移除先前儲存的標籤，則會在您儲存變更時移除該標籤。
7. 選擇訓練模型。
8. 在您是否要訓練模型？的對話框中，選擇訓練模型。
9. 在「模型」(Model) 檢視中，您可以看到訓練已經開始，您可以檢視模型版本的`Status`欄來檢查目前的狀態。培訓模型需要一段時間才能完成。
10. 訓練完成後，您可以評估其效能。如需詳細資訊，請參閱 改善您的 Amazon Lookout for Vision 模型。

培訓模型 (SDK)

您可以使用此CreateModel作業來開始模型的訓練、測試和評估。Amazon Lookout for Vision 使用與該項目關聯的培訓和測試數據集來訓練模型。如需詳細資訊，請參閱 建立專案 (SDK)。

每次呼叫時`CreateModel`，都會建立模型的新版本。來自的回應`CreateModel`包括模型的版本。

每次成功的模型訓練都會向您收取費用。使用 `ClientToken` input 參數可協助避免因使用者不必要或意外重複模型訓練而產生的費用。 `ClientToken`是冪等輸入參數，可確保`CreateModel`僅針對特定參數集完成一次 — 具有相同`ClientToken`值`CreateModel`的重複呼叫可確保不會重複訓練。

如果您沒有提供值ClientToken，您正在使用的 AWS 開發套件會為您插入一個值。這樣可以防止網路錯誤後重試啟動多個訓練工作，但您需要為自己的使用案例提供自己的價值。如需詳細資訊，請參閱CreateModel。

培訓需要一段時間才能完成。要檢查當前狀態，請調用DescribeModel並傳遞項目名稱（在調用對象中指定CreateProject）和模型版本。此status欄位會指出模型訓練的目前狀態。如需範例程式碼，請參閱 檢視您的模型 (SDK)。

如果訓練成功，您可以評估模型。如需詳細資訊，請參閱 改善您的 Amazon Lookout for Vision 模型。

若要檢視您在專案中建立的模型，請呼叫ListModels。如需範例程式碼，請參閱 檢視您的模型。

培訓模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定 AWS CLI 和 AWS SDK。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來訓練模型。

   CLI

   變更下列值：

   - project-name至包含您要建立之模型的專案名稱。
   - output-config到您要儲存訓練結果的位置。取代以下的值：
     - output bucket與 Amazon S3 存儲桶的名稱，其中 Amazon Lookout for Vision 保存培訓結果。
     - output folder以您要儲存訓練結果的資料夾名稱。
     - Key帶有標籤鍵的名稱。
     - Value與要關聯的值tag_key。

   ```
   aws lookoutvision create-model --project-name "project name"\
     --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":
    "output folder" } }'\
     --tags '[{"Key":"Key","Value":"Value"}]' \
     --profile lookoutvision-access
   ```

   Python

   此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def create_model(
        lookoutvision_client,
        project_name,
        training_results,
        tag_key=None,
        tag_key_value=None,
    ):
        """
        Creates a version of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project in which you want to create
 a
                             model.
        :param training_results: The Amazon S3 location where training results
 are stored.
        :param tag_key: The key for a tag to add to the model.
        :param tag_key_value - A value associated with the tag_key.
        return: The model status and version.
        """
        try:
            logger.info("Training model...")
            output_bucket, output_folder = training_results.replace("s3://",
 "").split(
                "/", 1
            )
            output_config = {
                "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}
            }
            tags = []
            if tag_key is not None:
                tags = [{"Key": tag_key, "Value": tag_key_value}]

            response = lookoutvision_client.create_model(
                ProjectName=project_name, OutputConfig=output_config, Tags=tags
            )

            logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
            logger.info("Version: %s", response["ModelMetadata"]
["ModelVersion"])
            logger.info("Started training...")
```

```
            print("Training started. Training might take several hours to
 complete.")

            # Wait until training completes.
            finished = False
            status = "UNKNOWN"
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name,
                    ModelVersion=response["ModelMetadata"]["ModelVersion"],
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "TRAINING":
                    logger.info("Model training in progress...")
                    time.sleep(600)
                    continue

                if status == "TRAINED":
                    logger.info("Model was successfully trained.")
                else:
                    logger.info(
                        "Model training failed: %s ",
                        model_description["ModelDescription"]["StatusMessage"],
                    )
                finished = True
        except ClientError:
            logger.exception("Couldn't train model.")
            raise
        else:
            return status, response["ModelMetadata"]["ModelVersion"]
```

Java V2

此代碼取自 AWS 文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Creates an Amazon Lookout for Vision model. The function returns after model
 * training completes. Model training can take multiple hours to complete.
 * You are charged for the amount of time it takes to successfully train a
 model.
 * Returns after Lookout for Vision creates the dataset.
```

```
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @param projectName The name of the project in which you want to create a
     *                    model.
     * @param description A description for the model.
     * @param bucket      The S3 bucket in which Lookout for Vision stores the
     *                    training results.
     * @param folder      The location of the training results within the S3
     *                    bucket.
     * @return ModelDescription The description of the created model.
     */
    public static ModelDescription createModel(LookoutVisionClient lfvClient, String
     projectName,
                    String description, String bucket, String folder)
                    throws LookoutVisionException, InterruptedException {

            logger.log(Level.INFO, "Creating model for project: {0}.", new Object[]
     { projectName });

            // Setup input parameters.
            S3Location s3Location = S3Location.builder()
                            .bucket(bucket)
                            .prefix(folder)
                            .build();

            OutputConfig config = OutputConfig.builder()
                            .s3Location(s3Location)
                            .build();

            CreateModelRequest createModelRequest = CreateModelRequest.builder()
                            .projectName(projectName)
                            .description(description)
                            .outputConfig(config)
                            .build();

            // Create and train the model.
            CreateModelResponse response =
     lfvClient.createModel(createModelRequest);

            String modelVersion = response.modelMetadata().modelVersion();
            boolean finished = false;
            DescribeModelResponse descriptionResponse = null;

            // Wait until training finishes or fails.
```

```
        do {
                DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                                .projectName(projectName)
                                .modelVersion(modelVersion)
                                .build();

                descriptionResponse =
lfvClient.describeModel(describeModelRequest);

                switch (descriptionResponse.modelDescription().status()) {
                        case TRAINED:
                                logger.log(Level.INFO, "Model training completed
for project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        case TRAINING:
                                logger.log(Level.INFO,
                                                "Model training in progress for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                TimeUnit.SECONDS.sleep(60);

                                break;

                        case TRAINING_FAILED:
                                logger.log(Level.SEVERE,
                                                "Model training failed for for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when training
model project {0} version {1}: {2}.",
```

```
                                                    new Object[] { projectName,
modelVersion,

descriptionResponse.modelDescription()

.status() });
                                finished = true;
                                break;

                }
        } while (!finished);

        return descriptionResponse.modelDescription();

}
```

3. 訓練完成後，您可以評估其效能。如需詳細資訊，請參閱 改善您的 Amazon Lookout for Vision 模型。

模型訓練疑難排

資訊清單檔案或訓練影像的問題可能會導致模型訓練失敗。在重新訓練模型之前，請檢查下列潛在問題。

異常標籤顏色與遮色片影像中異常的顏色不相符

如果您正在訓練影像分割模型，資訊清單檔案中異常標籤的顏色必須與遮色片影像中的顏色相符。資訊清單檔案中影像的 JSON 行具有中繼資料 (internal-color-map)，該中繼資料 () 會告知 Amazon Lookout 視覺哪種顏色對應於異常標籤。例如，下列 JSON 行中scratch異常標籤的顏色為。#2ca02c

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
```

```
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

如果遮色片影像中的顏色與中的值不符hex-color，訓練會失敗，您需要更新資訊清單檔案。

更新資訊清單檔案中的顏色值

1. 使用文字編輯器，開啟您用來建立資料集的資訊清單檔案。
2. 對於每個 JSON 行 (影像)，請檢查internal-color-map欄位中的顏色 (hex-color) 是否與遮色片影像中異常標籤的顏色相符。

   您可以從現場獲取蒙版圖像的*anomaly-mask*-ref位置。將圖像下載到您的計算機，然後使用以下代碼獲取圖像中的顏色。

   ```
   from PIL import Image
   ```

```
img = Image.open('path to local copy of mask file')
colors = img.convert('RGB').getcolors() #this converts the mode to RGB
for color in colors:
    print('#%02x%02x%02x' % color[1])
```

3. 對於每個具有不正確色彩指派的影像，請更新影像 JSON 行中的hex-color欄位。
4. 儲存更新資訊清單檔案。
5. 從專案中刪除現有的資料集。
6. 使用更新的資訊清單檔案在專案中建立新資料集。
7. 訓練模型。

或者，對於步驟 5 和 6，您可以呼叫「UpdateDataset項目」作業，並為您要更新的影像提供更新的 JSON 行，來更新資料集中的個別影像。如需範例程式碼，請參閱 新增更多影像 (SDK)。

遮色片影像不是 PNG 格式

如果您要訓練影像分割模型，遮色片影像必須是 PNG 格式。如果您從資訊清單檔案建立資料集，請確定您參考的遮罩影像*anomaly-mask*-ref為 PNG 格式。如果遮罩圖像不是 PNG 格式，則需要將它們轉換為 PNG 格式。將影像檔案的副檔名重新命名為是不夠的.png。

您在 Amazon Lookout for Vision 主控台或使用「 SageMaker Ground Truth」任務建立的遮罩影像會以 PNG 格式建立。您不需要變更這些影像的格式。

更正資訊清單檔案中的非 PNG 格式遮罩影像

1. 使用文字編輯器，開啟您用來建立資料集的資訊清單檔案。
2. 對於每個 JSON 行（圖像），請確保圖像*anomaly-mask*-ref引用了 PNG 格式的圖像。如需詳細資訊，請參閱 建立清單檔案。
3. 儲存更新的資訊清單檔案。
4. 從專案中刪除現有的資料集。
5. 使用更新的資訊清單檔案在專案中建立新資料集。
6. 訓練模型。

分段或分類標籤不正確或遺失

缺少或不正確的標籤可能會導致訓練失敗或建立效能不佳的模型。建議您為資料集中的所有影像加上標籤。如果您未標記所有影像且模型訓練失敗，或者您的模型效能不佳，請新增更多影像。

請檢查以下內容：

- 如果您要建立區段模型，遮罩必須緊密覆蓋資料集影像上的異常狀況。若要檢查資料集中的遮罩，請檢視專案資料集圖庫中的影像。如有必要，請重繪影像遮色片。如需詳細資訊，請參閱 分割圖像（控制台）。

- 請確定資料集影像中的異常影像已分類。如果您要建立影像分割模型，請確定異常影像具有異常標籤和影像遮色片。

  請務必記住您要建立的模型類型 (區段或分類)。分類模型不需要異常影像上的影像遮色片。請勿將遮罩新增至用於分類模型的資料集影像。

  更新遺失標示的步驟

  1. 開啟專案的資料集庫。
  2. 過濾未標記的圖像以查看哪些圖像沒有標籤。
  3. 執行以下任意一項：
     - 如果您要建立影像分類模型，請將每個未標籤的影像分類。
     - 如果您要建立影像分割模型，請對每個未標籤的影像進行分類和區段。
  4. 如果您要建立影像分割模型，請在遺失遮色片的任何已分類異常影像中新增遮色片。
  5. 訓練模型。

如果您選擇不修正不良或遺失的標籤，建議您新增更多已標記的影像，或從資料集中移除受影響的影像。您可以從主控台或使用「項目」作業新增更多UpdateDataset項目。如需詳細資訊，請參閱 將影像新增至資料集。

如果您選擇移除影像，您必須重新建立沒有受影響影像的資料集，因為您無法從資料集中刪除影像。如需更多詳細資訊，請參閱 從資料集中移除影像。

改善您的 Amazon Lookout for Vision 模型

訓練期間 Lookout to Vision 會使用測試資料集測試您的模型，並使用結果建立效能指標。您可以使用效能指標來評估模型的效能。如有必要，您可以採取步驟來改善資料集，然後重新訓練模型。

如果對模型的效能滿意，則可以開始使用模型 如需詳細資訊，請參閱運行訓練有素的亞馬遜 Lookout for Vision 模型。

主題

步驟 1：評估模型的效能

您可以從主控台和DescribeModel作業存取效能指標。Amazon Lookout for Vision 版提供測試資料集的摘要效能指標，以及所有個別映像的預測結果。如果您的模型是區段模型，主控台也會顯示每個異常標籤的摘要指標。

若要在主控台中檢視效能指標和測試影像預測，請參閱檢視效能指標 (主控台)。如需使用`DescribeModel`作業存取效能指標和測試影像預測的相關資訊，請參閱檢視效能指標。

影像分類指標

Amazon Lookout for Vision 模型在測試期間進行的分類提供下列摘要指標：

- 精確度
- 取回
- F1 比分

影像分割模型指標

如果模型是影像分段模型，Amazon Lookout for Vision 每個異常標籤提供摘要影像分類指標和摘要效能指標：

- F1 比分
- 聯集上的平均交點 (IoU)

精確度

精確度量回答了這個問題 — 當模型預測圖像包含異常時，該預測多久是正確的？

對於誤報的成本很高的情況，精確度是一個有用的指標。例如，從組裝的機器中移除沒有瑕疵的機器零件的成本。

Amazon Lookout for Vision 察提供整個測試資料集的摘要精確度指標值。

精確度是在所有預測的異常（真正和誤報）上正確預測異常（真正正值）的分數。精確度的公式如下。

精度值 = 真正值/(真正 + 誤報)

精確度的可能值範圍為 0 到 1。亞馬遜 Lookout for Vision 控制台以百分比值 (0—100) 的形式顯示精確度。

較高的精確度值表示更多的預測異常是正確的。例如，假設您的模型預測 100 個圖像是異常的。如果 85 個預測是正確的 (真正值) 且 15 不正確 (誤報)，則精確度的計算方式如下：

85 個真正值/(85 個真正值 + 15 個誤報) = 0.85 精確度值

但是，如果模型僅在 100 個異常預測中正確預測 40 張影像，則產生的精確度值會低於 0.40 (也就是 40/(40 + 60) = 0.40)。在這種情況下，您的模型所做的預測比正確的預測更多不正確的預測。若要修正此問題，請考慮改善您的模型。如需詳細資訊，請參閱步驟 2：改善您的模型。

如需詳細資訊，請參閱精確度和召回。

取回

召回量度回答了這個問題-在測試資料集中的異常影像總數中，有多少正確地預測為異常？

調用量度對於假負成本很高的情況非常有用。例如，不移除瑕疵零件的成本很高時。Amazon Lookout for Vision 察提供整個測試資料集的摘要召回指標值。

回想一下是正確檢測到的異常測試圖像的一小部分。它是一種衡量模型可以正確預測異常圖像的頻率，當它實際存在於測試數據集的圖像中時。召回的公式計算方式如下所示：

召回值 = 真陽性/（真陽性 + 假陰性）

召回的範圍是 0 到 1。亞馬遜 Lookout for Vision 察控制台以百分比值（0—100）顯示召回。

較高的召回值表示可正確識別更多異常影像。例如，假設測試資料集包含 100 個異常影像。如果模型正確檢測到 100 個異常圖像中的 90 個，則召回如下：

90 個真正正值/(90 個真正正值 + 10 個假陰性) = 0.90 回收值

回復值 0.90 表示您的模型正確預測測試資料集中的大部分異常影像。如果模型只能正確預測 20 個異常影像，則回收率會低於 0.20 (即 20/(20 + 80) = 0.20)。

在此情況下，您應該考慮對您的模型進行改進。如需詳細資訊，請參閱步驟 2：改善您的模型。

如需詳細資訊，請參閱精確度和召回。

F1 比分

亞馬遜 Lookout for Vision 提供測試資料集的平均模型效能分數。具體而言，異常分類的模型效能是由 F1 評分量度來測量，這是精確度和召回分數的諧波平均值。

F1 分數是考慮到精度和召回的彙總度量。模型效能分數是介於 0 到 1 之間的值。值越高，模型在調用和精確度方面的效果就越好。例如，對於精確度為 0.9 且召回 1.0 的模型，F1 的分數為 0.947。

例如，如果模型表現不佳，精確度為 0.30，高回收率為 1.0，則 F1 分數為 0.46。同樣，如果精度很高（0.95）並且召回率低（0.20），則 F1 得分為 0.33。在這兩種情況下，F1 分數都很低，表示模型出現問題。

如需詳細資訊，請參閱 F1 分數。

聯集上的平均交點 (IoU)

測試影像中的異常遮罩與模型為測試影像預測的異常遮罩之間的平均百分比重疊。Amazon Lookout for Vision 會傳回每個異常標籤的平均 IOU，且僅由影像分割模型傳回。

較低的百分比值表示模型未準確地將標籤的預測遮罩與測試影像中的遮色片相符。

以下圖像具有較低的 IOU。橙色遮罩是來自模型的預測，不會緊緊覆蓋測試圖像中代表蒙版的藍色蒙版。

以下圖像具有較高的 IoU。藍色遮罩（測試圖像）被橙色遮罩（預測的遮罩）緊密覆蓋。

測試結果

在測試期間，模型預測測試數據集中的每個測試圖像的分類。每個預測的結果將與相應測試圖像的標籤（正常或異常）進行比較，如下所示：

- 正確地預測圖像是異常的被認為是真正的積極的。
- 錯誤地預測圖像是異常的被認為是誤報。
- 正確預測圖像是正常的被認為是真正的負面。
- 錯誤地預測圖像是正常的被認為是假陰性。

如果模型是分段模型，則模型也會預測測試影像上異常位置的遮罩和異常標籤。

亞馬遜 Lookout for Vision 使用比較結果來產生效能指標。

步驟 2：改善您的模型

效能指標可能會顯示您可以改善模型。例如，如果模型未偵測到測試資料集中的所有異常，則您的模型具有較低的召回率 (也就是說，召回量度的值較低)。如果需要改進您的模型，請考慮下列事項：

- 檢查訓練和測試資料集映像檔是否已正確標記。
- 降低影像擷取條件 (例如光線和物件姿勢) 的變異性，並在相同類型的物件上訓練模型。
- 確保您的圖像只顯示所需的內容。例如，如果您的專案偵測到機器零件中的異常，請確定影像中沒有其他物件。
- 在訓練和測試資料集中新增更多標籤影像。如果您的測試數據集具有出色的召回和精確度，但模型在部署時效能不佳，則您的測試數據集可能不夠具有代表性，因此您需要擴展它。
- 如果您的測試資料集導致召回和精確度不佳，請考慮訓練和測試資料集中的異常情況和影像擷取條件的相符程度。如果您的訓練影像不代表預期的異常和條件，但測試影像中的影像是，請將影像新增至具有預期異常和條件的訓練訓練資料集。如果測試資料集映像檔不在預期的條件下，但訓練影像是，請更新測試資料集。

  如需詳細資訊，請參閱添加更多圖像。另一種將標記影像新增至訓練資料集的方法是執行試驗偵測工作並驗證結果。然後，您可以將已驗證的映像新增至訓練資料集。如需詳細資訊，請參閱使用試驗偵測任務來驗證您的模型。
- 確保您的訓練和測試資料集中具有足夠多樣化的正常和異常影像。這些影像必須代表模型將遇到的一般和異常影像類型。例如，在分析電路板時，正常的影像應代表元件 (例如電阻器和電晶體) 的位置和焊接方面的變化。異常影像應代表系統可能遇到的不同異常類型，例如放錯位置或遺失元件。
- 如果您的模型偵測到的異常類型具有較低的平均 IOU，請檢查分割模型中的遮罩輸出。對於某些使用案例 (例如刮痕)，模型可能會輸出刮痕，這些刮痕在測試影像中非常接近地面真實刮痕，但像素重疊較低。例如，兩條相距 1 個像素距離的 parallel 線。在這些情況下，平均 IOU 是衡量預測成功的不可靠指標。
- 如果影像尺寸較小或影像解析度較低，請考慮以較高的解析度拍攝影像。影像尺寸的範圍可以從 64 x 64 像素到 4096 像素 X 4096 像素之間。
- 如果異常大小很小，請考慮將影像分割成不同的拼貼，並使用拼貼影像進行訓練和測試。這可讓模型在影像中看到較大尺寸的瑕疵。

改善訓練和測試資料集之後，請重新訓練並重新評估模型。如需詳細資訊，請參閱培訓您的模型。

如果指標顯示您的模型具有可接受的效能，您可以將試用偵測工作的結果新增至測試資料集，以驗證其效能。重新訓練後，績效指標應確認先前訓練的績效指標。如需詳細資訊，請參閱使用試驗偵測任務來驗證您的模型。

檢視效能指標

您可以從控制台和調用`DescribeModel`操作來獲取性能指標。

主題

檢視效能指標 (主控台)

訓練完成後，主控台會顯示效能指標。

Amazon 觀測視覺主控台會針對測試期間進行的分類顯示下列效能指標：

- 精確度
- 取回
- F1 比分

如果模型是區段模型，則主控台也會針對每個異常標籤顯示下列效能指標：

- 發現異常標籤的測試影像數量。
- F1 比分
- 聯集上的平均交點 (IoU)

測試結果概觀部分會顯示測試資料集中影像的正確和不正確預測總數。您也可以查看測試資料集中個別映像的預測和實際標籤指派。

下列程序說明如何從專案的模型清單檢視中，取得效能指標。

檢視效能指標 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 Project。
4. 在專案檢視中，選擇專案，內含您要檢視的模型版本。
5. 在左側導覽窗格中，選擇專案名稱下的 Mo dels。
6. 在模型清單檢視中，選擇您要檢視的模型版本。
7. 在模型詳細資訊頁面上，檢視「效能測量結果」頁籤上的效能測量結果。
8. 注意下列事項：

a. 「模型效能指標」區段包含模型為測試影像所建立之分類預測的整體模型度量 (精確度、召回、F1 分數)。

b. 如果模型是影像分割模型，則「每個標籤效能」區段會包含發現異常標籤的測試影像數量。您也會看到每個異常標籤的量度 (F1 分數、平均 IoU)。

c. [測試結果概觀] 區段提供瞭望視覺用來評估模型的每個測試影像的結果。其包括以下內容：

- 所有測試影像的正確（真正）和不正確（假陰性）分類預測（正常或異常）的總數。
- 每個測試圖像的分類預測。如果您在影像下看到「校正」，則預測的分類會符合影像的實際分類。否則，模型沒有正確分類圖像。
- 使用影像分割模型時，您會看到模型指派給影像的異常標籤，以及影像上符合異常標籤顏色的遮色片。

檢視效能指標

您可以使用此DescribeModel作業取得模型的摘要效能指標 (分類)、評估資訊清單，以及模型的評估結果。

取得摘要效能測量結果

模型在測試期間所做的分類預測摘要效能指標 (精確度取回、和F1 比分) 會在呼叫傳回的`Performance`欄位中傳回`DescribeModel`。

```
        "Performance": {
            "F1Score": 0.8,
            "Recall": 0.8,
            "Precision": 0.9
        },
```

`Performance`欄位不包含區段模型傳回的異常標籤效能指標。你可以從`EvaluationResult`現場得到他們。如需詳細資訊，請參閱複查評估結果。

如需摘要效能測量結果的資訊，請參閱步驟 1：評估模型的效能 如需範例程式碼，請參閱檢視您的模型 (SDK)。

使用評估資訊清單

評估資訊清單為用於測試模型的個別映像提供測試預測指標。對於測試數據集中的每個圖像，JSON 行包含原始測試（地面真相）信息和模型對圖像的預測。Amazon Lookout for Vision 將評估資訊清單存

放在 Amazon S3 儲存貯體中。您可以從DescribeModel操作響應中的EvaluationManifest字段中獲取位置。

```
"EvaluationManifest": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationManifest-my-sdk-
project-1.json"
        }
```

檔案名稱格式為EvaluationManifest-*project name*.json。如需範例程式碼，請參閱檢視您的模型。

在下面的示例 JSON 行中，class-name是圖像內容的基本真相。在此範例中，影像包含異常。該confidence字段顯示了亞馬遜 Lookout for Vision 在預測中的信心。

```
{
    "source-ref"*: "s3://customerbucket/path/to/image.jpg",
    "source-ref-metadata": {
        "creation-date": "2020-05-22T21:33:37.201882"
    },

    // Test dataset ground truth
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "type": "groundtruth/image-classification",
        "human-annotated": "yes",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "labeling-job/anomaly-detection"
    },
    // Anomaly label detected by Lookout for Vision
    "anomaly-label-detected": 1,
    "anomaly-label-detected-metadata": {
        "class-name": "anomaly",
        "confidence": 0.9,
        "type": "groundtruth/image-classification",
        "human-annotated": "no",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "training-job/anomaly-detection",
        "model-arn": "lookoutvision-some-model-arn",
        "project-name": "lookoutvision-some-project-name",
```

```
        "model-version": "lookoutvision-some-model-version"
    }
}
```

複查評估結果

評估結果具有整組測試影像的下列彙總效能指標 (分類)：

- 精確度
- 取回
- ROC 曲線（未顯示在控制台中）
- 平均精度（未顯示在控制台中）
- F1 比分

評估結果還包括用於訓練和測試模型的影像數量。

如果模型是區段模型，評估結果也會針對測試資料集中找到的每個異常標籤包含下列指標：

- 精確度
- 取回
- F1 比分
- 聯集上的平均交點 (IoU)

Amazon Lookout for Vision 將評估結果存放在 Amazon S3 儲存貯體中。您可以透過檢查來自作業的回應`EvaluationResult`中的值來`DescribeModel`取得位置。

```
"EvaluationResult": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationResult-my-sdk-project-1.json"
        }
```

檔案名稱格式為`EvaluationResult-`*`project name`*`.json`。如需範例，請參閱 檢視您的模型。

下列結構描述顯示評估結果。

```
    {
```

```
       "Version": 1,
       "EvaluationDetails":
       {
           "ModelArn": "string", // The Amazon Resource Name (ARN) of the model
version.
           "EvaluationEndTimestamp": "string",   // The UTC date and time that
evaluation finished.
           "NumberOfTrainingImages": int,        // The number of images that were
successfully used for training.
           "NumberOfTestingImages": int          // The number of images that were
successfully used for testing.
       },
       "AggregatedEvaluationResults":
       {
           "Metrics":
           {                       //Classification metrics.
               "ROCAUC": float,            // ROC area under the curve.
               "AveragePrecision": float,  // The average precision of the model.
               "Precision": float,         // The overall precision of the model.
               "Recall": float,            // The overall recall of the model.
               "F1Score": float,           // The overal F1 score for the model.

               "PixelAnomalyClassMetrics":     //Segmentation metrics.
               [
                   {
                       "Precision": float,       // The precision for the anomaly
label.
                       "Recall": float,          // The recall for the anomaly label.
                       "F1Score": float,         // The F1 score for the anomaly
label.
                       "AIOU" : float,           // The average Intersection Over
Union for the anomaly label.
                       "ClassName": "string"   // The anomaly label.
                   }
               ]
           }
       }
   }
```

使用試驗偵測任務來驗證您的模型

如果您要驗證或改善模型的品質，您可以執行試驗偵測工作。試驗偵測任務會偵測您提供的新映像中的異常情況。

您可以驗證偵測結果，並將已驗證的影像新增至資料集。如果您有個別的訓練和測試資料集，則已驗證的影像會新增至訓練資料集。

您可以從您的 Amazon S3 儲存貯體中，驗證本機電腦或 Amazon S3 儲存貯體中的映像。如果您想要將經過驗證的映像新增至資料集，位於 S3 儲存貯體中的映像檔必須與資料集中的映像位於相同的 S3 儲存貯體中。

> **Note**
>
> 若要執行試用偵測任務，請確保您的 S3 儲存貯體已啟用版本控制。如需詳細資訊，請參閱使用版本控制。在啟用版本控制的情況下建立主控台值區。

根據預設，您的映像會使用 AWS 擁有和管理的金鑰加密。您還可以選擇使用自己的 AWS Key Management Service (KMS) 金鑰。如需詳細資訊，請參閱 AWS Key Management Service 概念。

主題

執行試驗偵測工作

執行下列步驟以執行試用偵測工作。

執行試用偵測 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 Project。
4. 在專案檢視中，選擇專案，內含您要檢視的模型版本。
5. 在左側導覽窗格中，選擇試用偵測。
6. 在試驗偵測檢視中，選擇 [執行試驗偵測]。
7. 在 [執行試驗偵測] 頁面上，於 [工作名稱] 中輸入試用偵測工作的名稱。
8. 在 「選擇模型」中，選擇您要使用的模型版本。
9. 根據圖像的來源導入圖像，如下所示：

- 如果您要從 Amazon S3 儲存貯體匯入來源映像檔，請輸入 S3 URI。

  > **Tip**
  >
  > 如果您使用的是入門範例影像，請使用 extra_images 資料夾。Amazon S3 URI 是`s3://your bucket/circuitboard/extra_images`。

- 如果您要從電腦上傳影像，請在選擇「偵測異常」後新增影像。

10. (選擇性) 如果您想要使用自己的 AWS KMS 加密金鑰，請執行下列動作：

    a. 對於 [影像資料加密]，選擇 [自訂加密設定 (進階)]。

    b. 在加密 .aws_kms_key 中，輸入金鑰的亞馬遜資源名稱 (ARN)，或選擇現有的 AWS KMS 金鑰。若要建立新金鑰，請選擇建立 AWS IMS 金鑰。

11. 選擇 [偵測異常]，然後選擇 [執行試驗偵測] 以開始試用偵測工作。

12. 在試驗偵測檢視中檢查目前的狀態。試驗偵測可能需要一段時間才能完成。

驗證試驗檢測結果

驗證試驗偵測的結果可協助您改善模型。

如果效能指標不佳，請執行試用偵測來改善模型，然後將已驗證的影像新增至資料集 (如果您有個別的資料集，則訓練資料集)。

如果模型的效能指標良好，但試驗偵測的結果不佳，您可以透過將已驗證的影像新增至資料集 (訓練資料集) 來改善模型。如果您有個別的測試資料集，請考慮將更多影像新增至測試資料集。

將已驗證的影像新增至資料集後，請重新訓練並重新評估模型。如需詳細資訊，請參閱培訓您的模型。

驗證試驗偵測的結果

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 在左側導覽窗格中，選擇 Project。
3. 在專案頁面中，選擇您要使用的專案。顯示專案的儀表板。
4. 在左側導覽窗格中，選擇試用偵測。
5. 選擇您要驗證的試用偵測。
6. 在試驗偵測頁面上，選擇驗證機器預測。

7. 選擇 [選取此頁面上的所有影像]。
8. 如果預測正確，請選擇驗證為正確。否則，請選擇「驗證為不正確」。預測和預測信賴度分數會顯示在每張影像下方。
9. 如果需要變更影像的標，請執行下列動作：

   a. 選擇影像下方的「正確」或「錯誤」。

   b. 如果無法判斷影像的正確標籤，請在圖庫中選擇影像來放大影像。

> **Note**
>
> 您可以在「篩選器」區段中選擇所需的標籤或標籤狀態來篩選影像標籤。您可以在 [排序選項] 區段中依信賴度分數排序。

10. 如果您的模型是分割模型，且影像的遮色片或異常標籤有誤，請選擇影像下的「異常」區域，然後開啟註解工具。通過執行更新分段信息使用註解工具修正分段標示。
11. 視需要在每頁上重複步驟 7-10，直到所有影像都已驗證完畢。
12. 選擇新增已驗證的影像至資料集。如果您有個別的資料集，影像會新增至訓練資料集。
13. 重新訓練您的模型 如需詳細資訊，請參閱the section called “培訓您的模型”。

使用註解工具修正分段標示

您可以使用註解工具，透過使用遮色片標記異常區域來劃分影像。

使用註解工具修正影像的分段標示的步驟

1. 在資料集圖庫中選取影像下方的異常區域，以開啟註解工具。
2. 如果遮色片的異常標籤不正確，請選擇遮色片，然後在「異常」標籤下選擇正確的異常標籤。如有必要，請選擇「新增異常標籤」以新增異常標籤。
3. 如果遮色片不正確，請選擇頁面底部的繪圖工具，然後繪製緊密覆蓋異常標籤異常區域的遮色片。下列影像是緊密覆蓋異常的遮色片範例。

以下是不能緊密覆蓋異常狀況的不良遮罩範例。

4. 如果您有更多要校正的影像，請選擇「下一步」，然後重複步驟 2 和 3。
5. 選擇「提交」並關閉以完成更新影像。

運行訓練有素的亞馬遜 Lookout for Vision 模型

若要使用模型偵測影像中的異常，您必須先使用StartModel作業來啟動模型。Amazon 視 Lookout for Vision 察主控台提供可用來啟動和停止模型的AWS CLI命令。本節包含您可以使用的範例程式碼。

模型啟動後，您可以使用此DetectAnomalies作業偵測影像中的異常。如需詳細資訊，請參閱偵測影像中的異常。

主題

推論單位

當您啟動模型時，Amazon Lookout for Vision 會佈建至少一個運算資源，稱為推論單元。您可以在 StartModel API 的MinInferenceUnits輸入參數中指定要使用的推論單元數。模型的預設配置為 1 個推論單位。

> **Important**
>
> 系統會根據您設定模型執行的方式，向您收取模型執行中模型執行的小時數，以及模型執行時使用的推論單元數量。例如，如果您使用兩個推論單位啟動模型，並使用該模型 8 小時，則需支付 16 個推論小時的費用（8 小時執行時間 * 兩個推論單位）。如需詳細資訊，請參閱亞馬遜 Lookout for Vision 定價。如果您沒有透過呼叫來明確停止模型 StopModel，即使您沒有使用模型主動分析影像，仍需支付費用。

單一推論單元支援的每秒交易數 (TPS) 會受到下列因素影響：

- 「Lookout for Vision」用來訓練模型的演算法。訓練模型時，會訓練多個模型。Lookout for Vision 會根據資料集的大小及其正常和異常影像的組成，選取具有最佳效能的模型。
- 解析度較高的影像需要更多時間進行分析。
- 較小尺寸的影像 (以 MB 為單位) 的分析速度比較大的影像更快。

使用推論單元管理輸送量

您可以根據應用程式的需求增加或減少模型的輸送量。若要增加輸送量，請使用其他推論單元。每個額外的推論單元都會將您的處理速度提高一個推論單元。如需計算所需推論單位數量的詳細資訊，請參閱計算 Amazon Rekognition 自訂標籤的推論單位和 Amazon 瞭望視覺模型的推論單位。如果您想要變更模型支援的輸送量，您有兩種選擇：

手動加入或移除推論單位

停止模型，然後使用所需數量的推論單元重新啟動。這種方法的缺點是模型在重新啟動時無法接收請求，也無法用於處理需求峰值。如果您的模型具有穩定的輸送量，而且您的使用案例可以容忍 10—20 分鐘的停機時間，請使用此方法。例如，如果您想要使用每週排程來批次呼叫模型。

自動縮放推論單位

如果您的模型必須因應高峰的需求，Amazon Lookout for Vision 可以自動調整模型使用的推論單元數量。隨著需求的增加，Amazon Lookout for Vision 會在模型中新增額外的推論單元，並在需求減少時將其移除。

若要讓 Lookout for Vision 自動縮放模型的推論單位，請啟動模型並使用參數設定其可使用的最大推論單位數。`MaxInferenceUnits`設定推論單位的最大數目可讓您透過限制模型可用的推論單位數量來管理執行模型的成本。如果您未指定最大單位數，Lookout for Vision 將不會自動縮放模型，只會使用您開始使用的推論單位數量。如需推論單元數目上限的相關資訊，請參閱 Service Quotas。

您也可以使用參數來指定推論單位的最小`MinInferenceUnits`數目。這可讓您指定模型的最小輸送量，其中單一推論單位代表 1 小時的處理時間。

> **Note**
>
> 您無法使用 Lookout for Vision 主控台設定推論單元的最大數量。請改為指定`StartModel`作業的`MaxInferenceUnits`輸入參數。

Lookout for Vision 提供下列 Amazon CloudWatch 日誌指標，您可以使用這些指標來判斷模型目前的自動擴展狀態。

指標	描述
DesiredInferenceUnits	Lookout for Vision 正在擴展或縮小的推論單元的數量。
InServiceInferenceUnits	模型正在使用的推論單位數。

如果 DesiredInferenceUnits =InServiceInferenceUnits，Lookout for Vision 目前不會縮放推論單元的數量。

如果 DesiredInferenceUnits >InServiceInferenceUnits，則「查看視覺」將擴展到的值DesiredInferenceUnits。

如果 DesiredInferenceUnits <InServiceInferenceUnits，[Lookout for Vision] 會縮小至的值DesiredInferenceUnits。

如需瞭解視覺檢視和篩選維度傳回的指標的詳細資訊，請參閱使用 Amazon 監控 Lookout for Vision CloudWatch。

若要找出您要求的模型推論單元數目上限，請呼叫DescribeModel並檢查回應中的MaxInferenceUnits欄位。

可用區域

Amazon Lookout for Vision；將推論單元分配到一AWS個區域內的多個可用區域，以提供更高的可用性。如需詳細資訊，請參閱可用區域。為了協助保護您的生產模型免於可用區域中斷和推論單元故障的影響，請至少使用兩個推論單元來啟動生產模型。

如果發生可用區域中斷，則可用區域中的所有推論單元將無法使用，且模型容量也會降低。呼叫DetectAnomalies會重新分配至其餘的推論單元。如果這類呼叫未超過其餘推論單位所支援的每秒交易數 (TPS)，就會成功。AWS修復可用區域之後，推論單元會重新啟動，並恢復完整容量。

如果單一推論單元發生故障，Amazon Lookout for Vision 會自動在相同的可用區域中啟動新的推論單元。模型容量會減少，直到新的推論單元啟動為止。

開始您的亞馬遜 Lookout for Vision 模型

您必須先啟動模型，才能使用 Amazon Lookout for Vision 模型偵測異常。您可以透過呼叫 StartModelAPI 並傳遞下列指令來啟動模型：

- ProjectName— 包含您要啟動之模型的專案名稱。
- ModelVersion— 您要啟動的模型版本。
- MinInferenceUnits— 推論單位的最小數目。如需詳細資訊，請參閱推論單位。
- (選擇性) MaxInferenceUnits— Amazon 觀察視覺可用來自動擴展模型的推論單元數目上限。如需詳細資訊，請參閱自動縮放推論單位。

Amazon Lookout for Vision 主控台提供範例程式碼，您可以使用這些程式碼來啟動和停止模型。

> **Note**
>
> 您需支付模型執行時間的費用。若要停止執行中的模型，請參閱停止您的亞馬遜 Lookout for Vision 模型。
>
> 您可以使用 AWS SDK 來檢視所有提供瞭解視覺的AWS區域的執行中模型。如需範例程式碼，請參閱 find_running_models.py。

主題

啟動您的模型（控制台）

Amazon 視 Lookout for Vision 察主控台提供可用來啟動模型的AWS CLI命令。模型啟動後，您可以開始偵測影像中的異常。如需詳細資訊，請參閱偵測影像中的異常。

啟動模型（控制台）

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
3. 選擇 Get started (開始使用)。
4. 在左側導覽窗格中，選擇 [專案]。
5. 在 [專案資源] 頁面上，選擇包含您要啟動的訓練模型的專案。

6. 在「模型」區段中，選擇您要啟動的模型。
7. 在模型的詳細資料頁面上，選擇 [使用模型]，然後選擇 [將 API 整合至雲端]。

> **Tip**
>
> 如果要將模型部署到邊緣裝置，請選擇 [建立模型封裝工作]。如需詳細資訊，請參閱包裝您的亞馬遜 Lookout for Vision 模型。

8. 在 AWS CLI 命令下，複製呼叫的 AWS CLI 命令`start-model`。
9. 於指令提示下，輸入您在上一個步驟中複製的`start-model`指令。如果您使用`lookoutvision`設定檔取得認證，請新增`--profile lookoutvision-access`參數。
10. 在主控台中，選擇左側導覽頁面中的 [模型]。
11. 檢查「狀態」欄中的模型目前狀態，當狀態為「託管」時，您可以使用模型偵測影像中的異常。如需詳細資訊，請參閱偵測影像中的異常。

開始您的亞馬遜 Lookout for Vision 模型 (SDK)

您可以透過呼叫StartModel作業來啟動模型。

模型可能需要一段時間才能啟動。您可以通過調用來檢查當前狀態DescribeModel。如需詳細資訊，請參閱檢視您的模型。

若要啟動您的模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來啟動模型。

   CLI

   變更下列值：

   - `project-name`到包含您要啟動的模型的專案名稱。
   - `model-version`到您要啟動的模型版本。
   - `--min-inference-units`到您要使用的推論單元的數量。
   - (選擇性) Amazon Lookout for Vision 可用`--max-inference-units`來自動擴展模型的推論單元數目上限。

```
aws lookoutvision start-model --project-name "project name"\
  --model-version model version\
  --min-inference-units minimum number of units\
  --max-inference-units max number of units \
  --profile lookoutvision-access
```

Python

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
   @staticmethod
   def start_model(
           lookoutvision_client, project_name, model_version,
min_inference_units, max_inference_units = None):
       """
       Starts the hosting of a Lookout for Vision model.

       :param lookoutvision_client: A Boto3 Lookout for Vision client.
       :param project_name:  The name of the project that contains the version
of the
                             model that you want to start hosting.
       :param model_version: The version of the model that you want to start
hosting.
       :param min_inference_units: The number of inference units to use for
hosting.
       :param max_inference_units: (Optional) The maximum number of inference
units that
       Lookout for Vision can use to automatically scale the model.
       """
       try:
           logger.info(
               "Starting model version %s for project %s", model_version,
project_name)

           if max_inference_units is None:
               lookoutvision_client.start_model(
                   ProjectName = project_name,
                   ModelVersion = model_version,
                   MinInferenceUnits = min_inference_units)

           else:
```

```
                lookoutvision_client.start_model(
                    ProjectName = project_name,
                    ModelVersion = model_version,
                    MinInferenceUnits = min_inference_units,
                    MaxInferenceUnits = max_inference_units)

            print("Starting hosting...")

            status = ""
            finished = False

            # Wait until hosted or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STARTING_HOSTING":
                    logger.info("Host starting in progress...")
                    time.sleep(10)
                    continue

                if status == "HOSTED":
                    logger.info("Model is hosted and ready for use.")
                    finished = True
                    continue

                logger.info("Model hosting failed and the model can't be used.")
                finished = True

            if status != "HOSTED":
                logger.error("Error hosting model: %s", status)
                raise Exception(f"Error hosting model: {status}")
        except ClientError:
            logger.exception("Couldn't host model.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Starts hosting an Amazon Lookout for Vision model. Returns when the model has
```

```
     * started or if hosting fails. You are charged for the amount of time that a
     * model is hosted. To stop hosting a model, use the StopModel operation.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @param projectName The name of the project that contains the model that you
     *                    want to host.
     * @modelVersion The version of the model that you want to host.
     * @minInferenceUnits The number of inference units to use for hosting.
     * @maxInferenceUnits The maximum number of inference units that Lookout for
     *                    Vision can use for automatically scaling the model. If the
     *                    value is null, automatic scaling doesn't happen.
     * @return ModelDescription The description of the model, which includes the
     *         model hosting status.
     */
    public static ModelDescription startModel(LookoutVisionClient lfvClient, String
     projectName, String modelVersion,
            Integer minInferenceUnits, Integer maxInferenceUnits) throws
     LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Starting Model version {0} for project {1}.",
                new Object[] { modelVersion, projectName });

        StartModelRequest startModelRequest = null;

        if (maxInferenceUnits == null) {

            startModelRequest =
     StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)
                    .minInferenceUnits(minInferenceUnits).build();
        } else {
            startModelRequest =
     StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)

     .minInferenceUnits(minInferenceUnits).maxInferenceUnits(maxInferenceUnits).build();
        }

        // Start hosting the model.
        lfvClient.startModel(startModelRequest);

        DescribeModelRequest describeModelRequest =
     DescribeModelRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).build();

        ModelDescription modelDescription = null;
```

```
   boolean finished = false;
   // Wait until model is hosted or failure occurs.
   do {

       modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

       switch (modelDescription.status()) {

       case HOSTED:
           logger.log(Level.INFO, "Model version {0} for project {1} is
running.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       case STARTING_HOSTING:
           logger.log(Level.INFO, "Model version {0} for project {1} is
starting.",
                   new Object[] { modelVersion, projectName });

           TimeUnit.SECONDS.sleep(60);

           break;
       case HOSTING_FAILED:
           logger.log(Level.SEVERE, "Hosting failed for model version {0} for
project {1}.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       default:
           logger.log(Level.SEVERE, "Unexpected error when hosting model
version {0} for project {1}: {2}.",
                   new Object[] { projectName, modelVersion,
modelDescription.status() });
           finished = true;
           break;

       }

   } while (!finished);
```

```
    logger.log(Level.INFO, "Finished starting model version {0} for project {1}
 status: {2}",
            new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

    return modelDescription;

}
```

3. 如果程式碼的輸出為`Model is hosted and ready for use`，您可以使用模型偵測影像中的異常。如需詳細資訊，請參閱偵測影像中的異常。

停止您的亞馬遜 Lookout for Vision 模型

要停止正在運行的模型，請調用`StopModel`操作並傳遞以下內容：

- 專案 (Project) — 包含您要停止之模型的專案名稱。
- ModelVersion— 您要停止的模型版本。

Amazon 觀察視覺主控台提供可用於停止模型的範例程式碼。

> **Note**
>
> 您需支付模型執行時間的費用。

主題

- 停止您的模型（控制台）
- 停止您的亞馬遜 Lookout for Vision 模型（SDK）

停止您的模型（控制台）

執行下列程序中的步驟，使用控制台停止模型。

若要停止您的模型 (主控台)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。

2. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
3. 選擇 Get started (開始使用)。
4. 在左側導覽窗格中，選擇 [專案]。
5. 在 [專案資源] 頁面上，選擇包含您要停止的執行中模型的專案。
6. 在「模型」區段中，選擇您要停止的模型。
7. 在模型的詳細資料頁面上，選擇 [使用模型]，然後選擇 [將 API 整合至雲端]。
8. 在 AWS CLI 命令下，複製呼叫的 AWS CLI 命令`stop-model`。
9. 於指令提示下，輸入您在上一個步驟中複製的`stop-model`指令。如果您使用`lookoutvision`設定檔取得認證，請新增`--profile lookoutvision-access`參數。
10. 在主控台中，選擇左側導覽頁面中的 [模型]。
11. 檢查「狀態」(Status) 欄以瞭解模型的目前狀態。當「狀態」欄值為「訓練完成」時，模型已停止。

停止您的亞馬遜 Lookout for Vision 模型（SDK）

您可以呼叫StopModel作業來停止模型。

模型可能需要一段時間才能停止。若要檢查目前狀態，請使用`DescribeModel`。

若要停止您的模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來停止執行中的模型。

   CLI

   變更下列值：

   - `project-name`到包含要停止的模型的項目的名稱。
   - `model-version`到您要停止的模型版本。

```
aws lookoutvision stop-model --project-name "project name"\
    --model-version model version \
  --profile lookoutvision-access
```

Python

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def stop_model(lookoutvision_client, project_name, model_version):
        """
        Stops a running Lookout for Vision Model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version
of
                             the model that you want to stop hosting.
        :param model_version:  The version of the model that you want to stop
hosting.
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version
            )
            logger.info("Stopping hosting...")

            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
                    continue

                if status == "TRAINED":
                    logger.info("Model is no longer hosted.")
                    finished = True
                    continue
```

```
                logger.info("Failed to stop model: %s ", status)
                finished = True

            if status != "TRAINED":
                logger.error("Error stopping model: %s", status)
                raise Exception(f"Error stopping model: {status}")
        except ClientError:
            logger.exception("Couldn't stop hosting model.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Stops the hosting an Amazon Lookout for Vision model. Returns when model has
 * stopped or if hosting fails.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to stop hosting.
 * @modelVersion The version of the model that you want to stop hosting.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */

public static ModelDescription stopModel(LookoutVisionClient lfvClient, String
 projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        logger.log(Level.INFO, "Stopping Model version {0} for project {1}.",
                        new Object[] { modelVersion, projectName });

        StopModelRequest stopModelRequest = StopModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        // Stop hosting the model.
```

```
        lfvClient.stopModel(stopModelRequest);

        DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        ModelDescription modelDescription = null;

        boolean finished = false;
        // Wait until model is stopped or failure occurs.
        do {

                modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

                switch (modelDescription.status()) {

                        case TRAINED:
                                logger.log(Level.INFO, "Model version {0} for
project {1} has stopped.",
                                                new Object[] { modelVersion,
projectName });
                                finished = true;
                                break;

                        case STOPPING_HOSTING:
                                logger.log(Level.INFO, "Model version {0} for
project {1} is stopping.",
                                                new Object[] { modelVersion,
projectName });

                                TimeUnit.SECONDS.sleep(60);

                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when stopping
model version {0} for project {1}: {2}.",
                                                new Object[] { projectName,
modelVersion,
```

```
 modelDescription.status() });
                                finished = true;
                                break;

                }

        } while (!finished);

        logger.log(Level.INFO, "Finished stopping model version {0} for project
 {1} status: {2}",
                        new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

        return modelDescription;

}
```

偵測影像中的異常

若要使用訓練有素的 Amazon 視覺瞭望模型偵測影像中的異常，請呼叫DetectAnomalies操作。結果來自DetectAnomalies包含布林預測，將影像分類為包含一或多個異常，以及預測的置信度值。如果模型是圖像分割模型，則結果還包括顯示不同類型異常位置的彩色遮色片。

您提供的圖像DetectAnomalies必須具有與用於訓練模型的影像相同的寬度和高度尺寸。

DetectAnomalies接受圖像為 PNG 或 JPG 格式的圖像。我們建議影像的編碼和壓縮格式與用於訓練模型的格式相同。例如，如果您使用 PNG 格式影像訓練模型，請呼叫DetectAnomalies與 PNG 格式的圖像。

打電話之前DetectAnomalies，您必須先使用StartModel操作。如需詳細資訊，請參閱開始您的亞馬遜 Lookout for Vision 模型。您需支付模型執行的時間量 (以分鐘為單位)，以及模型使用的異常偵測單位數量收費。如果您不使用模型，請使用StopModel操作以停止模型。如需詳細資訊，請參閱停止您的亞馬遜 Lookout for Vision 模型。

主題

- 呼叫 DetectAnomalies
- 了解來自的回應DetectAnomalies
- 判斷影像是否異常
- 顯示分類和區段資訊
- 使用尋找異常AWS Lambda功能

呼叫 DetectAnomalies

要打電話DetectAnomalies」中，指定下列項目：

- 項目— 包含您要使用之模型的專案名稱。
- ModelVersion— 您要使用的模型版本。
- ContentType— 您要分析的圖像類型。有效值為image/png（PNG 格式圖像）和image/jpeg（JPG 格式的圖像）。
- 身體— 代表影像的未編碼二進位位元組。

  影像的尺寸必須與用於訓練模型的影像具有相同的尺寸。

下面的例子演示了如何調用DetectAnomalies。您可以使用 Python 和 Java 實例中的函數響應來調用函數判斷影像是否異常。

AWS CLI

此 AWS CLI 命令顯示 DetectAnomalies CLI 操作的 JSON 輸出。變更下列輸入參數的值：

- project name使用您要使用的項目的名稱。
- model version使用您要使用的模型版本。
- content type與您要使用的圖像的類型。有效值為image/png（PNG 格式圖像）和image/jpeg（JPG 格式的圖像）。
- file name使用您要使用的圖像的路徑和文件名。確定檔案類型符合的值content-type。

```
aws lookoutvision detect-anomalies  --project-name project name\
    --model-version model version\
    --content-type content type\
    --body file name \
    --profile lookoutvision-access
```

Python

如需完整的程式碼範例，請參閱GitHub。

```
    def detect_anomalies(lookoutvision_client, project_name, model_version, photo):
        """
        Calls DetectAnomalies using the supplied project, model version, and image.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project: The project that contains the model that you want to use.
        :param model_version: The version of the model that you want to use.
        :param photo: The photo that you want to analyze.
        :return: The DetectAnomalyResult object that contains the analysis results.
        """

        image_type = imghdr.what(photo)
        if image_type == "jpeg":
            content_type = "image/jpeg"
        elif image_type == "png":
            content_type = "image/png"
        else:
            logger.info("Invalid image type for %s", photo)
```

```
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file: {photo}")


        # Get images bytes for call to detect_anomalies
        with open(photo, "rb") as image:
            response = lookoutvision_client.detect_anomalies(
                ProjectName=project_name,
                ContentType=content_type,
                Body=image.read(),
                ModelVersion=model_version)

        return response['DetectAnomalyResult']
```

Java V2

```
  public static DetectAnomalyResult detectAnomalies(LookoutVisionClient lfvClient,
String projectName,
            String modelVersion,
            String photo) throws IOException, LookoutVisionException {
        /**
         * Creates an Amazon Lookout for Vision dataset from a manifest file.
         * Returns after Lookout for Vision creates the dataset.
         *
         * @param lfvClient    An Amazon Lookout for Vision client.
         * @param projectName  The name of the project in which you want to create a
         *                     dataset.
         * @param modelVersion The version of the model that you want to use.
         *
         * @param photo        The photo that you want to analyze.
         *
         * @return DetectAnomalyResult The analysis result from DetectAnomalies.
         */

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes.

        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
        byte[] imageBytes = imageSDKBytes.asByteArray();

        // Get the image type. Can be image/jpeg or image/png.
```

```
        String contentType = getImageType(imageBytes);

        // Detect anomalies in the supplied image.
        DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).contentType(contentType).build();

        DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
                RequestBody.fromBytes(imageBytes));

        /*
         * Tip: You can also use the following to analyze a local file.
         * Path path = Paths.get(photo);
         * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
         */
        DetectAnomalyResult result = response.detectAnomalyResult();

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert confidence to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Log classification result.
        String photoPath = "File: " + photo;
        String[] imageLines = { photoPath, prediction, confidence };
        logger.log(Level.INFO, "Image: {0}\nAnomalous: {1}\nConfidence {2}",
imageLines);

        return result;

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.
    private static String getImageType(byte[] image) throws IOException {

        InputStream is = new BufferedInputStream(new ByteArrayInputStream(image));
```

```
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);

        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }
```

了解來自的回應DetectAnomalies

來自的回應DetectAnomalies根據您訓練的模型類型 (分類模型或分段模型) 而有所不同。在這兩種情況下，響應都是DetectAnomalyResult物件。

分類模型

如果您的模型是圖像分類模型，來自的回應DetectAnomalies包含以下內容：

- IsAnomalous— 圖像包含一個或多個異常的布爾指示器。
- 信心— 亞馬遜的視覺瞭望在異常預測的準確性的信心 (IsAnomalous).Confidence是介於 0 和 1 之間的浮點值。較高的值表示信賴度越高。
- 來源— 有關傳遞給圖像的信息DetectAnomalies。

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996867775917053
    }
}
```

您可以通過檢查來確定圖像中是否異常IsAnomalous欄位並確認Confidence價值足以滿足您的需求。

如果您找到傳回的置信度值DetectAnomalies太低，請考慮重新訓練模型。如需範例程式碼，請參閱分類。

分割模型

如果您的模型是圖像分割模型，回應包括分類資訊和區段資訊，例如影像遮罩和異常類型。分類資訊是與區段資訊分開計算的，您不應該假設它們之間的關係。如果您在回應中未取得區段資訊，請檢查您是否擁有最新版本的AWS已安裝的 SDK (AWS Command Line Interface，如果您正在使用AWS CLI). 如需範例程式碼，請參閱分段和顯示分類和區段資訊。

- IsAnomalous(分類) — 將影像分類為正常或異常的布林指示器。
- 信心(分類) — 亞馬遜的信心瞭望視覺在圖像的分類的準確性 (IsAnomalous).Confidence是介於 0 和 1 之間的浮點值。較高的值表示信賴度越高。
- 來源— 有關傳遞給圖像的信息DetectAnomalies。
- AnomalyMask(分割) — 涵蓋分析影像中發現異常的像素遮色片。影像上可能有多個異常。遮色片貼圖的顏色表示異常的類型。遮色片顏色會對應至指派給訓練資料集中異常類型的顏色。若要從遮色片顏色尋找異常類型，請核取Color在PixelAnomaly中傳回的每個異常的欄位Anomalies列表。如需範例程式碼，請參閱顯示分類和區段資訊。
- 異常(分割) — 在影像中找到的異常清單。每個異常包括異常類型 (Name) 和像素資訊 (PixelAnomaly).TotalPercentageArea是異常覆蓋的影像百分比區域。Color是異常的遮色片顏色。

  清單中的第一個元素永遠是代表影像背景的異常類型 (BACKGROUND ），並且不應該被視為異常。亞馬遜視覺觀察會自動將背景異常類型添加到響應中。您不需要在資料集中宣告背景異常類型。

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996814727783203,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.998999834060669,
                    "Color": "#FFFFFF"
```

```
                }
            },
            {
                "Name": "scratch",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0004034999874420464,
                    "Color": "#7ED321"
                }
            },
            {
                "Name": "dent",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0005966666503809392,
                    "Color": "#4DD8FF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0....."
    }
}
```

判斷影像是否異常

您可以透過多種方式判斷影像是否異常。您選擇的方法取決於您的使用案例和模型的類型。以下是潛在的解決方案。

主題

- 分類
- 分段

分類

IsAnomalous將影像分類為異常，請使用Confidence欄位，以協助決定影像是否實際上是異常的。較高的值表示信心越大。例如，只有當信心超過 80% 時，您可能才會決定某個產品有缺陷。您可以分類按分類模型或圖像分割模型分析的圖像進行分類。

Python

如需完整的程式碼範例，請參閱GitHub。

```
   def reject_on_classification(image, prediction, confidence_limit):
       """
       Returns True if the anomaly confidence is greater than or equal to
       the supplied confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence_limit: The minimum acceptable confidence. Float value
between 0 and 1.
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       reject = False

       logger.info("Checking classification for %s", image)

       if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
           reject = True
           reject_info=(f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                   f" than limit ({confidence_limit:.2%})")
           logger.info("%s", reject_info)

       if not reject:
           logger.info("No anomalies found.")
       return reject
```

Java V2

```
   public static boolean rejectOnClassification(String image, DetectAnomalyResult
prediction, float minConfidence) {
       /**
        * Rejects an image based on its anomaly classification and prediction
        * confidence
        *
        * @param image         The file name of the analyzed image.
        * @param prediction    The prediction for an image analyzed with
        *                      DetectAnomalies.
        * @param minConfidence The minimum acceptable confidence for the prediction
        *                      (0-1).
        *
        * @return boolean True if the image is anomalous, otherwise False.
```

```
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking classification for {0}", image);

        String[] logParameters = { prediction.confidence().toString(),
String.valueOf(minConfidence) };

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is greater than
confidence limit {1}",
                    logParameters);
            reject = true;
        }
        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;

    }
```

分段

如果您的模型是影像分割模型，您可以使用分割資訊來判斷影像是否包含異常。您也可以使用影像分割模型來分類影像。如需取得並顯示影像遮色片的範例程式碼，請參閱顯示分類和區段資訊

異常區域

使用百分比覆蓋範圍 (TotalPercentageArea) 影像上的異常狀況。例如，如果某個異常區域大於影像的 1%，您可能會決定某個產品有瑕疵。

Python

如需完整的程式碼範例，請參閱GitHub。

```
    def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
and if
```

```
        the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
 DetectAnomalies
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :anomaly_label: The anomaly label for the type of anomaly that you want to
 check.
        :coverage_limit: The maximum acceptable percentage coverage of an anomaly
 (float 0-1).
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False

        logger.info("Checking coverage for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
 confidence_limit:
            for anomaly in prediction['Anomalies']:
                if (anomaly['Name'] == anomaly_label and
                        anomaly['PixelAnomaly']['TotalPercentageArea'] >
 (coverage_limit)):
                    reject = True
                    reject_info=(f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) "
                        f"is greater than limit ({confidence_limit:.2%}) and
 {anomaly['Name']} "
                        f"coverage ({anomaly['PixelAnomaly']
['TotalPercentageArea']:.2%}) "
                        f"is greater than limit ({coverage_limit:.2%})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")

        return reject
```

Java V2

```
   public static Boolean rejectOnCoverage(String image, DetectAnomalyResult
 prediction, float minConfidence,
```

```
            String anomalyType, float maxCoverage) {
        /**
         * Rejects an image based on a maximum allowable coverage area for an
anomaly
         * type.
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
         * @param minConfidence The minimum acceptable confidence for the prediction
         *                      (0-1).
         * @param anomalyTypes  The anomaly type to check.
         * @param maxCoverage   The maximum allowable coverage area of the anomaly
type.
         *                      (0-1).
         *
         * @return boolean True if the coverage area of the anomaly type exceeds the
         *         maximum allowed, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking coverage for {0}", image);

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            for (Anomaly anomaly : prediction.anomalies()) {

                if (Objects.equals(anomaly.name(), anomalyType)
                        && anomaly.pixelAnomaly().totalPercentageArea() >=
maxCoverage) {

                    String[] logParameters = { prediction.confidence().toString(),
                            String.valueOf(minConfidence),

String.valueOf(anomaly.pixelAnomaly().totalPercentageArea()),
                            String.valueOf(maxCoverage) };
                    logger.log(Level.INFO,
                            "Rejected: Anomaly confidence {0} is greater than
confidence limit {1} and " +
                                    "{2} anomaly type coverage is higher than
coverage limit {3}\n",
                            logParameters);
                    reject = true;
```

```
                }
            }
        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

異常類型數

使用不同異常類型的計數 (Name) 在圖像上找到。例如，如果存在兩種以上的異常類型，您可能會決定某個產品有缺陷。

Python

如需完整的程式碼範例，請參閱GitHub。

```
   def reject_on_anomaly_types(image, prediction, confidence_limit,
anomaly_types_limit):
        """
        Checks if the number of anomaly types is greater than than the anomaly types
        limit and if the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
        :param confidence: The minimum acceptable confidence. Float value between 0
and 1.
        :param anomaly_types_limit: The maximum number of allowable anomaly types
(Integer).
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        logger.info("Checking number of anomaly types for %s",image)

        reject = False

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
```

```
            anomaly_types = {anomaly['Name'] for anomaly in prediction['Anomalies']\
                if anomaly['Name'] != 'background'}

            if len (anomaly_types) > anomaly_types_limit:
                    reject = True
                    reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) "
                    f"is greater than limit ({confidence_limit:.2%}) and "
                    f"the number of anomaly types ({len(anomaly_types)-1}) is "
                    f"greater than the limit ({anomaly_types_limit})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static Boolean rejectOnAnomalyTypeCount(String image, DetectAnomalyResult
prediction,
            float minConfidence, Integer maxAnomalyTypes) {

        /**
         * Rejects an image based on a maximum allowable number of anomaly types.
         *
         * @param image           The file name of the analyzed image.
         * @param prediction      The prediction for an image analyzed with
         *                        DetectAnomalies.
         * @param minConfidence   The minimum acceptable confidence for the
predictio
         *                        (0-1).
         * @param maxAnomalyTypes The maximum allowable number of anomaly types.
         *
         * @return boolean True if the image contains more than the maximum allowed
         *         anomaly types, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking coverage for {0}", image);

        Set<String> defectTypes = new HashSet<>();
```

```
        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            for (Anomaly anomaly : prediction.anomalies()) {
                defectTypes.add(anomaly.name());
            }
            // Reduce defect types by one to account for 'background' anomaly type.
            if ((defectTypes.size() - 1) > maxAnomalyTypes) {
                String[] logParameters = { prediction.confidence().toString(),
                        String.valueOf(minConfidence),
                        String.valueOf(defectTypes.size()),
                        String.valueOf(maxAnomalyTypes) };
                logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is >=
minimum confidence {1} and " +
                        "the number of anomaly types {2} > the allowable number of
anomaly types {3}\n", logParameters);
                reject = true;
            }

        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

顯示分類和區段資訊

此範例顯示分析的影像並覆疊分析結果。如果回應包含異常遮罩，則遮色片會以相關異常類型的顏色顯示。

若要顯示影像分類和影像分割資訊

1. 如果您尚未這麼做，請執行下列動作：

   a. 如果您尚未這樣做，請安裝並配置AWS CLI和AWS軟體開發套件。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。

   b. 訓練您的模型。

   c. 開始您的模型。

2. 確保用戶呼叫DetectAnomalies可以存取您要使用的模型版本。如需詳細資訊，請參閱設定 SDK 權限。
3. 使用下面的代碼。

Python

下列範例程式碼會偵測您提供的映像中的異常。此範例採用下列命令列選項：

- project— 您要使用的專案名稱。
- version— 您要在專案中使用的模型版本。
- image— 本機影像檔案 (JPEG 或 PNG 格式) 的路徑和檔案。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Shows how to detect and show anomalies in an image using a trained Amazon
 Lookout
for Vision model. The script displays the analysed image and overlays mask and
 analysis
output.
"""

import argparse
import logging
import io
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ShowAnomalies:
    """
    Class to detect and show anomalies in an image analyzed by detect_anomalies.
    """



    @staticmethod
    def draw_line(draw, text, fnt,  y_coordinate, color):
```

```
        """
        Draws a line of text on the supplied drawing surface.
        :param draw: The surface on which to draw the text.
        :param text: The text to draw in the drawing surface.
        :param fnt: The font for the text.
        :param y_coordinate: The y position for the text.
        :param color: The color for the text.
        :returns The y coordinate for the next line of text.
        """
        text_width, text_height = draw.textsize(text, fnt)
        draw.rectangle([(10, y_coordinate), (text_width + 10,
                                             y_coordinate + text_height)],
 fill="black")
        draw.text((10, y_coordinate), text, fill=color, font=fnt)

        y_coordinate += text_height

        return y_coordinate

    @staticmethod
    def draw_analysis_text(image, analysis):
        """
        Draws classification and segmentation info onto supplied image
        overlay analysis results on an image analyzed by detect_anomalies.
        :param analysis: The response from a call to detect_anomalies.
        :returns Nothing
        """

        ## Calculate a reasonable font size based on image width.
        font_size = int(image.size[0]/32)

        fnt = ImageFont.truetype('/Library/Fonts/Tahoma.ttf', font_size)

        draw = ImageDraw.Draw(image)

        y_coordinate = 0

        # Draw classification information.
        prediction = "Anomalous" if analysis["DetectAnomalyResult"]
["IsAnomalous"] \
            else "Normal"

        confidence = analysis["DetectAnomalyResult"]["Confidence"]
        found_anomalies = analysis["DetectAnomalyResult"]['Anomalies']
```

```
        segmentation_info = False

        logger.info("Prediction: %s", format(prediction))
        logger.info("Confidence: %s", format(confidence))

        y_coordinate = 0
        y_coordinate = ShowAnomalies.draw_line(
            draw, "Classification", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Prediction: {prediction}", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Confidence: {confidence:.2%}", fnt, y_coordinate, "white")

        # Draw segmentation information, if present.
        if (len(found_anomalies)) > 1:
            logger.info("Anomalies:")

            y_coordinate = ShowAnomalies.draw_line(
                draw, "Segmentation:", fnt, y_coordinate, "white")
            for i in range(1, len(found_anomalies)):

                # Only display info if more than 0% coverage found.
                percent_coverage = found_anomalies[i]['PixelAnomaly']
['TotalPercentageArea']
                if percent_coverage > 0:
                    segmentation_info = True
                    logger.info("  %s", found_anomalies[i]['Name'])
                    logger.info("    Color: %s",
                                found_anomalies[i]['PixelAnomaly']['Color'])
                    logger.info("    Area: %s", percent_coverage)
                    y_coordinate = ShowAnomalies.draw_line(
                        draw,
                        f"  Anomaly: {found_anomalies[i]['Name']}. Area:
 {percent_coverage:.2%}",
                        fnt,
                        y_coordinate,
                        found_anomalies[i]['PixelAnomaly']['Color'])

            if not segmentation_info:
                y_coordinate = ShowAnomalies.draw_line(
                    draw, "No segmentation information found.", fnt,
 y_coordinate, "white")
```

```
   @staticmethod
   def show_anomaly_prediction(lookoutvision_client, project_name,
model_version, photo):
       """
       Detects anomalies in an image (jpg/png) by using your Amazon Lookout for
Vision
       model. Displays the image and overlays prediction information text.
       :param lookoutvision_client: An Amazon Lookout for Vision Boto3 client.
       :param project_name: The name of the project that contains the model
that
       you want to use.
       :param model_version: The version of the model that you want to use.
       :param photo: The path and name of the image in which you want to detect
       anomalies.
       """
       try:

           logger.info("Detecting anomalies in %s", photo)

           image = Image.open(photo)
           image_type = Image.MIME[image.format]

           # Check that image type is valid.
           if image_type not in ("image/jpeg", "image/png"):
               logger.info("Invalid image type for %s", photo)
               raise ValueError(
                   f"Invalid file format. Supply a jpeg or png format file:
{photo}"
               )

           # Get images bytes for call to detect_anomalies.
           image_bytes = io.BytesIO()
           image.save(image_bytes, format=image.format)
           image_bytes = image_bytes.getvalue()

           # Analyze the image.
           response = lookoutvision_client.detect_anomalies(
               ProjectName=project_name,
               ContentType=image_type,
               Body=image_bytes,
               ModelVersion=model_version
           )
```

```
            # Overlay mask onto analyzed image.
            image_mask_bytes = response["DetectAnomalyResult"]["AnomalyMask"]
            image_mask = Image.open(io.BytesIO(image_mask_bytes))

            final_img = Image.blend(image, image_mask, 0.5) \
                if response["DetectAnomalyResult"]["IsAnomalous"] else image

            # Overlay analysis output on image.
            ShowAnomalies.draw_analysis_text(final_img, response)

            final_img.show()

        except ClientError as err:
            logger.info(format(err))
            raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project", help="The project containing the model that you want to use."
    )
    parser.add_argument(
        "version", help="The version of the model that you want to use."
    )
    parser.add_argument(
        "image",
        help="The file that you want to analyze. "
        "Supply a local file path.",
    )


def main():
    """
    Entrypoint for anomaly detection example.
    """

    try:
        logging.basicConfig(level=logging.INFO,
```

```
                            format="%(levelname)s: %(message)s")

        session = boto3.Session(
            profile_name='lookoutvision-access')

        lookoutvision_client = session.client("lookoutvision")

        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

        add_arguments(parser)

        args = parser.parse_args()

        # Analyze the image and show results.
        ShowAnomalies.show_anomaly_prediction(
            lookoutvision_client, args.project, args.version, args.image
        )

    except ClientError as err:
        print("A service error occured: " +
              format(err.response["Error"]["Message"]))
    except FileNotFoundError as err:
        print("The supplied file couldn't be found: " + err.filename)
    except ValueError as err:
        print("A value error occured. " + format(err))
    else:
        print("Successfully completed analysis.")


if __name__ == "__main__":
    main()
```

Java 2

下列範例程式碼會偵測您提供的映像中的異常。此範例採用下列命令列選項：

- project— 您要使用的專案名稱。
- version— 您要在專案中使用的模型版本。
- image— 本機影像檔案 (JPEG 或 PNG 格式) 的路徑和檔案。

```
/*
```

```
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/


package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.Anomaly;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesResponse;
import software.amazon.awssdk.services.lookoutvision.model.DetectAnomalyResult;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.net.URLConnection;

import java.text.NumberFormat;
import java.awt.*;
import java.awt.font.LineMetrics;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.util.logging.Level;
import java.util.logging.Logger;

// Finds anomalies on a supplied image.
public class ShowAnomalies extends JPanel {
/**
 * Finds and displays anomalies on a supplied image.
 */
```

```
   private static final long serialVersionUID = 1L;
   private transient BufferedImage image;
   private transient BufferedImage maskImage;
   private transient Dimension dimension;
   public static final Logger logger =
Logger.getLogger(ShowAnomalies.class.getName());

   // Constructor. Finds anomalies in a local image file.
   public ShowAnomalies(LookoutVisionClient lfvClient, String projectName,
String modelVersion,
           String photo) throws IOException, LookoutVisionException {

       logger.log(Level.INFO, "Processing local file: {0}", photo);

       maskImage = null;

       // Get image bytes and buffered image.
       InputStream sourceStream = new FileInputStream(new File(photo));
       SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
       byte[] imageBytes = imageSDKBytes.asByteArray();
       ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageSDKBytes.asByteArray());
       image = ImageIO.read(inputStream);

       // Get the image type. Can be image/jpeg or image/png.
       String contentType = getImageType(imageBytes);

       // Set the size of the window that shows the image.
       setWindowDimensions();

       // Detect anomalies in the supplied image.
       DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
               .modelVersion(modelVersion).contentType(contentType).build();

       DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
               RequestBody.fromBytes(imageBytes));

       /*
        * Tip: You can also use the following to analyze a local file.
        * Path path = Paths.get(photo);
        * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
```

```
         */
        DetectAnomalyResult result = response.detectAnomalyResult();


        if (result.anomalyMask() != null){
            SdkBytes maskSDKBytes = result.anomalyMask();

            ByteArrayInputStream maskInputStream = new
ByteArrayInputStream(maskSDKBytes.asByteArray());
            maskImage = ImageIO.read(maskInputStream);
        }

        drawImageInfo(result);

    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller.
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getWidth() < dimension.width || image.getHeight() <
dimension.height) {
            dimension.width = image.getWidth();
            dimension.height = image.getHeight();
        }
        setPreferredSize(dimension);

    }

    private int drawLine(Graphics2D g2d, String line, FontMetrics metrics, int
yPos, Color color) {
    /**
    * Draws a line of text at the spsecified y position and color.
    * confidence
    *
    * @param g2D The Graphics2D object for the image.
    * @param line The line of text to draw.
    * @param metrics The font information to use.
    * @param yPos The y position for the line of text.
    *
    * @return  The yPos for the next line of text.
```

```
    */

        int indent = 10;

        // Get text height, width, and descent.
        int textWidth = metrics.stringWidth(line);
        LineMetrics lm = metrics.getLineMetrics(line, g2d);
        int textHeight = (int) lm.getHeight();
        int descent = (int) lm.getDescent();

        int y2Pos = (yPos + textHeight) - descent;

        // Draw black rectangle.
        g2d.setColor(Color.BLACK);
        g2d.fillRect(indent, yPos, textWidth, textHeight);

        // Draw text.
        g2d.setColor(color);
        g2d.drawString(line, indent, y2Pos);

        yPos += textHeight;

        return yPos;

    }

    public void drawImageInfo(DetectAnomalyResult result) {
    /**
     * Draws the results from DetectAnomalies onto the output image.
     *
     * @param result The response from a call to
     *               DetectAnomalies.
     *
     */

        // Set up drawing.
        Graphics2D g2d = image.createGraphics();

        if (result.anomalyMask() != null){
            Composite composite = g2d.getComposite();
            g2d.setComposite(AlphaComposite.SrcOver.derive(0.5f));
            int x = (image.getWidth() - maskImage.getWidth()) / 2;
            int y = (image.getHeight() - maskImage.getHeight()) / 2;
            g2d.drawImage(maskImage, x, y, null);
```

```
            // Set composite for overlaying text.
            g2d.setComposite(composite);
        }

        //Calculate font size based on arbitary 32 pixel image width.
        int fontSize = (image.getWidth() / 32);


        g2d.setFont(new Font("Tahoma", Font.PLAIN, fontSize));
        Font font = g2d.getFont();
        FontMetrics metrics = g2d.getFontMetrics(font);

        // Get classification information.

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert prediction to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Draw classification information.
        int yPos = 0;

        yPos = drawLine(g2d, "Classification:", metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, prediction, metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, confidence, metrics, yPos, Color.WHITE);

        // Draw segmentation info.
        yPos = drawLine(g2d, "Segmentation:", metrics, yPos, Color.WHITE);

        // Ignore background label, so size must be > 1
        if (result.anomalies().size() > 1) {
            for (Anomaly anomaly : result.anomalies()) {
                if (anomaly.name().equals("background"))
                    continue;
                String label = String.format("Anomaly: %s. Area: %s",
anomaly.name(),
```

```
defaultFormat.format(anomaly.pixelAnomaly().totalPercentageArea()));
                Color anomalyColor =
Color.decode((anomaly.pixelAnomaly().color()));
                yPos = drawLine(g2d, label, metrics, yPos, anomalyColor);

            }

        } else {
            drawLine(g2d, "None found.", metrics, yPos, Color.WHITE);

        }

        g2d.dispose();

    }

    @Override
    public void paintComponent(Graphics g)
    /**
     * Draws the image and analysis results.
     *
     * @param g The Graphics context object for drawing.
     *          DetectAnomalies.
     *
     */
    {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.

    private String getImageType(byte[] image) throws IOException
    /**
     * Gets the file type of a supplied image. Raises an exception if the image
     * isn't compatible with with Amazon Lookout for Vision.
     *
     * @param image The image that you want to check.
     *
```

```
     * @return String The type of the image.
     */

    {
        InputStream is = new BufferedInputStream(new
ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);

        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }

    public static void main(String[] args) throws Exception {

        String photo = null;
        String projectName = null;
        String modelVersion = null;

        final String USAGE = "\n" +
                "Usage:\n" +
                "    DetectAnomalies <project> <version> <image> \n\n" +
                "Where:\n" +
                "    project - The Lookout for Vision project.\n\n" +
                "    version - The version of the model within the project.\n\n"
+
                "    image - The path and filename of a local image. \n\n";

        try {

            if (args.length != 3) {
                System.out.println(USAGE);
                System.exit(1);
            }

            projectName = args[0];
            modelVersion = args[1];
            photo = args[2];
```

```
            ShowAnomalies panel = null;

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

.credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                .build();

            // Create frame and panel.
            JFrame frame = new JFrame(photo);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            panel = new ShowAnomalies(lfvClient, projectName, modelVersion,
photo);

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);


        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Lookout for Vision client error: {0}:
{1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("lookout for vision client error:
%s", lfvError.getMessage()));
            System.exit(1);

        } catch (FileNotFoundException fileError) {
            logger.log(Level.SEVERE, "Could not find file: {0}",
fileError.getMessage());
            System.out.println(String.format("Could not find file: %s",
fileError.getMessage()));
            System.exit(1);

        } catch (IOException ioError) {
            logger.log(Level.SEVERE, "IO error {0}", ioError.getMessage());
            System.out.println(String.format("IO error: %s",
ioError.getMessage()));
            System.exit(1);
        }

    }
```

```
}
```

4. 如果您不打算繼續使用模型，停止您的模型。

使用尋找異常AWS Lambda功能

AWS Lambda 是一項運算服務，可讓您執行程式碼，無需佈建或管理伺服器。例如，您可以分析從行動應用程式提交的影像，而不必建立伺服器來託管應用程式程式碼。以下說明顯示瞭如何在 Python 中創建一個調用 Lambda 函數DetectAnomalies。此函數會分析提供的影像，並傳回該影像中存在異常的分類。這些指示包括範例 Python 程式碼，示範如何使用 Amazon S3 儲存貯體中的映像呼叫 Lambda 函數，或從本機電腦提供的映像呼叫 Lambda 函數。

主題

- 步驟 1：建立AWS Lambda功能（控制台）
- 步驟 2：（可選）創建一個層（控制台）
- 第 3 步：添加 Python 代碼（控制台）
- 步驟 4：嘗試您的 Lambda 函數

步驟 1：建立AWS Lambda功能（控制台）

在此步驟中，您將創建一個空的AWS函數和 IAM 執行角色，可讓您的函數呼叫`DetectAnomalies`操作。它還授予對存放映像以進行分析的 Amazon S3 儲存貯體的存取權。您也可以為下列項目指定環境變數：

- 您希望 Lambda 函數使用的亞馬遜觀察視覺專案和模型版本。
- 您希望模型使用的可信度限制。

稍後，您可以將原始程式碼和選擇性地新增至 Lambda 函數的圖層。

若要建立AWS Lambda功能（控制台）

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 建立函數 。如需詳細資訊，請參閱〈使用主控台建立 Lambda 函數。
3. 選擇下列選項。

- 選擇 從頭開始撰寫。
- 輸入以下項目的值函數名稱。
- 對於运行时選擇蟒蛇。

4. 選擇建立函數以建立AWS Lambda功能。
5. 在功能頁面上，選擇配置標籤。
6. 在「」環境變量」窗格中，選擇編輯。
7. 新增下列環境變數。對於每個變量選擇加入環境變數然後輸入變量鍵和值。

索引鍵	值
專案名稱	包含您要使用的模型的觀察視覺專案。
模型版本	您要使用的模型版本。
信心	模型對於預測是異常的可信度的最小值 (0-100)。如果信心較低，則分類被認為是正常的。

8. 選擇儲存以儲存環境變數。
9. 在「」權限窗格，「下」角色名稱」下方，選擇執行角色以在 IAM 主控台中開啟角色。
10. 在權限」頁籤上，選擇新增權限然後建立內嵌政策。
11. 選擇JSON並以下列原則取代現有原則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lookoutvision:DetectAnomalies",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
}
```

12. 選擇 下一步。
13. 在政策詳情，輸入策略的名稱，例如DetectAnomalies-訪問。

14. 選擇 建立政策。
15. 如果您要將映像存放在 Amazon S3 儲存貯體中進行分析，請重複步驟 10-14。

    a. 對於步驟 11，請使用下列原則。取代*##/#####*使用 Amazon S3 儲存貯體和您要分析之映像檔的資料夾路徑。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

    b. 對於步驟 13，選擇不同的策略名稱，例如S3 儲存貯體存取。

步驟 2：（可選）創建一個層（控制台）

若要執行此範例，您不需要執行此步驟。該DetectAnomalies作業包含在預設的 Lambda Python 環境中，AWS開發套件為蟒蛇 (肉毒 3). 如果您的 Lambda 函數的其他部分需要最近AWS不在預設 Lambda Python 環境中的服務更新，請執行此步驟，將最新的 Boto3 SDK 版本新增為函數的一個層。

首先，您要建立一個包含 Boto3 SDK 的 .zip 檔案歸檔。然後建立圖層，並將 .zip 檔案封存新增至圖層。如需詳細資訊，請參閱〈搭配 Lambda 函數使用圖層。

建立和新增圖層 (控制台) 的步驟

1. 打開命令提示符，然後輸入以下命令。

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. 請記下壓縮檔案的名稱 (boto3-layer.zip)。您在此程序的步驟 6 中需要它。
3. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
4. 在導覽窗格中，選擇 Layers (層)。

5. 選擇 Create layer (建立 Layer)。
6. 輸入 Name (名稱) 與 Description (描述) 的值。
7. 選擇上傳一個 .zip 文件並選擇上传。
8. 在對話方塊中，選擇您在此程序的步驟 1 中建立的 .zip 檔案封存 (boto3-layer.zip)。
9. 如需相容的執行階段，請選擇蟒蛇 3.9。
10. 選擇創建以建立圖層。
11. 選擇導覽窗格功能表圖示。
12. 在導覽窗格中，選擇函數。
13. 在資源清單中，選擇您在其中建立的函數步驟 1：建立AWS Lambda功能（控制台）。
14. 選取程式碼索引標籤。
15. 在圖層區段中，選擇新增圖層。
16. 選擇自訂圖層。
17. 在自訂圖層，選擇您在步驟 6 中輸入的圖層名稱。
18. 在版本選擇圖層版本，應為 1。
19. 選擇 Add (新增)。

第 3 步：添加 Python 代碼（控制台）

在此步驟中，您可以使用 Lambda 主控台程式碼編輯器，將 Python 程式碼新增至 Lambda 函數。該代碼分析提供的圖像`DetectAnomalies`並返回一個分類（如果圖像異常為 true，如果圖像是正常的則為 false）。提供的映像檔可以位於 Amazon S3 儲存貯體中，或以 byte64 編碼的影像位元組提供。

若要新增 Python 程式碼 (主控台)

1. 如果您不在 Lambda 主控台中，請執行下列動作：
   a. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
   b. 開啟您在其中建立的 Lambda 函數步驟 1：建立AWS Lambda功能（控制台）。
2. 選取程式碼索引標籤。
3. 在源代碼，取代程式碼lambda_function.py具有以下內容：

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
```

```
Purpose
An AWS lambda function that analyzes images with an Amazon Lookout for Vision
 model.
"""
import base64
import imghdr
from os import environ
from io import BytesIO
import logging
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


# Get the model and confidence.
project_name = environ['PROJECT_NAME']
model_version = environ['MODEL_VERSION']
min_confidence = int(environ.get('CONFIDENCE', 50))

lookoutvision_client = boto3.client('lookoutvision')


def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        file_name = ""

        # Determine image source.
        if 'image' in event:
            # Decode the encoded image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image_type = get_image_type(img_b64decoded)
```

```
            image = BytesIO(img_b64decoded)
            file_name = event['filename']

        elif 'S3Object' in event:
            bucket = boto3.resource('s3').Bucket(event['S3Object']['Bucket'])
            image_object = bucket.Object(event['S3Object']['Name'])
            image = image_object.get().get('Body').read()
            image_type = get_image_type(image)
            file_name = f"s3://{event['S3Object']['Bucket']}/{event['S3Object']
['Name']}"

        else:
            raise ValueError(
                'Invalid image source. Only base 64 encoded image bytes or images
 in S3 buckets are supported.')

        # Analyze the image.
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=image_type,
            Body=image,
            ModelVersion=model_version)

        reject = reject_on_classification(
            response['DetectAnomalyResult'],
 confidence_limit=float(environ['CONFIDENCE'])/100)

        status = "anomalous" if reject else "normal"

        lambda_response = {
            "statusCode": 200,
            "body": {
                "Reject": reject,
                "RejectMessage": f"Image {file_name} is {status}."
            }
        }

    except ClientError as err:
        error_message = f"Couldn't analyze {file_name}. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
```

```
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message,
                "Image": file_name
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, error_message)

    except ValueError as val_error:

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error),
                "Image": event['filename']
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response

def get_image_type(image):
    """
    Gets the format of the image. Raises an error
    if the type is not PNG or JPEG.
    :param image: The image that you want to check.
    :return The type of the image.

    """
    image_type = imghdr.what(None, image)

    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Invalid image type")
        raise ValueError(
            "Invalid file format. Supply a jpeg or png format file.")
    return content_type
```

```
def reject_on_classification(prediction, confidence_limit):
    """
    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from DetectAnomalies
    :param confidence_limit: The minimum acceptable confidence. Float value between
 0 and 1.
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        reject = True
        reject_info = (f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) is greater"
                       f" than limit ({confidence_limit:.2%})")
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject
```

4. 選擇部署以部署您的 Lambda 函數。

步驟 4：嘗試您的 Lambda 函數

在此步驟中，您可以使用電腦上的 Python 程式碼，將本機映像或 Amazon S3 儲存貯體中的映像傳遞至 Lambda 函數。從本機電腦傳送的影像必須小於 6291456 位元組。如果您的映像檔較大，請將映像上傳到 Amazon S3 儲存貯體，然後使用 Amazon S3 路徑呼叫指令碼到映像。如需將影像檔案上傳至 Amazon S3 儲存貯體的相關資訊，請參閱上載物件。

確保你運行相同的代碼AWS您在其中建立 Lambda 函數的區域。您可以檢視AWS在函數詳細資訊頁面的導覽列中，您 Lambda 函數的區域主控台。

如果AWS Lambda函數傳回逾時錯誤，延長 Lambda 函數函數的逾時期限，如需詳細資訊，請參閱配置功能超時（控制台）。

如需有關從程式碼叫用 Lambda 函數的詳細資訊，請參閱調用AWS Lambda函数。

若要嘗試您的 Lambda 函數

1. 如果您尚未這麼做，請執行下列動作：

   a. 確保使用客戶端代碼的用戶具有`lambda:InvokeFunction`權限。您可以使用下列權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LambdaPermission",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

   您可以從中的函數概觀中獲取 Lambda 函數函數的 ARN主控台。

   若要提供存取權，請新增許可到您的使用者、群組或角色：

   - AWS IAM Identity Center 中的使用者和群組：

     建立許可集合。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。

   - 透過身分提供者在 IAM 中管理的使用者：

     建立聯合身分的角色。請遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。

   - IAM 使用者：

     - 建立您的使用者可擔任的角色。請遵循《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。
     - (不建議) 將政策直接連接至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的新增許可到使用者 (主控台)中的指示。

   b. 安裝和配置AWS開發套件的 Python。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。

   c. 啟動模型您在步驟 7 中指定的步驟 1：建立AWS Lambda功能（控制台）。

2. 將以下代碼保存到名為的文件中`client.py`。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose: Shows how to call the anomaly detection
AWS Lambda function.
"""
from botocore.exceptions import ClientError

import argparse
import logging
import base64
import json
import boto3
from os import environ

logger = logging.getLogger(__name__)

def analyze_image(function_name, image):
    """
    Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
```

```
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")
            lambda_payload = {"image": data, "filename": image}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))
    return json.loads(response['Payload'].read().decode())


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function "
        "that you want to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Analyze image and display results.

        result = analyze_image(args.function, args.image)

        status = result['statusCode']

        if status == 200:
            classification = result['body']
            print(f"classification: {classification['Reject']}")
```

```
            print(f"Message: {classification['RejectMessage']}")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


if __name__ == "__main__":
    main()
```

3. 執行程式碼。對於命令列引數，請提供 Lambda 函數名稱和要分析的本機映像路徑。例如：

```
python client.py function_name /bucket/path/image.jpg
```

如果成功，則輸出為在影像中找到的異常分類。如果未傳回分類，請考慮降低您在步驟 7 中設定的信賴度值步驟 1：建立AWS Lambda功能（控制台）。

4. 如果您已完成 Lambda 函數，且該模型未被其他應用程式使用，停止模型。請記住啟動模型下次您想要使用 Lambda 函數時。

在邊緣設備上使用您的亞馬遜 Lookout for Vision 察模型

您可以在由管理的邊緣裝置上使用您的 Amazon Lookout for Vision 模型AWS IoT Greengrass Version 2。AWS IoT Greengrass 是開放原始碼物聯網 (IoT) 邊緣執行階段和雲端服務。您可以使用它在您的裝置上建置、部署和管理 IoT 應用程式。如需詳細資訊，請參閱AWS IoT Greengrass。

您可以將已在雲端中訓練的相同 Amazon Lookout 視覺模型部署到相AWS IoT Greengrass V2容的邊緣裝置上。然後，您可以使用部署的模型在內部部署 (例如工廠現場) 執行異常偵測，而無需持續將資料串流到雲端。如此一來，您就可以將頻寬成本降至最低，並透過即時影像分析在本機偵測異常

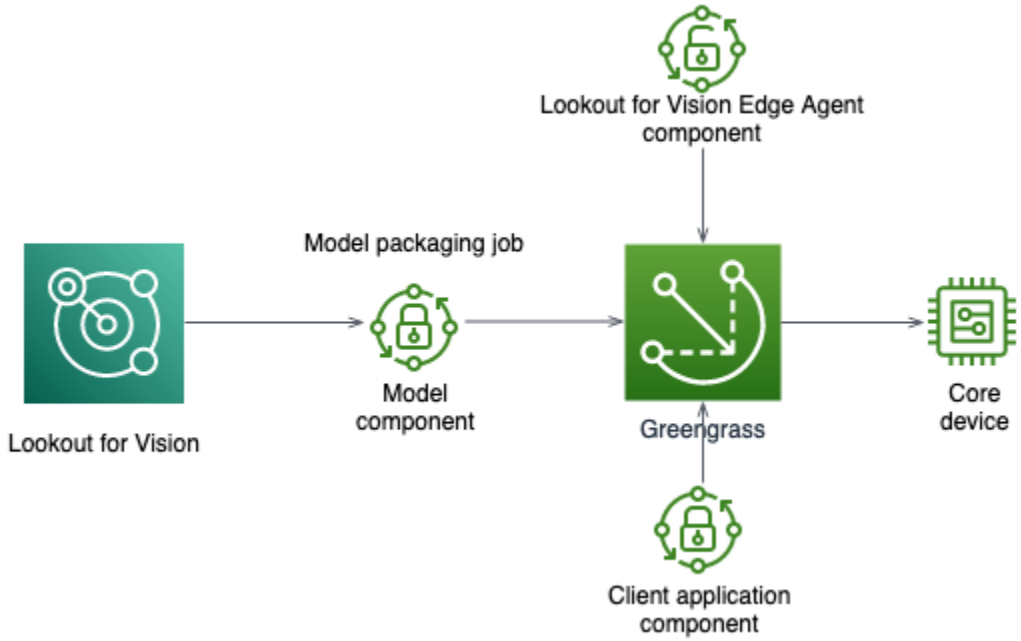
> **Tip**
>
> 在使用部署 Lookout for Vision 模型之前AWS IoT Greengrass，我們建議您閱讀AWS IoT Greengrass Version 2開發人員指南。如需詳細資訊，請參閱什麼是 AWS IoT Greengrass？ 。

若要在AWS IoT Greengrass V2核心裝置上使 Lookout for Vision 模型，您可以將模型和支援軟體做為元件部署到核心裝置。組件是在 Greengrass 核心設備上運行的軟件模塊，例如 Lookout for Vision 模型。有兩種形式的組件。自訂元件是您建立的元件，只有您可以存取。它也被稱為私有組件。提AWS供的組件是提AWS供的預構建組件。它也被稱為一個公共組件。如需詳細資訊，請參閱https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html。

您針對 Lookout for Vision 模型和支援軟體部署到核心裝置的元件為：

- 模型元件。包含 Lookout for Vision 模型的自訂元件。若要建立模型元件，您可以使 Lookout for Vision」來建立模型封裝工作。模型封裝工作會為模型建立元件，並使其可作為中的自訂元件使用 AWS IoT Greengrass V2。如需詳細資訊，請參閱包裝您的亞馬遜 Lookout for Vision 模型。
- 用戶端應用程式元件 您建立的自訂元件，可針對您的業務需求實作程式碼。例如，從組裝後拍攝的圖像中查找異常電路板。如需詳細資訊，請參閱撰寫用戶端應用程式元件。
- 亞馬遜 Lookout for Vision 邊緣代理組件。提AWS供的元件，可提供 API 以使用和管理模型。例如，用戶端應用程式元件中的程式碼可以使用 `DetectAnomalies` API 偵測影像中的異常情況。Lookout for Vision 邊緣代理組件是模型組件的依賴關係。當您部署模型元件時，它會自動安裝在核心裝置上。如需詳細資訊，請參閱亞馬遜 Lookout for Vision 邊緣代理 API 參考。

建立模型元件和用戶端應用程式元件之後，您可AWS IoT Greengrass V2以使用將元件和相依性部署到核心裝置。如需詳細資訊，請參閱將元件部署到裝置。



> **Important**
>
> 您的模型在核心裝置`DetectAnomalies`上所做的預測可能與使用雲端託管的相同模型所做的預測不同。 建議您先在核心裝置上測試模型，然後再在生產環境中使用該模型。
> 若要減少裝置託管模型與雲端託管模型之間的預測不相符，建議您增加訓練資料集中一般和異常影像的數量。我們不建議重複使用現有的映像來增加訓練資料集的大小。

將模型和用戶端應用程式元件部署到AWS IoT Greengrass Version 2核心裝置

在AWS IoT Greengrass Version 2核心裝置上部署 Amazon Lookout for Vision 模型和用戶端應用程式元件的程序如下：

1. 使用設定核心裝置AWS IoT Greengrass Version 2。
2. 使用 Lookout for Vision 建立模型封裝工作。工作會建立您的模型元件。
3. 撰寫用戶端應用程式元件。該組件實現您的業務邏輯。
4. 使用將模型元件和用戶端應用程式元件部署到核心裝置AWS IoT Greengrass V2。

將元件和相依性部署到核心裝置之後，您就可以在核心裝置上使用該模型。

> **Note**
>
> 您必須使用相同的AWS區域和AWS帳戶來建立和部署 Lookout 視覺模型和用戶端應用程式元件。

AWS IoT Greengrass Version 2核心裝置需求

若要在AWS IoT Greengrass Version 2核心裝置上使用 Amazon Lookout for Vision 模型，您的模型具有核心裝置的各種需求。

主題

經過測試的裝置、晶片架構和作業系統

我們希望亞馬遜觀景視覺能夠在以下硬件上工作：

- CPU 架構
  - X86_64 (64 位元版本的 x86 指令集)
  - 64 位元 CPU (64 位元)
- (僅適用於 GPU 加速推論) 具有足夠記憶體容量的 NVIDIA GPU 加速器 (執行中的機型至少需要 6.0 GB)。

亞馬遜 Lookout for Vision 察團隊已在以下設備、芯片架構和操作系統上測試了 Lookout for Vision 望模型。

裝置

裝置	作業系統	架構	加速器	編譯器選項
傑特森沙維爾 (NVIDIA® 傑特森 AGX 澤維爾)	Linux	Aarch64	NVIDIA	{"gpu-code": "sm_72",

裝置	作業系統	架構	加速器	編譯器選項
				"trt-ver" : "7.1.3", "cuda-ver": "10.2"} {"gpu-code": "sm_72", "trt-ver" : "8.2.1", "cuda-ver": "10.2"}
大型 (EC2 執行個體 (含 NVIDIA T4 張量核心 GPU 的 EC2 執行個體 (G4))	Linux	X86_64/X86-64	NVIDIA	{"gpu-code": "sm_75", "trt-ver" : "7.1.3", "cuda-ver": "10.2"}
大型 (含 NVIDIA A10G 張量核心 GPU 的 EC2 執行個體 (G5))	Linux	X86_64/X86-64	NVIDIA	{"gpu-code": "sm_80", "trt-ver" : "8.2.0", "cuda-ver": "11.2"}
大型 (Amazon EC2 C5 執行個體)	Linux	X86_64/X86-64	CPU	{"mcpu": "core-avx2"}

核心裝置記憶體與儲存

若要執行單一模型和 Amazon Lookout for Vision 邊緣代理程式，您的核心裝置具有下列記憶體和儲存需求。用戶端應用程式元件可能需要更多記憶體和儲存空間。

- 儲存空間 — 至少 1.5 GB。
- 記憶體 — 執行中模型至少 6.0 GB。

所需軟體

核心裝置需要下列軟體。

傑森設備

如果您的核心設備是 Jetson 設備，則需要在核心設備上安裝以下軟件。

軟體	支援的版本
噴氣背包 SDK	4 至 6
Python 和 Python 虛擬環境，用於瞭解視覺邊緣代理程式版本 1.x	或

硬體

如果您的核心裝置使用 x86 硬體，則需要在核心裝置上安裝下列軟體。

推斷

軟體	支援的版本
Python 和 Python 虛擬環境，用於瞭解視覺邊緣代理程式版本 1.x	或

GPU 加速推論

軟體版本會根據您使用的 NVIDIA GPU 的微架構而有所不同。

NVIDIA GPU 具備安培以前的微架構 (運算能力低於 8.0)

具有安培之前微架構的 NVIDIA GPU 所需的軟體 (運算能力低於 8.0)。必gpu-code須小於sm_80。

軟體	支援的版本
英偉達	10.2
英偉 TensorRT	至少為 7.1.3 且小於 8.0.0
Python 和 Python 虛擬環境，用於瞭解視覺邊緣代理程式版本 1.x	或

配備安培微架構的 NVIDIA GPU (運算能力 8.0)

具有安培微架構的 NVIDIA GPU 所需的軟體 (運算能力為 8.0)。必gpu-code須是sm_80)。

軟體	支援的版本
英偉達	11.2
英偉 TensorRT	8.2.0
Python 和 Python 虛擬環境，用於瞭解視覺邊緣代理程式版本 1.x	或

設定您的AWS IoT Greengrass Version 2核心裝置

Amazon Lookout 視覺用AWS IoT Greengrass Version 2於簡化模型元件、視覺邊緣代理程式元件的 Amazon 瞭望台，以及用戶端應用程式元件的部署到您的AWS IoT Greengrass V2核心裝置。如需有關可使用之裝置和硬體的資訊，請參閱AWS IoT Greengrass Version 2核心裝置需求。

設定您的核心裝置

請使用下列資訊來設定核心裝置。

若要設定核心裝置

1. 設定您的 GPU 資料庫。如果您不使用 GPU 加速推論，請勿執行此步驟。

a. 請確認您擁有支援 CUDA 的 GPU。如需詳細資訊，請參閱確認您擁有支援 CUDA 的 GPU。

b. 透過執行下列其中一項動作，在您的裝置上設定 CUDA、cuDNN 和 TensorRT：

- 如果您使用的是傑森裝置，請安裝 4.4-4.6.1 JetPack 版。如需詳細資訊，請參閱JetPack 封存。
- 如果您使用基於 x86 的硬體，且 NVIDIA GPU 微架構早於安培 (運算能力低於 8.0)，請執行下列動作：
  1. 依照 NVIDIA CUDA 安裝指南中的指示，設定 CUDA 10.2 版。
  2. 按照 NVIDIA cuDNN 說明文件中的說 cu DNN 進行安裝。
  3. 請依照 NVIDIA TENSORRT 說明文件中的指示設定 TensorRT (版本 7.1.3 或更新版本，但早於 8.0.0 版)。
- 如果您使用基於 x86 的硬件，並且 NVIDIA GPU 微架構是安培（計算能力為 8.0），請執行以下操作：
  1. 依照 NVIDIA CUDA 安裝指南中的指示設定 CUDA (11.2 版)。
  2. 按照 NVIDIA cuDNN 說明文件中的說 cu DNN 進行安裝。
  3. 依照 NVIDIA 網站文件中的指示，設定 TensorRT (版本 8.2.0)。

2. 在AWS IoT Greengrass Version 2核心裝置上安裝核心軟體。如需詳細資訊，請參閱開發人員指南中的安裝 AWS IoT Greengrass 核心軟體。AWS IoT Greengrass Version 2

3. 若要從存放模型的 Amazon S3 儲存貯體讀取，請將權限附加到您在AWS IoT Greengrass Version 2安裝期間建立的 IAM 角色 (權杖交換角色)。如需詳細資訊，請參閱允許存取元件成品的 S3 儲存貯體。

4. 在命令提示字元中，輸入下列指令，將 Python 和 Python 虛擬環境安裝到核心裝置上。

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. 使用以下命令將 Greengrass 用戶添加到視頻組。這可以讓 Greengrass 部署的組件訪問 GPU：

```
sudo usermod -a -G video ggc_user
```

6. (選擇性) 如果您想要從其他使用者呼叫監視視覺邊緣代理程式 API，請將所需的使用者新增至ggc_group. 這可讓使用者透過 Unix 網域套接字與 Lookout for Vision 邊緣代理進行通訊：

```
sudo usermod -a -G ggc_group $(whoami)
```

包裝您的亞馬遜 Lookout for Vision 模型

模型包裝任務將亞馬遜 Lookout for Vision 模型打包為模型組件。

若要建立模型封裝工作，請選擇要封裝的模型，並為工作建立的模型元件提供設定。您只能封裝已成功訓練的模型。

您可以使用 Lookout for Vision 主控台或 AWS SDK 來建立模型封裝工作。您也可以取得有關您建立的模型封裝工作的資訊。如需詳細資訊，請參閱取得模型封裝工作的相關資訊。您可以使用AWS IoT Greengrass V2主控台或 AWS SDK 將元件部署到AWS IoT Greengrass Version 2核心裝置。

主題

Package 設定

使用下列資訊決定模型封裝工作的套件設定。

若要建立模型封裝工作，請參閱打包您的模型（控制台）或封裝您的模型 (SDK)。

主題

目標硬體

您可以為您的型號選擇目標裝置或目標平台，但不能同時選擇兩者。如需詳細資訊，請參閱經過測試的裝置、晶片架構和作業系統。

目標裝置

該模型的目標設備，如 NVIDIA® 傑特森 AGX 澤維爾。您不需要指定編譯器選項。

目標平台

亞馬遜 Lookout for Vision 支援以下平台組態：

- X86_64 (64 位元版本的 x86 指令集) 和 Aarch64 (64 位元 CPU) 架構。
- 操作系統。
- 使用 NVIDIA 或 CPU 加速器進行推論。

您需要為目標平台指定正確的編譯器選項。

編譯器選項

編譯器選項可讓您指定AWS IoT Greengrass Version 2核心裝置的目標平台。目前您可以指定下列編譯器選項。

加速器

- `gpu-code`— 指定執行模型元件之核心裝置的 gpu 程式碼。
- `trt-ver`— 以 X.y.z. 格式指定 TensorRT 版本。
- `cuda-ver`— 以 x.y 格式指定 CUDA 版本。

CPU 加速器

- (選擇性) `mcpu` — 指定指令集。例如：`core-avx2`。如果您未提供值，則「Lookout for Vision」會使用該值`core-avx2`。

您可以使用 JSON 格式指定選項。例如：

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

如需更多範例，請參閱 經過測試的裝置、晶片架構和作業系統。

元件設定

模型封裝工作會建立包含您模型的模型元件。工作會建立AWS IoT Greengrass V2用來將模型元件部署到核心裝置的成品。

您無法建立具有與現有元件相同的元件名稱和元件版本的模型元件。

元件名稱

檢視視覺在模型封裝期間建立的模型元件名稱。您指定的元件名稱會顯示在主AWS IoT Greengrass V2控台中。您可以在為用戶端應用程式元件建立的方案中使用元件名稱。如需詳細資訊，請參閱建立用戶端應用程式元件。

元件描述

(選擇性) 模型元件的說明。

元件版本

模型元件的版本號碼。您可以接受預設版本號碼，也可以選擇自己的版本號碼。版本號必須遵循語義版本號碼系統-主要的 .minor.patch。例如，版本 1.0.0 代表元件的第一個主要發行版本。如需詳細資訊，請參閱語意版本控制 2.0.0。如果您未提供值，則 Lookout for Vision 會使用模型的版本號碼為您產生版本。

元件位置

您希望模型封裝任務儲存模型元件成品的 Amazon S3 位置。Amazon S3 儲存貯體必須位於您使用的相同 AWS 區域和 AWS 帳戶中AWS IoT Greengrass Version 2。若要建立 Amazon S3 儲存貯體，請參閱建立儲存貯體。

Tags (標籤)

您可以使用標籤識別、組織、搜尋和篩選元件。每個標籤都是由使用者定義的津要和值組成的標籤。當模型封裝工作在 Greengrass 中建立模型元件時，標籤會附加至模型元件。元件是 AWS 物聯網綠色 V2 資源。這些標籤不會附加到您的任何「Lookout for Vision」資源，例如您的模型。如需詳細資訊，請參閱標記 AWS 資源。

打包您的模型（控制台）

您可以使用 Amazon Lookout for Vision 察主控台建立模型封裝任務。

如需有關套件設定的資訊，請參閱Package 設定。

封裝模型（控制台）

1. 建立 Amazon S3 儲存貯體，或重複使用現有儲存貯體，該儲存貯體用於存放包裝工作成品 (模型元件)。
2. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
3. 選擇 Get started (開始使用)。
4. 在左側導覽窗格中，選擇 [專案]。

5. 在「專案」區段中，選擇包含您要封裝之模型的專案。
6. 在左側導覽窗格的專案名稱下，選擇 [邊緣模型套件]。
7. 在「模型封裝工作」區段中，選擇「建立模型封裝工作」。
8. 輸入套件的設定。如需詳細資訊，請參閱Package 設定。
9. 選擇「建立模型封裝工作」。
10. 等到包裝工作完成。工作的狀態為「成功」時，就會完成工作。
11. 在「模型封裝工作」區段中選擇封裝工作。
12. 選擇繼續在 Greengrass 中部署，以繼續在中部署模型元件。AWS IoT Greengrass Version 2如需詳細資訊，請參閱將元件部署到裝置。

封裝您的模型 (SDK)

您可以透過建立模型封裝工作將模型封裝為模型元件。若要建立模型封裝工作，請呼叫StartModelPackagingJobAPI。工作可能需要一段時間才能完成。要了解當前狀態，請調用DescribeModelPackagingJob並檢查響應中的`Status`字段。

如需有關套件設定的資訊，請參閱Package 設定。

下列程序說明如何使用 AWS CLI 啟動封裝工作。您可以封裝目標平台或目標裝置的模型。如需 Java 程式碼範例，請參閱StartModelPackagingJob。

若要封裝您的模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 請確定您擁有啟動模型封裝工作的正確權限。如需詳細資訊，請參閱StartModelPackagingJob。
3. 使用下列 CLI 命令為目標裝置或目標平台封裝模型。

   Target platform

   下面的 CLI 命令顯示了如何使用 NVIDIA 加速器為目標平台打包模型。

   變更下列值：

   - `project_name`到包含要打包的模型的項目的名稱。
   - `model_version`至您要封裝的模型版本。
   - (選擇性) `description` 至模型封裝工作的說明。

- architecture到運行模型組件的AWS IoT Greengrass Version 2核心設備的體系結構（ARM64或X86_64）。
- gpu_code到運行模型組件的核心設備的 gpu 代碼。
- trt_ver轉到您在核心設備上安裝的 TensorRT 版本。
- cuda_ver到您的核心設備上安裝的 CUDA 版本。
- component_name到您要在其上建立的模型元件的名稱AWS IoT Greengrass V2。
- (選擇性) component_version 封裝工作所建立之模型元件的版本。使用 major.minor.patch 格式。例如，1.0.0 代表元件的第一個主要發行版本。
- bucket到包裝任務存放模型元件成品的 Amazon S3 儲存貯體。
- prefix到包裝任務存放模型元件成品的 Amazon S3 儲存貯體內的位置。
- (選擇性) component_description 模型元件的描述。
- (可選) tag_key1 和貼附tag_key2至模型元件之標籤的關鍵字。
- (可選) tag_value2 以tag_value1及貼附至模型元件之標籤的關鍵值。

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='architecture',Accelerator='NVIDIA'},Compi
code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\":
 \"cuda_ver\"}',S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='Compon
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

例如：

```
aws lookoutvision start-model-packaging-job \
  --project-name test-project-01 \
  --model-version 1 \
  --description="Model Packaging Job for G4 Instance using TargetPlatform
 Option" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='X86_64',Accelerator='NVIDIA'},CompilerOpt
code\": \"sm_75\", \"trt-ver\": \"7.1.3\", \"cuda-ver\":
 \"10.2\"}',S3OutputLocation={Bucket='bucket',Prefix='test-project-01/
folder'},ComponentName='SampleComponentNameX86TargetPlatform',ComponentVersion='0.1.0',C
```

```
 is my component',Tags=[{Key='modelKey0',Value='modelValue'},
 {Key='modelKey1',Value='modelValue'}]}" \
  --profile lookoutvision-access
```

Target Device

使用下列 CLI 指令為目標裝置封裝模型。

變更下列值：

- `project_name`到包含要打包的模型的項目的名稱。
- `model_version`至您要封裝的模型版本。
- (選擇性) `description` 至模型封裝工作的說明。
- `component_name`到您要在其上建立的模型元件的名稱AWS IoT Greengrass V2。
- (選擇性) `component_version` 封裝工作所建立之模型元件的版本。使用 `major.minor.patch` 格式。例如，1.0.0 代表元件的第一個主要發行版本。
- `bucket`到包裝任務存放模型元件成品的 Amazon S3 儲存貯體。
- `prefix`到包裝任務存放模型元件成品的 Amazon S3 儲存貯體內的位置。
- (選擇性) `component_description` 模型元件的描述。
- (可選) `tag_key1` 和貼附`tag_key2`至模型元件之標籤的關鍵字。
- (可選) `tag_value2` 以`tag_value1`及貼附至模型元件之標籤的關鍵值。

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='pre
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

例如：

```
aws lookoutvision start-model-packaging-job \
  --project-name project_01 \
  --model-version 1 \
```

```
 --description="description" \
 --configuration
"Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='com
model component',Tags=[{Key='tag_key1',Value='tag_value1'},
{Key='tag_key2',Value='tag_value2'}]}" \
 --profile lookoutvision-access
```

4. 請注意回應 JobName 中的值。下一個步驟需要此值。例如：

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. 用DescribeModelPackagingJob於取得工作的目前狀態。變更下列項目：

   - project_name到您正在使用的項目的名稱。
   - job_name為您在上一個步驟中記下的工作名稱。

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

   如果的Status值為，則模型封裝工作即完成SUCCEEDED。如果值不同，請等待一分鐘，然後再試一次。

6. 使用繼續部署AWS IoT Greengrass V2。如需詳細資訊，請參閱將元件部署到裝置。

取得模型封裝工作的相關資訊

您可以使用 Amazon Lookout for Vision 主控台和 AWS SDK 來取得您建立之模型封裝任務的相關資訊。

主題

- 取得模型封裝工作資訊 (主控台)
- 取得模型封裝工作資訊 (SDK)

取得模型封裝工作資訊 (主控台)

取得模型封裝工作資訊 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。
4. 在「專案」區段中，選擇包含您要檢視之模型封裝工作的專案。
5. 在左側導覽窗格的專案名稱下，選擇 [邊緣模型套件]。
6. 在「模型封裝工作」區段中，選擇您要檢視的模型封裝工作。顯示模型封裝工作的詳細資訊頁面。

取得模型封裝工作資訊 (SDK)

您可以使用 AWS SDK 列出專案中的模型封裝工作，並取得有關特定模型封裝工作的資訊。

列出模型包裝工作

您可以呼叫 ListModelPackagingJobsAPI，列出專案中的模型封裝工作。回應包含ModelPackagingJobMetadata物件清單，提供每個模型封裝工作的相關資訊。還包括一個分頁令牌，如果列表不完整，您可以使用它來獲取下一組結果。

若要列出您的模型封裝工作

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下面的 CLI 命令。變`project_name`更為您要使用的專案名稱。

```
aws lookoutvision list-model-packaging-jobs \
  --project-name project_name \
  --profile lookoutvision-access
```

描述模型封裝工作

使用 DescribeModelPackagingJobAPI 取得有關模型封裝工作的資訊。響應是一個ModelPackagingDescription對象，它包括工作的當前狀態和其他信息。

描述套件

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下面的 CLI 命令。變更下列項目：

   - `project_name`到您正在使用的項目的名稱。
   - `job_name`到工作的名稱。打電話時，您會得到工作名稱（`JobName`）StartModelPackagingJob。

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

撰寫用戶端應用程式元件

用戶端應用程式元件是您撰寫的自訂AWS IoT Greengrass Version 2元件。它實現了在AWS IoT Greengrass Version 2核心設備上使用亞馬遜 Lookout for Vision 模型所需的業務邏輯。

若要存取模型，您的用戶端應用程式元件會使用 Lookout for Vision 邊緣代理程式元件。Lookout for Vision Edge 代理程式元件提供一個 API，可讓您用來分析具有模型的影像，以及管理核心裝置上的模型。

Lookout for Vision 邊緣代理程式 API 是使用 gRPC，這是一種用於進行遠端程序呼叫的通訊協定來實作。如需詳細資訊，請參閱 gRPC。要編寫代碼，您可以使用 gRPC 支持的任何語言。我們提供示例 Python 代碼。如需詳細資訊，請參閱在用戶端應用程式元件中使用模型。

> **Note**
>
> 監視視覺邊緣代理程式元件是您部署的模型元件的相依性。當您將模型元件部署到核心裝置時，它會自動部署到核心裝置。

若要撰寫用戶端應用程式元件，請執行下列動作。

1. 設定您的環境以使用 gRPC 並安裝協力廠商程式庫。

2. 撰寫程式碼以使用模型。
3. 將程式碼做為自訂元件部署到核心裝置。

如需示範如何在自訂 GStreamer 管道中執行異常偵測的範例用戶端應用程式元件，請參閱 https://github.com/awslabs/-gstreamer。aws-greengrass-labs-lookoutvision

設定您的環境

若要撰寫用戶端程式碼，您的開發環境會從遠端連線到已部署 Amazon Lookout for Vision 模型元件和相依性的AWS IoT Greengrass Version 2核心裝置。或者，您可以在核心裝置上撰寫程式碼。如需詳細資訊，請參閱 AWS IoT Greengrass 開發工具和開發 AWS IoT Greengrass 元件。

您的用戶端程式碼應該使用 gRPC 用戶端來存取視覺邊緣代理程式的亞馬遜瞭望台。本節說明如何使用 gRPC 設定開發環境，以及如何安裝`DetectAnomalies`範例程式碼所需的協力廠商相依性。

完成編寫用戶端程式碼之後，您可以建立自訂元件，並將自訂元件部署到 Edge 裝置。如需詳細資訊，請參閱建立用戶端應用程式元件。

主題

- 設定 gRPC
- 添加第三方依賴

設定 gRPC

在您的開發環境中，您需要在程式碼中使用的 GrPC 用戶端，以呼叫 Lookout for Vision 邊緣代理程式 API。若要這麼做，您可以使用檢視視覺邊緣代理程式的`.proto`服務定義檔案來建立 gRPC 虛設常式。

> **Note**
>
> 您也可以從檢視視覺邊緣代理程式應用程式套件組合中取得服務定義檔案。當安裝瞭望 Vision Edge 代理程式元件做為模型元件的相依性時，就會安裝應用程式套件組合。應用程式套件位於`/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent`。取`edge_agent_version`代為您正在使用的 Lookout for Vision 邊緣代理程式的版本。若要取得應用程式套件，您必須將 Lookout for Vision 邊緣代理程式部署至核心裝置。

若要設定 gRPC

1. 下載壓縮文件，proto.zip。壓縮檔案包含 .proTo 服務定義檔案 () edge-agent.proto。
2. 解壓縮內容。
3. 開啟命令提示字元並導覽至包含的資料夾edge-agent.proto。
4. 使用下列命令來產生 Python 用戶端介面。

   ```
   %%bash
   python3 -m pip install grpcio
   python3 -m pip install grpcio-tools
   python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=.
    edge-agent.proto
   ```

   如果命令成功，則在工作目錄中創建存根edge_agent_pb2_grpc.py和edge_agent_pb2.py。
5. 撰寫使用您模型的用戶端程式碼。如需詳細資訊，請參閱在用戶端應用程式元件中使用模型。

添加第三方依賴

範DetectAnomalies例程式碼會使用 Pillow 程式庫來處理影像。如需詳細資訊，請參閱使用影像位元組偵測異常。

使用下面的命令來安裝枕頭庫。

```
python3 -m pip install Pillow
```

在用戶端應用程式元件中使用模型

從用戶端應用程式元件使用模型的步驟類似於使用雲端託管的模型。

1. 開始執行模型。
2. 偵測影像中的異常。
3. 如果不再需要，請停止模型。

Amazon LoLookout for Vision 邊緣代理程式提供 API 來啟動模型、偵測映像中的異常，以及停止模型。您也可以使用 API 列出裝置上的模型，並取得已部署模型的相關資訊。如需詳細資訊，請參閱亞馬遜 Lookout for Vision 邊緣代理 API 參考。

您可以通過檢查 gRPC 狀態碼來獲取錯誤信息。如需詳細資訊，請參閱取得錯誤資訊。

要編寫代碼，您可以使用 gRPC 支持的任何語言。我們提供示例 Python 代碼。

主題

在客戶端應用程序組件中使用存根

使用下列程式碼，透過檢視視覺邊緣代理程式設定對模型的存取權限。

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
 with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent resources
 leakage
```

開始模型

您可以透過呼叫 StartModel API 來啟動模型。模型可能需要一段時間才能啟動。您可以通過調用來檢查當前狀態DescribeModel。如果該status字段的值為「運行」，則模型正在運行。

範例程式碼

將####取代為模型元件的名稱。

```
import time
```

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"


def start_model_if_needed(stub, model_name):
    # Starting model if needed.
    while True:
        model_description_response =
 stub.DescribeModel(pb2.DescribeModelRequest(model_component=model_name))
        print(f"DescribeModel() returned {model_description_response}")
        if model_description_response.model_description.status == pb2.RUNNING:
            print("Model is already running.")
            break
        elif model_description_response.model_description.status == pb2.STOPPED:
            print("Starting the model.")
            stub.StartModel(pb2.StartModelRequest(model_component=model_name))
            continue
        elif model_description_response.model_description.status == pb2.FAILED:
            raise Exception(f"model {model_name} failed to start")
        print(f"Waiting for model to start.")
        if model_description_response.model_description.status != pb2.STARTING:
            break
        time.sleep(1.0)


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
```

偵測異常

您可以使用 DetectAnomalies API 來偵測影像中的異常。

此DetectAnomalies作業預期影像點陣圖會以 RGB888 封裝格式傳遞。第一個字節表示紅色通道，第二個字節表示綠色通道，第三個字節表示藍色通道。如果您以不同的格式（例如 BGR）提供圖像，則來自的預測不 DetectAnomalies 正確。

根據預設，OpenCV 會將 BGR 格式用於影像點陣圖。如果您使用 OpenCV 擷取要分析的影像DetectAnomalies，則必須在將影像傳遞至之前將影像轉換為 DetectAnomalies RGB888 格式。

您提供的影像的寬度和高度尺寸DetectAnomalies必須與用於訓練模型的影像具有相同的寬度和高度尺寸。

使用影像位元組偵測異常

您可以將影像提供為影像位元組來偵測影像中的異常。在下列範例中，會從儲存在本機檔案系統中的影像擷取影像位元組。

將 *sample.jpg* 取代為您要分析的影像檔案的名稱。將####取代為模型元件的名稱。

```
import time

from PIL import Image
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"

....
# Detecting anomalies.
def detect_anomalies(stub, model_name, image_path):
    image = Image.open(image_path)
    image = image.convert("RGB")
    detect_anomalies_response = stub.DetectAnomalies(
        pb2.DetectAnomaliesRequest(
            model_component=model_name,
            bitmap=pb2.Bitmap(
                width=image.size[0],
                height=image.size[1],
                byte_data=bytes(image.tobytes())
            )
        )
    )
    print(f"Image is anomalous -
 {detect_anomalies_response.detect_anomaly_result.is_anomalous}")
    return detect_anomalies_response.detect_anomaly_result
```

```
# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
    detect_anomalies(stub, model_component_name, "sample.jpg")
```

使用共用記憶體區段偵測異常

您可以在 POSIX 共用記憶體區段中以影像位元組的形式提供影像來偵測影像中的異常。為了獲得最佳效能，建議針對 DetectAnomalies 要求使用共用記憶體。如需詳細資訊，請參閱DetectAnomalies。

停止模型

如果您不再使用該模型，StopModelAPI 將停止模型運行。

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

在裝置上列出型號

您可以使用 the section called “ListModels” API 列出部署到設備的模型。

```
models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

描述模型

您可以透過呼叫 DescribeModel API 取得部署至裝置之模型的相關資訊。使用DescribeModel對於取得模型的目前狀態很有用。例如，您需要知道模型是否正在運行，然後才能調用DetectAnomalies。如需範例程式碼，請參閱開始模型。

取得錯誤資訊

gRPC 狀態碼用於報告 API 結果。

您可以通過捕獲RpcError異常來獲取錯誤信息，如下面的例子所示。如需有關錯誤狀態碼的資訊，請參閱 API 的參考主題。

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

建立用戶端應用程式元件

您可以在產生 GrPC 虛設常式並準備好用戶端應用程式程式碼之後，建立用戶端應用程式元件。您建立的元件是您使用部署至AWS IoT Greengrass Version 2核心裝置的自訂元件AWS IoT Greengrass V2。您建立的配方會描述您的自訂元件。配方包括任何也需要部署的相依性。在這種情況下，您可以指定在中建立的模型元件包裝您的亞馬遜 Lookout for Vision 模型。如需有關元件配方的詳細資訊，請參閱AWS IoT Greengrass Version 2元件配方參考。

本主題的程序說明如何從 recipe 檔案建立用戶端應用程式元件，並將其發佈為AWS IoT Greengrass V2自訂元件。您可以使用AWS IoT Greengrass V2主控台或 AWS SDK 來發佈元件。

如需有關建立自訂元件的詳細資訊，請參閱AWS IoT Greengrass V2文件中的下列內容。

- 在您的裝置上開發和測試元件
- 創建 AWS IoT 環境組件
- 發佈元件以部署至核心裝置

主題

- 用於發佈用戶端應用程式元件的 IAM 許可
- 創建配方
- 發佈用戶端應用程式元件 (主控台)
- 發佈用戶端應用程式元件 (SDK)

用於發佈用戶端應用程式元件的 IAM 許可

若要建立和發佈用戶端應用程式元件，您需要下列 IAM 許可：

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

創建配方

在此程序中，您會建立簡單用戶端應用程式元件的方案。中的程式碼會`lookoutvision_edge_agent_example.py`列出部署至裝置的模型，並在您將元件部署到核心裝置後自動執行。若要檢視輸出，請在部署元件之後檢查元件記錄檔。如需詳細資訊，請參閱將元件部署到裝置。當您準備就緒時，請使用此程序來建立實作您商務邏輯的程式碼方案。

您可以將配方建立為 JSON 或 YAML 格式檔案。您也會將用戶端應用程式程式碼上傳到 Amazon S3 儲存貯體。

若要建立用戶端應用程式元件方案

1. 如果您尚未建立，請建立 GrPC 虛設常式檔案。如需詳細資訊，請參閱設定 gRPC。
2. 將以下代碼保存到名為的文件 `lookoutvision_edge_agent_example.py`

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent
 resources leakage

    models_list_response = stub.ListModels(
        pb2.ListModelsRequest()
    )
    for model in models_list_response.models:
        print(f"Model Details {model}")
```

3. 建立 Amazon S3 儲存貯體 (或使用現有儲存貯體) 來存放用戶端應用程式元件的來源檔案。存儲桶必須在您的AWS帳戶和您使用的同一AWS區域AWS IoT Greengrass Version 2和 Amazon Lookout for Vision。
4. 上傳`lookoutvision_edge_agent_example.py`、`edge_agent_pb2_grpc.py` and `edge_agent_pb2.py`到您在上一個步驟中建立的 Amazon S3 儲存貯體。請注意每個檔案的 Amazon S3 路徑。您已建立`edge_agent_pb2_grpc.py`並`edge_agent_pb2.py`在中設定 gRPC。
5. 在編輯器中創建以下 JSON 或 YAML 配方文件。

   - `model_component`到您的模型元件的名稱。如需詳細資訊，請參閱元件設定。
   - 將 URI 項目變更為`lookoutvision_edge_agent_example.py`、`edge_agent_pb2_grpc.py`和的 S3 路徑`edge_agent_pb2.py`。

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.lookoutvision.EdgeAgentPythonExample",
  "ComponentVersion": "1.0.0",
  "ComponentType": "aws.greengrass.generic",
  "ComponentDescription": "Lookout for Vision Edge Agent Sample Application",
  "ComponentPublisher": "Sample App Publisher",
  "ComponentDependencies": {
    "model_component": {
      "VersionRequirement": ">=1.0.0",
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install grpcio grpcio-tools protobuf Pillow",
        "run": {
          "script": "python3 {artifacts:path}/
lookoutvision_edge_agent_example.py"
        }
```

```
      },
      "Artifacts": [
        {
          "Uri": "S3 path to lookoutvision_edge_agent_example.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2_grpc.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2.py"
        }
      ]
    }
  ],
  "Lifecycle": {}
}
```

YAML

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.lookoutvison.EdgeAgentPythonExample
ComponentVersion: 1.0.0
ComponentDescription: Lookout for Vision Edge Agent Sample Application
ComponentPublisher: Sample App Publisher
ComponentDependencies:
  model_component:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: |-
        pip3 install grpcio
        pip3 install grpcio-tools
        pip3 install protobuf
        pip3 install Pillow
      run:
        script: |-
          python3 {artifacts:path}/lookout_vision_agent_example.py
    Artifacts:
      - URI: S3 path to lookoutvision_edge_agent_example.py
```

```
        - URI: S3 path to edge_agent_pb2_grpc.py
        - URI: S3 path to edge_agent_pb2.py
```

6. 將 JSON 或 YAML 檔案儲存至您的電腦。
7. 執行下列其中一項作業，建立用戶端應用程式元件：

   - 如果要使用AWS IoT Greengrass控制台，請執行發佈用戶端應用程式元件 (主控台)。
   - 如果您想要使用 AWS SDK，請執行發佈用戶端應用程式元件 (SDK)。

發佈用戶端應用程式元件 (主控台)

您可以使用AWS IoT Greengrass V2主控台來發佈用戶端應用程式元件。

若要發佈用戶端應用程式元件

1. 如果您還沒有，請執行以下操作為您的客戶端 appication 組件創建配方。創建配方
2. 請在以下位置開啟AWS IoT Greengrass主控台 https://console.aws.amazon.com/iot/
3. 在左側導覽窗格的 [Greengrass] 下，選擇 [元件]。
4. 在我的組件下選擇創建組件。
5. 如果您要使用 JSON 格式配方，請在 [建立元件] 頁面上選擇 [將方案輸入為 JSON]。如果您要使用 YAML 格式配方，請選擇「輸入方法為 YAML」。
6. 在 創建配方「配方」底下，以您在中建立的 JSON 或 YAML 方案取代現有的方案。
7. 選擇 [建立元件]。
8. 接下來，部署用戶端應用程式元件。

發佈用戶端應用程式元件 (SDK)

您可以使用 CreateComponentVersionAPI 發佈用戶端應用程式元件。

若要發佈用戶端應用程式元件 (SDK)

1. 如果您還沒有，請執行以下操作為您的客戶端 appication 組件創建配方。創建配方
2. 在命令提示字元中，輸入下列命令以建立用戶端應用程式元件。以您在中建立的 recipe 檔案的名稱取`recipe-file`代創建配方。

```
aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
```

請注意回應中的元件的 ARN。下一個步驟需要此值。

3. 使用下面的命令來獲取客戶端應用程序組件的狀態。以您在上一個步驟中記下的 ARN 取`component-arn`代。如果的`componentState`值為，則用戶端應用程式元件已就緒`DEPLOYABLE`。

```
aws greengrassv2 describe-component --arn component-arn
```

4. 接下來，部署用戶端應用程式元件。

將元件部署到裝置

若要將模型元件和用戶端應用程式元件部署到AWS IoT Greengrass Version 2核心裝置，請使用AWS IoT Greengrass V2主控台或使用 CreateDeploymentAPI。如需詳細資訊，請參閱建立部署或AWS IoT Greengrass Version 2開發人員指南中的。如需更新部署至核心裝置之元件的相關資訊，請參閱修訂部署。

主題

- 用於部署元件的 IAM 許可
- 部署您的元件 (主控台)
- 部署元件 (SDK)

用於部署元件的 IAM 許可

若要使用部署元件，AWS IoT Greengrass V2您需要下列權限：

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`並`GetDeployment`具有相關操作。如需詳細資訊，請參閱 AWS IoT 綠色 V2 定義的動作。

如需變更 IAM 許可的相關資訊，請參閱變更使用者的許可。

部署您的元件 (主控台)

使用下列程序將用戶端應用程式元件部署到核心裝置。客戶端應用程序取決於模型組件（這反過來取決 Lookout for Vision 邊緣代理）。部署用戶端應用程式元件也會啟動模型元件的部署，以及 Lookout for Vision Edge 代理程式。

> **Note**
>
> 您可以將元件加入至現有部署。您也可以將元件部署至物件群組。

若要執行此程序，您必須擁有已設定的AWS IoT Greengrass V2核心裝置。如需詳細資訊，請參閱設定您的AWS IoT Greengrass Version 2核心裝置。

若要將元件部署到裝置

1. 請在以下位置開啟AWS IoT Greengrass主控台。 https://console.aws.amazon.com/iot/
2. 在左側導覽窗格的 [Greengrass] 下，選擇 [部署]。
3. 在部署下選擇建立。
4. 在 [指定目標] 頁面上，執行下列動作：
    1. 在「部署資訊」下，輸入或修改部署的易記名稱。
    2. 在部署目標下，選取核心裝置並輸入目標名稱。
    3. 選擇下一步。
5. 在 [選取元件] 頁面上，執行下列動作：
    1. 在 [我的元件] 下，選擇用戶端應用程式元件 (`com.lookoutvison.EdgeAgentPythonExample`) 的名稱。
    2. 選擇 Next (下一步)
6. 在 [設定元件] 頁面上，保留目前的組態，然後選擇 [下一步]。
7. 在 [設定進階設定] 頁面上，保留目前的設定，然後選擇 [下一步]。
8. 在「複查」頁面上，選擇「部署」以開始部署元件。

檢查部署狀態 (主控台)

您可以從AWS IoT Greengrass V2主控台檢查部署狀態。如果您的用戶端應用程式元件使用的範例方案和程式碼the section called “建立用戶端應用程式元件”，請在部署完成後檢視用戶端應用程式元件記錄檔。如果成功，記錄檔會包含部署至元件 Lookout for Vision 模型清單。

如需使用 AWS SDK 檢查部署狀態的相關資訊，請參閱檢查部署狀態。

若要檢查部署狀態

1. 請在以下位置開啟AWS IoT Greengrass主控台 https://console.aws.amazon.com/iot/
2. 在左側導覽窗格中，選擇 [核心裝置]。
3. 在 Greengrass 核心設備下選擇您的核心設備。
4. 選擇部署索引標籤以檢視目前的部署狀態。
5. 部署成功後 (狀態為「已完成」)，請在核心裝置上開啟終端機視窗，並檢視用戶端應用程式元件記錄檔，位於`/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log`。如果您的部署使用範例方案和程式碼，則記錄會包含來自的輸出`lookoutvision_edge_agent_example.py`。例如：

```
Model Details model_component:"ModelComponent"
```

部署元件 (SDK)

使用下列程序將用戶端應用程式元件、模型元件和 Amazon Lookout for Vision Edge 代理程式部署到您的核心裝置。

1. 建立 `deployment.json`檔案以定義元件的部署規劃。這個文件應該看起來像下面的例子。

```
{
  "targetArn":"targetArn",
  "components": {
    "com.lookoutvison.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

- 在targetArn欄位中，*targetArn*以下列格式將物件或物群組的 Amazon 資源名稱 (ARN) 取代為目標部署：
  - 事情：arn:aws:iot:*region*:*account-id*:thing/*thingName*
  - 物件群組：arn:aws:iot:*region*:*account-id:*thinggroup/*thingGroupName*

2. 檢查部署目標是否具有您要修訂的現有部署。請執行下列動作：

   a. 執行下列命令以列出部署目標的部署。以目標 AWS IoT 物件或物群組的 Amazon 資源名稱 (ARN) 取targetArn代。若要取得目前 AWS 區域中物件的 ARN，請使用指令aws iot list-things。

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   回應包含一份包含目標最新部署的清單。如果回應為空白，表示目標沒有現有的部署，您可以跳至步驟 3。否則，請deploymentId從響應中復制以在下一步中使用。

   b. 執行下列命令以取得部署的詳細資料。這些詳細資料包括中繼資料、元件和工作組態。deploymentId使用上一個步驟的 ID 取代。

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   c. 將上一個命令響應中的以下任何鍵值對複製到部署.json 中。您可以變更新部署的這些值。

   - deploymentName— 部署的名稱。
   - components— 部署的元件。若要解除安裝元件，請將其從此物件中移除。
   - deploymentPolicies— 部署的原則。
   - tags— 部署的標籤。

3. 執行下列命令以在裝置上部署元件。請注意回應deploymentId中的值。

   ```
   aws greengrassv2 create-deployment \
       --cli-input-json file://path/to/deployment.json
   ```

4. 執行下列命令以取得部署狀態。變更deployment-id為您在上一個步驟中記下的值。如果的deploymentStatus值為，則部署已成功完成。COMPLETED

   ```
   aws greengrassv2 get-deployment --deployment-id deployment-id
   ```

5. 部署成功之後，請在/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log核心裝置上開啟終端機視窗，並在檢視用戶端應用程式元件記錄檔。如果您的部署使用範例方案和程式碼，則記錄會包含來自的輸出lookoutvision_edge_agent_example.py。例如：

```
Model Details model_component:"ModelComponent"
```

亞馬遜 Lookout for Vision 邊緣代理 API 參考

本節是亞馬遜觀察視覺邊緣代理程式的 API 參考。

使用模型偵測異常

您可以使用 DetectAnomalies API 在AWS IoT Greengrass Version 2核心裝置上使用執行中的模型來偵測映像中的異常情況。

取得模型資訊

取得部署至核心裝置之模型相關資訊的 API。

- ListModels
- DescribeModel

執行模型

用於啟動和停止部署到核心裝置的 Amazon Lookout for Vision 模型的 API。

- StartModel
- StopModel

DetectAnomalies

偵測提供的影像中的異常。

來自的回應DetectAnomalies包括布林預測，該預測圖像包含一個或多個異常以及預測的可信度值。如果模型是分段模型，則回應包括下列項目：

- 以獨特顏色覆蓋每種異常類型的遮色片影像。您可以將遮罩圖像存DetectAnomalies儲在共享內存中，或者將掩碼作為圖像字節返回。
- 異常類型涵蓋的影像百分比區域。
- 遮色片影像上異常類型的十六進位顏色。

> **Note**
>
> 您使用的模型DetectAnomalies必須在執行中。您可以通過調用獲取當前狀態DescribeModel。若要開始執行模型，請參閱StartModel。

DetectAnomalies支援交錯式 RGB888 格式的封裝點陣圖 (影像)。第一個字節表示紅色通道，第二個字節表示綠色通道，第三個字節表示藍色通道。如果您以不同的格式（例如 BGR）提供圖像，則來自的預測不 DetectAnomalies 正確。

根據預設，OpenCV 會將 BGR 格式用於影像點陣圖。如果您使用 OpenCV 擷取要分析的影像DetectAnomalies，則必須在將影像傳遞至之前將影像轉換為 DetectAnomalies RGB888 格式。

支援的最小影像尺寸為 64x64 像素。支援的影像尺寸上限為 4096x4096 像素。

您可以在 protobuf 消息或通過共享內存段發送圖像。將大型映像序列化到 protobuf 消息中可以顯著增加呼叫的延遲。DetectAnomalies為了達到最低的延遲，我們建議您使用共用記憶體。

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

的輸入參數DetectAnomalies。

```
message Bitmap {
  int32 width = 1;
  int32 height = 2;
  oneof data {
    bytes byte_data = 3;
    SharedMemoryHandle shared_memory_handle = 4;
  }
```

```
}
```

```
message SharedMemoryHandle {
  string name = 1;
  uint64 size = 2;
  uint64 offset = 3;
}
```

```
message AnomalyMaskParams {
SharedMemoryHandle shared_memory_handle = 2;
}
```

```
message DetectAnomaliesRequest {
string model_component = 1;
Bitmap bitmap = 2;
AnomalyMaskParams anomaly_mask_params = 3;
}
```

點陣圖

您要分析的影像DetectAnomalies。

width

圖像的寬度，以像素為單位。

height

圖像的高度，以像素為單位。

位元組資料

在原始消息中傳遞的圖像字節。

共用記憶體處理碼

在共享內存段傳遞的圖像字節。

SharedMemoryHandle

表示一個 POSIX 共享內存段。

name

POSIX 存儲器段的名稱。若要取得有關建立共用記憶體的資訊，請參閱 shm_open。

size

從偏移量開始的圖像緩衝區大小（以字節為單位）。

offset

偏移量，以字節為單位，從共享存儲器段的開始圖像緩衝區的開始。

AnomalyMaskParams

輸出異常遮罩的參數。（分段模型）。

共用記憶體處理碼

如果shared_memory_handle未提供，則包含遮罩的圖像位元組。

DetectAnomaliesRequest

模型元件

包含您要使用之模型的AWS IoT Greengrass V2元件名稱。

點陣圖

您要分析的影像DetectAnomalies。

異常遮罩參數

用於輸出掩碼的可選參數。（分段模型）。

DetectAnomaliesResponse

來自的響應DetectAnomalies。

```
message DetectAnomalyResult {
bool is_anomalous = 1;
float confidence = 2;
Bitmap anomaly_mask = 3;
repeated Anomaly anomalies = 4;
float anomaly_score = 5;
```

```
float anomaly_threshold = 6;
}
```

```
message Anomaly {
string name = 1;
PixelAnomaly pixel_anomaly = 2;
```

```
message PixelAnomaly {
float total_percentage_area = 1;
string hex_color = 2;
}
```

```
message DetectAnomaliesResponse {
  DetectAnomalyResult detect_anomaly_result = 1;
}
```

異常

代表在影像上發現的異常。（分段模型）。

name

在影像中找到的異常類型名稱。 name對應至訓練資料集中的異常類型。服務會自動將背景異常類型插入回應中 DetectAnomalies。

像素異常

涵蓋異常類型之像素遮色片的相關資訊。

PixelAnomaly

涵蓋異常類型之像素遮色片的相關資訊。（分段模型）。

總百分比面積 (_)

異常類型涵蓋的影像百分比區域。

十六進制顏色

代表影像上異常類型的十六進位顏色值。顏色會對應至訓練資料集中使用之異常類型的顏色。

DetectAnomalyResult

是異常的

指出影像是否包含異常。 `true`如果圖像包含異常。 `false`如果圖像是正常的。

confidence

在預測的準確性的信心。`DetectAnomalies` `confidence`是介於 0 和 1 之間的浮點數值。

異常遮罩

如果未提供共享記憶體處理，則包含遮罩的圖像位元組。（分段模型）。

異常

在輸入影像中找到 0 個或多個異常的清單。（分段模型）。

異常分數

一個數字，用於量化對圖像預測的異常量偏離圖像而沒有異常。 `anomaly_score``0.0`為浮點值，範圍從一般影像 (與一般影像的最低偏差) 到 1.0 (與一般影像的最大偏差)。亞馬遜 Lookout for Vision 返回一個值`anomaly_score`，即使圖像的預測是正常的。

異常閾值

一個數字（浮點數），用於確定圖像的預測分類何時為正常或異常。等於或高`anomaly_score`於值的影像`anomaly_threshold`會被視為異常。下方的`anomaly_score`值`anomaly_threshold`表示正常影像。該模型使用`anomaly_threshold`的值是由亞馬遜 Lookout for Vision，當您訓練模型時計算。您無法設定或變更 `anomaly_threshold`

狀態碼

代碼	Number	描述
OK	0	`DetectAnomalies` 成功地進行了預測
UNKNOWN (不明)	2	發生未知的錯誤。
無效參數	3	一或多個輸入參數無效。檢查錯誤消息以獲取更多詳細信息。

代碼	Number	描述
沒有找到	5	找不到具有指定名稱的模型。
資源耗盡	8	沒有足夠的資源來執行此操作。例如，Lookout for Vision 邊緣代理程式無法跟上呼叫的速率`DetectAnomalies` 。檢查錯誤消息以獲取更多詳細信息。
失敗前提條件 (_C)	9	`DetectAnomalies` 被稱為不處於 RUNNING 狀態的模型。
內部	13	發生內部錯誤。

DescribeModel

描述部署到AWS IoT Greengrass Version 2核心裝置的 Amazon Lookout for Vision 模型。

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {
  string model_component = 1;
}
```

模型元件

包含您要描述之模型的AWS IoT Greengrass V2元件名稱。

DescribeModelResponse

```
message ModelDescription {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
```

```
  string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

ModelDescription

模型元件

包含亞馬遜 Lookout for Vision 模型的AWS IoT Greengrass Version 2組件的名稱。

查找視覺模型

亞馬遜資源名稱 ARN 用於生成AWS IoT Greengrass V2組件的亞馬遜 Lookout for Vision 模型。

status

模型的目前狀態。如需詳細資訊，請參閱ModelStatus。

狀態消息

模型的狀態訊息。

狀態碼

代碼	Number	描述
OK	0	通話成功了。
UNKNOWN (不明)	2	發生未知的錯誤。
無效參數	3	一或多個輸入參數無效。檢查錯誤消息以獲取更多詳細信息。
沒有找到	5	找不到具有提供名稱的模型。
內部	13	發生內部錯誤。

ListModels

列出部署到AWS IoT Greengrass Version 2核心裝置的模型。

```
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

ListModelsRequest

```
message ListModelsRequest {}
```

ListModelsResponse

```
message ModelMetadata {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message ListModelsResponse {
  repeated ModelMetadata models = 1;
}
```

ModelMetadata

模型元件

包含亞馬遜 Lookout for Vision 模型的AWS IoT Greengrass Version 2組件的名稱。

查找視覺模型

亞馬遜資源名稱（ARN）用於生成AWS IoT Greengrass V2組件的亞馬遜 Lookout for Vision 模型。

status

模型的目前狀態。如需詳細資訊，請參閱 ModelStatus**。**

狀態消息

模型的狀態訊息。

狀態碼

代碼	Number	描述
OK	0	通話成功了。
UNKNOWN (不明)	2	發生未知的錯誤。
內部	13	發生內部錯誤。

StartModel

啟動AWS IoT Greengrass Version 2核心裝置上執行的模型。模型可能需要一段時間才能開始執行。若要檢查目前的狀態呼叫DescribeModel。如果該Status字段是模型正在運行RUNNING。

您可以同時執行的型號數量取決於核心裝置的硬體規格。

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {
  string model_component = 1;
}
```

模型元件

包含您要啟動之模型的AWS IoT Greengrass Version 2元件名稱。

StartModelResponse

```
message StartModelResponse {
  ModelStatus status = 1;
}
```

status

模型的目前狀態。響應是，STARTING如果調用成功。如需詳細資訊，請參閱ModelStatus。

狀態碼

代碼	Number	描述
OK	0	模型正在開始
UNKNOWN (不明)	2	發生未知的錯誤。
無效參數	3	一或多個輸入參數無效。檢查錯誤消息以獲取更多詳細信息。
沒有找到	5	找不到具有提供名稱的模型。
資源耗盡	8	沒有足夠的資源來執行此操作。例如，沒有足夠的記憶體來載入模型。檢查錯誤消息以獲取更多詳細信息。
失敗前提條件 (_C)	9	該方法被調用為不處於停止或失敗狀態的模型。
內部	13	發生內部錯誤。

StopModel

停止AWS IoT Greengrass Version 2核心裝置上執行的模型。 StopModel在模型停止後返回。如果回應中的Status欄位是，則模型已成功停止STOPPED。

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
```

```
}
```

模型元件

包含要停止之模型的AWS IoT Greengrass Version 2元件名稱。

StopModelResponse

```
message StopModelResponse {
  ModelStatus status = 1;
}
```

status

模型的目前狀態。響應是，STOPPED如果調用成功。如需詳細資訊，請參閱ModelStatus。

狀態碼

代碼	Number	描述
OK	0	模型正在停止。
UNKNOWN (不明)	2	發生未知的錯誤。
無效參數	3	一或多個輸入參數無效。檢查錯誤消息以獲取更多詳細信息。
沒有找到	5	找不到具有提供名稱的模型。
失敗前提條件 (_C)	9	該方法被調用的模型不處於 RUNNING 狀態。
內部	13	發生內部錯誤。

ModelStatus

部署至AWS IoT Greengrass Version 2核心裝置的模型狀態。要獲取當前狀態，請致電DescribeModel。

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

使用 Amazon LoLookout for Vision 儀表板

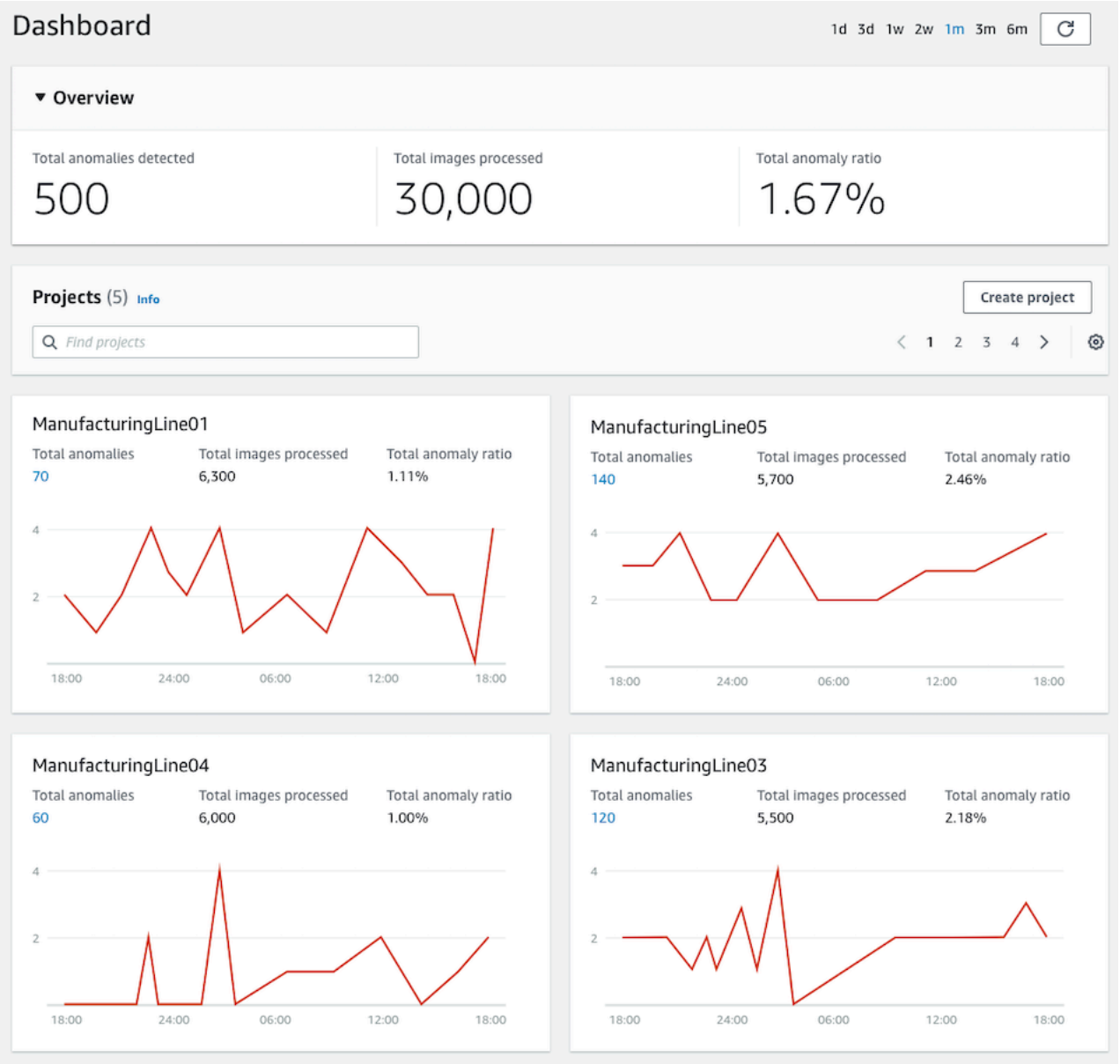
儀表板提供 Amazon Lookout for Vision 專案的指標概觀，例如上週偵測到的異常總數。使用儀表板，您可以獲得所有項目的概述以及每個單獨項目的概述。您可以選擇顯示量度的時間表。您也可以使用儀表板建立新專案。

「概觀」區段會顯示專案總數、影像總數，以及所有專案偵測到的影像總數。

「專案」區段會顯示個別專案的下列概觀資訊：

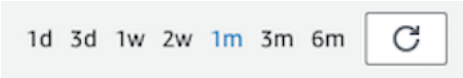
- 偵測到的異常總數。
- 處理的影像總數。
- 總異常比率 (亦即偵測到異常的影像百分比)。
- 圖表顯示所選時間範圍內的異常偵測。

您還可以獲取有關項目的更多信息。

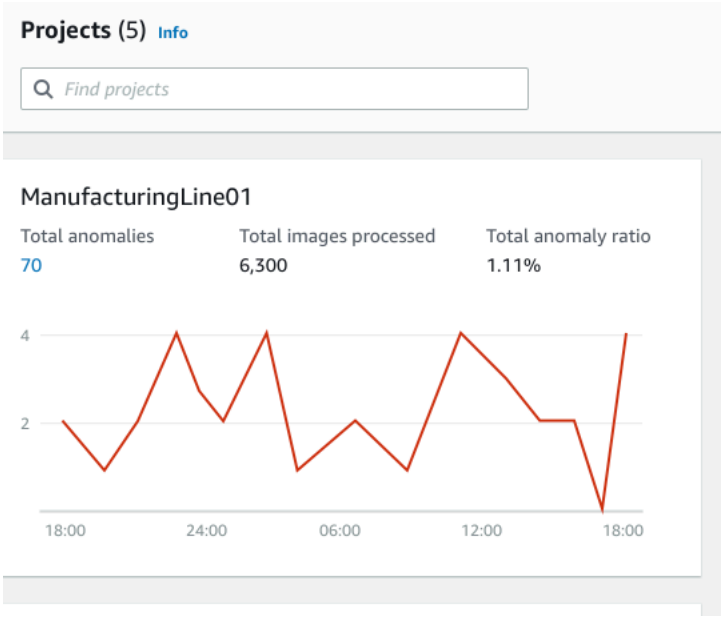


若要使用儀表板

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中選擇儀表板。
4. 若要檢視特定時間範圍內的指標，請依下列步驟執行：
   a. 選擇儀表板右上角的時間範圍。
   b. 選擇重新整理按鈕，以顯示儀表板與新時間表。



5. 若要取得有關專案的進一步詳細資訊，請在「專案」區段中選擇專案名稱 (例如**`ManufacturingLine01`**)。



6. 若要建立專案，請在 [專案] 區段中選擇 [建立專案]。

管理您的亞馬遜 Lookout for Vision 資源

您可以使用主控台或 AWS SDK 來管理您的 Amazon Lookout for Vision 資源。亞馬遜 Lookout for Vision 具有以下資源：

- 專案
- 資料集
- 模型
- 試驗偵測

> **Note**
>
> 您無法刪除試用偵測工作。此外，您無法使用 AWS SDK 來管理試用偵測。

主題

檢視您的專案

您可以從主控台或使用 AWS SDK 取得適用於視覺的 Amazon Lookout 專案清單以及個別專案的相關資訊。

> **Note**
>
> 項目列表最終是一致的。如果您建立或刪除專案，您可能需要稍等片刻，才能將專案清單保持在最新狀態。

查看您的項目（控制台）

執行下列程序中的步驟，在主控台中檢視您的專案。

若要檢視您的專案

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。將顯示專案視圖。
4. 選擇項目名稱以查看項目的詳細信息。

檢視您的專案 (SDK)

專案會針對單一使用案例管理資料集和模型。例如，偵測機器零件中的異常情況。下列範例會呼叫`ListProjects`以取得專案清單。

若要檢視您的專案 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來檢視您的專案。

   CLI

   使用指`list-projects`令列出您帳戶中的專案。

   ```
   aws lookoutvision list-projects \
     --profile lookoutvision-access
   ```

   使用指`describe-project`令取得有關專案的資訊。

project-name將的值變更為您要描述的專案名稱。

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"]
                )
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"
                    ]:
                        print(f"\ttype: {dataset['DatasetType']}")
                        print(f"\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"]
                )
                if not response_models["Models"]:
                    print("\tNo models")
```

```
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client,
                            project["ProjectName"],
                            model["ModelVersion"],
                        )


 print("-----------------------------------------------------------\n")
            print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 AWS
 * Region.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return List<ProjectMetadata> Metadata for each project.
 */
public static List<ProjectMetadata> listProjects(LookoutVisionClient lfvClient)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                        .maxResults(100)
                        .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();

        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);
```

```
        projects.stream().flatMap(r -> r.projects().stream())
                        .forEach(project -> {
                                projectMetadata.add(project);
                                logger.log(Level.INFO, project.projectName());
                        });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

}
```

刪除專案

您可以從主控台的 [專案檢視] 頁面或使用DeleteProject作業刪除專案。

不會刪除專案資料集所參照的影像。

刪除專案 (主控台)

請使用下列步驟來刪除專案。如果您使用主控台程序，則會為您刪除相關聯的模型版本和資料集。

刪除專案

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。
4. 在 [專案] 頁面上，選取您要刪除的專案。
5. 在頁面頂端，選擇 Delete (刪除)。
6. 在「刪除」對話方塊中，輸入 delete 以確認您要刪除專案。
7. 如有必要，請選擇刪除任何相關聯的資料集和模型。
8. 選擇 Delete project (刪除專案)。

刪除專案 (SDK)

您可以通過調用DeleteProject並提供要刪除的項目的名稱刪除亞馬遜觀景專案。

刪除專案之前，必須先刪除專案中的所有模型。如需詳細資訊，請參閱刪除模型 (SDK)。您還必須刪除與模型相關聯的數據集。如需詳細資訊，請參閱刪除資料集。

專案可能需要一些時間才能刪除。在此期間，專案的狀態為DELETING。如果後續呼叫DeleteProject未包含您刪除的專案，則會刪除專案。

若要刪除專案 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下面的代碼來刪除一個項目。

   AWS CLI

   project-name將的值變更為您要刪除的專案名稱。

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

   此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

   ```
       @staticmethod
       def delete_project(lookoutvision_client, project_name):
           """
           Deletes a Lookout for Vision Model

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that you want to delete.
           """
           try:
               logger.info("Deleting project: %s", project_name)
               response =
    lookoutvision_client.delete_project(ProjectName=project_name)
               logger.info("Deleted project ARN: %s ", response["ProjectArn"])
           except ClientError as err:
               logger.exception("Couldn't delete project %s.", project_name)
               raise
   ```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Deletes an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return String The ARN of the deleted project.
 */
public static String deleteProject(LookoutVisionClient lfvClient, String
 projectName)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting project: {0}", projectName);

        DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder()
                        .projectName(projectName)
                        .build();

        DeleteProjectResponse response =
 lfvClient.deleteProject(deleteProjectRequest);

        logger.log(Level.INFO, "Deleted project: {0} ARN: {1}",
                        new Object[] { projectName, response.projectArn() });

        return response.projectArn();

}
```

檢視資料集

專案可以有用於訓練和測試模型的單一資料集。或者，您可以有單獨的訓練和測試資料集。您可以使用主控台來檢視資料集。您也可以使用此DescribeDataset作業取得資料集 (訓練或測試) 的相關資訊。

檢視專案中的資料集 (主控台)

執行下列程序中的步驟，在主控台中檢視專案的資料集。

若要檢視資料集 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。
4. 在 [專案] 頁面上，選取包含您要檢視之資料集的專案。
5. 在左側導覽窗格中，選擇 [資料集] 以檢視資料集詳細資料。如果您有訓練和測試資料集，則會顯示每個資料集的索引標籤。

檢視專案中的資料集 (SDK)

您可以使用此`DescribeDataset`作業取得與專案相關聯之訓練或測試資料集的相關資訊。

若要檢視您的資料集 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來檢視資料集。

   CLI

   變更下列值：

   - `project-name`至包含您要檢視之模型的專案名稱。
   - `dataset-type`到您要查看的數據集類型（`train`或`test`）。

   ```
   aws lookoutvision describe-dataset --project-name project name\
     --dataset-type train or test \
     --profile lookoutvision-access
   ```

   Python

   此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you
 want
                             to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(f"Images: {response['DatasetDescription']['ImageStats']
['Total']}")
            print(f"Labeled: {response['DatasetDescription']['ImageStats']
['Labeled']}")
            print(f"Normal: {response['DatasetDescription']['ImageStats']
['Normal']}")
            print(f"Anomaly: {response['DatasetDescription']['ImageStats']
['Anomaly']}")
        except ClientError:
            logger.exception("Service error: problem listing datasets.")
            raise
        print("Done.")
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
```

```
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to describe a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to describe (train
 *                    or test).
 * @return DatasetDescription A description of the dataset.
 */
public static DatasetDescription describeDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType) throws LookoutVisionException {

        logger.log(Level.INFO, "Describing {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        DescribeDatasetResponse describeDatasetResponse =
 lfvClient.describeDataset(describeDatasetRequest);
        DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

        logger.log(Level.INFO, "Project: {0}\n"
                        + "Created: {1}\n"
                        + "Type: {2}\n"
                        + "Total: {3}\n"
                        + "Labeled: {4}\n"
                        + "Normal: {5}\n"
                        + "Anomalous: {6}\n",
                        new Object[] {
                                        datasetDescription.projectName(),
                                        datasetDescription.creationTimestamp(),
                                        datasetDescription.datasetType(),

 datasetDescription.imageStats().total().toString(),

 datasetDescription.imageStats().labeled().toString(),

 datasetDescription.imageStats().normal().toString(),

 datasetDescription.imageStats().anomaly().toString(),
```

```
                        });

        return datasetDescription;

}
```

將影像新增至資料集

建立資料集之後，您可能會想要在資料集中新增更多影像。例如，如果模型評估指出模型不良，您可以透過新增更多影像來增強模型的品質。如果您已建立測試資料集，則新增更多影像可以提高模型效能指標的準確性。

更新資料集後重新訓練模型。

主題

- 添加更多圖像
- 新增更多影像 (SDK)

添加更多圖像

您可以從本機電腦上傳影像，將更多影像新增至資料集。若要使用 SDK 新增更多已標記的影像，請使用此UpdateDatasetEntries作業。

若要將更多影像新增至資料集 (主控台)

1. 選擇 [動作]，然後選取要新增影像的資料集。
2. 選擇您要上傳至資料集的影像。您可以拖動圖像或選擇要從本地計算機上傳的圖像。您一次最多可以上傳 30 張圖片。
3. 選擇 [上傳圖片]。
4. 選擇 Save Changes (儲存變更)。

完成新增更多影像後，您需要為其加上標籤，以便使用它們來訓練模型。如需詳細資訊，請參閱分類映像（控制台）。

新增更多影像 (SDK)

若要使用 SDK 新增更多已標記的影像，請使用此UpdateDatasetEntries作業。您提供一個資訊清單檔案，其中包含您要新增的映像檔。您也可以透過在資訊清單檔案中 JSON 行的`source-ref`欄位中指定影像來更新現有的映像檔。如需詳細資訊，請參閱建立清單檔案。

若要將更多影像新增至資料集 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼，將更多影像新增至資料集。

   CLI

   變更下列值：

   - `project-name`到包含您要更新的數據集的項目的名稱。
   - `dataset-type`到您要更新（`train`或`test`）的數據集類型。
   - `changes`到包含數據集更新的清單文件的位置。

```
aws lookoutvision update-dataset-entries\
    --project-name project\
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

   Python

   此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def update_dataset_entries(lookoutvision_client, project_name, dataset_type,
updates_file):
        """
        Adds dataset entries to an Amazon Lookout for Vision dataset.
        :param lookoutvision_client: The Amazon Rekognition Custom Labels Boto3
client.
        :param project_name: The project that contains the dataset that you want
to update.
```

```
        :param dataset_type: The type of the dataset that you want to update
 (train or test).
        :param updates_file: The manifest file of JSON Lines that contains the
 updates.
        """

        try:
            status = ""
            status_message = ""
            manifest_file = ""

            # Update dataset entries
            logger.info(f"""Updating {dataset_type} dataset for project
 {project_name}
with entries from {updates_file}.""")

            with open(updates_file) as f:
                manifest_file = f.read()

            lookoutvision_client.update_dataset_entries(
                ProjectName=project_name,
                DatasetType=dataset_type,
                Changes=manifest_file,
            )

            finished = False
            while finished == False:

                dataset =
 lookoutvision_client.describe_dataset(ProjectName=project_name,

 DatasetType=dataset_type)

                status = dataset['DatasetDescription']['Status']
                status_message = dataset['DatasetDescription']['StatusMessage']

                if status == "UPDATE_IN_PROGRESS":
                    logger.info(
                        (f"Updating {dataset_type} dataset for project
 {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_IN_PROGRESS":
```

```
                    logger.info(
                        (f"Update failed, rolling back {dataset_type} dataset
for project {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_COMPLETE":
                    logger.info(
                        f"Dataset updated: {status} : {status_message} :
{dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_COMPLETE":
                    logger.info(
                        f"Rollback complated after update failure: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                logger.exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                raise Exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} :{dataset_type} dataset for project {project_name}.")

            logger.info(f"Added entries to dataset.")

            return status, status_message

        except ClientError as err:
            logger.exception(
                f"Couldn't update dataset: {err.response['Error']['Message']}")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
```

```
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to update a
 *                     dataset.
 * @param datasetType  The type of the dataset that you want to update (train or
 *                     test).
 * @param manifestFile The name and location of a local manifest file that you
 want to
 * use to update the dataset.
 * @return DatasetStatus The status of the updated dataset.
 */

public static DatasetStatus updateDatasetEntries(LookoutVisionClient lfvClient,
 String projectName,
                String datasetType, String updateFile) throws
 FileNotFoundException, LookoutVisionException,
                InterruptedException {

        logger.log(Level.INFO, "Updating {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        InputStream sourceStream = new FileInputStream(updateFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
 UpdateDatasetEntriesRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .changes(sourceBytes)
                        .build();

        lfvClient.updateDatasetEntries(updateDatasetEntriesRequest);

        boolean finished = false;
        DatasetStatus status = null;

        // Wait until update completes.

        do {

                DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
```

```
                                .build();
                DescribeDatasetResponse describeDatasetResponse = lfvClient
                                .describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                status = datasetDescription.status();

                switch (status) {

                        case UPDATE_COMPLETE:
                                logger.log(Level.INFO, "{0} Dataset updated for
project {1}.",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        case UPDATE_IN_PROGRESS:

                                logger.log(Level.INFO, "{0} Dataset update for
project {1} in progress.",
                                                new Object[] { datasetType,
projectName });
                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_IN_PROGRESS:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rolling back",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_COMPLETE:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rollback completed.",
```

```
                                                    new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                    "{0} Dataset update failed for
project {1}. Unexpected error returned.",
                                                    new Object[] { datasetType,
projectName });
                                finished = true;

                }

        } while (!finished);

        return status;

}
```

3. 重複上一個步驟，並提供其他資料集類型的值。

從資料集中移除影像

您無法直接從資料集刪除影像。相反，您必須刪除現有的資料集，然後建立不含要移除之影像的新資料集。移除映像的方式取決於您將映像匯入現有資料集的方式 (資訊清單檔案、Amazon S3 儲存貯體或本機電腦)。

您也可以使用 AWS SDK 移除影像。這在建立沒有影像分割資訊清單檔案的影像分割模型時非常有用，因此不必使用 Amazon Lookout to Vision 主控台重繪影像遮罩。

主題

- 從資料集移除影像 (主控台)
- 從資料集移除影像 (SDK)

從資料集移除影像 (主控台)

使用下列程序，透過 Amazon 觀察視覺主控台移除資料集中的映像。

若要從資料集中移除影像 (主控台)

1. 開啟專案的資料集庫。
2. 請記下您要移除的每個影像的名稱。
3. 刪除現有的資料集。
4. 執行下列任意一項：
   - 如果您使用資訊清單檔案建立資料集，請執行下列動作：
     a. 在文字編輯器中，開啟您用來建立資料集的資訊清單檔案。
     b. 移除您在步驟 2 中記下的每個影像的 JSON 行。您可以透過檢查`source-ref`欄位來識別影像的 JSON 行。
     c. 儲存資訊清單檔案。
     d. 使用更新的資訊清單檔案建立新資料集。
   - 如果您是從 Amazon S3 儲存貯體匯入的映像建立資料集，請執行下列動作：
     a. 從 Amazon S3 儲存貯體刪除您在步驟 2 中記下的映像。
     b. 使用 Amazon S3 儲存貯體中剩餘的映像建立新資料集。如果您依資料夾名稱分類影像，則不需要在下一個步驟分類影像。
     c. 執行下列任意一項：
        - 如果您要建立影像分類模型，請將每個未標籤的影像分類。
        - 如果您要建立影像分割模型，請對每個未標籤的影像進行分類和區段。
   - 如果您是從本機電腦匯入的影像建立資料集，請執行下列動作：
     a. 在電腦上，建立包含您要使用之影像的資料夾。請勿包含您要從資料集中移除的影像。如需詳細資訊，請參閱使用儲存在本機電腦上的影像建立資料集。
     b. 使用您在步驟 4.a 中建立的資料夾中的影像建立資料集。
     c. 執行下列任意一項：
        - 如果您要建立影像分類模型，請將每個未標籤的影像分類。
        - 如果您要建立影像分割模型，請對每個未標籤的影像進行分類和區段。
5. 訓練模型。

從資料集移除影像 (SDK)

您可以使用 AWS SDK 從資料集中移除影像。

若要從資料集中移除影像 (SDK)

1. 開啟專案的資料集庫。
2. 請記下您要移除的每個影像的名稱。
3. 使用作業匯出資料集的 JSON ListDatasetEntries行。
4. 使用導出的 JSON 行創建一個清單文件。
5. 在文字編輯器中，開啟資訊清單檔案。
6. 移除您在步驟 2 中記下的每個影像的 JSON 行。您可以透過檢查`source-ref`欄位來識別影像的 JSON 行。
7. 儲存資訊清單檔案。
8. 刪除現有的資料集。
9. 使用更新的資訊清單檔案建立新資料集。
10. 訓練模型。

刪除資料集

您可以使用主控台或`DeleteDataset`作業從專案中刪除資料集。不會刪除資料集所參照的影像。如果您從具有訓練和測試資料集的專案中刪除測試資料集，專案會還原為單一資料集專案，而在訓練期間會分割剩餘的資料集，以建立訓練和測試資料集。如果您刪除訓練資料集，則必須先建立新的訓練資料集，才能在專案中訓練模型。

刪除資料集 (主控台)

執行下列程序中的步驟來刪除資料集。如果您刪除專案中的所有資料集，則會顯示「建立資料集」頁面。

若要刪除資料集 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。

4. 在 [專案] 頁面上，選取包含您要刪除之資料集的專案。
5. 在左側導覽窗格中，選擇 [資料集]。
6. 選擇 [動作]，然後選取要刪除的資料集。
7. 在 [刪除] 對話方塊中，輸入 delete 以確認您要刪除資料集。
8. 選擇 [刪除訓練資料集] 或 [刪除測試資料集] 以刪除資料集

刪除資料集 (SDK)

使用此DeleteDataset作業刪除資料集。

若要刪除資料集 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來刪除模型。

   CLI

   變更下列項目的值

   - project-name到包含您要刪除的模型的專案名稱。
   - dataset-type到train或test，具體取決於您要刪除的數據集。如果您有單一資料集專案，請指train定刪除資料集。

   ```
   aws lookoutvision delete-dataset --project-name project name\
     --dataset-type dataset type \
     --profile lookoutvision-access
   ```

   Python

   此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

   ```
       @staticmethod
       def delete_dataset(lookoutvision_client, project_name, dataset_type):
           """
           Deletes a Lookout for Vision dataset

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
   ```

```
        :param project_name: The name of the project that contains the dataset
 that
                             you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                             want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type,
 project_name
            )
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to delete a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to delete (train or
 *                    test).
 * @return Nothing.
 */
public static void deleteDataset(LookoutVisionClient lfvClient, String
 projectName, String datasetType)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder()
```

```
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        lfvClient.deleteDataset(deleteDatasetRequest);

        logger.log(Level.INFO, "Deleted {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

}
```

從專案匯出資料集 (SDK)

您可以使用AWS開發套件將資料集從亞馬遜 Lookout for Vision 專案匯出到 Amazon S3 儲存貯體位置。

透過匯出資料集，您可以執行諸如使用來源專案資料集副本建立 Lookout of Vision 專案之類的工作。您也可以建立用於特定模型版本之資料集的快照。

此程序中的 Python 程式碼會將專案的訓練資料集 (資訊清單和資料集映像) 匯出到您指定的目的地 Amazon S3 位置。如果專案中存在，程式碼也會匯出測試資料集資訊清單和資料集影像。目的地可以位於與來源專案相同的 Amazon S3 儲存貯體，也可以位於不同的 Amazon S3 儲存貯體中。程式碼會使用ListDatasetEntries作業取得資料集資訊清單檔案。Amazon S3 操作會將資料集映像和更新的資訊清單檔案複製到目的地 Amazon S3 位置。

此程序示範如何匯出專案的資料集。同時也會示範如何使用匯出的資料集建立新專案。

若要從專案匯出資料集 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 決定資料集匯出的目的地 Amazon S3 路徑。確保目的地位於亞馬遜觀景支持的AWS區域。若要建立新的 Amazon S3 儲存貯體，請參閱建立儲存貯體。
3. 確保使用者具有資料集匯出目的地 Amazon S3 路徑的存取權限，以及來源專案資料集中映像檔案的 S3 位置。您可以使用下列原則，假設影像檔案可以位於任何位置。將值區/##取代為資料集匯出的目的地值區和路徑。

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

```
    "Statement": [
        {
            "Sid": "PutExports",
            "Effect": "Allow",
            "Action": [
                "S3:PutObjectTagging",
                "S3:PutObject"
            ],
            "Resource": "arn:aws:s3:::bucket/path/*"
        },
        {
            "Sid": "GetSourceRefs",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion"
            ],
            "Resource": "*"
        }
    ]
}
```

若要提供存取權，請新增許可到您的使用者、群組或角色：

- AWS IAM Identity Center 中的使用者和群組：

  建立許可集合。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。

- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。

- IAM 使用者：
  - 建立您的使用者可擔任的角色。請遵循《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。
  - (不建議) 將政策直接連接至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的新增許可到使用者 (主控台)中的指示。

4. 將以下代碼保存到名為的文件中`dataset_export.py`。

```
"""
Purpose

Shows how to export the datasets (manifest files and images)
from an Amazon Lookout for Vision project to a new Amazon
S3 location.
"""

import argparse
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_file(s3_resource, source_file, destination_file):
    """
    Copies a file from a source Amazon S3 folder to a destination
    Amazon S3 folder.
    The destination can be in a different S3 bucket.
    :param s3: An Amazon S3 Boto3 resource.
    :param source_file: The Amazon S3 path to the source file.
    :param destination_file: The destination Amazon S3 path for
    the copy operation.
    """

    source_bucket, source_key = source_file.replace("s3://", "").split("/", 1)
    destination_bucket, destination_key = destination_file.replace("s3://",
 "").split(
        "/", 1
    )

    try:
        bucket = s3_resource.Bucket(destination_bucket)
        dest_object = bucket.Object(destination_key)
        dest_object.copy_from(CopySource={"Bucket": source_bucket, "Key":
 source_key})
        dest_object.wait_until_exists()
        logger.info("Copied %s to %s", source_file, destination_file)
    except ClientError as error:
        if error.response["Error"]["Code"] == "404":
```

```
            error_message = (
                f"Failed to copy {source_file} to "
                f"{destination_file}. : {error.response['Error']['Message']}"
            )
            logger.warning(error_message)
            error.response["Error"]["Message"] = error_message
        raise


def upload_manifest_file(s3_resource, manifest_file, destination):
    """
    Uploads a manifest file to a destination Amazon S3 folder.
    :param s3: An Amazon S3 Boto3 resource.
    :param manifest_file: The manifest file that you want to upload.
    :destination: The Amazon S3 folder location to upload the manifest
    file to.
    """

    destination_bucket, destination_key = destination.replace("s3://",
 "").split("/", 1)

    bucket = s3_resource.Bucket(destination_bucket)

    put_data = open(manifest_file, "rb")
    obj = bucket.Object(destination_key + manifest_file)

    try:
        obj.put(Body=put_data)
        obj.wait_until_exists()
        logger.info("Put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name
        )
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()


def get_dataset_types(lookoutvision_client, project):
    """
```

```
    Determines the types of the datasets (train or test) in an
    Amazon Lookout for Vision project.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to check.
    :return: The dataset types in the project.
    """

    try:
        response = lookoutvision_client.describe_project(ProjectName=project)

        datasets = []

        for dataset in response["ProjectDescription"]["Datasets"]:
            if dataset["Status"] in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                datasets.append(dataset["DatasetType"])
        return datasets

    except lookoutvision_client.exceptions.ResourceNotFoundException:
        logger.exception("Project %s not found.", project)
        raise


def process_json_line(s3_resource, entry, dataset_type, destination):
    """
    Creates a JSON line for a new manifest file, copies image and mask to
    destination.
    :param s3_resource: An Amazon S3 Boto3 resource.
    :param entry: A JSON line from the manifest file.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest
    file and dataset images.
    :return: A JSON line with details for the destination location.
    """
    entry_json = json.loads(entry)

    print(f"source: {entry_json['source-ref']}")

    # Use existing folder paths to ensure console added image names don't clash.
    bucket, key = entry_json["source-ref"].replace("s3://", "").split("/", 1)
    logger.info("Source location: %s/%s", bucket, key)

    destination_image_location = destination + dataset_type + "/images/" + key
```

```
    copy_file(s3_resource, entry_json["source-ref"], destination_image_location)

    # Update JSON for writing.
    entry_json["source-ref"] = destination_image_location

    if "anomaly-mask-ref" in entry_json:
        source_anomaly_ref = entry_json["anomaly-mask-ref"]
        mask_bucket, mask_key = source_anomaly_ref.replace("s3://", "").split("/",
 1)

        destination_mask_location = destination + dataset_type + "/masks/" +
 mask_key
        entry_json["anomaly-mask-ref"] = destination_mask_location

        copy_file(s3_resource, source_anomaly_ref, entry_json["anomaly-mask-ref"])

    return entry_json


def write_manifest_file(
    lookoutvision_client, s3_resource, project, dataset_type, destination
):
    """
    Creates a manifest file for a dataset. Copies the manifest file and
    dataset images (and masks, if present) to the specified Amazon S3 destination.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to use.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest file
    and dataset images.
    """

    try:
        # Create a reusable Paginator
        paginator = lookoutvision_client.get_paginator("list_dataset_entries")

        # Create a PageIterator from the Paginator
        page_iterator = paginator.paginate(
            ProjectName=project,
            DatasetType=dataset_type,
            PaginationConfig={"PageSize": 100},
        )
```

```
        output_manifest_file = dataset_type + ".manifest"

        # Create manifest file then upload to Amazon S3 with images.
        with open(output_manifest_file, "w", encoding="utf-8") as manifest_file:
            for page in page_iterator:
                for entry in page["DatasetEntries"]:
                    try:
                        entry_json = process_json_line(
                            s3_resource, entry, dataset_type, destination
                        )

                        manifest_file.write(json.dumps(entry_json) + "\n")

                    except ClientError as error:
                        if error.response["Error"]["Code"] == "404":
                            print(error.response["Error"]["Message"])
                            print(f"Excluded JSON line: {entry}")
                        else:
                            raise
        upload_manifest_file(
            s3_resource, output_manifest_file, destination + "datasets/"
        )

    except ClientError:
        logger.exception("Problem getting dataset_entries")
        raise


def export_datasets(lookoutvision_client, s3_resource, project, destination):
    """
    Exports the datasets from an Amazon Lookout for Vision project to a specified
    Amazon S3 destination.
    :param project: The Lookout for Vision project that you want to use.
    :param destination: The destination Amazon S3 folder for the exported datasets.
    """
    # Add trailing backslash, if missing.
    destination = destination if destination[-1] == "/" else destination + "/"

    print(f"Exporting project {project} datasets to {destination}.")

    # Get each dataset and export to destination.

    dataset_types = get_dataset_types(lookoutvision_client, project)
    for dataset in dataset_types:
```

```
        logger.info("Copying %s dataset to %s.", dataset, destination)

        write_manifest_file(
            lookoutvision_client, s3_resource, project, dataset, destination
        )

    print("Exported dataset locations")
    for dataset in dataset_types:
        print(f"   {dataset}: {destination}datasets/{dataset}.manifest")

    print("Done.")


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument("project", help="The project that contains the dataset.")
    parser.add_argument("destination", help="The destination Amazon S3 folder.")


def main():
    """
    Exports the datasets from an Amazon Lookout for Vision project to a
    destination Amazon S3 location.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    try:
        session = boto3.Session(profile_name="lookoutvision-access")
        lookoutvision_client = session.client("lookoutvision")
        s3_resource = session.resource("s3")

        export_datasets(
            lookoutvision_client, s3_resource, args.project, args.destination
        )
    except ClientError as err:
        logger.exception(err)
```

```
        print(f"Failed: {format(err)}")


if __name__ == "__main__":
    main()
```

5. 執行程式碼。提供下列命令列引數：

   - project — 包含您要匯出之資料集的來源專案名稱。
   - 目的地 — 資料集的目的地 Amazon S3 路徑。

   例如：`python dataset_export.py myproject s3://bucket/path/`

6. 請注意程式碼所顯示的資訊清單檔案位置。您在步驟 8 中需要它們。
7. 按照中的說明，使用導出的數據集創建一個新的 Lookout for Vision 項目建立您的專案。
8. 執行下列任意一項：

   - 使用 LoLookout for Vision 主控台，依照中的指示，為您的新專案建立資料集使用清單文件（控制台）創建數據集。您不需要執行步驟 1—6。

     對於步驟 12，請執行以下操作：

     a. 如果來源專案有測試資料集，請選擇 [分離訓練和測試資料設定者]，否則請選擇單一資料集。

     b. 對於 .manifest 檔案位置，請輸入您在步驟 6 中記下的適當資訊清單檔案 (訓練或測試) 的位置。

   - 使用此CreateDataset作業，使用位於的程式碼為您的新專案建立資料集使用資訊清單檔案 (SDK) 建立資料集。對於`manifest_file`參數，請使用您在步驟 6 中記下的資訊清單檔案位置。如果來源專案有測試資料集，請再次使用程式碼建立測試資料集。

9. 如果您已準備就緒，請依照中的指示來訓練模型培訓您的模型。

檢視您的模型

一個專案可以有多個模型版本。您可以使用主控台來檢視專案中的模型。您也可以使用該`ListModels`操作。

> **Note**
>
> 模型列表最終是一致的。如果您建立模型，您可能需要稍等片刻，才能更新模型清單。

查看您的模型（控制台）

執行下列程序中的步驟，在主控台中檢視專案的模型。

若要檢視您的型號 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。
4. 在「專案」頁面上，選取包含您要檢視之模型的專案。
5. 在左側導覽窗格中，選擇「模型」，然後檢視模型詳細資料。

檢視您的模型 (SDK)

要查看模型的版本，請使用該`ListModels`操作。若要取得有關特定模型版本的資訊，請使用此`DescribeModel`作業。下列範例會列出專案中的所有模型版本，然後顯示個別模型版本的效能和輸出組態資訊。

若要檢視您的模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼列出您的模型並取得有關模型的資訊。

   CLI

   使用`list-models`指令列示專案中的模型。

   變更下列值：

   - `project-name`至包含您要檢視之模型的專案名稱。

```
aws lookoutvision list-models --project-name project name \
```

```
  --profile lookoutvision-access
```

使用指describe-model令取得有關模型的資訊。變更下列值：

- project-name至包含您要檢視之模型的專案名稱。
- model-version到您要描述的模型版本。

```
aws lookoutvision describe-model --project-name project name\
    --model-version model version \
   --profile lookoutvision-access
```

Python

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
    @staticmethod
    def describe_models(lookoutvision_client, project_name):
        """
        Gets information about all models in a Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response =
 lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"]
                )
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
 * Lists the models in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the models that
 *                    you want to list.
 * @return List <Metadata> A list of models in the project.
 */
public static List<ModelMetadata> listModels(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

        ListModelsRequest listModelsRequest = ListModelsRequest.builder()
                        .projectName(projectName)
                        .build();

        // Get a list of models in the supplied project.
        ListModelsResponse response = lfvClient.listModels(listModelsRequest);

        for (ModelMetadata model : response.models()) {
                logger.log(Level.INFO, "Model ARN: {0}\nVersion: {1}\nStatus:
 {2}\nMessage: {3}", new Object[] {
                                        model.modelArn(),
                                        model.modelVersion(),
                                        model.statusMessage(),
                                        model.statusAsString() });
        }

        return response.models();

}
```

刪除模型

您可以使用控制台或使用DeleteModel操作來刪除模型的版本。您無法刪除正在執行或正在訓練的模型版本。

如果模型正在執行版本，請先使用該StopModel作業停止模型版本。如需詳細資訊，請參閱停止您的亞馬遜 Lookout for Vision 模型。如果模型正在訓練，請等到完成後再刪除模型。

刪除模型可能需要幾秒鐘的時間。要確定模型是否已被刪除，請調用ListProjects並檢查 model (ModelVersion) 的版本是否在Models數組中。

刪除模型（控制台）

執行下列步驟，從主控台刪除模型。

刪除模型（控制台）

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [專案]。
4. 在「專案」頁面上，選取包含您要刪除之模型的專案。
5. 在左側導覽窗格中選擇 Models (模型)。
6. 在模型視圖中，選擇要刪除的模型的單選按鈕。
7. 在頁面頂端，選擇 Delete (刪除)。
8. 在「刪除」對話方塊中，輸入 delete 以確認您要刪除模型。
9. 選擇刪除模型以刪除模型。

刪除模型 (SDK)

使用下列步驟來透過DeleteModel操作刪除模型。

若要刪除模型 (SDK)

1. 如果您尚未這樣做，請安裝並設定AWS CLI和 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。
2. 使用下列範例程式碼來刪除模型。

   CLI

   變更下列值：

   - project-name到包含您要刪除的模型的專案名稱。
   - model-version至您要刪除的模型版本。

```
aws lookoutvision delete-model --project-name project name\
```

```
  --model-version model version \
  --profile lookoutvision-access
```

Python

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
   @staticmethod
   def delete_model(lookoutvision_client, project_name, model_version):
       """
       Deletes a Lookout for Vision model. The model must first be stopped and
can't
       be in training.

       :param lookoutvision_client: A Boto3 Lookout for Vision client.
       :param project_name: The name of the project that contains the desired
model.
       :param model_version: The version of the model that you want to delete.
       """
       try:
           logger.info("Deleting model: %s", model_version)
           lookoutvision_client.delete_model(
               ProjectName=project_name, ModelVersion=model_version
           )

           model_exists = True
           while model_exists:
               response =
lookoutvision_client.list_models(ProjectName=project_name)

               model_exists = False
               for model in response["Models"]:
                   if model["ModelVersion"] == model_version:
                       model_exists = True

               if model_exists is False:
                   logger.info("Model deleted")
               else:
                   logger.info("Model is being deleted...")
                   time.sleep(2)

           logger.info("Deleted Model: %s", model_version)
       except ClientError:
```

```
            logger.exception("Couldn't delete model.")
            raise
```

Java V2

此代碼取自AWS文檔 SDK 示例 GitHub 存儲庫。請參閱此處的完整範例。

```
/**
* Deletes an Amazon Lookout for Vision model.
*
* @param lfvClient    An Amazon Lookout for Vision client. Returns after the
 model is deleted.
* @param projectName  The name of the project that contains the model that you
 want to delete.
* @param modelVersion The version of the model that you want to delete.
* @return void
*/
public static void deleteModel(LookoutVisionClient lfvClient,
                String projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        lfvClient.deleteModel(deleteModelRequest);

        boolean deleted = false;

        do {

                ListModelsRequest listModelsRequest =
 ListModelsRequest.builder()
                                .projectName(projectName)
                                .build();

                // Get a list of models in the supplied project.
                ListModelsResponse response =
 lfvClient.listModels(listModelsRequest);
```

```
                ModelMetadata modelMetadata = response.models().stream()
                                .filter(model ->
model.modelVersion().equals(modelVersion)).findFirst()
                                .orElse(null);

                if (modelMetadata == null) {
                        deleted = true;
                        logger.log(Level.INFO, "Deleted: Model version {0} of
project {1}.",
                                        new Object[] { modelVersion,
projectName });

                } else {
                        logger.log(Level.INFO, "Not yet deleted: Model version
{0} of project {1}.",
                                        new Object[] { modelVersion,
projectName });
                        TimeUnit.SECONDS.sleep(60);
                }

        } while (!deleted);

}
```

標記模型

您可以使用標籤識別、組織、搜尋和篩選 Amazon Lookout for Vision 模型。每個標籤都是由使用者定義的津要和值組成的標籤。例如，若要協助判斷模型的計費方式，您可以使用Cost center金鑰標記模型，並新增適當的成本中心編號作為值。如需詳細資訊，請參閱標記 AWS 資源。

使用標籤可以：

- 使用成本分配標籤追蹤模型的帳單。如需詳細資訊，請參閱使用成本分配標籤。
- 使用 Identity and Access Management (IAM) 控制對模型的存取。如需詳細資訊，請參閱使用資源標籤控制 AWS 資源的存取。
- 自動化模型管理。例如，您可以執行自動啟動或停止指令碼，在非上班時間關閉開發模型以降低成本。如需詳細資訊，請參閱運行訓練有素的亞馬遜 Lookout for Vision 模型。

您可以使用 Amazon Lookout for Vision 主控台或使用 AWS SDK 來標記模型。

主題

標記模型（控制台）

您可以使用 Amazon Lookout 視覺主控台為模型新增標籤、檢視附加至模型的標籤，以及移除標籤。

新增或移除標籤 (主控台)

此程序說明如何在現有模型中新增標籤或從中移除標籤。您也可以在訓練時將標籤新增至新模型。如需詳細資訊，請參閱培訓您的模型。

將標籤新增至現有模型，或從中移除標籤 (主控台)

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在導覽窗格中，選擇 Projects (專案)。
4. 在 [專案資源] 頁面上，選擇包含您要標記之模型的專案。
5. 在導覽窗格中先前選擇的專案下，選擇「模型」。
6. 在「模型」區段中，選擇您要新增標籤的模型。
7. 在模型的詳細資訊頁面上，選擇「標籤」標籤。
8. 在 Tags (標籤) 區段中，選擇 Manage tags (管理標籤)。
9. 在「管理標籤」頁面上，選擇「新增標籤」。
10. 輸入一個鍵和一個值。
    a. 在金鑰中，輸入金鑰的名稱。
    b. 針對 Value (值)，輸入值。
11. 若要新增更多標籤，請重複步驟 9 和 10。
12. (選擇性) 若要移除標記，請在您要移除的標籤旁選擇「移除」。如果您要移除先前儲存的標籤，則會在您儲存變更時移除該標籤。
13. 請選擇 Save changes (儲存變更) 儲存您所做的變更。

檢視模型標籤 (主控台)

您可以使用亞馬遜 Lookout for Vision 控制台來查看附加到模型的標籤。

若要檢視專案內所有模型附加的標籤，您必須使用 AWS 開發套件。如需詳細資訊，請參閱列出模型標籤 (SDK)。

檢視貼附至模型的標籤

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在導覽窗格中，選擇 Projects (專案)。
4. 在 [專案資源] 頁面上，選擇包含您要檢視其標籤之模型的專案。
5. 在導覽窗格中先前選擇的專案下，選擇「模型」。
6. 在「模型」區段中，選擇您要檢視其標籤的模型。
7. 在模型的詳細資訊頁面上，選擇「標籤」標籤。標籤會顯示在「標籤」區段中。

標記模型 (SDK)

您可以使用 AWS SDK 來：

- 新增標籤至新模型
- 將標籤新增至現有模型
- 列出貼附至模型的標籤
- 從模型中移除標籤

本節包括AWS CLI範例。如果您尚未安裝AWS CLI，請參閱步驟 4：設定 AWS CLI 和 AWS 軟體開發套件。

將標籤新增至新模型 (SDK)

您可以在使用CreateModel操作建立模型時將標籤新增至模型。在Tags陣列輸入參數中指定一或多個標籤。

```
aws lookoutvision create-model --project-name "project name"\
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }'\
```

```
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

如需建立和訓練模型的資訊，請參閱培訓模型 (SDK)。

將標籤新增至現有模型 (SDK)

若要將一個或多個標籤新增至現有模型，請使用此TagResource作業。指定模型的 Amazon 資源名稱 (ARN) (ResourceArn) 和您要新增的標籤 (Tags)。

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

如需 Java 程式碼範例，請參閱TagModel。

列出模型標籤 (SDK)

要列出附加到模型的標籤，請使用該ListTagsForResource操作並指定模型的 Amazon 資源名稱（ARN），（ResourceArn）。響應是附加到指定模型的標籤鍵和值的映射。

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

要查看項目中哪些模型具有特定標籤，請調用ListModels以獲取模型列表。然後在響應中調ListTagsForResource用每個模型ListModels。檢查來源的響應ListTagsForResource以查看是否存在所需的標籤。

如需 Java 程式碼範例，請參閱ListModelTags。如需在所有專案中搜尋標籤值的 Python 程式碼，請參閱 find_tag.py。

從模型中移除標籤 (SDK)

若要從模型中移除一個或多個標籤，請使用此UntagResource作業。指定模型的 Amazon 資源名稱 (ARN) (ResourceArn) 和要移除的標籤金鑰 (Tag-Keys)。

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

如需 Java 程式碼範例，請參閱UntagModel。

檢視您的試用偵測工作

您可以使用主控台檢視您的試用偵測。您無法使用 AWS SDK 來檢視試用偵測工作。

> **Note**
>
> 試驗偵測清單最終是一致的。如果您建立試用偵測，您可能需要稍等片刻，試驗偵測清單才會是最新的。

檢視您的試用偵測工作 (主控台)

請使用下列程序來檢視您的試用偵測。

若要檢視您的試用偵測工作

1. 打開亞馬遜 Lookout for Vision 控制台 https://console.aws.amazon.com/lookoutvision/.
2. 選擇 Get started (開始使用)。
3. 在左側導覽窗格中，選擇 [試用偵測]。
4. 在試驗偵測頁面上，選擇試驗偵測工作以檢視其詳細資料。

範例程式碼和資料集

以下是您可以搭配 Amazon Lookout for Vision 使用的程式碼範例和資料集。

主題

範例程式碼

Amazon Lookout for Vision 的下列程式碼範例。

範例	描述
GitHub	用於訓練和託管亞馬遜觀察視覺模型的示例 Python 代碼。
Amazon Lookout for Vision 實驗室	可用來建立具有電路板範例影像的模型的 Python 筆記本。
示例代碼	在 Amazon Lookout in Vision 文檔中使用的 Python 示例。
範例程式碼	在 Amazon Lookout in Vision 文檔中使用的 Java 例子。

範例資料集

以下是您可以搭配 Amazon Lookout for Vision 使用的範例資料集。

主題

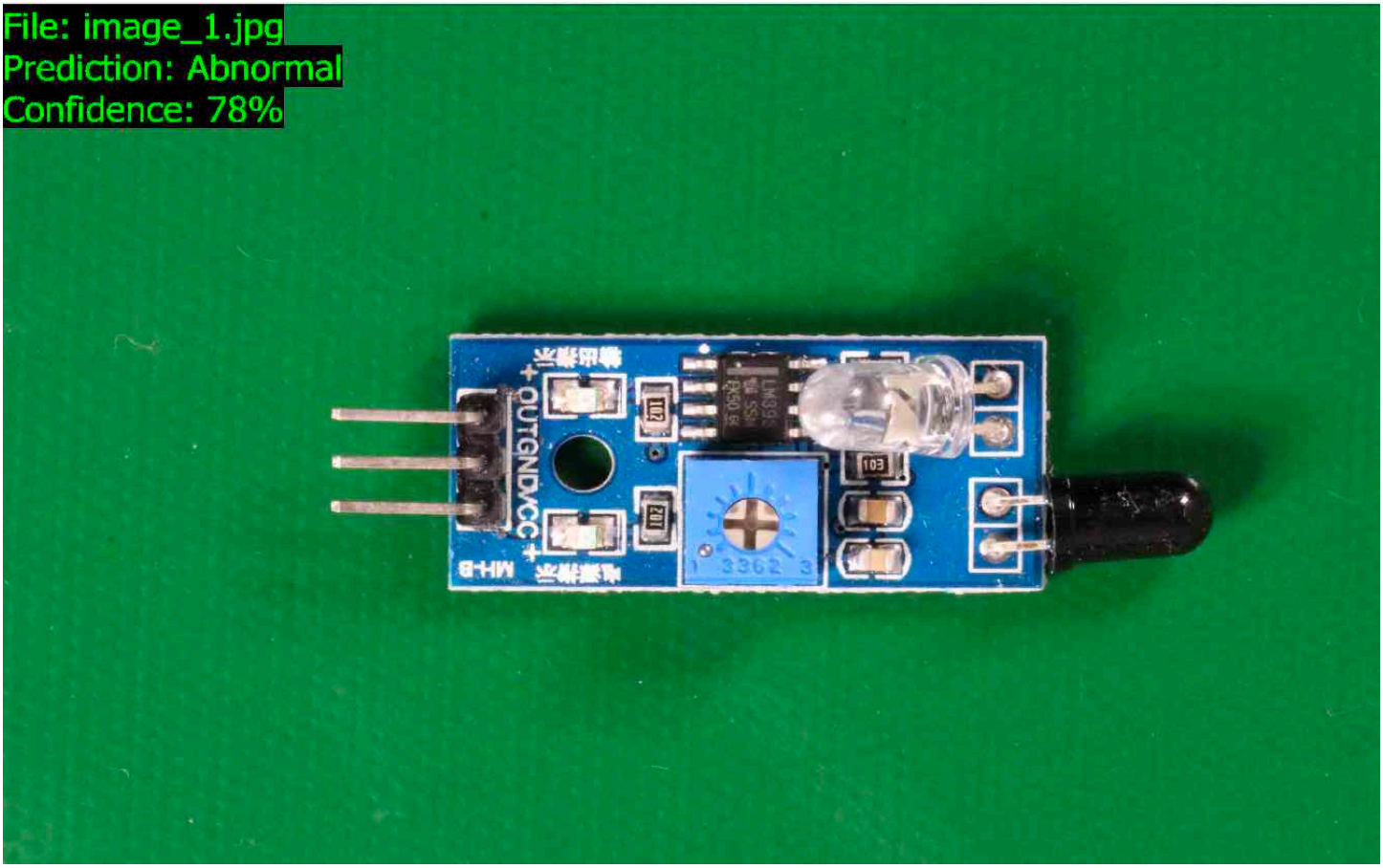
影像分割資料集

Amazon Lookout for Vision提供損壞的 Cookie 資料集，您可以用來建立影像分割模型。

如需建立影像分段模型的其他資料集，請參閱在邊緣使用 Amazon Lookout for Vision 識別異常的位置，而不使用 GPU。

影像分類資料集

Amazon Lookout for Vision 版提供電路板範例影像，您可以使用這些影像來建立影像分類模型。



您可以從 https://github.com/aws-samples/amazon-lookout-for-vision GitHub 存儲庫中複製圖像。圖像位於文件`circuitboard`夾中。

該`circuitboard`文件夾具有以下文件夾。

- `train`— 您可以在訓練資料集中使用的影像。
- `test`— 您可以在測試資料集中使用的影像。

- `extra_images`— 您可以用來執行試用偵測或透過DetectAnomalies操作試用訓練過的模型的影像。

每個`train`和`test`資料夾都有一個名為`normal` (包含正常影像) 的子資料夾，以及名為`anomaly` (包含具有異常影像) 的子資料夾。

> **Note**
>
> 稍後，當您使用主控台建立資料集時，Amazon Lookout 視覺版可以使用資料夾名稱 (`normal`和`anomaly`) 自動為影像加上標籤。如需詳細資訊，請參閱the section called “Amazon S3 儲存貯體”。

若要準備資料集影像

1. 將 https://github.com/aws-samples/ 存amazon-lookout-for-vision儲庫克隆到您的計算機。如需詳細資訊，請參閱複寫儲存庫。
2. 建立 Amazon S3 儲存貯體。如需詳細資訊，請參閱如何建立 S3 儲存貯體？ 。
3. 在命令提示字元中，輸入以下命令，將資料集映像從電腦複製到 Amazon S3 儲存貯體。

```
aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
circuitboard
```

上傳圖像後，您可以創建一個模型。您可以從先前將電路板映像上傳到的 Amazon S3 位置新增映像來自動分類映像。請記住，您需要為每次成功訓練模型以及模型執行 (託管) 的時間量付費。

建立分類模型

1. 做創建項目（控制台）。
2. 做使用存放在 Amazon S3 儲存貯體中的映像建立資料集。
   - 對於步驟 6，請選擇 [個別訓練和測試資料集] 索引標籤。
   - 對於步驟 8a，請為您在中上傳的訓練映像輸入 S3 URI 準備資料集映像。例如：`s3://your-bucket/circuitboard/train`。在步驟 8b 中，輸入測試資料集的 S3 URI。例如：`s3://your-bucket/circuitboard/test`。
   - 請務必執行步驟 9。

3. 做訓練模型（控制台）。
4. 做啟動您的模型（控制台）。
5. 做偵測影像中的異常。您可以使用test_images資料夾中的影像。
6. 當您完成模型時，請執行停止您的模型（控制台）。

Amazon Lookout for Vision

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計劃的一部分。若要了解適用於 Amazon Lookout for Vision 的合規計畫，請參閱合規計畫。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Lookout for Vision。下列主題將示範如何設定 Lookout for Vision。您也會了解如何使用其他 AWS 服務來協助監控並保護 Lookout for Vision。

主題

Amazon 尋找視覺中的數據保護

AWS共同責任模型適用於 Amazon Lookout for Vision 中的資料保護。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。您也必須負責您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。

- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 使用進階的受管安全服務（例如 Amazon Macie），協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如 Name (名稱) 欄位。這包括當您使用控制台，API 或 AWS SDK AWS 服務 使用 Lookout for Vision 或其他工作時。AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

資料加密

下列資訊說明 Amazon Lookout for Vision 使用資料加密來保護您的資料的地方。

靜態加密

映像

為了訓練您的模型，Amazon Lookout 視覺會製作一份來源訓練和測試影像的副本。複製的映像會在 Amazon Simple Storage Service (S3) 中使用伺服器端加密和您提供AWS 擁有的金鑰的金鑰進行靜態加密。金鑰是使用 AWS Key Management Service (SSE-KMS) 存放的。您的來源圖像不受影響。如需詳細資訊，請參閱培訓您的模型。

Amazon Lookout for Vision 模型

根據預設，訓練過的模型和資訊清單檔案會使用伺服器端加密，並使用存 Amazon S3 在 AWS Key Management Service (SSE-KMS) 中的 KMS 金鑰加密。Lookout for Vision 使用AWS 擁有的金鑰. 如需詳細資訊，請參閱使用伺服器端加密保護資料。訓練結果會寫入至`output_bucket`輸入參數中指定的值區`CreateModel`。訓練結果會使用儲存貯體 (`output_bucket`) 所設定的加密加密組態。

Amazon Lookout for Vision 控制台桶

亞馬遜 Lookout for Vision 察控制台創建一個 Amazon S3 存儲桶（控制台存儲桶），您可以用來管理您的項目。主控台儲存貯體使用預設的 Amazon S3 加密進行加密。如需詳細資訊，請參閱 Amazon

Simple Storage Service 的 S3 儲存貯體預設加密。如果您使用自有 KMS 金鑰，請在建立主控台儲存貯體之後進行設定。如需詳細資訊，請參閱使用伺服器端加密保護資料。Amazon 視覺觀察阻止對控制台存儲桶的公共訪問。

傳輸中加密

Amazon Lookout for Vision API 端點僅支援透過 HTTPS 的安全連線。所有通訊都是使用 Transport Layer Security (TLS) 加密。

金鑰管理

您可以使用 AWS Key Management Service (KMS) (KMS) 來管理存放在 Amazon S3 儲存貯體中的輸入映像的加密。如需詳細資訊，請參閱步驟 5：(選用) 使用您自己的 AWS Key Management Service 金鑰。

根據預設，您的映像會使用 AWS 擁有和管理的金鑰加密。您也可以選擇使用自己的 AWS Key Management Service (KMS) (KMS) 金鑰。如需更多詳細資訊，請參閱 AWS Key Management Service 概念。

網際網路流量隱私權

Amazon Lookout for Vision 的 Amazon Virtual Private Cloud (Amazon VPC) 端點是 VPC 中的一個邏輯實體，僅允許連接到 Amazon Lookout for Vision。Amazon VPC 會將請求路由至 Amazon Lookout for Vision，並將回應路由傳回 VPC。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。如需將 Amazon VPC 端點與 Amazon 觀察視覺搭配使用的相關資訊，請參閱使用介面端點 (AWS PrivateLink) 存取 Amazon Lookout for Vision ()。

Amazon Lookout for Vision 的身份和訪問管理

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使 Lookout for Vision 資源。您可以使用 IAM AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon 視覺瞭望如何與 IAM 合作

- Amazon 瞭望視覺基於身份的政策示例
- AWS亞馬遜視覺瞭望的受管政策
- Amazon 觀察視覺身分和存取的疑難排解

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在「瞭望視覺」中所做的工作。

服務使用者 — 如果您使 Lookout for Vision 服務來完成工作，則管理員會為您提供所需的認證和權限。當您使用更多瞭解視覺功能來完成工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在「Lookout for Vision」中存取某項功能，請參閱Amazon 觀察視覺身分和存取的疑難排解。

服務管理員 — 如果您負責公司 Lookout for Vision 資源，您可能擁有完整的存取權限。決定您的服務使用者應該存取哪些 Lookout for Vision 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何使用 IAM 搭配 Lookout for Vision，請參閱Amazon 視覺瞭望如何與 IAM 合作。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策以管理 Lookout for Vision 存取權限的詳細資訊。若要檢視您可以在 IAM 中使用的 Lookout for Vision 身分型政策範例，請參閱。Amazon 瞭望視覺基於身份的政策示例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體

的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源

的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon 視覺瞭望如何與 IAM 合作

在您使用 IAM 管理 Lookout for Vision 的存取權限之前，請先了解哪些 IAM 功能可與 Lookout for Vision 測搭配使用。

您可以與 Amazon 觀景視覺一起使用的 IAM 功能

IAM 功能	Lookout for Vision 支持
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	否
服務連結角色	否

若要深入瞭解瞭望視覺和其他 AWS 服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的可與 IAM 搭配使用的AWS 服務。

以身分識別為基礎的「Lookout for Vision」政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

以身分識別為基礎的政策範例，以 Lookout for Vision

若要檢視 Lookout for Vision 基礎的身分識別原則的範例，請參閱。Amazon 瞭望視覺基於身份的政策示例

《瞭望遠景》內的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

關 Lookout for Vision 的政策行動

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看檢 Lookout for Vision 動作清單，請參閱服務授權參考資料中的 Amazon Lookout for Vision 定義的動作。

「Lookout for Vision」中的原則動作會在動作之前使用下列前置詞：

```
lookoutvision
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

若要檢視 Lookout for Vision 基礎的身分識別原則的範例，請參閱。Amazon 瞭望視覺基於身份的政策示例

守望 Lookout for Vision 的政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

若要查看檢 Lookout for Vision 源類型及其 ARN 的清單，請參閱服務授權參考資料中的 Amazon Lookout 視覺定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon 瞭望視覺定義的動作。

若要檢視 Lookout for Vision 基礎的身分識別原則的範例，請參閱。Amazon 瞭望視覺基於身份的政策示例

守 Lookout for Vision 的政策條件索引鍵

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看檢 Lookout for Vision 條件金鑰清單，請參閱服務授權參考中的 Amazon Lookout for Vision 的條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon Lookout 視覺定義的動作。

若要檢視 Lookout for Vision 基礎的身分識別原則的範例，請參閱。Amazon 瞭望視覺基於身份的政策示例

在 Lookout for Vision ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

ABAC 與 Lookout for Vision

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱 IAM 使用者指南中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

使用臨時憑據與 Lookout for Vision

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Lookout for Vision 的轉發訪問會話

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

「Lookout for Vision」的服務角色

支援服務角色	否

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。

> **Warning**
>
> 變更服務角色的權限可能會中 Lookout for Vision 功能。只有在「Lookout for Vision」提供指引時，才編輯服務角色。

以服務連結的角色 Lookout for Vision

支援服務連結角色。	否

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon 瞭望視覺基於身份的政策示例

根據預設，使用者和角色沒有建立或修改 Lookout for Vision 源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

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》中的建立 IAM 政策。

如需瞭解 Vision 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon Lookout for Vision 動作、資源和條件金鑰。

主題

政策最佳實務

以身分識別為基礎的原則會決定使用者是否可以建立、存取或刪除您帳戶中的 Lookout Vision 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access

Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。

- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

訪問單個 Amazon Lookout for Vision 項目

在此範例中，您想要授予 AWS 帳戶中的使用者存取您其中一個 Amazon Lookout for Vision 專案的存取權。

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

標籤型政策範例

以標籤為基礎的原則是 JSON 政策文件，可指定主參與者可對標記資源執行的動作。

使用標籤存取資源

此範例政策授予 AWS 帳戶中的使用者或角色許可，可將`DetectAnomalies`操作與任何標記為金鑰`stage`和值的模型搭配使用`production`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
    ]
}
```

使用標籤拒絕訪問特定的 Amazon Lookout for Vision 操作

此範例政策拒絕 AWS 帳戶中使用者或角色的許可，以呼叫標有金鑰`stage`和值`production`之任何模型的`DeleteModel`或`StopModel`操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
    ]
}
```

AWS亞馬遜視覺瞭望的受管政策

AWS 受管政策是由 AWS 建立和管理的獨立政策。AWS 受管政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授與您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

AWS 受管政策： AmazonLookoutVisionReadOnlyAccess

使用AmazonLookoutVisionReadOnlyAccess政策允許使用者透過下列 Amazon 觀景視覺動作 (SDK 操作) 對 Amazon Lookout 視覺 (及其相依性) 進行唯讀存取。例如，您可以使用DescribeModel以取得有關現有模型的資訊。

- DescribeDataset
- DescribeModel
- DescribeModelPackagingJob
- DescribeProject
- ListDatasetEntries
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

若要呼叫唯讀動作，使用者不需要 Amazon S3 儲存貯體許可。不過，作業回應可能包含 Amazon S3 儲存貯體的參考資料。例如，source-ref在響應中的條目ListDatasetEntries是亞馬遜 S3 存儲桶中映像的引用。如果使用者需要存取參考的儲存貯體，請新增 Amazon S3 儲存貯體許可。例如，使用者可能想要下載由source-ref欄位。如需詳細資訊，請參閱授予 Amazon S3 存儲桶許可。

您可將 AmazonLookoutVisionReadOnlyAccess 政策連接到 IAM 身分。

許可詳細資訊

此政策包含以下許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策：AmazonLookoutVisionFullAccess

使用AmazonLookoutVisionFullAccess政策允許使用者透過亞馬遜觀景視覺操作 (SDK 操作) 完全存取亞馬遜觀景視覺 (及其相依性)。例如，您可以訓練模型，而不必使用 Amazon 視覺觀察主控台。如需詳細資訊，請參閱動作。

若要建立資料集 (CreateDataset) 或建立模型 (CreateModel)，您的使用者必須擁有存放資料集映像檔的 Amazon S3 儲存貯體 (Amazon) 的完整存取權限SageMaker地面真相清單文件, 和培訓輸出. 如需詳細資訊，請參閱步驟 2：設定權限。

您也可以授予權限亞馬遜瞭望視覺 SDK 動作，使用AmazonLookoutVisionConsoleFullAccess政策。

您可將 AmazonLookoutVisionFullAccess 政策連接到 IAM 身分。

許可詳細資訊

此政策包含以下許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策： AmazonLookoutVisionConsoleFullAccess

使用AmazonLookoutVisionFullAccess允許使用者完全存取 Amazon Lookout 視覺主控台、動作 (SDK 操作) 以及服務具有的任何相依性的政策。如需詳細資訊，請參閱Amazon Lookout for Vision。

該LookoutVisionConsoleFullAccess政策包括對您的亞馬遜觀察視覺控制台存儲桶的許可。如需主控台值區的相關資訊，請參閱步驟 3：建立主控台儲存貯體。若要存放資料集、影像和AmazonSageMaker地面真相資訊清單檔案位於不同的 Amazon S3 儲存貯體中，您的使用者需要其他許可。如需詳細資訊，請參閱the section called “設置 Amazon S3 存儲桶許可”。

您可將 AmazonLookoutVisionConsoleFullAccess 政策連接到 IAM 身分。

許可群組

根據提供的權限集，將此原則分組為陳述式：

- LookoutVisionFullAccess— 允許訪問執行所有觀察視覺操作.
- LookoutVisionConsoleS3BucketSearchAccess— 允許列出呼叫者擁有的所有 Amazon S3 儲存貯體。瞭解 Vision 會使用此動作來識別 AWS 區域特定的 Vision 控制台值區 (如果來電者帳戶中存在)。
- LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions— 允許建立和設定符合瞭望視覺主控台儲存貯體名稱模式的 Amazon S3 儲存貯體。Lookout Vision 會使用這些動作，在找不到特定區域的 Vision 控制台值區時，建立和設定區域特定的 Lookout the 控制台值區。

- LookoutVisionConsoleS3BucketAccess— 允許在符合瞭望視覺主控台儲存貯體名稱模式的儲存貯體上執行相依的 Amazon S3 動作。瞭解視覺用途s3:ListBucket在從 Amazon S3 儲存貯體建立資料集時以及開始試用偵測任務時搜尋影像物件。瞭解視覺用途s3:GetBucketLocation和s3:GetBucketVersioning以驗證存儲桶AWS地區、擁有者和組態作為下列項目的一部分：
  - 建立資料集
  - 訓練模型
  - 開始試驗偵測工作
  - 執行試驗偵測回饋

  LookoutVisionConsoleS3ObjectAccess— 允許讀取和寫入儲存貯體內符合瞭望視覺主控台儲存貯體名稱模式的儲存貯體內的 Amazon S3 物件。Lookout the Vision 會使用這些動作在主控台圖庫檢視中顯示影像，並上傳新影像以供資料集使用。此外，這些權限允許 Lookout in Vision 在建立資料集、訓練模型、開始試用偵測工作，以及執行試驗偵測回饋時寫出中繼資料。
- LookoutVisionConsoleDatasetLabelingToolsAccess— 允許依賴亞馬遜SageMaker GroundTruth標示動作。瞭解視覺使用這些動作來掃描 S3 儲存貯體中的映像、建立GroundTruth清單文件，並用驗證標籤註釋試驗檢測任務結果。
- LookoutVisionConsoleDashboardAccess-允許閱讀亞馬遜CloudWatch度量標準。Lookout Vision 會使用這些動作來填入儀表板圖形和模組偵測到的統計資料。
- LookoutVisionConsoleTagSelectorAccess— 允許讀取帳戶特定的標籤鍵和標籤值建議。瞭解 Vision 會使用這些權限來提供標籤索引鍵和標籤值的建議管理標籤主控台頁面。
- LookoutVisionConsoleKmsKeySelectorAccess-允許上市AWS Key Management Service(KMS) 金鑰和別名。亞馬遜觀景使用此許可填充建議的 KMS 密鑰标签針對支援客戶管理的 KMS 金鑰進行加密的特定「尋找視覺」動作上的選取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketAcl",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDatasetLabelingToolsAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",
                "groundtruthlabeling:AssociatePatchToManifestJob",
                "groundtruthlabeling:DescribeConsoleJob"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleKmsKeySelectorAccess",
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策： AmazonLookoutVisionConsoleReadOnly存取

使用AmazonLookoutVisionConsoleReadOnlyAccess政策允許使用者以唯讀方式存取 Amazon Lookout 視覺主控台、動作 (SDK 操作)，以及服務具有的任何相依性。

該AmazonLookoutVisionConsoleReadOnlyAccess政策包括亞馬遜觀察視覺控制台存儲桶的 Amazon S3 許可。如果您的數據集圖像或亞馬遜SageMaker地面真相資訊清單檔案位於不同的

Amazon S3 儲存貯體中，您的使用者需要其他許可。如需詳細資訊，請參閱the section called “設置 Amazon S3 存儲桶許可”。

您可將 AmazonLookoutVisionConsoleReadOnlyAccess 政策連接到 IAM 身分。

許可群組

根據提供的權限集，將此原則分組為陳述式：

- LookoutVisionReadOnlyAccess-允許訪問以執行只讀觀察視覺操作。
- LookoutVisionConsoleS3BucketSearchAccess— 允許列出呼叫者擁有的所有 S3 儲存貯體。瞭解 Vision 會使用此動作來識別 AWS 區域特定的 Vision 控制台值區 (如果來電者帳戶中有存在)。
- LookoutVisionConsoleS3ObjectReadAccess— 允許在瞭解視覺主控台儲存貯體中讀取 Amazon S3 物件和 Amazon S3 物件版本。Lookout Vision 會使用這些動作在資料集、模型和試驗偵測中顯示影像。
- LookoutVisionConsoleDashboardAccess— 允許閱讀亞馬遜CloudWatch度量標準。Lookout the Vision 使用這些動作來填入偵測到的儀表板圖形和異常的統計資料。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeTrialDetection",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListTrialDetections",
                "lookoutvision:ListModelPackagingJobs"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

尋找願景更新AWS受管理政策

檢視有關更新的詳細資訊AWS由於此服務開始跟踪這些更改，因此監視視覺的管理策略。如需有關此頁面變更的自動警示，請訂閱「瞭解 Vision 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
新增模型包裝操作	亞馬遜視覺瞭望添加了以下模型包裝操作AmazonLookoutVisionFullAccess和AmazonLookoutVisionConsoleFullAccess政策： • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob 亞馬遜視覺瞭望添加了以下模型包裝操作AmazonLookoutVisionReadOnlyAccess和AmazonLookoutVisionConsoleReadOnlyAccess政策： • DescribeModelPackagingJob • ListModelPackagingJobs	二○二一年十二月七
添加了新策略	亞馬遜視覺瞭望添加了以下政策。 • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	五月十一日
尋找視覺開始跟踪更改	亞馬遜視覺瞭望開始跟踪其變化AWS受管理的策略。	二○二一年三月一日

Amazon 觀察視覺身分和存取的疑難排解

使用下列資訊可協助您診斷並修正使用 Lookout Vision 和 IAM 時可能會遇到的常見問題。

主題

我沒有授權在 Lookout for Vision 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 lookoutvision:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 lookoutvision:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策，以允許您將角色傳遞給 Lookout for Vision。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 Lookout Vision 中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪問我 AWS 帳戶 的 Lookout for Vision 源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解 Lookout for Vision 是否支援這些功能，請參閱Amazon 視覺瞭望如何與 IAM 合作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何向第三方提供對資源的存取權 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

Amazon Lookout for Vision 合規驗證

協力廠商稽核員會評估 Amazon Lookout for Vision 的安全性和合規性，這是多項 AWS 合規計畫的一部分。Amazon Lookout for Vision 察符合通用數據保護條例（GDPR）。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於您資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準 AWS 服務 與技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您滿足特定合規性架構所要求的入侵偵測需求，例如 PCI DSS 等各種合規性需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon LoLookout for Vision

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

亞馬遜視覺瞭望中的基礎設施安全

作為一項受管服務，亞馬遜觀察視覺受到AWS全球網絡安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可以使用AWS已發佈的 API 呼叫，透過網路存取瞭望視覺。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密 (PFS) 的密碼套件，例如 DHE (Ephemeral Diffie-Hellman) 或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon Lookout for Vision

監控是維護 Amazon Lookout for Vision 和您其他 AWS 解決方案可靠性、可用性和效能的重要環節。AWS 提供下列監控工具以監督 Lookout for Vision、在發現錯誤時回報，並適時自動採取動作：

- Amazon 會即時 CloudWatch監控您的AWS資源，以及您AWS在上執行的應用程式。您可以收集和追蹤指標、建立自訂儀表板，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以使用 CloudWatch 追蹤 CPU 使用量或其他 Amazon EC2 執行個體指標，並在需要時自動啟動新的執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon EC2 執行個體、或其他來源監控、存放和存取來自 Amazon EC2 執行個體 CloudTrail、或其他來源的 CloudWatch 日誌檔案。 CloudWatch 日誌可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- Amazon EventBridge 可用來自動化您的AWS服務，並自動回應系統事件，例如應用程式可用性問題或資源變動。AWS服務的事件，會以接近即時 EventBridge 的速度傳送到。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南。
- AWS CloudTrail 擷取您 AWS 帳戶發出或代表發出的 API 呼叫和相關事件，並傳送記錄檔案至您指定的 Simple Storage Service (Amazon S3) 儲存貯體。您可以找出哪些使用者和帳戶呼叫 AWS、發出呼叫的來源 IP 地址，以及呼叫的發生時間。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南。

Amazon Lookout for Vision CloudWatch

您可以使用監控 Lookout for Vision CloudWatch，這會收集原始數字並將其處理成可讀取、近乎即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

LoLookout for Vision 服務會在`AWS/LookoutVision`命名空間中報告下列指標。

指標	描述
`DetectedAnomalyCount`	在專案中偵測到的異常數 有效統計數字：`Sum`,`Average`

指標	描述
	單位：計數
ProcessedImageCount	透過異常偵測執行的影像總數 有效統計數字：Sum,Average 單位：計數
InvalidImageCount	無效且無法傳回結果的影像數 有效統計數字：Sum,Average 單位：計數
SuccessfulRequestCount	成功的 API 呼叫數量 有效統計數字：Sum,Average 單位：計數
ErrorCount	應用程式介面錯誤的數目 有效統計數字：Sum,Average 單位：計數
ThrottledCount	由於節流造成的 API 錯誤數 有效統計數字：Sum,Average 單位：計數
Time	觀察視覺計算異常偵測的時間 (以毫秒為單位) 有效統計數字：Data Samples,Average 單位：Average統計資料的毫秒和統計資Data Samples料的計數

指標	描述
MinInferenceUnits	StartModel 請求期間指定的推論單元數目下限。 有效的統計資訊：Average 單位：計數
MaxInferenceUnits	StartModel 請求期間指定的推論單元數目上限。 有效的統計資訊：Average 單位：計數
DesiredInferenceUnits	Lookout for Vision 正在擴展或縮小的推論單元的數量。 有效的統計資訊：Average 單位：計數
InServiceInferenceUnits	模型正在使用的推論單位數。 有效的統計資訊：Average 建議您使用「平均值」統計資料來取得使用多少執行處理的 1 分鐘平均值。 單位：計數

以下維度支援 Lookout for Vision 指標。

維度	描述
ProjectName	您可以依專案分割指標，以查看哪些專案有問題或需要更新。
ModelVersion	您可以依模型版本分割量度，以查看哪些模型有問題或需要更新。

記錄 Lookout for Vision API 調用AWS CloudTrail

Amazon Lookout for Vision out in VisionAWS CloudTrailAWS CloudTrail 會將的所有 API 呼叫擷 Lookout for Vision 事件。擷取的呼叫包括來自 LoLookout for Vision 如果您建立追蹤記錄，就可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Lookout for Vision 即使未設定追蹤記錄，您依然可以在 CloudTrail 主控台中檢視最新的事件。您可以利用所 CloudTrail收集的資訊來判斷向 Lookout for Vision 提出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

Lookout in Vision CloudTrail

CloudTrail 當您建立AWS帳戶時，系統會在您的帳戶中啟用。當活動發生時發生活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他AWS服務事件記錄到事件歷史記錄中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

若要持續記錄您的AWS帳戶中的事件，包括 Lookout for Vision 線索能 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還能設定其他AWS服務，以進一步分析和處理 CloudTrail 日誌中收集的事件資料，並採取相應動作。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

所有 Lookout for Vision 操作記錄， CloudTrail 並記錄在 Lookout for Vision API 參考文檔。例如，呼叫`DetectAnomalies`和`StartModel`動作會`CreateProject`在 CloudTrail 記錄檔中產生項目。

如果您監控 Amazon Lookout for Vision

- 展望視野：StartTriallDetection
- 展望視野：ListTriallDetection
- 展望視野：DescribeTrialDetection

亞馬遜 Lookout for Vision 使用這些特殊呼叫來支持與試驗檢測相關的各種操作。如需詳細資訊，請參閱使用試驗偵測任務來驗證您的模型。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 AWS Identity and Access Management 使用者登入資料提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

了解 Lookout for Vision 日誌文件條目

追蹤是一種組態，能讓事件以日誌檔案的形式交付至您指定的 Amazon S3 儲存貯體。 CloudTrail 日誌檔案包含一個或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。 CloudTrail 日誌檔案並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示該`CreateDataset`動作的 CloudTrail 日誌項目。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAYN4CJAYDEXAMPLE:user",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/MyUser",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYN4CJAYEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAYN4CJAYDCGEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-20T13:15:09Z"
```

```
      }
    }
  },
  "eventTime": "2020-11-20T13:15:43Z",
  "eventSource": "lookoutvision.amazonaws.com",
  "eventName": "CreateDataset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "128.0.0.1",
  "userAgent": "aws-cli/3",
  "requestParameters": {
    "projectName": "P1",
    "datasetType": "train",
    "datasetSource": {
      "groundTruthManifest": {
        "s3Object": {
          "bucket": "myuser-bucketname",
          "key": "training.manifest"
        }
      }
    },
    "clientToken": "EXAMPLE-0526-47dd-a5d3-2ca975820a34"
  },
  "responseElements": {
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "EXAMPLE-15e1-4bc9-be38-cda2537c75bf",
  "eventID": "EXAMPLE-c5e7-43e0-8449-8d9b87e15acb",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

使用記錄 Amazon Lookout for Vision 資源AWS CloudFormation

Amazon Lookout for Vision 的模型和設定AWS CloudFormation，以減少建立和管理AWS資源和基礎設施的時間。您建立一個描述所有所需之AWS資源的範本，描述所有所AWS CloudFormation需之資源，而負責為您佈建與設定這些資源。

您可以使用佈AWS CloudFormation建和設定視覺專案的亞馬遜瞭望台。

當您使用時AWS CloudFormation，您可以重複使用您的範本，重複、一致的設定您 Lookout in Vision 專案。只需描述一次您的專案，然後在多個 AWS 帳戶和區域內重複佈建相同專案。

Lookout for VisionAWS CloudFormation

若要佈建和設定 Lookout for Vision 和相關服務的相關服務，您必須了解AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的什麼是 AWS CloudFormation Designer？。

如需瞭解視覺專案的參考資訊，包括 JSON 和 YAML 範本的範例，請參閱LookoutVision 資源類型參考。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- 《AWS CloudFormation 使用者指南》https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
- AWS CloudFormation API 參考
- 《AWS CloudFormation 命令列介面使用者指南》https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html

使用介面端點 (AWS PrivateLink) 存取 Amazon Lookout for Vision ()

您可以使AWS PrivateLink用在 VPC 與 Amazon Lookout for Vision 之間建立私有連線。您可以如在 VPC 中一樣存取 Lookout for Vision，無需使用網際網路閘道、NAT 裝置、VPN 連線或AWS Direct Connect連線。VPC 中的執行個體無需公有 IP 地址，即可存取 Lookout for Vision。

您可以建立由 AWS PrivateLink 提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 Lookout for Vision 之流量的進入點。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的透過 AWS PrivateLink 存取 AWS 服務。

檢視視覺 VPC 端點的考量

在設定 Lookout Vision 的介面端點之前，請先檢閱AWS PrivateLink指南中的考量事項。

Lookout for Vision 援透過端點介面端點呼叫其所有 API 動作。

Lookout for Vision 不支援 VPC 端點政策。依預設，允許透過端點介面端點介面 Lookout for Vision 進行完整存取。或者，您也可以將安全群組與端點網路介面相關聯，以控制透過端點傳輸至 Lookout for Vision 的流量。

為 Lookout for Vision 建立介面 VPC 端點

您可使用 Amazon VPC 主控台或 ()) Lookout for VisionAWS Command Line Interface (AWS CLI) 建立介面端點。如需詳細資訊，請參閱 《AWS PrivateLink 指南》中的建立介面端點。

使用以下服務名 Lookout for Vision 建立介面端點：

```
com.amazonaws.region.lookoutvision
```

如果您為介面端點啟用私有 DNS，您可以使用其預設的區域 DNS 名稱向 Lookout for Vision 發出 API 要求。例如：`lookoutvision.us-east-1.amazonaws.com`。

為 Lookout for Vision 建立 VPC 端點政策

端點政策為 IAM 資源，您可將其連接至介面端點。預設端點政策可允許透過端點介面端點完整存取 Vision。若要控制 VPC Lookout for Vision 存取權限，請將自訂端點政策連接至介面端點。

端點政策會指定以下資訊：

- 可執行動作 (AWS 帳戶IAM 使用者和 IAM 角色) 的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制對服務的存取。

範例：Lookout for Vision 動作的 VPC 端點政策

以下是 Lookout for Vision 的自訂端點政策的範例。當您將此原則附加到介面端點時，它會指定允許所有具有 VPC 介面端點存取權限的使用者呼叫與專案`myModel`關聯的 Lookout for Vision 模型的`DetectAnomalies` API 作業`myProject`。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
      }
   ]
}
```

亞馬遜視覺瞭望中的配額

下表說明亞馬遜視覺瞭望中的當前配額。如需可變更配額的相關資訊，請參閱AWS 服務配額。

模型配額

下列配額適用於模型的測試、訓練和功能。

資源	配額
支援的檔案格式	PNG 和 JPEG 圖像格式
Amazon S3 儲存貯體中影像檔案的最小影像尺寸	64 像素 x 64 像素
Amazon S3 儲存貯體中影像檔案的最大影像尺寸	最大值為 4096 個像素 x 4096 個像素。較小的尺寸可以更快地上傳。
項目中使用的圖像文件的不同圖像尺寸	資料集中的所有影像都必須具有相同的維度
Amazon S3 儲存貯體中映像檔的最大檔案大小	8 MB
缺少標籤	在訓練之前，圖像必須標記為正常或異常。訓練期間會忽略沒有標籤的影像。
訓練資料集中標示為正常的影像數目下限	10 適用於具有獨立訓練與測試資料集的專案。20 個適用於單一資料集的專案。
訓練資料集中標示為異常的影像數目下限	0 適用於具有不同訓練和測試資料集的專案。具有單一資料集的專案為 10。
分類訓練資料集中的影像數目上限	16,000
分類測試資料集中的影像數目上限	4,000
測試資料集中標示為正常的影像數量下限	10
測試資料集中標示為異常的影像數量下限	10

資源	配額
異常本地化訓練資料集中的影像數目上限	8000
異常本地化測試資料集中的影像數目上限	800
試驗偵測資料集中的影像數目上限	2,000
資料集清單檔案大小上限	1 GB
模型中訓練資料集的最大數量	1
最長訓練時間	24 小時
最長測試時間	24 小時
專案中異常標籤的最大數量	100
遮色片影像上的異常標籤數目上限	20
異常標籤的影像數目下限。若要計數，影像必須只包含一種異常標籤類型。	20 適用於單一資料集專案。專案中的每個資料集有 10 個，其中包含不同的訓練和測試資料集。

Amazon Lookout for Vision 的文件歷史記錄

下表說明 Amazon Lookout of Desion 指南每個版本的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

- 最新文件更新：2023 年 2 月 20 日

變更	描述	日期
添加示例 Lambda 函數	示範如何使用AWS Lambda 函數尋找異常的範例。如需詳細資訊，請參閱使用 AWS Lambda 函數尋找異常情況。	2023 年 2 月 20 日
更新適用於AWS WAF	更新了指南以符合 IAM 最佳實務。如需詳細資訊，請參閱 IAM 中的安全最佳實務。	2023 年 2 月 8 日
新增資料集匯出範例	已新增 Python 範例，說明如何使用該`ListDatasetEntries` 作業從 Amazon Lookout for Vision 專案匯出資料集。如需詳細資訊，請參閱從專案匯出資料集 (SDK)。	2022 年 12 月 2 日
更新了開始使用主題	已更新開始使用說明如何使用範例資料集建立影像分割模型。如需詳細資訊，請參閱 Amazon LoLookout for Vision 入門。	2022 年 10 月 20 日
已新增疑難排解	新增模型訓練疑難排解主題。如需詳細資訊，請參閱疑難排解模型訓練。	2022 年 10 月 17 日

已新增有關使用 Amazon SageMaker Ground Truth 工作的主題	您可以使用 Amazon SageMaker Ground Truth 任務為分類和圖像細分模型標記影像，而不是自行標記影像。如需詳細資訊，請參閱使用 Amazon SageMaker Ground Truth 工作。	2022 年 8 月 19 日
亞馬遜 Lookout for Vision 現在提供異常本地化。	您可以建立區段模型，尋找影像上存在不同類型異常 (例如刮痕、凹痕或撕裂) 的位置、異常標籤和異常大小。如需詳細資訊，請參閱執行訓練過的 Amazon Lookout the Vision 模型。	2022 年 8 月 16 日
亞馬遜 Lookout for Vision 現在可在邊緣設備上提供 CPU 推論。	現在可以部署亞馬遜 Lookout for Vision 模型，在僅使用 CPU 執行 Linux 的 x86 運算平台上本機執行推論，而不需要 GPU 加速器。如需詳細資訊，請參閱 CPU 加速器。	2022 年 8 月 16 日
亞馬遜 Lookout for Vision 察現在可以自動擴展推論單元。	為了幫助滿足需求激增，Amazon Lookout 視覺現在可以擴展模型使用的推論單元數量。如需詳細資訊，請參閱執行訓練有素的 Amazon Lookout for Vision 模型。	2022 年 8 月 16 日
添加了 Java 實例	亞馬遜 Lookout for Vision 開發人員指南現在包括 Java 示例。如需詳細資訊，請參閱開始使用AWS SDK。	2022 年 5 月 2 日

邊緣裝置的模型部署的一般可用性	由管理的邊緣裝置的模型部署現AWS IoT Greengrass Version 2已正式推出。如需詳細資訊，請參閱在邊緣裝置上使用 Amazon Lookout for Vision 模型。	2022 年 3 月 14 日
更新主控台值區資訊	已更新主控台值區內容的相關資訊，以及建立主控台值區的替代方法。如需詳細資訊，請參閱步驟 4：建立主控台值區。	2022 年 3 月 7 日
從 CSV 檔案建立資訊清單檔案	您現在可以使用從 CSV 檔案讀取分類資訊的指令碼，簡化資訊清單檔案的建立作業。如需詳細資訊，請參閱從 CSV 檔案建立資訊清單檔案。	2022 年 2 月 10 日
預覽邊緣裝置的模型部署版本	預覽版模型部署至由管理的邊緣裝置現AWS IoT Greengrass Version 2已推出。如需詳細資訊，請參閱在邊緣裝置上使用 Amazon Lookout for Vision 模型。	2021 年 12 月 7 日
添加了新的 Python 和 Java 2 示例	添加了 Python 和 Java 2 用於分析圖像的示例`DetectAnomalies` 。如需詳細資訊，請參閱偵測影像中的異常。	2021 年 9 月 7 日
已新增AWS受管理的原則。	亞馬遜 Lookout for Vision 添加了對AWS受管政策的支持。如需詳細資訊，請參閱 AWSAmazon Lookout for Vision 受管政策。	2021 年 5 月 11 日

已更新推論單位資訊。	已新增說明推論單位及其收費方式的資訊。如需詳細資訊，請參閱執行訓練有素的 Amazon Lookout for Vision 模型。	2021 年 3 月 15 日
Amazon Lookout of Vision 的正式版本。	Amazon Lookout for Vision 現已正式上市。Python 程式碼範例已更新，以處理非同步工作，例如訓練模型。	2021 年 2 月 17 日
添加了標記和AWS CloudFormation支持。	您現在可以標記亞馬遜 Lookout for Vision 模型和創建項目AWS CloudFormation。如需詳細資訊，請參閱使用 AWS 標記模型和為視覺專案建立 Amazon 瞭望台 CloudFormation。	2021 年 1 月 31 日
新功能和指南	這是初版 Lookout for Vision 發 Lookout for Vision 指南	2020 年 12 月 1 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。